Movie Recommentation

Vivien Leung

May 4, 2019

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.

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

```r
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

```r
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

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

Transform movies table as dataframe

```r
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

```r
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]

```r
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.25, list = FALSE)
```

```r
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

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

Add rows removed from validation set back into edx set

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

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

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

Cleanup

```r
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:

```r
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

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

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

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

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

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

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

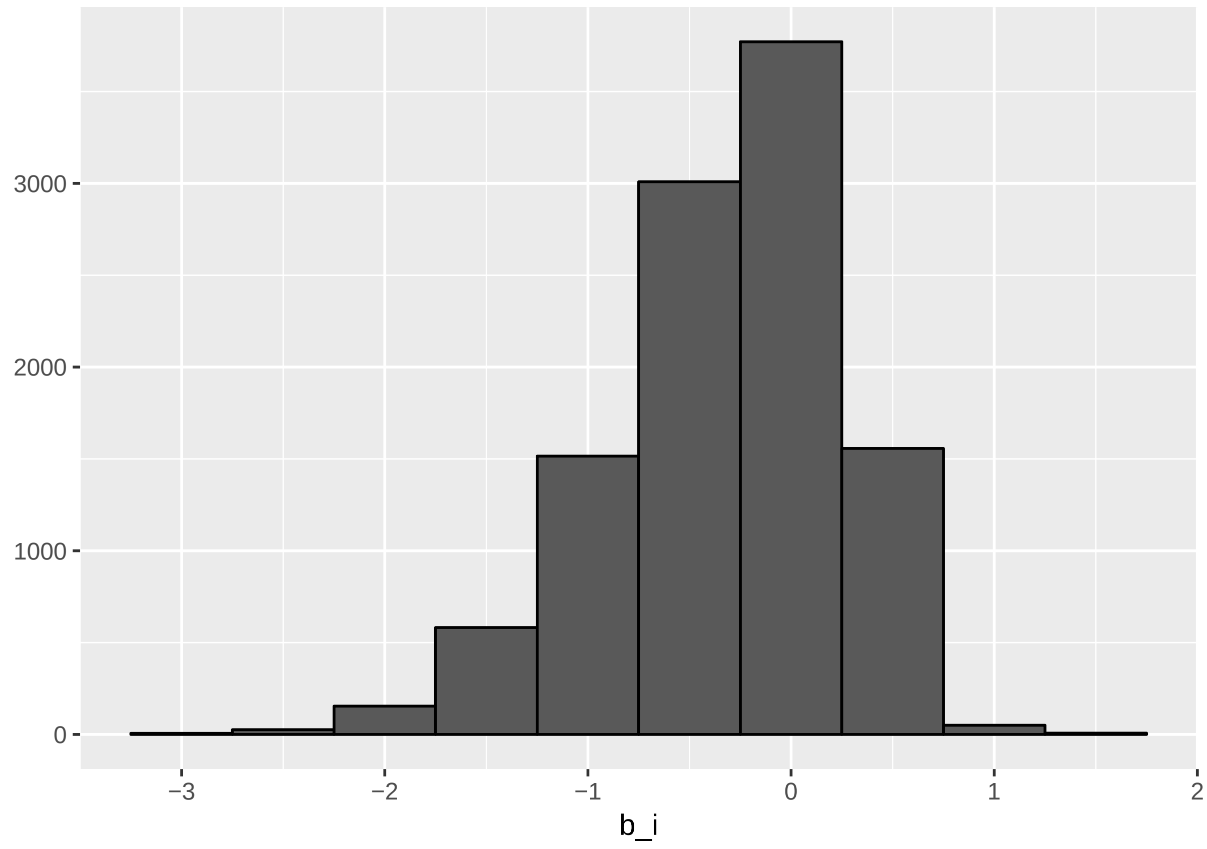
```r
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)
```

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(^Yu,i) = overall average(^mu) + average rating for a movie (^bi)

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:

```r
rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Naive Model",
                                     RMSE = rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0606558
Naive Model	0.9441729

The RMSE is:

```r
rmse
```

```
## [1] 0.9441729
```