Final Report: Netflix!

Brian Huey, Renee Rao

December 5th 2014

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 need to predict which users rated which movies in 2006. To test this we use a provided set of roughly 100 000 (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 is in the 2006 set of ratings). 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. Our success at this task will be computed by looking at the root mean squared error (RMSE) between our predictions for each pairs and actual values for the pairs (which are 0 or 1 if the pair is unrated or rated respectively).

We are provided with a training data set of roughly 100.5 million ratings for the previous years of the format user_id movie_id date of rating.

We note this task is very difficult, as a trivial method of predicting every movie is not rated gives an RMSE of 0.27. The KDD cup winner had an RMSE score of 0.256.

We feel that an interesting feature is one that is different for rated movies than for unrated movies, so to assess feature utility heuristically, we graphed the distribution of each feature on the our validation set, and looked for features that produced a differential distribution for rated pairs versus unrated pairs. These graphs can be viewed below and reproduced on github. Using this metric we selected three features to input to our model.

1. Non-Negative Matrix Factorization (NMF)
2. Global Effects of Time on User and Movie
3. Cosine Similarity

We then computed these features for our validation set using AWS MapReduce and an EC2 instance, and used the distribution of these features to create a bayesian transformation used to normalize the features values on the test set, before feeding them into a logistic regression model to make predictions and compute RMSE.

(PUT BEAUTIFUL GRAPHIC BELOW) https://www.sharelatex.com/blog/2013/08/29/tikz-series-pt3.html

Our final result was an RMSE of .2667, which compares favorably with the basic prediction of all zeros but is significantly less than the winner of the 2007 competition.

A detailed description of our data, features, transformation, model processing, and data selection follows.

2 The Data

In this project we viewed the data pre-2005 as training data and 2005 as validation data and preserved the 2006 answer set as the test set for use in a final evaluation. This was done to avoid making predictions based on models or features that were calculated from data observed further in to the future than the validation data. We noted that the test set for 2006 was sampled proportionally to counts for user ratings and movie ratings made in 2006, so we created our 2005 validation sets accordingly, splitting them into:

- random subset of rated user,movie pairs in 2005
- popularity distributed random subset of unrated user, movie pairs from 2005

Doing so allowed us to better analyze feature utility by visualizing the difference of features between the two sets as well as to compute the probability the was rated for given feature values used in the transformation step for the final 2006 test set.

2.0.1 Data Locations

- Pre-2005 training set: s3://stat157-uq85def/shared/netflix/pre2005_training_join/
- Post-2005 validation set: s3://stat157-uq85def/shared/netflix/post2005_training.txt
- Training set: s3://stat157-uq85def/shared/netflix/training_set/training_set_reshape.txt
- Test set: s3://stat157-uq85def/shared/netflix/test_sets/
- Answer set: s3://stat157-uq85def/shared/netflix/answer_sets/

3 Features

3.1 Non-negative matrix factorization (NNMF)

We 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.

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

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.

The distribution of the NMF feature on validation ratings and non-ratings is below:

3.2 Global Effects

The global effect feature is an adjustment to the overall probability that a user/movie pair is rated in 2006, assuming no information (i.e. number of ratings in the training set divided by the total number of user/movie pair combinations), which is approximately 1.17%. Beginning at the first step, we started with the overall probability and subtracted the first effect from it, creating a residual which was the input to the next step, repeating until all effects have been accounted for. This process is based on work described by Bell and Koren.[1]

$$global\_effect_{mu} = overall\_avg - avg_m - avg_u - time_m - time_u$$

Where u and m are user and movie, respectively.

3.2.1 User/Movie Averages

The user and movie averages were calculated for each unique user and movie id by taking the total number of observed ratings for that id and dividing it by the total number of possible ratings. For a given movie id, sum the total number of users rating that movie divided by the total number of users:

$$avg_m = \frac{\eta_m}{\sum_u x_u}$$

Where η_m is the number of ratings that movie m received and x is identically 1.

[1] Bell, R., Koren, Y. ?Improved Neighborhood-based Collaborative Filtering? KDD Cup ?07, August 12, 2007, San Jose, CA.

3.2.2 Time Effects

The time effect was estimated from the date timestamp for each row of data. For a given movie or user, we measure the time in days since the first rating and take the square root of the number of days. The coefficient, $\hat{\theta}$ was estimated by the following equation provided by Bell and Koren:

$$\hat{\theta}_m = \frac{\sum_u r_{mu} x_{mu}}{\sum_u x_{mu}^2}$$

Where r_{ui} is the residual probability and x_{mu} is equal to the square root of the number of days since the movie was first rated. The final estimate of θ is:

$$\theta_m = \frac{\eta_m \hat{\theta}_m}{\eta_m + \alpha}$$

Where α is a tuned parameter.

3.2.3 Feature locations

- User averages: s3://stat157-uq85def/home/sbhuey/outputs/user_averages.txt
- Movie averages: s3://stat157-uq85def/home/sbhuey/outputs/movie_averages.txt
- User coefficients: s3://stat157-uq85def/home/sbhuey/outputs/user_coefficients.txt
- Movie coefficients: s3://stat157-uq85def/home/sbhuey/outputs/movie_coefficients.txt

3.3 Cosine Similarity

For these feature we view each movie as a vector with an entry for each user. The entry is a 1 if the movie was rated by the user and it is a zero otherwise. With this view we can compute the cosine of the angle of the two movie vectors by taking their dot product and dividing it by the magnitudes of the two vectors. Initiatively movies with a high cosine similarity have a high fraction of users who have rated both movies. In this feature for a user-movie pair, (U,M), we find all the movies rated by U and find their average value of the cosine similarity with M. The high this value the more likely we would expect this movie to be rated by this user, since many users rated this movie and other movies that this user also rated.

4 Transformation and Logistic Regression

4.1 Transformation

Upon computing features for the test set we normalized the values of each feature. The natural way to do this was to use Bayes rule to map the arbitrary values of our features to probabilities of rating based on those values. After using the pre-2004 training set to compute features for our 2005 validation sets (described in the data section) we used the distributions of the feature values on the rated validation set and the unrated validation set to compute the probability that a movie-user pair with a given feature value using Bayes rule, that is to divide the number of pairs with that feature value in the rated validation set by the total number of pairs with that feature value in both validation sets (rated and unrated). We, of course, must bucket our features values to do this computation. We then use this validation set mapping to transform features computed for the 2006 test using the full training set into numbers which could be viewed as 'probabilities'. We originally used these probabilities for a naive Bayes model, but as some of our features appeared to be correlated we decided to pass these normalized values into a logistic regression model instead. This allows logistic regression to be fed features that are between 0 and 1 and thus may avoid the danger of its predictions being influenced too much by any single variation in a feature.

4.2 Logistic Regression

We used logistic regression to make a model using data from rating and unrated 2005 validation set pairs and the transformed features we discussed above. The 'training set' for the model was balance among rated and unrated pairs. The true test set, however is imbalance; the rated pairs only comprise roughly 7 percent of the total. Thus we scale the output of the logistic regression model so that the average predicted probability is around 7 percent.

5 Results and Discussion

5.1 Results

Our results using the logistic regression model and combinations of our features are represented below, they should be compared to the 2007 KDD cup winner's RMSE of .256. (Note: The baseline RMSE is computed by predicting .07 for all pairs)

Baseline	Global	NMF	Cosine	RMSE
✓				.2684
✓	✓			.2681
✓		✓		.2682
✓			✓	.2682
✓	✓	✓	✓	.2667

5.2 Discussion

While our model could not compete with the first place winner it did place us between the 3rd and 4th place groups in the competition, so ultimately our model is learning something. Ideally we would like to further this project by running our features through a greater variety of models, tuning the many parameters we used and adjusted heuristically when computing our features, and continuing to explore other features that demonstrated promise on the validation sets, such as outside information about release dates of movies from websites like Rotten Tomatoes. Ultimately, however, we did not have enough time to flesh the project out as fully as we originally envisioned, as we spent quite a bit of time computing our current features and testing them on our validation sets. Creating the validation set correctly was particularly key to getting good measure of the utility of our features, and ultimately cost us some time.

6 Tools

- sklean.liblinear.LogisticRegresion from the scikit package.
- A variety of python libraries including numpy, and scipy.spare.
- unix tools such as grep
- python, R
- AWS EC2, S3, MapReduce/Hadoop

7 Division of Labor

Note: Authorship, comments and readme's for all code can be found on the github website for the project. **Brian**

1. Join movie_titles.txt to Rotten Tomatoes info
2. Set up and organized github
3. Upload of data to sc3
4. Subset training data in to pre and post 2005 (map reduce).
5. Calculate the average number of movies rated over all users in the training set (mapreduce).
6. Calculate the average number of ratings over all movies in the training set (mapreduce).
7. Calculate the average number of movies rated for each user in the training set (mapreduce).
8. Calculate the average number of ratings for each movie in the training set (mapreduce).
9. Alternate Baseline method.

Renee:

1. Created Validation Sets.
2. Wrote predict_with_features.py, some_tools.py, logistic.py to go from features computed on validation and test set to predictions and RMSE
3. Wrote Non-Negative Matrix Factorization Model and Cosine Similarity in Python.
4. Evaluated feature worthiness.
5. Generated associated figures for evaluation.
6. Wrote reports.