Collaborative Filtering Using Orthogonal Nonnegative Matrix Tri-factorization

Gang Chen[1], Fei Wang[1], Changshui Zhang[2]
State Key Laboratory of Intelligent Technologies and Systems
Tsinghua University
[1]{g-c05,feiwang03}@mails.thu.edu.cn,
[2]zcs@mail.thu.edu.cn

Abstract

Collaborative filtering aims at predicting a test user's ratings for new items by integrating other like-minded users' rating information. Traditional collaborative filtering methods usually suffer from two fundamental problems: sparsity and scalability. 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. Experimental results on benchmark data sets are presented to show that our algorithm is indeed more tolerant against both sparsity and scalability, and achieves good performance in the meanwhile.

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, research on collaborative filtering can be mainly categorized into two categories: *memory-based* and *model-based*.

Memory-based algorithms, including user-based [2, 12, 14, 19] and item-based [4, 20], first compute similarities between the test user and other users (user-based), or, between the test item and other items (item-based), and apply them to identify the top K most similar neighbors of the test one. 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 coverage and accuracy. More concretely, since the correlation coefficient is only defined between customers who have rated at least two products 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 procedure significantly since K nearest neighbor based algorithms will require too much computations in this case [20].

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* [2], *Bayesian network* [2], *clustering models* [21], *latent factor models* [3], *aspect models* [13], *dimension-reduction methods* [11], etc.. However, just as pointed out by [23], generating and updating of a model are often time-consuming since there are usually many free parameters to tune.

The other technique that is closely related to this paper is the co-clustering approaches, which solves the problem of simultaneously clustering rows and columns of a data matrix. [5, 24] individually developed a co-clustering approach by using the bipartite graph formulation and spectral methods, but that demanded that each row cluster was associated with a column cluster. Recently, [6] proposed a information-theoretic co-clustering technique by minimizing the loss of mutual information, and [1] 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, [8] gave a novel co-clustering approach based on nonnegative matrix factorization, and [17] also provided a similar co-clustering method called Co-clustering by Block Value Decomposition. [10] considered a novel collaborative filtering approach based on the co-clustering algorithm in [1], but that was only a model-based approach. In this

paper, we use *ONMTF* rather than the method in [10] to co-cluster, for [17] verified that the co-clustering method based on nonnegative matrix factorization was usually superior to the information-theoretic co-clustering methods in [6, 9] by empirical studies.

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*)[8]. Our algorithm first applies *ONMTF* to simultaneously cluster the rows and columns of the user-item matrix, 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. Section 2 describes orthogonal nonnegative matrix tri-factorization and its relation to clustering. In section 3, 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 4, followed by our conclusions in section 5.

2 Orthogonal Nonnegative Matrix Tri-factorization

The *Nonnegative Matrix Factorization* (*NMF*) is first brought into machine learning and data mining fields by Lee and Seung [15, 16], and now it has been widely applied in pattern recognition, multimedia, text mining and bioinformatics [8]. Recently, Ding et al. [7, 8] 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 matrix factors is required, we can obtain the following *orthogonal NMF*.

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

where $\|\cdot\|$ is *Frobenius* norm, i.e. $\|A\| = \sqrt{\sum_{ij} a_{ij}^2}$. [7] proves that orthogonal *NMF* is equivalent to K-means clustering, where V is the cluster indicator for clustering X's columns, U is the set of cluster centroids. Now, we explain that in detail. Suppose $V = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n]^T$, $\boldsymbol{v}_j = (v_{j1}, \cdots, v_{jk})^T$ and $v_{js} = \max_{m=1,\cdots,k}\{v_{jm}\}$, then the jth column vector of X belongs to the sth cluster and the sth cluster centroid is the sth 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 [8], Ding *et al.* derived the following *Orthogonal Nonnegative Matrix Tri-Factorization* (*ONMTF*):

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

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 [8]. 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 [8]

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

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

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

Ding et al. [8] 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.

3 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 $\boldsymbol{X}$, 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 $r \in \{1, \cdots, R\}$. When the item is not rated, $x_{jm} = \emptyset$.

Let

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

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

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

where the vector $\boldsymbol{i}_m$ indicates all users' ratings to the mth item.

3.1 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.

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

$$x_{jm} = \overline{u}_j + \frac{\sum_{\boldsymbol{u}_h} sim(\boldsymbol{u}_h, \boldsymbol{u}_j)(u_{hm} - \overline{u}_h)}{\sum_{\boldsymbol{u}_h} sim(\boldsymbol{u}_h, \boldsymbol{u}_j)} \quad (7)$$

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

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 [20].

The rating for an unknown item is calculated by

$$x_{jm} = \frac{\sum_{\boldsymbol{i}_h} sim(\boldsymbol{i}_h, \boldsymbol{i}_m)(x_{jh})}{\sum_{\boldsymbol{i}_h} sim(\boldsymbol{i}_h, \boldsymbol{i}_m)} \quad (8)$$

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

3.2 Clustering Techniques for Collaborative Filtering

In our framework, the user-item matrix $\boldsymbol{X}$ is first approximated by using orthogonal nonnegative matrix trifactorization, 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 the K nearest neighbors of test users or test items. Traditional methods usually need to search the whole database, which will definitely suffers from the scalability problem as the growth of data. In this case, since the clustering techniques can significantly reduce the search space, thus, we can apply them to the large scale data. Based on the clustering results, as suggested by [23], the neighbors' selection has two steps. Take the user's neighbor selection as an example (the item's neighbor selection is similar):

1. Compute the similarities between the 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 the neighbor pre-selection, which discards some irrelevant information [23].

2. Choose the K neighbors of the test user in the candidate set by using ordinary user-based methods.

Apparently, in the previous two steps, the 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 Eq.(7) and Eq.(8) to obtain two individual predictions for an unknown rating based on user and item based methods.

3.3 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 [22], 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

$$\widetilde{x}_{jm} = \lambda \widetilde{nx}_{jm} + \delta(1-\lambda)\widetilde{ux}_{jm} + (1-\delta)(1-\lambda)\widetilde{ix}_{jm} \quad (9)$$

where $\widetilde{nx}_{jm}$, $\widetilde{ux}_{jm}$, $\widetilde{ix}_{jm}$ are the prediction results of *ONMTF*, user-based, item-based, and $0 \leq \lambda \leq 1$, $0 \leq \delta \leq 1$ 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, [22] proposes a probabilistic fusion framework by using the independence assumption on different types of ratings and the Bayes rule. Obviously, the principled probabilistic interpretation can be easily endowed with our framework.

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

Algorithm 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 the user/item candidate set
5. Predict the unknown ratings by using user-based approaches (Eq. 7) and item-based approaches (Eq. 8) respectively
6. Linearly combine three different predictions by *ONMTF* itself, user-based approaches and item-based approaches

4 Experiments

4.1 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 [22, 23], 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 size, the test user set is fixed, i.e. the last 200 users. In our experiments, the available ratings of each test user are split into an observed set and a held out set. The observed ratings are used to predict the held out ratings.

Additionally, as done by [23], we randomly selected 5, 10 and 20 items rated by test users in the observed set, which were also called *Given5*, *Given10*, and *Given20* respectively.

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

4.2 Evaluation Metric

There are several types of measures for evaluating the performance of collaborative filtering methods [20]. For consistency with the experiments in [22, 23], 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,

$$MAE = \frac{\sum_{j,m} |x_{jm} - \widetilde{x}_{jm}|}{N} \tag{10}$$

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

4.3 Combination Coefficients

Due to limited space, we do not report the choices for number of clusters and size of neighbors in detail. During all the following experiments on *ML_300* we choose 20 as the number of user/item clusters, 30% as the percentage of pre-selected user/item neighbors and 20 as the size of user/item neighbors.

As shown in Eq. 9, 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. 1 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. 1, 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. 2, we observe that the optimal value of λ is about between 0.2 and 0.4. The optimal value of λ is a trade-off between *ONMTF* and memory-based approaches. Altogether, 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.

4.4 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 [22]. Formally,

$$\widetilde{x'}_{jm} = \mu\widetilde{ux'}_{jm} + (1-\mu)\widetilde{ix'}_{jm} \quad (11)$$

where $0 \leq \mu \leq 1$, $\widetilde{ux'}_{jm}$ 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 [22]. We conducted an experiment on *ML_300* that compared the execution time of *CFONMTF* and *SF1* under the same computational environment.

As shown in Fig. 3, the execution time for *CFONMTF* on a PC with a 2.0GHz Intel PIV CPU and a 1GB 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 one hundred iterative steps, which requires much less cost of execution time than searching the left 70% of the whole data set. Therefore, the execution time of our method is greatly reduced.

4.5 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. 4 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 at random respectively. As shown in Fig. 4, *CFONMTF* outperforms *SF1* more as the rating data gets sparser. Just as discussed in [22], 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.

4.6 Performance Comparison with Other Methods

Finally, we compared our fusion scheme *CFONMTF* with state-of-the-art collaborative filtering algorithms listed in [23, 22], i.e. *similarity fusion* (*SF2*) ([22]), *cluster-based Pearson Correlation Coefficient* (*SCBPCC*) ([23]), *Aspect Model* (*AM*) ([13]), *Personality Diagnosis* (*PD*) ([18]), *user-based Pearson Correlation Coefficient* (*PCC*) ([2]) and *cluster-based collaborative filtering* (*CBCF*) ([21]). Table 1 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 a few cases, also a little worse than *SCBPCC*). But as discussed in Section 2, *SF2* suffers from the scalability problem while our method successfully resolves the problem. Hereby, it can be said that the overall performance of our approach is the best, considering the balance between computation efficiency and prediction accuracy.

Table 1. Comparison with the results reported in [22, 23]. A small value means a better performance

Training Set	Algorithms	*Given5*	*Give10*	*Give20*
ML_100	*CFONMTF*	**0.838**	**0.801**	**0.804**
	SF2	0.847	0.774	0.792
	SCBPCC	0.848	0.819	0.789
	CBCF	0.924	0.896	0.890
	AM	0.963	0.922	0.887
	PD	0.849	0.817	0.808
	PCC	0.874	0.836	0.818
ML_200	*CFONMTF*	**0.827**	**0.791**	**0.787**
	SF2	0.827	0.773	0.783
	SCBPCC	0.831	0.813	0.784
	CBCF	0.908	0.879	0.852
	AM	0.849	0.837	0.815
	PD	0.836	0.815	0.792
	PCC	0.859	0.829	0.813
ML_300	*CFONMTF*	**0.801**	**0.780**	**0.782**
	SF2	0.804	0.761	0.769
	SCBPCC	0.822	0.810	0.778
	CBCF	0.847	0.846	0.821
	AM	0.820	0.822	0.796
	PD	0.827	0.815	0.789
	PCC	0.849	0.841	0.820

5 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 further investigate the new co-clustering techniques, find a method

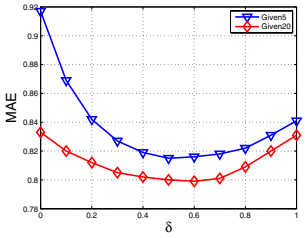

Figure 1. δ **on** *ML_300*

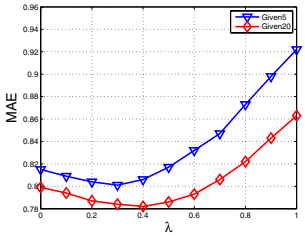

Figure 2. λ **on** *ML_300*

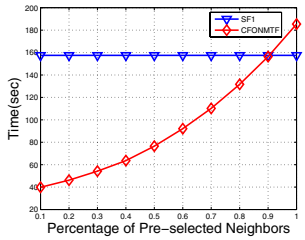

Figure 3. Scalability on *ML_300*

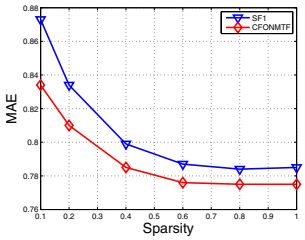

Figure 4. Sparsity on *ML_300*

that can automatically solve the fusion coefficients and develop other better fusion models.

References

[1] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proc. of SIGKDD*, 2004.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI*, 1998.

[3] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. of SIGIR*, 1999.

[4] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[5] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. of SIGKDD*, 2001.

[6] I. S. Dhillon, S. Mallela, and D. S. Modha. Information theoretic co-clustering. In *Proc. of SIGKDD*, 2003.

[7] C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proc. SIAM Data Mining Conf.*, 2005.

[8] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proc. of SIGKDD*, 2006.

[9] R. El-Yaniv and O. Souroujon. Iterative double clustering for unsupervised and semi-supervised learning. In *Proc. of ECML*, 2001.

[10] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proc. of the Fifth IEEE International Conference on Data Mining*, 2005.

[11] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[12] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of SIGIR*, 1999.

[13] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of IJCAI*, 1999.

[14] R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *Proc. of SIGIR*, 2004.

[15] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[16] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. In *Advances in Neural Information Processing Systems.*, 2001.

[17] B. Long, Z. Zhang, and P. S. Yu. Co-clustering by block value decomposition. In *Proc. of SIGKDD*, 2005.

[18] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: a hybird memory- and model-based approach. In *Proc. of UAI*, 2000.

[19] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. of ACM CSCW*, 1994.

[20] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the WWW Conference*, 2001.

[21] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Proc. Workshop on Recommendation Systems at AAAI*, Menlo Park, CA, 1998. AAAI Press.

[22] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proc. of SIGIR*, 2006.

[23] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of SIGIR*, 2005.

[24] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Bipartite graph partitioning and data clustering. In *Proc. of CIKM*, 2001.