Information Processing and Management 45 (2009) 368–379

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Collaborative filtering using orthogonal nonnegative matrix tri-factorization

Gang Chen *, Fei Wang, Changshui Zhang

State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 15 November 2007
Received in revised form 1 December 2008
Accepted 14 December 2008
Available online 20 January 2009

Keywords:
Collaborative filtering
Orthogonal nonnegative matrix tri-factorization
Co-clustering
Fusion

ABSTRACT

Collaborative filtering aims at predicting a test user's ratings for new items by integrating other like-minded users' rating information. The key assumption is that users sharing the same ratings on past items tend to agree on new items. Traditional collaborative filtering methods can mainly be divided into two classes: memory-based and model-based. The memory-based approaches generally suffer from two fundamental problems: sparsity and scalability, and the model-based approaches usually cost too much on establishing a model and have many parameters to be tuned.

In this paper, we propose a novel framework for collaborative filtering by applying *orthogonal nonnegative matrix tri-factorization* (*ONMTF*), which (1) alleviates the sparsity problem via matrix factorization; (2) solves the scalability problem by simultaneously clustering rows and columns of the user-item matrix. Experiments on the benchmark data set show that our algorithm is indeed more tolerant against both sparsity and scalability, and achieves good performance in the mean time.

1. Introduction

Collaborative filtering aims at predicting a test user's ratings for new items based on a collection of other like-minded users' ratings information. It assumes that users sharing the same ratings on past items tend to agree on new items. Up to now, researches on collaborative filtering can mainly be categorized into two types: *memory-based* and *model-based*.

Memory-based algorithms, including user-based (Breese, Heckerman, & Kadie, 1998; Herlocker, Konstan, Borchers, & Riedl, 1999; Jin, Chai, & Si, 2004; Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994) and item-based (Deshpande & Karypis, 2004; Sarwar, Karypis, Konstan, & Riedl, 2001), first compute similarities between a test user and other users (user-based), or, between a test item and other items (item-based), according to which the top K most similar neighbors of the test one can be identified. Then the unknown rating is predicted by combining the known rating of the neighbors. However, there exist two fundamental challenges for memory-based methods. The first one is *sparsity*. These algorithms rely upon exact matches of two user/item vectors, which cause the algorithms to sacrifice recommender system's coverage and accuracy. More concretely, since the correlation coefficient is only defined between customers who have rated at least one product in common, or the items which have been co-purchased, then many pairs of customers/items will have no correlation at all. As a result, memory-based recommender systems cannot accurately compute the neighborhood and identify the items to recommend, which will certainly lead to poor recommendations. The second one is *scalability*. In practice, both the number of users and items can be quite large (often millions of users and thousands of items). This may slow down the recommendation

* Corresponding author. Tel.: +86 10 627 96 872; fax: +86 10 627 86 911.
E-mail address: g-c05@mails.thu.edu.cn (G. Chen).

0306-4573/$ - see front matter
doi:10.1016/j.ipm.2008.12.004

procedure significantly since K nearest neighbor based algorithms will require too much computations in this case (Sarwar et al., 2001).

Different from memory-based approaches, model-based approaches need to establish a model using training examples that can predict the unknown ratings of a test user. Examples within this category include *decision tree* (Breese et al., 1998), *Bayesian network* (Breese et al., 1998), *clustering models* (Kohrs & Merialdo, 1999; Ungar & Foster, 1998), *latent factor models* (Canny, 2002), *aspect models* (Hofmann & Puzicha, 1999; Si & Jin, 2003), *dimension–reduction methods* (Fisher et al., 2000; Goldberg, Roeder, Gupta, & Perkins, 2001; Rennie & Srebro, 2005; Sarwar, Karypis, Konstan, & Riedl, 2000), etc.. However, just as pointed out by Xue et al. (2005), generating and updating of a model are often time-consuming since there are usually many free parameters to tune.

In this paper, we provide a novel framework for collaborative filtering, which circumvents the problems of traditional memory-based and model-based approaches by applying *orthogonal nonnegative matrix tri-factorization* (*ONMTF*) (Ding, Li, Peng, & Park, 2006). Our algorithm first applies *ONMTF* to simultaneously cluster the rows and columns (also called co-clustering) of the user-item matrix (Ding, He, & Simon, 2005, 2006), and then adopts the user-based and item-based clustering approaches respectively to attain individual predictions for an unknown test rating, finally we will fuse these resultant ratings via a linear combination. Our algorithm possesses the following superiorities:

(1) The sparsity problem has been circumvented due to the application of matrix factorization.
(2) The scalability problem can be greatly alleviated by the application of *ONMTF* technique, since the users and items are simultaneously clustered.
(3) Our method can fuse the prediction results of different types of algorithms (user-based approach and item-based approach), which can get better performances according to our experiments.

The rest of this paper is organized as follows. We first give a brief summary of related work on collaborative filtering in Section 2. Section 3 describes orthogonal nonnegative matrix tri-factorization and its relation to clustering. In Section 4, we elaborate our framework for collaborative filtering by using *ONMTF* and fusing all ratings with predictive value. The data and results of experiments are presented in Section 5, followed by our conclusions in Section 6.

2. Related work

As introduced in Section 1, there are mainly two classes of collaborative filtering algorithms: memory-based and model-based. The memory-based approaches have been widely used in practical systems (Linden, Smith, & York, 2003), including user-based (Breese et al., 1998; Herlocker et al., 1999; Jin et al., 2004; Resnick et al., 1994) and item-based (Deshpande & Karypis, 2004; Sarwar et al., 2001). The key to memory-based approaches is to compute the similarities between pairwise users or items. Many methods have been developed for this task, such as correlation-based, cosine-based and adjusted-cosine (Sarwar et al., 2001). Compared to model-based approaches, the memory-based approaches do not need to train a model and do not have too many parameters to be tuned. Recently, Wang, de Vries, and Reinders (2006) suggested a unifying user-based and item-based collaborative filtering approach by similarity fusion which improves the performances on sparse data. However, their method is based on the traditional memory-based algorithms which have to search for the whole data set to identify the neighbors of a test user or item. Therefore, it still suffers from the scalability problem.

The popular model-based approaches include the clustering techniques (Kohrs & Merialdo, 1999; Ungar & Foster, 1998), aspect models (Hofmann & Puzicha, 1999) and dimensionality reduction approaches for collaborative filtering (Fisher et al., 2000; Goldberg et al., 2001; Rennie & Srebro, 2005; Sarwar et al., 2000). Typically, the clustering techniques can achieve a compact model and circumvent the scalability problem to some extent. The dimensionality reduction methods can alleviate the data sparsity problem. However, just as pointed out by Huang, Chen, and Zeng (2004) and Xue et al. (2005), some potentially useful information may be lost during the dimensionality reduction process. Besides, one disadvantage of the model-based approaches is that there are often many parameters to be tuned such as the number of clusters and the final dimensionality of the space in which the data is embedded.

Recently, researchers have proposed some hybrid approaches in order to combine the strengths of memory-based approaches and model-based approaches (Pennock, Horvitz, Lawrence, & Giles, 2000; Xue et al., 2005). For example, Xue et al. (2005) resolve the sparsity and scalability problems by using clusters to smooth ratings and clustering, respectively. However, they only employ the user-based approaches. Our framework in this paper extends the ideas in Wang et al. (2006) and Xue et al. (2005) and naturally integrates clustering techniques (model-based), user-based approaches and item-based approaches (memory-based) by orthogonal nonnegative matrix tri-factorization.

There is also another type of collaborative filtering methods which is called the content-based approaches (Balabanovic & Shoham, 1997; Claypool et al., 1999; Giles, Bollacker, & Lawrence, 1998; Popescul, Ungar, Pennock, & Lawrence, 2001). This kind of methods mine the content information contained in the items which can be incorporated into traditional collaborative filtering techniques, so they can overcome some shortages of using collaborative filtering methods alone. However, as pointed out by Xue et al. (2005), the content information may be different or expensive to acquire in many cases.

The other technique that is closely related to this paper is the co-clustering approach, which solves the problem of simultaneously clustering rows and columns of a data matrix. Dhillon (2001) and Zha et al. (2001) individually developed a co-

clustering approach by using the bipartite graph formulation and spectral methods, but that demanded each row cluster to be associated with a column cluster. In the past years, co-clustering based on information theory has attracted more and more attention. Tishby, Pereira, and Bialek (1999) first introduced an information bottleneck framework for one-sided clustering. Later, Slonim and Tishby (2000) gave an agglomerative hard clustering version of the information bottleneck method, which could cluster documents after clustering words. El-Yaniv and Souroujon (2001) further extended the framework and provided a co-clustering technique called iterative double clustering. Dhillon, Mallela, and Modha (2003) proposed an information-theoretic co-clustering technique by minimizing the loss of mutual information, and Banerjee, Dhillon, Ghosh, Merugu, and Modha (2004) extended the information-theoretic co-clustering and suggested a generalized maximum entropy co-clustering approach by appealing to the minimum of Bregman information principle. Recently, Ding et al. (2006) gave a novel co-clustering approach based on nonnegative matrix factorization, and Long, Zhang, and Yu (2005) also provided a similar co-clustering method called co-clustering by block value decomposition. George and Merugu (2005) considered a novel collaborative filtering approach based on the co-clustering algorithm in Banerjee et al. (2004), but that was only a model-based approach. In this paper, we use *ONMTF* rather than the method in George and Merugu (2005) to co-cluster, for Long et al. (2005) verified that the co-clustering method based on nonnegative matrix factorization was usually superior to the information-theoretic co-clustering methods in Dhillon et al. (2003) and El-Yaniv and Souroujon (2001) by empirical studies.

3. Orthogonal nonnegative matrix tri-factorization

The *nonnegative matrix factorization* (*NMF*) is first brought into machine learning and data mining fields by Lee and Seung (1999, 2001), and has been widely used in pattern recognition, multimedia, text mining and bioinformatics (Ding et al., 2006). Recently, Ding et al. (2005, 2006) proved the equivalence between *NMF* and *K*-means/spectral clustering, and extended *NMF* to *ONMTF* which could simultaneously cluster rows and columns of an input matrix.

Now let's briefly review the basic idea of *NMF*. Given a nonnegative matrix *X*, *NMF* aims to find two nonnegative matrix factors *U* and *V* such that

$$X \approx UV^T \tag{1}$$

where $X \in \mathbb{R}_+^{p\times n}, U \in \mathbb{R}_+^{p\times k}$ and $V \in \mathbb{R}_+^{n\times k}$ ($\mathbb{R}_+^{n\times k}$ is the set of n-by-k matrices whose elements are all nonnegative). In general, the rank of matrices *U* and *V* is much lower than the one of matrix *X*, i.e. $k \ll \min(p, n)$. Thus, *NMF* can be viewed as a special low-rank matrix factorization.

Furthermore, if the orthogonality of the matrix factors is required, we can obtain the following *orthogonal NMF*.

$$\min_{U\geqslant 0, V\geqslant 0} \|X - UV^T\|^2, \quad s.t.\ V^TV = I \tag{2}$$

where $\|\cdot\|$ is *Frobenius* norm, i.e. $\|A\| = \sqrt{\sum_{ij} a_{ij}^2}$. Ding et al. (2005) proves that orthogonal *NMF* is equivalent to *K*-means clustering, where *V* is the cluster indicator for clustering *X*'s columns, and *U* is the set of cluster centroids. Now, we explain that in detail. Suppose $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]^T$, $\boldsymbol{v}_j = (v_{j1}, \ldots, v_{jk})^T$ and $v_{js} = \max_{m=1,\ldots,k}\{v_{jm}\}$, then the *j*th column vector of *X* belongs to the *s*th cluster and the *s*th cluster centroid is the *s*th column of *U* (i.e. each row of *V* is the cluster indicator of corresponding column of *X*, and each column of *U* is a cluster centroid).

In a recent paper Ding et al. (2006) derived the following *orthogonal nonnegative matrix tri-factorization* (*ONMTF*):

$$\min_{U\geqslant 0, S\geqslant 0, V\geqslant 0} \|X - USV^T\|^2, \quad s.t.\ U^TU = I, \quad V^TV = I \tag{3}$$

where $X \in \mathbb{R}_+^{p\times n}, U \in \mathbb{R}_+^{p\times k}, S \in \mathbb{R}_+^{k\times l}$, and $V \in \mathbb{R}_+^{n\times l}$. They showed that *ONMTF* is equivalent to the simultaneous clustering of the rows and columns of *X*. Similarly, *U* indicates which cluster every row of *X* belongs to when clustering *X*'s rows, and *V* indicates which cluster every column of *X* belongs to when clustering *X*'s columns.

The optimization problem (3) can be solved using the following update rules (Ding et al., 2006)

$$V_{jk} \leftarrow V_{jk}\sqrt{\frac{(X^TUS)_{jk}}{(VV^TX^TUS)_{jk}}} \tag{4}$$

$$U_{ik} \leftarrow U_{ik}\sqrt{\frac{(XVS^T)_{ik}}{(UU^TXVS^T)_{ik}}} \tag{5}$$

$$S_{ik} \leftarrow S_{ik}\sqrt{\frac{(U^TXV)_{ik}}{(U^TUSV^TV)_{ik}}} \tag{6}$$

Ding et al. (2006) further proved that under the update rule (6), the object function of Eq. (3): $\|X - USV^T\|^2$ is monotonically decreasing, i.e. the update rule (6) ensures that $\|X - USV^T\|^2$ can converge to a local minimum.

In the next section, we will show how to apply the *ONMTF* technique to collaborative filtering.

4. Our framework

We will first introduce some notations that will be used throughout the paper. In a typical collaborative filtering scenario, there is a $p \times n$ user-item matrix $\boldsymbol{X}$ (see Fig. 1), where p is the number of users and n is the number of items. Each element of $\boldsymbol{X}$: $x_{jm} = r$ denotes that the jth user rates the mth item by r, where $r \in \{1, \ldots, R\}$. When the item is not rated, $x_{jm} = \emptyset$.

Let

$$\boldsymbol{X} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p]^T, \quad \boldsymbol{u}_j = (x_{j1}, \ldots, x_{jn})^T, \quad j \in \{1, \ldots, p\}$$

where the vector $\boldsymbol{u}_j$ indicates the jth user's ratings to all items.

Likewise, the user-item matrix $\boldsymbol{X}$ can be decomposed into column vectors

$$\boldsymbol{X} = [\boldsymbol{i}_1, \ldots, \boldsymbol{i}_n], \quad \boldsymbol{i}_m = (x_{1m}, \ldots, x_{pm})^T, \quad m \in \{1, \ldots, n\}$$

where the vector $\boldsymbol{i}_m$ indicates all users' ratings to the mth item. The two representations of $\boldsymbol{X}$ can lead to different memory-based approaches: the former corresponds to user-based and the later corresponds to item-based.

4.1. Memory-based approaches

In our method, we choose the cosine similarity as the user similarity measure. Mathematically, the cosine similarity between the j_1th user and the j_2th user is the cosine of the angle between these two user vectors. Formally,

$$sim(\boldsymbol{u}_{j_1}, \boldsymbol{u}_{j_2}) = \frac{\sum_{m=1,\ldots,n}(x_{j_1 m})(x_{j_2 m})}{\sqrt{\sum_{m=1,\ldots,n}(x_{j_1 m})^2}\sqrt{\sum_{m=1,\ldots,n}(x_{j_2 m})^2}} \tag{7}$$

Subsequently, the test user' rating for an unknown item can be given by

$$x_{jm} = \overline{u}_j + \frac{\sum_{\boldsymbol{u}_h} sim(\boldsymbol{u}_h, \boldsymbol{u}_j)(u_{hm} - \overline{u}_h)}{\sum_{\boldsymbol{u}_h} sim(\boldsymbol{u}_h, \boldsymbol{u}_j)} \tag{8}$$

where $\overline{u}_j$ is the average rating of the jth user, and $\boldsymbol{u}_h \in \{\text{the most similar } K\text{-users of } \boldsymbol{u}_j\}$.

In addition, we select the adjusted-cosine similarity as the item similarity measure. Different from cosine-based similarity, the adjusted-cosine similarity removes the differences in rating scale between different users by subtracting the corresponding user rating average from each co-rated pair (Sarwar et al., 2001). The adjusted-cosine similarity between the m_1th item and the m_2th item is defined by

$$sim(\boldsymbol{i}_{m_1}, \boldsymbol{i}_{m_2}) = \frac{\sum_{t \in T}(x_{tm_1} - \overline{u}_t)(x_{tm_2} - \overline{u}_t)}{\sqrt{\sum_{t \in T}(x_{tm_1} - \overline{u}_t)^2}\sqrt{\sum_{t \in T}(x_{tm_2} - \overline{u}_t)^2}} \tag{9}$$

where T is the set of users who both rated m_1 and m_2, and $\overline{u}_t$ is the average rating of the tth user. The rating for an unknown item is calculated by

$$x_{jm} = \frac{\sum_{\boldsymbol{i}_h} sim(\boldsymbol{i}_h, \boldsymbol{i}_m)(x_{jh})}{\sum_{\boldsymbol{i}_h} sim(\boldsymbol{i}_h, \boldsymbol{i}_m)} \tag{10}$$

where $\boldsymbol{i}_h \in \{\text{the most similar } K\text{-items of } \boldsymbol{i}_m\}$.

	$\boldsymbol{i}_1$									$\boldsymbol{i}_n$
$\boldsymbol{u}_1$	1	?	2	?	?	1	?	?	?	?
	?	?	?	3	?	?	4	?	3	?
	?	2	?	?	1	3	?	?	?	?
	?	?	4	?	3	?	5	?	2	1
	?	1	?	?	?	?	?	2	?	?
	?	?	3	?	?	?	4	?	?	?
$\boldsymbol{u}_p$	2	4	?	1	2	?	?	3	?	?

Fig. 1. A user-item matrix. "?" means the item is not rated by the corresponding user.

4.2. Clustering techniques for collaborative filtering

In our framework, the user-item matrix $\boldsymbol{X}$ is first approximated by using orthogonal nonnegative matrix tri-factorization, so that the co-clustering of both users and items can be achieved. Suppose that $\boldsymbol{X}$ is factorized as $USV^T (U^TU = I, V^TV = I)$, where $U \in \mathbb{R}_+^{p\times k}, S \in \mathbb{R}_+^{k\times l}$, and $V \in \mathbb{R}_+^{n\times l}$. Thereinto, each row vector of U is the cluster indicator of the corresponding user vector and each row vector of SV^T denotes a user-cluster centroid. Similarly, each row vector of V is the cluster indicator of the corresponding item vector and each column vector of US denotes a item-cluster centroid.

Now we will introduce how our method identifies K nearest neighbors of test users or test items. Traditional methods usually need to search for the whole database, which will definitely suffers from the scalability problem as the growth of data. In this case, since clustering techniques can significantly reduce the search space, thus, we can apply them to large scale data. Based on clustering results, as suggested by Xue et al. (2005), neighbors' selection has two steps. Take user's neighbor selection as an example (the item's neighbor selection is similar):

(1) Compute the similarities between a test user and all the user-cluster centroids. By sorting the similarities we can take the most similar C clusters as candidates. This step is called neighbor pre-selection, which discards some irrelevant information.
(2) Choose K neighbors of the test user in the candidate set by using ordinary user-based methods.

Apparently, in the previous two steps, pre-selection directly determines the search space and the subsequent prediction.

After the K most similar neighbors of a test user or a test item having been decided, we can apply Eqs. (8) and (10) to obtain two individual predictions for an unknown rating.

4.3. Prediction fusion

Purely using user-based or item-based algorithms for collaborative filtering often gives poor predictions, especially when the user-item matrix is quite sparse. Similar to the idea in Wang et al. (2006), we provide a framework that can effectively improve the performances by fusing the results of user-based and item-based predictions.

In addition, since USV^T already makes an estimation for all unknowing ratings in the user-item matrix $\boldsymbol{X}$, we should also consider the prediction of *ONMTF* itself.

By linearly combining the previous three different types of predictions, we can obtain the final prediction result as

$$\widetilde{x}_{jm} = \lambda\widetilde{nx}_{jm} + \delta(1-\lambda)\widetilde{ux}_{jm} + (1-\delta)(1-\lambda)\widetilde{ix}_{jm} \tag{11}$$

where $\widetilde{nx}_{jm}, \widetilde{ux}_{jm}, \widetilde{ix}_{jm}$ are the prediction results of *ONMTF*, user-based and item-based, and $0 \leqslant \lambda \leqslant 1, 0 \leqslant \delta \leqslant 1$ are the fusion coefficients.

Therefore, our prediction model can be viewed as the weighted sum of three types of predictors, in which λ and δ control the weight values. Additionally, in order to give a good interpretation to linear combination, Wang et al. (2006) proposes a probabilistic fusion framework by using the independence assumption on different types of ratings and Bayes rule. Obviously, the principled probabilistic interpretation can be easily endowed with our framework.

Finally, we summarize our algorithm (called *CFONMTF*) in Table 1.

5. Experiments

5.1. Dataset

We used the *MovieLens*[1] data set to evaluate our algorithm. The *MovieLens* data set is composed of 943 users and 1682 items (1–5 scales), where each user has more than 20 ratings. For conveniently comparing with collaborative filtering algorithms listed in Wang et al. (2006) and Xue et al. (2005), we also extracted a subset which contained 500 users with more than 40 ratings and 1000 items. The first 100, 200 and 300 users in the data set are selected into three different training user sets respectively, which are denoted as *ML_100*, *ML_200* and *ML_300*. But for different training sizes, the test user set is fixed, i.e. the last 200 users. In our experiments, the available ratings of each test user are equally split into an observed set and a held out set. The observed ratings are used to predict the held out ratings.

Additionally, we randomly selected 5, 10 and 20 items rated by test users in the observed set, which were called *Given5*, *Given10*, and *Given20* respectively. For example, in *Given5*, we only select 5 items rated by test users in the observed set to predict the held out ratings.

It should be noted that before factorizing the user-item matrix X, we tried three different treatments to unknown ratings ($x_{jm} = \emptyset$): using zero, using the average rating of a user and using the average rating of a item. The results showed that using zero leads to the worst prediction performance, and the latter two approaches provide similar results of prediction. In our experiments, we replace the unknown ratings with the average rating of the corresponding user.

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

Table 1
Collaborative filtering using orthogonal nonnegative matrix tri-factorization (*CFONMTF*).

1.	The user-item matrix $\boldsymbol{X}$ is factorized as USV^T by using *ONMTF*
2.	Calculate the similarities between the test user/item and user/item-cluster centroids by using the clustering information contained in U, S, V
3.	Sort the similarities and select the most similar C user/item clusters as the test user/item neighbor candidate set
4.	Identify the most similar K neighbors of the test user/item by searching for the user/item candidate set
5.	Predict the unknown ratings by using user-based approaches (Eq. (8)) and item-based approaches (Eq. (10)) respectively
6.	Linearly combine three different predictions by *ONMTF* itself, user-based approaches and item-based approaches

5.2. Evaluation metric

There are several types of measures for evaluating the performance of collaborative filtering methods (Sarwar et al., 2001). For consistency with the experiments in Wang et al. (2006) and Xue et al. (2005), we choose the mean absolute error (*MAE*) as evaluation metric. The *MAE* is calculated by averaging the absolute deviation between predicted values and true values. Formally,

$$MAE = \frac{\sum_{j,m}|x_{jm} - \widetilde{x}_{jm}|}{N} \tag{12}$$

where N is the number of tested ratings. The lower the *MAE*, the better the performance.

5.3. Parameters tuning

5.3.1. Number of clusters

In the *ONMTF* $\boldsymbol{X} = USV^T$, the column number of U: k is the number of user clusters and the column number of V: l is the number of item clusters. Apparently, k may differ from l. But in most cases, we can set $k = l$ in order to decrease the work of parameter tuning (Ding et al., 2006). We studied how the two parameters: k and l ($k = l$) influence the performance of our fusion method.

Fig. 2 shows *MAE* of *CFONMTF* when k or l varies. We use *ML_300* dataset for training, and try 10 different values of k or l (2, 5, 10, 20, 30, 40, 50, 60, 70, 80). The best value of k or l is about 20 for *ML_300*.

As shown in Fig. 2, the number of clusters does affect the performance of our method. When the number of clusters becomes larger, the clusters information gets more specific. On the other hand, a smaller number of clusters makes the clusters information so general that some similar neighbors can be missed during the pre-selection.

5.3.2. Size of neighbors

Traditional memory-based methods need to calculate the similarities between the test user/item and all the training users/items in the whole data set to select K nearest neighbors, while the process are separated into two steps in our method

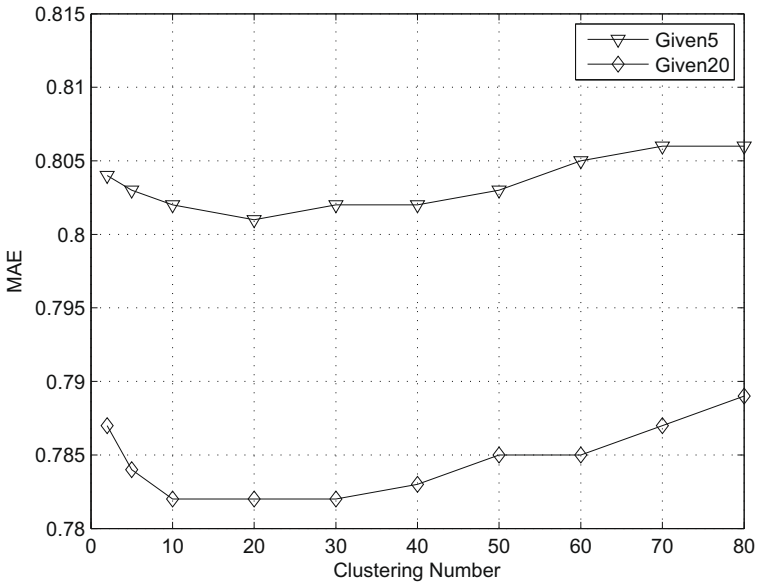

Fig. 2. Different cluster number on *ML_300*.

just as described in Section 4: first, we perform a pre-selection to obtain a subset of the whole data set as candidates; second, the K nearest neighbors are found by searching for the candidate set. The size of candidates and the number K of neighbors may affect the prediction accuracy.

The first experiment on *ML_300* shows how the percentage of pre-selected neighbors affects the performance of our method. Here we select the percentage of pre-selected neighbors rather than the number of candidate clusters C because C cannot clearly reflect the amount of pre-selected neighbors. Based on the percentage of pre-selected neighbors, the effects of the candidate neighbors' size can be analyzed much better. As indicated in Fig. 3, the performance of *CFONMTF* tends to be stable when the percentage of pre-selected neighbors reaches around 30%, which guarantees that the pre-selection can find the major similar neighbors of a test user or item.

The second experiment on *ML_300* shows the performance as a function of the size of neighbors. Here, we suppose that the size of user neighbors is equal to the size of item neighbors. Fig. 4 shows *MAE* of *CFONMTF* when the size of neighbors K varies. We observe that the optimal accuracy can be achieved when K varies between 20 and 40, so we choose 20 as the size of neighbors.

5.3.3. Combination coefficients

As shown in Eq. (11), λ and δ reflect the weights of three different models: *ONMTF*, user-based and item-based. We conducted two experiments on *ML_300* to identify the optimal combination coefficients.

First, let $\lambda = 0$ and test the property of δ. Fig. 5 indicates *MAE* of *CFONMTF* when δ varies from 0 to 1. The value of δ balances the predictions between user-based and item-based. We observe that the optimal value of δ is approximately between 0.5 and 0.7. From Fig. 5, we find that on *ML_300* the performance for user-based methods ($\delta = 1$) is better than the performance for item-based methods ($\delta = 0$), so the optimal value of δ emphasizes the user-based methods.

Second, We fix δ to be 0.6 and continue to test the property of λ. From Fig. 6, we observe that the optimal value of λ is approximately between 0.2 and 0.4. The optimal value of λ is a trade-off between *ONMTF* and memory-based approaches. On the whole, our fusion framework integrates the complementary information from three different methods: *ONMTF*, user-based and item-based. Generally speaking, the complementary information can improve the prediction accuracy. Our experiments demonstrate that the fusion method is indeed superior to any individual approach.

5.4. Scalability

Relative to traditional memory-based methods, our fusion algorithm adds a process: simultaneously clustering rows and columns of a user-item matrix (i.e. *ONMTF*), but the neighbor search space can be reduced. Consider a simple fusion scheme that just linearly combines user-based and item-based methods (Wang et al., 2006). Formally,

$$\widetilde{x}'_{jm} = \mu \widetilde{ux}'_{jm} + (1-\mu)\widetilde{ix}'_{jm} \tag{13}$$

where $0 \leqslant \mu \leqslant 1$, and $\widetilde{ux}'_{jm}$ and $\widetilde{ix}'_{jm}$ are the predictions for x_{jm} by using user-based and item-based methods respectively. The scheme is called *SF1* in Wang et al. (2006). We conducted an experiment on *ML_300* that compared the execution time of *CFONMTF* and *SF1* under the same computational environment.

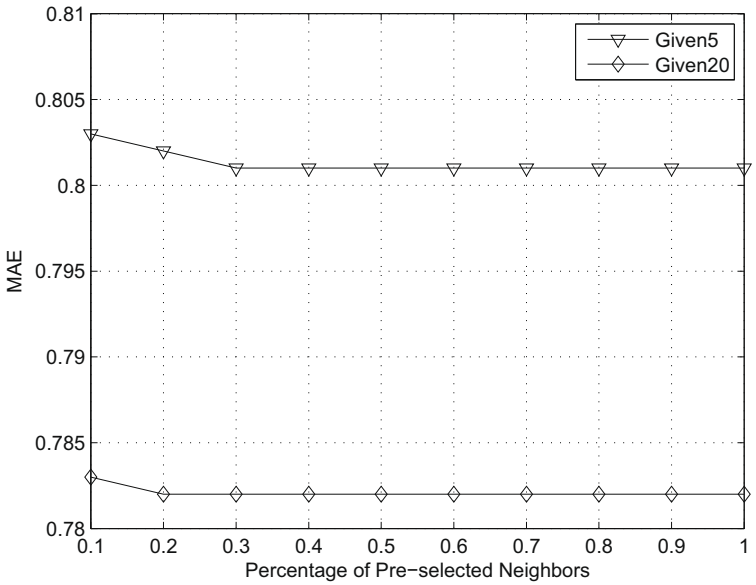

Fig. 3. Percentage of pre-selected neighbors on *ML_300*.

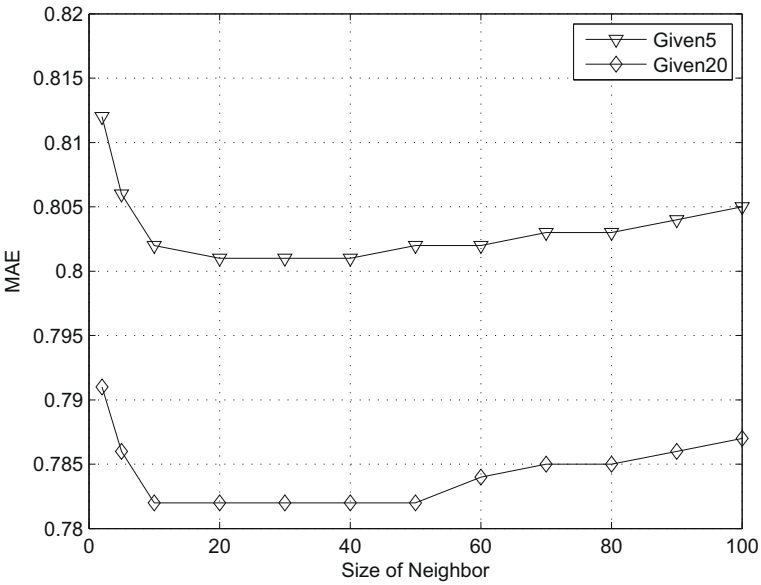

Fig. 4. Size of neighbors on *ML_300*.

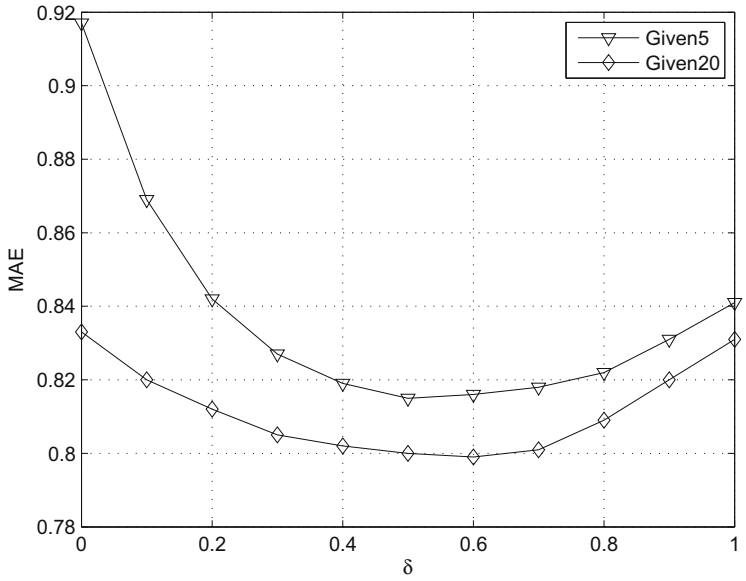

Fig. 5. δ on *ML_300*.

As shown in Fig. 8, the execution time for *CFONMTF* on a PC with a 2.0 GHz Intel PIV CPU and a 1 GB Memory is far shorter than the execution time for *SF1* when the percentage of pre-selected neighbors is 30%. This sufficiently illuminates our method's resistance to scalability. Generally, *ONMTF* in our experiments converges to the optimal value within dozens of iterative steps (Fig. 7 shows the convergence speed of *ONMTF* on *ML_300*), and its cost of execution time is much less than that of searching for the left 70% of the whole data set. Therefore, the execution time is greatly reduced.

5.5. Sparsity

Data sparsity has an important effect on the performance of collaborative filtering approaches. We did an experiment on *ML_300* that showed our algorithm's tolerance to sparsity.

Fig. 9 shows the performances of *SF1* and *CFONMTF* when the sparsity of the data set *ML_300* varies. We selected the 10%, 20%, 40%, 60%, 80%, 100% of the known ratings from the whole data set at random respectively. As shown in Fig. 9, *CFONMTF*

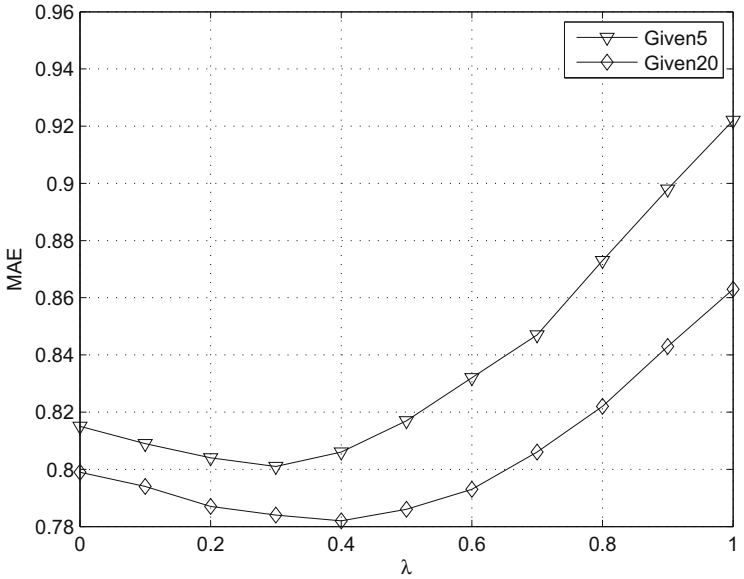

Fig. 6. λ on *ML_300*.

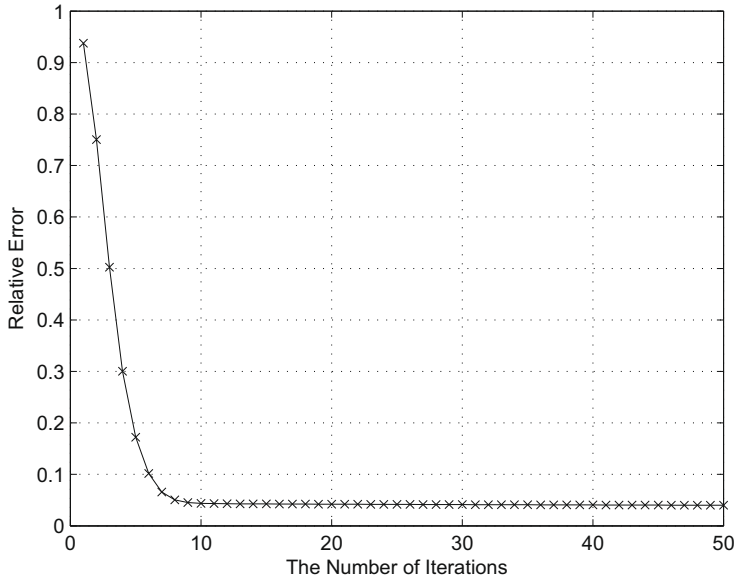

Fig. 7. The convergence speed of *ONMTF* on *ML_300*. The relative error decreases rapidly as the number of iterations increases.

outperforms *SF1* more as the rating data gets more sparse. Just as discussed by Wang et al. (2006), when the user-item matrix is quite sparse, we cannot obtain sufficient similar ratings by similar users or similar items. This results in the poor performance of collaborative filtering algorithms based on memory-based alone. In our scheme, *ONMTF* itself can be considered as a background model which provides the complementary information and improve the prediction accuracy when only sparse data is available. From the experiment we conclude that our fusion framework is quite suitable for sparse data.

5.6. Performance comparison with other methods

Finally, we compared our fusion scheme *CFONMTF* with state-of-the-art collaborative filtering algorithms, i.e. *similarity fusion* (*SF2*) (Wang et al., 2006), *cluster-based Pearson correlation coefficient* (*SCBPCC*) (Xue et al., 2005), *fast maximum margin matrix factorization* (*MMMF*) (Rennie & Srebro, 2005), *aspect model* (*AM*) (Hofmann & Puzicha, 1999), *personality diagnosis* (*PD*) (Pennock et al., 2000), *user-based Pearson correlation coefficient* (*PCC*) (Breese et al., 1998) and *cluster-based collaborative*

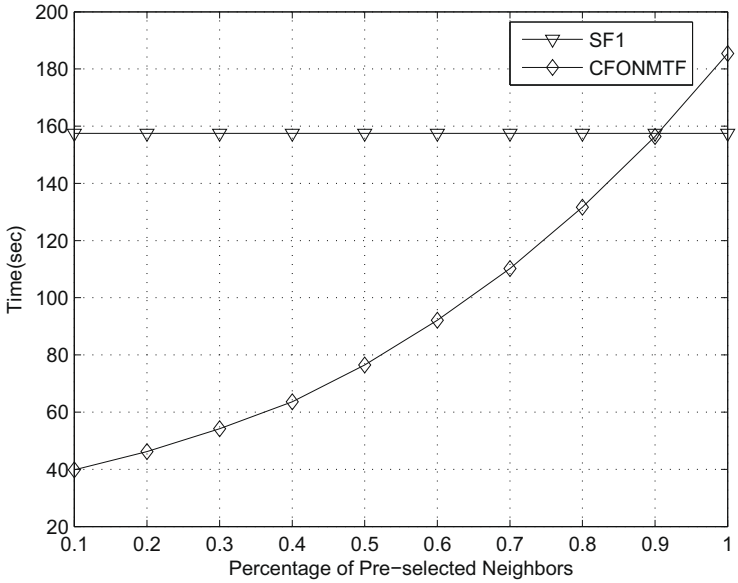

Fig. 8. Scalability on *ML_300*.

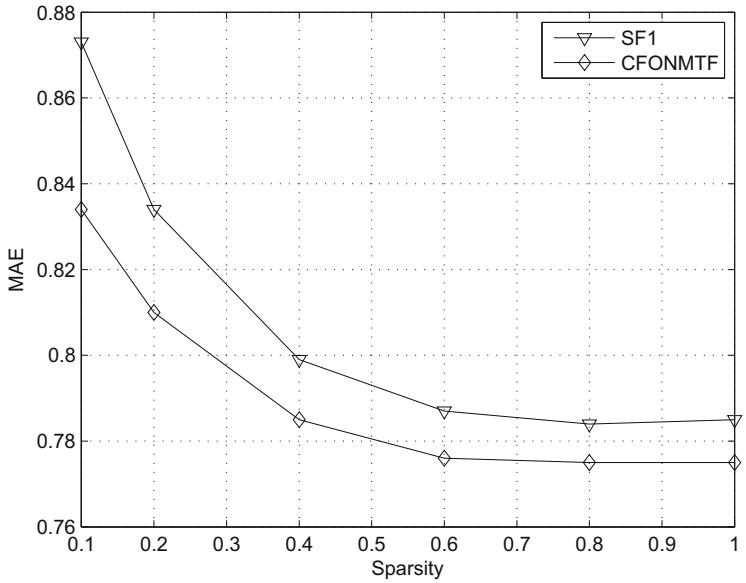

Fig. 9. Sparsity on *ML_300*.

filtering (*CBCF*) (Ungar & Foster, 1998). Table 2 shows the results of seven algorithms. Our method outperforms all the other methods in *Given5*, and is just a little worse than *SF2* in *Given10* and *Given20* (only in few cases, also a little worse than *SCBPCC* and *MMMF*). But as discussed in Section 2, *SF2* suffers from the scalability problem while our method successfully resolves it. Therefore, it can be seen that the overall performance of our approach is the best, considering the tradeoff between computation efficiency and prediction accuracy.

In summary, our algorithm *CFONMTF* includes two phases: first, it does *ONMTF* to the user-item matrix (in this phase, the active user vectors have been used); second, for an active user, our algorithm searches for its neighbors and predict the unknown ratings. Clearly, the main computational burden of *CFONMTF* lies in matrix factorization and neighbor search. *ONMTF* can be done offline and the user-item matrix can be updated periodically (just like Google updates its webpage pool). Moreover, after matrix factorization, we can also search for the neighbors of each user or each item and store them offline. Then if a user comes, we just need to draw its neighbors from the memory and predict its unknown ratings online, which takes very short time. One may argue that what will happen if the active user has no historical ratings, which is called the cold-start

Table 2
Comparison with the results reported in (Wang et al., 2006; Xue et al., 2005). A small value means a better performance.

Algorithms	*ML_100*			*ML_200*			*ML_300*		
	Given5	*Given10*	*Given20*	*Given5*	*Given10*	*Given20*	*Given5*	*Given10*	*Given20*
CFONMTF	**0.838**	0.801	0.804	**0.827**	0.791	0.787	**0.801**	0.780	0.782
SF2	0.847	**0.778**	**0.791**	0.828	**0.776**	**0.784**	0.805	**0.763**	**0.770**
SCBPCC	0.849	0.821	0.792	0.833	0.817	0.786	0.824	0.813	0.779
CBCF	0.923	0.897	0.887	0.905	0.877	0.850	0.843	0.840	0.823
MMMF	0.945	0.861	0.803	0.930	0.849	0.786	0.929	0.843	0.773
AM	0.960	0.919	0.885	0.846	0.832	0.812	0.825	0.820	0.795
PD	0.847	0.816	0.806	0.835	0.814	0.790	0.825	0.810	0.787
PCC	0.872	0.835	0.816	0.856	0.828	0.811	0.846	0.839	0.819

problem (Schein, Popescul, Ungar, & Pennock, 2001) in collaborative filtering algorithms, and our algorithm also suffers from this problem.

6. Conclusions

In this paper, we represented a novel fusion framework for collaborative filtering. The model-based approaches (i.e. clustering techniques here) and the memory-based approaches (i.e. user-based and item-based here) are naturally assembled via *ONMTF*. Our method greatly mitigates the two fundamental problems: sparsity and scalability by using the proposed hybrid technique. Empirical studies verified that our framework effectively improves the prediction accuracy of collaborative filtering and resolves the sparsity and scalability problems. In the future, we will investigate new co-clustering techniques and develop better fusion models.

Acknowledgement

This research was supported in part by the National Natural Science Foundation of China under the Grant No. 60835002.

References

Balabanovic, M., & Shoham, Y. (1997). Fab: Content-based collaborative recommendation. *Communications of the ACM, 40*(3), 66–72.
Banerjee, A., Dhillon, I. S., Ghosh, J., Merugu, S., & Modha, D. S. (2004). A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. In *Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining*. Seattle, WA, USA (pp. 509–514).
Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on uncertainty in artificial intelligence*. Madison, Wisconsin, USA (pp. 43–52).
Canny, J. (2002). Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval*. Berkeley, CA, USA (pp. 238–245).
Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*. Berkeley, California, USA.
Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems, 22*(1), 143–177.
Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*. San Francisco, California, USA (pp. 269–274).
Dhillon, I. S., Mallela, S., & Modha, D. S. (2003). Information theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*. Washington, DC, USA (pp. 89–98).
Ding, C., He, X., & Simon, H. D. (2005). On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the fifth SIAM international conference on data mining*. Newport Beach, CA, USA (pp. 606–610).
Ding, C., Li, T., Peng, W., & Park, H. (2006). Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining*. Philadelphia, PA, USA (pp. 126–135).
El-Yaniv, R., & Souroujon, O. (2001). Iterative double clustering for unsupervised and semi-supervised learning. In *Proceedings of the 12th European conference on machine learning*. Freiburg, Germany (pp. 121–132).
Fisher, D., Hildrum, K., Hong, J., Newman, M., Thomas, M., & Vuduc, R. (2000). Swami: A framework for collaborative filtering algorithm development and evaluation. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval*. Athens, Greece (pp. 366–368).
George, T., & Merugu, S. (2005). A scalable collaborative filtering framework based on co-clustering. In *Proceeds of the fifth IEEE international conference on data mining*. New Orleans, Louisiana, USA (pp. 625–628).
Giles, C. L., Bollacker, K. D., & Lawrence, S. (1998). Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on digital libraries*. Pittsburgh, Pennsylvania, USA (pp. 89–98).
Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval, 4*(2), 133–151.
Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval*. Berkeley, California, USA (pp. 230–237).
Hofmann, T., & Puzicha, J. (1999). Latent class models for collaborative filtering. In *Proceedings of the 16th international joint conference on artificial intelligence*. San Francisco, CA, USA (pp. 688–693).
Huang, Z., Chen, H., & Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems, 22*(1), 116–142.
Jin, R., Chai, J. Y., & Si, L. (2004). An automatic weighting scheme for collaborative filtering. In *Proceedings of the 27nd annual international ACM SIGIR conference on research and development in information retrieval*. Sheffield, United Kingdom (pp. 337–344).
Kohrs, A., & Merialdo, B. (1999). Clustering for collaborative filtering applications. In *Proceedings of the international conference on computational intelligence for modelling, control and automation*. Vienna, Austria (pp. 199–204).

Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* (Vol. 13, pp. 556–562).
Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*, 788–791.
Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations item-to-item collaborative filtering. *IEEE Internet Computing, 7*(1), 76–80.
Long, B., Zhang, Z., & Yu, P. S. (2005). Co-clustering by block value decomposition. In *Proceedings of the 11th ACM SIGKDD international conference on knowledge discovery and data mining*. Chicago, Illinois, USA (pp. 635–640).
Pennock, D. M., Horvitz, E., Lawrence, S., & Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the 16th conference on uncertainty in artificial intelligence*. San Francisco, CA, USA (pp. 473–480).
Popescul, A., Ungar, L. H., Pennock, D. M., & Lawrence, S. (2001). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environment. In *Proceedings of the 17th conference in uncertainty in artificial intelligence*. San Francisco, CA, USA (pp. 437–444).
Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of 22nd international conference on machine learning*. Bonn, Germany (pp. 713–719).
Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on computer supported cooperative work* (pp. 175–186). North Carolina, USA: Chapel Hill.
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender systems: A case study. In *Proceedings of ACM WebKDD web mining for e-commerce workshop*. Boston, MA, USA (pp. 82–90).
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web*. Hong Kong, China (pp. 285–295).
Schein, A., Popescul, A., Ungar, L., & Pennock, D. (2001). Generative models for cold-start recommendations. In *Proceedings of the 2001 SIGIR workshop on recommender systems*. New Orleans, USA.
Si, L., & Jin, R. (2003). Flexible mixture model for collaborative filtering. In *Proceedings of the 12th international conference on machine learning*. Washington DC, USA (pp. 704–711).
Slonim, N., & Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval*. Athens, Greece (pp. 208–215).
Tishby, N., Pereira, F., & Bialek, W. (1999). The information bottleneck method. In *Proceedings of the 37th annual Allerton conference on communication, control and computing*. Allerton House, USA (pp. 368–377).
Ungar, L. H., & Foster, D. P. (1998). Clustering methods for collaborative filtering. In *Proceedings of AAAI workshop on recommendation systems*. Menlo Park, CA (pp. 112–125).
Wang, J., de Vries, A. P. & Reinders, M. J. T. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*. Seattle, Washington, USA (pp. 501–508).
Xue, G. -R., Lin, C., Yang, Q., Xi, W., Zeng, H. -J., & Yu, Y., et al. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*. Salvador, Brazil (pp. 114–121).
Zha, H., He, X., Ding, C., Gu, M. & Simon, H. (2001). Bipartite graph partitioning and data clustering. In *Proceedings of the 10th international conference on information and knowledge management*. Atlanta, Georgia, USA (pp. 25–32).

Gang Chen is a Ph.D. Student of Department of Automation at Tsinghua University. His research interests include machine learning and it applications on data mining, information retrieval and computer vision.
Fei Wang is a Ph.D. Student of Department of Automation at Tsinghua University. His research interests include machine learning, computer vision, pattern recognition, data mining.
Changshui Zhang is a professor of Department of Automation at Tsinghua University. His research interests include pattern recognition, machine learning, artificial intelligence, computer vision, image processing, evolutional computing, complex network. He is an Associate Editor for Pattern Recognition published by Elsevier Science.