Published in final edited form as:
Proc Congr Evol Comput. 2008 June 1; 2008: 2725–2733. doi:10.1109/CEC.2008.4631164.

A New Adaptive Framework for Collaborative Filtering Prediction

Ibrahim A. Almosallam[Computer science graduate student] and
University of Missouri, Columbia, MO 65211 USA (phone: 646-359-9635

Yi Shang[Computer science professor]
University of Missouri, Columbia, MO 65211 USA. (phone: 573- 884-7794

Ibrahim A. Almosallam: cs.ibrahim@gmail.com; Yi Shang: shangy@missouri.edu

Abstract

Collaborative filtering is one of the most successful techniques for recommendation systems and has been used in many commercial services provided by major companies including Amazon, TiVo and Netflix. In this paper we focus on memory-based collaborative filtering (CF). Existing CF techniques work well on dense data but poorly on sparse data. To address this weakness, we propose to use z-scores instead of explicit ratings and introduce a mechanism that adaptively combines global statistics with item-based values based on data density level. We present a new adaptive framework that encapsulates various CF algorithms and the relationships among them. An adaptive CF predictor is developed that can self adapt from user-based to item-based to hybrid methods based on the amount of available ratings. Our experimental results show that the new predictor consistently obtained more accurate predictions than existing CF methods, with the most significant improvement on sparse data sets. When applied to the Netflix Challenge data set, our method performed better than existing CF and singular value decomposition (SVD) methods and achieved 4.67% improvement over Netflix's system.

I. Introduction

Recommendation systems predict users' preferences towards items based on user-item interaction by the use of either explicit or implicit information. Explicit information is information given explicitly by the user such as ranking or ratings. Example of implicit information is the user's transaction history, time taken to browse for items or any other type of information where users' feedback is not required. Another type of information that can be used is the content information about the items' or user's profiles. One of the most successful algorithms in recommendation systems is collaborative filtering which has been implemented in services provided by corporations such as Netflix, TiVo and Amazon. The premise of collaborative filtering is that users who agreed in the pass tend to agree in the future.

Many collaborative filtering (CF) techniques have been developed. They can be categorized into three types, content-based, memory-based and model-based methods. Content-based CF methods use content information about items' and users' profiles to find similarities between users or items [1]. In the movie recommendation domain, content information about movies, for example, includes genre, director, or awards. User' profiles could include demographic information such as age, gender, or marital status. Content based collaborative filtering has the advantage of not requiring user's feedback on new items. However, information gathering is usually a difficult task and user' feedback is still required to form their profiles. Moreover, content information differs across domains making content based filtering domain specific. For instance, a content based movie recommendation system might not be easily applied to a book recommendation system.

Memory-based CF uses user-to-user or item-to-item correlations based on users' rating behavior to recommend or predict ratings for users on future items [2]. Correlations can be measured by various distance metrics, such as Pearson correlation coefficient, cosine distance, and Euclidean distance. Memory-based collaborative filtering uses the whole training set each time it computes a prediction, which makes it easy to incorporate new data but suffers slow performance on large data sets. Speedup can be achieved by pre-calculating correlations and other needed information and incrementally updating them. For some applications, however, the size requirement makes the approach infeasible.

Unlike memory-based CF, model-based approach does not use the whole data set to compute a prediction. Instead, it builds a model of the data based on a training set and uses that model to predict future ratings. For example, clustering-based CF method builds a model of the data set as clusters of users, and then uses the ratings of users within the cluster to predict. A very successful model-based method is the Singular Value Decomposition (SVD) [3], which represents the data by a set of vectors, one for each item and user, such that the dot product of the user vector and the movie vector is the best approximation for the training set. Typical the model building process is computationally expensive and memory intensive. After models are constructed, predictions can be done very fast with small memory requirement. Model-based CF methods usually achieve less accurate prediction than memory-based methods on dense data sets where a large fraction of user-item values are available in the training set, but perform better on sparse data sets.

In this paper, we focus on the memory-based CF approach. Existing memory-based CF techniques work well on dense data but poorly on sparse data. To address this weakness, we propose to use z-scores instead of explicit ratings and introduce a mechanism that adaptively combines global statistics with item-based values based on data density level. We present a new adaptive framework that encapsulates various CF algorithms and the relationships among them. An adaptive CF predictor is developed that can self adapt from user-based to item-based to hybrid methods based on the amount of available ratings. Our experimental results show that the new predictor consistently obtained more accurate predictions than existing CF methods, with the most significant improvement on sparse data sets.

We compare the performance of our new method with various CF algorithms and SVD using the Netflix Prize data set. In October 2006, Netflix posted a million-dollar challenge to the public seeking to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. The training set contains 100 million ratings given by 480,189 users to 17,770 movies and the dates of the ratings. The test set contains 2.7 million queries in the form <user, movie, date>. A system that can improve the prediction accuracy of their system, Cinematch, on the test set by 10% wins the million-dollar prize.

In the Netflix challenge, accuracy is measured by the root mean squared error (RMSE)--the square root of the average squared difference between each prediction and the actual rating. Cinematch's RMSE on the test set is 0.9514. When applied to the Netflix data set, our new method performed better than existing CF algorithms and singular value decomposition (SVD) methods. It achieved 4.67% improvement over Cinematch. When its predictions are combined with those of SVD through a simple averaging, the result is 5.6% better than Cinematch.

The paper is organized as follows. In Section II, the basic CF algorithm is introduced and related work is discussed. In Section III, we describe several new techniques to address existing problems of CF algorithms. In Section IV, we present a new adaptive framework representing various CF algorithms and adaptations among them according to data set density conditions. In Section V, we present experimental results showing the improvement of the new techniques and performance on the Netflix Challenge data sets. Finally, In Section VI, we summarize the paper.

II. Related Work

A key element of memory-based CF methods is the similarity measure between users or items. A commonly used similarity measure is the *Pearson correlation coefficient (PCC)* that measures the correlation between two sets of numbers. Given the ratings of one user on a set of items (e.g., movies) and those of another user on a different set of items, the PCC, $p_{u1,u2}$, is calculated as follows:

$$p_{u1,u2}=\frac{1}{N}\sum_{i\in C}\frac{R_{i,u1}-\mu_{u1}}{\sigma_{u1}}\times\frac{R_{i,u2}-\mu_{u2}}{\sigma_{u2}} \quad (1)$$

where $R_{i,u}$ is the rating of user u on item i, μ_u is the average rating of user u, σ_u is the standard deviation of ratings of user u, C is a common set of items between the two users, and $N= |C|$, is the size of C.

Unlike the Euclidean distance, PCC captures the similarity between users based on their correlation or their z-score. The *z-score*, also called normal score or standard score, is derived by subtracting the population mean (μ) from an individual raw score (x) and then dividing the difference by the population standard deviation $(\sigma):z=\frac{x-\mu}{\sigma}$.

Two users can have different averages and variances, but as long as their z-scores are equal, they will be considered identical. PCC also captures dissimilarities between two users. If two users have opposite z-scores, they will have a correlation of −1, which helps to predict ratings of movies that a user will like based on movies that dissimilar users disliked.

The basic user-based collaborative filtering (*uCF*) formula as described in [2] [4] [5] uses PCC to predict the rating of user, u, on item, m, as follows:

$$uCF(u,m)=\mu_u+\frac{\sum_{i\in C}^{n}p_{i,u}\times(R_{i,m}-\mu_i)}{\sum_{i\in C}^{n}|p_{i,u}|} \quad (2)$$

Where the top ***n*** users that have the largest PCC values with user u are used in the prediction.

Other methods, such as the ones in [6] [7], use explicit ratings, not the difference between the explicit ratings and the average, in formula as follows:

$$uCF(u,m)=\frac{\sum_{i\in C}^{n}p_{i,u}\times R_{i,m}}{\sum_{i\in C}^{n}|p_{i,u}|} \quad (3)$$

These methods have the following drawbacks:

1. Failing to consider the variances of the user ratings.
2. Failing to correctly predict the rating of constant raters, i.e., users giving constant rating. In the basic CF formula, the prediction is the user's average plus the average

of deviations of similar users' rating from their averages, which is not likely to be 0.

3. Failing to consider the number of available ratings. A small number of available ratings makes the average and variance calculations unreliable and erroneous. It causes these algorithms to perform poorly on sparse data.
4. For item-based (or movie-based) CF, correlations and predictions are calculated based on the explicit ratings of the users. This is not accurate since users can have very different rating scales.
5. The basic CF prediction formula takes the top ***n*** users even when they have small similarity values, which can lead to inaccurate results.

We have developed various techniques to address these problems, which are presented next.

III. New Techniques to Improve Memory-based CF

In this section, we present several new techniques developed to improve existing memory-based CF methods. Specifically, we incorporate rating variances into the prediction formula, develop a global gravitation mechanism to minimize the effect of sparsity, apply non-linear transformation to reduce the effects of changes by dissimilar users or movies, use the z-scores instead of explicit ratings in calculating prediction and correlations, and integrate clustering methods to reduce the dimensionality of the user-based collaborative filtering.

A. Using rating variances in user-based prediction

As mentioned previously, the basic CF prediction formula in Eq. (2) doesn't consider the variance of user ratings. The consequence is inaccurate prediction. To illustrate the effect of the variances of user ratings, consider the example in Table 1 of two sets of perfectly correlated user ratings.

To predict the rating of User 2 on Movie 1 based on other available ratings, Eq. (2) gives

$$uCF(user_2, movie_1) = 1.5 + \frac{1 \times (1-3)}{1} = 1.5 - 2 = -0.5$$

Not only that the predicted value −0.5 is far away from the real value 1, it was outside the range of feasible values of the rating scale (1 to 5).

To overcome this problem, we modify Eq. (2) by taking variance into account as follows:

$$uCF(u, m) = \mu_u + \left(\frac{\sum_{i \in C}^{n} (p_{u,i} \times z_{i,m})}{\sum_{i \in C}^{n} |p_{u,i}|} \right) \sigma_u \tag{4}$$

$$z_{i,m} = \frac{R_{i,m} - \mu_i}{\sigma_i} \tag{5}$$

First, *n* most similar scores are normalized to get their z-scores. Then, the weighted average of the z-scores based on the PCC is multiplied by the standard deviation of the user's rating to form the predicted displacement from the average rating of the user. In this way, the displacement is transformed from the z-score scale to the raw rating scale of the user.

When the formula in Eq. (4) is applied to the example in Table 1, the predicted rating of User 2 on Movie 1 based on other available ratings is 1, which matches the true value.

Another benefit of Eq. (4) is that ratings of constant raters are predicted correctly. Since the standard deviation of constant rater is 0, the second term of Eq. (4) is always 0 for a constant rater, meaning that the predicted rating for a constant rater is always the average rating of the user, independent of the ratings of other users.

A special case of the *uCF* formula is when $p_{i,j} = 1$ for all *i* and *j*, which we call the z-score average predictor (*zAvg*):

$$zAvg(m)=\mu_u+\left(\frac{\sum_{i=0}^{N} z_{i,m}}{\mathrm{N}}\right)\sigma_u \tag{6}$$

B. A global gravitation mechanism for sparse data

The basic CF method performs poorly on sparse data. For example, a user who has rated only one movie as 5 doesn’t necessarily have an average of 5 and a standard deviation of 0. Techniques have been developed to overcome this problem. One idea is that when the amount of available ratings is low, adjust the user’s average rating and standard deviation toward the global values, e.g., using the weighted average between the calculated average and the global average, where the weight of the calculated average is the number of available ratings and the weight of the global average is some constant [8].

In our study, we derive empirical relationship between the number of available ratings and the weights in the weighted some formula. From the Netflix training data set, 2000 movies were extracted each with at least 10,000 available ratings. We compute the overall average. Then we randomly select from 1 to 1000 sample ratings and calculate the difference between the sample average and the overall average. The results are shown in Fig. 1. The relationship between the error and the number of available ratings is a power function of the sample size (***N***), which is consistent with the Large Number Theorem.

Based on the empirical result, we design an adaptive weighting scheme called *global gravitation* to adjust the average rating of a movie or a user between the value calculated from its available ratings and the global average calculated from all related ratings. The global gravitation mechanism is described in Procedure 1.

Global gravitation first calculates the error range based on the number of samples and the relationship derived from Fig. 1. Then, it returns the value from the range [*Value±error*] that is closest to the global average. The temperature parameter, *T*, controls the weight of the global gravitation.

Global gravitation will set all users’ averages and standard deviations to constants (the global values) on sparse data sets, making the z-score average predictor in Eq. (6) equivalent to the explicit rating predictor in Eq. (7). The proof is shown as follows:

$$Avg(m)=\frac{\sum_{i=0}^{N} R_{i,m}}{N} \tag{7}$$

Let GA = global average

Let GSD = global standard deviation

Set μ_u = GA for all users

Set σ_u = GSD for all users

$$\begin{aligned}
zAvg(m)&=GA+\left(\frac{1}{N}\sum_{i=0}^{N}\left(\frac{R_{i,m}-GA}{GSD}\right)\right)GSD\\
&=GA+\frac{1}{N}\frac{1}{GSD}\left(\sum_{i=0}^{N}R_{i,m}-\sum_{i=0}^{N}GA\right)\times GSD\\
&=GA+\frac{\sum_{i=0}^{N}R_{i,m}}{N}-\frac{\sum_{i=0}^{N}GA}{N}\\
&=GA+\frac{\sum_{i=0}^{N}R_{i,m}}{N}-GA\\
&=\frac{\sum_{i=0}^{N}R_{i,m}}{N}\\
&=Avg(m)
\end{aligned}$$

C. Non-linear transformation to handle low similarity

A problem of the basic CF formula is that it uses the top ***n*** similar users' ratings, even when all of them have very low similarities, i.e., correlation coefficients close to 0. It has been observed that low similarities produce inaccurate prediction, which may be less accurate than the user average.

To handle the low similarity case, we first modify Eq. (4) by multiplying each term of the weighted average by a sigmoid function of the absolute value of the PCC:

$$uCF(u,m)=\mu_u+\left(\frac{\sum_{i\in C}^{n}(p_{u,i}\times z_{i,m})S_{u,i}}{\sum_{i\in C}^{n}|p_{u,i}|}\right)\sigma_u \tag{8}$$

$$S_{u,i}=\frac{1}{1+e^{-25(|p_{u,i}|-0.1)}} \tag{9}$$

The sigmoid function acts as a soft threshold, limiting the effects of low-similar ratings. For the case that the most similar ratings all have small values, i.e., the *p*'s are small, the *S*'s will be small, making the second term in Eq. (8) small and the predicted rating approaches the user average rating. The parameters in the formula were derived empirically.

The global gravitation can be applied to PCC, making it go to zero as the number of available ratings becomes low, which makes the weighted average go to the user average as the data becomes sparser.

We have observed that the relationship between the accuracy of the prediction and the value of the PCC is not linear. For example, a prediction based on a PCC of 0.5 usually has an error less than half of that based on a PCC of 0.25. We tried various nonlinear transformation functions, including polynomial, sigmoid, and Gaussian functions, on the PCCs and found that the square function works well. Thus, the final version of our user-based CF formula is as follows:

$$uCF(u,m)=\mu_u+\left(\frac{\sum_{i\in C}^{n}(P_{u,i}\times z_{i,m})S_{u,i}}{\sum_{i\in C}^{n}|P_{u,i}|}\right)\sigma_u \tag{10}$$

$$P_{u,i}=\frac{p_{u,i}}{|p_{u,i}|}p_{u,i}^{2} \tag{11}$$

D. Combining movie-based and user-based prediction

The formula of the basic movie-based CF (*mCF*) is similar to the user-based CF in Eq. (4):

$$mCF(u,m)=\mu_m+\left(\frac{\sum_{i\in C}^{n}\left(p_{m,i}\left(\frac{R_{u,i}-\mu_i}{\sigma_i}\right)\right)S_{m,i}}{\sum_{i\in C}^{n}|p_{m,i}|}\right)\sigma_m \tag{12}$$

When comparing two users, each set of ratings is subject to only one user's evaluation. On the other hand, the movie-to-movie correlation is a comparison between two sets of ratings that have been rated by different users, each commonly having his/her own interpretation of the rating scale. It is necessary to convert the ratings of different users into comparable scales in order to make more meaningful comparison between the ratings of two movies.

In our improved method, before calculating the similarity between two movies, we first transform their ratings into z-scores. Fig. 2 and Fig. 3 show the significant difference between the correlation coefficient distributions using the raw ratings and the z-scores. Using the Netflix data set, Fig. 2 shows the distribution of the movie-to-movie PCCs based on raw ratings. The majority values are positive. On the same data set, Fig. 3 shows the distribution of the movie-to-movie PCCs based on corresponding z-scores (called **zPearson**). Now, the zPearson value distribution is closer to a normal distribution with balanced negative and positive correlations. The two peaks at 0 and 1 are due to the fact that many movie pairs in the data set have a 0 or a very small common set of users.

In this method, we combine movie-based with user-based prediction and use z-scores instead of the raw ratings. The z-score movie-based-CF predictor (called *zCF*) is as follows:

$$zCF(u,m)=\mu_u+\mathbb{Z}_{u,m}\times\sigma_u \tag{13}$$

$$\mathbb{Z}_{u,m}={}_z\mu_m+\left(\frac{\sum_{i\in C}^{n}{}_zp_{m,i}\left(\frac{z_{i,m}-{}_z\mu_i}{{}_z\sigma_i}\right)S_{m,i}}{\sum_{i\in C}^{n}|{}_zp_{m,i}|}\right){}_z\sigma_m \tag{14}$$

Where ${}_z\mu_m$ is the z-score average of movie *m*, ${}_z\sigma_m$ is the standard deviation of the z-scores of movie *m*, and ${}_zp_{m,i}$ is the z-score based PCC between movie *m* and movie *i*.

Other techniques have been applied to identify users who rate consistently lower or higher than others by the use of the deviation from the average, such as the removal of global effect technique in [8] and the adjusted cosine similarity in [7]. The z-score is more informative and effective than the others since it considers the variance.

E. Cluster-based CF

Clustering has been used in recommendation systems mostly for finding similar users [4] [5]. In our method, we use clustering for a different purpose --- to decompose the global movie-average vector (given by all users) into ***k*** local movie-average vectors (given by similar users). One problem is each local vector may be much sparser than the global average vector, which makes predictions based on clusters with small number of users unreliable and inaccurate. To overcome this problem, we apply the global gravitation mechanism to each local movie-average vector (cluster) such that the global movie-average vector will carry more weight when a cluster is sparse. The global gravitation mechanism also alleviates missing data problem since not all clusters have seen all movies. Fig. 4 and 5 illustrate the concept.

In Fig. 4 and Fig. 5, each rectangle represents a movie average vector (cluster centriod). Each column, labeled from m_1 to m_n, is a different movie. The first and the second rows are the average of the movie and the number of users who have seen it, respectively. The shaded areas in Fig. 4 denote missing ratings due to the fact that no user in that cluster has seen the movie. In some clusters, there are some data missing and some movie averages are based on just one rating. By applying the global gravitation technique, we ensure that predictions based on clustering will approach the global movie average predictor if the clusters are too sparse. In Fig. 4 and Fig. 5, raw ratings were used to represent the centroids, whereas in our implementation we use z-scores.

We have implemented two clustering-based CF methods, one based on raw rating as in Eq. (15) and the other based on z-scores as in Eq. (16). The k-means clustering algorithm is used with preset k values, e.g., k=100, and the Euclidean distance as the distance measure. The prediction is calculated as a weighted average of the top ***K*** clusters, where the weights are the inverse Euclidean distance between the user and the centroids.

$$Cluster(u,m)=\frac{\sum_{c=1}^{K}(\alpha_{u,c}\times\mu_{c,m})}{\sum_{c=1}^{K}\alpha_{u,c}} \tag{15}$$

$$zCluster(u,m)=\mu_u+\left(\frac{\sum_{c=1}^{K}(\alpha_{u,c}\times{}_z\mu_{c,m})}{\sum_{c=1}^{K}\alpha_{u,c}}\right)\sigma_u \tag{16}$$

$$\alpha_{u,c}=\frac{1}{Euclidean_{u,c}} \tag{17}$$

Where $\mu_{c,m}$ is the average of movie *m* in cluster *c* and ${}_z\mu_{c,m}$ is the z-score average of movie *m* in cluster c. The global gravitation technique is also applied to adjust the Euclidean distance in case the common set between the cluster and the user is small.

IV. A New Adaptive CF Framework

Putting the new techniques together, we develop a new CF method called z-score-based clustered collaborative filtering (*zCCF*). It is based on z-scores and clustering, integrates complementary user-based and movie-based methods, and adapts to the density of data sets.

$$zCCF(u,m)=\mu_u+\left(\frac{\sum_{c=1}^{K}(\alpha_{u,c}\times\mathbb{Z}_{u,m,c})}{\sum\alpha_{u,c}}\right)\sigma_u \tag{18}$$

$$\mathbb{Z}_{u,m,c} = {}_z\mu_{c,m} + \left(\frac{\sum_{i \in C}^{n} {}_z p_{i,m} \left(\frac{z_{u,i} - {}_z\mu_{c,i}}{{}_z\sigma_{c,i}} \right) S_{m,i}}{\sum_{i \in C}^{n} |{}_z p_{i,m}|} \right) {}_z\sigma_{c,m} \tag{19}$$

All symbols have been defined in previous equations or can be similarly defined.

zCCF is an adaptive CF framework encapsulating the various CF methods discussed in the previous section, including *Avg* as in Eq. (7), *zAvg* in Eq. (6), *uCF* in Eq. (4), *mCF* in Eq. (12), *zCF* in Eq. (13), *Cluster* in Eq. (15), and *zCluster* in Eq. (16). Fig. 6 shows these methods and their relationships.

In Fig. 6, GA is the global average of all ratings in a data set, which equals to 3.45 for the Netflix data set. Through the global gravitation mechanism and non-linear transformations, we can adjust the method to adapt to the amount of data so that it becomes a more model-based algorithm as the amount of information reduces and a more item-based one when the amount of data increases. Moreover, any z-score based predictor can be converted to an explicit rating predictor by simply setting all users' averages and standard deviation to constants.

The relationships between various algorithms are represented as labeled arrows in Fig. 6. Their means are as follows:

1. *Avg* becomes *GA* for sparse data due to the global gravitation mechanism.
2. *Cluster* becomes *Avg* for sparse data as the global gravitation mechanism pulls the clusters' centroids towards the global average vector or when the number of clusters (*k*) is set to 1.
3. *mCF* becomes *Avg* as the nonlinear transformation removes the effect of low-similar movies from *mCF*, i.e., when setting $S_{i,j}$ to 0 in Eq. (12).
4. *zAvg* becomes *Avg* when the global gravitation mechanism sets all users' averages and standard deviation to constants for sparse data.
5. *zCF* becomes z*Avg* when the nonlinear transformation removes the effect of low-similar movies from *zCF*, i.e., by setting $S_{i,j}$ to 0 in Eq. (13).
6. *uCF* becomes *zAvg* by ignoring similarities and setting $p_{i,j}$ = 1 in Eq. (4).
7. *zCluster* becomes *zAvg* when the global gravitation mechanism pulls the clusters' centroids towards the global z-score average vector for sparse data or if the number of clusters (*k*) is set to 1.
8. *zCluster* becomes *uCF* by setting the number of clusters (*k*) to be equal to the number of users.
9. *zCCF* becomes *zCluster* as the nonlinear transformation removes the effect of low-similar movies from *zCCF* by setting $S_{i,j}$ to 0 in Eq. (19).

V. Experimental results

In the experiments, we compare the prediction accuracies of the proposed techniques on data sets with different densities. The data sets were extracted from the Netflix data set. First, we selected 10,000 users with the most number of ratings, and then the 2,000 most popular movies for these 10,000 users were selected. This set contains 9,866,056 ratings and is 49.33% full.

One million ratings were randomly removed from the set to be used as the test set. To generate sparser data sets, we randomly remove more ratings.

A. Global gravitation

Fig. 7 compares the results of *Avg*, z*Avg*, and z*Avg*+*GG* (z*Avg* with the global gravitation mechanism) predictors with data sets of different density. z*Avg* is better than *Avg* using dense data sets, but worse on sparse data sets. For example, z*Avg* is 10.2% better than *Avg* using a 45% dense set. However, z*Avg* performs much worse when density is below 10%. By using global gravitation, z*Avg*+*GG* achieves good results on both sparse and dense sets, consistently better than *Avg* and z*Avg*. Its accuracy is similar to z*Avg* on dense sets and approaches *Avg* on very sparse data sets.

B. Non-linear transformation

Fig. 8 compares the results of *Avg*, *zCF* without transformation, *zCF* with a transformation function of |zPearson| (*zCF* × |zPearson|), and *zCF* with a transformation function of *Sigmoid(zPearson)* (*zCF* ×Sigmoid) on data sets of different density. *zCF* fails on sparse data because it allows ratings for movies with low similarities to contribute to the prediction. Bellow the 10% density level, *zCF* performs worse than *Avg* by up to 16%. On the other hand, *zCF* is much better than *Avg* on dense data, with 21.5% improvement at 45% density level. *zCF* with transformation functions |zPearson| and *Sigmoid(zPearson)* consistently outperforms *Avg*. *zCF* with *Sigmoid(zPearson)* consistently outperforms *zCF* with |zPearson| as well.

C. Z-scores vs. raw ratings

Fig. 9, Fig. 10 and Fig. 11 compares the results using z-sores and raw ratings. Z-score based algorithms consistently outperform their counterparts based on raw ratings (after applying global gravitation), especially on dense data. At 45% density level, the z-score based z*Avg* is 10.25% better than *Avg*, *zCluster* is 4.2% better than *Cluster*, and *zCF* is 3.4% better than *mCF*.

D. Methods hierarchy test

Fig. 12 compares the performances of six methods in the adaptive framework, clearly showing their performance relationship across data sets of different density. The incremental and consistent improvement of a generalized method over a specialized method is a result of the global gravitation mechanism and the non-linear transformations of low-similarity. *zCCF* is the best overall, achieving excellent results on dense data sets and good results on sparse data sets as well.

E. zCCF vs. SVD

We have also compared *zCCF* with the improved regularized SVD in [3], which is significantly better than the basic implementation of SVD. As shown in Fig. 13, *zCCF* is slightly better than SVD and achieves 2% improvement on dense data.

F. Results on Netflix Challenge test set

Netflix uses a test set unknown to the public to evaluate the accuracy of submitted entries. Participants can submit their predictions to the Netflix challenge website and receive the RMSE of their predictions. Fig. 14 shows the RMSEs of predictions by the methods studied in this paper, together with Netflix's Cinematch result and the grand prize target. The results show that z-score based methods are more accurate than their raw rating based counterparts. Overall *zCCF* is the best among individual method and is 4.67% better than Cinematch. We also generated a submission by averaging *zCCF*'s predictions and *SVD*'s predictions and achieved 5.6% improvement over Cinematch.

VI. Summary

In this paper, we have presented a number of new techniques developed to improve existing CF methods and an adaptive framework for CF that captures a range of CF algorithms, from the simple global constant average to the more complicated z-score based clustered CF. Our method can adapt to the amount of information available using relationships extracted from the training data. The method is competitive with SVD, the state of the art model-based approach, on sparse data sets and outperforms it on dense data sets. Promising results on the Netflix Challenge data set were obtained. Additional improvement was achieved by combining the model-based approach and the memory-based approach. In future work, we plan to incorporate the information extracted from SVD in more systematic and effective ways and incorporated it into our adaptive framework.

References

1. Mooney, RJ.; Roy, L. Content-based book recommending using learning for text categorization. Proc the Fifth ACM Conference on Digital Libraries; 2000. p. 195-204.
2. Breese, JS.; Heckerman, D.; Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proc. the Fourteenth Conference on Uncertainty in Artificial Intelligence; 1998. p. 43-52.
3. Paterek A. Improving regularized singular value decomposition for collaborative filtering. KDD-Cup and Workshop. 2007
4. Chen, G.; Wang, F.; Zhang, C. Collaborative Filtering Using Orthogonal Nonnegative Matrix Tri-factorization. Workshop on High Performance Data Mining, in conjunction with the 7th IEEE International Conference on Data Mining (ICDM); 2007.
5. Rashid A, Lam SK, Karypis G, Riedl J. ClustKNN: A Highly Scalable Hybrid Model- & Memory-Based CF Algorithm. Proc of WebKDD 2006. 2006
6. Ali, K.; van Stam, W. TiVo: Making Show Recommendations Using a Distributed Collaborative Filtering Architecture. Proc. the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2004. p. 394-401.
7. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based Collaborative Filtering Recommendation Algorithms. Proc. 10th International Conference on the World Wide Web; 2001. p. 285-295.
8. Bell, R.; Koren, Y.; Volinsky, C. Modeling Relationships at Multiple Scales to Improve Accuracy of Large Recommender Systems. Proc. the 13th ACM Int. Conference on Knowledge Discovery and Data Mining (KDD'07); 2007.

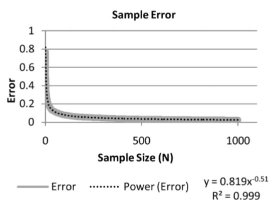

Fig. 1.
The errors between the subset sample average and the overall average for different number of samples, ***N***. It is a power function of ***N***.

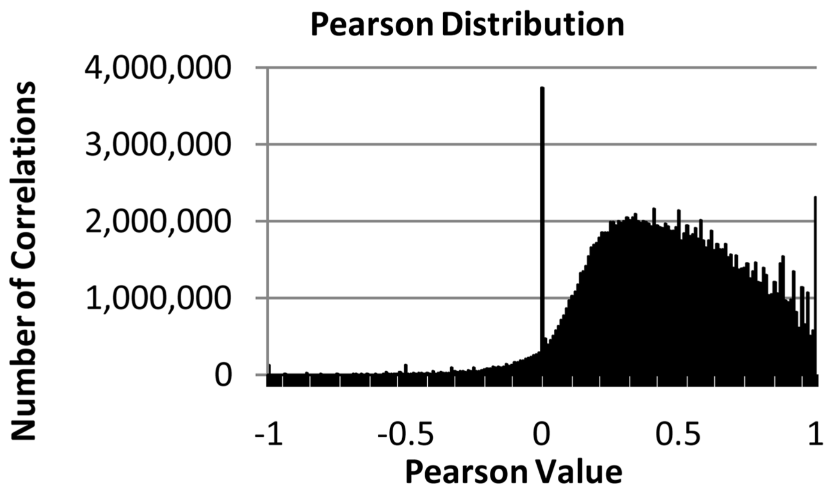

Fig 2.
The distribution of the movie-to-movie PCC based on the raw ratings of the Netflix data set.

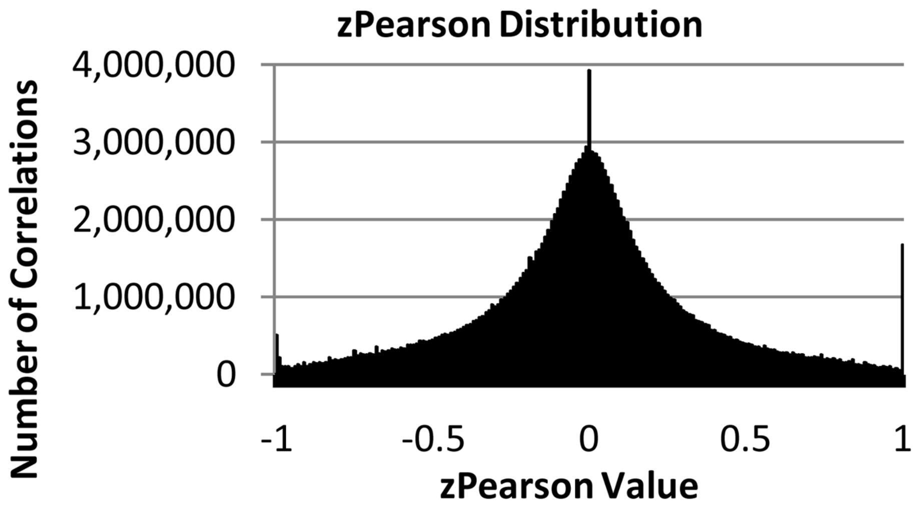

Fig 3.
The distribution of the movie-to-movie PCC based on the z-scores of the Netflix data set.

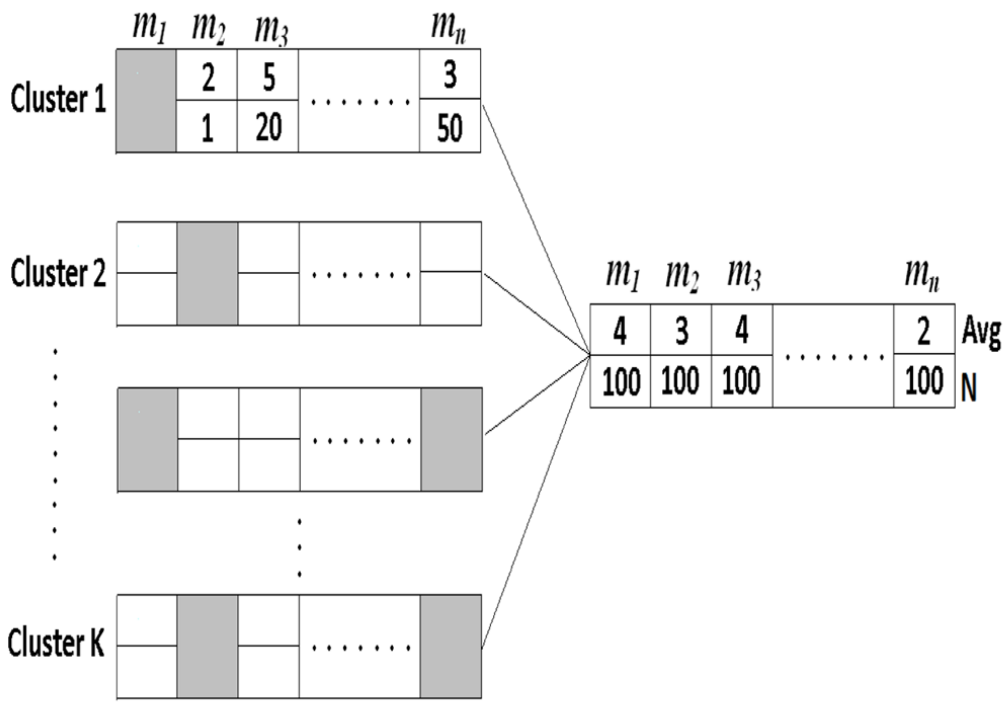

Fig. 4.
Decomposing the global average vector into *k* sub-vectors

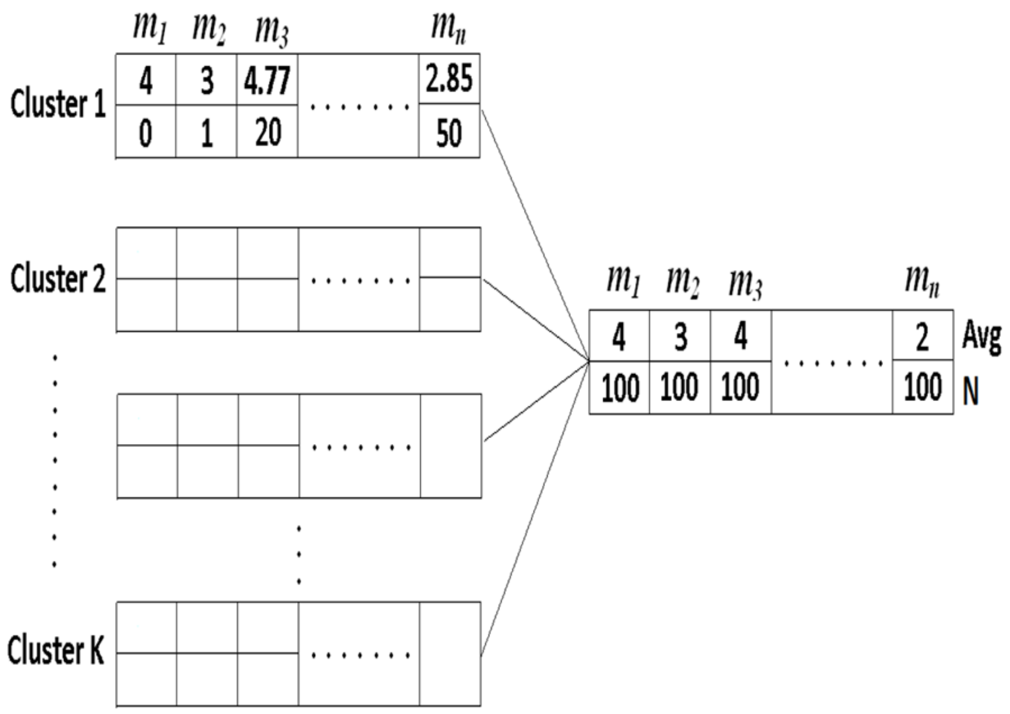

Fig. 5.

Applying the global gravitation to the *k* sub-vectors with *T* = 1

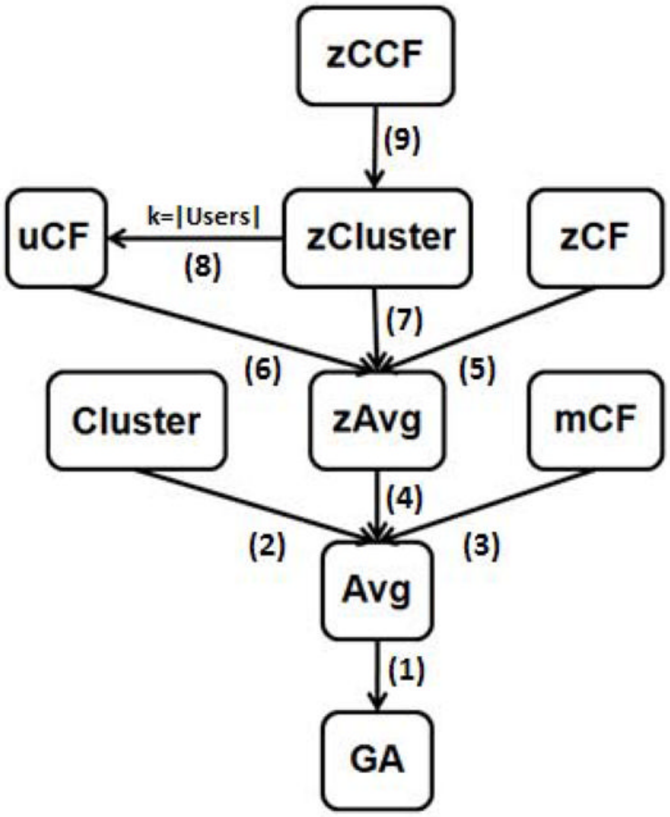

Fig. 6.
Methods hierarchy of the *zCCF* framework: The arrows can be read as “is a general form of”. |Users| is the number of users and k is the number of clusters.

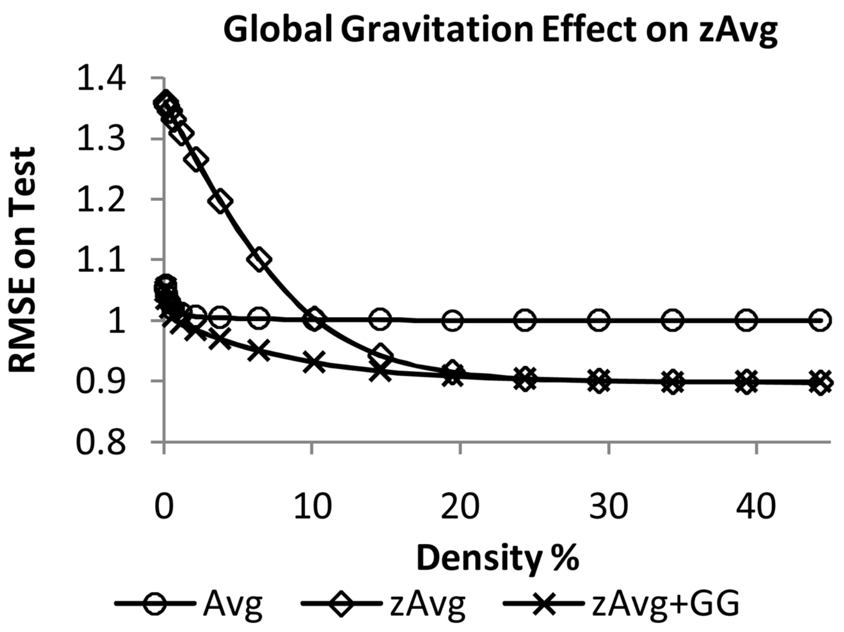

Fig. 7.
Comparison between *Avg*, *zAvg*, and *zAvg+GG* (*zAvg* with the global gravitation mechanism) predictors on data sets of different density.

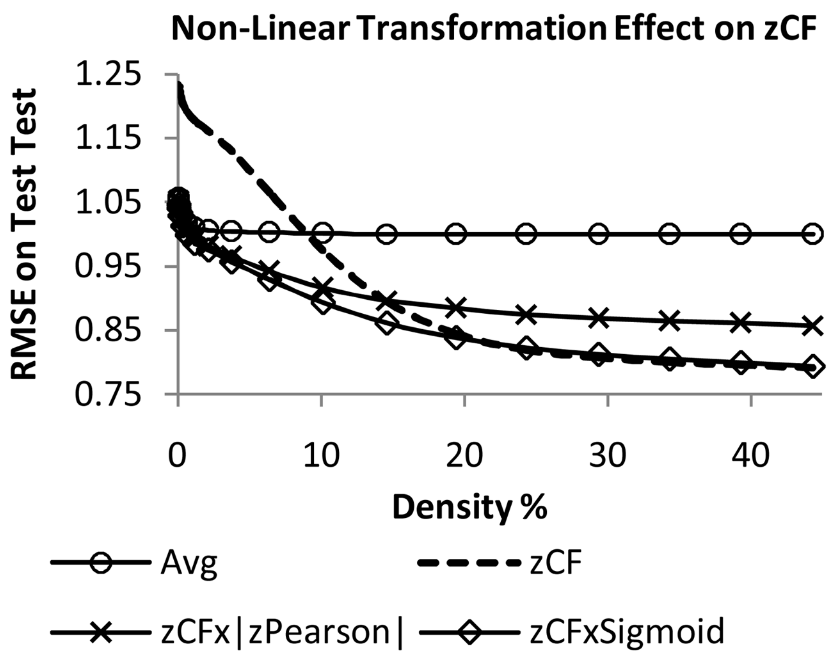

Fig. 8.
Comparison between *Avg*, *zCF* without transformation, *zCF* with a transformation function of |zPearson| (*zCF* × |zPearson|), and *zCF* with a transformation function of *Sigmoid(zPearson)* (*zCF* × Sigmoid) on data sets of different density.

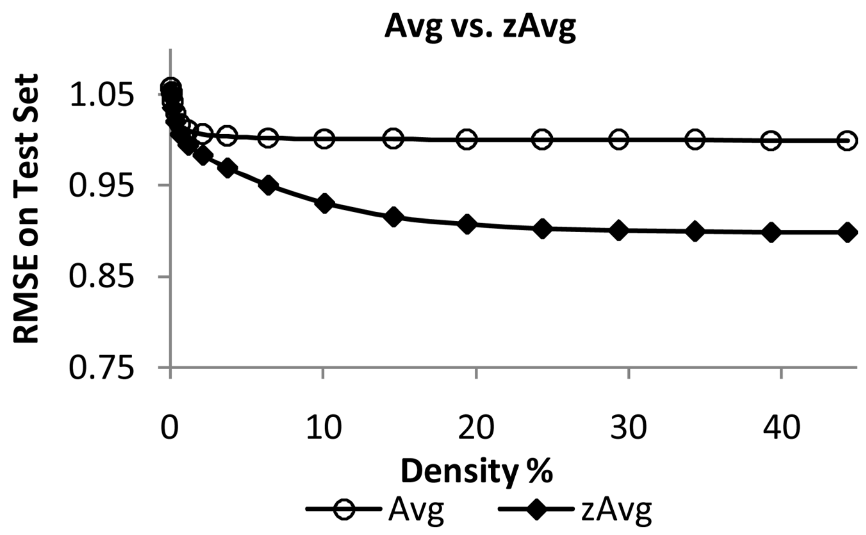

Fig. 9.
Comparison between *Avg* (average of raw ratings) and *zAvg* (average of z-scores) on data sets of different density.

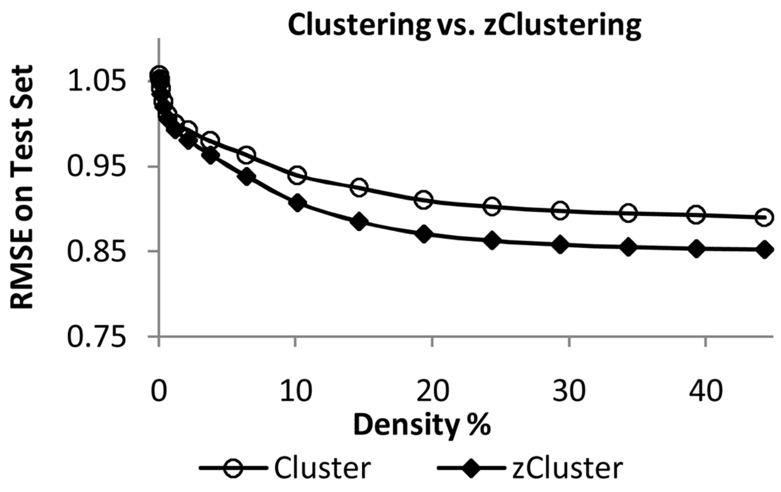

Fig. 10.
Comparison between clustering based on the z-scores and raw ratings on data sets of different density.

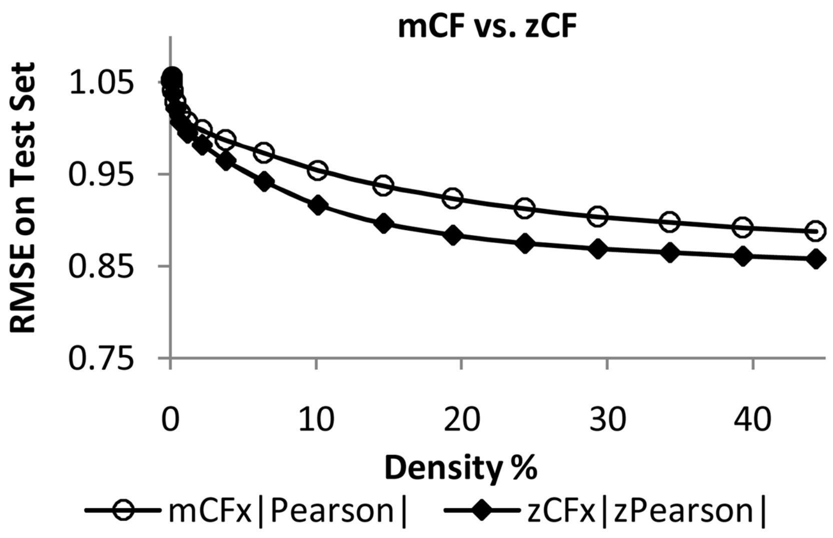

Fig. 11.
Comparison between *zCF* and *mCF* on data sets of different density.

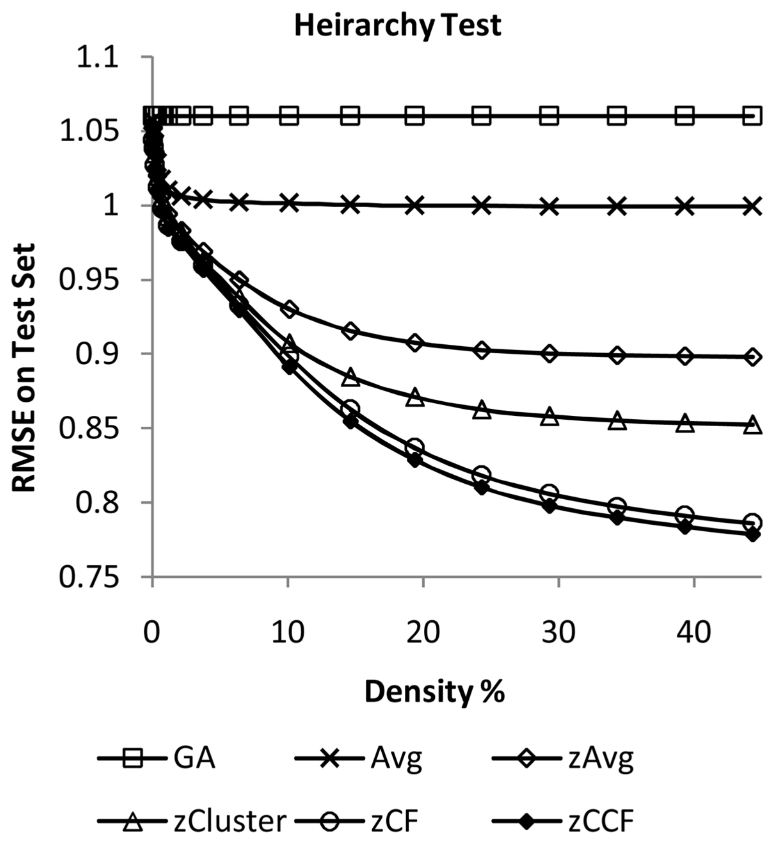

Fig. 12.
Performance comparison of various methods represented in the adaptive framework of *zCCF* hierarchy on data sets of different density.

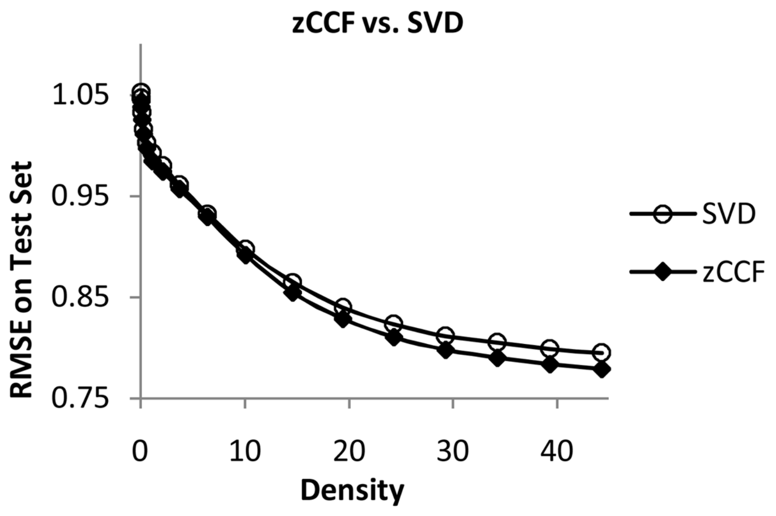

Fig. 13.
Comparison between the improved regularized SVD in [3] and zCCF on data sets of different density.

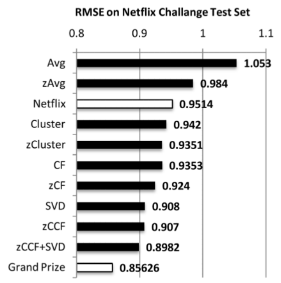

Fig. 14.
Results on the Netflix challenge test data set.

TABLE 1

An example of two sets of perfectly correlated user ratings

	movie$_1$	**movie$_2$**	**movie$_3$**	**movie$_4$**	**μ**	**σ**
user$_1$	1	5	1	5	3	2
user$_2$	1	2	1	2	1.5	0.5

Procedure 1

The Global Gravitation Mechanism

```
Input:
    Value    //Parameter value
    GA       //The global average
    N        //The number of samples
    T        //Desired temperature value
Output:
    New_ Value
Global Gravitation Mechanism:
    $error = \frac{T}{\sqrt{N}}$
    IF (Value >GA)
        New_ Value = max (GA, Value − error)
    ELSE
        New_ Value = min (GA, Value + error)
    END IF
    RETURN New_ Value
```