An Exploration of Improving Collaborative Recommender Systems via User-Item Subgroups

Bin Xu Jiajun Bu Chun Chen
Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science
Zhejiang University, Hangzhou, China
xbzju,bjj,chenc@zju.edu.cn

Deng Cai
State Key Lab of CAD&CG
College of Computer Science
Zhejiang University, Hangzhou, China
dengcai@cad.zju.edu.cn

ABSTRACT

Collaborative filtering (CF) is one of the most successful recommendation approaches. It typically associates a user with a group of like-minded users based on their preferences over all the items, and recommends to the user those items enjoyed by others in the group. However we find that two users with similar tastes on one item subset may have totally different tastes on another set. In other words, there exist many user-item subgroups each consisting of a subset of items and a group of like-minded users on these items. It is more natural to make preference predictions for a user via the correlated subgroups than the entire user-item matrix. In this paper, to find meaningful subgroups, we formulate the Multiclass Co-Clustering (MCoC) problem and propose an effective solution to it. Then we propose an unified framework to extend the traditional CF algorithms by utilizing the subgroups information for improving their top-N recommendation performance. Our approach can be seen as an extension of traditional clustering CF models. Systematic experiments on three real world data sets have demonstrated the effectiveness of our proposed approach.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithms, Performance

Keywords

Collaborative Filtering, Recommender Systems, User-Item Subgroups, Clustering Model

1. INTRODUCTION

Recommender systems have been indispensable nowadays due to the incredible increasing of information in the world, especially on the Web. These systems apply knowledge discovery techniques to make personalized recommendations that can help people sift through huge amount of available articles, movies, music, webpages, etc. Popular examples of such systems include product recommendation in Amazon[1], music recommendation in Last.fm[2], and movie recommendation in Movielens[3].

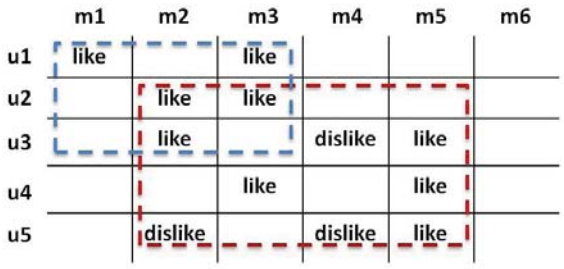


Figure 1: A toy example of the user-item matrix and subgroups. Note that users and items are not necessarily adjacent in a subgroup.

Collaborative filtering (CF) [1, 14, 30] is one of the most widely adopted and successful recommendation approaches. Unlike many content-based approaches which utilize the attributes of users and items, CF approaches make predictions by using only the user-item interaction information. These methods can capture the hidden connections between users and items and have the ability to provide serendipitous items [21], which are helpful to improve the diversity of recommendation.

The user-item interaction information can be either explicit or implicit [3]. Explicit interactions refer to users consciously expressing their preferences for items, e.g., discrete ratings for movies. Implicit interactions can be any source of user-generated information, such as purchases, clicks, bookmarks, listening times, etc. Usually, both explicit and implicit interactions can be recorded in a large but very sparse user-item matrix (Fig.1 shows an example).

Typical CF-based recommender systems associate a user with a group of like-minded users based on their preferences over all the items, and then recommends to the user those items enjoyed by others in the group. The basic assumption is that users with similar behaviors (e.g., ratings) will have similar tastes on all the items. However, we find that this assumption is not always tenable – two users having similar tastes on one item subset may have totally different tastes on another set. Moreover, one user's interests are usually

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

[1]http://www.amazon.com
[2]http://www.last.fm
[3]http://www.movielens.org

concentrative on some topics, but not dispersive over all items. So it is more natural to say a group of users are like-minded on a subset of items. In this paper, we call a subset of items and a group of interested users as a **user-item subgroup**. Fig.1 shows two subgroups. Note that users and items are not necessarily adjacent in the matrix. We expect that subgroups can help to capture similar user tastes on a subset of items.

Many clustering CF models utilize user clusters [29], item clusters [23], or co-clusters [9] to design CF algorithms. In these models, each user or item can only belong to a single cluster. In reality, it is more natural to assume that users (items) can join multiple clusters (subgroups), e.g., a user could like some movie topics and a movie could belong to multiple movie categories.

In this paper we extend traditional clustering CF models by co-clustering both users and items into multiple subgroups, and try to use them to improve the performance of CF-based recommender systems. Many previous works focus on the prediction accuracy, but the low prediction error can not guarantee a good recommendation quality. We focus on the top-N recommendation performance, which is more meaningful for real recommender systems. The main contributions of this paper include: (1) we formulate the Multiclass Co-Clustering problem (MCoC) and propose an effective solution to it for finding user-item subgroups; (2) we propose an unified framework to combine subgroups with (any) pure CF models; (3) we provide top-N recommendation comparisons of many CF models (before and after using our framework) on three real data sets and make a systematic empirical analysis for the results.

The remaining of this paper is structured as follows. Section 2.1 introduces CF and clustering CF models. Section 3 describes our proposed approach in detail. Experimental settings and results are discussed in section 4 and 5. In section 6 we provide a conclusion.

2. BACKGROUND

2.1 Collaborative Filtering

Breese et al. [3] divide collaborative filtering approaches into two classes: memory based and model based algorithms.

Memory based algorithms maintain the original setup of the CF task. They use statistical techniques to build the neighborhood relationship for an active user, and then usually use a weighted sum of the ratings to prediction missing values. This process is a bit like a ranking-analogue of nearest neighbor classifier, whereas the result is a real score but not a category label. A general user-based formulation of the weighted sum scheme can be [3]:

$$p_{a,j} = \overline{\mathbf{r}}_a + \kappa \sum_{i=1}^{n} w(a,i)(r_{i,j} - \overline{\mathbf{r}}_i), \qquad (1)$$

where n is the size of neighbors and $\overline{r}_a$, $\overline{r}_i$ are the average ratings for the active user a and neighbor user i respectively. Actually, The most important part for memory based algorithms is the similarity measurement. Popular examples are pearson correlation (PC) [26], vector similarity and various extensions of them [30]. Saewar et al. [27] and Deshpande et al. [7] compute item-item similarities and obtain the predictions or top-N recommendation via item-based ways.

Model based algorithms, in contrast, utilize the collection of training data to learn a model first and then use it to make predictions instead of directly manipulating the original database. The modeling process is always performed by machine learning or data mining techniques such as the Bayesian model [11], Regression-based model [32], Latent Semantic model [5,12] and Clustering model [9,23,29,31].

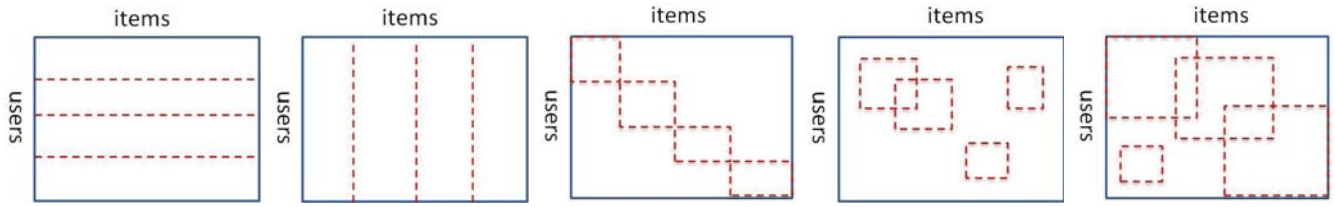
Although traditional CF models have been successful in many areas, they all have to face several critical problems: data sparsity, scalability and cold-start. To alleviate the sparsity problem, many matrix factorization models are used, such as the Singular Value Decomposition (SVD) [28], Nonnegative Matrix Factorization (NMF) [4,16], Maximum Margin Matrix Factorization (MMMF) [25] and Nonparametrix pPCA (NPCA) [33]. These models usually reduce the dimensions of the user-item matrix and smoothing out the noise information, which is also helpful to algorithm scalability. Many evidence have shown that many matrix factorization models outperform traditional CF methods in prediction accuracy. The cold-start situation is very common for real recommender systems, since they have many new users and new items in different time windows. For a new user with only a few user-generated information, normal CF methods can not capture his (her) personal taste accurately. An intuitive solution is to adding the content-based characteristics to collaborative models. Typical works using content information include [2, 21]. Personality diagnosis (PD) [24] is a special kind of hybrid approach which combines memory based and model based CF methods and retains some advantages of both algorithms. With the hot development of Web2.0, recently many new collaborative filtering algorithms are designed to integrate social or trust information [15,18].

2.2 Clustering Collaborative Filtering Models

The most related model to this paper is the clustering collaborative filtering model. A cluster is a collection of data samples having similar features or close relationships. For the collaborative filtering task, clustering is often an intermediate process and the resulting clusters are used for further analysis [30].

In general, the clustering models can be classified into several different types. We draw the sketch maps in Fig. 2. The most straightforward way is to partition the users into distinct groups. Sarwar et al. [29] cluster the complete user set based on user-user similarity and use the cluster as the neighborhood. In contrast, O'Connor et al. [23] use clustering algorithms to partition the set of items based on user rating data. Unger et al. [31] propose to cluster users and items separately by variants of k-means and Gibbs sampling. Users can then be re-clustered based on the number of items in each item cluster they rated, and items can similarly be re-clustered based on the number of user in each user cluster that rated them. The above three algorithms are all *one-sided* clutering, either for users or items. See Fig. 2(a) and Fig. 2(b), after some row (column) exchanges, we get the hard partitions of users (items).

Some other works consider of the *two-sided* clustering model. Typical works are [9,13]. We could see these methods as co-clustering (CoC) based CF models, since their clustering strategies are traditional co-clustering, e.g., the key idea of [9] is to simultaneously obtain user and item neighborhoods via co-clustering and generate predictions based on the average ratings of the co-clusters while taking the bi-



(a) User Clustering (b) Item Clustering (c) Co-Clustering (d) Biclustering (e) Multiclass CoC

Figure 2: Comparison of five clustering models for collaborative filtering.

ases of the users and items into account. See Fig. 2(c), after some row and column exchanges, we can get the distinct co-clusters with both users and items (we call them user-item subgroups in this paper).

One big limitation of the co-clustering approaches as well as the above one-sided clustering approaches is that, each user or item can be clustered into one single cluster only, whereas some recommender systems may benefit from the ability of clustering users and items into several clusters at the same time [1]. For example, in a movie web site, a user may be interested in multiple topics of movies and a movie can be liked by different groups of users from different aspects. So the multiclass co-clustering (MCoC) model, which is shown in Fig. 2(e), is more reasonable. It allows each user and item to be in multiple subgroups at the same time, i.e., subgroups may have some overlaps.

The last clustering type is the biclustering model (see Fig.2(d)) which is well studied in gene expression data analysis [6, 19]. It seems similar to MCoC – a bicluster is a subgroup of genes (users) and conditions (items). But they are different for that biclustering usually finds some maximum biclusters with low residue scores [6], i.e., biclusters always can not cover all rows and columns.

In this paper, we pay our most attention to the model of multiclass co-clustering.

3. OUR ALGORITHM

Our primary goal is to find potential user-item interest subgroups flooded in the large user-item matrix, and then use them to improve the performance of collaborative recommender systems. There are two main questions:

1. How to find meaningful user-item subgroups from limited information? The only information we have is the user-item matrix, such as ratings for movies and listening times for music.

2. How to combine user-item subgroups with existing collaborative filtering methods and improve their performance? We need a strategy to handle the cases that one user and one item can both belong to one, two (or more), or zero subgroups.

Our algorithm is to answer these two questions – we find user-item subgroups by solving a Multiclass Co-Clustering problem (MCoC) and propose an unified strategy to combine subgroups with existing collaborative filtering methods. Considering that this paper is just an exploration work – to explore a new improving space for collaborative recommender systems, we choose to pay our attention to the pure CF situation.

3.1 Problem Formulation of MCoC

Suppose there are n users and m items, and the only information we have is the user-item matrix $T \in \mathbb{R}^{n\times m}$ where each element T_{ij} is the preference of user i to item j. We use $\mathbf{u}_i$ to denote the i-th user and $\mathbf{y}_j$ to denote the j-th item.

The goal is to simultaneously divide the users $\{\mathbf{u}_1, \cdots, \mathbf{u}_n\}$ and items $\{\mathbf{y}_1, \cdots, \mathbf{y}_m\}$ into c subgroups. This is a bit like the Co-Clustering (CoC) problem [8, 9, 34], but the main difference and also the key point is that any user or item can appear in multiple subgroups. So we call it Multiclass Co-Clustering problem, or just MCoC for short.

We want the MCoC result be represented by a partition matrix $P \in [0,1]^{(n+m)\times c}$, where each element P_{ij} is an indicator value of the corresponding entry (a user or an item). $P_{ij} > 0$ if the i-th entry belongs to the j-th subgroup, and $P_{ij} = 0$ otherwise. The magnitude of P_{ij} shows the relative weight of entry i belonging to subgroup j, thus each row of P sums to 1. If we fix the number of subgroups that each entry belongs to, e.g., k subgroups ($1 \le k \le c$), then we get exactly k non-zeros in each row of P. When $k = 1$, the problem above is equal to the traditional Co-Clustering problem. Naturally, partition matrix P can be written as

$$P = \begin{bmatrix} Q \\ R \end{bmatrix}, \tag{2}$$

where $Q \in [0,1]^{n\times c}$ is the partition matrix of users and $R \in [0,1]^{m\times c}$ is the partition matrix of items.

3.2 Solution of MCoC

We use a very simple but reasonable method to solve MCoC Problem. However, better methods could be explored to further improve the quality of user-item subgroups in future work. Intuitively, if one user and one item have a high rating score, they are very likely to appear in one or more subgroups together. In order to make those strongly associated users and items together, inspired by [10, 34, 35], we adopt the following loss function to model the user-item relationships:

$$\epsilon(Q,R) = \sum_{i=1}^{n}\sum_{j=1}^{m}(\|\frac{\mathbf{q}_i}{\sqrt{D_{ii}^{row}}} - \frac{\mathbf{r}_j}{\sqrt{D_{jj}^{col}}}\|^2 T_{ij}), \tag{3}$$

where $\mathbf{q}_i$ is the i-th row of Q and $\mathbf{r}_j$ is the j-th row of R. $D^{row} \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix of users with $D_{ii}^{row} = \sum_{j=1}^{m} T_{ij}$ and $D^{col} \in \mathbb{R}^{m\times m}$ is the diagonal degree matrix of items with $D_{jj}^{col} = \sum_{i=1}^{n} T_{ij}$.

The loss function is easy to understand. Since we have only the user-item interaction information, minimizing Eq.(3) means that the indicator vectors of user i (q_i) and item j (r_j) should be very close if they have a high rating score. And

for those missing value pairs, there is no restriction to their indicator vectors. By some simple linear algebra derivations, we can get :

$$\begin{aligned}
&\epsilon(Q,R)\\
&= \sum_{i=1}^{n}\|\mathbf{q}_i\|^2 + \sum_{j=1}^{m}\|\mathbf{r}_i\|^2 - \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{2\mathbf{q}_i^T\mathbf{r}_j T_{ij}}{\sqrt{D_{ii}^{row}D_{jj}^{col}}}\\
&= Tr(Q^TQ + R^TR - Q^TSR)\\
&= Tr\left([Q^T\ R^T]\begin{bmatrix} I_n & -S\\ -S^T & I_m\end{bmatrix}\begin{bmatrix} Q\\ R\end{bmatrix}\right)\\
&= Tr(P^TMP),
\end{aligned} \tag{4}$$

where

$$S = (D^{row})^{-\frac{1}{2}}T(D^{col})^{-\frac{1}{2}},\ M = \begin{bmatrix} I_n & -S\\ -S^T & I_m\end{bmatrix}. \tag{5}$$

I_n is an identity matrix in the size of $n \times n$. We have several strong constraints on the partition matrix P (described in section 3.1), so finally, we are to solve the following optimization problem:

$$\begin{aligned}
\min_{P}\quad & Tr(P^TMP)\\
\texttt{s.t.}\quad & P \in \mathbb{R}^{(m+n)\times c},\\
& P \geq 0,\\
& P\mathbf{1}_c = \mathbf{1}_{m+n},\\
& |P_i| = k,\ i = 1,\ldots,(m+n).
\end{aligned} \tag{6}$$

The combined constraints $P \geq 0$ and $P\mathbf{1}_c = \mathbf{1}_{m+n}$ force each element of P to stay in the range of $[0, 1]$. Parameter c is the total number of subgroups and k is the number of subgroups each user or item can join in ($1 \leq k \leq c$). Notation $|\cdot|$ is the cardinality constraint which means the number of non-zero elements of a vector. Vector P_i is the i-th row of partition matrix P.

Solving objective function (6) directly is not easy, since it is nonconvex and discontinuous. In this paper, we propose to use an efficient approximation method to solve this problem. Similar to the spectral clustering model [22] and the bipartite graph model [8], our method has two primary stages:

Stage 1. Mapping all users and items into a shared low-dimensional space.

According to the objective function (6), we think that the optimal r-dimensional embedding X^*, which preserves the user-item preference information, could be got by solving the following problem:

$$\begin{aligned}
\min_{X}\quad & Tr(X^TMX)\\
\texttt{s.t.}\quad & X \in \mathbb{R}^{(m+n)\times r}, X^TX = I.
\end{aligned} \tag{7}$$

The constraint $X^TX = I$ is to avoid the arbitrary scaling of X. From Eq. (3) and (4), we find that M is a positive semidefinite matrix. The optimal solution X^* that minimizes the objective function (7) is given by the solution of eigenvalue problem $MX = \lambda X$. So $X^* = [\mathbf{x}_1, \ldots, \mathbf{x}_r]$, where $\mathbf{x}_1, \ldots, \mathbf{x}_r$ are the smallest eigenvectors of matrix M sorted by their corresponding eigenvalues.

Stage 2. Discovering subgroups.

As long as we have the unified representation X of users and items, we need to find the user-item subgroups, i.e., to compute the partition matrix P.

There are two cases: a user or an item can belong to one single class or multiple classes. For single class case, we use k-means to cluster the data X. Then each row of P has only one positive number and the index is the cluster label. For multiple class case, which is the main focus of this paper, we choose to use the fuzzy c-means [17], a soft clustering method. The algorithm is an iterative optimization that minimizes the criterion function:

$$J_m(P,V) = \sum_{i=1}^{m+n}\sum_{j=1}^{c}(P_{ij})^l d(\mathbf{x}_i,\mathbf{v}_j)^2, \tag{8}$$

where P_{ij} is the membership of entry $\mathbf{x}_i$ (a user or an item) in cluster j, and v_j is the center of cluster j. The function d is a distance function which can be predefined and the parameter l is a weighting exponent controlling the fuzziness of the resulting partition. In our method, d is the Euclidean distance and l is 2. During each iteration, we update P and V as follows:

$$P_{ij} = (d(\mathbf{x}_i,\mathbf{v}_j))^{2/(1-l)} \Big/ \left[\sum_{k=1}^{c}(d(\mathbf{x}_i,\mathbf{v}_k))^{2/(1-l)}\right], \tag{9}$$

and

$$\mathbf{v}_j = \left[\sum_{i=1}^{m+n}P_{ij}^l\mathbf{x}_i\right] \Big/ \left[\sum_{i=1}^{m+n}P_{ij}^l\right], \tag{10}$$

for all $i = 1, \ldots, (n+m)$, and $j = 1, \ldots, c$. The algorithm is stopped if the improvement of objective function values at two successive iteration steps is less than a threshold ε. In our experiments, ε is 1e-5. Then for each row of P, only the top-k biggest elements are retained and each row sums to one (normalized).

3.3 Recommendation with Subgroups

By now we have find some user-item subgroups, then, we'll describe how to combine them with existing collaborative recommender systems. The main idea is applying some CF algorithm in each subgroup and try to merge the prediction results together. For a pure CF method, the only input is the user-item matrix and the output is the prediction score for each missing value in the matrix. Actually, for each subgroup, we could get a submatrix from the original big user-item matrix T, i.e., we could adopt any CF method without any modification. This is a big benefit for us that we can pay our most attention to subgroup discovering, for better subgroup quality, but concern little about the effectiveness of a CF method.

The last problem is how to merge the prediction results from all subgroups. One user and one item can belong to one , two (or more), or zero subgroups, so we propose an unified framework to handle all cases. Let $Pre(u_i, y_j, k)$ be the prediction score of user i to item j in subgroup k by some CF method, and Y_{ij} be the final prediction score of user i to item j, we define

$$Y_{ij} = \begin{cases} \sum_k Pre(\mathbf{u}_i,\mathbf{y}_j,k)\cdot\delta_{ik} & \text{if } \mathbf{u}_i \text{ and } \mathbf{y}_j \text{ belong to one or more subgroups,}\\ 0 & \text{otherwise.}\end{cases} \tag{11}$$

In the above formulation, δ_{ik} is an indicator value of user i representing whether subgroup k is his (her) most interesting

subgroup in the shared subgroups with item j. Then

$$\delta_{ik} = \begin{cases} 1 & \text{if } P_{ik} \text{ is } \max(R_{u_i} \bigcap Q_{y_j}), \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

where operator $\bigcap$ returns the index of non-zero overlap items of two vectors and function $\max(\cdot)$ returns the maximal element value in a vector and matrix P is the partition matrix which has been described above. This simple framework can handle different overlapping cases for users and items.

We could see δ_{ik} as a weight of the prediction $Pre(\mathbf{u}_i, \mathbf{y}_j, k)$, but in this paper we just use the hard weight, 0 or 1. The performance of using a soft weight could be investigated in future work. By the strategy above, we get a lot of (but not all) prediction scores for each user and commonly the items (un-rated) having top scores are recommended to the user. This is the end of our method. It is independent of any specific CF algorithm.

3.4 Discussion

In this section, we will talk about several detailed issues for our proposed approach.

By Eq.(11), we just consider of those correlated users and items – an item could be a candidate recommendation item for a user if and only if they both belong to one or more subgroups. But occasionally, some subgroups may have very few elements due to the unbalance of clustering, e.g., less than 10. So under the extreme case, some user may have not enough correlated items for recommendation. We could remove the small subgroups or recompute the fuzzy c-mean process, or just make an easy solution – add some popular items to those users for this time of recommendation. The latter one is a good strategy for a real recommender system. But our goal is to investigate the usefulness of subgroups, so we don't add any additional uncorrelated items.

Although the user-item interaction data is very sparse, our method utilize the dimensionality reduction technique to alleviate the data sparsity problem. Even for some new users with very few interaction data (we call it Cold Start problem), the algorithm is still competent for this case and recommend items for them in the same way. Moreover, since users and items belong to multiple groups, most submatrices are more dense but smaller than the original user-item matrix. So the sparsity and scalability problems for some recommendation methods are both reduced in a degree in our framework.

The last critical issue is the computational cost. Our method seems to be costly – both the eigenvector computation and fuzzy clustering are time consuming. However, due to the high sparsity of the user-item matrix, we just require very few eigenvectors, e.g., 3, for data representation (see section 5.1.2). Clustering a data set with only 3 dimensions is always very fast. So for a common size data set, our framework will not affect the recommender system's efficiency. Both the sparsity and computational cost are compared by experiments in section 5.5.

3.5 Algorithm Overview

Here we make a brief summary of our framework for top-N recommendation. In practice, we do following steps:

Step1: Dimensionality reduction.

1a. Calculate matrix S by normalizing the user-item matrix T and construct matrix M. See Eq.(5).

1b. Compute the r smallest eigenvectors of M: $\mathbf{x}_1, \ldots, \mathbf{x}_r$, and form the shared low dimensional representation $X^* = [\mathbf{x}_1, \ldots, \mathbf{x}_r]$. See Eq.(7)

Step2: Finding subgroups.

2a. Cluster users and items into c subgroups by fuzzy c-means for multi-class case or by k-means for single-class case. See Eq.(9) and (10). So we get the partition matrix P describing group information of all the users and items.

Step3: Top-N recommendation.

3a. Run any CF method on each submatrix (subgroup) to predict missing values, and merge the results by an unified framework. See Eq.(11).

3b. For each user, N items with highest predict scores are recommended.

4. EXPERIMENTAL SETTINGS

We conduct many experiments to evaluate the effectiveness of the proposed method. In this chapter, we describe the experimental settings in detail.

4.1 Data Description

Our experiments are performed on three real data sets: MovieLens-100K[4], MovieLens-1M and Lastfm.

The MovieLens-100K data set is a classic movie rating data set collected through the MovieLens web site. It consists of 100,000 ratings (1-5) from 943 users on 1,682 movies and each user has rated at least 20 movies.

The MovieLens-1M data set is a larger data set which consists of 1,000,209 ratings (1-5) from 6,040 users on 3,952 movies and each user has rated at least 20 movies.

We collected Lastfm data set from Last.fm web site in December 2009. We firstly crawled the listening counts of 2,598 users on 30,727 songs. Then we reduced the candidate set of songs and users by restricting that each song has been listened by more than 30 users and each user has listened more than 50 songs. So we get a subset of 2,059 users and 4,173 songs. The basic statistics of these three data sets are shown in Table 1.

Table 1: Basic statistics of the data sets MovieLens-100K, MovieLens-1M and Lastfm.

	ML-100K	ML-1M	Lastfm
# of users	943	6,040	2,059
# of items	1,682	3,952	4,173
# of ratings	100,000	1,000,209	257,117
# of ratings per user	106.04	165.60	124.87
# of ratings per item	59.45	253.09	61.61

4.2 Comparisons

We investigate many popular CF methods including two memory based algorithms with User-based and Item-based, one hybrid method of personality diagnosis (PD), four well known matrix factorization methods (model based) of SVD,

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

NMF, MMMF, NPCA, and also recommendation by popularity (POP). Later we combine these recommendation methods with our framework and check whether their performance is improved.

User-based: For user-based algorithm, we use a representative similarity metric – pearson correlation, to measure the user-user similarities and use the user-based model in [14].

Item-based: For item-based algorithm, we use the vector cosine similarity model to compute the item-item similarities and use the item-based model in [14].

PD: Personality diagnosis is a representative hybrid CF approach that combines memory based and model based algorithms and retains some advantages of both algorithms [24]. We adopt the procedure in the CF toolkit[5] for PD.

SVD: This is maybe the most popular collaborative filtering technique. Once the user-item matrix is decomposed into three component matrices with k features: $T = U_k S_k V_k^T$, the prediction score of user i to item j is $Pred_{ij} = \overline{\mathbf{r}_i} + U_k\sqrt{S_k}^T(i) \cdot \sqrt{S_k}V_k^T(j)$, where $\overline{\mathbf{r}_i}$ is the i-th row average of T, and T is the normalization of T according to [28].

NMF: We use the most commonly used algorithm for Non-negative Matrix Factorization described in [16]. Similar to SVD, the user-item matrix is decomposed into two component matrices with the base matrix having k bases. For both SVD and NMF, we use $k = 6$.

MMMF: The Maximum Margin Matrix Factorization is a state-of-the-art algorithm with good collaborative prediction performance. We use the procedure in [25].

NPCA: The algorithm was proposed in [33] and according to the authors, it produced one of the most accurate prediction results among matrix factorization methods.

POP: Recommendation by popularity is wildly used in many recommender systems. Usually, it has a competitive recommendation precision due to the hot effect – hot items are more likely to be clicked. But it's result could not reflect personalization. In this paper, we use the number of ratings for each item as the popularity score.

4.3 Evaluation Metric Discussion

Many previous works compute Root Mean Square Error (RMSE) or Mean Absolute Error (MAE) to evaluate accuracy of the predict scores. In our consideration, they are good measurements for prediction or matrix completion task, but not for a general top-N recommendation task. For a real top-N recommender system, no matter what strategy it adopts, the final output for a user is just a ranked list of items, e.g., a list of movies in a video web site, a list of songs in a music web site, or a list of hotels in a travel web site. The low error prediction scores can't guarantee a good recommendation list, e.g., prediction score 3 and 5 have the same absolute error with the test score 4 (ground truth), but their positions in the ranked list are extremely different in a 1-5 rating system. So the most important is to evaluate the quality of the ranked list.

The click action at a recommended item maybe the most straightforward but useful feedback from the user – at least he (she) is interested in the item. The click may cause a user to listen to a song in a music web site or buy a product in a e-commerce web site. So if one recommended item is clicked, it is treated as a hit item, otherwise it's a miss item. But we don't have the click information, so we use the test data (e.g., ratings) to approximate the click action – if one recommended item has a rating score on the test data, it's a hit item, otherwise it's a miss item.

To make the experimental results comparable, we use many well known metrics to measure the ranked list. Similar to information retrieval evaluation, we use Precision@K to evaluate the quality of the top K recommended items. Considering that some users may have a large number of ratings in the test data while some other users just have a few, so F_1 score may be more reasonable. MAP (Mean Average Precision) provides a single-figure measure of quality across recall levels [20]. For a single user j, Average Precision is the average of precisions computed at the point of each correctly recommended item $(d_1, \ldots, d_{mj})$ in the ranked list, and this value is then averaged over the user set U:

$$\text{MAP}(U) = \frac{1}{|U|}\sum_{j=1}^{|U|}\frac{1}{m_j}\sum_{k=1}^{m_j}\text{Precision}(R_{jk}), \qquad (13)$$

where R_{jk} is the set of ranked results from the top result until you get to item d_k. NDCG [20] is a wildly used metric for a ranked list. NDCG@K is defined as:

$$\text{NDCG@}K = \frac{1}{\text{IDCG}} \times \sum_{i=1}^{K}\frac{2^{r_i-1}}{\log_2(i+1)}, \qquad (14)$$

where r_i is 1 if the item at position i is a hit item and 0 otherwise. IDCG is chosen so that the perfect ranking has a NDCG value 1.

5. EXPERIMENTAL RESULTS AND ANALYSIS

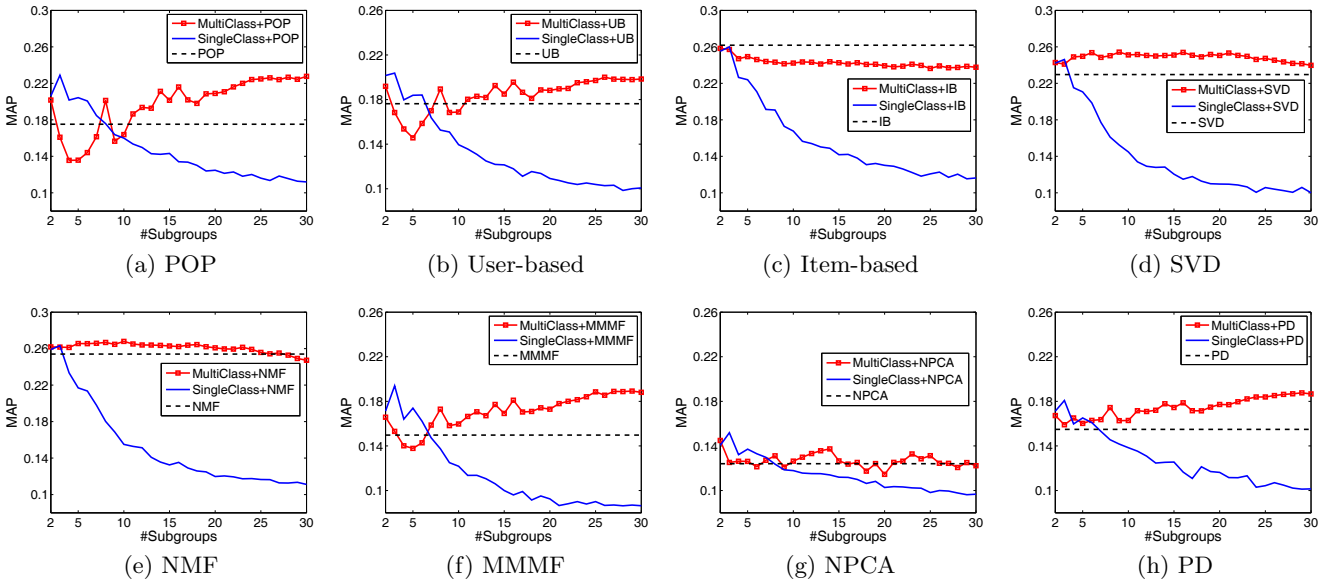
We run many state-of-the-art recommendation methods and check whether their top-N recommendation performance are improved after using our framework. After that, we make an empirical analysis to the results, as well as the sparsity and scalability.

5.1 Performance on MovieLens-100K

The MovieLens-100K data set is divided into five disjoint splits. Each split is used for testing and the rest four for training, so there are five different training/testing sets. We repeat our experiment with each set and average the results. In Fig.3 we plot MAP of the top-N recommendation results versus different number of subgroups by eight methods described above. In each subgraph, the horizontal broken line represents original performance of that method, and the other two lines represent the performance of using subgroups. The red line with small squares is for the case that users and items belong to multiple classes and the blue line for single class. From the results, we find that our method has positive effect for most of the methods including POP, User-based, SVD, NMF, MMMF, and PD. However it has unstable effect to NPCA and has negative effect to Item-based recommendation method.

Then in Table 2, we record other three evaluation metrics: Precision, NDCG and F_1 score on position 10. We compare the performance of using 15 and 25 subgroups with the base performance. The bold number indicates that its value has an obvious improvement than the base value (difference $\geq$ 0.01). The magnitude of Precision directly shows the hit rate of top-10 recommendation, while F_1 score, which both considering of the corresponding recall value, is more fair

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



(a) POP (b) User-based (c) Item-based (d) SVD

(e) NMF (f) MMMF (g) NPCA (h) PD

Figure 3: Comparisons of the recommendation performance on MovieLens-100K data set by eight CF methods. In each subgraph, the black broken line shows the original performance while the red (with squares) and blue real lines show the performance of using subgroups for multi-class and single-class respectively. The number of subgroup varies along the horizontal axis.

Table 2: Performance Comparisons on MovieLens-100K in terms of Precision, F1 and NDCG. 15 and 25 subgroups are used for our approach. The bold number indicates an obvious improvement (difference ≥ 0.01).

Recommendation Method	base performance			performance on 15 subgroups			performance on 25 subgroups		
	P@10	NDCG@10	F1@10	P@10	NDCG@10	F1@10	P@10	NDCG@10	F1@10
POP	0.222	0.301	0.118	**0.258**	**0.351**	**0.139**	**0.282**	**0.373**	**0.156**
User-based	0.221	0.333	0.109	**0.235**	**0.347**	**0.121**	**0.244**	**0.351**	**0.128**
Item-based	0.335	0.421	0.188	0.302	0.394	0.172	0.292	0.384	0.165
SVD	0.292	0.395	0.149	**0.304**	**0.409**	**0.160**	0.296	**0.414**	0.154
NMF	0.341	0.422	0.189	0.340	0.416	0.190	0.324	0.411	0.179
MMMF	0.176	0.295	0.082	**0.199**	**0.306**	**0.103**	**0.211**	**0.317**	**0.112**
NPCA	0.151	0.283	0.066	0.151	**0.297**	0.067	0.157	**0.303**	0.074
PD	0.184	0.293	0.092	**0.205**	0.297	**0.114**	**0.216**	0302	**0.121**

and comprehensive. But to evaluate the quality of a ranked list, only the hit rate is not enough – a hit item in position 1 or position 10 has the same Precision@10. Thus NDCG is more meaningful to compare the ranked lists. From the NDCG values in Table 2, we can see that our method has positive effect for the methods of POP, User-based, SVD, MMMF, NPCA and PD, but has negative effect for NMF and Item-based methods.

5.1.1 *Multi-class vs. Single-class*

An interesting phenomenon is that when the number of subgroups (c) is very small, e.g., just 2 or 3, the performance of our method with single-class is very good (see the blue lines in Fig.3). This is easy to understand: one people's interests are usually concentrative on some topics or some correlated items, but not dispersive over all the item set. Clustering with a small number of clusters actually do the work of denoising – separating uncorrelated items into different subgroups. This result is conform with some previous clustering CF models [9, 29]: they have good performance with small number of clusters, e.g., in [9], the number of clusters is just 3. As c increases, the total number of items in each cluster decreases, which makes the performance dropped quickly.

For some recommendation methods, such as POP, User-based and MMMF, the performance of our method with multi-class is just opposite to the single-class: bad in small subgroup number but good in large number. In our consideration, that means small number of subgroups can not clearly partition different user-item interest subgroups. Under such condition, the fuzzy weights are inaccurate to capture the user's preferences. As c increases, better subgroups are got and the fuzzy weights are more meaningful. Then for those algorithms, the effect of our method becomes more and more evident.

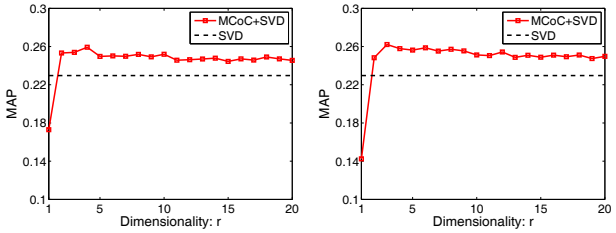
5.1.2 *Parameter Selection*

Parameter selection plays a key role to many algorithms. Sometimes, one algorithm's performance may drastically vary with different choices of parameters. For our method, there are two main parameters: r and k. Parameter r is the number of eigenvectors computed in the dimensionality reduction step. Fig.4 shows the performance of combining MCoC with SVD by different number of dimensionality r. From the figure, we find that the performance is competitive when using just a few eigenvectors, so in our experiments, we just select $r = 3$. Parameter k is the number of subgroups that

Table 3: Performance Comparisons on MovieLens-1M in terms of MAP and NDCG. 5, 15 and 25 subgroups are used for our approach. The bold number indicates an obvious improvement (difference ≥ 0.01).

Recommendation Method	base		5 subgroups		15 subgroups		25 subgroups	
	NDCG@10	MAP	NDCG@10	MAP	NDCG@10	MAP	NDCG@10	MAP
POP	0.379	0.264	**0.391**	0.272	**0.419**	**0.294**	**0.429**	**0.300**
User-based	0.368	0.201	**0.382**	**0.213**	**0.385**	**0.231**	**0.409**	**0.246**
Item-based	0.493	0.368	0.465	0.348	0.444	0.334	0.479	0.356
SVD	0.477	0.311	**0.492**	**0.334**	**0.506**	**0.342**	**0.497**	**0.343**
NMF	0.480	0.342	**0.496**	**0.363**	0.487	**0.361**	**0.492**	**0.368**
MMMF	0.329	0.220	0.309	0.207	**0.361**	**0.245**	**0.355**	**0.244**
NPCA	0.311	0.202	0.299	0.197	**0.352**	0.154	**0.326**	0.187
PD	0.293	0.213	0.297	0.222	0.299	**0.225**	**0.303**	**0.227**

each user or item can belong to. For different subgroup number c, we set $k = \lceil \log_2(c) \rceil$.



(a) 10 subgroups (b) 20 subgroups

Figure 4: Performance on different number of dimensionality r with 10 and 20 subgroups.

5.2 Performance on MovieLens-1M

MovieLens-1M is a large data set with more distinct users than items. It is randomly divided into 60% training data and 40% testing data. Parameters are the same with those in section 5.1.2. We record the MAP and NDCG@10 of each algorithm's result before and after using our method in Table 3. We select to use 5, 15 and 25 subgroups for our approach. The bold number indicates that its value has an obvious improvement than base value (difference ≥ 0.01). From NDCG values, we find that our method has positive effect to POP, User-based, SVD, NMF, MMMF, NPCA and PD. But from the MAP values, our method's performance falls in NPCA. That means in MovieLens-1M data set, by using MCoC, the rate of hit items in NPCA's recommendation list is reduced, whereas the positions of hit items are moved up. Overall, the performance of our approach on MovieLens-1M is similar to that on MovieLens-100K, since they are from the same web source.

5.3 Performance on Lastfm

Lastfm data set is quite different with the above two MovieLens data sets. Its user-item data is not user specified ratings. In last.fm web site, we can get one user's listening logs. So we use the listening count as the implicit interaction data and form the user-item listening matrix. It is randomly divided into 60% training data and 40% testing data. Parameters are the same with those in section 5.1.2.

Obviously, the elements of the user-item matrix vary in a large range – for a user, some music are just listened once and some are listened more than one hundred times. So how to use the user-item matrix is a problem. Besides to use the original listening data, we use the re-scaled data to alleviate the big variance. Let $\hat{T}_{ij}$ be the new element of user-item matrix T, then

$$\hat{T}_{ij} = \log_2(T_{ij} + 1). \quad (15)$$

By the equation, zero values will be still zero and positive values will be re-scaled by logarithm. This is quite reasonable for dealing with the listening number: listening one song 10 times means likeness for some users, but does listening 100 times for some other users means a 10 times stronger likeness? Sometimes, it doesn't mean a strong likeness for those users, but just means that they are active users. The usage of re-scaling could be observed from Table 4. Most algorithms' base performance are improved in a large degree after using the re-scaled data.

Due to the uncertain range of the data, we can't use the method of PD, since its procedure requires to input the number of distinct preference values. We use the rest seven methods in Lastfm data set and the results are showed in Table 4. From NDCG and MAP values, we find that our approach is useful for the methods of POP, User-based, SVD and NMF, but not for Item-based, MMMF and NPCA.

5.4 Results Analysis

The results on two MovieLens data sets (100K and 1M) and Lastfm data set evidently show that our approach has positive effect for many CF methods, but not for all.

Among these methods, the Item-based method is a very special one – our approach always lowers its performance on all the data sets. This is mainly because the Item-based method depends on the prior information of one user's history ratings and all item-item similarities, i.e., it doesn't benefit from the neighborhood relationships as traditional CF models does. When we use the subgroups, the prior information is reduced for each user, which negatively affects the recommendation. But for the other seven methods, our approach has positive effect for them on one or more data sets. The Item-based method has almost the best base performance on all the data sets. This proves our assumption that one user's interests are concentrative on some topics or some correlated items, because Item-based method tends to recommend similar items agreeing with the user's preference history. However, although Item-based method is good in precision, it is hard to extend the selection range for the user, i.e, the diversity of the recommendation is low. In our consideration, a good recommendation list should consider of both relevance and diversity.

Our approach acts differently on explicit rating data and implicit listening data. On explicit rating data, it is useful for most of the methods. On implicit listening data, it is only useful for four methods, but the improvement is relatively

Table 4: Performance Comparisons on Lastfm in terms of MAP and NDCG. 30 subgroups are used for our approach. The left part is the result of using original data and the left part is the result of using normalized data. The bold number indicates an obvious improvement (difference ≥ 0.01) and symbol * indicates a big improvement (difference ≥ 0.05).

Recommendation Method	Original Data				Re-scaled Data			
	base		30 subgroups		base		30 subgroups	
	NDCG@10	MAP	NDCG@10	MAP	NDCG@10	MAP	NDCG@10	MAP
POP	0.224	0.107	**0.261**	**0.122**	0.224	0.107	**0.277***	**0.157***
User-based	0.180	0.057	**0.221**	**0.104**	0.337	0.107	0.320	**0.171***
Item-based	0.602	0.458	0.544	0.423	0.694	0.545	0.625	0.493
SVD	0.230	0.096	**0.345***	**0.191***	0.340	0.123	**0.511***	**0.297***
NMF	0.230	0.097	**0.347***	**0.200***	0.380	0.213	**0.552***	**0.403***
MMMF	0.219	0.121	0.193	0.082	0.294	0.155	0.241	0.116
NPCA	0.234	0.081	0.170	0.078	0.598	0.475	0.467	0.310

large, e.g., the NDCG improvement of SVD on re-scaled Lastfm data is very large (difference > 0.15).

At last, we find that people's behaviors are largely affected by the factor of popularity – for all the three data sets, POP method has a relatively good performance. And the popularity in a subgroup still works, so POP is always improved by our approach. Actually, both the popularity and personal taste can attract the user to click an item in a real web system.

5.5 Sparsity and Scalability

In Table 5 we record the sparsity (percent of zero elements in a matrix) of original user-item matrix and the average sparsity of subgroups, as well as the computational time of MCoC. 10, 20 and 30 subgroups are used for comparison. Each number of our method are the average result of ten independent runs. Experiments are run on a computer with double 3.16 GHz CPU and 3 GB RAM.

From the results, we find that when more subgroups are used, each subgroup becomes more dense, which is good for reducing data sparsity problem for some CF methods. Note that when 30 subgroups are used, the sparsity of Lastfm is very low. In our opinion, this is one reason of why MCoC has a big improvement for some CF methods on Lastfm. The short runtime of MCoC shows its good efficiency.

Table 5: Sparsity and runtime comparisons on MovieLens-100K, MovieLens-1M and Lastfm data sets. Term subg is short for subgroups.

		ML-100K	ML-1M	Lastfm
Sparisity	Base	0.950	0.975	0.982
	10 subg	0.933	0.940	0.922
	20 subg	0.736	0.925	0.542
	30 subg	0.652	0.838	0.325
		ML-100K	ML-1M	Lastfm
MCoC Runtime (s)	10 subg	0.548	3.977	2.150
	20 subg	0.853	3.778	2.769
	30 subg	1.203	3.098	4.166

6. CONCLUSIONS

User-item subgroups can help to capture similar user tastes on a subset of items. In this paper, we explore a new improving space for collaborative recommender systems – utilizing user-item subgroups. This is a natural extension of traditional clustering CF models. Experimental results show that using subgroups is a promising way to further improve the top-N recommendation performance for many popular CF methods. We expect our exploration can attract further research or practice on the topic of clustering CF model. Future works are needed in two main aspects: one is to find better user-item subgroups and the other is to design new methods to fully utilize subgroups.

7. ACKNOWLEDGEMENT

This work was supported in part by National Natural Science Foundation of China (Grant No: 61125203, 61173186, 90920303) , National Basic Research Program of China (973 Program) under Grant 2011CB302206 and Fundamental Research Funds for the Central Universities and Program for New Century Excellent Talents in University (NCET-09-0685).

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, pages 734–749, 2005.

[2] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[3] J. Breese, D. Heckerman, C. Kadie, et al. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[4] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized non-negative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, 2011.

[5] D. Cai, X. Wang, and X. He. Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[6] Y. Cheng and G. Church. Biclustering of expression data. In *Proceedings of International Conference on Intelligent Systems for Molecular Biology*, volume 8, page 93, 2000.

[7] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[8] I. Dhillon. 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*, pages 269–274, 2001.

[9] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. 2005.

[10] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems 16*. 2003.

[11] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *The Journal of Machine Learning Research*, 1:49–75, 2001.

[12] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[13] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 688–693, 1999.

[14] Z. Huang, D. Zeng, and H. Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *Intelligent Systems, IEEE*, 22(5):68–78, 2007.

[15] I. Konstas, V. Stathopoulos, and J. Jose. On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 195–202, 2009.

[16] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2001.

[17] J. Leski. Towards a robust fuzzy clustering. *Fuzzy Sets and Systems*, 137(2):215–233, 2003.

[18] H. Ma, D. Zhou, C. Liu, M. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296, 2011.

[19] S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE Transactions on computational Biology and Bioinformatics*, pages 24–45, 2004.

[20] C. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. 2008.

[21] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 187–192, 2002.

[22] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.

[23] M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 1999.

[24] D. Pennock, E. Horvitz, S. Lawrence, and C. Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the 16th conference on uncertainty in artificial intelligence*, pages 473–480, 2000.

[25] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719, 2005.

[26] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[27] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. 2000.

[29] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology*, pages 158–167, 2002.

[30] X. Su and T. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[31] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, pages 112–125, 1998.

[32] S. Vucetic and Z. Obradovic. Collaborative filtering using a regression-based approach. *Knowledge and Information Systems*, 7(1):1–22, 2005.

[33] K. Yu, S. Zhu, J. Lafferty, and Y. Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 211–218, 2009.

[34] L. Zhang, C. Chen, J. Bu, Z. Chen, D. Cai, and J. Han. Locally discriminative co-clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[35] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 595–602, 2004.