Movielens Project Report

Min Zhou

2/12/2019

Contents

1 Introduction:

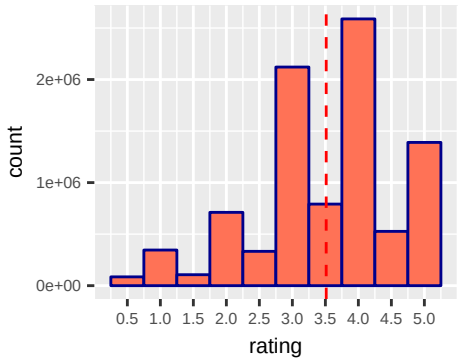
Netflix uses a movie recommendation system to predict if a specific user would like a specific movie. The system uses ratings that each user has given to a set of movies to predict future ratings that the same user or similar users may give to new items. This Harvard Data Science Capstone project is motivated by the Netflix challenge in October, 2006. The company offered a million dollars reward to anyone who can improve their recommendation algorithm by 10%. Since the Netflix rating data is private, the rating data for developing better algorithm is generated by the GroupLens research lab with over 20 million ratings for over 27,000 movies by more than 138,000 users. To lessen the computational burden, this course project will be using the 10M version of the MovieLens dataset and split the data into `edx` and `validation` sets. The `edx` data set will be used to develop a machine learning algorithm to predict the movie ratings in the `validation` set which is around 10% of the 10M dataset. The goal of this course project is to minimize the root mean square error (RMSE) of the predicted ratings for the `validation` set. The general approach used in this report involves **baseline predictors with temporal effects** combined with **matrix factorization** to account for interactions between users and movies.

2 Data Analysis and Feature Selection:

The `edx` and `validation` data sets are created by `the code in the Movielens course project`. The `edx` set contains all the users and movies in the `validation` set with 69878 users and 10677 movies. Below is a quick overview of the edx data set:

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

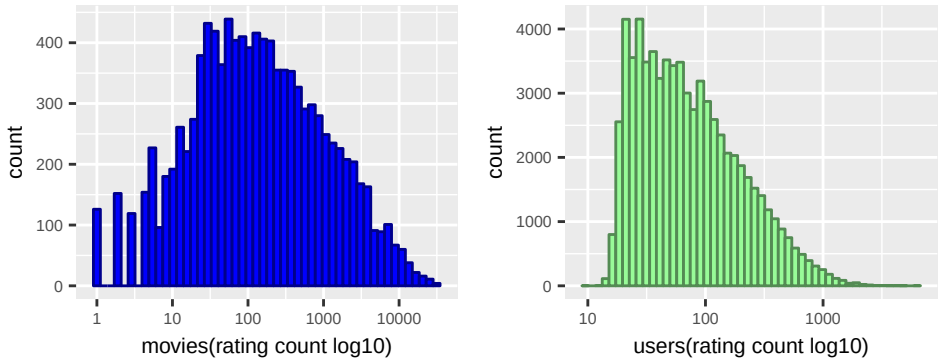
As you can see above, userId and movieId are numbers that represent specific users and movies. Timestamp is the number of seconds since January 1, 1970 (midnight UTC), not counting leap seconds (also known as Unix time). Both title and genres are of variable type character, providing details about each movie. Our label, rating is of variable type numeric with the following distribution:



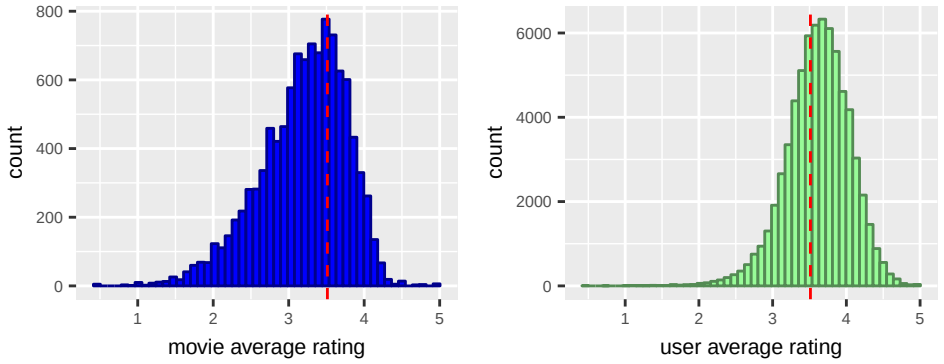
The red dashed line represents the overall rating average across all users and movies. It will be referred to as μ throughout this report. As we can see, the ratings range from 0.5 stars to 5 stars. We are going to take a closer look at userId, movieId, timestamp, and genres to carefully select our features for rating prediction. Titles are redundant with movieIds, so we will remove the title column to save memory.

2.1 Movie and User Effects

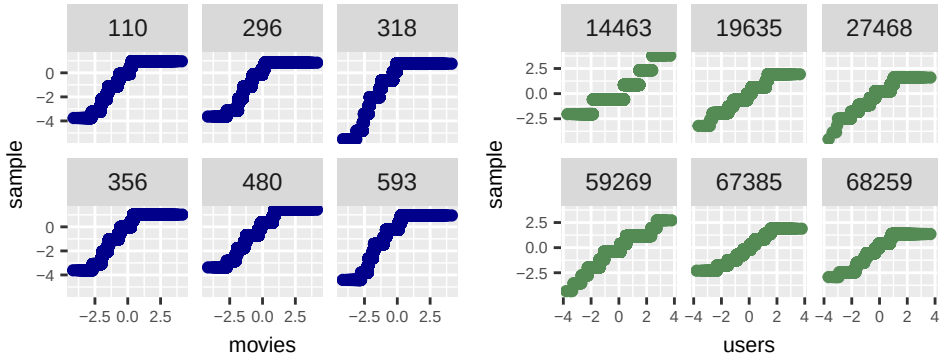
First, we are going to focus on movieIds and userIds. Intuitively, we should see movie only effects and user only effects on ratings. We will call them baseline predictors. For example, if movie A is a Hollywood hit, then naturally it will perform better than average movies. Similarly, if user B is a harsh critic, then he/she will tend to rate lower than other users. Below is rating count distribution for all movieIds and all userIds:



As you can see, some movies are more popular than others and some users are more active than others.
Next, let's take a look at the average rating distribution for all movieIds and userIds:



Again, the red dotted line is μ. Both distributions, especially that of the users, appear to approximate a gaussian distribution. The mode for movies is very similar to μ while the mode for users is slightly higher than μ. Movies distribution also has a longer tail for lower ratings.
Finally, let's select the top 6 most rated movies and top 6 most active users and perform quantile-quantile (Q-Q) normal plots to see if they approximate gaussian distributions:

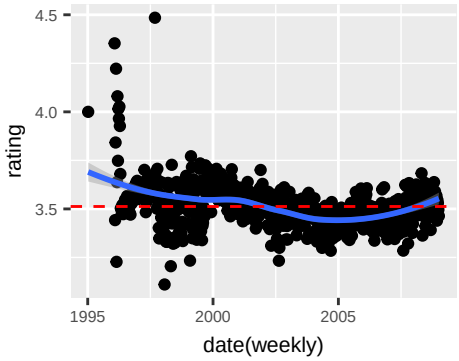


It appears that both user and movie groups approximate normal distribution, which implies ratings and these two features follow multivariate normal distribution. This means we can use linear regression to predict baseline predictors if we set userId and movieId as factors. To save computation time, we will perform this manually in the **method and analysis** section.

2.2 Time and Genre Effects

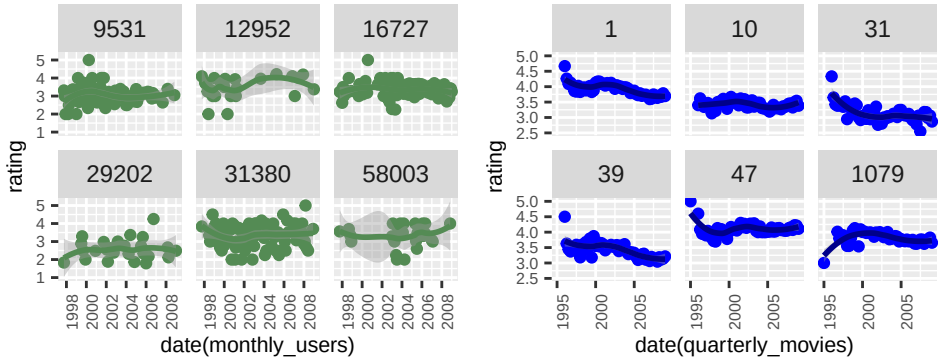
2.2.1 Time Effect

First, let's look at the effect of time on ratings alone:



Once again, the dotted red line is μ. The timestamp is rounded into intervals of weeks and as you can see there are

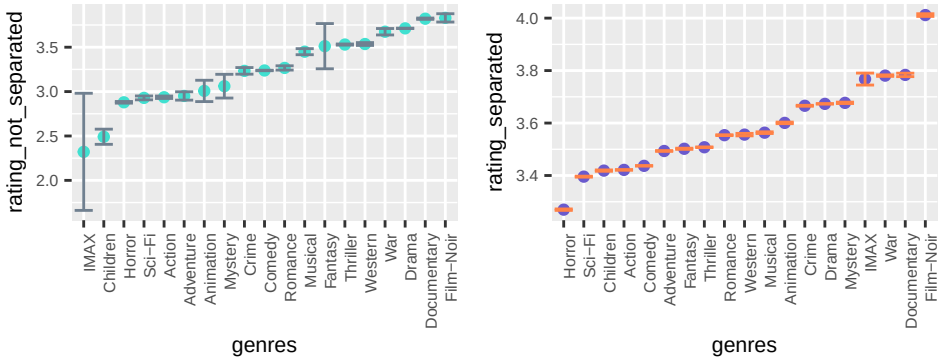
only minor temporal effects on movie ratings. But what about user specific and movie specific temporal effects? Some blockbuster movies may lose popularity as new Hollywood hits come out and some users may become harsher or softer critics over time. Generally speaking, movie temporal bias should change over longer time spans than user temporal bias because behavioral changes tend to be more temperamental. For this reason, we bin timestamp for movies into intervals of quarters (3 months period) and timestamp for users into intervals of one month. Let's select top 6 movies rated over the longest time span and top 6 users with the longest rating history and look at their average rating over the course of time:



It appears that there are some temporal effects for both user and movie biases on ratings so our baseline predictors will include user specific and movie specific temporal effects.

2.2.2 Genre Effect

Now let's look at our last feature, genres. Genres can be an interactive component between users and movies. Some users may have a preference for documentaries and they tend to rate movies with this genre higher than others. That being said, although there are only 19 distinctive (including IMAX) genres, many movies contain more than just one genre and there are total 797 various genre combinations for the `edx` data set. This complicates the interaction between users and movies. Of course, we can separate the genre combinations into each row and look at individual user genre preference. However, one movie genre may contain action, romance, and comedy, not all 3 genres are always good representations of that movie. Separating genres into each row then calculating user preferences can generate false information. To confirm this suspicion, we compare the genre rating distribution for observations containing only one genre to that of all observations with genres separated into each row:



We notice some definite genre bias changes when we separate the genres columns. As there are too many complex components involved in the genres column, we decided to remove the genres column and focus on **matrix factorization** to detect any latent factors for interactions between users and movies.

3 Method and Analysis:

3.1 Model Approach:

After some in depth data analysis, we decided to split our prediction model into 2 components:

1) **Baseline predictors**:

$$B(u,i) = \mu + b_i + b_u + b_i(t) + b_u(t)$$

$B(u,i)$ stands for the baseline predictor for user u and movie i, b_i is constant movie bias, b_u is constant user bias, $b_i(t)$ is temporal movie bias (date in intervals of a quarter) and $b_u(t)$ is temporal user bias (date in intervals of one month).

2) **Matrix Factorization for residuals**:

$$R(u,i) = U_u \bullet V_i$$

$R(u,i)$ stands for the residual for user u and movie i, U_u is the user specific factor vector, V_i is the movie specific factor vector and $\bullet$ stands for vector dot product.

The **Final prediction model** is as follows:

$$Rating(u,i) = B(u,i) + R(u,i)$$

$Rating(u,i)$ stands for the rating for user u and movie i, $B(u,i)$ stands for the baseline predictor for user u and movie i and $R(u,i)$ stands for the residual for user u and movie i.

3.2 Data Preprocessing:

As mentioned in the **Data Analysis and Feature Selection** section, we are going to remove genres and title columns and mutate timestamp column into 2 separate columns:

- $date_q$: date format with intervals of a quarter for $b_i(t)$ calculation.
- $date_m$: date format with intervals of a month for $b_u(t)$ calculation.

These transformations will be performed on both `edx` and `validation` sets.

3.3 Baseline predictor fitting:

We will use regularization for baseline predictors to punish small sample sizes: l_1 for b_i and b_u, l_2 for $b_i(t)$, and l_3 for $b_u(t)$. To optimize our regularization terms based upon minimum RMSE, we will use a *greedy approach* to fit l_1 first, then l_2, and finally l_3. Ideally, a *stochastic gradient descent* method would be more accurate, however, a *greedy approach* works well enough and saves computation time. Each fitting will use *cross validation* over 5 different sets of train(~90%) and test data (~10%) splitting using the `edx` set. The partition will be done in a way that guarantees the train set contains all users and movies in the corresponding test set.

Steps are outlined below:

1. Fit l_1:
    a. Split `edx` set into train and test sets randomly.
    b. Calculate μ_t, b_i, and b_u using the train set and different l_1s.
    c. Add μ_t, b_i, and b_u into the test set and calculate the RMSE.
    d. Repeat 4 more times and calculate average RMSEs for all l_1s and pick the l_1 with the lowest average RMSE.

$$b_i = \left(\frac{\sum rating - \mu_t}{n_i + l_1}\right)$$

rating stands for the label after stratification by movieId, μ_t is the overall rating average across all users and movies for each training set, n_i is the rating count for movieId i and l_1 is the regularization term.

$$b_u = \left(\frac{\sum rating - \mu_t - b_i}{n_u + l_1}\right)$$

rating stands for the label after stratification by userId, μ_t is the overall rating average across all users and movies for each training set, n_u is the rating count for userId u and l_1 is the regularization term.

2. Fit l_2:

   a. Split `edx` set into train and test sets randomly.
   b. Using the train set, calculate μ_t, b_i, and b_u with the best fit l_1 obtained in step 1.
   c. Calculate $b_i(t)$ using the train set, μ_t, b_i, b_u values, and different l_2s.
   d. Add μ_t, b_i, b_u, and $b_i(t)$ into the test set and calculate the RMSE.
   e. Repeat 4 more times and calculate average RMSEs for all l_2s and pick the l_2 with the lowest average RMSE. **It is important to note that in some cases the train sets may not have all of the movie and date combinations in the corresponding test sets.** $b_i(t)$ **will be set to zero in these situations.**

$$b_i(t) = \left(\frac{\sum rating - \mu_t - b_i - b_u}{n_i(t) + l_2} \right)$$

   rating stands for the label after stratification by movieId and $date_q$, μ_t is the overall rating average across all users and movies for each training set, $n_i(t)$ is the rating count for movieId i during the $date_q$,t and l_2 is the regularization term.

3. Fit l_3:

   a. Split `edx` set into train and test set randomly.
   b. Using the train set, calculate μ_t, b_i, b_u, and $b_i(t)$ with the best fit l_1 and l_2 from the previous steps.
   c. Calculate $b_u(t)$ using the train set, μ_t, b_i, b_u, $b_i(t)$ values, and different l_3s.
   d. Add μ_t, b_i, b_u, $b_i(t)$ and $b_u(t)$ into the test set and calculate the RMSE.
   e. Repeat 4 more times and calculate average RMSEs for all l_3s and pick the l_3 with the lowest average RMSE. **Similar to when calculating** $b_i(t)$**, the train sets may not have all of the user and date combinations in the corresponding test sets.** $b_u(t)$ **will be set to zero in those situations as well.**

$$b_u(t) = \left(\frac{\sum rating - \mu_t - b_i - b_u - b_i(t)}{n_u(t) + l_3} \right)$$

   rating stands for the label after stratification by userId and $date_m$, μ_t is the overall rating average across all users and movies for each training set, $n_u(t)$ is the rating count for userId u during the $date_m$,t and l_3 is the regularization term.

4. Calculate μ, b_i, b_u, $b_i(t)$ and $b_u(t)$ using the optimized l_1, l_2, l_3 for the `edx` set, and add the biases to the `validation` set. Zeros will be given to $b_i(t)$ and $b_u(t)$ if `validation` set doesn't have the movie date or user date combinations.

5. Calculate $B(\hat{u}, i)$ for the `validation` set using the baseline biases from step 4.

$$B(\hat{u}, i) = \mu + b_i + b_u + b_i(t) + b_u(t)$$

   $B(\hat{u}, i)$ stands for the predicted baseline predictor for user u and movie i in the `validation` set.

3.4 Matrix Factorization:

We will use *funkSVD* from the *recommenderlab package* to calculate factor vectors for each user and movie. The *recommenderlab package* also contains convenient ways to convert data frames into realRatingMatrix. The detailed steps are below:

1. Calculate the residuals for the `edx` set as shown below, then select the first column as userId, the second column as movieId and the third column as residuals.

$$R(u, i) = Rating(u, i) - B(u, i)$$

2. Convert the 3 column data frame into a realRatingMatrix with rows as users, columns as movies, and residuals as "ratings".

3. The realRatingMatrix is then converted to a matrix format because *funkSVD* can only be performed on matrices.
4. Run *funkSVD* on the matrix produced in step 3. The *funkSVD* function returns two matrices: `matrix U`, representing the user factor matrix; and `matrix V`, representing the movie factor matrix.
5. Predict the $R(\hat{u}, i)$ for the `validation` set by taking the dot product of the factor vector for user u from `matrix U` with the movie i factor vector from `matrix V`.

$$R(\hat{u}, i) = U_u \bullet V_i$$

$R(\hat{u}, i)$ stands for the predicted residual for user u and movie i in the `validation` set.

3.5 Final Prediction:

The predicted rating for the `validation` set will be the sum of $B(\hat{u}, i)$ and $R(\hat{u}, i)$:

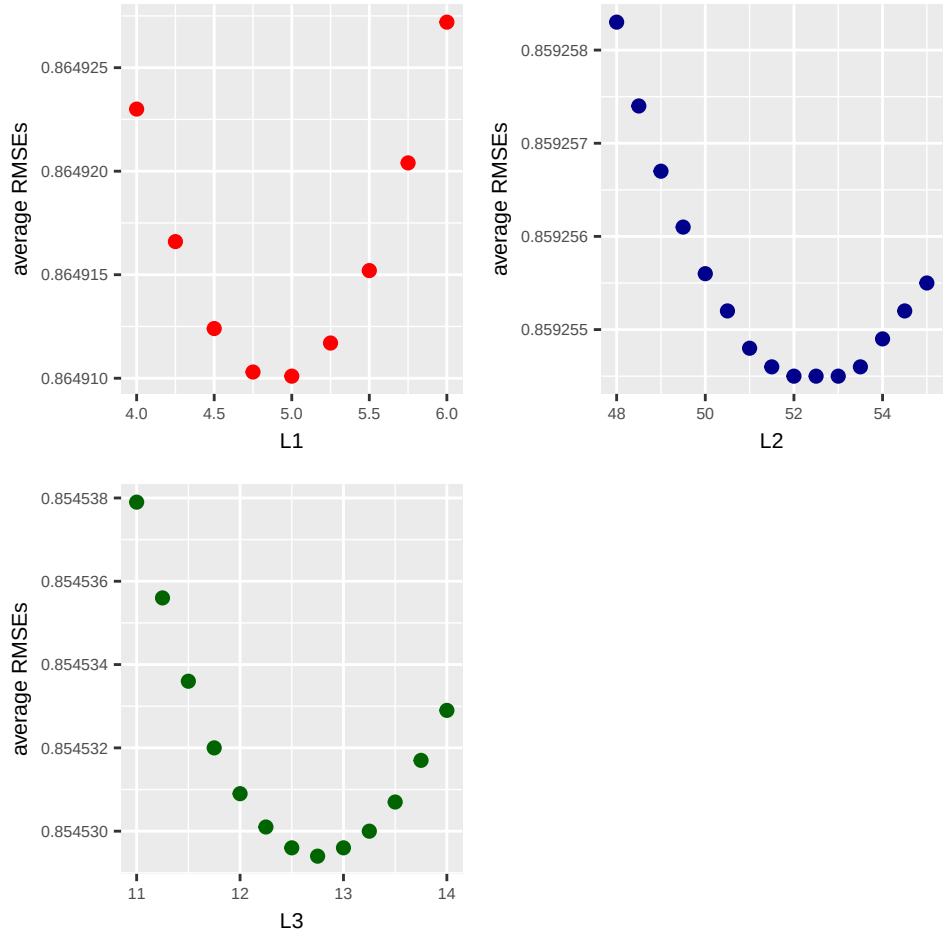
$$\hat{Rating}(u, i) = B(\hat{u}, i) + R(\hat{u}, i)$$

$\hat{Rating}(u, i)$ stands for the predicted rating for user u and movie i in the `validation` set.

4 Results:

4.1 Baseline predictors:

Below are results of baseline predictor model fittings for b_i, b_u, $b_i(t)$ and $b_u(t)$. The x axes are L_1, L_2 and L_3 values; and the y axes are the corresponding average RMSEs:



4.2 RMSE Comparisons:

Here is a table comparing diffrent RMSEs after each sub model fitting. **Note that the first 3 RMSEs are the lowest average RMSEs after fitting l_1, l_2 and l_3 using only the `edx` random test data sets.** Adding **temporal effects** reduces the average RMSE for the `edx` random test data sets by over 0.01 and **Matrix Factorization** reduces the `validation` RMSE by over 0.06.

models	**RMSE**
constant movie user biases	0.8649101
movie temporal bias added	0.8592545
movie user temporal biases added	0.8545294
validation baseline predictor	0.8541734
validation residual added	0.7943898

5 Conclusions:

After using both **baseline predictors** and **matrix factorization**, we are able to achieve an RMSE of 0.794, well below the requirement for this course project. That being said, the prediction is off by almost one star. More improvements can be made by including *frequency* as part of the **baseline predictors** for user and movie biases, performing **matrix factorization** with **temporal dynamics**, and using **Restricted Boltzmann Machines** for inactive users and unpopular movies. **Gradient Boosted Decision Trees** could also provide improvement by acting as a *blending scheme* with extra features such as *number of ratings for each movie*, *number of movies rated by each user*, *days since the movies first rated* and *days since the user first rated a movie*. Recommendation systems are a very complex machine learning problem with many unpredictable user and movie interactions. That being said, with an RMSE of 0.794 on ratings, Netflix should be able to predict if a specific user would love or hate a specific movie.

6 Appendix:

Helper Functions: To save memory and to speed up computation, 9 different helper functions are created for this project along with removing unnecessary variables followed by garbage collections:

- Helper function 1: performs the data preprocess described in the **Data Preprocessing** section.
- Helper function 2: `edx` data partition function to allow cross validation for all three regularization terms.
- Helper function 3 to 5: optimization functions for the three regularization terms. Each function returns a numeric vector of average RMSEs.
- Helper function 6: calculates the predicted baseline biases for user u and movie i in the `validation` set, and returns `edx` as a 3 column data frame consisting of userId, movieId and residuals for **matrix factorization**, `validation` data frame with only userId, movieId and predicted baseline biases for final rating prediction and the RMSE value without **matrix factorization**.
- Helper function 7: converts `edx` data frame from `helper function 6` into matrix, performs *funkSVD*, and returns movie and user factor matrices.
- Helper function 8: find correct user factor vector and correct movie factor vector by matching userId and movieId and perform vector dot product to calculate predicted residuals for user u and movie i. **`Helper function 8` is inside the `helper function 9` in the `R script`.**
- Helper function 9: uses `helper function 8` to calculate predicted residuals for user u and movie i in the `validation` set, calculates the final predicted ratings, and returns the predicted ratings along with the final RMSE.