MovieLens Project Report

Aileen Pan

4/11/2020

Overview

The purpose of the project is to create a movie recommendation system using the MovieLens dataset. We developed algorithm using the test set and predicted movie ratings in the validation set as if they were unknown for the final test of our algorithm. Root mean squared error (RMSE) was used to evaluate how close our predictions are to the true values in the validation set.

After spliting the train dataset further into separate training and test sets, five steps were performed to design and test the algorithm, including: -Step 0: Explore the data -Step 1: Predict with average ratings for all movies across all users -Step 2: Include moviei-specific effect b_i -Step 3: Include user-specific effect b_u -Step 4: Regularization -Step 5: Test final algorithm with validation set

Methods

———Create edx set, validation set———

Run the code given in the Movielens project instruction.

———Split edx into training and test sets———

```
set.seed(755)
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

#Make sure userId and movieId in test set are also in training set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

rm(test_index)
```

———Set up RMSE function———

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

———Setp 0: Explore the data———

General properties of the data.

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
```

A quick look at the data.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

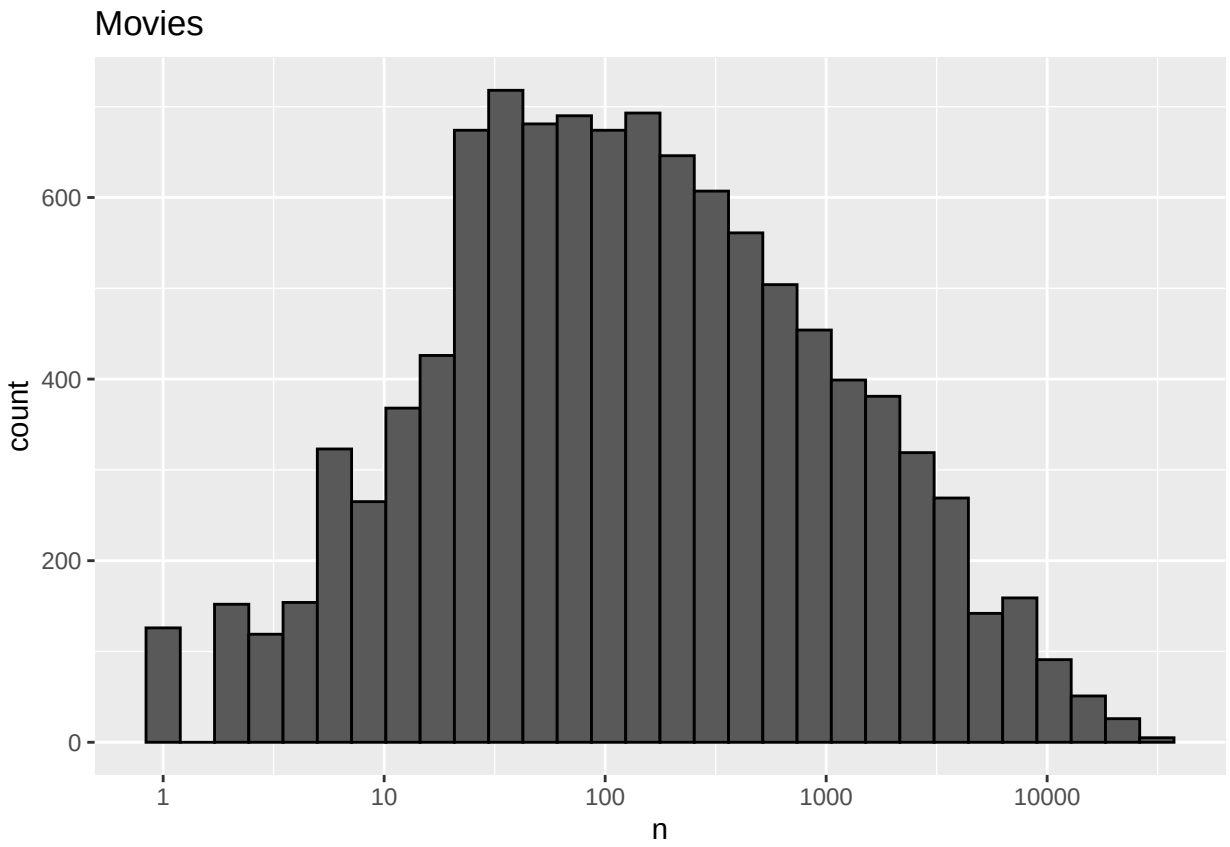
The number of unique users and movies:

n_users	n_movies
69878	10677

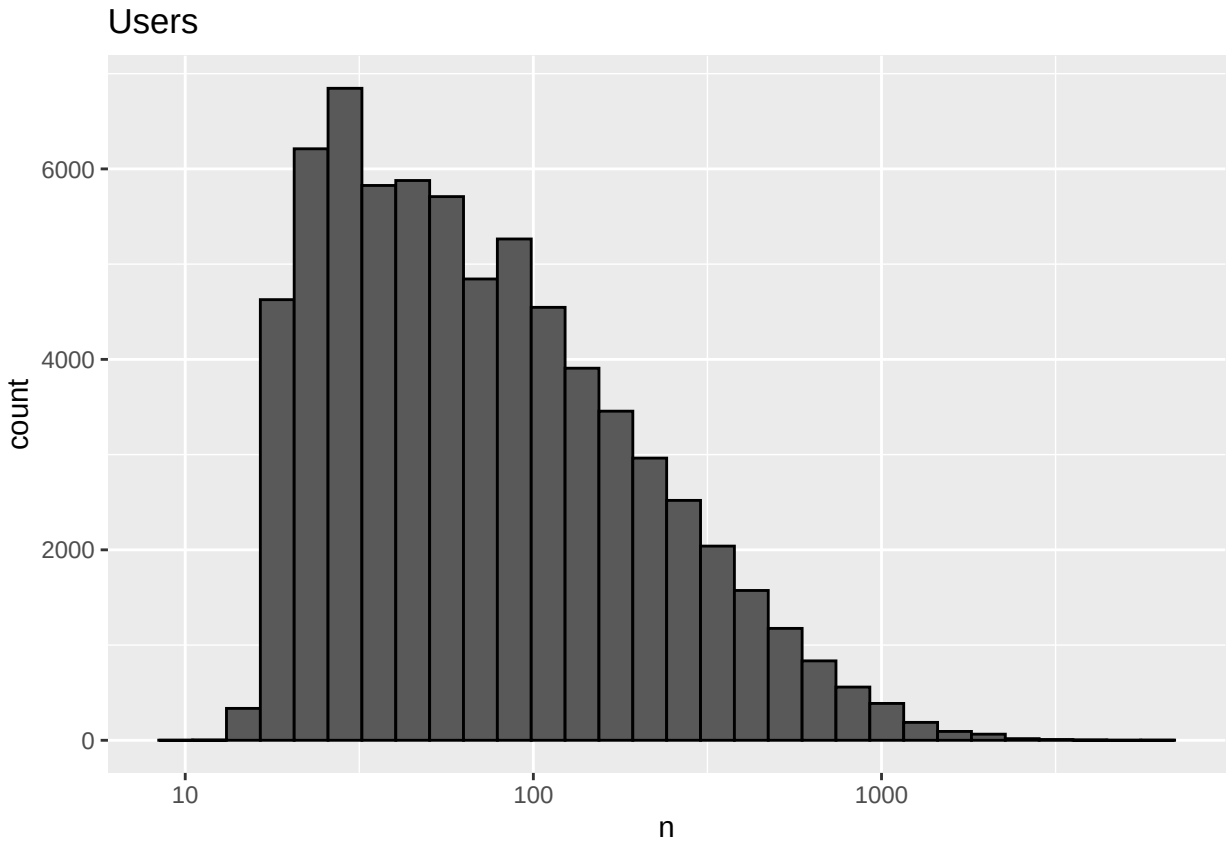
TOP 10 movies that have the greatest number of ratings:

movieId	title	count
296	Pulp Fiction (1994)	31362
356	Forrest Gump (1994)	31079
593	Silence of the Lambs, The (1991)	30382
480	Jurassic Park (1993)	29360
318	Shawshank Redemption, The (1994)	28015
110	Braveheart (1995)	26212
457	Fugitive, The (1993)	25998
589	Terminator 2: Judgment Day (1991)	25984
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
150	Apollo 13 (1995)	24284

Distribution of the number of ratings recieved by movies:



Distribution of the number of ratings given by users:



———Step 1: Predict with average ratings for all movies across all users———

Get the average rating.

```
mu_hat <- mean(train_set$rating)
cat(mu_hat)
```

```
## 3.512527
```

Predict all rating with the average rating and calculate RMSE.

```
naive_rmse <- RMSE(test_set$rating, mu_hat)
cat(naive_rmse)
```

```
## 1.060561
```

Create a table to store the results of RMSE.

method	RMSE
Just the average	1.060561

———Step 2: Include moviei-specific effect b_i———

Get b_i that represents the average rating for each movie i.

```
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

Predict with fitted model.

```
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
```

Calculate RMSE.

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868

Compared with the 'Just the average' model, the RMSE has decreased significantly.

———Step 3: Include user-specific effect b_u———

Get b_u for each user.

```
user_avgs <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

Predict with fitted model.

```
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
```

Calculate RMSE.

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8439865

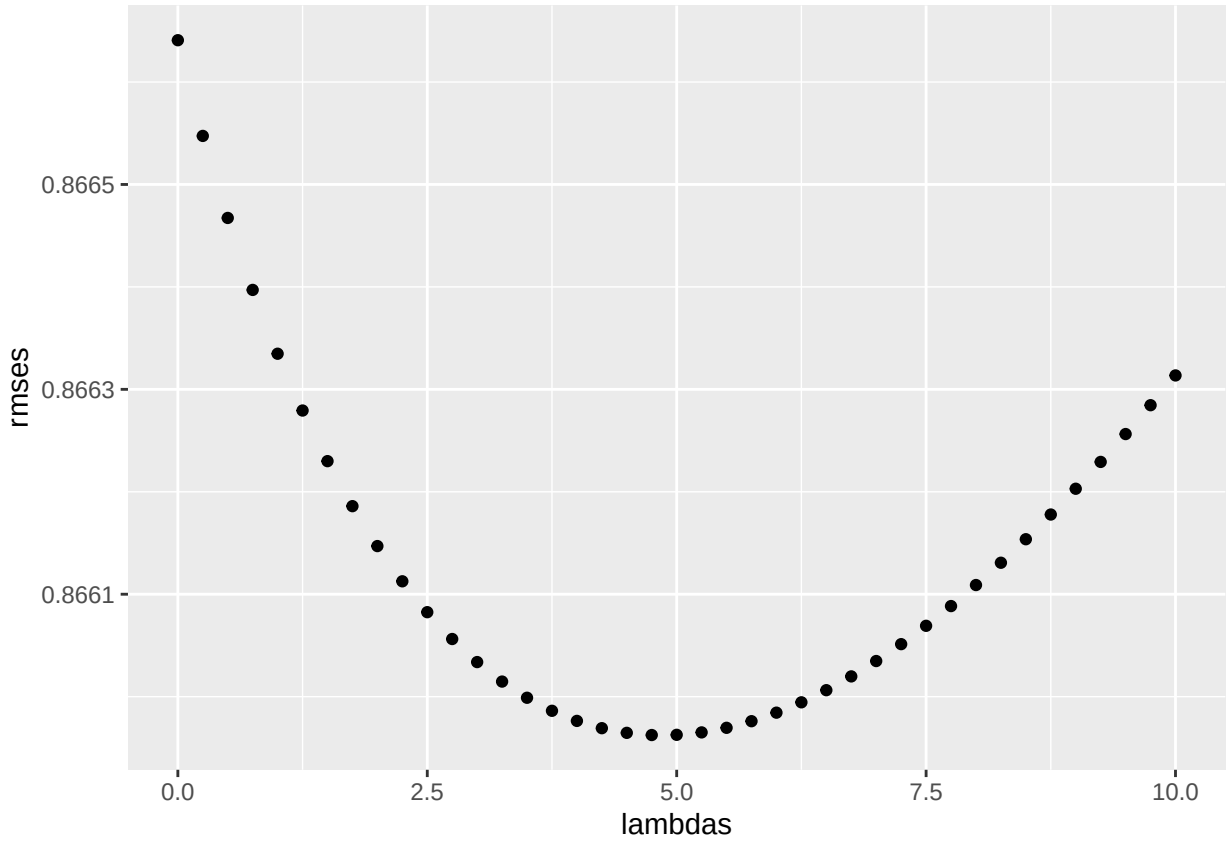
Now the RMSE continued going down after adding the user effect to the prediction.

———Step 4: Regularization———

Apply a range of lambdas to find the minimum RMSE.

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})
```

Plot the RMSE over lamdas.



According to the plot, the minimum RMSE is acheived when lamda equals to 5 approximately.

Generate the lamda that has minimum RMSE.

```
lambda <- lambdas[which.min(rmses)]
cat(lambda)
```

```
## 4.75
```

Store the result of RMSE.

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8439865
Regularized Movie + User Effect Model	0.8659626

Based on the test set, the RMSE after regularize movie and user effect increased a little compared to the unregularized model.

Results

———Step 5: Test final algorithm with validation set———

At this step, we will apply the regularization model to the validation set using the whole edx dataset as the training set.

Generate the new average ratings 'mu' using the whole edx dataset.

```
mu_edx<- mean(edx$rating)
cat(mu_edx)
```

```
## 3.512465
```

Get regularized movie and user-specific effects using the 'lamda' created before.

```
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_edx)/(n()+lambda))
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu_edx)/(n()+lambda))
```

Predict the validation set with the fitted model.

```
predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu_edx + b_i + b_u) %>%
  .$pred
```

Calculate the final RMSE.

```
model_final_rmse <- RMSE(predicted_ratings, validation$rating)
cat(model_final_rmse)
```

```
## 0.8648201
```

The results of each step was presented accordingly in the method section. The RMSE of our final model is 0.8648201, which is an acceptable result. By using the whole edx dataset as the training set, we were able to further bring the RMSE to the lower, from the previous 0.8659626.

Conclusion

The performance of the model met our requirement in terms of RMSE. However, our prediction was based on individuals and failed to account for an important source of variation related to the fact that groups of movies and groups of users have similar rating patterns. For the future work, We can observe these patterns by studying the residuals and matrix factorization.