Movie Recommentation

Vivien Leung

May 4, 2019

Introduction

For this project, we will be creating a movie recommendation system using the MovieLens dataset. The version of movielens included in the dslabs package is just a small subset of a much larger dataset with millions of ratings. We will be creating our own recommendation system. We will use the 10M version of the MovieLens dataset to make the computation a little easier. We will train a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set.

Analysis

Create edx set, validation set

Download the MovieLens 10M dataset:

https://grouplens.org/datasets/movielens/10m/ http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

Extract ratings data into the ratings table from the zip file

```
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))
```

Extract movies data into the movies table from the zip file

```
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
```

Transform movies table as dataframe

```
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
```

Join the table ratings and movies into a new dataframe movielens

```
movielens <- left_join(ratings, movies, by = "movieId")
```

Training data

Partition the movielens dataframe into edx (training set) and temp (validation set). Validation set will be 25% of MovieLens data [UPDATED 1/16/2019]

```
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.25, list = FALSE)
edx <- movielens[-test_index,] #train set
temp <- movielens[test_index,] #validation set
```

Make sure userId and movieId in validation set are also in edx set

```
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

Add rows removed from validation set back into edx set

```
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)
```

Cleanup

```
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Calculating RMSE

Let's write a function that computes the RMSE for vectors of ratings and their corresponding predictors:

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

We compute the average rating of all movies across all users on the edx training data

```
mu_hat <- mean(edx$rating)
mu_hat
```

```
## [1] 3.512537
```

And then we compute the residual mean squared error on the validation set data.

```
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

```
## [1] 1.060656
```

Let's create a results table with this naive approach:

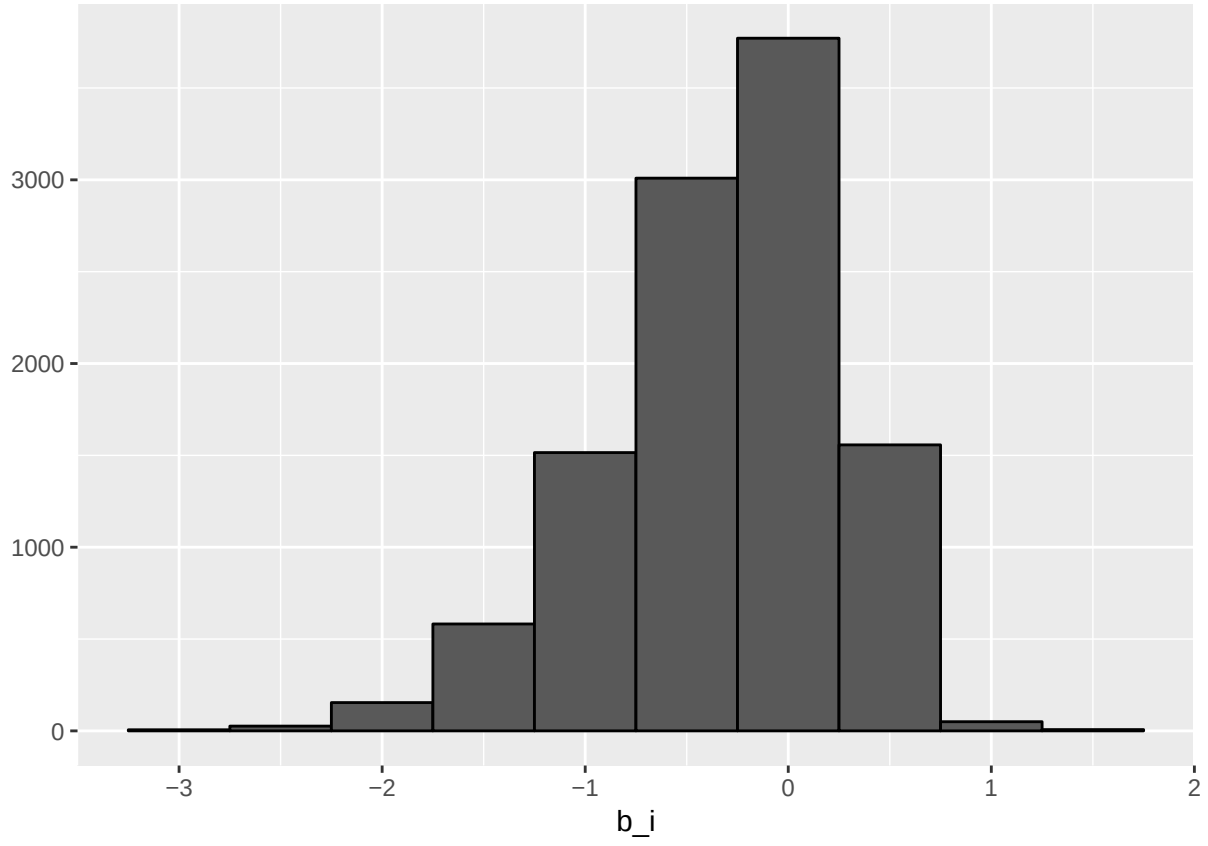
```
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)
```

We know that the average ranking for a movie is just the average of rating - mu. We can compute it like this:

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

We can see that these estimates vary substantially:

```
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```



Prediction

Using the following formula, > predicted ratings(ˆYu,i) = overall average(ˆmu) + average rating for a movie (ˆbi)

we can see that our RMSE did drop a little bit.

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

To view the rmse results of just the average vs the model generated using naive approach:

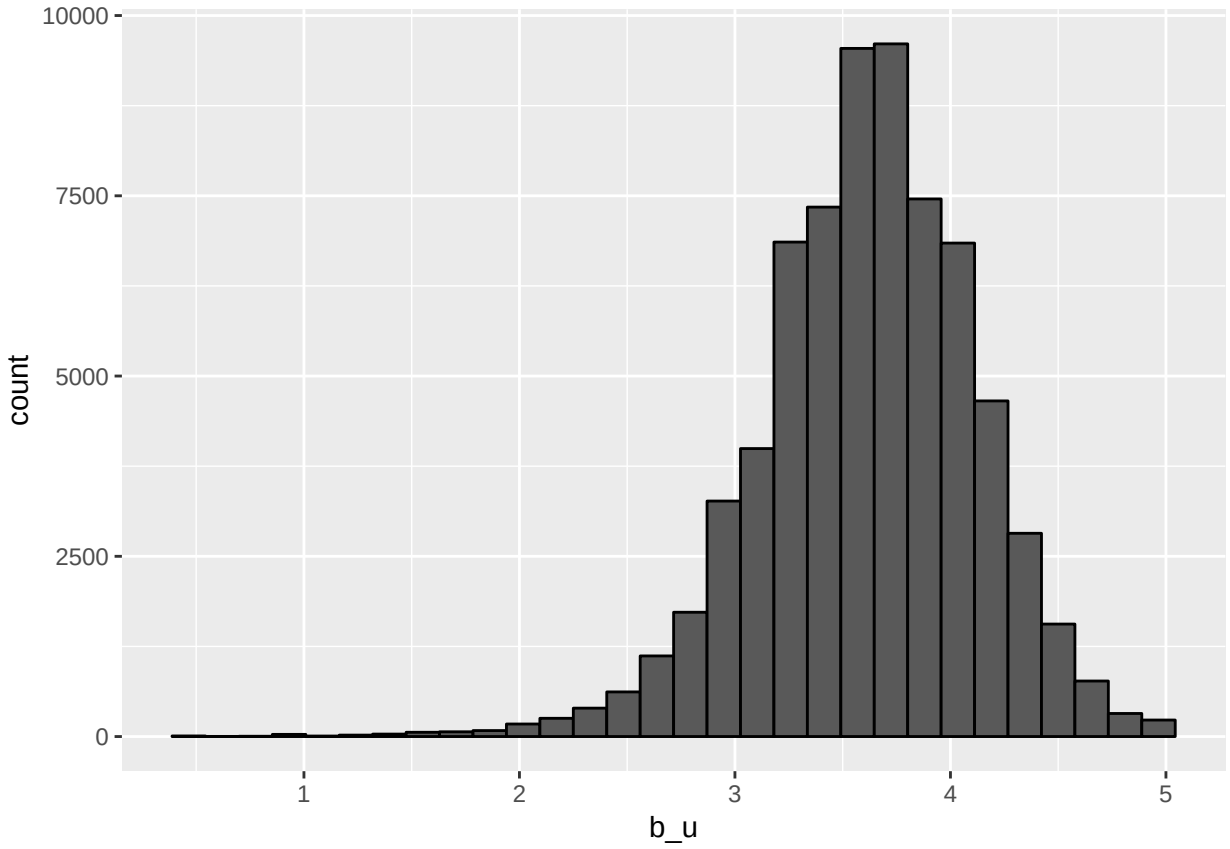
```
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0606558
Movie Effect Model	0.9441729

Improvements

The RMSE is 0.944 which is still high. Let's try something better. Let's compute the average rating for user u, for those that have rated over 100 movies.

```
validation %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



How about cranky users who often give bad ratings to good movies? Or how about users who love everything they watch? We will use bu as the user-specific effect. So now if a cranky user– this is a negative bu– rates a great movie, which will have a positive b i, the effects counter each other, and we may be able to correctly predict that this user gave a great movie a three rather than a five, which will happen, and that should improve our predictions.

We will compute our approximation by computing the overall mean, u-hat, the movie effects, b-hat i, and then estimating the user effects, b u-hat, by taking the average of the residuals obtained after removing the overall mean and the movie effect from the ratings yui. The code looks like this.

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

Results

We can now construct predictors and see how much the RMSE improves:

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

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))

rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0606558
Movie Effect Model	0.9441729
Movie + User Effects Model	0.8492176

Our residual error dropped down to about 0.84!

```
rmse <- model_2_rmse
rmse
```

```
## [1] 0.8492176
```

Conclusion

In this project, we have tried different methods of prediction. First we computed the RMSE for just using the average. Then we tried using the naive approach and improved RMSE from 1.06 to 0.944. However, we managed to improve it even further by selecting users with more ratings, then adjust results for cranky users who often give bad ratings to good movies. After that, we improved the RMSE to 0.849.