

# Predicting ‘Who Rated What’ using Nearest Neighbor and Decision trees

Abijit Bej

Anand Kishore

Sudheer Ramoji

University of Southern California  
Los Angeles CA 90089-0273  
{abej, kishore, ramoji}@usc.edu

## ABSTRACT

Recommender systems apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction. These systems, especially the k-nearest neighbor collaborative filtering based ones, is achieving widespread success on the Web. This paper describes three possible approaches to solve who rated what of the KDD cup 2007. The key points of our approaches are (1) augmenting the data set with genre information (2) extracting unique features for identifying user similarity and movie similarity (3) exploring three different classification algorithms, which are appropriate to this domain. Finally we do a comparative study of above-mentioned approaches.

## Categories and Subject Descriptors

I.5.1 [Pattern Recognition]: Models - Statistical

## Keywords

Classification, predictive modeling, supervised learning, data mining

## 1. INTRODUCTION

The “who rated what” task of the KDD CUP 2007 is based on the competition organized by Netflix. For this task Netflix has provided a historic database of more than one million movie ratings up to December 2005. The goal of this task is to predict whether a user has rated a given movie during 2006. The outcome is binary by predicting whether a user rated the given movie or not.

To solve this task we have experimented with two kinds of approaches namely (1) clustering based method and (2) decision tree. In clustering based methods we restricted the search space of each user and made a local decision for that particular user based on the neighborhood. We did this by extracting

certain features from the dataset such that the user-user similarity neighborhood was meaningful for the given task. In decision tree based mechanism we derived global features, which can potentially build a meaningful classifier.

The paper is organized as follows: first we provide the details about the related work done in this field. After this we describe about our extension to this related work. Then we describe about the training and test data set. We then state our method and their experimental results.

## 2. RELATED WORK

In this section we briefly present some of the research literature related to collaborative filtering, recommendation systems, data mining and personalization.

The classical predictive modeling approach [1] describes sampling algorithm to estimate a baseline prediction. This paper tries to replicate the procedure to create the scoring data in order to produce a training dataset which similar characteristics. It also describes three types of variables from the training data set namely (1) Movie variables (2) User variables (3) User-Movie interactions. The user related variables focus on the historic rating behavior of each user. The movie related variables focus on the historic rated behavior of each movie. The user-movie pair variables focus on the common historic behaviors of users and movies.

Paper [2] describes collaborative filtering based recommendation algorithms. Clustering is used to identify groups of user who appear to have similar preferences. Once the clusters are created, averaging the opinions of the other users in that cluster can make predictions for an individual. Some clustering techniques represent each user with partial participation in several clusters. The prediction is

then an average across the clusters, weighted by the degree of participation. Clustering techniques usually produce less-personal recommendations than other methods, and in some cases, the clusters have worse accuracy than nearest neighbor algorithm [3]. Once the clustering is complete, however, performance can be very good, since the size of the group that must be analyzed is much smaller. Clustering techniques can also be applied as a “first step” for shrinking the candidate set in a nearest neighbor algorithm or for disturbing nearest neighbor computation across several recommender engines.

Our work explores extensions to the nearest neighbor recommendation algorithms and decision tree algorithm.

### 3. What’s new

In nearest neighbor algorithm while looking for similar users, we have taken into account the user similarity across all the genres and not just one genre. After the users are grouped together in collaborative filtering we have developed methods (1) Weighted Sum (2) nth order Markov Chain for classification. In decision tree based classifier we used sampling algorithm to prune the data set in generating the negative examples for the supervised learning.

## 4. EXPERIMENTAL DATA

We are currently performing experiments on the movie rating data provided by Netflix (<http://www.netflixprize.com>). The dataset contains one million ratings from 480,188 users and 17770 movies. The ratings in the Netflix data was explicitly entered by users, and are integers ranging from 1 to 5.

For each movie we are provided a list of ratings where each rating consists of the user-id of the user who gave the rating, the rating, which is a number from 1 to 5 and the date when the user provided this rating. We are also provided some details about each movie like the movie-id, the date when the movie was released and the title of the respective movie.

To model the user similarity, we augmented the given dataset with genre information for each movie. This information is extracted from the Internet Movies Database (IMDB) using a screen scraping script. The genre information for 1923 movies is not available on IMDB. Hence the dataset was cleansed of these 1923 movies along with the ratings provided for these movies.

Since the original test set was biased we modified the test to contain an equal proportion of positives and negatives.

## 5. EXPERIMENTAL PROCEDURE

### 5.1 Decision Tree based classifier

#### 5.1.1 Sampling Algorithm

For any classification algorithm to learn a meaningful classifier, requires examples from all the classes that it has to learn. The dataset provided contains only positive examples (only user-movie pairs which were actually rated in 2006). We used the sampling algorithm [1] to generate negative instances. The sampling algorithm consists of three primary steps:

1. Divide the dataset into five groups with each group corresponding to one year’s ratings.
2. Draw with replacement, user-ids and movie-ids proportionate to frequency of ratings provided and ratings received respectively.
3. From the user-ids and movie-ids drawn above, randomly pair user-ids with movie-ids and throw away any duplicate pairs.
4. If this pair exists in the original dataset for that year, we mark it as a positive instance otherwise as a negative instance. With this we ensure that the negative examples are drawn with the same distribution as the positive examples.
5. Merge the instances generated above with historic data, updating historic instances only if the recent instance is a positive instance.

The assumption here is that all user-movie pairs are drawn with some unknown probability distribution.

#### 5.1.2 Feature Extraction

The following types of features [1] were extracted:

1. User features
2. Movie features

The extracted user features are:

- 1) Number of historic user ratings.
- 2) Number of months since the first rating of the user.
- 3) Percentage of 1-star ratings of the user.

- 4) Percentage of 5-star ratings of the user.
- 5) Average rating of the user.
- 6) Standard deviation of the user rating.
- 7) Number of months since the last rating of the user.
- 8) Percentage of the user ratings during last year.
- 9) Percentage of the user ratings during last three months.
- 10) Percentage of the user ratings during last three months over ratings during last year.
- 11) Ratio between the number of ratings by user during last year and number of ratings during the year before.

The extracted movie features are:

- 1) Number of historic ratings received by the movie.
- 2) Number of months since the movie was first rated.
- 3) Percentage of 1-star ratings received by the movie.
- 4) Percentage of 5-star ratings received by the movie.
- 5) Average rating received by the movie.
- 6) Standard deviation of the ratings received by the movie.
- 7) Number of months since the last rating received by the movie.
- 8) Percentage of the ratings received during last year.
- 9) Percentage of the ratings received during last three months.
- 10) Percentage of the ratings received during last three months over ratings received during last year.
- 11) Ratio between the number of ratings received during last year and number of ratings received during the year before.

## 5.2 Collaborative Filtering Algorithms

### 5.2.1 Feature Extraction

Previous work like the GroupLens automated collaborative filtering system [4] has focused on clustering items in the genre space. A user's prediction within a genre is computed using only ratings from other users within that genre. The collaborative filtering approach in [2] on the other hand, partitions items based on user rating data irrespective of the content of the data.

The previous methods [2,4] individually succeed in only partially capturing the semantics of user similarity. Our approach tries to mitigate this problem by capturing the user similarity across genres. In general, a user's interest is never always concentrated in a specific genre. The true profile of a user is evident only when we look at the user's interest across all genres.

For determining the user's affinity towards other users in a particular genre, we need to consider the total number of movies rated by the user in that genre as well as the sum of the ratings by the user in that genre. This gives us the users density for every genre. For example, two users where one user rated five movies with sum of ratings equal to 5 and another user who rated one movie with sum of ratings equal to 5 would be considered dissimilar by our approach.

There are 30 distinct genres in the dataset. For each genre, the user has two variables (1) the total number of movies rated in the genre (2) the sum of ratings provided by user in that genre. Hence the feature vector for every user consists of thirty such pairs. Thus the overall feature vector size is sixty.

### 5.2.2 Nearest Neighbor

Similar users are obtained by querying the neighborhood of a user in the above feature space.

### 5.2.3 Prediction Computation

The most important step in a collaborative filtering system is to generate the output interface in terms of prediction. Once we isolate the set of most similar users based on the similarity measure, the next step is to look into the target users ratings and use a technique to obtain predictions. Here we consider two such techniques.

#### 5.2.3.1 Weighted Sum

Let  $u$  be the query user and  $\pi$  be defined as the user similarity function. Given a movie  $m$ , the prediction

Pred of whether the user  $u$  rated the movie  $m$  in 2006 is given by,

$$\text{Pred}(u,m) = 1, \text{ if } \sum_{i=1 \text{ to } k} R(u_i,m) \pi_i > T$$

where,

These counts are collected from the neighborhood of the user. The assumption is that  $m_1, m_2, m_3$  are independent given  $m_4$ .

$$\text{Pred}(u,m) = 1, \text{ if } P(m|\text{history}_u) > T$$

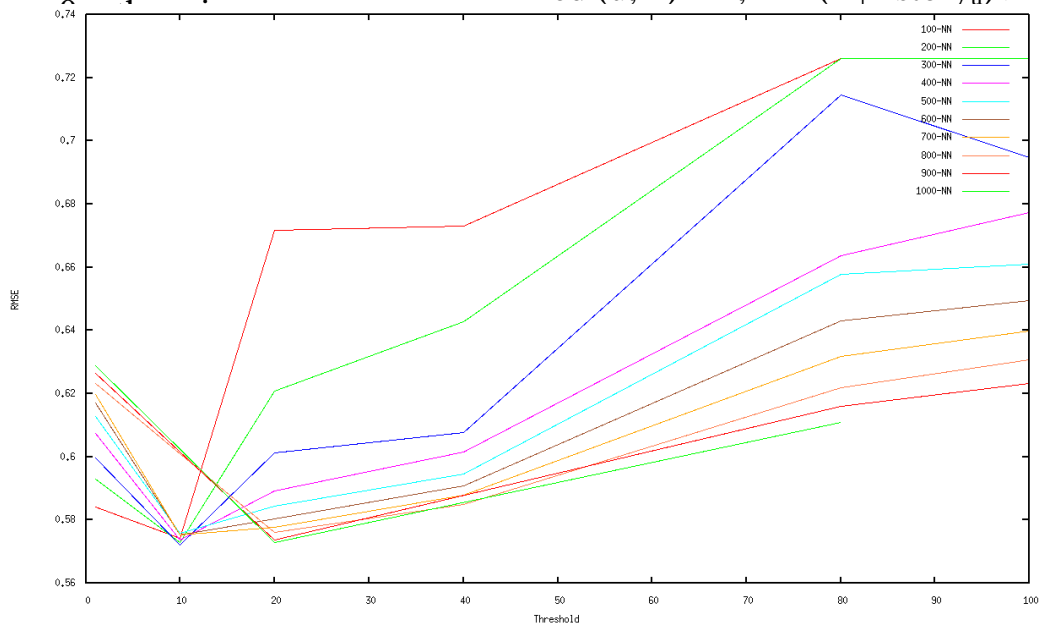


Figure 1: Plot of RMSE for Weighted Sum Nearest Neighborhood for different neighborhoods and thresholds.

$u_i$  is the  $i$ th user in the neighborhood of  $u$

$R(v,m)$  is the function which returns 1 if the user  $v$  has rated the movie  $m$

$k$  is the neighborhood size

$T$  is the threshold

Basically this approach tries to predict on how popular the movie is in the users neighborhood.

### 5.2.3.2 $n^{\text{th}}$ -order Markov Chain

In this approach we model the influence of the recent history, of the movies the user watched, on his future ratings. For example, lets assume that a user watched the following movies in the specific order,

$$m_1, m_2, m_3$$

The probability of the user rating the movie  $m_4$  is given by,

$$P(m_4|m_1, m_2, m_3) = P(m_4) P(m_4|m_3) P(m_4|m_2) P(m_4|m_1)$$

where,

$$P(m_i | m_j) = \text{count}(m_i m_j) / \text{count}(m_j)$$

We defined a variation of the above method in computing the bigram probability. In this variation we define a hypothetical movie  $Y$ . We then replace all occurrences of the movies in the history by this movie variable  $Y$ . Thus in the above example,

$$(m_1, m_2, m_3) = Y$$

Now,

$$P(m_4|Y) = \text{count}(m_4 Y) / \text{count}(Y)$$

Since the dataset is very large, this variation solves the problem of interpreting very small probability values.

## 6. EVALUATION

Statistical accuracy metrics evaluate the accuracy of a system by comparing the numerical recommendation scores against the actual user ratings for the user-item pairs in the test dataset. Root Mean Squared Error (RMSE) between ratings and predictions is a widely used metric.

$$\text{RMSE} = \left[ n^{-1} \sum_{i=1}^n |e_i|^2 \right]^{1/2}$$

The lower the RMSE, the more accurate the recommendation engine is in its prediction.

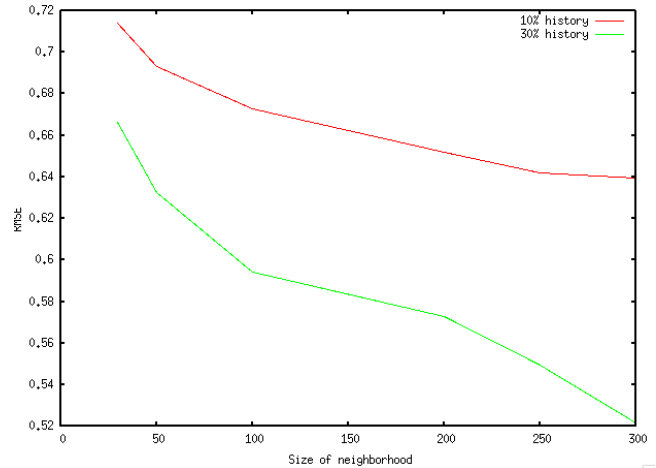
## 7. EXPERIMENTAL RESULTS

### 7.1 Weighted Sum Nearest Neighbor

The Weighted Sum Nearest Neighbor approach is controlled by two parameters:  $k$ , which is the size of the nearest neighborhood and  $T$ , which is a threshold point that determines the movie popularity required in the neighborhood to make a prediction. Figure 1 shows a plot of the RMSE values, obtained for this approach, plotted against the threshold for varying sizes of the neighborhood. Results obtained for Weighted Sum are significantly better than the baseline. Our experiments show that the best prediction can be obtained for a threshold of around 10 and 20, irrespective of the size of the neighborhood. It can be observed that the best prediction accuracy is obtained for 300-NN and a threshold of 10.

### 7.2 nth Order Markov Chain

Figure 2 shows plot of the RMSE against the nearest neighborhood size for two different periods of history. The graph indicates the accuracy increases with the increase in neighborhood size and history duration. This confirms the hypothesis we have drawn saying that the recent movies the user has seen have good impact on his decision for the next movie he is going to rate. Also the neighborhood has a similar impact on the user.



### 7.3 Decision tree

Figure 2 shows the plot of the accuracy of this approach against number of features. Different decision trees were tested with varying number of decision stumps. We observe that once we increase the number of features for the decision tree, the accuracy goes up. The best performance is observed when the number of features is 20 as compared to the entire feature set.

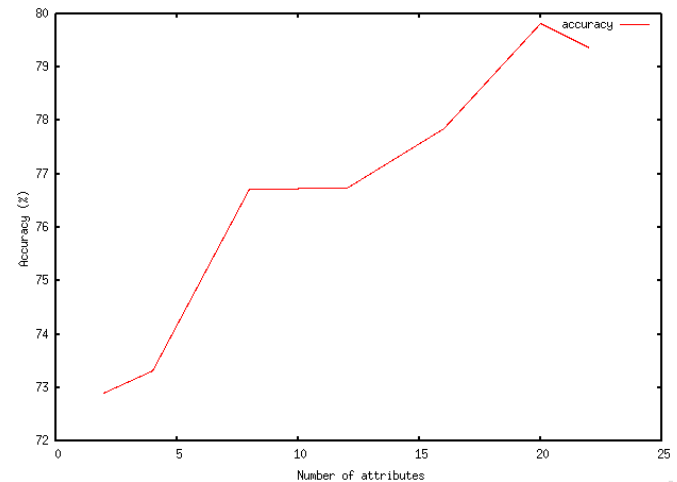


Figure 3: Plot of accuracy for varying number of features used to build the decision tree.

## 8. CONCLUSION

Our different experiments indicate that the statistical approaches give good results as we can see from the results. But our intuition says that there is always a barrier for the accuracy they reach. We think that constructing the behavioral model like encoding the intuition behind the user when he makes a choice for a movie, how does human interests vary across the

genres will make the system more robust. This requires a lot of domain knowledge and interdisciplinary study of this problem. The extension to our work will be to augment the current system with deterministic features as explained above.

## 9. REFERENCES

- [1] Jorge Sueiras, Alfonso Salafranca, Jose Luis Florez. A Classical Predictive modeling approach for task “who rated what?” of the KDD cup 2007
- [2] O’Connor, M., et al. *Clustering items for Collaborative Filtering*. SIGIR-2001
- [3] Sarwar et al. Item Based Collaborative Filtering Recommendation Algorithms.
- [4] Resnick, P., et al. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *In Proceedings of ACM CSCW’94 Conference on Computer-Supported Cooperative Work*, pages 175—186. 1994.

4 - Resnick, P., et al. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *In Proceedings of ACM CSCW’94 Conference on Computer-Supported Cooperative Work*, pages 175—186. 1994.

### 5 - Clustering Items for Collaborative Filtering

Mark O’Connor & Jon Herlocker