A NEW ADAPTIVE FRAMEWORK FOR COLLABORATIVE FILTERING PREDICTION

A Thesis
presented to
the Faculty of the Graduate School
University of Missouri-Columbia

In Partial Fulfillment
of the Requirements for the Degree
of Master of Science

---

by

Ibrahim Almosallam

Dr. Yi Shang, Thesis Supervisor

May 2008

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

A NEW ADAPTIVE FRAMEWORK
FOR COLLABORATIVE FILTERING PREDICTION

presented by Ibrahim Almosallam
a candidate for the degree of Master of Science
and hereby certify that in their opinion it is worthy of acceptance.

Dr. Yi Shang

Dr. James Keller

Dr. Kannappan Palaniappan

ACKNOWLEDGMENTS

First I would like to thank my adviser Dr. Yi Shang for his continuous support in this research. He has been very generous in sharing his impressive knowledge in computer science, academic skills and his experience in life in general.

I also appreciate the help given by the University of Missouri's community from students and faculty during my graduate studies.

I would also like to thank the Ministry of Higher Education in Saudi Arabia for supporting me and my family during our studies in the United States.

Thanks also to Dr. James Keller and Dr. Kannappan Palaniappan for their help and support during the courses they taught me and for agreeing to review my master thesis.

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 project 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. A new adaptive framework that encapsulates various CF algorithms and the relationships among them is presented. 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. The 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, The method performed better than existing CF and singular value decomposition (SVD) methods and achieved 4.67% improvement over Netflix's system.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Recommendation systems predict users' preferences towards items based on user-item interaction by the use of either explicit or implicit information [8]. 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 users' 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 [9]. 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 [10] [2] [3]. 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 users' feedback on new items. However, information gathering is usually a difficult task and users' 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 [5] [7]. 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, the 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) [11], 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. 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 project, 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. A new adaptive framework that encapsulates various CF algorithms and the relationships among them is presented. 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. The 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. The performance of the new method is compared with various CF algorithms and SVD using the Netflix Prize data set. When applied to the Netflix data set, the 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.

1.2 An Overview of the Netflix Challenge 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 rate a movie based on their movie preferences. The training set contains 100,480,507 ratings , 1 to 5, 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. The data set is very sparse, only 1.18% of the 17,770 by 480,189 matrix is given. The following figures shows some statistics about the data set for the reader to have an overview of the data.

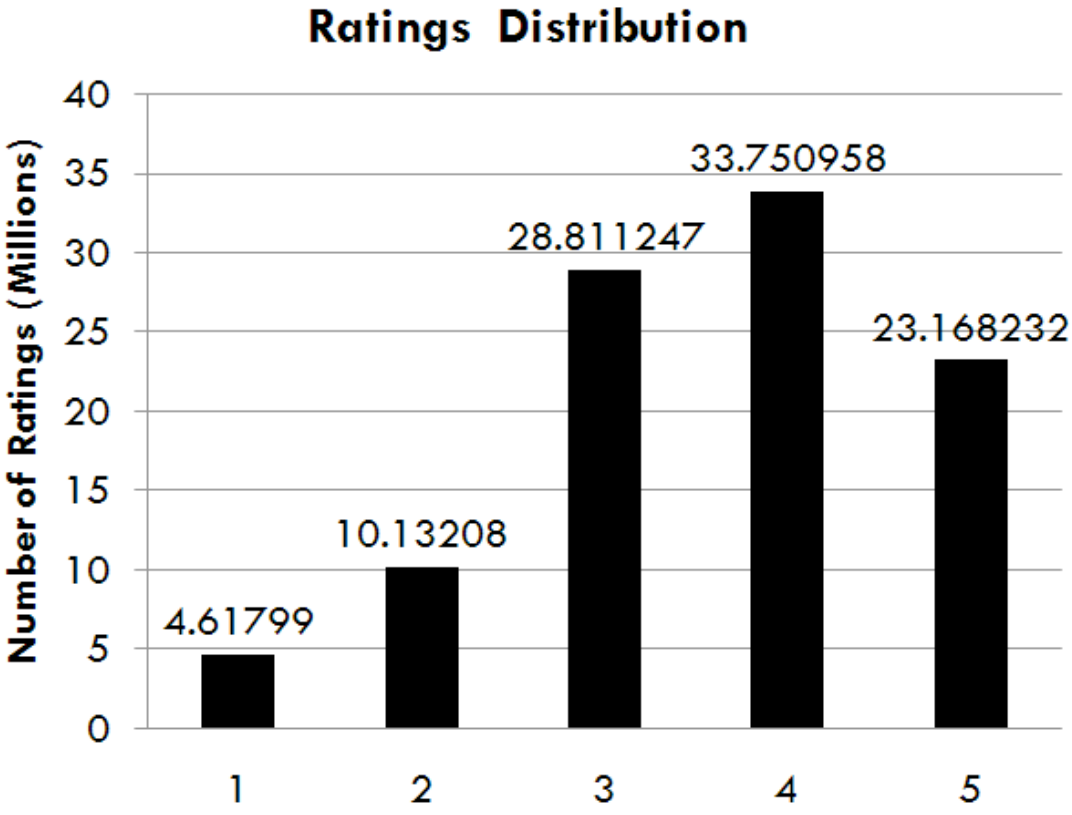

Figure 1.1: The overall distribution of the ratings in the data set.

Fig. 1.1 shows the distribution of the ratings (1 to 5) in the data set. It is important to emphasis to the reader that Netflix's users don't have to rent a movie in order to rate it. They can simply choose the movies they have seen before and rate them. Therefore, it can be concluded from Fig. 1.1 that most users choose to rate the movies they liked thinking that it will help the system

to recommend better movies for them. However, it makes it more difficult to predict movies that they will dislike.

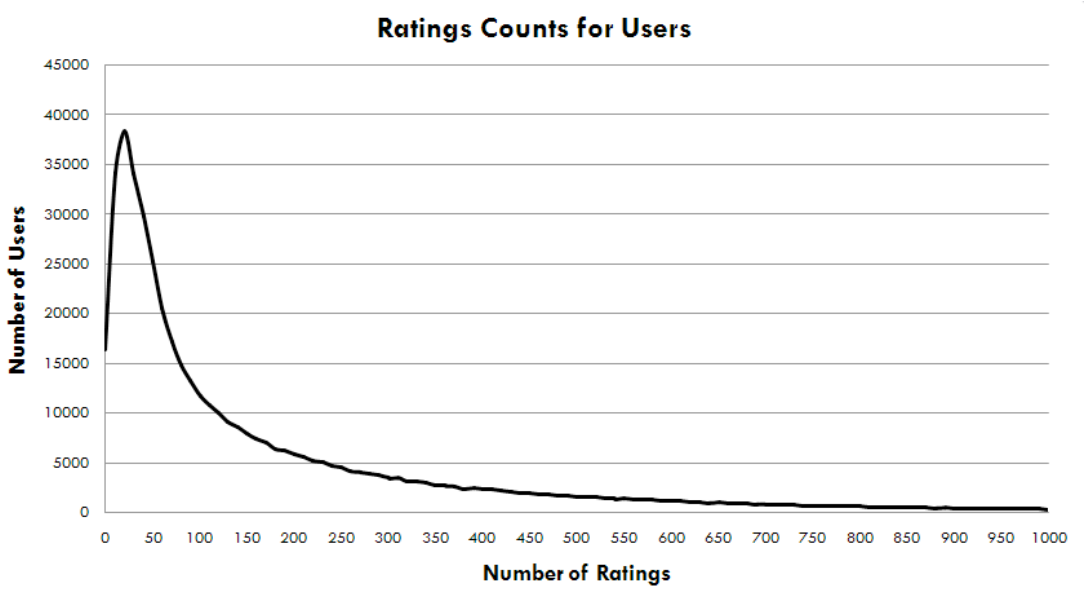

Figure 1.2: The distribution of the number of ratings given by users. The 25th, 50th and 75th percentiles occurs at 38, 95 and 258 respectively.

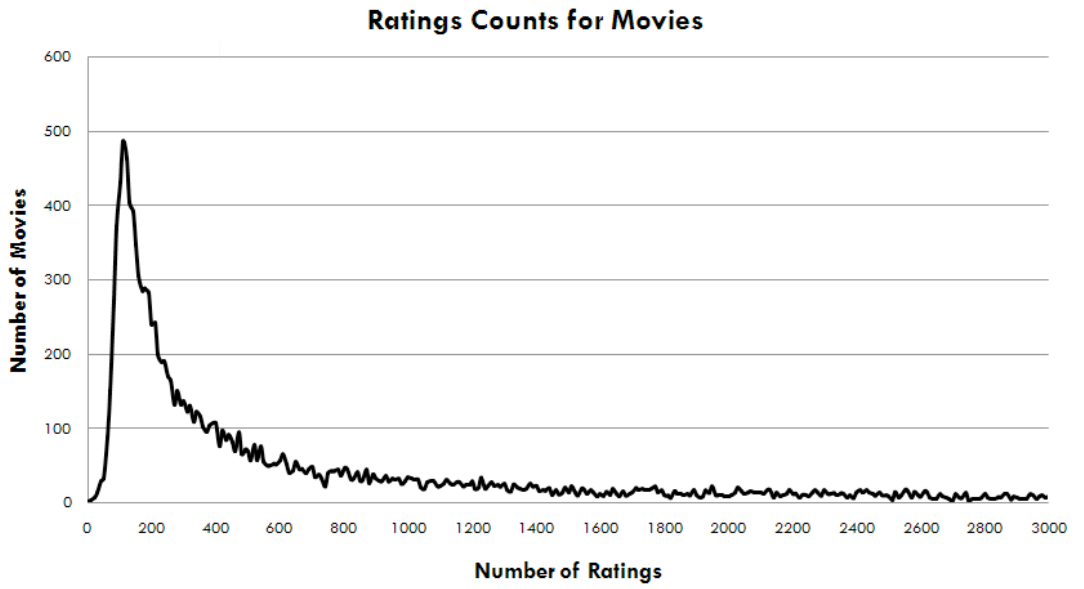

Figure 1.3: The distribution of the number of ratings given to movies. The 25th, 50th and 75th percentiles occurs at 191, 560 and 2,667 respectively.

The above figures, Fig. 1.2 and Fig. 1.3, clearly show the sparsity of the data, especially on the user side. 25% of the users have seen less than 38 movies and 50% have seen less than 95 movies. This will cause a problem latter when calculating parameters such as their average or standard deviation. It is more dense on the movie side where 25% of the movies have been rated 191 times or less.

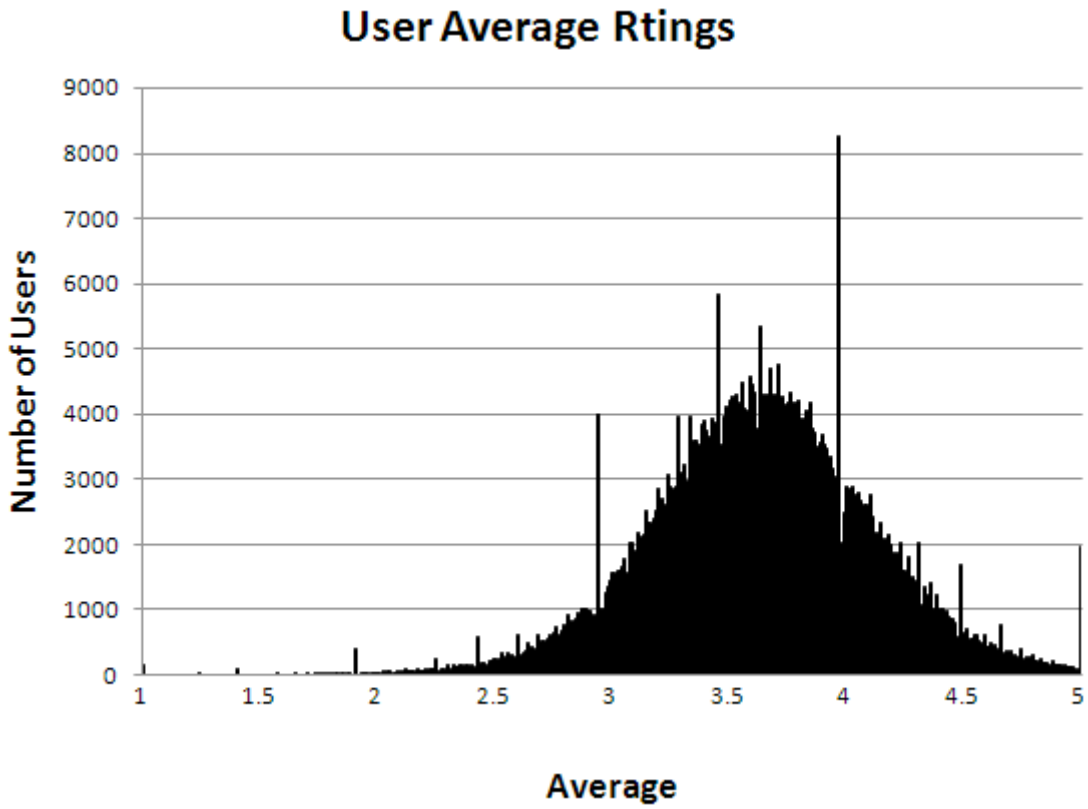

Figure 1.4: The distribution of users' average ratings.

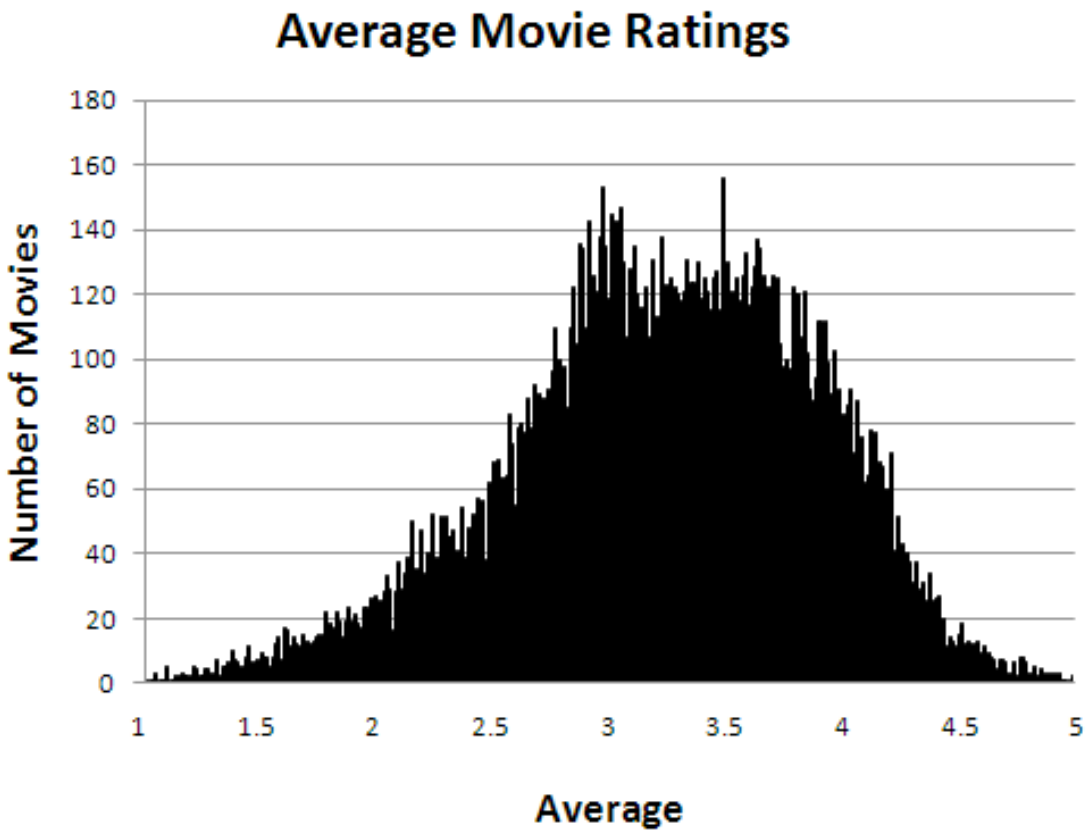

Figure 1.5: The distribution of movies' average ratings.

The main points to be observed from Fig. 1.4 is that there exist a great deal of users with integer averages, which is due to the fact that those users have seen very few movies which caused the distribution to look noisy and not normal as one would expect. It is less of a problem in the movies' averages, Fig. 1.5, where the distribution is smother and there is not a lot of movies with integer averages.

1.3 Related Work

The basic memory based collaborative filtering method is to compute the similarity between the user in question and all the users who have rated the movie in question. The similarities are then used as weights to the users' ratings and

the weighted average is returned as the prediction. There are different similarity measures such as the Euclidean distance, cosine, slope one and Pearson correlation coefficient. The most commonly used similarity measure in the literature 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 between user a and user b, $p_{a,b}$, is calculated as follows:

$$p_{a,b} = \frac{1}{N} \sum_{i \epsilon C} \frac{R_{a,i} - \mu_a}{\sigma_a} \times \frac{R_{b,i} - \mu_b}{\sigma_b} \tag{1.1}$$

where $R_{u,i}$ 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 the 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 (σ): $z = \frac{x-\mu}{\sigma}$. 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 [5] [6] [12] uses PCC to predict the rating of user u on item m is shown in Eq. (1.2):

$$uCF(u, m) = \mu_u + \frac{\sum_{i=1}^{n} p_{u,i} \times (R_{i,m} - \mu_i)}{\sum_{i=1}^{n} |p_{u,i}|} \tag{1.2}$$

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 [1] [13], use explicit ratings, not the

difference between the explicit ratings and the average, in formula as follows:

$$uCF(u,m) = \mu_u + \frac{\sum_{i=1}^{n} p_{u,i} \times R_{i,m}}{\sum_{i=1}^{n} |p_{u,i}|} \tag{1.3}$$

Existing CF 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

In this project, we have developed several techniques to address these problems, which are presented in the remaining chapters.

Chapter 2

New Techniques to Improve Memory-Based CF

In this chapter, several new techniques developed to improve existing memory-based CF methods are presented. 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.

2.1 Using Rating Variances in User-Based Prediction

As mentioned previously, the basic CF prediction formula in Eq. (1.2) doesnt consider the variance of user ratings. The consequence is inaccurate predictions. To illustrate the effect of the variances of user ratings, consider the example in Table 1 of two sets of perfectly correlated user ratings.

Table 2.1: 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

To predict the rating of $user_2$ on $movie_1$ based on other available ratings, Eq. (1.2) gives:

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

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, Eq. (1.2) is modified by taking variance into account as follows:

$$uCF(u,m) = \mu_u + \left(\frac{\sum_{i=1}^{n} (p_{u,i} \times z_{i,m})}{\sum_{i=1}^{n} |p_{u,i}|} \right) \sigma_u \tag{2.1}$$

$$z_{u,m} = \frac{R_{u,m} - \mu_u}{\sigma_u} \tag{2.2}$$

First, the 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. (2.1) is applied to the example in Table (2.1), 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. (2.1) is that ratings of constant raters are predicted correctly. Since the standard deviation of constant raters is 0, the second term of Eq. (2.1) 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 Eq. (2.1) is when $p_{i,j} = 1$ for all i and j, which is called the z-score average predictor ($zAvg$):

$$zAvg(u.m) = \mu_u + \left(\frac{\sum_{i=1}^{N} z_{i,m}}{N} \right) \sigma_u \tag{2.3}$$

2.2 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 [4]. In this study, an empirical relationship between the number of available ratings and the weights in the weighted sum formula is derived from the data set. From the Netflix training data set, 2000 movies were extracted each with at least 10,000 available ratings. The overall average was computed and then a random sample of different sizes, 1 to 1000, were selected to calculate the difference between the sample average and the overall average. The results are shown in Fig. 2.1. 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.

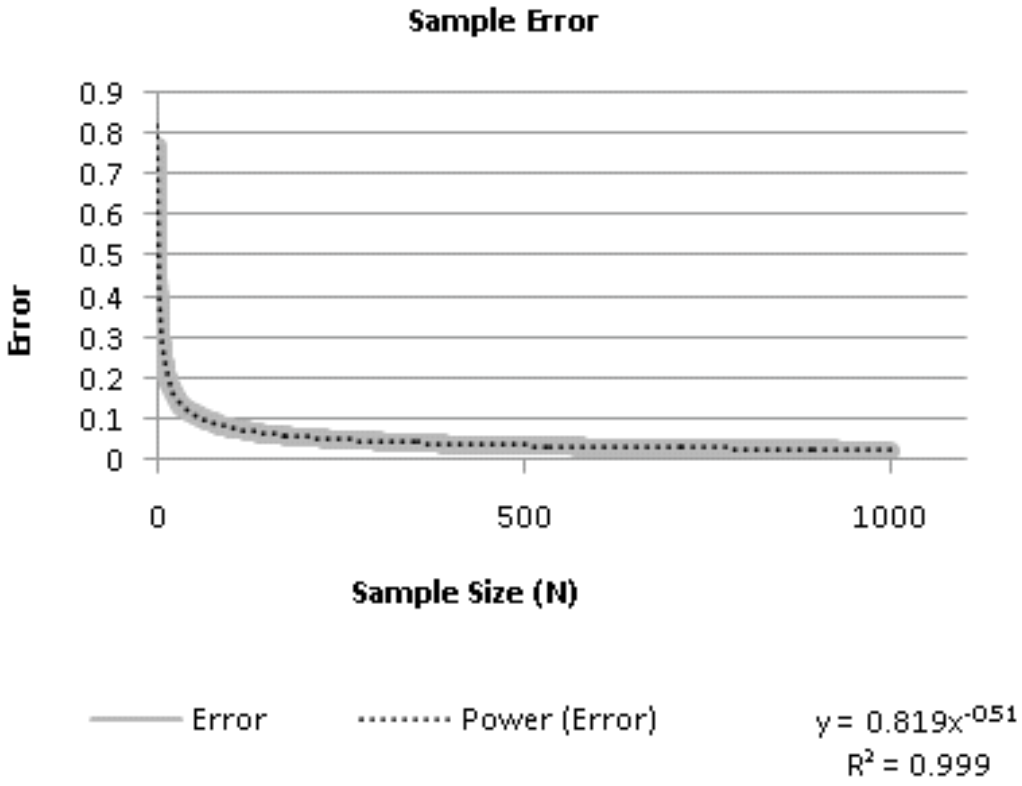

Figure 2.1: The errors between the subset sample average and the overall average for different number of samples, N. It is a power function of N.

Based on the empirical result, an adaptive weighting scheme called the *global gravitation mechanism* was designed 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 Algorithm 1.

Algorithm 1 Global Gravitation Mechanism

procedure ADJUST($Value, GA, T, N$)

$error \leftarrow \sqrt{\frac{T}{N}}$

if $Value > GA$ **then**

$newValue \leftarrow max(Value - error, GA)$

else

$newValue \leftarrow min(Value + error, GA)$

end if

return $newValue$

end procedure

Global gravitation first calculates the error range based on the number of samples (N) and the relationship derived from Fig. 2.1. Then, it returns the value from the range $[Value \pm error]$ that is closest to the global average (GA). 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. (2.3) equivalent to the explicit rating predictor in Eq. (2.4). The proof is shown as follows:

$$Avg(m) = \frac{\sum_{i=1}^{N} R_{i,m}}{N} \tag{2.4}$$

$$\begin{aligned}
GA &= \text{Global Average} \\
GSD &= \text{Global Standard Deviation} \\
\mu_u &= GA \text{ for all users} \\
\sigma_u &= GSD \text{ for all users} \\
zAvg(m) &= GA + \left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{R_{i.m} - GA}{GSD}\right)\right) GSD \\
&= GA + \frac{1}{N} \times \frac{1}{GSD}\left(\sum_{i=1}^{N} R_{i.m} - \sum_{i=1}^{N} GA\right) GSD
\end{aligned}$$

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

2.3 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, Eq. (2.1) is first modified by multiplying each term of the weighted average by a sigmoid function of the absolute value of the PCC:

$$uCF(u,m) = \mu_u + \left(\frac{\sum_{i=1}^{n} \left(p_{u,i} \times z_{i,m}\right) S(p_{u,i})}{\sum_{i=1}^{n} |p_{u,i}|} \right) \sigma_u \tag{2.5}$$

$$S(x) = \frac{1}{1 + e^{-25(|x|-0.1)}} \tag{2.6}$$

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 $p_{u,i}$ is small, the $S(p_{u,i})$ will be small, making the second term in Eq. (2.5) 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 the PCC, making it go to zero as the number of available ratings becomes low (the size of the common set between two users), which makes the weighted average go to the user average as the data becomes sparser.

It has been 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. Various nonlinear transformation functions were tested, including polynomial, sigmoid, and Gaussian functions, on the PCCs and found that the square function works well. Thus, the final version of the user-based CF formula is as follows:

$$uCF(u,m) = \mu_u + \left(\frac{\sum_{i=1}^{n} \left(P_{u,i} \times z_{i,m} \right) S(P_{u,i})}{\sum_{i=1}^{n} |P_{u,i}|} \right) \sigma_u \tag{2.7}$$

$$P_{u,i} = \frac{p_{u,i}}{|p_{u,i}|} \times p_{u,i}^2 \tag{2.8}$$

2.4 Combining Item-Based and User-Based Prediction

The movie-base collaborative filtering in the literature is computed similarly to the user based CF in Eq. (2.1). The formula of the basic movie-based CF (mCF) is:

$$mCF(u,m) = \mu_m + \left(\frac{\sum_{i=1}^{n} P_{m,i} \left(\frac{R_{u,i} - \mu_i}{\sigma_i} \right) S(P_{m,i})}{\sum_{i=1}^{n} |P_{m,i}|} \right) \sigma_m \tag{2.9}$$

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 the improved method, before calculating the similarity between two items, their ratings were first transformed into z-scores as a way of putting all users on common grounds before comparison. Fig. 2.2 and Fig. 2.3 show the significant difference between the correlation coefficient distributions using the raw ratings and the z-scores. Using the Netflix data set, Fig. 2.2 shows the distribution of the movie-to-movie PCCs based on raw ratings. The majority values are positive. On the same

data set, Fig. 2.3 shows the distribution of the movie-to-movie PCCs based on corresponding z-scores (called $zPearson$). 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.

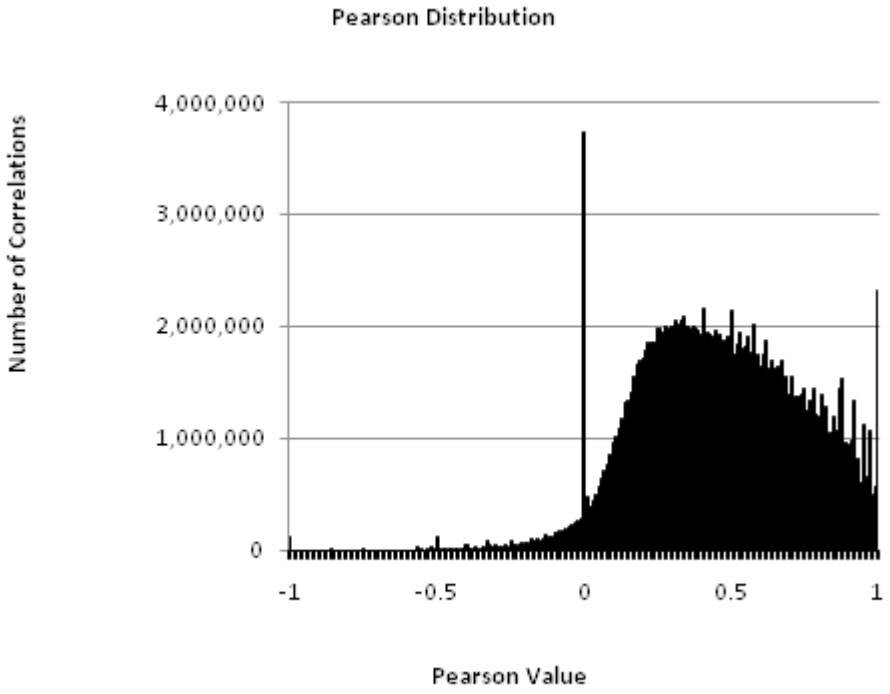

Figure 2.2: The distribution of the movie-to-movie PCC based on the raw ratings of the Netflix data set.

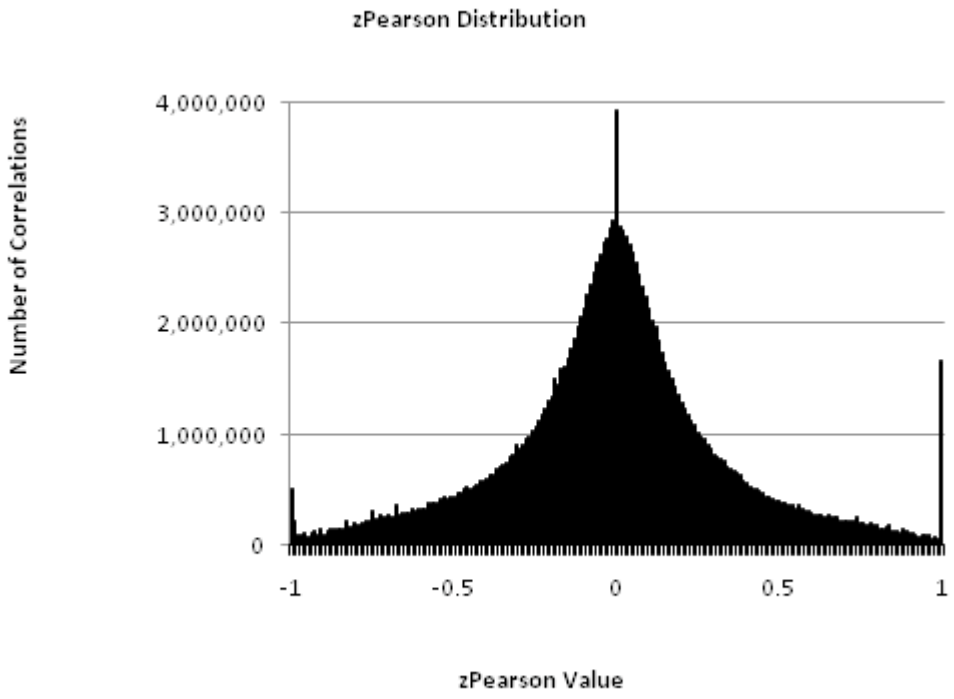

Figure 2.3: The distribution of the movie-to-movie PCC based on the z-scores of the Netflix data set.

In this method, the movie-based prediction is combined with user-based prediction by using z-scores instead of the raw ratings. Since the similarities were computed by comparing the z-scores, the prediction formula has to be adjusted such that it will be a z-score predictor. The predicted z-score will then be transformed back to a raw score based on the average and standard deviation of the user in question. The z-score movie-based-CF predictor (called zCF) is as follows:

$$zCF(u,m) = \mu_u + Z_{u,m} \times \sigma_u \tag{2.10}$$

$$Z_{u,m} = {}_z\mu_m + \left(\frac{\sum_{i=1}^{n} {}_zP_{m,i} \left(\frac{z_{u,i} - {}_z\mu_i}{{}_z\sigma_i} \right) S({}_zP_{m,i})}{\sum_{i=1}^{n} |{}_zP_{m,i}|} \right) {}_z\sigma_m \tag{2.11}$$

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 ${}_zP_{i,m}$ 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 [4] and the adjusted cosine similarity in [13]. The z-score is more informative and effective than the others since it considers the variance. For example, two users A and B with the same average (e.g. 3) but different standard deviation (e.g. 1 and 2) will have a different z-score for the same movie that they rated 4. The z-scores will be 1 for A and 0.5 for B which gives us an insight of how much the user liked the movie based on how often they deviate from their average. In this example, user A is less likely to deviate from his average than user B (more strict), therefore It can be concluded that he has liked the item more than user B.

2.5 Cluster-Based CF

Clustering has been used in recommendation systems mostly for finding similar users [6] [12]. In this project, clustering is used 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, the global gravitation mechanism is applied 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. 2.4 and

(2.5) illustrate the concept.

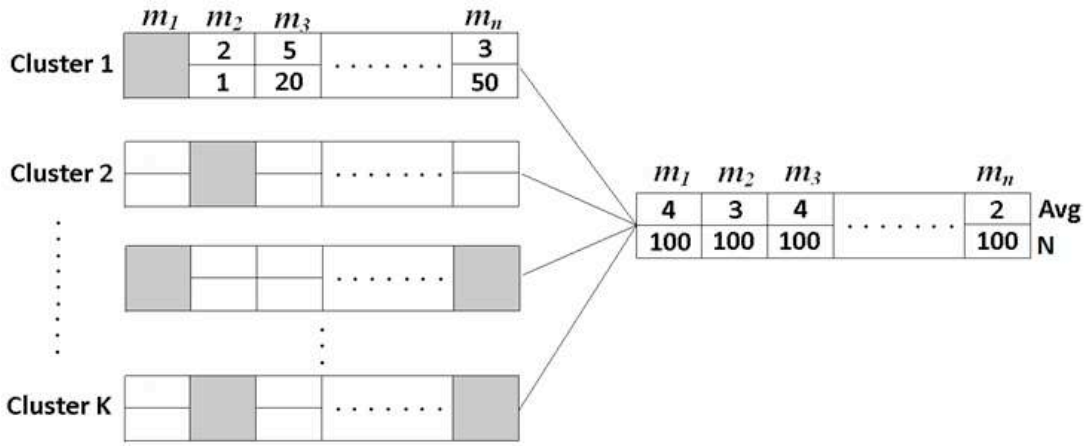

Figure 2.4: Decomposing the global average vector into k sub-vectors (clusters).

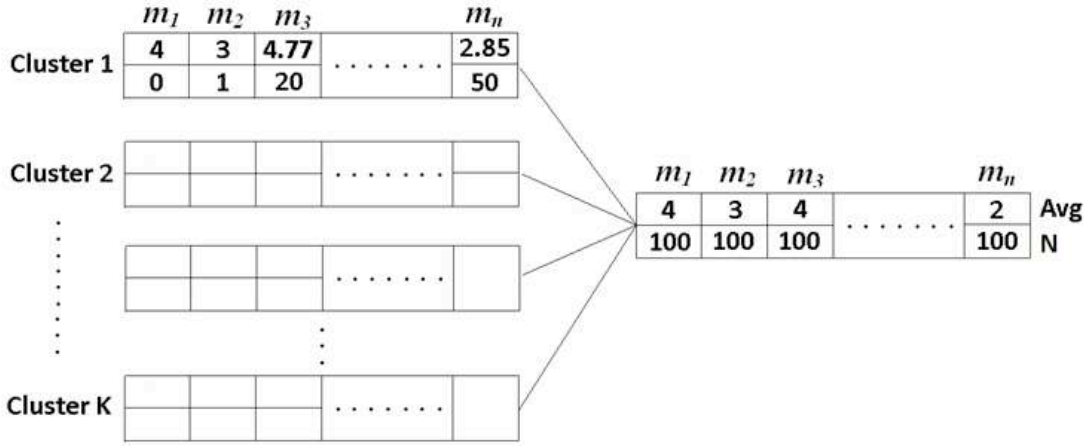

Figure 2.5: Applying the global gravitation mechanism to the k sub-vectors (clusters) with $T = 1$.

In Fig. 2.4 and Fig. 2.5, 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. 2.4 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, predictions based on clustering will approach the global movie average predictor if the clusters are too sparse. In Fig. 2.4 and Fig. 2.5, raw ratings were used to represent the centroids, but the z-scores can be used as well. Two clustering-based CF methods were implemented, one based on raw rating as in Eq. (2.12) and the other based on z-scores as in Eq. (2.13). 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.

$$Cluster(u, m) = \frac{\sum_{c=1}^{K} (\alpha_{u,c} \times \mu_{c,m})}{\sum_{c=1}^{K} \alpha_{u,c}} \tag{2.12}$$

$$zCluster(u, m) = \mu_u + \left(\frac{\sum_{c=1}^{K} (\alpha_{u,c} \times {}_z\mu_{c,m})}{\sum_{c=1}^{K} \alpha_{u,c}} \right) \sigma_u \tag{2.13}$$

$$\alpha_{u,c} = \frac{1}{Euclidean_{u,c}} \tag{2.14}$$

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.

Chapter 3

The New Adaptive Framework

3.1 Building the Framework

Recommendation systems normally implement a user-based CF, an item-based CF or a combination of the two methods [14]. The combination is usually done by fusing the final result of individual algorithms and treating them as black boxes instead of integrating them during the prediction calculation process. The idea of converting raw scores into more comparable scales has been introduced before but failed on sparse data. In this study, a new CF framework is introduced that integrates the user-based CF and the item-based CF in a complimentary way during the prediction calculation process such that it will lean towards the item-based CF, the user-based CF, or a hybrid depending on the amount of data available. The new framework is developed to adapt to the sparsity so that it will lean more towards the comparable scale, the z-score, if the amount of data is large, or towards the raw score, the ratings, if the amount of ratings is small.

The idea behind the new framework is specialization when more data is available and generalization when less data is available. To illustrate the concept, consider the simplest predictor, the global average predictor, which is also the most general one. We can of course improve the prediction by being more specific, i.e., using the movie average. However, if the movie has not been seen a lot, then its

average is not reliable. This is when the global gravitation is beneficial. To gain more accuracy we try to be as specific as possible from the given information in the query, using the movie average, z-score, cluster average, etc., while adjusting for low number of ratings by generalizing (applying global gravitation).

The final framework is basically a multiple runs of zCF where in each run the global movie average is replaced by a cluster average. The closest K clusters are selected and their weighted average is returned, using the inverse Euclidean distance as weights. The final framework is called $zCCF$ (z-score based Clustered Collaborative Filtering), shown in Eq. (3.1).

$$zCCF(u,m) = \mu_u + \left(\frac{\sum_{c=1}^{K} \alpha_{u,c} \times zCF(u,m,c)}{\sum_{c=1}^{K} \alpha_{u,c}} \right) \sigma_u \tag{3.1}$$

$$zCF(u,m,c) = {}_z\mu_{m,c} + \left(\frac{\sum_{i=1}^{n} {}_zP_{m,i} \left(\frac{z_{u,i} - {}_z\mu_{i,c}}{{}_z\sigma_{i,c}} \right) S({}_zP_{m,i})}{\sum_{i=1}^{n} |{}_zP_{m,i}|} \right) \times_z \sigma_{m,c} \tag{3.2}$$

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

3.2 The Adaptability of the Framework

$zCCF$ is an adaptive CF framework encapsulating the various CF methods discussed in chapter 2, including Avg as in Eq. (2.4), $zAvg$ in Eq. (2.3), uCF in Eq. (2.5), mCF in Eq. (2.9), zCF in Eq. (2.10), $Cluster$ in Eq. (2.12), and $zCluster$ in Eq. (2.13). Fig. 3.1 shows these methods and their relationships.

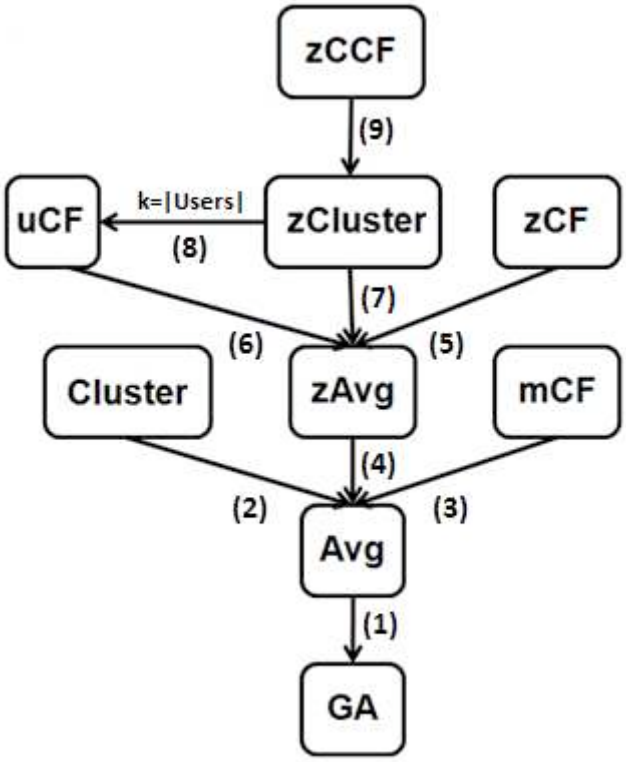

Figure 3.1: Methods hierarchy of the $zCCF$ framework: The arrows can be read as "is a general form of". $|Users|$ is the number of users and k is the number of clusters.

In Fig. 3.1, 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, the method can be adjusted to adapt to the amount of data so that it becomes a general algorithm as the amount of information reduces and a specific 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. 3.1. Their meanings 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 $S(x)$ to 0 in Eq. (2.9)

4. *zAvg* becomes *Avg* when the global gravitation mechanism sets all users' averages and standard deviation to the global values for sparse data

5. zCF becomes *zAvg* when the nonlinear transformation removes the effect of low-similar movies from zCF, i.e., by setting $S(x)$ to 0 in Eq (2.10)

6. uCF becomes *zAvg* by ignoring similarities and setting $p_{i,j} = 1$ in Eq. (2.5)

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 $S(x)$ to 0 in Eq (3.1)

Chapter 4

Experimental Results

In the following experiments, the prediction accuracies of the proposed techniques are compared on data sets with different densities. The data sets were extracted from the Netflix data set. First, 10,000 users with the most number of ratings were selected, 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, a random number of ratings were removed.

4.1 Global Gravitation

Fig. 4.1 compares the results of Avg, $zAvg$, and $zAvg{+}GG$ ($zAvg$ with the global gravitation mechanism) predictors with data sets of different density. $zAvg$ is better than Avg using dense data sets, but worse on sparse data sets. For example, $zAvg$ is 10.2% better than Avg using a 45% dense set. However, $zAvg$ performs much worse when density is below 10%. By using global gravitation, zAvg+GG achieves good results on both sparse and dense sets, consistently better than Avg and $zAvg$. Its accuracy is similar to $zAvg$ on dense sets and approaches Avg on very sparse data sets.

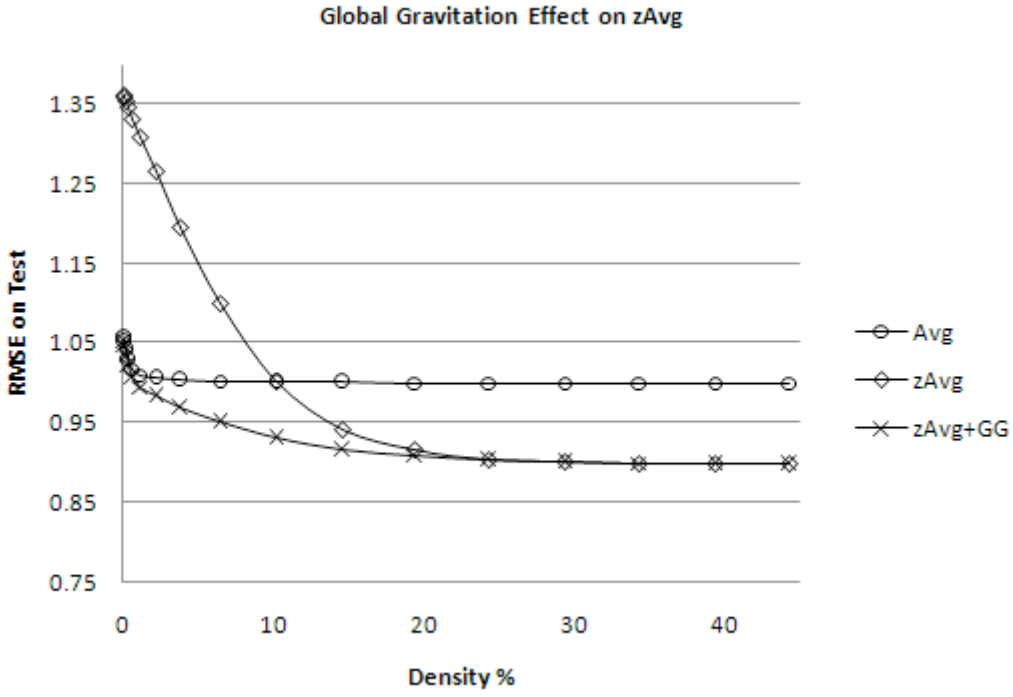

Figure 4.1: Comparison between Avg, $zAvg$, and $zAvg + GG$ ($zAvg$ with the global gravitation mechanism) predictors on data sets of different density.

4.2 Non-linear Transformation

Fig. 4.2 compares the results of Avg, zCF without transformation, zCF with a transformation function of $|{}_zp_{a,b}|$ ($zCF \times |zPearson|$), and zCF with a transformation function of $Sigmoid({}_zp_{a,b})$ ($zCF \times Sigmoid$) on data sets of different densities. 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 $|{}_zp_{a,b}|$ and $Sigmoid({}_zp_{a,b})$ consistently outperforms Avg. zCF with $Sigmoid({}_zp_{a,b})$ consistently outperforms zCF with $|{}_zp_{a,b}|$ as well.

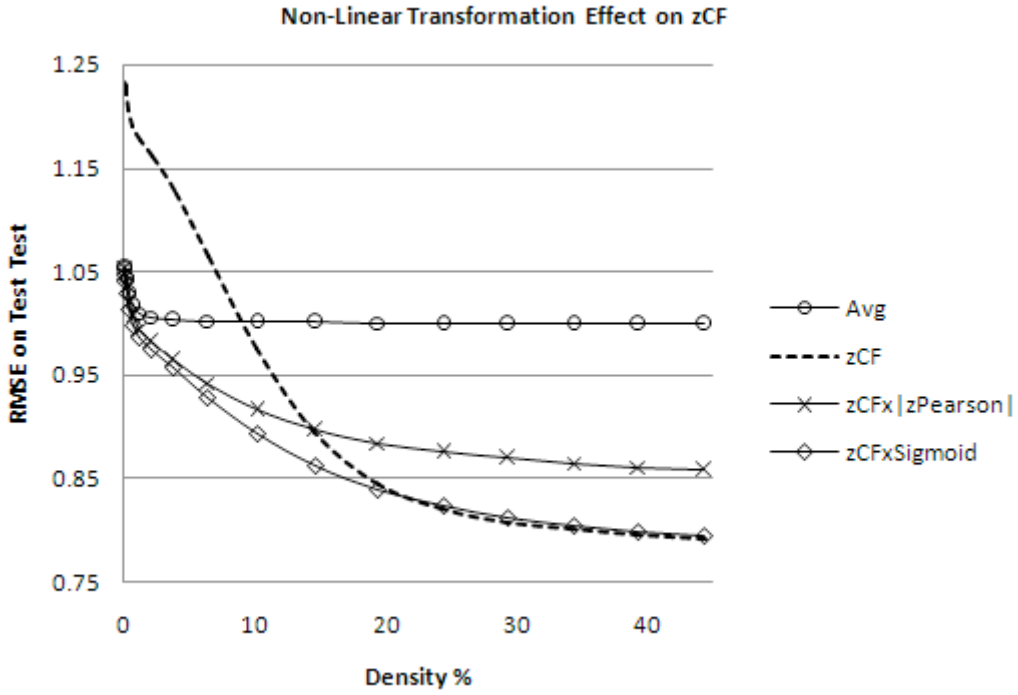

Figure 4.2: Comparison between Avg, zCF without transformation, zCF with a transformation function of $|zPearson|$ ($zCF \times |zPearson|$), and zCF with a transformation function of $Sigmoid(zPearson)$ ($zCF \times Sigmoid$) on data sets of different density.

4.3 Z-scores vs. Explicit Ratings

Fig. 4.3, Fig. 4.4 and Fig. 4.5 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 $zAvg$ is 10.25% better than Avg, $zCluster$ is 4.2% better than $Cluster$, and zCF is 3.4% better than mCF.

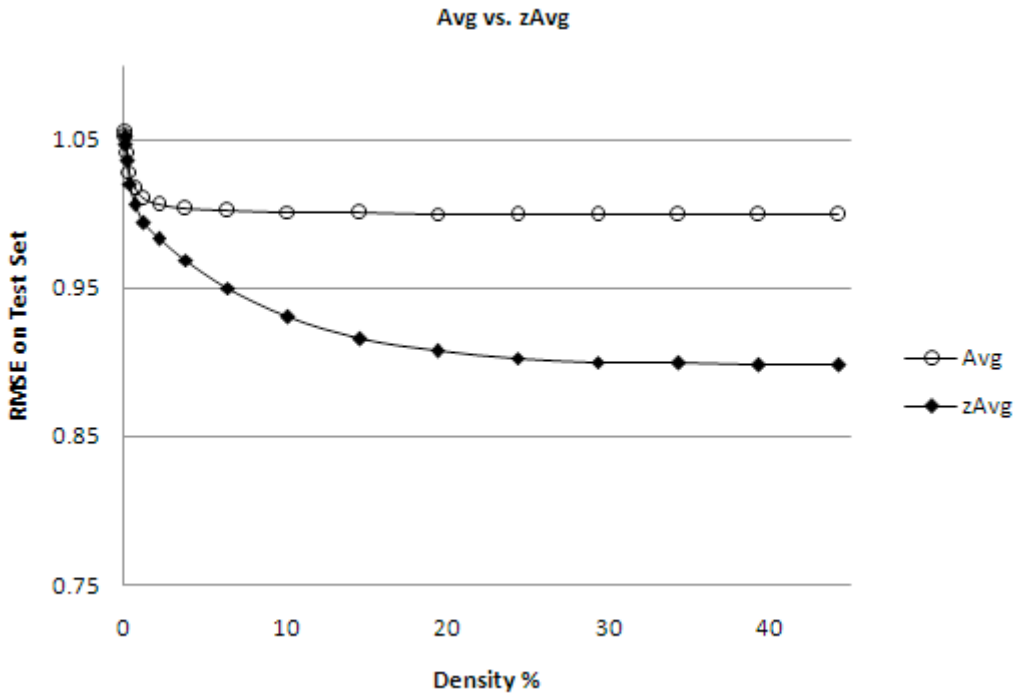

Figure 4.3: Comparison between Avg (average of raw ratings) and $zAvg$ (average of z-scores) on data sets of different densities.

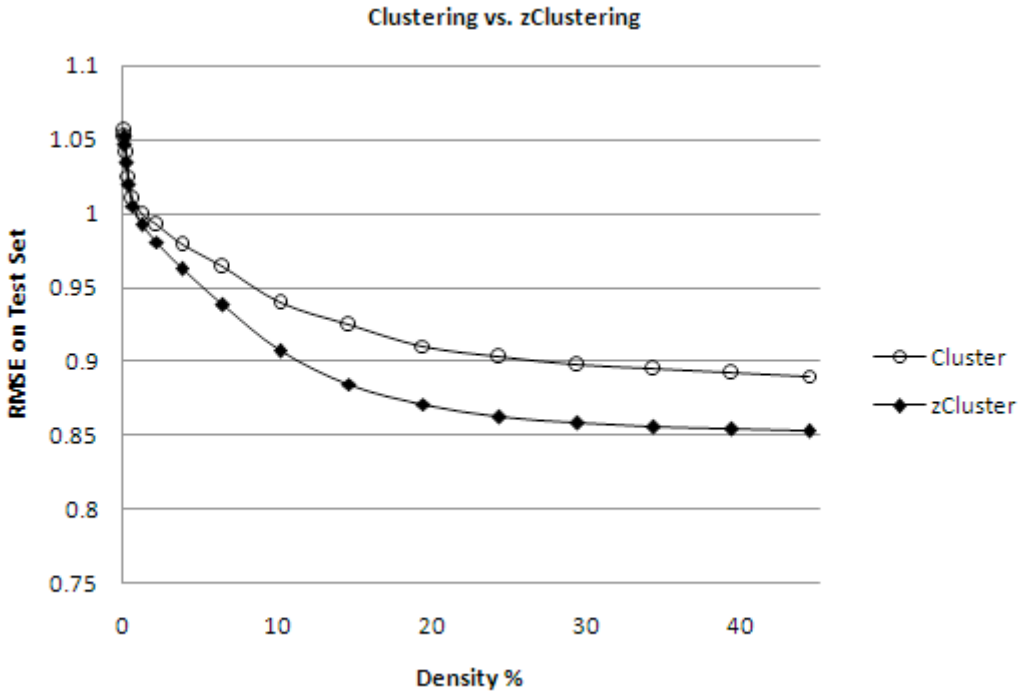

Figure 4.4: Comparison between clustering based on the z-scores and raw ratings on data sets of different densities.

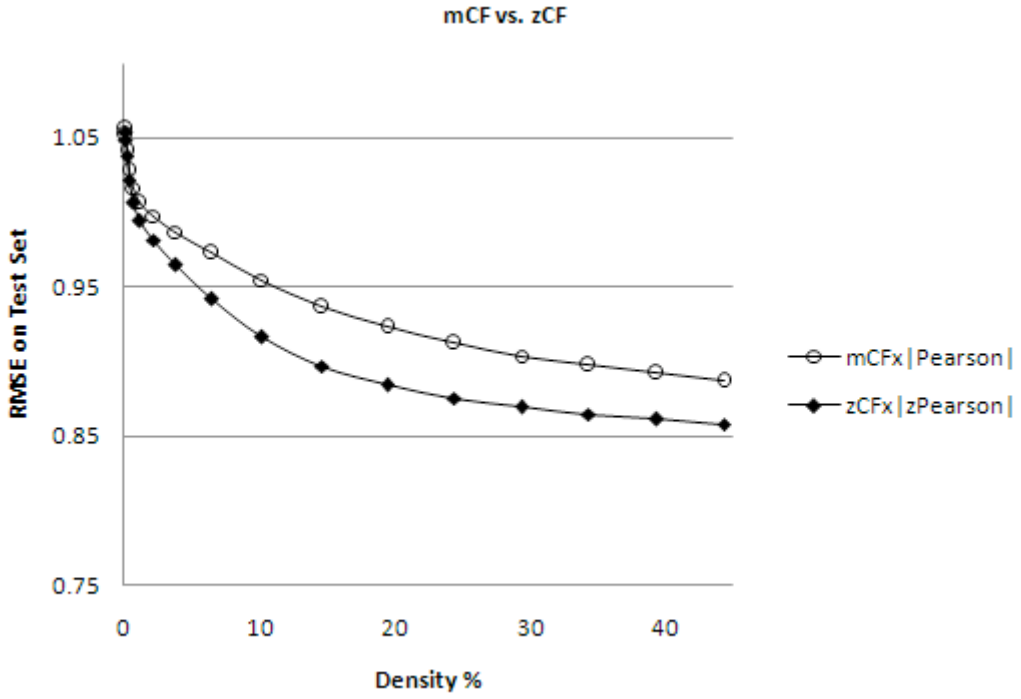

Figure 4.5: Comparison between zCF and mCF on data sets of different densities.

4.4 Methods Hierarchy Test

Fig. 4.6 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.

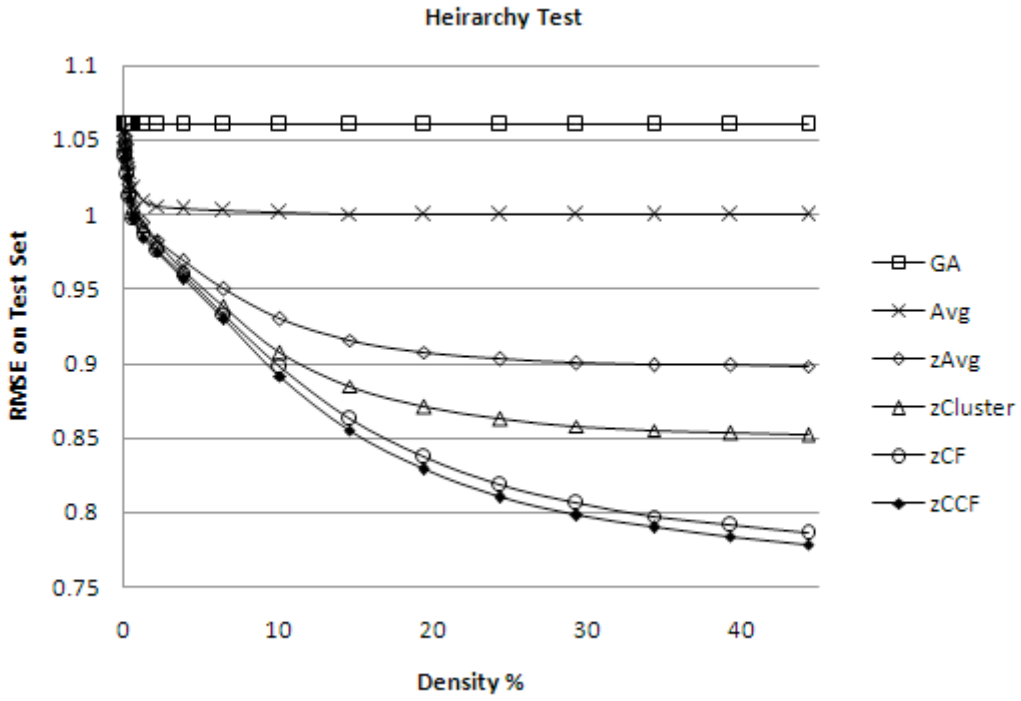

Figure 4.6: Performance comparison of various methods represented in the adaptive framework of $zCCF$ hierarchy on data sets of different density.

4.5 zCCF vs. Improved Regularized SVD

$zCCF$ was also compared to the Improved Regularized SVD ($IR{-}SVD$) in [11], which is significantly better than the basic implementation of SVD. As shown in Fig. 4.7, $zCCF$ is slightly better than SVD and achieves 2% improvement on dense data.

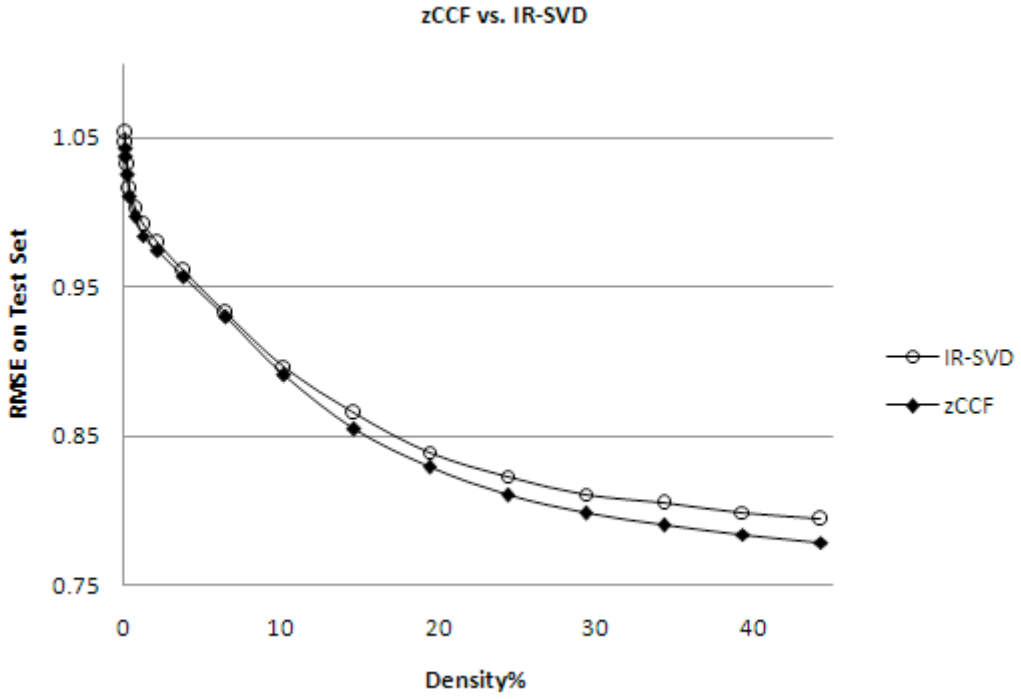

Figure 4.7: Comparison between the improved regularized SVD in [11] and $zCCF$ on data sets of different densities.

4.6 Results on the 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. 4.8 shows the RMSEs of predictions by the methods studied in this project, 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 methods and is 4.67% better than Cinematch. The averaging of $zCCF$'s predictions and SVD's predictions was submitted and achieved 5.6% improvement over Cinematch.

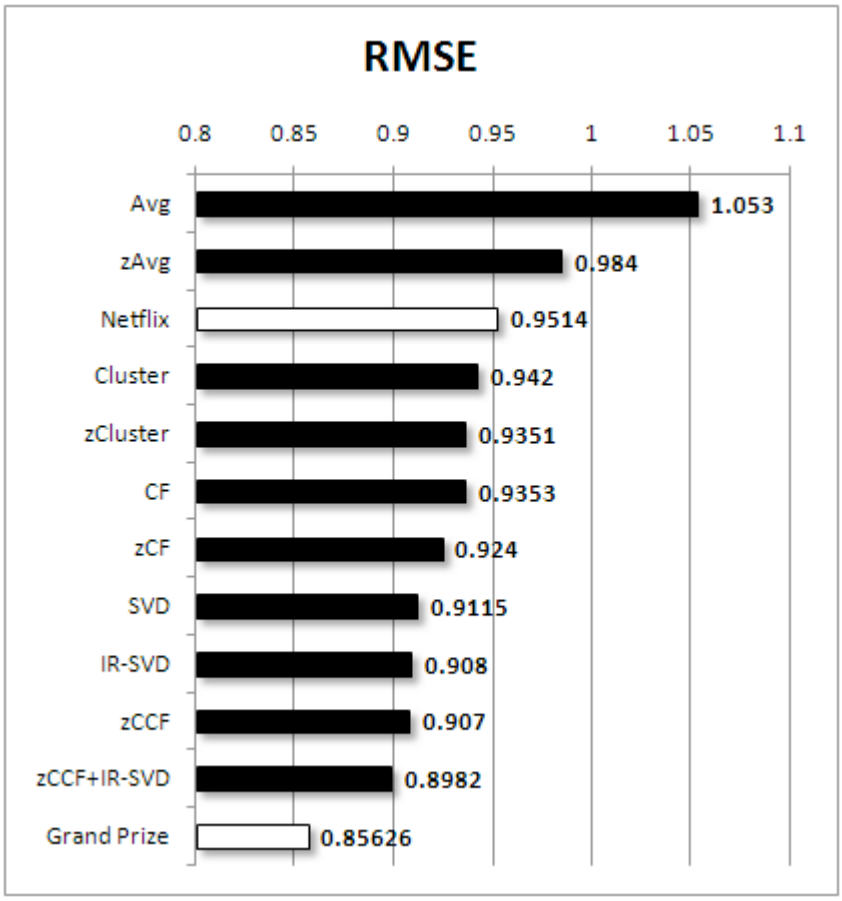

Figure 4.8: Results on the Netflix challenge test set.

Chapter 5

Failed Attempts

In this chapter, we present the results and lessons learned from the un-successful techniques that were implemented during the course of this study. They were not successful mostly due to the nature of the date set. Those techniques were movie-based clustering, nearest neighbor user-based clustering, and content based average prediction.

5.1 Movie Based Clustering

A movie based clustering algorithm similar to the user based clustering algorithm was implemented but produced little improvement over simple averaging. It is because the user to movie ratio is much higher than movie to user ratio. In other words, most of the users have seen very few movies, which is evident in Fig. 1.2 and Fig. 1.3, or very similar movies that made the clusters so sparse that the global gravitation had a very large effect on them, therefore approaching the global average. That's not true for the user clustering since movies tend to have a much diverse set of users who have rated them.

5.2 User-Based Nearest Neighbor Clustering

K-means has the disadvantage of mishandling outliers since they have to be placed in one of the k clusters and therefore affecting the centroid of that cluster. For that reason, Nearest Neighbor clustering algorithm was implemented so that outliers will be placed in a new cluster. However, NN has two disadvantages. One is being sensitive to the order in which users are presented to the algorithm. For example, if the ten data points 1,2,3,4,5,6,7,8,9 and 10 were presented to the NN algorithm in the same order with a threshold of 1, all the data points will be placed in the same cluster. The other is the choice of appropriate threshold is difficult.

A combination of k-means and NN was implemented that solved both problems, but was very slow on the Netflix data set. The method works as follows. First, a user is placed into the closest cluster where its distance to the cluster centroid is less than the threshold, instead of the cluster where the closest user with a distance less than the threshold belongs, therefore eliminating the order problem. From the previous example, after presenting the first two data points are presented to the algorithm, 1 and 2, the will be placed in the same cluster with a centroid of 1.5. After that, 3 will be presented to the algorithm which has a distance of 1.5 from the cluster centroid, which is larger than the threshold. Therefore, 3 will be placed into a new cluster. Second, the threshold, for each user, was chosen to be the distance between the user's ratings and the global average vector. The idea is, if there is no cluster that can produce an error less than the error produced by using the global average vector, then place that user in its own cluster where it can be ignored later if no one else joined its cluster. The pseudo code for the algorithm is shown in Algorithm 2:

Algorithm 2 User-Based Nearest Neighbor Clustering

```
clusters ← initialize k empty clusters
Assign users randomly to the k clusters
repeat
    Compute centroids
    for each user in users do

        threshold ← distance(user, average vector)
        minimum ← ∞
        newCluster ← ∅

        for each cluster in clusters do
            distance ← distance(user, cluster)
            if distance < minimum AND distance < threshold then
                minimum ← distance
                newCluster ← cluster
            end if
        end for

        current ← getCluster(user)
        current ← current − user

        if newCluster = ∅ then
            clusters ← clusters ⋃ {user}
        else
            newCluster ← newCluster ⋃ user
        end if

    end for
until there is no more user movement or a certain number of maximum iterations has been reached
```

This clustering approach was applied to the dense sub matrix in which the RMSE of the algorithm reached 0.823 compared to the 0.8522 reached by *zCluster*. The technique could not be applied to the full data set because most users were placed into their own clusters which caused a memory overhead. However, it is highly unlikely that it will translate into a significant improvement since the

dense data set only achieved a 0.03 improvement. From the results we have seen, that will likely translate into a 0.00X improvement in the full data set.

5.3 Content-Based Average Prediction

The $zCCF$ algorithm uses the global average (3.45) in the case of low number of ratings. An extension to the framework was implemented such that the algorithm will use the average of the most similar movie based on the content. The content information was very difficult to gather due to the nature of the Netflix data set. The data set is more of a DVD data set than a movie data set. There exist a great deal of training videos, music videos, 2 movies pack, bounce DVDs and multiple versions of the same movie like special edition, collector's edition, 20th anniversary edition etc. There is also the problem of multiple titles with the same name but different years. The DVDEmpire web site, which is an on line DVD store, was chosen to gather the information. A crawler was implemented that took a title as its input and returned the genres, cast, writer(s) and director(s). The crawler was not accurate due to the previously mentioned problems but a manual inspection for 100 movies was 80% accurate. Every feature was treated equally, meaning that a director, writer, genre or cast have equal weights when comparing two movies. The Jaccard coefficient was chosen as the similarity measure. The Jaccard coefficient calculates the similarity between two sets A and B as follows:

$$Similarity(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{5.1}$$

The idea of using the content information is to replace the constant global average with the average produced by the content based predictor. The same most popular 2000 movies that were selected for the global gravitation experiment in Fig. 2.1 were chosen to measure the accuracy of the content based predictor. In this case, we are trying to predict the average of the movie based on the most similar movies by using the content information. The predicted average is

a result of a weighted average of the top n similar movies. On the 2000 movies selected for that experiment, the constant global average predictor produced an RMSE of 0.39, while the content based predictor produced an RMSE of 0.31. However, when the approach was tested on the entire data set, there was no significant gain, only in the 0.000X range. It is likely due to the inaccuracy of the information gathering process. The top 2000 movies are the most popular ones, therefore it is likely that the content information gathering was more accurate in this subset than the whole data set.

Chapter 6

Summary and Conclusions

In this study, we have developed a number of new techniques 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. The new 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 into the adaptive framework.

A conference paper resulting from this study has been accepted for publication in the proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC 2008) within the IEEE World Congress on Computational Intelligence (WCCI 2008) on June 2008 by Ibrahim Almosallam and Yi Shang.

Bibliography

[1] Kamal Ali and Wijnand Van Stam. Tivo: Making show recommendations using a distributed collaborative filtering architecture. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 394–401, 2004.

[2] Marko Balabanovic and Yoav Shoham. Fab: content-based, collaborative recommendation. In *Communications of the ACM*, volume 40, pages 66–72. ACM, March 1997.

[3] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 714 – 720. AAAI, 1998.

[4] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–107, 2007.

[5] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52, 1998.

[6] Gang Chen, Fei Wang, and Changshui Zhang. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. In *Proceedings of the*

Seventh IEEE International Conference on Data Mining Workshops, pages 303–308, 2007.

[7] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. In *ACM Transactions on Information Systems (TOIS)*, volume 22, pages 143 – 177. ACM, January 2004.

[8] Adomavicius Gediminas and Tuzhilin Alexande. Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. In *IEEE Transactions on Knowledge and Data Engineering*, volume 17, pages 734–749, June 2005.

[9] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative. In *IEEE INTERNET COMPUTING*, volume 7, pages 76–80. IEEE COMPUTER SOCIETY, 2003.

[10] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of DL-00, 5th ACM Conference on Digital Libraries*, pages 195–204, San Antonio, US, 2000. ACM Press, New York, US.

[11] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD Cup and Workshop 2007*, 2007.

[12] Al Mamunur Rashid, Shyong K Lam, George Karypis, and John Riedl. ClustKNN: a highly scalable hybrid model and memory-based cf algorithm. In *Web KDD 2006*, 2006.

[13] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.

[14] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.