An Efficient Incremental Collaborative Filtering System

Aghiles Salah(✉), Nicoleta Rogovschi, and Mohamed Nadif

LIPADE, University of Paris Descartes, 45, Rue des Saints-Peres, Paris, France
{Aghiles.Salah,Nicoleta.Rogovschi,Mohamed.Nadif}@parisdescartes.fr

Abstract. Collaborative filtering (CF) systems aim at recommending a set of personalized items for an active user, according to the preferences of other similar users. Many methods have been developed and some, such those based on *Similarity* and *Matrix Factorization (MF)* can achieve very good recommendation accuracy, but unfortunately they are computationally prohibitive. Thus, applying such approaches to real-world applications in which available information evolves frequently, is a non-trivial task. To address this problem, we propose a novel efficient incremental CF system, based on a weighted clustering approach. Our system is able to provide a high quality of recommendations with a very low computation cost. Experimental results on several real-world datasets, confirm the efficiency and the effectiveness of our method by demonstrating that it is significantly better than existing incremental CF methods in terms of both scalability and recommendation quality.

Keywords: Collaborative filtering · Recommender systems · Clustering

1 Introduction

In order to filter large amounts of information, recommender systems (RSs) have been adopted by many real-world applications such as Amazon [8] and Netflix [7]. Collaborative filtering (CF) is the most often used approach in RSs. It consists in predicting the items that an active user will enjoy, based on the items that the people who are most similar to this user have enjoyed. Among CF systems two distinct types of approach are to be found:

Memory-based CF. These approaches are based on computing similarities [1]. User-based collaborative filtering looks for similarities between the active user $\mathbf{u}_a$ and all other users and tries to predict the preference of $\mathbf{u}_a$ for a set of new items, according to the preferences of the K most similar users to $\mathbf{u}_a$. Item-based collaborative filtering consists in finding the K nearest neighbors of each item and making recommendations according to the neighborhood of items enjoyed by the user $\mathbf{u}_a$.

Model-based CF. These approaches begin by suggesting a model that will learn from the user/item rating matrix $\mathbf{U}$ in order to capture the hidden features

S. Arik et al. (Eds.): ICONIP 2015, Part III, LNCS 9491, pp. 375–383, 2015.
DOI: 10.1007/978-3-319-26555-1_42

of the users and items. They then predict the missing ratings according to this model. Many model-based CF techniques have been proposed, the most popular being those based on clustering [12], co-clustering [4], and matrix factorization MF [2,11].

Traditional CF approaches, such as matrix factorization (MF) and memory-based methods, can achieve good prediction accuracy, but their computation time rises steeply as the number of users and items increases. Further, these methods need to be performed periodically (i.e., offline) in order to take into account new ratings, new users and items. However, with this strategy, new information which appear between two offline computations are not considered. As a result, applying traditional CF techniques to real-world applications such as Netflix in which the sets of users, items and ratings are frequently updated, remains therefore a challenge.

To overcome the problem of computation time, incremental CF systems have been proposed. The most popular are incremental CF based on MF approaches [5,10], incremental CF based on co-clustering [4,6], and incremental memory-based CF, including user [9] and item [13] based approaches. All these efforts have demonstrated the effectiveness of developing incremental models to provide scalable collaborative filtering systems. But often, these will significantly reduce the quality of recommendations. Further, most of these approaches (except memory-based CF) do not handle all possible dynamic scenarios (i.e., submission of new ratings, update of existing ratings, appearance of new users and new items). For instance incremental CF based on singular value decomposition [10], do not treat the two first scenarios.

In this paper we focus on the problem of computation time in CF systems. In order to overcome this drawback we propose a novel incremental CF approach, which is based on a weighted version of the online spherical k-means algorithm: OSK-means [14]. Our method is able to handle in a very short time the frequent changes in CF data; including the submission of new ratings, the update of existing ratings, the appearance of new users and items. Below, we summarize the key contributions we make in this paper.

- We derive a novel efficient CF system, based on a weighted clustering approach.
- In order to handle frequent changes in CF data, we design incremental updates, which allow to efficiently treat submissions of new ratings, updates of existing ratings, and occurrence of new users and items.

Numerical experiments validate our approach. The results on several real datasets show that our method outperforms significantly state-of-the-art incremental methods in terms of scalability and recommendation quality.

The rest of this paper is organized as follows. Section 2 introduces the formalism of traditional OSK-means. Section 3 provides details about the weighted version of OSK-means and our CF system: training, prediction and incremental training steps. Section 4 presents the results obtained on real-world datasets, in terms of recommendation quality and computation time. Finally, the conclusion summarizes the advantages of our contribution.

2 Online Spherical K-Means

In this paper matrices are denoted with boldface uppercase letters and vectors with boldface lowercase letters. Given a matrix $\mathbf{U} = (u_{ij})$ of size $n \times p$, the ith row (user) of this matrix is represented by a vector $\mathbf{u}_i = (u_{i1}, \ldots, u_{ip})^T$, where T denotes the transpose. The column j corresponds to the jth item. The partition of the set of rows into K clusters can be represented by a classification matrix $\mathbf{z}$ of elements z_{ik} in $\{0,1\}^K$ satisfying $\sum_{k=1}^{K} z_{ik} = 1$. The notation $\mathbf{z} = (z_1, \ldots, z_n)^T$, where $z_i \in \{1, \ldots, K\}$ represents the cluster of i, will be also used.

Before describing OSK-means, we will first introduce spherical K-means (SK-means). The SK-means [3] algorithm is a K-means algorithm in which the objects (users) $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are assumed to lie on a hypersphere. SK-means originally maximizes the sum of the dot product between the elements of the data points and the K means directions characterizing the clusters. This is equivalent to maximizing the sum of the cosine similarity of the normalized data. The algorithm then maximizes the following objective function:

$$L = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \cos(\mathbf{u}_i, \boldsymbol{\mu}_k) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \mathbf{u}_i^T \boldsymbol{\mu}_k, \tag{1}$$

where $z_{ik} \in \{0,1\}$; $z_{ik} = 1$ if $\mathbf{u}_i \in k^{th}$ cluster, $z_{ik} = 0$ otherwise. SK-means repeats the following two steps:

- For $i = 1, \ldots, n$, assign $\mathbf{u}_i$ to the k^{th} cluster, where $z_i = \arg\max_k \left(\mathbf{u}_i^T \boldsymbol{\mu}_k\right), k = 1, \ldots, K$.
- Calculate $\boldsymbol{\mu}_k = \frac{\sum_{i,k} z_{ik} \mathbf{u}_i}{||\sum_{i,k} z_{ik} \mathbf{u}_i||}$.

OSK-means [14] uses competitive learning (Winner-Takes-All strategy) to minimize the objective function (1), which leads to

$$\boldsymbol{\mu}_k^{new} = \frac{\boldsymbol{\mu}_k + \eta \frac{\partial L_i}{\partial \boldsymbol{\mu}_k}}{||\boldsymbol{\mu}_k + \eta \frac{\partial L_i}{\partial \boldsymbol{\mu}_k}||} = \frac{\boldsymbol{\mu}_k + \eta \mathbf{u}_i}{||\boldsymbol{\mu}_k + \eta \mathbf{u}_i||},$$

where η is the learning rate, $\boldsymbol{\mu}_k$ is the closest centroid to the object $\mathbf{u}_i$, and L_i denotes $\sum_{k=1}^{K} z_{ik} \mathbf{u}_i^T \boldsymbol{\mu}_k$.

In the OSK-means method, each centroid is updated incrementally with a learning rate η. Zhong [14] proposed an exponentially decreasing learning rate $\eta^t = \eta_0 (\frac{\eta_f}{\eta_0})^{\frac{t}{n \times B}}$, where $\eta_0 = 1.0$, $\eta_f = 0.01$, B is the number of batch iterations and $t, (0 \leq t \leq n \times B)$ is the current iteration.

3 Efficient Incremental Collaborative Filtering System (EICF)

In this section we describe our collaborative filtering system EICF, designed to provide a high quality of recommendations with a very low computation cost. This system can be divided into three main steps: training, prediction, and incremental training. The different steps are described as follows:

3.1 Training Step

This step, consists in clustering the users into K groups. Unfortunately the traditional OSK-means which has been proposed in the context of documents clustering, is not adapted for CF data. Unlike text data, the sparsity in CF is caused by unknown ratings, which requires a different handling than if the sparsity is caused by entries of "zero". To address this problem, we propose novel version of OSK-means which is more suitable for CF. It consists in introducing user weights, in order to tackle the sparsity problem; by giving more importance for users who provided many ratings. Thereby, the resulting clusters will be highly influenced by the most useful users (i.e., users with high weights). Below we give more details about this weighted version of OSK-means. Let w_i denote the weight of the i^{th} user, the weighted objective function of the SK-means is given by:

$$L^w = \sum_{i=1}^{n} L_i^w, \text{ where } L_i^w = \sum_{k=1}^{K} w_i z_{ik} \mathbf{u}_i^T \boldsymbol{\mu}_k, \tag{2}$$

Thus, the corresponding update centroid for the weighted OSK-means is given by:

$$\boldsymbol{\mu}_k^{new} = \frac{\boldsymbol{\mu}_k + \eta \frac{\partial L_i^w}{\partial \boldsymbol{\mu}_k}}{||\boldsymbol{\mu}_k + \eta \frac{\partial L_i^w}{\partial \boldsymbol{\mu}_k}||} = \frac{\boldsymbol{\mu}_k + \eta w_i \mathbf{u}_i}{||\boldsymbol{\mu}_k + \eta w_i \mathbf{u}_i||}, \tag{3}$$

We now give an intuitive formulation of user weights. Let $\mathbf{M} = (m_{ij})$ be an $(n \times p)$ binary matrix, such that $m_{ij} = 1$ if the rating u_{ij} is available, and $m_{ij} = 0$ otherwise. Its i^{th} row corresponds to a vector $\mathbf{m}_i = (m_{i1}, \ldots, m_{ip})^T$ indicating which items have been rated by the i^{th} user. Thus, we define the weight of the i^{th} user to be proportional to the number of his available ratings as follows:

$$w_i = (\mathbf{m}_i^T \mathbb{1}) \times \sigma(\mathbf{u}_i) \tag{4}$$

where $\mathbb{1}$ is the vector of the appropriate dimension which all its values are 1, and $\sigma(\mathbf{u}_i)$ denotes the standard deviation of ratings provided by $\mathbf{u}_i$. We consider the standard deviation in order to give less importance for users who provide only low ratings or similarly, only high ratings (i.e., users who expressed the same preference for all items they have rated). Algorithm algotrain describes in more details our training step.

3.2 Prediction Step

In this step, unknown ratings are predicted according to the clustering results. However, it is difficult to make consistent predictions, even when the best clustering results are achieved, because there are so many unknown ratings in $\mathbf{U}$. To overcome this difficulty we propose to estimate unknown ratings by a weighted average of observed ratings, as follows:

$$u_{aj} = \frac{\sum_{i=1}^{n} w_i z_{ik} \mathbf{u}_i^T \boldsymbol{\mu}_k \times u_{ij}}{\sum_{i=1}^{n} w_i z_{ik} \mathbf{u}_i^T \boldsymbol{\mu}_k}, \tag{5}$$

Algorithm 1. EICF training.

Input: n users $\mathbf{u}_i$ in $\mathbb{R}^p$, K is the number of clusters;
Output: K centers $\boldsymbol{\mu}_k$ in $\mathbb{R}^p$, and $\mathbf{z} = (z_1, \ldots, z_n)$;
Steps:
1. Normalize the users (i.e., $\mathbf{u}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$, $\forall i$)
2. Compute user weights: $w_i = (\mathbf{m}_i^T \mathbb{1}) \times \sigma(\mathbf{u}_i)$;
3. Initialization: random initialization of the partition $\mathbf{z}$, $t = 1$;
4. Estimation of the initial centroids:

$$\mu_{kj} = \frac{\sum_i m_{ij} w_i z_{ik} u_{ij}}{\| \sum_i w_i z_{ik} \mathbf{u}_i \|}$$

for $b = 1$ **to** B **do**
  for each $\mathbf{u}_i$ **in U do**
    5. User assignment: compute $z_i = \arg\max_k (w_i \mathbf{u}_i^T \boldsymbol{\mu}_k)$;
    6. Centroid update: compute the winner centroid by $\hat{\boldsymbol{\mu}}_{z_i} = \frac{\boldsymbol{\mu}_{z_i} + \eta w_i \mathbf{u}_i}{\|\boldsymbol{\mu}_{z_i} + \eta w_i \mathbf{u}_i\|}$;
    $t = t + 1$;
  end for
end for

Let $\mathbf{u}_a$ denote the active user, $k = z_a$. The key idea behind this strategy is to weight the available ratings u_{ij} according to the similarity between each user $\mathbf{u}_i$ and its corresponding centroid $\boldsymbol{\mu}_k$, and to weight by w_i, in order to give greater importance for users closest to their centroid, and respectively to give more importance for ratings provided by most important users.

The prediction Eq. (5) is attractive because it depends only on the clustering results, which means that it can be performed offline and stored in a $(K \times p)$ matrix $\mathbf{P}$, which leads to very short prediction times.

3.3 Incremental Training Step

In the sequel, we design incremental updates, in order to handle the frequent changes in CF data. Thus, we can distinguish four main situations: (1) submission of new ratings, (2) update existing ratings, (3) appearance of new users, (4) appearance of new items. In the following, we give the update formulas for each situation.

Submission of a new rating. Let $\mathbf{u}_a$ denote an active user who submits a new rating for an item j. The equations below, give the different incremental updates to perform in this case.

- Update the norm of $\mathbf{u}_a$: $\|\mathbf{u}_a^+\| = \sqrt{\|\mathbf{u}_a\|^2 + u_{aj}^2}$
- For each k, update the similarity between $\mathbf{u}_a$ and $\boldsymbol{\mu}_k$:

$$cos(\mathbf{u}_a^+, \boldsymbol{\mu}_k) = \frac{1}{\|\mathbf{u}_a^+\|} [\|\mathbf{u}_a\| \times \mathbf{u}_a^T \boldsymbol{\mu}_k + u_{aj} \mu_{kj}],$$

– Update the weight of the active user: $\hat{w}_a = (\frac{w_a}{\sigma(\mathbf{u}_a)} + 1) \times \sigma(\mathbf{u}_a^+)$
– Update the assignment of $\mathbf{u}_a$: $\hat{z}_a = \arg\max_k cos(\mathbf{u}_a^+, \boldsymbol{\mu}_k)$.
– Update the corresponding centroid $\boldsymbol{\mu}_{\hat{z}_a}$, by using formula (3) where

$$\sigma(\mathbf{u}_a^+)^2 = \frac{N_a \times (\sigma(\mathbf{u}_a)^2 + \bar{u}_a^2) + u_{aj}^2}{N_a + 1} - \left(\frac{N_a \bar{u}_a + u_{aj}}{N_a + 1}\right)^2,$$

thanks to König-Huygens formula, i.e., $\sigma(\mathbf{u}_a) = \sqrt{\frac{1}{N_a}\sum_j u_{aj}^2 - \bar{u}_a^2}$. The notation $\mathbf{u}_a^+$ denotes the active user $\mathbf{u}_a$ with the new rating u_{ij} available, N_a and $\bar{u}_a$ denote respectively, the number of ratings and the average rating of $\mathbf{u}_a$ before evaluating item j. Note that, as the centroids are stable at the end of training, the two latter incremental updates concerning the assignment of $\mathbf{u}_a$, do not need to be performed after each one new rating.

Update an existing rating. In this case, the active user updates an existing rating for an item j. As for the submission of a new rating, the main updates are summarized below.

– Update the norm of $\mathbf{u}_a$: $\|\mathbf{u}_a^+\| = \sqrt{\|\mathbf{u}_a\|^2 - u_{aj}^2 + \hat{u}_{aj}^2}$
– For each k, update the similarity between $\mathbf{u}_a$ and $\boldsymbol{\mu}_k$:

$$cos(\mathbf{u}_a^+, \boldsymbol{\mu}_k) = \frac{1}{\|\mathbf{u}_a^+\|}[\|\mathbf{u}_a\| \times \mathbf{u}_a^T \boldsymbol{\mu}_k - u_{aj}\mu_{kj} + \hat{u}_{aj}\mu_{kj}]$$

– Update the weight of the active user: $\hat{w}_a = \frac{w_a}{\sigma(\mathbf{u}_a)} \times \sigma(\mathbf{u}_a^+)$
– Update the assignment of $\mathbf{u}_a$: $\hat{z}_a = \arg\max_k cos(\mathbf{u}_a^+, \boldsymbol{\mu}_k)$.
– Update the corresponding centroid $\hat{z}_a$, by using Eq. (3) where

$$\sigma(\mathbf{u}_a^+)^2 = \left(\sigma(\mathbf{u}_a)^2 + \bar{u}_a^2 + \frac{\hat{u}_{aj}^2 - u_{aj}^2}{N_a}\right) - \left(\bar{u}_a + \frac{\hat{u}_{aj} - u_{aj}}{N_a}\right)^2,$$

$\hat{u}_{aj}$ denotes the new value substituted for the existing rating u_{aj}, and the notation $\mathbf{u}_a^+$ represents the active user after updating the known rating u_{aj}.

Appearance of new user. In this situation, a new user is incorporated into the model in real time. Let $\hat{\mathbf{u}}_a$ denote a new user. The model is incremented as follows:

– Compute the weight of $\hat{\mathbf{u}}_a$, by using Eq. (4).
– Assign $\hat{\mathbf{u}}_a$ to k^{th} cluster where $k = \arg\max_{1 \le k \le K}(\frac{\hat{\mathbf{u}}_a^T \boldsymbol{\mu}_k}{\|\hat{\mathbf{u}}_a\|})$.
– Update the corresponding centroid: $\hat{\boldsymbol{\mu}}_k = \frac{\boldsymbol{\mu}_k + \eta w_a \frac{\hat{\mathbf{u}}_a}{\|\hat{\mathbf{u}}_a\|}}{\|\boldsymbol{\mu}_k + \eta w_a \frac{\hat{\mathbf{u}}_a}{\|\hat{\mathbf{u}}_a\|}\|}$.

Appearance of new item. When a new item appears, it has no ratings, so there nothing to change in the model. When a new item starts receiving ratings, handling new item, reduces to handling the submission of new ratings.

4 Experimental Results

Hereafter, we propose to evaluate the performances of our CF system on three real-world datasets. The first is *MovieLens*[1] *(ML-1M)*, consisting of 1,000,209 ratings provided by 6040 users for 3952 movies (only 4.2 % of ratings are observed). The second is *MovieLens (ML-100k)*, containing 100,000 ratings given by 943 users for 1664 movies. The proportion of observed ratings in this dataset is 6.4 %. The last dataset is Epinions[2], with 664,824 ratings from 49,290 users on 139,738 items (movies, musics, electronic products, books, ...). The Epinion dataset is more than 99 % sparse.

We compare EICF with several popular methods, namely: incremental user-based CF IUCF [9], incremental item-based CF IICF [13], and incremental CF based on co-clustering COCLUST [4]. All the evaluations are made under the same machine (OS: ubuntu 14.04 LTS 64-bit, Memory: 16 GiB, Processor: Intel® Core™ i7-3770 CPU @ 3.40 GHz × 8). To evaluate our CF system we focus on the quality of the recommendations and computational time. Thus, we chose the F-measure F1 [1] as the evaluation metric. Unlike accuracy metrics such as mean average error and root means square error, the F-measure allows to evaluate the quality of the set of recommendations [1], which is more relevant in the context of CF. The results reported in Table 1 are obtained as follows: (1) We generate ten random training-test (80–20 %) sets from each dataset. (2) Users in test sets are considered as new ones, and are incorporated incrementally. (3) Finally, we report the average F-measure for each method, over different recommendation lists (i.e., containing 10, 25 and 40 items). We also report the average computation time required by each method, for incorporating and generating recommendations, for users from the test sets. Note that, in terms of computation time, IICF is favoured in this comparison; unlike the other methods, incorporating new users is not the most expensive computation for this approach.

From Table 1, we note that our method provides a high quality of recommendations, thanks to our strategy for alleviating the sparsity problem; by introducing user weights. In fact, our CF system EICF exhibits the best quality of recommendations, over all datasets. Moreover, from Table 1 we observe that EICF requires much less time for handling new information and generating recommendations, than the other incremental methods, including IICF although it is favoured. This performance rises significantly as the volume of data increases. In fact, contrary to the other methods, the complexity of EICF does not depend on the number of users and items, as reported in Table 2. Therefore, EICF is more suitable than the other incremental methods, for real world-applications involving large databases in which users, items and ratings are frequently updated. Note that, the computation time of COCLUST reported in Table 1 is high, even

[1] http://grouplens.org/datasets/movielens/.
[2] http://www.epinions.com.

Table 1. Comparison of several CF systems in terms of F1 and computation time

			CF methods			
Datasets		Recom. lists	COCLUST	IICF	IUCF	EICF
ML-100k	F1	10	0.02	0.10	0.16	**0.21**
		25	0.06	0.16	0.22	**0.30**
		40	0.05	0.10	0.24	**0.33**
	Comp. time (s)		0.97	0.88	1.78	**0.51**
ML-1M	F1	10	0.2e-4	0.04	0.10	**0.14**
		25	0.02	0.08	0.15	**0.21**
		40	0.05	0.11	0.17	**0.25**
	Comp. time (s)		11.85	7.96	138.1	**2.86**
Epinion	F1	10	0.45e-4	0.008	0.022	**0.038**
		25	0.38e-4	0.011	0.029	**0.043**
		40	0.73e-4	0.012	0.032	**0.046**
	Comp. time (s)		212.71	927.22	4041.01	**149.20**

Table 2. Comparison of computational times (in the worst case) in various situations. W^* denotes the number of observed ratings in $\mathbf{U}$. K and L are the number of row and column clusters, K also denotes the number of neighbours for memory CF (IUCF, IICF). p^* is the number of observed ratings for a new user, n^* denotes the number of available ratings for a new item. Finally, B denotes the number of iterations

Algorithm	Static training	Prediction	Inc. train
IUCF	$O(nW^*)$	$O(K)$	$O(np^*)$
EICF	$O(BKW^*)$	$O(1)$	$O(Kp^*)$
COCLUST	$O(BW^* + nBKL + pBKL)$	$O(1)$	$O(p^*)$
IICF	$O(pW^*)$	$O(p^*)$	$O(pn^*)$

if its complexity in the dynamic situation (i.e., inc. train: $O(p^*)$) might appear attractive. The reason is that, this approach provides only partial updates, and the co-clustering is performed periodically to completely incorporate new information.

5 Conclusion

We presented EICF, a novel efficient and effective incremental CF system, which is based on a weighted clustering approach. To achieve high quality of recommendations, we introduced user weights into the clustering process, to lessen the effect of users who provided only few ratings. In order to address the computational time problem, we designed incremental updates, which allows our system to handle in a very short time, the frequent changes in CF data; such as submissions of new ratings, appearance of new users and items. Numerical experiments

on real-world datasets demonstrate the efficiency and the effectiveness of our method which provides better quality of recommendations than existing incremental CF systems, while requiring less computation time. Thus, our CF system is more suitable than existing incremental approaches, for real-world applications involving huge databases, in which available information (i.e., users, items and ratings) changes frequently. For future work, we will investigate other strategies for handling the sparsity problem in CF, and try to develop a parallel version of EICF, that can support distributed computations.

References

1. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl. Based Syst. **46**, 109–132 (2013)
2. Delporte, J., Karatzoglou, A., Matuszczyk, T., Canu, S.: Socially enabled preference learning from implicit feedback data. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part II. LNCS, vol. 8189, pp. 145–160. Springer, Heidelberg (2013)
3. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Mach. Learn. **42**(1–2), 143–175 (2001)
4. George, T., Merugu, S.: A scalable collaborative filtering framework based on co-clustering. In: Fifth IEEE International Conference on Data Mining, pp. 625–628 (2005)
5. Han, S., Yang, Y., Liu, W.: Incremental learning for dynamic collaborative filtering. J. Softw. **6**, 969–976 (2011)
6. Khoshneshin, M., Street, W.N.: Incremental collaborative filtering via evolutionary co-clustering. In: RecSys, pp. 325–328 (2010)
7. Koren, Y.: The Bellkor solution to the Netflix grand prize (2009)
8. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**, 76–80 (2003)
9. Papagelis, M., Rousidis, I., Plexousakis, D., Theoharopoulos, E.: Incremental collaborative filtering for highly-scalable recommendation algorithms. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 553–561. Springer, Heidelberg (2005)
10. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: ICIS, pp. 27–28 (2002)
11. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.: Application of dimensionality reduction in recommender system - a case study. In: ACM WEBKDD Workshop (2000)
12. Ungar, L.H., Foster, D.P.: Clustering methods for collaborative filtering. In: AAAI Workshop on Recommendation Systems, vol. 1, pp. 114–129 (1998)
13. Yang, X., Zhang, Z., Wang, K.: Scalable collaborative filtering using incremental update and local link prediction. In: CIKM, pp. 2371–2374 (2012)
14. Zhong, S.: Efficient online spherical k-means clustering. In: IJCNN, pp. 3180–3185 (2005)