

Introduction

This project is motivated by the "Netflix Prize Challenge" and the fact that predictions algorithms are still far from being satisfactory. Netflix is a movie renting company that has developed a movie recommendation system: **CinematchSM**. Its job is to predict whether someone will enjoy a movie based on how much they liked or disliked other movies. They use those predictions to make personal movie recommendations based on each customer's unique tastes. They are looking to improve the prediction of Cinematch to serve their customers better. Therefore, they announced this contest carrying a mega Award of \$1 million for achieving 10% more accuracy over current Cinematch prediction and progress award of \$50k for best program achieving 1% more accuracy over a declared rmse.

All of these recommendation systems use the principal collaborative filtering in one way or the other way. Collaborative filtering (CF) is the method of making automatic predictions (filtering) about the interests of a user by collecting taste information from many users (collaborating). The underlying assumption of CF approach is that: Those who agreed in the past tend to agree again in the future. A lot of research has been done in this field giving birth to item based filtering, user based filtering, content based predictions and many mixture models. We also aim to solve this problem of prediction using one of the most common approaches of collaborative filtering called neighborhood-based predictions.

Goal

Our goal is similar to the goals set by NetFlix Challenge which has been listed below according to priority. We need to achieve

- $rmse < 0.9525$ on probe data.
- $rmse < 0.9514$ on qualifying data.
- $rmse < 0.8572$ on qualifying data.

In words of contest organizers themselves achieving rmse of 0.8572 is pretty tough but nothing is impossible.

Dataset

Following is a brief description of the dataset.

- There are 17770 movies.
- There are 480189 users. CustomerIDs range from 1 to 2649429, with gaps.
- Ratings are on a five star (integral) scale from 1 to 5.
- YearOfRelease range from 1890 to 2005.
- Training set consists of 100 million records.
- Qualifying dataset size is 2817131. It contains from 1-9999 movies ids. Prediction needs to be submitted on this dataset.
- Probe dataset size is 1408395. It contains from 1-9999 movies ids. This dataset is meant to be used for checking the rmse before proceeding for qualifying dataset prediction.

File Format :
MovieID1:
CustomerID11,Rating,Date11
CustomerID12,Rating,Date12
...
MovieID2:
CustomerID21,Rating,Date21

First Approach of Averages

The most primitive approach will be to predict the rating of a movie for an active user same as the average rating given by him. But, we go a step ahead to make a hypothesis and test it.

"A person tends to rate a movie primarily based on his own test which occasionally gets biased by the environment."

Therefore, we hypothesize that a weighted sum of the average rating given by user and the average rating a movie receives should give less rmse on predictions than any one of these considered alone. So prediction value p of a movie m for an active user x is given by

$$p = \alpha * \mu_x + \beta * \mu_m$$

where μ_x mean rating user x and μ_m is the mean rating of a movie m . We tried to learn α and β on a given training set through iteration and minimization of rmse. Value of α converged to 1.0 and β to 0 with rmse of 1.0162. Training set had 1 million records. So, our hypothesis is no longer true and we can say that user average rating can be alone considered for computing maximum boundary of rmse.

Result

Applying the global average rating (~3.6) to all predictions: rmse = 1.13.

Applying each movie's average rating on a per-movie basis: rmse = 1.05.

Applying the average of the movie's average rating and the User's average rating: rmse = 1.017.

Doing the above but weighted w/ regard to standard deviation: rmse = 1.013.

To start with, we seek to propose methods that should perform better than rmse obtained on predications based on user average rating.

Methods and Algorithms

We are trying to approach the problem using pure collaborative filtering. Neighborhood-based algorithms are well known in literature for obtaining good predictions. In the approach a subset of users are chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for the active user. The algorithm can be summarized in the following steps:

- Weight all users with respect to similarity with the active user.
- Select n users that have the highest similarity with the active user. They form the neighborhood.
- Compute a prediction from a weighted combination of the selected neighbor's ratings.

We have tried three different approaches of finding similarity, clustering users and finally predicting rating for a given movie of a given active user.

Correlation Approach (Literature)

Similarity between the active user x and any other user y is computed using the Pearson correlation coefficient as it is well known to produce better results than other similarity measures. Top n users sharing highest correlation with x form its neighborhood. Prediction of the rating for the given movie m for the user x is done by

$$p_{x,m} = \mu_x + \sum (r_{y,m} - \mu_Y) / n$$

where $r_{y,m}$ is rating given by user y to movie m and μ_Y is average rating given by user y . Correlation coefficients are calculated as preprocessing step using MTL library for optimized multiplication of sparse matrices. Parallel technique is used to compute the coefficients efficiently.

Result

On Probe test dataset rmse is 1.0010.

Resource Used

We used 5 computers running for 5 days to complete computation of coefficients for 480189 users.

Error Method (New)

This is a new approach proposed by us. Motivation of this approach comes from the very fact that we are trying to minimize the rmse on our prediction set. To determine neighborhood of an active user x we do two level of filtering. Steps are as follows.

Step1:

Filtering

This is the first level of filtering. Here we determine a set of users who have seen at least ($k\%$ of n) movies similar to the active user where n is the number of movies seen by the active user. All the users who passes the threshold value k are considered candidate for next step.

Determining Threshold

Threshold is not a fixed value but depends on the value n . We claim that since 10% of 100 \neq 10% of 1000, Therefore it will not be fair to keep the value of threshold constant for all values of n . As n increases, threshold should increase to recognize stronger relation between 2 users e.g. if $n=100$ than number of common movies between users should be 50 or more which comes at 50%. If $n=1000$, than number of common movies should be more than 600 to get a stronger relation that is 60%.

Our values are following:

Scale1: 40% , Scale2: 50% ,Scale3 : 60% , Scale4: 67% and Scale5:75%

where scale stands for number of digits in n .

These threshold values can also be learnt by doing a number of iterations on a training data. It should be varied such that the prediction error is minimized on iteration. Iterations should be stopped if error starts increasing.

Step2:

If candidate set obtained from step1 is empty then we default the predicted rating of the movie to the average rating of the active user otherwise go to step3. Experimentally we found that defaulting to average performs better than considering candidates with lower number common movies.

Step3:

This is the second level of filtering. This step is performed in addition to step1 to get a tighter set of candidates who not only share a good number of common movies with active user but rmse between their ratings is small. All the candidates obtained from step1 are subjected to a rmse test. We compute the rmse between the ratings of the active user and a candidate on all the common movies they have watched. All the candidates who passes a threshold R are considered for further processing. R can again be learnt using a sample training dataset. In our case this value came out to be 1.50.

Step4:

If no candidate is obtained from the step3 then we use the candidate set obtained from step1 in place of defaulting to average value. Experiments showed that considering candidates with even higher rmse performed better than defaulting to averages. Average rating is our max boundary. If we get a set of candidates we proceed to step5.

Step5:

We try to find the rating of the required movie m from all the candidates. Ratings obtained from each candidate are adjusted to include the absolute error between the candidate and the active user on the set of common movies. If we get a set of ratings we proceed to step6 otherwise we are forced to take average rating of the active user as the predicted rating of the movie.

Step6:

All the ratings obtained from step5 are put into 5 bins depending on their values. Bins are 0-1.5,1.5-2.5,2.5-3.5,3.5-4.5,4.5-5.0. Finally the bins having largest number of members are considered. Final predicted value is the average of these values.

Results

Probe dataset : 500,000 movie prediction with rmse 0.9053. It's better than Cinematch rmse of 0.9525.

Quiz dataset : 2million predictions , rmse 0.9941.

Resources Used

For Probe set:

- 30 hrs of computation power.
- 10 machines.

For Quiz set:

- 24 hr of computation power.
- 40 machines.

Suspected reasons for poor performance on Qualifying Set

- A large number of predictions defaulted to average because none of the candidates selected in step3 (error filtering) had watched the movie for which we are trying to predict. So rmse slowly approaches towards the rmse obtained on prediction set using user average rating.

Look ahead is included at step5 of error method approach if we fail to find a set of ratings for the required movie.

Result

Probe dataset rmse of 0.8710.

Resource Used

- 10 hours of computation .
- 10 machines.

User Association Method (New)

In this method, we go through all the users who rated the movie in question. For all those users, we find the movies that are rated by both that person and the person in question. For all those movies rated by both of them, we find how much they rated them in a correlated way. The average of all those correlations defines a distance between the two users. After finding all the distances between users, we assign a weighted rating to our users depending on those distances. If there isn't enough "support", then we assign a combination of user and movie averages to this pair.

Results

On probe data set rmse is 0.9622.

Resources used

We ran this after pre-processing and fitting everything to memory efficiently, it took 8 hours overnight on a single computer.

Improving on Time and Space Efficiency

We are using Java, which is not directly optimized for dealing with large amounts of data, it is an important direction to try to fit all this data to memory in an efficient data structure. First of all, we found out that all Objects (including Vectors, i.e. dynamic arrays) are stored in the heap space. Not only those objects take more time to reach (it is hard to go through the heap), they also take more memory. However, we can eliminate the use of heap space by first pre-processing the data and finding the exact sizes of each vector. Then, in the runs after preprocessing, we can store the data in static arrays which take up much less space and are just the fastest data structures available. After the preprocessing process (which took weeks), we were able to optimize run-time to 6 minutes to construct all data structures (from 17,770 movie files to 17,770 movie and 480,189 user structures), and just 950mb memory (It is 2GB in text files). We took advantage of storing each rating in byte arrays, each user in integer arrays and each movie in short arrays.

RMSE Result Summary

Methods	Probe Dataset	Qualifying Dataset
User Average	1.0150	1.0170
Correlation Coefficient	1.0010	
Error Method	0.9053	0.9941
Error Method with Look Ahead	0.8701	
User Association	0.9622	

Work Ahead

- Improving rmse on the prediction of extreme values.
- Trying SVD to reduce dimensionality and then use the error method approach to predict. Some participants have achieved good error rate on qualifying dataset using conventional methods.
- Combining content based prediction with pure collaborative filtering using IMDB data for movies.

Contributions

- Vishwakarma Singh : Average and Error Method approach.
- Sayan Ranu : Average and Correlation Coefficient approach.
- Onur Sakarya : Preprocessing and user association

Acknowledgements

- Martin Rhodes (DBL lab)
- Kris Kevikal (ITR lab)
- Vebjorn Ljosa (DBL lab)

References

- [Application of Dimensionality Reduction in Recommender System \(2000\)](#), by Badrul M. Sarwar, George Karypis, Joseph A. Konstan, John T. Riedl (GroupLens Research Group Army HPC Research Center Department of Computer Science and Engineering University of Minnesota) .
- [Clustering Items for Collaborative Filtering \(1999\)](#), by Mark O'Connor & Jon Herlocker (Dept. of Computer Science and Engineering, University of Minnesota)
- [Content-Boosted Collaborative Filtering for Improved Recommendations \(2002\)](#), by Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan (University of Texas at Austin).