Progress Report: Non-negative matrix factorization for Who Rated What

Brian Huey, Renee Rao

Abstract

1 Introduction

We are working the Netflix dataset used in the 2007 KDD cup competition which provides information on characteristics of users of the Netflix video services who rated movies from the years 1998 to 2006. (http://www.kdd.org/kdd-cup-2007-consumer-recommendations)

We will predict which users rated which movies in 2006. To test this we will have a provided set of roughly 100000 (user_id movie_id) pairs where the users and movies are drawn from the Netflix Prize training data set (where none of the pairs were rated in the training set.) Using this list we will try to predict the probability that each pair was rated in 2006 (i.e.the probability that user_id rated movie_id in 2006). It is important to note that the actual rating is irrelevant; we are only interested in whether the movie was rated by that user sometime in 2006.

We are provided with a data set of roughly 200 million ratings for the previous years.

We note this task is very difficult, a trivial method of predicting every movie is not rated gives an root mean squared error between predictions (viewed as "probabilities") and ratings (which are 0 or 1) of .27, where the winning score was .246.

Our overall plan is to use non-negative factorization to produce an interesting feature or possibly more than one feature to use with other standard features (from movie databases such as release dates) to address this task.

We feel that an interesting feature is one that is different for rated movies than for unrated movies. We show that our non-negative matrix factorization can produce a feature that appears to do this quite well. It compares quite favorably to the baseline method (employed by all contestants) of predicting a rating based on the independent calculation of a movie's probability of being rated times the probability of a user producing a rating.

We describe the other features that we will use in the final project proposal.

2 The Data

For this progress report, we viewed the data pre-2005 as training data and 2005 as validation data. (We are preserving the 2006 answer set as the test set for use in a final evaluation.)

We computed using mapreduce counts for number of ratings per user and per movie for use in the baseline algorithm. We also are computing these counts for the sample. The sample we used in this preliminary study is of size 100K ratings.

We note further that the test set for 2006 was sampled proportionally to counts for user ratings and movie ratings made in 2006. So a substantial part of the challenge is to predict this.

Our focus, however, is to concentrate on predicting the feature which captures the variance in distribution for particular users and movies. That is, user "Bob" rates lots of movies but never rates Zombie movies.

3 Baseline Prediction

The baseline prediction requires computes the number of ratings that each user makes in the training set, and the number of movies that

4 Non-negative matrix factorization (NNMF) and our task

To use non-negative matrix factorization to develop user features and movie features to then predict ratings for the Netflix data set. To do this we estimate A, the matrix of movie and user ids, most of which is considered unknown, by decomposing it into a user matrix, V and a movie matrix, U based on K latent factors, such that $A \approx UxV = \overline{A}$. Under this model each row of matrix U is considered a movie factor and each column of the matrix V is considered a "user factor". A prediction for a user-movie pair would mean computing the dot product of the user factor vector and the movie factor vector.

4.1 Method: NNMF Algorithm.

Input: n by m matrix A, integer k Output: n by k matrix U, k by m matrix V with nonnegative entries.

Algorithm Outline.

Initialization: Form initial matrix U (V) by choosing a random subset of the columns (rows) of A and averaging them, K times. We tune this so each element is expected to be added to some U, one time. This is a parameter that can be changed.

Main Loop:

- **Gradient Descent:** We used gradient descent on the Mean Squared cost function for the difference between A and $\bar{A} = U \times V$. We compute the rmse in each step
  - summing MSE over non-zeros in A.
  - and then summing over random pairs of movie-users to ensure that $\bar{A}$ does not converge to all ones.

  It would be prohibitive to sum over all non-zeros.

- **Nonnegativity:** We enforce positivity (which is certainly non-negative) on the weights in the matrix factor by moving all weights away from zero if they get too close to zero, and making negative weights positive if the gradient pushes them to negative.

- **Spread:** We also normalize the factors (using a Gram-Schmidt type procedure to make sure all the factors don't simply repeat). We again enforce positivity here by staying away from zero.

5 Evalution of NNMF as Model.

We intend to use the NNMF as a feature for input into a learning algorithm but, in fact, we can directly measure its effectiveness as a model.

The RMSE for prediction with 40 iterations on the training set is not great, around .379 and on the test set around .412. As noted in [?] this is to be expected. One must first tune to get the correct background probabilities to score well in absolute terms. For example, the winners worked very hard just to get the "background" probability on number of ratings for the specific distribution of the chosen movie-user pairs in the test set.

Thus, we will just try to measure whether the NNMF feature gives any information on whether a user rated a movie or not. We discuss this in the next section.

6 Evaluation of Features

We view a feature as interesting if the feature behaves substantially differently on the user-movie pairs that have been rated versus those which have not. Where the user-movie pairs have been chosen according to the test set sample rules essentially according to the baseline distribution of movie-user pairs proportional to the independent counts.

In figure **??**, we see that we do indeed get substantial differentiation for the Non-negative matrix user-movie feature. We note the baseline prediction does as well, as can be seen in **??**, but the baseline prediction has less separation and does not have separate peaks.

This fact is really encouraging, though tuning its use will certainly require working carefully with the machine learning algorithm that we will ultimately use to make predictions.

7 Other setup tasks and efforts.

We have pulled and processed data from Rotten Tomatoes which we will use in our final model.

We computed a different baseline method from one of the papers, but still need to further tune it to see if it is useful. We also have implemented a different version of baseline which shifts the predictions by user and movie which has the same intuitive notion of independence but has somewhat different behavior. We have not looked deeply into these other methods.

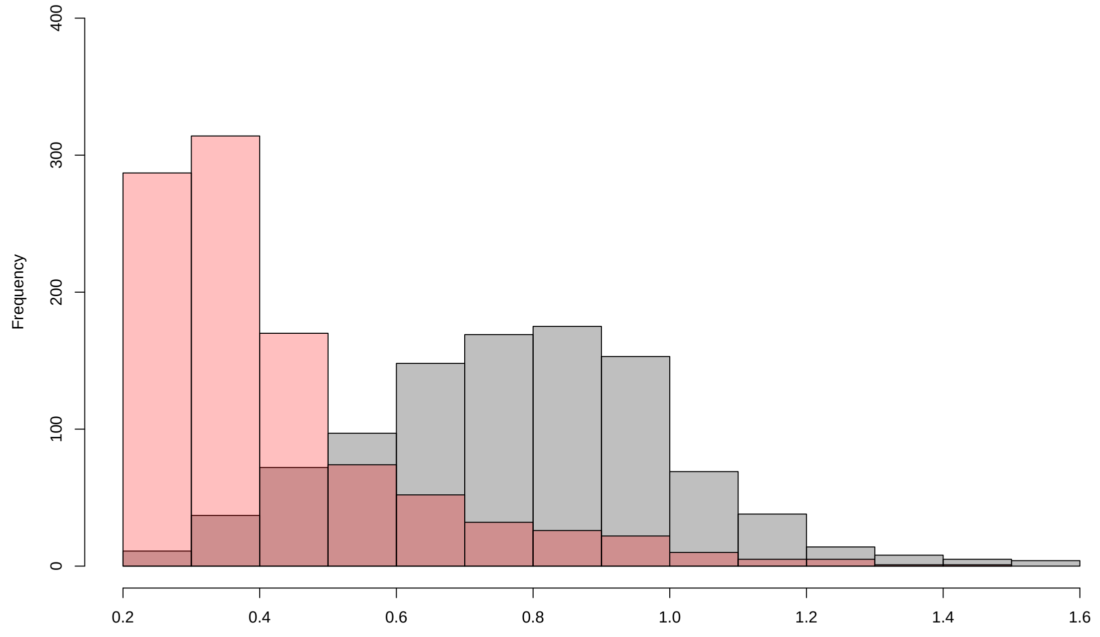

Figure 1: Non-negative matrix feature distributions are separate for ratings and non-ratings. The rightmost peak are the ratings, and the leftmost are nonratings.

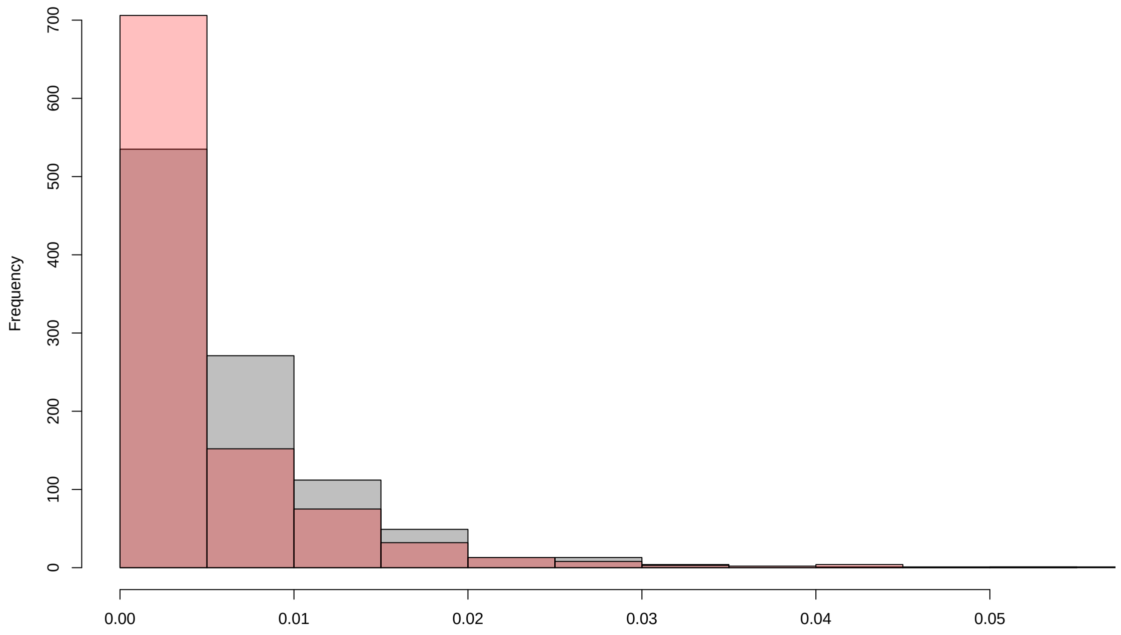

Figure 2: Baseline feature highly overlaps for ratings and non-ratings. The higher distribution is for nonratings, the flatter one is for ratings, so the average for ratings is higher.

8 Division of Labor

Brian

1. Join movie_titles.txt to Rotten Tomatoes info
2. Set up and organized github
3. Upload of data to sc3
4. Calculate the average number of movies rated over all users in the training set (mapreduce).
5. Calculate the average number of ratings over all movies in the training set (mapreduce).
6. Calculate the average number of movies rated for each user in the training set (mapreduce).
7. Calculate the average number of ratings for each movie in the training set (mapreduce).
8. Researched K Nearest Neighbor algorithms.
9. Alternate Baseline method.

Renee:

1. Subset the small data into (train) 100K of pre-2005 and 100K (validation) post-2005 data.
2. Baseline Estimation
3. Wrote Non-Negative Matrix Factorization Model in Python.
4. Evaluated feature worthiness of NNMF model.
5. Generated associated figures for evaluation.
6. Wrote up progress report and Final Proposal.