Latent Features Based Prediction on New Users' Tastes

Ming-Chu Chen and Hung-Yu Kao

Department of Computer Science and Information Engineering,
National Cheng Kung University, Tainan, Taiwan, R.O.C.
mcchen@ikmlab.csie.ncku.edu.tw, hykao@mail.ncku.edu.tw

Abstract. Recommendation systems have become popular in recent years. A key challenge in such systems is how to effectively characterize new users' tastes — an issue that is generally known as the cold-start problem. New users judge the system by the ability to immediately provide them with what they consider interesting. A general method for solving the cold-start problem is to elicit information about new users by having them provide answers to interview questions. In this paper, we present Matrix Factorization K-Means (MFK), a novel method to solve the problem of interview question construction. MFK first learns the latent features of the user and the item through observed rating data and then determines the best interview questions based on the clusters of latent features. We can determine similar groups of users after obtaining the responses to the interview questions. Such recommendation systems can indicate new users' tastes according to their responses to the interview questions. In our experiments, we evaluate our methods using a public dataset for recommendations. The results show that our method leads to better performance than other baselines.

Keywords: Recommendation System, Collaborative Filtering, Cold Start.

1 Introduction

Recommendation systems have become increasingly popular in recent years and are widely used in e-commerce and in social networks such as Amazon[1], Netflix[2] and Facebook[3]. Amazon.com recommends specific products that customers may be interesting when they are shopping. Facebook recommends friends to users by analyzing the relationships among users. The goal of a recommendation system is to provide personalized recommendations for products that are aligned with users' tastes.

There are two types of users in recommendation systems: existing users and new users. Existing users are those who already have historical data attached to them, and new users are those who have not previously been evaluated in the recommendation system. However, we can also categorize all items as existing items or new items by considering whether the item has received ratings. Thus, there are four partitions in a recommendation system, consisting of two types of users and two types of items. Fig. 1 illustrates these partitions.

[1] http://www.amazon.com
[2] https://signup.netflix.com/global
[3] https://www.facebook.com

V.S. Tseng et al. (Eds.): PAKDD 2014, Part I, LNAI 8443, pp. 176–187, 2014.

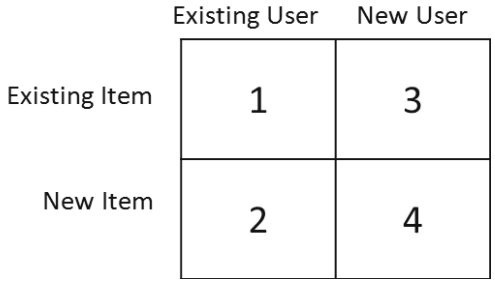

Fig. 1. The four partitions in a recommendation system

Partition 1 consists of the recommendations of existing items to existing users and represents the standard situation in recommendation systems. Many *Collaborative Filtering* techniques, such as *k-nearest neighbors* (KNN) and *singular value decomposition* (SVD), work well in this situation. Partition 2 is a situation that includes recommendations of new items to existing users. *Content-based Filtering* works well in this category because it does not rely on the historical data of users. This approach can recommend items if the content information of the items is available [3], for example, the actors, genre and release year of a movie or the text in a book. Partition 3 is an important part of recommendation systems and represents the situation of recommendation of existing items to new users. To survive, recommendation systems always need to attract new users as customers. The system should indicate the taste of a new user within a short time. New users may continue to use the system if they find what they are looking for in the recommendations offered by the system. Therefore, the situation defined by Partition 3 is the focus of this paper. Partition 4 indicates recommendations of new items to new users, which is difficult because it represents a case in which there are no historical data for both items and users.

This situation with new users and new items is referred to as the *cold-start problem* [12]. A natural method for solving the cold-start problem to elicit new users' information by having them answer interview questions [9]. The system characterizes the users based on their responses to the questions, thus it can provide appropriate recommendations in the future. The interview process should not be time-consuming. Customers will become impatient and leave the system if they are presented with too many interview questions. Furthermore, the system should construct a rough profile for each new user by asking a limited number of interview questions. The decision tree method has been found to work well for the interview process [4, 5, 14]. In a decision tree, each node represents a question that is determined by certain measurements. They build the best decision tree in the interview process through the optimization of a certain loss function defined in the training step. However, some users who have similar tastes might have different paths because of their different responses to just one question. The decision tree method locally chooses the interview question and groups the users in the nodes. The method only considers the behavior of users in certain nodes. We argue that this method should be used to globally build the framework of the interview process.

2 Related Work

2.1 Collaborative Filtering

The term *Collaborative Filtering* was developed by the author of the first recommendation system, called Tasestry [6]. The system was designed to recommend documents to prevent users from becoming inundated by a huge stream of documents. In recent years, many studies of recommendation systems have focused on collaborative filtering approaches. This method can identify the new user-item association by analyzing the relationships between both users and items. The two primary types of collaborative filtering methods are memory-based [2] and model-based [13]. Memory-based methods compute the relationships between items and users. The item-item approach [8] predicts the rating of a user of an item based on ratings by the same user of neighboring items. Two items are considered to be neighbors if they receive similar ratings by users. Model-based methods are an alternative approach that characterizes users and items by rating data. This approach has become popular in recent studies of collaborative filtering because it can handle large datasets effectively and has good prediction performance. *Matrix Factorization* [7] is one of the most popular approaches in collaborative filtering; it uses a low-dimension vector to represent each user and item and learns the vectors by user-item rating data.

2.2 Cold-Start Collaborative Filtering

A recommendation system knows nothing about the new users because there is no information available. The most direct way to learn the preferences of new users is to ask them for ratings of items. Each interview question asks the user to rate items selected by the system according some criteria. Several studies have focused on the strategies for finding the best items through interview questions. The GroupLens team [9, 11] provides a balanced strategy that considers both the popularity and the entropy of movies based on the selection of interview questions. In a decision tree, each node is an interview question, and users are directed to one of the child nodes according by their responses at the parent node. These responses serve [5, 14] to learn a ternary decision tree based on the interview questions. The decision is created and optimized by analyzing the user rating data. All the tree nodes are items for the interview questions. Users have three choices—“like,” “unlike,” or “unknown”—for their responses to the items in each node on their path. The users are split into three separate groups in each node. Thus, the next question is determined by the user’s response to the current question. Different responses to the current question result in different following questions. New users are characterized after answering the questions from the root node to the leaf nodes. The work in [1] presented a new similarity measure based on neural learning and also shown good results on Netflix and Movielens databases.

3 Our Approach

We first group the training users by observed the rating data. Because the real-world data are sparse, and no user provides ratings for all of the items, we group the users to find virtual users who have similar rating behaviors to the users in same group. When a new user comes into the system, she will be asked for responses to the interview questions. The system then calculates the similarity between the new user and the virtual users according to their responses. Thus, the system can predict the new users' tastes based on the similar virtual users. Our system's flowchart is presented in Fig. 2. We first apply the feature extraction model to determine the latent features of the users and the items, and then we use a question selection model to find the best interview questions by fitting the latent user features.

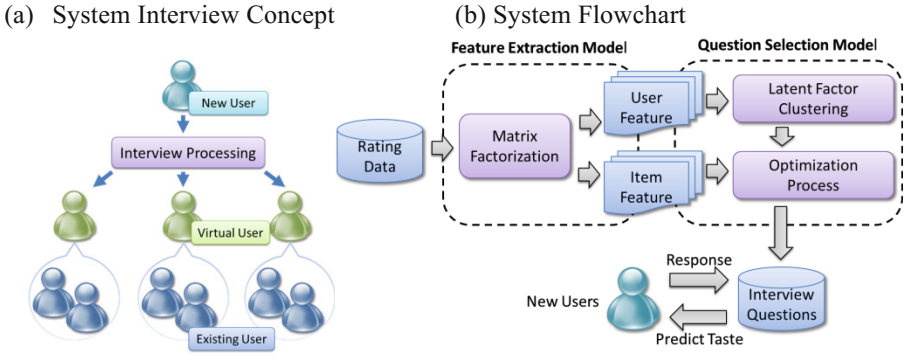

Fig. 2. The system interview concept and flowchart

3.1 Low-Rank Matrix Factorization

Matrix Factorization maps users and items to the latent factor space of dimensionality D. Each user i is associated with a vector $u_i \in \mathbb{R}^D$, and each item j is also associated with a vector $v_j \in \mathbb{R}^D$. For a given item j, the elements of v_j measure the extent of each factor. For a given user i, the elements of u_i measure the interest of each factor. The dot product, $u_i^T v_j$, captures the interaction between user i and item j and approximates the rating r_{ij}, which is user i's rating of item j. Thus, the rating matrix R can be approximated by the product of the user matrix $U \in \mathbb{R}^{D \times M}$ and item matrix $V \in \mathbb{R}^{D \times N}$, which contain M user feature vectors and N item feature vectors, respectively.

$$R \approx U^T V \tag{1}$$

This model is closely related to *singular value decomposition* (SVD), which is a well-known technique in information retrieval. The system learns the models by fitting the observed ratings to predict the unknown ratings. It should avoid overfitting the

observed data by regularizing the learned parameters. Thus, the regularization terms are added to the following equation. The system learns the user and item feature matrix by minimizing the regularized squared error. Here, $\mathbb{O}$ is the set that contains all observed ratings and λ is a constant that controls the extent of the regularization.

$$\underset{U,V}{\operatorname{argmin}} \sum_{r_{ij} \in \mathbb{O}} (r_{ij} - u_i^T v_j)^2 + \lambda(\|u_i\|^2 + \|v_j\|^2) \tag{2}$$

3.2 Latent Factor Clustering

The goal of grouping users is to find several cliques that can represent different tastes of users. Each user has his or her own attributes in the features vector, as described in the in previous section. Thus, we can group users based on these features. The *K-Means Clustering* method is a simple and fast clustering method that has been often used. We apply the *K-Means Clustering* method to group the user features vectors in the training data by minimizing the within-cluster sum of squares, which is define as follows:

$$\underset{c_p}{\operatorname{argmin}} \sum_{p=1}^{C} \sum_{u_i \in \mathbb{C}_p} (u_i - c_p)^2 \tag{3}$$

where c_p is the centroid of the cluster $\mathbb{C}_p$ and C is the number of clusters.

3.3 Question Selection Process

We present an optimization process to ensure that the questions we select can identify the new users' tastes. We first define an objective function:

$$Err(\mathbb{Q}) = \sum_{r_{ij} \in \mathbb{O}} (r_{ij} - \hat{r}_{ij,\mathbb{Q}})^2 \tag{4}$$

where $\mathbb{Q}$ denotes the interview question set. We define the value $Err(\mathbb{Q})$ as the error if we use $\mathbb{Q}$ as the interview questions. Thus, we can calculate the error for each item in the training data. The $\hat{r}_{ij,\mathbb{Q}}$ denote the predicting value for the observed rating r_{ij} after asking the interview questions in $\mathbb{Q}$. We define $\hat{r}_{ij,\mathbb{Q}}$ as the following:

$$\hat{r}_{ij,\mathbb{Q}} = \begin{cases} \frac{\sum_{p=1}^{K} c_p}{K} \cdot v_j & if\ user\ i\ answers\ more\ than\ one\ question \\ u_{avg} \cdot v_j & otherwise \end{cases} \tag{5}$$

where *K* denotes the number of nearest neighbors and c_p is one of cluster centers that approaches the user u_i. We use the average of the choices of the cluster centers by the *K-Nearest Neighbor* (KNN) *method* to prevent overfitting. Next, we calculate the similarity of the answers to the current interview questions to find the k-nearest clusters to the users. The similarity of cluster center c_k and user u_i is defined as follows:

$$sim(c_p, u_i) = \sum_{v_j \in \mathbb{Q}_t} I_{ij}(c_p v_j - r_{ij})^2 \tag{6}$$

where $\mathbb{Q}_t$ is the current interview question set and v_j is one of the items in $\mathbb{Q}_t$. I_{ij} equals 1 if $r_{ij} \in \mathbb{Q}$ and 0 otherwise. It is a constant that indicates whether the user u_i has a rating to item v_j. Finally, we select the interview question by considering the error of the current and previous questions. We choose an item to be an interview questions because it has minimum error when this item and the previous item we select are both interview questions. The following equation shows how we choose the interview questions:

$$q_h = \underset{j \in \mathbb{S}}{argmin}\{ Err(\mathbb{Q}_{q_h=j})\} \tag{7}$$

where q_h denotes the $\underline{h}$'th interview question and $\mathbb{Q}_{q_h=j}$ means item j is the $\underline{h}$'th interview question in $\mathbb{Q}$. $\mathbb{S}$ denotes a set that contains a number of items. The following section describes how we select the items in $\mathbb{S}$.

3.4 Question Selection Strategies

The popularity of an item indicates how familiar the users are with it. The more ratings the item receives, the more popular it is considered. Thus, we define these items as popular items. The advantage of choosing the most popular items to be interview questions is that new users usually provide answers. However, popular items are not effectively characterized by users. We cannot indicate the users' tastes by their responses to an item that nearly everyone liked. Users often give a high rating to such popular items. We cannot identify users' tastes by simply referencing the users' responses to popular items. The contention of the item indicates how widely the receiving ratings spread. The more widely the ratings spread, the more controversial the item is. Prior works [10] define the entropy of item to indicate how controversial it is, as defined by the following equation:

$$\text{Entropy}(v_j) = -\sum_{s} p_s \lg(p_s) \tag{8}$$

where p_s denotes the fraction of ratings of item v_j that equal s. For example, an item v_j has been rated by 1000 people, while there are a total of 2000 people in the data set. In the 1000 ratings data, there are 100 ratings equal to 1, 100 ratings equal to 2, 200 ratings equal to 3, 300 ratings equal to 4 and 300 ratings equal to 5. The $p_s, 1 \leq s \leq 5$ are 100/1000, 100/1000, 200/1000, 300/1000, 300/1000, respectively. The same amount of ratings for each score leads to the maximum entropy, while all ratings having the same score leads to the minimum entropy.

Popular items usually are not controversial, while items that have widely spread ratings are often not popular enough. [9] uses a balance strategy to consider both popularity and contention at the same time. It ranks the items by the product of entropy and the log of popularity. It takes the log of popularity because the number of ratings of items is an exponential distribution. Popularity will dominate the score if it does not take the log.

$$(\log Popularity(v_j)) \times \text{Entropy}(v_j) \tag{9}$$

4 Experiment

4.1 Data Set

We used the Movielens dataset for our experiment, which is a public dataset that can be downloaded from GroupLens[4] website. The data set contains a total of 1,000,209 rating data items from 6,040 users on 3,951 movies.

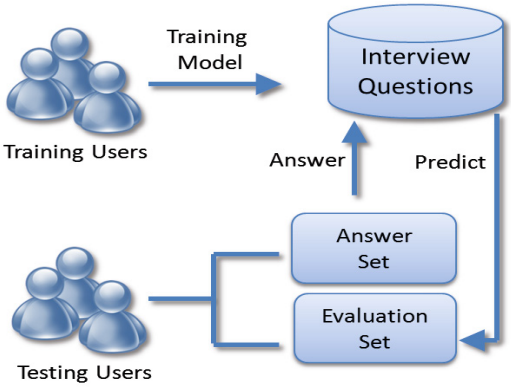

Fig. 3. System evaluation process

We split the dataset into two separate sets: the training set and the testing set. The training set contains 75% of all users, and the testing set contains 25% of the users. In the testing set, we split the users' ratings into an answer set and an evaluation set, which contain 75% and 25% of the ratings, respectively. The answer set is used to

[4] http://www.grouplens.org/node/73/

generate the user response for the interview questions. Each tested user answers the interview question based on the ratings in the answer set and answers "Unknown" if there is no such rating in the answer set. The evaluation set is used to evaluate the performance after the interview process. The system characterizes each new user after the interview process and predicts the ratings in the evaluation set. Then, the RMSE mentioned previously is calculated to evaluate the performance.

4.2 Evaluation

Root mean square error (RMSE) measurement has been widely used to evaluate the performance of collaborative filtering algorithm. The RMSE formula is defined as follows:

$$\text{RMSE} = \sqrt[2]{\frac{\sum_{r_{ij} \in T} (r_{ij} - \hat{r}_{ij})^2}{|T|}} \quad (10)$$

where T is the set of test ratings, r_{ij} is the ground-truth rating of user i on movie j and $\hat{r}_{ij}$ is the value predicted by models. The smaller value of RMSE means better performance of the model because the prediction ratings are approaching the ground-truth rating.

4.3 Comparisons

Because MFK methods predict the rating by finding the neighbors of feature clusters according to the responses to interview questions, we compare two types of baseline to our approach. The first type of baseline is the method of predicting ratings. We use three baselines to predict the new users' ratings in the evaluation set. The second baseline is the strategy used to select the interview questions. We use different strategies to determine the interview questions and predict the ratings by finding the neighbor cluster according to the response.

- *Global Average:*
  We do not characterize each new user by the interview questions. Instead, we return only the global average rating (i.e., 3.58) to be the prediction of new users' ratings.

- *User Average:*
  For each tested user, we return rating $\hat{r}_i$, as defined in the following, to predict the observed rating in the evaluation set.

- *Item Average:*
  We predict the observed rating r_{ij} by the average received rating of item j.

Table 1 shows the performance of our approach compared with baselines. Note that $MFK_{Balance}$ in this case is the performance we show for seven interview questions, and 0.9583 is the predicted performance of users who answer zero to the seven interview questions.

Table 1. RMSE for different methods

Global Average	1.1072
User Average	1.0299
Item Average	0.9745
$MFK_{Balance}$	0.9583

The following strategies are our approaches, which combine the selection criteria and optimization processes we described in Section 3.

- $MFK_{Popularity}$.
  We rank all items by their popularity and include the top 100 items in the question set. Then, we determine the interview questions by using the optimizing process described in Section 3.3.

- $MFK_{Convention}$.
  We rank all items by their entropy score and include the top 100 items in the question set. Then, we determine the interview questions by using the optimizing process described in Section 3.3.

- $MFK_{Balance}$.
  We rank all items by the score defined in Equation (9) and include the top 100 items in the question set. Then, we determine the interview questions by using the optimizing process described in Section 3.3.

In Table 2, the performance of the popularity strategy shows a substantial improvement when we show seven questions. However, the entropy strategy does not improve when the number of shown questions increases. Fig. 4 shows that more than 90% of users do not give any response to interview questions that have high entropy scores. For almost all the interview questions, the system cannot indicate users' tastes correctly through the "Unknown" response. The balance method shows better performance than the popularity methods. This finding indicates that considering the entropy of an item is a way to improve the performance. In our approach, we consider both the popularity and the entropy of items, and we also implement an optimization process to ensure that we obtain the best performance compared with other strategies.

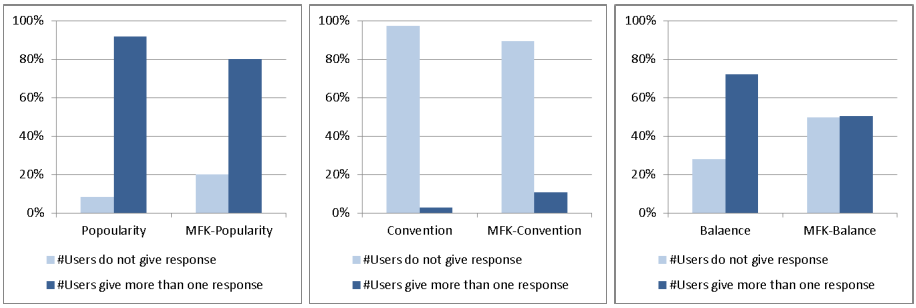

Fig. 4. The ratio of non-responses to users who provide more than one response

Table 2. Comparison with baselines

#Shown Questions	3	4	5	6	7
Random	0.9808	0.9812	0.9812	0.9812	0.9812
Popularity	0.9902	0.9847	0.9766	0.9722	0.9662
Convention	0.9833	0.9839	0.9833	0.9839	0.9837
Balance	0.9869	0.9825	0.9788	0.9671	0.9634
$MFK_{Popularity}$	0.9822	0.9820	0.9828	0.9743	0.9628
$MFK_{Convention}$	0.9839	0.9884	0.9892	0.9832	0.9892
$MFK_{Balance}$	0.9674	0.9614	0.9604	0.9591	0.9583

4.4 Impact of Parameters

Fig. 5 and Fig. 6 show the different numbers of neighbors in certain clusters. We will discuss the effect of the number of clusters and the number of user neighbors selected by their response to interview questions.

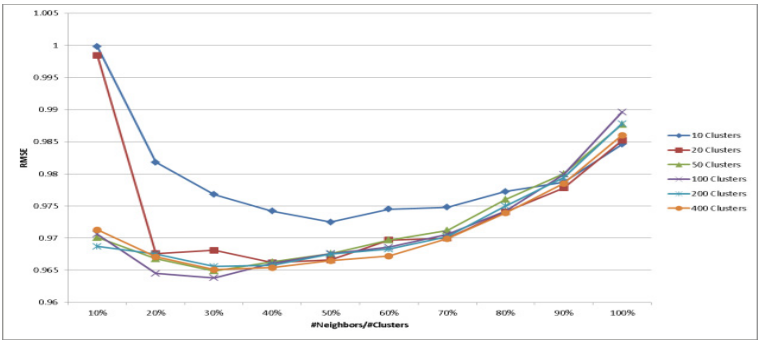

Fig. 5. RMSE in different numbers of neighbors with certain clusters

Fig. 5 shows the different numbers of neighbors in certain clusters. We calculate the RMSE by the different ratios of neighbors for different number of clusters. We observe good performance when we set 20% or 30% of the number of clusters as the number of neighbors. In the case of 100% clusters, the number of neighbors equals the number of clusters, indicating that the prediction of ratings is based on the average of all the clusters. Thus, the RMSEs are similar. When we consider only the smaller number of neighbors for the new users, we obtain bad performance because we consider a small amount of information.

We cluster users in different numbers of groups, as shown in Fig. 6. The figure reveals that we obtain better performance when we set larger numbers of clusters. In large numbers of clusters, users can find their neighbors more accurately because there are more different numerical responses to the interview questions.

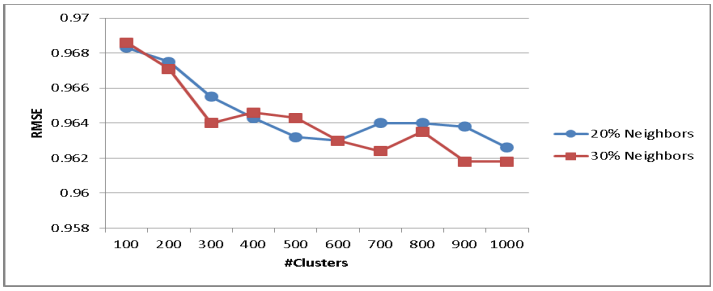

Fig. 6. RMSE in different numbers of clusters

5 Conclusions

In this paper, we proposed the MFK algorithm, which combines user feature selection and interview question optimization to address the new user problem. We can determine the best interview questions and extract the latent features by minimizing the objective functions. We first use the feature extraction model to extract the latent features of users and items. With the latent features, we can address the missing values in user-item rating data and indicate the existing users' tastes. After clustering the user features, we can ensure that the users in the same cluster are similar. We can indicate the new users' tastes according to their responses to the interview questions, which we learn through an optimization process.

References

[1] Bobadilla, J., Ortega, F., Hernando, A., Bernal, J.: A collaborative filtering approach to mitigate the new user cold start problem. Knowledge-Based Systems 26, 225–238 (2012)
[2] Connor, M., Herlocker, J.: Clustering items for collaborative filtering. In: Proceedings of the ACM SIGIR Workshop on Recommender Systems, SIGIR 1999 (1999)

[3] Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. Presented at the Proceedings of the 2010 IEEE International Conference on Data Mining (2010)
[4] Golbandi, N., Koren, Y., Lempel, R.: On bootstrapping recommender systems. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, Toronto, ON, Canada, pp. 1805–1808 (2010)
[5] Golbandi, N., Koren, Y., Lempel, R.: Adaptive bootstrapping of recommender systems using decision trees. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, Hong Kong, China, pp. 595–604 (2011)
[6] Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM 35, 61–70 (1992)
[7] Koren, Y., Bell, R., Volinsky, C.: Matrix Factorization Techniques for Recommender Systems. Computer 42, 30–37 (2009)
[8] Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing 7, 76–80 (2003)
[9] Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., et al.: Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, San Francisco, California, USA, pp. 127–134 (2002)
[10] Rashid, A.M., Karypis, G., Riedl, J.: Learning preferences of new users in recommender systems: an information theoretic approach. SIGKDD Explor. Newsl. 10, 90–100 (2008)
[11] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, United States, pp. 175–186 (1994)
[12] Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, pp. 253–260 (2002)
[13] Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. in Artif. Intell. 2009, 2 (2009)
[14] Zhou, K., Yang, S.-H., Zha, H.: Functional matrix factorizations for cold-start recommendation. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, Beijing, China, pp. 315–324 (2011)