Movie Recommendation System

Ray Sherbourne

March 17, 2021

# Project Overview

In 2006, Netflix started an open competion to build a machine learning model that would predict user ratings for movies using only past ratings as predictors. This report will provide one solution to a modified form of this problem, known as **The Netflix Challenge**, using a dataset provided by grouplens.org. The accuracy of the models presented will be measured against a validation set that is pulled from the dataset before any processing. Accuracy will be measured with the root mean squared error (RMSE), targeting a score of 0.8775 or lower. The key steps to be performed are:

1. Get the data
   * Download data from source
   * Combine the data into a single file
   * Extract the validation set
2. Data wrangling to clean the data for ease of use
3. Exploratory analysis
4. Model building
   * Start with the mean as a baseline approach
   * Introduce a movie bias term
   * Introduce a user bias term
   * Introduce a regularization term lambda
   * Use matrix factorization (with and without optimizing parameters)
5. Train with best performing model and test on validation set
6. Reflect on the results

# Method

The data was downloaded as a zip file from the grouplens website. Two files (movies.dat & ratings.dat) were extracted and joined together into a dataframe containing columns for userId, movieId, rating, timestamp, title, and genres. With the data collection complete, 10% was immediately set aside as the validation set.

From the website/readme file, the data is described as just over 10 million ratings from users who rated at least 20 movies each. Users are only represented by their a unique and anonymized id number. Ratings were made on a 5-star scale in increments of 1/2. The timestamp variable represents seconds form midnight UTC of January 1, 1970. The genres variable contains a list of 18 possible genres, separated with pipes. Looking at the first few lines of data, two data munging operations will help with further processing:

* Converting the timestamp to a date time format
* Extracting the release year from the title

Visual summaries are often useful in both conveying and understanding data. A few simple plots will serve as the primary method of data exploration. The following histogram shows the number of users that rate a given amount of movies.

The next histogram shows the number of movies that have a given amount of reviews. There are a significant amount of movies that have been rated less than 10 times. This could potentially be a source of error that may need to be solved with regularization.

The next plot is a line graph showing the frequency of each possible rating. Whole number ratings are much more common.

The next plot shows the mean rating for all movies based on the week the review was given. This shows that time-series analysis may have some predictive power, but it is relatively small.

After splitting the data into a training and test set (Reminder that the validation set remains untouched), the process of building/testing models begins. For this approach, instead of starting with a complex model, a simple model will be built and itteratively improved, while serving as a benchmark to gauge progress.

The first model will simply predict the average movie rating for all cases:

To improve this model, it is possible to account for the fact that movies are not received by audiences equally. A bias term is calculated by taking the average of the difference between the individual ratings of a movie and the average of all movie ratings. This will represent how a particular movie deviates from the average movie, and will be added onto our previous model:

Likewise, different users have different movie critiquing tendencies. A bias term for users, is calculated by taking the average of each users rating after subtracting the average movie rating and movie bias term. The user bias term will account for how a particular movie deviates from the average movie for a particular user. Again this term is added onto the previous model:

The formula for the second bias term included the first bias term, so a quick check is done to see the results of switching the order of calculation:

The best form of the model above will be itterated further by adding a regularization term, lambda, to combat over-fitting. In the context of The Netflix Challenge, over-fitting could occur due to a rarely watched movie being incorrectly labeled as exceptionally good and leading to unhelpful recommendations. Cross-validation is used to search for the optimal lambda term.

This model passes the accuracy goal for this project, however there is an alternate approach using matrix factorization. The idea behind this approach is that if the data is arranged into a matrix of users x movies, with the intersections representing an individual rating, then factorization would results in a matrix of user-preferences and a matrix of movie characteristics. It would be possible to group similar users and group similar movies, and then make recommendations based on these groups. The Recosystem package was built specifically to handle this type of matrix factorization. Recosystem takes three vectors: one for the users, one for the movies, and one for the ratings. This report includes two approaches: one without tuning the model parameters, and one after tuning the parameters. In tuning the parameters, multiple values were experimented with, however this process took a significant amount of computational time, so only these two models are included here.

This method of matrix factorization gives the best result. Even without tuning the parameters, it beats the other models.

# Results

The best performing model is selected to use on the validation set. It is trained with the full edx data set and the rmse of the predictions is calculated below:

# Conclusion

The goal of this report was to build a movie recomendation system using a dataset of past reviews. After importation, the data was cleaned and briefly explored. During the exploration phase, some potential patterns/problems in the data were identified. Several models were built using the insights from the explorations and prior intuition. In the end the goal of an rmse score below 0.8775 was achieved, but in practice the effort to improve the models would continue. One such further improvement could be implementing the genres feature into the model. Some of the limitations of the models presented here are:

* It is unclear how many users may be sharing a single id. Adding a feature on a movie streaming platform to account for multiple users would result in more personalized predictions.
* The data for a users search history is a powerfull predictor that a service like Netflix would have access to.
* The given database does not include features such as actors and directors due to limitations of the authors computer.
* The total amount of time a user spends watching a movie, could have some predictive power for other users. For example: multiple re-watches may have more value than a 5-star rating.

As the data set is constantly growing, and the algorithm is being tested in the wild, it is realistic to check the rmse of the model post-implementation. A data pipeline of importing the newest results and adding them to the existing model would be important. Verifying that the updating rmse is behaving as expected is an easy way of making sure the model has not broken down.