Movie Recommendation System

Ryan Reuther

11/7/2019

Introduction

As Netflix continues to experience increased competition in the streaming industry, the need to provide an improved viewer experience is of great importance. While producing original content, rotating movies in and out of their service, and adding blockbuster movies to their lineup are great ways to improve revenue and member retention, exposing viewers to existing content through movie recommendations is of great importance. While many movies rely on word of mouth and reviews to spread word of their movies, Netflix has scaled and personalized the process by analyzing data to predict a user's movie preferences. The goal of this algorithm is to develop a recommendation system like that of Netflix.

The recommendation system was trained off of the "MovieLens" dataset, which contains more than 10 million records. Key steps in the development of the algorithm were loading the data, separating it into two sets for training and testing the algorithm, exploring information within the data and developing the algorithm, and finally evaluating the algorithm by calculating the Root Mean Square error. If the algorithm achieved a predictive accuracy rating of less than 0.86499, then the algorithm was considered a success. Ultimately, an algorithm which accounted for the individual effect the specific movie, user, and genre had on ratings was enhanced and optimized by regularization and cross validation to provide a RMSE value below the target value.

Method

The entire MovieLens dataset contains several million ratings from various users, so a subset of the entire Movielens dataset, which is included in the "dslabs" R package, was used. This subset contains 10,000,054 rows of data, and was used to work around the associated performance issues with running calculations on millions of rows of data.

From the subset, two datasets were created: one for training the algorithm (called edx) and one for testing the algorithm (called validation). To reduce the MovieLens dataset into two subsequent sets, the CreateDataPartition function was used to gather indices from the Movielens dataset. The parameter p, which denotes the percent of data that goes into the validation set, was set to 0.10.

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = .1, list = F
ALSE)
```

Once the indices were collected, the Movielens data was split into the two datasets. The validation set was further altered to remove entries the movieID's and userID's that were not contained in the edx set.

The first, and most rudimentary, model was built by analyzing the training dataset without stratification. All movie ratings were equal to the average movie rating of the entire dataset, with all differences in ratings being described by random variation (noise), ε. The model is shown below:

Equation 1:

$$Y_{u,i} = \mu + \varepsilon_{u,i}$$

Where:

$$\mu = average\ movie\ rating$$

$$Y_{u,i} = rating\ of\ movie\ i\ from\ user\ u$$

$$\varepsilon_{u,i} = independent\ errors\ sampled\ centered\ at\ 0$$

This model did not account for deviations of ratings based on any predictions, so every prediction was set to the average movie rating for the entire dataset. The predictive quality of the model was evaluated by calculating the Root Mean Square Error (RMSE) based off the validation set's actual ratings:

```
RMSE <- function(y, y_hat){
  sqrt(mean((y - y_hat)^2))
}
```

Equation 2:

$$RMSE = \sqrt{(\sum_{i=1}^{n} (\hat{y_i} - y_i)^2 / n}$$

Where:

$$\hat{y}_i = predicted\ rating\ for\ movie\ i$$

$$y_i = actual\ rating\ for\ movie\ i$$

$$n = number\ of\ ratings\ for\ movie\ i$$

The RMSE is the standard deviation of residuals, where residuals are a measurement of how far from a regression line a predicted data point is.

Equation 3:

$$Residual = (\hat{y}_{i,} - y_i)$$

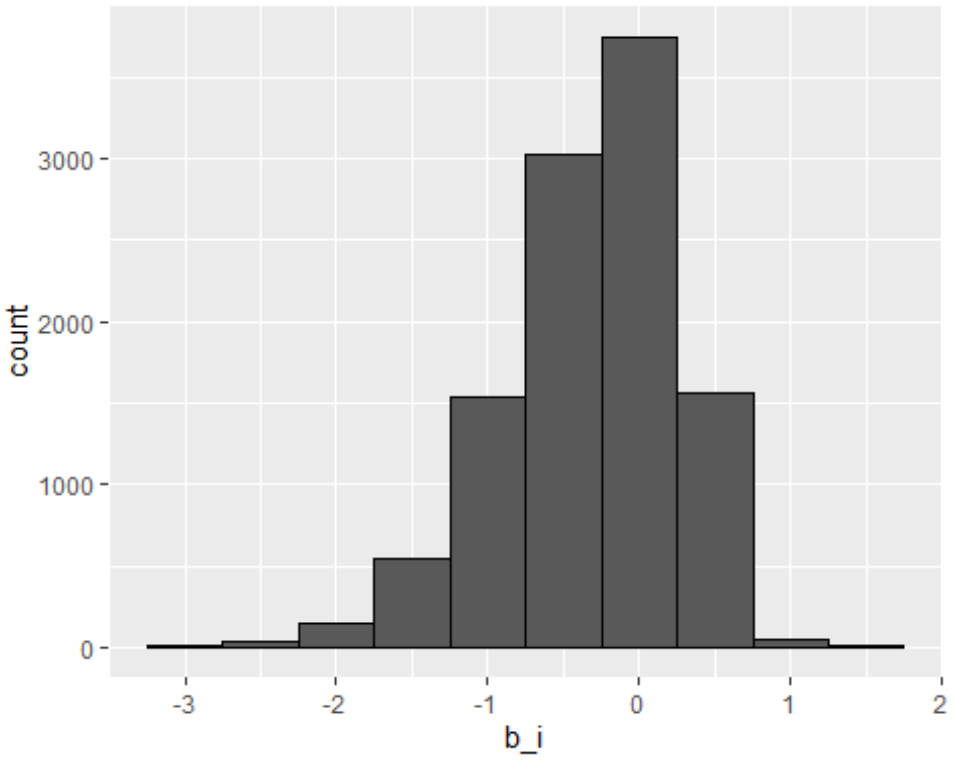
Therefore, to improve the accuracy of a movie recommendation system, the RMSE must be minimized.

When observing the entire edx dataset as one group, the average score for all movies was proven to provide the best prediction. This was proven by calculating the RMSE between the average rating of all movies, which was 3.5125, and by comparing it to arbitrary prediction values. In Table 1, the RMSE of the average of all movies was compared to that of an arbitrary rating value, 2.5.

Table 1:

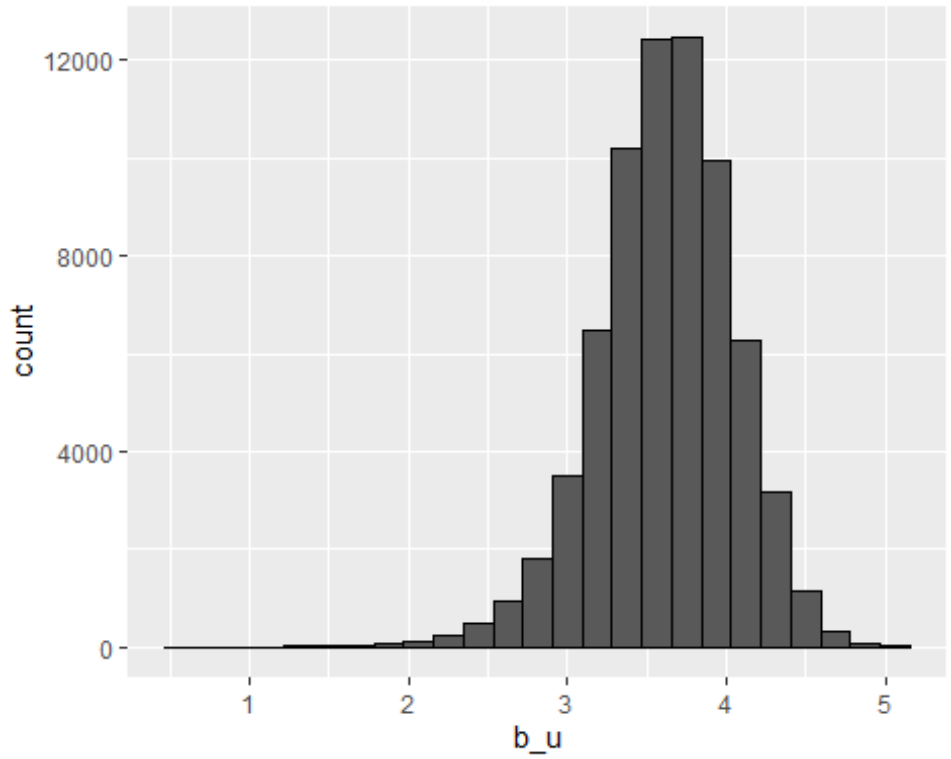
method	RMSE
Just Average	1.061135
Arbitrary Value	1.466388

To build a more accurate model, the ggplot2 package was used to visually examine movie ratings with different groupings. This package, when used with R dataframes, allows for the manipulation of the data to extract meaningful information about the dataset. In the case of this study, graphs examining the distribution of movie ratings by movieID, userID, or genre provided valuable information which was used to enhance the model. Graph 1 illustrates the deviations based on movieID:



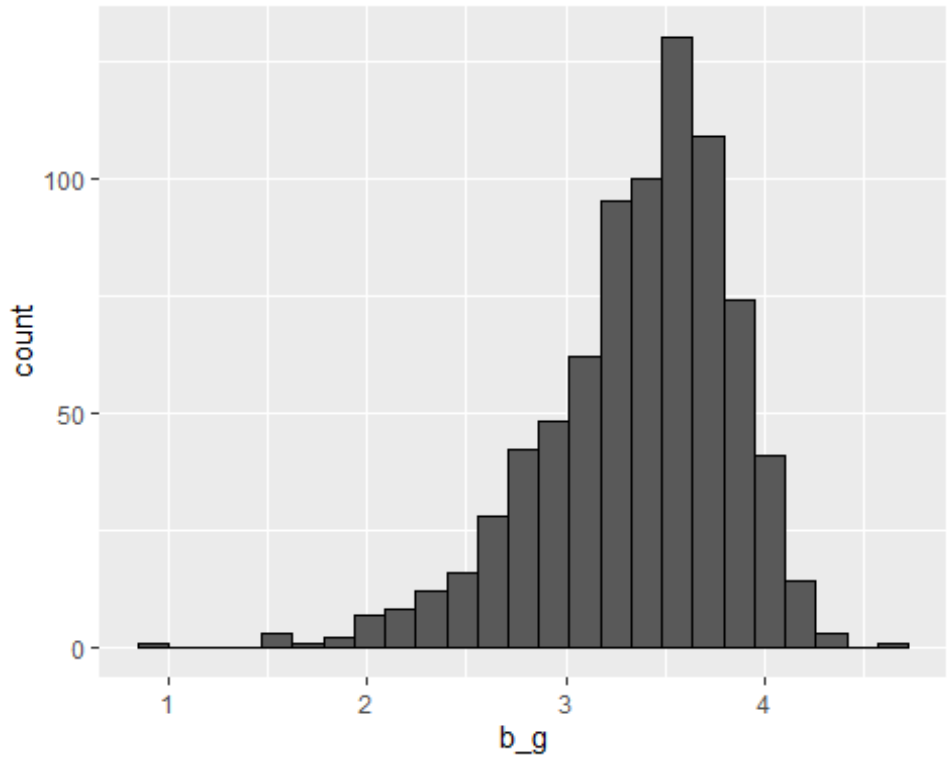
Graph 1: Histogram of average movie ratings by movieID. Certain movies have higher average ratings than others.

Furthermore, by exploring the relationship between users and their average ratings, there was determined to be a distribution in the average ratings per user. As Graph 2 indicates, accounting for individual user behavior would improve the accuracy of the algorithm.



Graph 2: Histogram of average movie ratings by userID. Certain users give higher average ratings than others.

Lastly, genre-based movie rating variations were explored.



Graph 3: Histogram of average movie ratings grouped by genre. Certain genres have higher average ratings than others.

With insight into how ratings differ based on the movie, user, and genre, Equation 1 could be enhanced to account for rating deviations for the three predictors:

Equation 2:

$$Y_{u,i,g} = \mu + b_i + b_u + b_g + \varepsilon_{u,i,g}$$

Where:

$$\mu = average\ movie\ rating$$

$$Y_{u,i,g} = predicted\ rating\ of\ movie\ i\ from\ user\ u\ from\ genre\ g$$

$$b_i = average\ rating\ of\ movie\ i,\ centered\ at\ 0$$

$$b_u = average\ rating\ of\ user\ u,\ centered\ at\ 0$$

$$b_g = average\ rating\ of\ genre\ g,\ centered\ at\ 0$$

$$\varepsilon_{u,i,g} = independent\ errors\ sampled,\ centered\ at\ 0$$

The code below was used to train and evaluate the algorithm accounting for the three effects:

```
movie_average <- edx_train %>% group_by(movieId) %>% summarize(b_i = mean(rat
ing - mu_hat))

user_average <- edx_train %>% left_join(movie_average, by='movieId') %>% grou
p_by(userId) %>% summarize(b_u = mean(rating - mu_hat - b_i))

genre_average<- edx_train %>% left_join(movie_average, by='movieId') %>% left
_join(user_average, by = 'userId') %>% group_by(genres) %>% summarize(b_g = m
ean(rating - mu_hat - b_i - b_u))

predicted_ratings <- edx_test %>% left_join(movie_average, by='movieId') %>%
left_join(user_average, by='userId') %>%  left_join(genre_average, by='genres
') %>% mutate(prediction = mu_hat + b_i + b_u + b_g) %>% .$prediction

RMSE(edx_test$rating, predicted_ratings )
```

The average of all movie ratings, μ, served as the starting point in development of the algorithm. The subsequent effects added to improve the model (bu, bi, and bg) are centered around 0 because they were calculated as a deviation from the average value μ. Therefore, negative values for any of these predictors would indicate that on average, a specific movie, user, or genre yields a rating lower than the average value μ.

The movieId, userId, and genre predictors were added to the algorithm and evaluated through the RMSE in a piece-wise manner. Since more predictors meant the algorithm could account for more deviations from different predictors, the accuracy of the model's predictions increased, and the closer the algorithm got to reaching the target RMSE value. Further adjustments to the model were made to account for the fact that certain movies, users, and genres had fewer ratings than others. This can be demonstrated by analyzing the top five highest-rated movies from the training set:

Table 2:

```
## # A tibble: 5 x 4
## # Groups:   movieId [5]
##   movieId title                                     avg_rating   num
##     <dbl> <chr>                                          <dbl> <int>
## 1    3226 Hellhounds on My Trail (1999)                      5     1
## 2   33264 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                5     1
## 3   42783 Shadows of Forgotten Ancestors (1964)             5     1
## 4   51209 Fighting Elegy (Kenka erejii) (1966)              5     1
## 5   63772 Bullfighter and the Lady (1951)                   5     1
```

Equation 3:

$$Standard\ Deviation = \sqrt{((\sum_{i=1}^{n}((x_i - \bar{x})^2/(n-1))}$$

Where:

$$n = number\ of\ observations$$

$$x_i = observed\ value$$

$$\bar{x} = mean\ value\ of\ observations$$

Since having fewer ratings can increase the standard deviation in ratings, regularization was used to penalize movies, users, and genres with few ratings. To account for this, the following equation, containing the least squares equation and penalty, were minimized:
Equation 4:

$$(1/N)\sum_{u,i,g}(y_{u,i,g} - \mu - b_i - b_u - b_g)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2)$$

The penalization values, which are multiplied by lambda, could be minimized through calculus:

$$\widehat{b_i}(\lambda) = 1/(\lambda + n_i)\sum_{i=1}^{n_i}(Y_i - \mu)$$

$$\widehat{b_u}(\lambda) = 1/(\lambda + n_u)\sum_{u=1}^{n_u}(Y_u - \mu - b_i)$$

$$\widehat{b_g}(\lambda) = 1/(\lambda + n_g)\sum_{g=1}^{n_g}(Y_g - \mu - b_i - b_u)$$

Where:

$$n = number\ of\ ratings\ for\ the\ specific\ movie,\ user,\ or\ genre$$

$$b = regularized\ rating\ for\ the\ specific\ movie,\ user,\ or\ genre$$

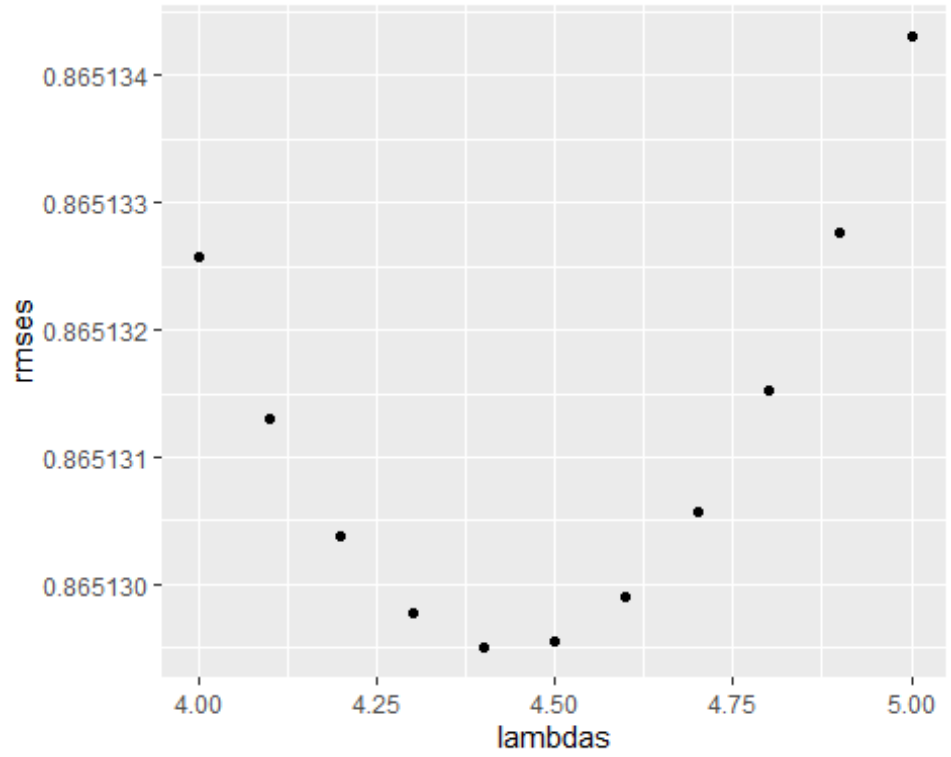
$$Y = individual\ movie\ rating$$

Once the individual values for bi, bu, and bg, are found, they are used in the following equation to gather a regularized prediction based on the movie, user, and genre:

Equation 5:

$$\hat{y} = \mu + b_i + b_u + b_g$$

Furthermore, since lambda is considered a value that can be adjusted, cross validation was implemented to determine which value of lambda, to the nearest tenth, minimized the RMSE. As shown in the graph below, the lowest RMSE the lowest RMSE came at lambda = 4.4.



Graph 4: Lambda vs RMSE value from 4.5 to 5.5, in intervals of 0.1.

Once lamda was optimized, the following code was used to develop the lambda-optimized algorithm:

```
reg_movie_average2 <- edx_train %>% group_by(movieId) %>% summarize(b_i = sum
(rating - mu_hat)/(n()+lambda), n_i = n())

reg_user_avg2 <- edx_train %>%
  left_join(reg_movie_average2, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu_hat - b_i)/(n()+lambda), n_i = n())

reg_genre_avg2 <- edx_train %>%
  left_join(reg_movie_average2, by='movieId') %>%
  left_join(reg_user_avg2, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu_hat - b_i - b_u)/(n()+lambda), n_i = n())

reg_predicted_ratings2 <- edx_test %>%
  left_join(reg_movie_average2, by='movieId') %>%
```

```
  left_join(reg_user_avg2, by='userId') %>%
  left_join(reg_genre_avg2, by='genres') %>%
  mutate(prediction = mu_hat + b_i + b_u + b_g) %>%
  .$prediction

RMSE(edx_test$rating, reg_predicted_ratings2)
```

The regularized model, with the minimized lambda of 4.4, was run against the validation set to gather a final predictions. When reviewing the predictions, there were two NA values. These NA's were from two records for movies that were only rated once in the original edx dataset. Since this record could not exist in both the training and test set at the same time, the movies were not placed in the sets. Therefore, when running the model for the validation dataset, values of NA were passed to the two for those two movies. The RMSE function had to be rewritten to include the argument "na.rm = TRUE" in the mean calculation to ignore the NA's. Once this was done, the final RMSE value was calculated successfully.

Results

Several different models were developed and evaluated to determine the model which minimized the root mean square error. Ultimately, the lambda-minimized (lambda equal to 4.4), regularized model that accounted for the Movie, User, and Genre effects was the most accurate model with a final RMSE value of 0.8648402.

Table 3: Algorithm method and its associated Root Mean Square Error. Granularizing the model to account for more predictors, along with regularizing the model to account for high standard deviations for users, movies, and genre rating counts, yielded the lowest RMSE.

method	RMSE
Just Average	1.0611350
Arbitrary Value	1.4663883
Movie Effect	0.9441568
Movie + User Effect	0.8659736
Movie + User + Genre Effect	0.8656019
Regularized (Lambda = 3) Movie + User + Genre Effect	0.8651655
Regularized (Lambda = 4.4) Movie + User + Genre Effect	0.8651295
VALIDATION: Regularized (Lambda = 4.4) Movie + User + Genre Effect	0.8648402

As Table 3 indicates, the more predictors incorporated into the training of the algorithm, the lower the RMSE. One important caveat is that each subsequent predictor added to the model improved the model by a decreasing margin. For instance, the difference between the “Just Average” model and the “Move Effect” model was greater than that of the “Movie Effect” model and the “Move + User Effect”. Additionally, as demonstrated in Table 2 and Graph 4, regularization and tuning the penalty parameter lambda improved the model.

Conclusion:

Ultimately, an algorithm was trained to accurately predict movie ratings based on the movie, user, and genre with a final RMSE value of 0.8648402. Our model was successful in doing so by partitioning the original dataset into a training and testing set, exploring the training set to extract useful information, and using that information to develop a model to predict ratings based off of notable predictor discrepancies. Furthermore, the results, which are in the form of Root Mean Square Error values in Table 2, illustrate that greater granularization of the model improved the Root Mean Square Error, however every subsequent predictor added to the model improved the model by a decreasing margin.

One limitation in this model is that the model does not account for the date and time in which a user rated the movie, nor the age of the user at the time of the rating. Viewing tastes change over time, and ratings from when a user was eight will not be helpful in predicting their viewing tastes when they're 18. In contradiction to the age difference, a smaller taste difference could be observed between a user at the ages of 68 and 78. In the case of younger viewers, age could play a major factor in ratings, specifically in an age-specific genre such as cartoons and animation.

In the future, Principal Component Analysis could be used to improve the model by determining which portions of the original dataset hold the most variation. Doing this could allow us to quickly analyze the whole dataset for points of great variation, which could then be used to create a more accurate version of the movie recommendation system.