MFCC: An Efficient and Effective Matrix Factorization Model Based on Co-clustering

Wenjuan Yang(✉), Le Wu, Xueliang Liu, and Chunxiao Fan

Hefei University of Technology, Hefei, China
xiaobie.hfut@gmail.com

Abstract. Collaborative Filtering (CF) is a popular way to build recommender systems and has been widely deployed by many e-commerce websites. Generally, there are two parallel research directions on CF, one is to improve the prediction accuracy $\sim$ (i.e., effectiveness) of CF algorithms and others focus on reducing time cost of CF algorithms $\sim$ (i.e., efficiency). Nevertheless, the problem of how to combine the complementary advantages of these two directions, and design a CF algorithm that is both effective and efficient remains pretty much open. To this end, in this paper, we provide a Matrix Factorization based on Co-Clustering (MFCC) algorithm to address the problem. Specifically, we first adopt a co-clustering algorithm to cluster the user-item rating matrix into several separate sub rating matrices. After that, we provide an efficient matrix factorization algorithm by utilizing the strong connections of users and items in each cluster. In the meantime, this process is also efficient as we can simultaneously compute the matrix factorization for each cluster as there exists little interactions among different clusters. Finally, the experimental results show both the effectiveness and efficiency of our proposed model.

Keywords: Co-clustering · Matrix factorization · Collaborative filtering

1 Introduction

The rapid development of Internet has brought massive information to users for their different types of demand in the information age [7, 10, 13, 25, 26]. However, it brings a new issue called the problem of information overload, which causes difficulties for users to choose their individual information and correspondingly reduces the utilization rate. To solve the problem, the recommender system begins to emerge [1, 2].

Collaborative filtering (CF) is one of the most widely used algorithms for many e-commerce sites in recommender systems [1–3] until now. It identifies users whose tastes are similar to those of the active users. Then it recommends items that those users have liked. And many online companies such as Yahoo and Amazon apply CF to provide recommendations to their customers [2]. In particular, CF mainly includes two types of algorithms: the neighbor-based CF [3] and model-based CF [4]. The neighbor-based CF algorithm is simple and easy to interpret. Nevertheless, these neighbor-based CF suffers from high time complexity and relatively low accuracy as it is hard for them to find reliable neighbors when the user-item rating matrix is sparse. On the contrary, the latent based models show good accuracies from both research and industry area.

The core idea of these latent factor models is to project both users and items in the low latent space, then the predicted rating can be computed in this latent space [4, 9]. Despite the success of these latent based models, in the real world, since there are millions of users and items in the recommender systems, the time efficiency of these CF algorithms needs to be further improved. Hence, how to improve these CF algorithms' efficiency turns to another research problem.

In fact, in order to solve the above problem, many researchers have integrated clustering methods into recommender systems [6, 8]. As a result, these algorithms can cluster the original user-item rating matrix into several sub small matrices, then the traditional CF algorithms can be applied in these small matrices. However, as reported by the current research results, the clustering stage can reduce the effectiveness to some degree.

In this paper, we study the problem of how to improve both efficiency and effectiveness of the CF algorithms, and propose a novel Matrix Factorization based on Co-Clustering (MFCC) algorithm to solve this problem. The core idea of this method is that it utilizes the properties of the clustering algorithm into the CF algorithm. Specifically, we first group the original user-item rating matrix into multiple small sub rating matrices by a co-clustering algorithm. Due to the properties of each sub matrix, the users and items in this matrix are strongly connected. We push these strong connections in each sub matrix into the latent models, and ensure the factorization of each user and item in each cluster are similar. As a result, the time complexity of the training phase is significantly reduced, and the effectiveness of the final prediction is guaranteed. Experiments on a publicly available dataset have demonstrated both the efficiency and effectiveness of the proposed MFCC.

The remainder of this paper is constructed as follows: we first introduce the related works on recommender systems, description of problem definitions and related knowledge in second part. In Sect. 3, we introduce the framework of this paper and the related description. Our experiments and conclusion are illustrated in the Sects. 4 and 5 separately.

2 Related Work

As the most popular technique in recommender systems, collaborative filtering (CF) has received a great success in various applications [2, 23, 24]. Specifically, CF mainly includes the neighbor-based CF and the model-based CF algorithm, moreover, the model-based CF has a better performance. Probabilistic Matrix Factorization (PMF) [9] is a classical model-based CF, which can not only handle large-scale datasets quickly, but also obtain a reasonable accuracy. So many algorithms are developed to further improve its effectiveness. For example, the NHPMF [5] algorithm significantly enhanced the accuracy than PMF.

Although the model-based CF has been widely studied, with the rapid increase of data size, the algorithm tends to be inefficient to meet the requirement of real-world applications. Applying clustering to CF [6, 19] is mainly due to the following reasons. Firstly, the sizes of generated classes by clustering are reduced. Secondly, the clustering can also reduce the scarcity of ratings. At first, many scholars only considered the application of clustering to items [6, 11] or users [12] to improve the efficiency.

However, these methods only consider one dimension of the matrix information and lost the other one. To address this issue, the co-clustering based on CF approaches are proposed [8]. Compared with the classical clustering methods, the co-clustering can effectively find the hidden clustering structure of the user-item rating matrix and cluster the above two dimensions at the same time.

Earlier, a co-clustering algorithm for information theory [14] was proposed. Later, Agarwal [15] developed a method by utilizing a generalized linear model to smooth the error function. Subsequently, several researchers suggested how to set the reference standard of the number of cluster categories [16]. But all these methods are hard clustering [17], namely, each user, item and rating only belongs to a single cluster. Therefore, some scholars proposed to employ the fuzzy clustering [18, 20, 21] to relax the restrictions on the attribution of categories.

As we can see, although model-based algorithms can obtain high accuracies, they are insufficiently efficient. And co-clustering algorithms can accelerate the process of handling large-scale datasets. However, they will sacrifice the accuracy. In this paper, we propose to combine the matrix factorization and co-clustering and present a method called MFCC, where the efficiency of the algorithm is improved and a reasonable accuracy is insured.

3 Description of the Recommendation Algorithm

As shown in Fig. 1, the overall flow chart of MFCC includes two main steps: co-clustering and rating training. The first step is to divide the user-item rating matrix into several small ones. The second step is used to predict the unknown ratings of these small matrices with matrix factorization in parallel. Based on the above strategy, we can recommend items based on the obtained ratings.

3.1 The Proposed Algorithm

The mathematical notations used in co-clustering are shown in Table 1. As mentioned above, the co-clustering algorithm groups the original matrix into several small clusters, and each small cluster is closely related. According to the close relations, we can obtain less computation and higher accuracy in the second step.

Suppose we want to divide the user-item rating matrix into small ones. Different from classical clustering approaches, in each iteration, co-clustering will first cluster all the users, items, and ratings respectively and assign them probabilities, namely, one for each cluster. Then the co-clustering integrates these obtained soft assignments to improve the next round of clustering. The above process will repeat until it is converged.

In particular, suppose is the probability that the user, item and rating will be assigned to the cluster. According to the co-clustering, we can formulate it as

$$p(k|u,v,r) = \frac{[p(k|u)+a] \times [p(k|v)+b] \times [p(r|k)+c]}{\sum_{k' \in K} [p(k'|u)+a] \times [p(k'|v)+b] \times [p(r|k')+c]} \tag{1}$$

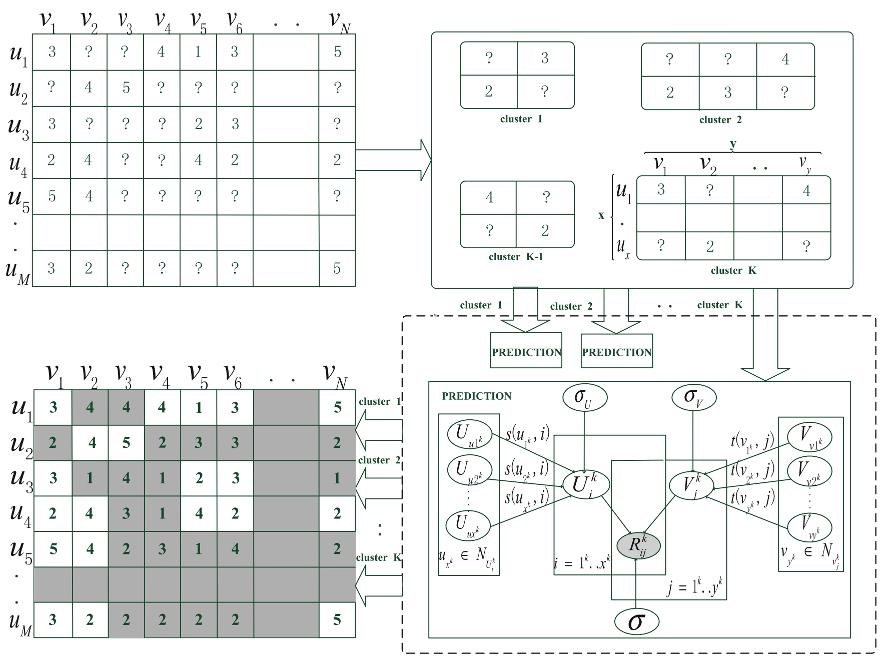

Fig. 1. Framework of MFCC recommendation

Table 1. Notations used in co-clustering.

Notations	Description
U, u	User sets, the current user
V, v	Item sets, the current item
K, k	Cluster sets, the current cluster
R, r	Rating matrix, the current rating
$p(k\|u)$	Probability of cluster given user
$p(k\|v)$	Probability of cluster given item
$p(k\|u, v, r)$	Probability of given user, item and rating

where $p(k|u)$, $p(k|v)$ and $p(r|k)$ denotes the probability that each element will be assigned to the cluster k. In practice, we set a, b, c to 0.00000001 to avoid the denominator from being zero. Meanwhile, the above mentioned probabilities can be estimated as:

$$p(k|u) = \frac{\sum_{v \in V(u)} p(k|u, v, r)}{\sum_{k' \in K} \sum_{v \in V(u)} p(k'|u, v, r)} \tag{2}$$

$$p(k|v) = \frac{\sum_{u \in U(v)} p(k|u, v, r)}{\sum_{k' \in K} \sum_{u \in U(v)} p(k'|u, v, r)} \tag{3}$$

$$p(r|k) = \frac{\sum p(k|u, v, r)}{\sum_{r'} \sum p(k|u, v, r')} \tag{4}$$

According to Eqs. (2), (3) and (4), when the above co-clustering process is converged by an iterative method, the elements in each cluster are neighbors, which constitute a neighbor set. Meanwhile, since the above cluster assignment is soft, one user may belong to several clusters. We simply assign the user to the cluster with the maximum probability. Then the user-item rating matrix is divided into K clusters, so that we can do parallel computing in each cluster. Moreover, the user cluster is set to C and the item cluster is set to D.

After excavating the mutual influence relations between users and items by co-clustering, in the training stage, we take cluster k as an example. As shown in Fig. 1, x^k represents the number of user cluster C_i in cluster k, and y^k denotes the number of items cluster D_j in cluster k. The model ensures that user i behaves like its neighbor set C_i and item j is similar to its neighbor set D_j. Based on this, the equations proposed are as follows:

$$U_i^k = \sum_{a \in C_i} s(i, a) * U_i^k + \phi_{U^k}, \quad \phi_{U^k} \sim N(0, \sigma_U^2 \omega) \tag{5}$$

$$V_j^k = \sum_{a \in D_j} t(j, a) * V_j^k + \phi_{V^k}, \quad \phi_{V^k} \sim N(0, \sigma_V^2 \omega) \tag{6}$$

From the above two formulas, we can see that the latent feature vectors of each user (item) consist of two terms. The first term is the weighted average of the user's (item's) neighbors, where s represents the similarity between user i and user a, and t represents the similarity between item j and item a. The second term is the divergence between each user and item parameterized by the variance σ_U^2 and σ_V^2. It is clear that Eqs. (5) and (6) can be transformed into the following formulas when the variance is zero:

$$p(U^k|S, \sigma_U^2) = \prod_{i=1^k}^{x^k} N(\sum_{a \in C_i} s(i, a) * U_a^k, \sigma_U^2 \omega) \tag{7}$$

$$p(V^k|T, \sigma_V^2) = \prod_{j=1^k}^{y^k} N(\sum_{a \in D_j} t(j, a) * V_a^k, \sigma_V^2 \omega) \tag{8}$$

Through the above two equations and the observed rating data, we define the following expression:

$$p(R^k|U^k, V^k, \sigma^2) = \prod_{i=1^k}^{x^k} \prod_{j=1^k}^{y^k} [N(R_{ij}^k|U_i^{kT} V_j^k, \sigma^2)]^{\omega_{ij}} \tag{9}$$

where ω_{ij} is an indicator function that is equal to 1 if user i rated item j and equal to 0 otherwise. According to Eqs. (7), (8) and (9), the following equation can be obtained from the Bayesian inference:

$$p(U^k, V^k|R^k, \sigma^2, \sigma_U^2, \sigma_V^2) \propto p(U^k|S, \sigma_U^2) * p(V^k|T, \sigma_V^2) * p(R^k|U^k, V^k, \sigma^2) \tag{10}$$

Fixing the hyperparameters $(\sigma^2, \sigma_U^2, \sigma_V^2)$, the maximization of Eq. (10) is equivalent to minimizing the following cost function:

$$\begin{aligned} E^k = & \frac{1}{2}\sum\nolimits_{i=1^k}^{x^k}\sum\nolimits_{j=1^k}^{y^k}\omega_{ij}(R_{ij}^k - U_i^{kT}V_j^k)^2 \\ & + \frac{1}{2}\lambda_U\sum\nolimits_{i=1^k}^{x^k}||U_i^k - \sum\nolimits_{a\in C_i}s(i,a)*U_a^k||_F^2 \\ & + \frac{1}{2}\lambda_V\sum\nolimits_{j=1^k}^{y^k}||V_j^k - \sum\nolimits_{a\in D_j}t(j,a)*V_a^k||_F^2 \end{aligned} \tag{11}$$

In the above equation, $\lambda_U = \sigma^2/\sigma_U^2, \lambda_V = \sigma^2/\sigma_V^2$. Obviously, it consists of three parts. The first is the relations between the actual ratings and the predicted ratings. The following two terms are the neighbors information and they are smoothed by the parameter λ_U and λ_V. What's more, the parameter λ_U controls how much the user neighbor influences while λ_V controls how much the item neighbor influences on the cost function. In order to reach the minimum value of Eq. (11), we use the random gradient descent method on U_i^k and V_j^k for each user and item. So we give the functions as follows:

$$\begin{aligned} \frac{\partial E^k}{\partial U_i^k} = & \sum\nolimits_{j=1^k}^{y^k}\left(R_{ij}^k - U_i^{kT}V_j^k\right)\left(-V_j^k\right) + \lambda_U\left(U_i^k - \sum\nolimits_{a\in C_i}s(i,a)*U_a^k\right) \\ & - \lambda_U\sum\nolimits_{i\in C_a}s(a,i)\left(U_a^k - \sum\nolimits_{j\in C_a}s(j,a)*U_j^k\right) \end{aligned} \tag{12}$$

$$\begin{aligned} \frac{\partial E^k}{\partial V_j^k} = & \sum\nolimits_{i=1^k}^{x^k}\left(R_{ij}^k - U_i^{kT}V_j^k\right)\left(-U_i^k\right) + \lambda_V\left(V_j^k - \sum\nolimits_{a\in D_j}t(j,a)*V_a^k\right) \\ & - \lambda_V\sum\nolimits_{j\in D_a}t(a,j)\left(V_a^k - \sum\nolimits_{i\in D_a}t(i,a)*V_i^k\right) \end{aligned} \tag{13}$$

3.2 Time Complexity Analysis

In this paper, the calculation process mainly includes three parts. For the first part of co-clustering, the time complexity is $O(iter1 \times L \times K)$, where $iter1$ is the number of iterations, generally within 20, L denotes non-zero values of rating matrix, and K means the number of clusters. The time complexity of computing users' and items' similarity is $O(M^2 + N^2)$. The third part is the time complexity of training. Since we first do co-clustering, we can perform the parallel computation of each cluster in the training stage. Then the time complexity of the partial derivative of the user is

$O(iter2 \times (L \times D + xMD)/K)$, and according to Eq. (13), the time complexity of the item cluster can be computed as $O(iter2 \times (L \times D + yND)/K)$. Thus in total, the time complexity is $O(iter2 \times (L \times D + xMD + yND)/K)$, where $iter2$ denotes the number of iterations on the training stage, D denotes the dimension, x denotes the number of each user cluster, and y denotes the number of each item cluster.

3.3 Discussion on MFCC

The MFCC algorithm incorporates the co-clustering into matrix factorization for recommendation. Theoretically, the model unifies advantages of the two algorithms. So we analyze the model from two aspects, efficiency and effectiveness. On one hand, the algorithm first splits the user-item rating matrix into K small clusters, and then each cluster of users (items) is equivalent to a neighbor set. When we do parallel computing on each cluster, the time complexity is reduced to $1/K$ of the training stage without co-clustering. On the other hand, MFCC can preserve a comparable accuracy even though it deals with large-scale datasets. It is significant in real-world applications.

4 Experiments

4.1 Dataset

We choose the considerably classical MovieLens 10M dataset (http://www.movielens.org) frequently used in recommender systems for evaluation. The dataset selected in this paper contains 71567 users of 10000054 rating records (0.5 to 5) for 10,681 movies, including 95580 tags.

4.2 Evaluation

In this paper, root mean square error (RMSE) is used to measure the performance of rating prediction [22]. Specifically, with smaller RMSE, the prediction accuracy is higher. Assuming that the rating vector of N movies is expressed as $\{p_1, \ldots, p_N\}$ and the corresponding actual rating vector is $\{r_1, \ldots, r_N\}$, then the RMSE of the algorithm is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (p_i - r_i)^2}{N}} \tag{14}$$

4.3 Comparative Methods

In this paper, we compare our proposed method with the following four baselines:

(1) PMF: It is a classical matrix factorization algorithm for recommendation [9]. Specifically, this algorithm models the rating matrix as a product of two lower-rank user and movie matrices. Then it recommends movies by the two matrices. Although the algorithm obtains a favorable efficiency, its effectiveness has room for improvement.

(2) Co-Clustering: This algorithm [19] splits the user-item rating matrix into several small matrices, and in each cluster, the elements are similar. Then the scholars utilize WNMF (weighted non-negative matrix factorization) to predict the unknown ratings. The method has some advantages for dealing with large-scale datasets, but its effectiveness is relatively poor.
(3) Co-Clustering + PMF (CCPMF): The method mainly includes two stages. The first stage is to cluster the user-item rating matrix to several sub rating matrices. In the second stage, we use PMF to get user and item matrices in each cluster, then we can know the missing values by the two matrices. As CCPMF is the combination of Co-Clustering and PMF, it can improve effectiveness less.
(4) NHPMF: To improve the accuracy of recommendations, NHPMF [5] uses extra information to select neighbors of each user and each item, then it does matrix factorization on each user's and item's latent feature vector. The proposed method has good performance on effectiveness, but the time cost can be much less.

As we can see from the above four baselines, NHPMF uses external information to improve accuracy. To be fair, similar as the experimental setup in NHPMF [5], we explore the external data source such as tag information in our experiments. Hence, we remove the tags with less than five different users and movies. For each user and movie, less than five different tags are also deleted. Finally, the dataset contains 447 users and 2335 items with 148183 ratings and 1389 tags.

4.4 Experimental Results and Analysis

4.4.1 The Effect of Parameter λ on the Algorithm

The experiments first observe the effect of parameter λ on MFCC's accuracy. The dimension D is set to be 10 and 25, $\lambda_U = \lambda_V = \lambda$, where λ indicates the degree that the user (item) is affected by its neighbors. Figure 2 shows that λ has a great impact on RMSE. As λ increases, the algorithm's accuracy is improved. But when λ surpasses 15, the algorithm's accuracy decreases. This indicates that λ is too large to lead to overfitting, thus we set $\lambda = 10$ in subsequent experiments. In the course of the experiment, as λ increases, the number of iterations increases when RMSE reaches the minimum. In other words, its increase slows down the convergence of RMSE to the minimum. The experimental results are shown in Fig. 3.

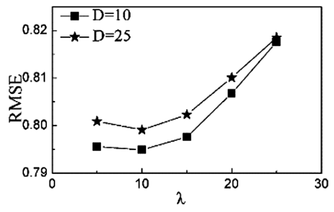

Fig. 2. λ on the impact of RMSE

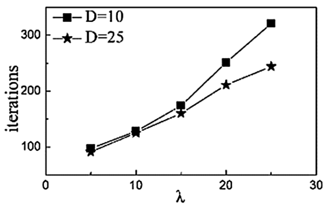

Fig. 3. λ on the impact of iterations

4.4.2 Comparative Evaluation of Different Methods

In Table 2, we compare MFCC with other CF algorithms by setting the feature vectors' dimension D to 5, 10, 15, 20, and 25 respectively.

The above experimental results demonstrate that MFCC outperforms PMF and Co-Clustering in terms of RMSE. This is mainly because the relations modeled by PMF and Co-Clustering are simpler. Moreover, MFCC uses the validity of the information such as tag information and neighbor relations to improve the accuracy. Owing to the similar reasons, the accuracy of CCPMF is much lower than MFCC's. In addition, MFCC achieves very close performance compared with NHPMF. Since the error of MFCC in the co-clustering stage leads to the improvement of the unreliability of neighbor sets, which affects the accuracy of training stage. However, as shown in Table 4, MFCC is more effective than NHPMF. Considering the big data, it can be applied in a wide range of fields and be more feasible in the practical application.

Table 2. RMSE comparisons for different latent feature dimension D

Models	D				
	5	0	15	20	25
PMF	0.8027	0.8538	0.8863	0.9021	0.9540
NHPMF	0.7878	0.7771	0.7747	0.7722	0.7705
Co-clustering	1.0901	1.0440	1.0258	1.0102	1.0095
CCPMF	1.0126	1.0117	1.0134	1.0032	1.0043
MFCC	0.7996	0.7991	0.7975	0.7965	0.7949

4.4.3 Algorithms' Run Time Comparison

In this section, we experimentally compare the time it takes to update the algorithm at each iteration. The experimental environment for the operation is Intel Core i5 CPU, 3.00 GHZ frequency, Windows10 system, 12 GB memory.

(1) We calculate the correlation between the number of clusters and the training time. The results are shown in Table 3.

As can be seen from Table 3, the time shows a rapid growth trend with the increasing number of clusters. This is mainly because the time complexity of the co-clustering phase is $O(iter1 \times L \times K)$, and the time is linearly related to the number of clusters K.

Table 3. Time on the impact of K

Number of clusters	Time(s)
K = 1	5.81
K = 5	8.38
K = 10	11.34
K = 20	17.11
K = 50	33.21

Table 4. Time on the impact of models

Models	Time(s)
PMF	0.14
NHPMF	5.61
Co-clustering	2.10
CCPMF	0.29
MFCC	0.13

(2) We set $\lambda_U = \lambda_V = \lambda$, and when D is 10, the run time comparison results of each algorithm are as follows:

From the experimental results, we can see that PMF and MFCC are more efficient, because PMF doesn't take into account neighbor relations, and the time complexity of PMF is only relevant to the amount of non-zero rating entries. NHPMF not only considers the relations between neighbors, but also joins the tag information, which results in a surprising time complexity. MFCC is theoretically the fusion of two algorithms, so the time complexity will be higher. But because of the parallel computing, it runs at a speed of more than 40 times than NHPMF algorithm with a comparable accuracy at the same time. Obviously, this effect is considerable.

5 Conclusions

In this paper, we propose the MFCC model to improve the time efficiency based on maintaining a comparable accuracy. In the proposed method, we utilize co-clustering into matrix factorization, and the method combines advantages of the two algorithms. Moreover, the experimental results on the MovieLens dataset show our method outperforms many typical recommendation algorithms.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China (NSFC) under grants 61472116 and 61502139, Natural Science Foundation of Anhui Province under grant 1608085MF128 and 1708085QF155, and the Open Projects Program of National Laboratory of Pattern Recognition under grant 201600006 and 201700017.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE TKDE **17**, 734–749 (2005)
2. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**, 76–80 (2003)
3. Sarwar, B., Karypis, G., Konstan, J., et al.: Item-based collaborative filtering recommendation algorithms. In: WWW, pp. 285–295 (2001)
4. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**, 30–37 (2009)

5. Wu, L., Chen, E., Liu, Q., et al.: Leveraging tagging for neighborhood-aware probabilistic matrix factorization. In: CIKM, pp. 1854–1858 (2012)
6. Najafabadi, M.K., Mahrin, M.N., Chuprat, S., et al.: Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data. Comput. Hum. Behav. **67**, 113–128 (2017)
7. Hong, R., Hu, Z., Wang, R., Wang, M., Tao, D.: Multi-view object retrieval via multi-scale topic models. IEEE Trans. Image Process. **25**, 5814–5827 (2016)
8. Wu, Y., Liu, X., Xie, M., et al.: Improving collaborative filtering via scalable user-item co-clustering. In: WSDM, pp. 73–82 (2016)
9. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. In: NIPS, pp. 1257–1264 (2007)
10. Zhang, H., Shen, F., Liu, W., He, X., Luan, H., Chua, T.-S.: Discrete collaborative filtering. In: SIGIR, pp. 325–334 (2016)
11. Wang, Z., Wang, X., Qian, H.: Item type based collaborative algorithm. In: CSO, pp. 387–390 (2010)
12. Shi, X.Y., Ye, H.W., Gong, S.J.: A personalized recommender integrating item-based and user-based collaborative filtering. In: ISBIM, pp. 264–267 (2008)
13. Zhang, H., Zha, Z.-J., Yang, Y., Yan, S., Chua, T.-S.: Robust semi nonnegative graph embedding. IEEE Trans. Image Process. **23**, 2996–3012 (2014)
14. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: KDD, pp. 89–98 (2003)
15. Agarwal, D., Merugu, S.: Predictive discrete latent factor models for large scale dyadic data. In: SIGKDD, pp. 26–35 (2007)
16. Xiao-Guang, L., Ge, Y., Da-Ling, W., et al.: Latent concept extraction and text clustering based on information theory. JSW, 2276–2284 (2008)
17. Geiger, B.C., Amjad, R.A.: Hard Clusters Maximize Mutual Information (2016)
18. Hu, L., Chan, K.C.C.: Fuzzy clustering in a complex network based on content relevance and link structures. TFS **24**, 456–470 (2016)
19. Hu, W.U., Wang, Y.J., Wang, Z., et al.: Two-phase collaborative filtering algorithm based on co-clustering. JSW **21**, 1042–1054 (2010)
20. Mei, J.P., Wang, Y., Chen, L., et al.: Large scale document categorization with fuzzy clustering. TFS **25**, 1239–1251 (2016)
21. Bu, J., Shen, X., Xu, B., et al.: Improving collaborative recommendation via user-item subgroups. TKDE **28**, 2363–2375 (2016)
22. Chai, T., Draxler, R.R.: Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature. GMD **7**, 1525–1534 (2014)
23. Liu, Q., Ge, Y., Li, Z., et al.: Personalized travel package recommendation. In: IEEE ICDM, pp. 407–416 (2011)
24. Wu, L., Ge, Y., Liu, Q., et al.: Modeling the evolution of users' preferences and social links in social networking services. IEEE TKDE **29**, 1240–1253 (2017)
25. Hong, R., Zhang, L., Zhang, C., Zimmermann, R.: Flickr circles: aesthetic tendency discovery by multi-view regularized topic modeling. IEEE Trans. Multimed. **18**, 1555–1567 (2016)
26. Wu, L., Liu, Q., Chen, E., Yuan, N.J., Guo, G., Xie, X.: Relevance meets coverage: a unified framework to generate diversified recommendations. ACM TIST **7**, 39 (2016)