Data Min Knowl Disc (2011) 22:393–418
DOI 10.1007/s10618-011-0211-4

Learning bidirectional asymmetric similarity for collaborative filtering via matrix factorization

Bin Cao · Qiang Yang · Jian-Tao Sun · Zheng Chen

Received: 1 May 2009 / Accepted: 22 January 2011 / Published online: 11 February 2011

Abstract Memory-based collaborative filtering (CF) aims at predicting the rating of a certain item for a particular user based on the previous ratings from similar users and/or similar items. Previous studies in finding similar users and items have several drawbacks. First, they are based on user-defined similarity measurements, such as Pearson Correlation Coefficient (PCC) or Vector Space Similarity (VSS), which are, for the most part, not adaptive and optimized for specific applications and data. Second, these similarity measures are restricted to symmetric ones such that the similarity between A and B is the same as that for B and A, although symmetry may not always hold in many real world applications. Third, they typically treat the similarity functions between users and functions between items separately. However, in reality, the similarities between users and between items are inter-related. In this paper, we propose a novel unified model for users and items, known as **S**imilarity **L**earning based **C**ollaborative **F**iltering (SLCF) , based on a novel *adaptive bidirectional asymmetric similarity measurement*. Our proposed model automatically learns asymmetric similarities between users and items at the same time through matrix factorization.

Responsible editor: Tao Li, Chris Ding and Fei Wang.

B. Cao (✉) · Q. Yang
The Hong Kong University of Science and Technology, Kowloon, Hong Kong
e-mail: caobin@cs.ust.hk

Q. Yang
e-mail: qyang@cs.ust.hk

J.-T. Sun · Z. Chen
Microsoft Research Asia, 49 Zhichun Road, Beijing, China
e-mail: jtsun@microsoft.com

Z. Chen
e-mail: zhengc@microsoft.com

Theoretical analysis shows that our model is a novel generalization of singular value decomposition (SVD). We show that, once the similarity relation is learned, it can be used flexibly in many ways for rating prediction. To take full advantage of the model, we propose several strategies to make the best use of the proposed similarity function for rating prediction. The similarity can be used either to improve the memory-based approaches or directly in a model based CF approaches. In addition, we also propose an online version of the rating prediction method to incorporate new users and new items. We evaluate SLCF using three benchmark datasets, including MovieLens, EachMovie and Netflix, through which we show that our methods can outperform many state-of-the-art baselines.

Keywords Collaborative filtering · Matrix factorization · Similarity learning

1 Introduction

With the explosive growth of the Web, personalized online services have become increasingly popular ranging from providing search results to giving product recommendations. Collaborative filtering (CF) aims at predicting the preference of items for online users based on the items previously rated by similar users. Examples of successful applications of CF include recommending products at Amazon.com,[1] movies by Netflix,[2] etc. Among the CF methods, memory-based methods are both simple and effective (Adomavicius and Tuzhilin 2005). They usually fall into two classes: user-based approaches (Breese et al. 1998; Herlocker et al. 1999) and item-based approaches (Sarwar et al. 2001; Deshpande and Karypis 2004). To predict a rating for an item for a given user, user-based methods find the ratings of similar users for prediction, while item-based methods rely on the ratings on similar items.

Despite much successes so far, memory-based methods suffer from several drawbacks. First, missing data is a major problem in CF, causing the so-called sparseness problem in CF (Xue et al. 2005). This is because there are usually millions of users and items in existence. However, a single user can only rate a relatively small number of items. Besides, the rating counts of items often follow long tail distributions (Park and Tuzhilin 2008), such that items in a long tail often have fewer ratings. The sparseness problem makes it difficult to find similar users or items accurately, especially for users and items who belong to the long tails. Second, in memory-based approaches, similar users and items are found by calculating a certain similarity measurement. Some well-known methods include Pearson Correlation Coefficient (PCC) (Resnick et al. 1994) and Vector Space Similarity (VSS) (Breese et al. 1998). However, being general methods, these measurements are often not *adaptive* to the application domains and particular data sets. These functions are often fixed in the sense that once given, they are not changeable; for example, we do not know when to switch from PCC to VSS in a single data set. To cope with these problems, many variations of similarity measurements, including

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

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

weighting approaches, combination measures and rating normalization methods have been developed (Herlocker et al. 2002). Although these measurements can capture the correlation between users or items to some extent, the similarity functions have fixed forms and are still not able to adapt to the underlying data (Herlocker et al. 2002). Third, these similarity measurements are restricted to be symmetric, so that the structure relation between users or items is a symmetric relation. The symmetry assumption can cause problems in many real world applications. For example, some users in a social network may have more impact than others. When they recommend items, the effect to their neighbors are much larger than the other way around. Finally, many previous studies in CF consider the similarities between users and items separately. However, similarities between users and items in reality are inter-dependent and can be used to reinforce each other. Therefore, it would be more appropriate if the similarities between users and items are jointly learned automatically. Our research is aimed at address all of the above shortcomings in a unified framework.

In this paper, we make two connected contributions. The first contribution is in similarity function learning. We propose a unified model to learn asymmetric similarities for items and users at the same time. Through novel reformulations of classic memory-based approaches, we first propose an one-directional similarity learning model which can learn either user-side similarity or item-side similarity. We further extend the one-directional learning to bi-directional similarity learning which learns user-side similarity and item-side similarity at the same time. We show that the similarity learning problems can be formulated as problems of matrix factorization with missing values. Our second contribution is a collection of algorithms that use the learned similarity functions for CF, in what we call similarity-learning collaborative filtering (SLCF). We consider two versions of SLCFs, the first improves the traditional memory-based approaches (M-SLCF), and second is based on matrix reconstruction (R-SLCF). We also propose an online version of the rating prediction method R-SLCF to allow new users and new items to be included in our model incrementally.

There are several advantages of our approaches. First, the similarity between two users is learned based on the global consistency of all ratings rather than their ratings only. Their similarity can be calculated even when they do not share any ratings. Therefore, the sparseness problem can be addressed satisfactorily. Second, the learned similarity function can be adaptive according to a number of real-world conditions. For example, the learned similarity would change if the degree of the sparseness of observed ratings changes. Third, we allow the similarity to be asymmetric. This can be achieved by allowing the similarity matrix to be asymmetric. Finally, we learn the similarity among users as well as items at the same time. In fact, the similarities between users and items can be regarded as being influenced by some latent factors. Theoretical analysis shows that our model corresponds to a novel generalization of the singular value decomposition (SVD) model, thus allowing a number of nice theoretical properties to be inherited from SVD research. We evaluate our model using three widely used benchmark datasets, including the MovieLens, EachMovie and Netflix data. We show that show that our methods outperform many of the well known baselines significantly on the tested benchmark datasets.

This paper is an extended version of our earlier work (Cao et al. 2008). In this paper, we make several major extensions. First, we extend the learning methods from symmetric similarity functions to the more general asymmetric functions. Then, we provide a more rigorous theoretical treatment of our learning algorithms and related computational properties. Finally, we introduce more extensive experimental results to validate our algorithms.

The remainder of the paper is organized as follows. In Sect. 2, we define the notations and the problems. In Sect. 3, we review some related works to our method and revisit the memory-based methods and SVD-based methods. Then, we introduce similarity learning based methods (SLCF) in Sect. 4. We discuss the relation between SLCF models and several matrix factorization models in Sect. 5. In Sect. 6, we discuss several rating prediction methods based on the learned similarity. We present our experiments in Sect. 7 and make conclusions in Sect. 8.

2 Notation and problem definition

In this section, we define the notations and give a formal statement of our problem.

- R: We denote the ground truth rating matrix by R, with rows representing users and columns representing items. We use r_{ui} to denote the rating value corresponding to a user u and an item i.
- X: We use X to denote the matrix with observed elements in R. Since only a part of the elements is known, X is sparse. We will use zero to denote the unobserved entries.
- Y: We use Y to denote the sparse matrix with elements to be predicted.
- I_X: We use I_X to denote the indicator matrix of sparse matrix X, whose element $(I_X)_{ij}$ is 1 if the X_{ij} is observed and 0 otherwise.

Both X and Y are subsets of rating matrix R. From the above definitions, we have $X = I_X \odot R$ and $Y = I_Y \odot R$. We use $\odot$ to denote elementwise products.

Problem definition: Given a matrix X that is a subset of rating matrix R, we predict the rating scores in another of R's subset Y based on X.

3 Related work

In the past, many researchers have explored memory-based approaches to CF, where many of them can be regarded as aiming at improving the similarity measurement (Breese et al. 1998; Delgado 1999; Herlocker et al. 2002; Ma et al. 2007). A drawback of these methods is that these similarity measurements were not adaptive to different datasets or contain some parameters needed to be tuned but not learned. Some researchers considered how to utilize the user-based and item-based approaches together (Wang et al. 2006; Ma et al. 2007). Wang et al. (2006) proposed a probabilistic fusion model to combine the user-based method with the item-based method. They found that fusing all the ratings in the user–item matrix can help solve the data sparseness problem. A drawback is that they estimated the user similarities and item similarities independently and thus omit the relationship between these similarities. Ma et al. (2007) proposed a method to fill in the missing values before making predictions, which had the drawback similar to that in Wang et al. (2006). Jin et al. (2004)

developed a method for similarity learning, where an automatic weighting scheme for items was designed. Their method aimed at finding the optimal weights that could form a clustered distribution for user vectors in an item space by bringing similar users closer and dissimilar users farther away. However, they only considered the similarity weights for items.

Model based approaches rely on using a statistical model to make predictions. A typical model-based approach is the low-rank matrix approximation based approach (Vozalis and Margaritis 2007; Brand 2003; Zhang et al. 2005), where CF is viewed as a missing value prediction problem for a rating matrix, which can be solved through SVD. Such SVD based approaches 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. Other similar approaches include Hofmann (2004) and Canny (2002). 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. Recognizing that a user may have diverse interests and an item may have multiple functions, Si and Jin (2003) proposed a flexible mixture model for CF. They are among the first to relax the restriction that users and items fall into the same classes. A drawback is that their probabilistic model regarded the ratings as discrete values. Another drawback is that they ignored the relation between ratings. A consequence is that they did not consider rating scores of 3 and 2 to be closer to each other than the rating scores of 5 and 1.

3.1 Memory-based CF

In this section, we review memory-based and SVD-based approaches for CF.

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

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

where r_{um} represents the rating for an item m from a user u. C_u is the set of nearest neighbors of a user u. A user v has an influence weight s_{uv} on u, which can be calculated by normalizing the PCC (Resnick et al. 1994). In other words, $s_{uv} = PCC(u, v)/\sum_{w \in C_u} PCC(u, w)$, where

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

An alternative similarity measure is the VSS model (Breese et al. 1998). Based on this similarity model, we have $s_{uv} = VSS(u, v)/\sum_{w \in C_u} VSS(u, w)$, 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 user u.

Similar to the user-based approach, item-based predictions can be made based on the following relation:

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

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

3.2 Model-based CF using low-rank matrix approximation

SVD is a matrix analysis method used by many researchers for CF (Vozalis and Margaritis 2007; Brand 2003; Zhang et al. 2005). 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. Therefore the loss function to optimize is (Funk 2006; Wu 2007)

$$l = ||I_X \odot (X - UV^{\mathrm{T}})||_F^2 + \alpha(||U||_F^2 + ||V||_F^2) \tag{5}$$

where $\odot$ stands for the element-wise multiplication, $|| \cdot ||_F^2$ denotes the Frobenius norm, and I_X is the indicator function, whose element $I_X(i, j)$ can take on a value of 1 if the user i rated the movie j, and 0 otherwise. The parameter α controls the weight of the regularization term. In addition, U is a lower dimensional representation for users and V is a lower dimensional representation for items. The diagonal matrix Σ in a traditional SVD model is merged into U and V for simplicity. The last term is a regularization term that prevents the model from overfitting. *Unobserved entries* Y are then predicted by $Y = I_Y \odot (UV^{\mathrm{T}})$. The regularized SVD method has been widely used as a key component in the competition of Netflix Prize (Funk 2006; Wu 2007).

Another adaptation of the SVD-based method is to use the EM algorithm to solve the missing value problem (Zhang et al. 2005). The basic idea is to iteratively estimate the missing ratings and conduct SVD decomposition. This method may be time consuming for a large matrix.

4 Similarity metric learning with matrix factorization

We present our main contributions in this section. To begin with, we formulate memory-based approaches in matrix form and based on the form, we can naturally extend them to one-directional similarity learning which is defined in the following section.

4.1 One-directional similarity learning

Memory-based CF methods are usually distinguished from model-based approaches and regarded as heuristic-based approaches (Adomavicius and Tuzhilin 2005). In a user perspective, a user-based approach assumes that similar users tend to have similar ratings on the same items. From the perspective of items, item-based approaches assume that users have similar ratings on the similar items. Although these ideas are intuitive, the concept of "similarity" is crucial and nontrivial to define.

Equation 1 presents the user-based approach. In a matrix form, it can be stated as

$$Y = I_Y \odot (\widehat{S}_1 X) \tag{6}$$

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

$$\widehat{S}_1(u, v) = \begin{cases} s_{uv}, & v \in C_u, \quad (7) \\ 0, & \text{otherwise.} \quad (7') \end{cases}$$

where C_u is the set of nearest neighbors of a user u.

Similar to the user-based approach, item-based methods (Eq. 4) can be stated in the matrix form as

$$Y = I_Y \odot (X \widehat{S}_2) \tag{8}$$

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

$$\widehat{S}_2(m, n) = \begin{cases} s_{mn}, & n \in C_m, \quad (9) \\ 0, & \text{otherwise.} \quad (9') \end{cases}$$

where C_m is the set of nearest neighbors of item m.

Noticing that X and Y are both subsets of the rating matrix R, for user based approach, we have

$$(I_Y \odot R) = I_Y \odot (\widehat{S}_1 (I_X \odot R)) \tag{10}$$

For the item-based approach, we have

$$(I_Y \odot R) = I_Y \odot ((I_X \odot R) \widehat{S}_2) \tag{11}$$

Equations 10 and 11 can actually be seen as matrix reconstruction equations with respect to R. By replacing Y on the left side of the equation with X, we can obtain matrix factorization equations for similarity matrix learning.

Finally, we can obtain a model for similarity learning in the user based approach

$$(I_X \odot R) = I_X \odot (S_1(I_X \odot R)) \tag{12}$$

The item based approach can be easily derived in the same way, which is omitted here.

In the above formulas, the similarity matrices S_1 and S_2 are no longer predefined as in previous memory-based approaches. Instead, they can be learned from the observed data X. This is one of our key contributions in this paper. Note that the above model is a one-directional similarity learning model; in the next section, we extend the model to bi-directional.

4.2 Bi-directional similarity learning

As stated earlier, one-directional models treat 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 (Wang et al. 2006; Ma et al. 2007) have found that the combination of user-based and item-based approaches can indeed boost the performance of CF. 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 advantages of both methods.

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

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

In this formula, we extend the neighborhood range to all users and all items, indicating that all ratings are interconnected. In this way, 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 a matrix form

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

where S_1 and S_2 are also variables we need to learn. The matrix S_1 is a row (user) similarity matrix and S_2 is a column (item) similarity matrix. Similar to one-directional similarity learning, we have a similarity learning problem in matrix factorization form stated as:

$$(I_X \odot R) = I_X \odot (S_1(I_X \odot R)S_2) \tag{15}$$

The above models are free-formed, which contain too many variables to learn. In the following section, we reduce the number of variables by requiring that the matrix be of low ranks.

4.3 Low-rank asymmetric similarity matrix

To reduce the number of parameters in a similarity matrix $S_i (i = 1, 2)$, we can require S_i to be low-rank matrix and factorize S_i with $S_1 = U_1 U_2^{\mathrm{T}}$ and $S_2 = V_1 V_2^{\mathrm{T}}$, where U_i are rank-K_U matrices and V_i are rank-K_V matrices with K_U denoting the number of latent factors for users and K_V be the number of latent factors for items.

If we let $U_1 = U_2$ and $V_1 = V_2$, the similarity matrices S_1 and S_2 are symmetric. Then, we have a factorization problem with missing values in a one-directional case:

$$(I_X \odot R) \approx I_X \odot (UU^{\mathrm{T}}(I_X \odot R)) \tag{16}$$

In bi-directional cases, we have

$$(I_X \odot R) \approx I_X \odot (UU^{\mathrm{T}}(I_X \odot R)VV^{\mathrm{T}}) \tag{17}$$

where we use $\approx$ because the exact factorization may not exist. We allow the "similarity matrix" to have negative value because negative correlation between users do exist in real word problems. The symmetry assumption can be used to decrease the number of variables that we need to learn. As can be seen in Sect. 3.1, we have $PCC(u, v) = PCC(v, u)$ and $VSS(u, v) = VSS(v, u)$ according to the definitions. Thus, the functions to calculate PCC and VSS are symmetric. This indicates that when considering the user relation (or item relation) as a graph, the graph is undirected and the influence of the user u on the user v is the same as that of the user v on the user u. This strong assumption may not be true in the real world. For example, some users in a social network may have more impact than others. When they recommend some items, their influence on their neighbors is much larger than vice versa.

The above consideration motivates us to consider a asymmetric similarity matrix model. We can naturally extend the above similarity learning-based model to an asymmetric model by requiring S to be asymmetric. With a free form low-rank similarity matrix, the asymmetric one-directional matrix factorization model becomes

$$(I_X \odot R) \approx I_X \odot (UV^{\mathrm{T}}(I_X \odot R)) \tag{18}$$

Likewise, the bi-directional matrix factorization model becomes

$$(I_X \odot R) \approx I_X \odot (U_1 U_2^{\mathrm{T}}(I_X \odot R)V_1 V_2^{\mathrm{T}}) \tag{19}$$

The asymmetric similarity learning formulation is more adaptive with the symmetric similarity learning problem regarded as a special case of the asymmetric similarity learning problem.

4.4 Algorithms for similarity learning

The approximate factorization problem can be converted to a minimization problem of the norm of the difference between the original matrix R and the approximation matrix, which we denote R'. We use the Frobenius norm which has the advantage of being easy to be optimized, but this does not prevent the approximation models to generalize to other loss functions as in Dhillon and Sra (2005).

The regularized matrix factorization (Eq. 5) comes from the work on maximum margin matrix factorization (Srebro et al. 2005). The last term in Eq. 5 is a regularization term which prevents the model from overfitting (Srebro et al. 2005). We can encode this term in our model. Now the loss function we are going to minimize is

$$l = ||E||_F^2 + \alpha \sum_i (||U_i||_F^2 + ||V_i||_F^2) \tag{20}$$

where E is the difference between the original matrix and the approximation matrix. We have four types of approximation factorization with the dimensions of one or bi-directional and symmetric or asymmetric. The loss functions of the corresponding four types are listed below.

– For the one-directional symmetric case, $E = I_X \odot R - I_X \odot (UU^{\mathrm{T}}R)$ and the loss is

$$l = ||I_X \odot R - I_X \odot (UU^{\mathrm{T}}(I_X \odot R))||_F^2 + \alpha||U||_F^2 \tag{21}$$

– For the one-directional asymmetric case $E = I_X \odot R - I_X \odot (UV^{\mathrm{T}}R)$ and the loss is

$$l = ||I_X \odot R - I_X \odot (UV^{\mathrm{T}}(I_X \odot R))||_F^2 + \alpha(||U||_F^2 + ||V||_F^2) \tag{22}$$

– For the bi-directional symmetric case $E = I_X \odot (X - UU^{\mathrm{T}}XVV^{\mathrm{T}})$ and the loss is

$$l = ||I_X \odot R - I_X \odot (UU^{\mathrm{T}}(I_X \odot R)VV^{\mathrm{T}})||_F^2 + \alpha(||U||_F^2 + ||V||_F^2) \tag{23}$$

– For the bi-directional asymmetric case $E = I_X \odot (X - U_1U_2^{\mathrm{T}}XV_1V_2^{\mathrm{T}})$ and the loss is

$$\begin{aligned} l = {} & ||I_X \odot R - I_X \odot (U_1U_2^{\mathrm{T}}(I_X \odot R)V_1V_2^{\mathrm{T}})||_F^2 \\ & + \alpha(||U_1||_F^2 + ||U_2||_F^2 + ||V_1||_F^2 + ||V_2||_F^2) \end{aligned} \tag{24}$$

We use gradient-descent approaches to solve the minimization problem. Many gradient-descent based algorithms have been developed for optimization problems such as conjugate gradient (Press et al. 1992) and SMD (Schraudolph 1999). In this paper we use the adaptive-gain gradient decedent algorithm (Almeida et al. 1998) to minimize the loss function, which is described in Algorithm 1. The advantage of adaptive

– Derivative for one-directional symmetric similarity loss

$$\frac{\partial l}{\partial U} = -2EX^{\mathrm{T}}U - 2XE^{\mathrm{T}}U + 2\alpha U$$

– Derivatives for one-directional asymmetric similarity loss

$$\frac{\partial l}{\partial U} = -2EX^{\mathrm{T}}V + 2\alpha U$$
$$\frac{\partial l}{\partial V} = -2XE^{\mathrm{T}}U + 2\alpha V$$

– Derivatives for bi-directional symmetric similarity loss

$$\frac{\partial l}{\partial U} = -2(EVV^{\mathrm{T}}X^{\mathrm{T}}U + XVV^{\mathrm{T}}E^{\mathrm{T}}U) + 2\alpha U$$
$$\frac{\partial l}{\partial V} = -2(E^{\mathrm{T}}UU^{\mathrm{T}}XV + X^{\mathrm{T}}UU^{\mathrm{T}}EV) + 2\alpha V$$

– Derivatives for bi-directional asymmetric similarity loss

$$\frac{\partial l}{\partial U_1} = -2EV_2V_1^{\mathrm{T}}X^{\mathrm{T}}U_2 + 2\alpha U_1$$
$$\frac{\partial l}{\partial U_2} = -2XV_1V_2^{\mathrm{T}}E^{\mathrm{T}}U_1 + 2\alpha U_2$$
$$\frac{\partial l}{\partial V_1} = -2X^{\mathrm{T}}U_2U_1^{\mathrm{T}}EV_2 + 2\alpha V_1$$
$$\frac{\partial l}{\partial V_2} = -2E^{\mathrm{T}}U_1U_2^{\mathrm{T}}XV_1 + 2\alpha V_2$$

Fig. 1 Derivatives for the loss functions (21)–(24)

gain-gradient decedent algorithm includes easy implementation and fast convergence speed.

A key component of gradient based optimization algorithms is to compute the gradients. Unlike EM-style algorithms, matrix based algorithms have the advantages of elegant formulation and easy implementation. We can derive the derivatives of the norm in Fig. 1.

We notice 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 matrix multiplication.

Theorem 1 *The time complexity of computing the derivative is $O(nnz(X)K + NK^2)$, where $nnz(X)$ is the number of nonzero elements in X and $K = \max\{K_U, K_V\}$. N is the maximum of number of users and items.*

Proof To compute the term $EVV^{\mathrm{T}}X^{\mathrm{T}}U$, we can choose the order of product computations to always conduct the product of sparse matrix to a low-rank matrix or two low-rank matrices. For example, $((EV)((V^{\mathrm{T}}X^{\mathrm{T}})U))$. For each matrix production, the complexity is $O(nnz(X)k + NK^2)$. Therefore, the overall time complexity of calculating the derivatives is $O(nnz(X)k + NK^2)$. □

Algorithm 1 Bi-directional asymmetric similarity learning using adaptive gain

```
Input: training data $X$, parameters $\mu$, $K_U$, $K_V$ and $T$
Output: $U_1, V_1, U_2, V_2$
Initialization: Randomly initialize $U$ and $V$
FOR $t = 1$ TO $T$:
    Update $U_1$:  $U_1^{(t+1)} = U_1^{(t)} - \eta_{U_1}^{(t)} \odot \frac{\partial l}{\partial U}_1^{(t)}$
    Update $U_2$:  $U_2^{(t+1)} = U_2^{(t)} - \eta_{U_2}^{(t)} \odot \frac{\partial l}{\partial U_2}^{(t)}$
    Update $V_1$:  $V_1^{(t+1)} = V_1^{(t)} - \eta_{V_1}^{(t)} \odot \frac{\partial l}{\partial V_1}^{(t)}$
    Update $V_2$:  $V_2^{(t+1)} = V_2^{(t)} - \eta_{V_2}^{(t)} \odot \frac{\partial l}{\partial V_2}^{(t)}$
    Update $\eta_{U_1}$:
        $\eta_{U_1}^{(t)} = \eta_{U_1}^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_{U_1}^{(t)} \odot \frac{\partial l}{\partial U_1}^{(t-1)} \odot \frac{\partial l}{\partial U_1}^{(t)})$
    Update $\eta_{U_2}$:
        $\eta_{U_2}^{(t)} = \eta_{U_2}^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_{U_2}^{(t)} \odot \frac{\partial l}{\partial U_2}^{(t-1)} \odot \frac{\partial l}{\partial U_2}^{(t)})$
    Update $\eta_{V_1}$:
        $\eta_{V_1}^{(t)} = \eta_{V_1}^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_{V_1}^{(t)} \odot \frac{\partial l}{\partial V_1}^{(t-1)} \odot \frac{\partial l}{\partial V_1}^{(t)})$
    Update $\eta_{V_2}$:
        $\eta_{V_2}^{(t)} = \eta_{V_2}^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_{V_2}^{(t)} \odot \frac{\partial l}{\partial V_2}^{(t-1)} \odot \frac{\partial l}{\partial V_2}^{(t)})$
```

5 Relation to other MF models

In this section, we discuss the relation between our model and other matrix factorization models. When the similarity matrix is of full rank, the solution to Eq. 14 is trivial but not useful since any U and V that can satisfy $UU^{\mathrm{T}} = I$ and $VV^{\mathrm{T}} = I$ is a solution. However, when the similarity matrix has low rank, the solution is not trivial anymore and has deep connections to the SVD. With the low rank assumption, the learned similarities between users as well as items can be regarded as being influenced by some latent factors. In contrast to some previous latent-factor models, such as SVD (Zhang et al. 2005) and Aspect Models (Hofmann 2004), our model is more flexible in that it does not require the number of factors underlying the user space and the item space to be the same.

Under the special case when these factors are the same, the following theorem relates the bi-directional symmetric SLCF model and SVD.

Theorem 2 *If we disregard the missing data and require that the ranks of U and V are the same, SVD find a solution to $X = UU^{\mathrm{T}}XVV^{\mathrm{T}}$.*

Proof Suppose that $X = U\Sigma V^{\mathrm{T}}$. By plugging it into $UU^{\mathrm{T}}XVV^{\mathrm{T}}$, we obtain $UU^{\mathrm{T}}XVV^{\mathrm{T}} = UU^{\mathrm{T}}U\Sigma V^{\mathrm{T}}VV^{\mathrm{T}} = U\Sigma V^{\mathrm{T}} = X$. □

The equivalence of bi-directional SLCF 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 will see this further in the experimental section again.

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 spaces for row vectors and column vectors. Therefore, our model can also be regarded as bi-dimensional reduction-based method for cases when row vectors and column vectors with different dimensions. We can also find the relationship between the two spaces, as the two sets of bases satisfy the following equation:

$$U \cdot B_1 = B_2 \cdot V \tag{25}$$

where the users' basis $B_1 = U^{\mathrm{T}} X V V^{\mathrm{T}}$ and the items' basis $B_2 = U U^{\mathrm{T}} X V$.

Xu and Gong (2004) proposed a concept factorization model for document clustering, which has the following form

$$X = X U V^T \tag{26}$$

We can observe it is similar to the one-directional similarity learning model, except that the missing value is just treated as 0. Ding et al. (2006) studied a similar model called convex-NMF that has nonnegative constraints on U and V,

$$X = X U V^T, U, V > 0 \tag{27}$$

We can naturally extend these one-directional models into bi-directional models using our above framework.

6 Rating prediction based on bidirectional similarity learning

To take the full advantage of the model, we design algorithms for rating prediction with different strategies based on learned similarity. In this section, we discuss two types of similarity learning based CF strategies; namely, reconstructed matrix based and traditional memory based strategies.

6.1 Matrix reconstruction strategy (R-SLCF)

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 a row u in the matrix U denoted by U_u and the item i's profile corresponds to a 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,

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

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).

An additional problem facing the matrix reconstruction strategy needs to be addressed. When new users and new items arrive, one needs to retrain the model based on the whole new dataset so as to make predictions for the new users and items. This procedure is clearly too time-consuming and is often infeasible. In the following, we present an online version of the R-SLCF method, which can bring new users and items into the model *without retraining* on the whole dataset. The key issue is how to embed the new users and items into the previous model.

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

$$\widehat{Y} = U_{\widehat{Y}} \cdot U^{\mathrm{T}} X V V^{\mathrm{T}} = U_{\widehat{Y}} \cdot B_1 \tag{28}$$

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

$$U_{\widehat{Y}} = \widehat{Y} \cdot ((I_{\widehat{Y}} \odot B_1) \cdot (I_{\widehat{Y}} \odot B_1)^{\mathrm{T}} + \lambda \mathbb{I})^{-1} \tag{29}$$

where $\mathbb{I}$ is the 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 (Kirsch 1996).

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

$$\widehat{Y} = U U^{\mathrm{T}} X V \cdot V_{\widehat{Y}}^{\mathrm{T}} = B_2 \cdot V_{\widehat{Y}}^{\mathrm{T}} \tag{30}$$

$$U_{\widehat{Y}} = \widehat{Y} \cdot ((I_{\widehat{Y}} \odot B_2) \cdot (I_{\widehat{Y}} \odot B_2)^{\mathrm{T}} + \lambda \mathbb{I})^{-1} \tag{31}$$

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

$$R_{\widehat{Y}} = U_{\widehat{Y}} U^{\mathrm{T}} X V V_{\widehat{Y}}^{\mathrm{T}}$$

Although we need to calculate the inverse of matrices in projection based strategy, the matrices are small and can be computed efficiently. In the following, we will refer to our batch matrix reconstruction method as R-SLCF, and the online version of the algorithm as online R-SLCF.

6.2 Improved memory-based strategy (M-SLCF)

An intuitive way to use the learned similarity is to improve memory-based approaches. 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, which is especially helpful for comparing our learned similarity with the pre-defined similarity such as PCC. One disadvantage of this strategy is it needs to compute pairwise similarity values between users or items in a way similar to memory-based approaches, which is computationally infeasible in large-scale recommendation systems. This rating prediction method is also used as a baseline to R-SLCF in our experiments (Sect. 7.3).

7 Experiments

In this section, we empirically demonstrate the computational and performance properties of our similarity learning methods as well as rating prediction strategies.

7.1 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 CF. It contains 2,811,983 ratings from 72,916 users on 1,628 movies with scale 1–6.
- 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.

7.2 Evaluation metrics

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

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

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

7.3 Comparison with other approaches

The baselines used in this paper include user-based method using PCC, item-based method using PCC and regularized SVD method. We also compare our algorithms with the method proposed in Wang et al. (2006), which fuses the similarities among

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

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

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

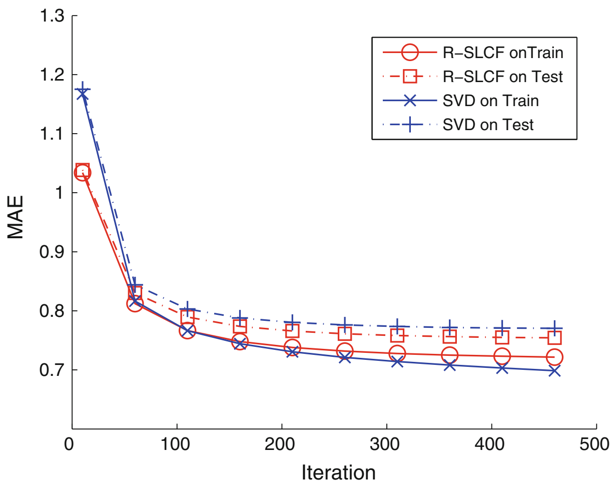

Fig. 2 Converge curves of R-SLCF and regularized SVD. Evaluated by MAE on Movielens with the parameter $K = 10$ for regularized SVD, $K_U = K_V = 10$ for R-SLCF

users as well as items. We also conduct experiments on Netflix dataset, although our results are not comparable with the top results on the leaderboard since they used a hybrid methods in the end. We should note that the regularized SVD, which is one of the best non-ensemble-based algorithms in the Netflix Prize competition (Funk 2006), is also included in our baselines.

7.3.1 Comparison with SVD-based approaches

In this section, we compare our SLCF model with the regularized SVD model introduced in Sect. 3.2 in different ways. Figure 2 shows the convergence curves of SLCF and regularized SVD. In this experiment, we use the same optimization algorithm (i.e., adaptive gain) with the same initial points[6] for U and V to run the algorithms. We also tune the best step length for each algorithm. We can see that SLCF converges faster than regularized SVD and is able to find better solutions. We also note that regularized SVD has smaller MAE values on the training data but larger MAE values on the test data when compared with R-SLCF, which indicates that the regularized SVD is more likely to overfit than SLCF.

Table 1 shows the performance comparison of SLCF and regularized SVD model with various values of K. In this experiment, R-SLCF uses the same number of variables with regularized SVD to ensure fairness of comparison. We can see that SLCF clearly outperforms the regularized SVD model. The results also indicate that our model is different from regularized SVD even when $K_U = K_V$ when there are missing values.

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

Table 1 MAE comparison of R-SLCF with SVD for different K

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**

For R-SLCF1 we require $K_U = K_V = K$ and for R-SLCF2 we require $K_U + K_V = 2K$

Table 2 MAE comparison of R-SLCF and M-SLCF with memory-based method and item-based method

Dataset	# Rating	I-based	U-based	M-SLCF	R-SLCF
MovieLens	$N = 30$	1.0936	0.8785	0.8676	**0.8418**
	$N = 40$	0.9587	0.8527	0.8405	**0.8113**
	$N = 50$	0.9144	0.8451	0.8398	**0.8104**
	$N = 60$	0.8648	0.8239	0.8106	**0.8056**
EachMovie	$N = 30$	1.7238	0.9919	0.9449	**0.9347**
	$N = 40$	1.6437	0.9908	0.9391	**0.9297**
	$N = 50$	1.7792	0.9836	0.9817	**0.9338**
	$N = 60$	1.6656	0.9886	0.9559	**0.9327**
Netflix	$N = 30$	0.9568	0.8804	0.8291	**0.7974**
	$N = 40$	0.8647	0.8390	0.8308	**0.7782**
	$N = 50$	0.8293	0.8114	0.7872	**0.7672**
	$N = 60$	0.7934	0.7774	0.7648	**0.7439**

The significance test is performed with significance level 0.05. $N = 30$ means only users with ratings no larger than 30 are included

7.3.2 Comparison with memory-based approaches

In this section, we compare SLCF with user-based (U-based) and item-based (I-based) approaches. Experimental results on three datasets are shown in Table 2. Comparisons are conducted under different degree of 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 both M-SLCF and R-SLCF outperform the baselines under different sparseness conditions and R-SLCF performs better than M-SLCF.

We also compare R-SLCF with another stat-of-the-arts memory-based algorithm called Similarity Fusion (SF) (Wang et al. 2006) 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 this algorithm, we follow exactly the same experiment settings in the paper. Then, for the performance of the SF method, we quote its results from the paper (Wang et al. 2006). We can see that SLCF outperforms SF significantly.

Table 3 Compare with SF on MovieLens

# Users	100			200			300		
# Ratings	5	10	20	5	10	20	5	10	20
R-SLCF	**0.838**	**0.770**	**0.771**	**0.799**	**0.768**	**0.763**	**0.787**	**0.753**	**0.739**
CFONMTF	**0.838**	0.801	0.799	0.825	0.790	0.787	0.801	0.781	0.770
SF	0.847	0.774	0.792	0.827	0.773	0.783	0.804	0.761	0.769
SCBPCC	0.848	0.819	0.789	0.831	0.813	0.784	0.822	0.810	0.778

# Users is the number of training users and # Ratings is the number of ratings

In addition, we have compared with two other state-of-the-arts methods that have utilized clustering on both user side and item side. One is cluster-based Pearson Correlation Coefficient (SCBPCC) (Xue et al. 2005) and the other one is based on orthogonal nonnegative matrix factorization (CFONMFTF) (Chen et al. 2009). The results are also shown in Table 3. We found that our proposed method still outperforms both these methods significantly.

7.4 Further study of SLCF models

In this section, we further study the properties of our proposed SLCF model. We first investigate the effect of bi-directional learning and asymmetric similarity to the performance compared to one-directional and symmetric in Sect. 7.4.1 and Sect. 7.4.2. Then we show the influence of the parameter K to our model in Sect. 7.4.3. In Sect. 7.4.4 we investigate the difference between the R-SLCF and online R-SLCF. We also show how sparseness influences the learned similarity compared with memory-based approaches.

7.4.1 Comparison between one-directional and bi-directional similarity

We first study the difference between one-directional similarity and bi-directional similarity. Figures 3 and 4 show the comparisons of MAE between one-directional and bi-directional similarity learning models of symmetric and asymmetric similarity matrices, respectively. They both show the performance advantages of bi-directional learning.

7.4.2 Comparison between symmetric and asymmetric similarity

Figure 5 compares the convergence curves of one-directional similarity for symmetric and asymmetric similarity learning. From this figure, we can observe that the asymmetric similarity model significantly outperforms the symmetric model with the same number of latent factors, which indicates the one-directional symmetric similarity measurement model is not effective enough for the CF problem. We can also see that our proposed asymmetric similarity shows its advantage over one-directional similarity learning models.

Figure 6 compares convergence curves of bi-directional similarity for symmetric and asymmetric similarity learning. The figure shows that the asymmetric model still

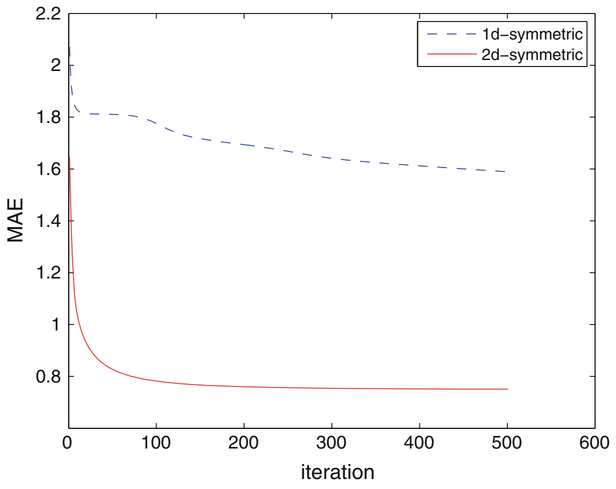

Fig. 3 Comparison of one-directional symmetric (1d-symmetric) and bi-directional symmetric (2d-symmetric) similarity matrix learning using MAE

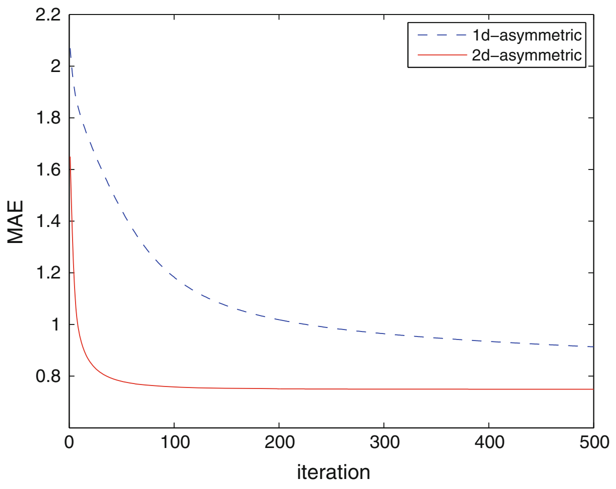

Fig. 4 Comparison of one-directional asymmetric (1d-asymmetric) and bi-directional asymmetric (2d-asymmetric) similarity matrix learning using MAE

outperforms the symmetric model. The asymmetric model also converges faster than symmetric model. Figure 7 shows the comparisons between CPU time of bi-directional asymmetric model and symmetric model with respect to MAE.

7.4.3 Impact of K_U and K_V

Two important parameters of bi-directional SLCF methods are the rank of user similarity matrix K_U and the rank of item similarity matrix K_V. We conduct experiments on the MovieLens dataset to study the impact of K_U and K_V to the performance.

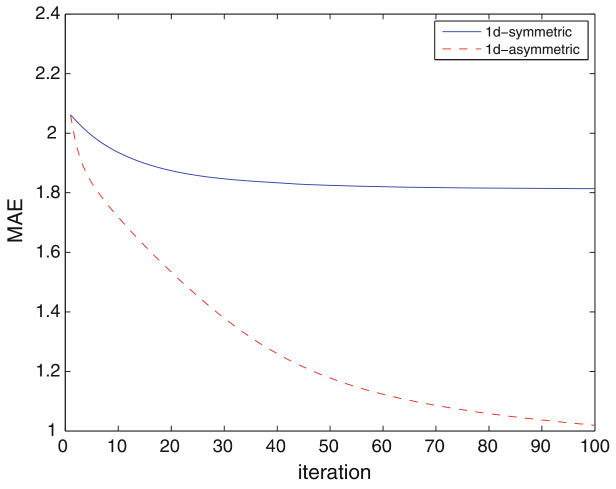

Fig. 5 Comparison of symmetric and asymmetric one directional similarity matrix learning using MAE

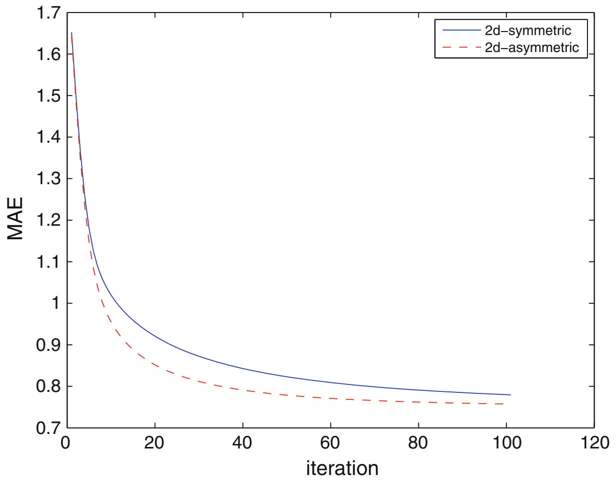

Fig. 6 Comparison of symmetric and asymmetric bidirectional similarity matrix learning using MAE

Figure 8 shows the two 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 4 shows the best K_V for given K_U and Table 5 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 indicates 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 the rating matrix, and vice versa. From the MAE results of Fig. 8, the best result is obtained when both user and item factors are considered ($K_U = 12$, $K_V = 8$). This confirms our initial conjecture that the user and

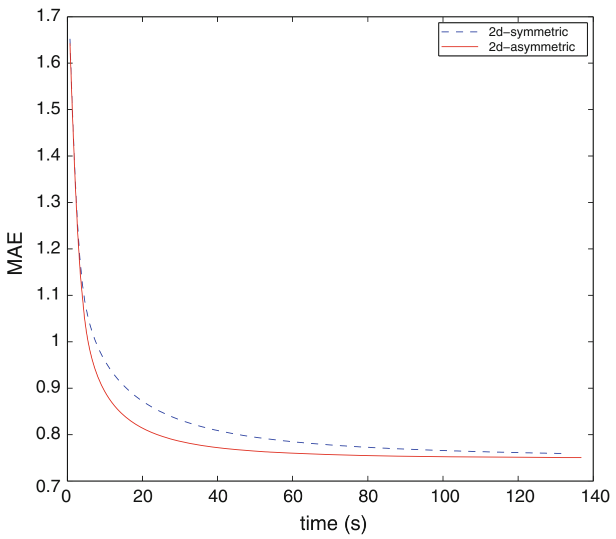

Fig. 7 Comparison of CPU time between asymmetric and symmetric models

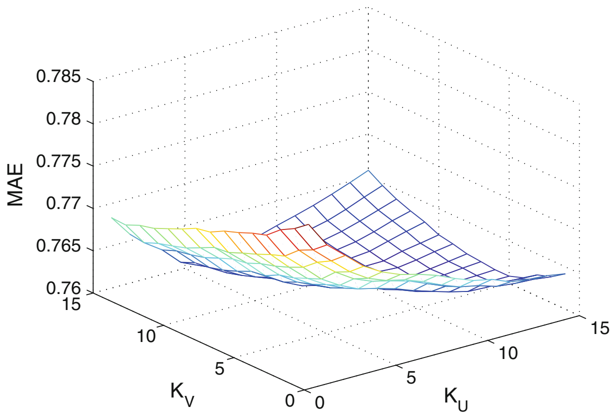

Fig. 8 MAE surface of R-SLCF on MovieLens dataset. Numbers of user factor (K_U) and item factor (K_V) are varied simultaneously

item spaces should be modeled with different numbers of factors. Another parameter in our model is α that controls the balance between prediction error on training data and model complexity. After testing on different values, we find $\alpha = 0.0001$ gives the best results in our experiments.

7.4.4 The difference of R-SLCF and online R-SLCF

The online R-SLCF algorithm is aimed at solving the new-user and new-item problem in CF. In this experiment, we compare the accuracy of prediction by batch R-SLCF and online R-SLCF. Figure 9 shows the comparison results on MovieLens. In this

Table 4 Optimal K_V given K_U

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

Table 5 Optimal K_U given K_V

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

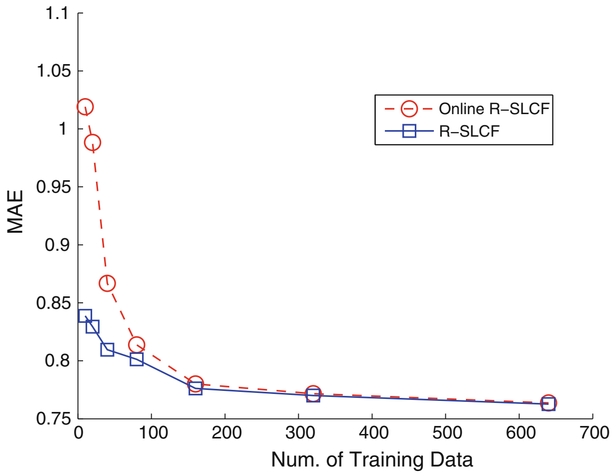

Fig. 9 Comparison of online R-SLCF and R-SLCF. Evaluated by MAE on Movielens with 200 user as test data, $K_U = K_V = 10$ for SLCF

experiment, we use 200 users as testing data. When training users are very few, online R-SLCF is not as accurate as R-SLCF. But as the number of training users increases, their performance become very close. Figure 10 shows the comparison of CPU time on R-SLCF and online R-SLCF, which shows the advantage of the online algorithm on efficiency.

An important parameter in online R-SLCF is λ. Figure 11 shows the influence of λ on the prediction accuracy. After testing different values of λ, we find $\lambda = 1$ to be a good choice, which we use in our other experiments.

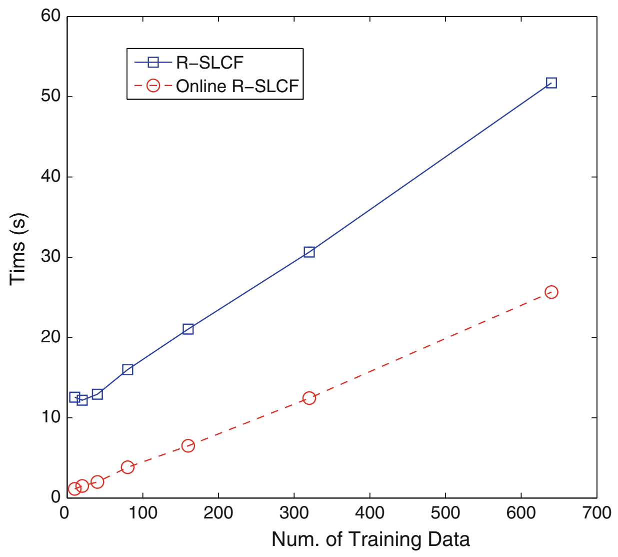

Fig. 10 Comparison of CPU time between R-SLCF and online R-SLCF

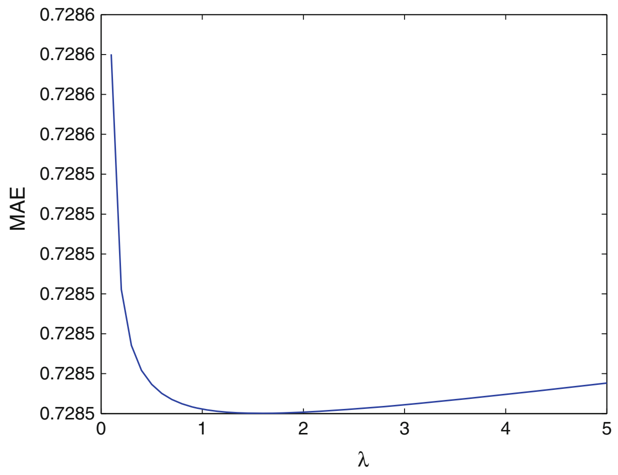

Fig. 11 Influence of parameter λ to the performance of online R-SLCF

7.4.5 Impact of data sparseness

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

We first filtered the EachMovie dataset by keeping the users who have rated different numbers of movies (from less than 50 to less than 5 in this experiment). In

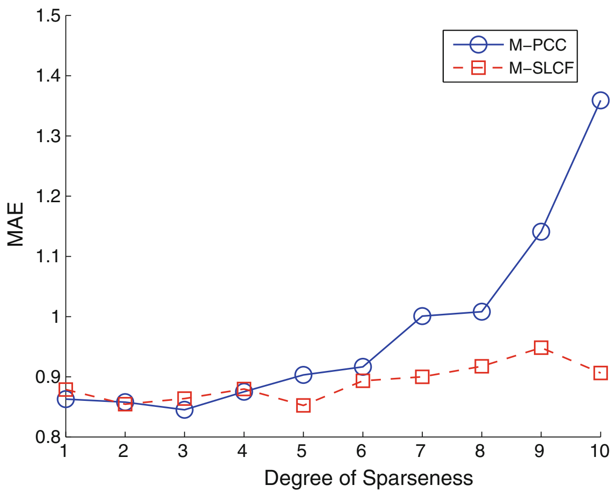

Fig. 12 Comparison of PCC and M-SLCF on similarity with different degree of sparseness. Evaluated using MAE on EachMovie with 50 nearest neighbors

this way, we constructed the datasets with different degrees of sparseness. We use the user-based method with neighbors found by SLCF and compare the results with PCC. When the data are not that sparse, PCC can do a good job in finding the nearest neighbors. However, when the degree of sparseness increases, the performance drops. In Fig. 12, we can clearly see that SLCF is able to find neighbors more accurately when sparseness is increased. Figure 13 verifies our conclusion from a different perspective. It shows how SLCF and PCC performed with different numbers 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 with higher recall. Therefore, when more neighbors are used, SLCF performs much better.

8 Conclusions and future work

In this paper, we proposed a novel matrix factorization based model for learning user and item similarities simultaneously for CF. The similarity measurement can be asymmetric and can be learned from the data using matrix factorization methods. We proposed an efficient learning algorithm as well as effective prediction strategies. We showed our learned similarity measurement significantly outperforms the predefined ones. The experiments showed our method can outperform baselines such as traditional memory-based approaches and a low-rank matrix approximation model. Furthermore, our online version of the prediction algorithm is shown to be effective and more efficient for handling new users and items.

For our future work, we plan to develop more efficient algorithms to learn our model in larger scale datasets. Although we focused on CF 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. We also plan

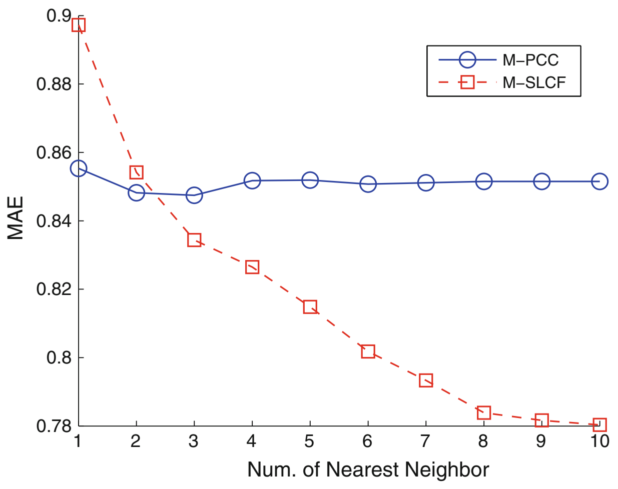

Fig. 13 Comparison of PCC and M-SLCF on similarity with different number of nearest neighbors. Evaluated using MAE on EachMovie with sparseness degree 5

to extend our model to multi-directional cases where more than two types of entities are involved.

Acknowledgments Bin Cao and Qiang Yang are supported by a grant from MSRA (MRA07/08.EG01) and RGC/NSFC grant N_HKUST624/09. We thank the anonymous reviewers for their useful comments.

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE TKDE 17(6):734–749

Almeida LB, Langlois T, Amaral JD, Plakhov A (1998) Parameter adaptation in stochastic optimization. In: Saad D (ed) On-line learning in neural networks. Cambridge University Press, Cambridge, pp 111–134

Brand M (2003) Fast online SVD revisions for lightweight recommender systems. In: Proceedings of SIAM ICDM

Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th conference on UAI, pp 43–52

Canny J (2002) Collaborative filtering with privacy via factor analysis. In: Proceedings of the 25th SIGIR. ACM, New York, pp 238–245

Cao B, Sun J-T, Wu J, Yang Q, Chen Z (2008) Learning bidirectional similarity for collaborative filtering. In: Proceedings of the ECML/PKDD, part 1, pp 178–194

Chen G, Wang F, Zhang C (2009) Collaborative filtering using orthogonal nonnegative matrix tri-factorization. Inf Process Manag 45(3):368–379

Delgado J (1999) Memory-based weightedmajority prediction for recommender systems. In: ACM SIGIR'99 workshop on recommender systems

Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM TOIS 22(1): 143–177

Dhillon IS, Sra S (2005) Generalized nonnegative matrix approximations with bregman divergences. In: Neural information processing systems. MIT Press, Cambridge, pp 283–290

Ding C, Li T, Jordan M (2006) Convex and semi-nonnegative matrix factorizations for clustering and low-dimension representation. Technical report LBNL-60428. Lawrence Berkeley National Laboratory, University of California, Berkeley

Funk S (2006) Netflix update: try this at home. http://sifter.org/~simon/journal/20061211.html

Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd SIGIR. ACM, New York, pp 230–237

Herlocker J, Konstan JA, Riedl J (2002) An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf Retr 5(4):287–310

Hofmann T (2004) Latent semantic models for collaborative filtering. ACM TOIS 22(1):89–115

Jin R, Chai JY, Si L (2004) An automatic weighting scheme for collaborative filtering. In: Proceedings of the 27th annual international ACM SIGIR. Content-based filtering & collaborative filtering, pp 337–344

Kirsch A (1996) An introduction to the mathematical theory of inverse problems. Springer-Verlag New York Inc., New York

Ma H, King I, Lyu MR (2007) Effective missing data prediction for collaborative filtering. In: Proceedings of the 30th SIGIR. ACM, New York, pp 39–46

Park YJ, Tuzhilin A (2008) The long tail of recommender systems and how to leverage it. In: RecSys'08: proceedings of the 2008 ACM conference on recommender systems. ACM, New York, pp 11–18

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C: the art of scientific computing, 2nd edn. Cambridge University Press, Cambridge

Resnick P, Iacovou N, Suchak M, Bergstorm P, Riedl J (1994) GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the ACM 1994 conference on CSCW. ACM, Chapel Hill, pp 175–186

Sarwar B, Karypis G, Konstan J, Reidl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on WWW. ACM, New York, pp 285–295

Schraudolph NN (1999) Local gain adaptation in stochastic gradient descent. Technical report IDSIA-09-99

Si L, Jin R (2003) A flexible mixture model for collaborative filtering. In: Proceedings of the 12th ICML

Srebro N, Rennie J, Jaakkola T (2005) Maximum-margin matrix factorization. In: Saul LK, Weiss Y, Bottou L (eds) Advances in neural information processing systems, vol 17. MIT Press, Cambridge, pp 1336, 1329

Vozalis MG, Margaritis KG (2007) Using SVD and demographic data for the enhancement of generalized collaborative filtering. Inf Sci 177(15):3017–3037

Wang J, de Vries AP, Reinders MJT (2006) Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th SIGIR. ACM, New York, pp 501–508

Wu M (2007) Collaborative filtering via ensembles of matrix factorizations. In: Proceedings of the KDD cup and workshop

Xu W, Gong Y (2004) Document clustering by concept factorization. In: Proceedings of the 27th SIGIR. ACM, New York, pp 202–209

Xue G-R, Lin C, Yang Q, Xi W, Zeng H-J, Yu Y, Chen Z (2005) Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of the 28th SIGIR. ACM, New York, pp 114–121

Zhang S, Wang W, Ford J, Makedon F, Pearlman J (2005) Using singular value decomposition approximation for collaborative filtering. In: Proceedings of the 7th IEEE CEC. IEEE Computer Society, Washington, DC, pp 257–264