MovieLens Machine Learning Project

Henry Chan

December 26, 2019

Executive Summary

MovieLens Project is about creating a movie recommendation system using the well known MovieLens dataset which is included in the dslab package To make the computation easier, in this project the 10M version of the MovieLens dataset will be used which is just a small subset of a much larger latest entire MovieLens dataset with millions of ratings.

The dataset consists of 10 millions of rows and 6 columns. The columns include userID, movieID, rating, timestamp, title and genres.

The foundation of the recommendation system is built by developing an algorithm to predict the ratings of movies to that the users haven't given. Movies with high predicted ratings will be recommended to the users.

Machine Learning Methods

Data Preparation

Before kicking start the machine learning, install the required packages and download the 10M MovieLens data file. Run the scripts to convert the data file into dataframe format. The dataframe is named as "movielens".

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------
------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------
----------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.or
g")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(caTools)) install.packages("caTools", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caTools
```

```
if(!require(dplyr)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(simEd)) install.packages("caTools", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: simEd
```

```
## Loading required package: rstream
```

```
##
## Attaching package: 'simEd'
```

```
## The following objects are masked from 'package:base':
##
##     sample, set.seed
```

```
if(!require(rstream)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(stats)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(tidyr)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(lattice)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("caTools", repos = "http://cran.us.r-project.org")
```

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

```
library(data.table)
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))
library(stringr)
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
library(dplyr)
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

Split the dataset into training and test set

To ensure that the data used in prediction is not included in training data used in building the algorithm, the dataset is splitted randomly into training set and valiation set by “caret” pacakge (here the random seed is set as 1). 10% of data is taken as test set. To make sure userId and movieId in test set also exist in training set, semi join to the test set against the training set is done.

```
library(caret)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
train <- movielens[-test_index,]
temp <- movielens[test_index,]
```

```
test <- temp %>%
  semi_join(train, by = "movieId") %>%
  semi_join(train, by = "userId")
```

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

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

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

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

Create RMSE function

Create the function of RMSE which evaluates the prediction accuracy of the algorithm. RMSE represents the square root of the mean of the square of the difference between predicted values and observed values.

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

Create the First Model

The first model is a simple tryout by predicting the same rating for all movies regardless of user. Variable mu represents the average of all ratings in the training set.

```
mu <- mean(train$rating)
mu
```

```
## [1] 3.512348
```

Output the RMSE of the first model.

```
model_1_rmse <- RMSE(test$rating, mu)
model_1_rmse
```

```
## [1] 1.059382
```

Create a table that is going to compare the RMSE results of various models.

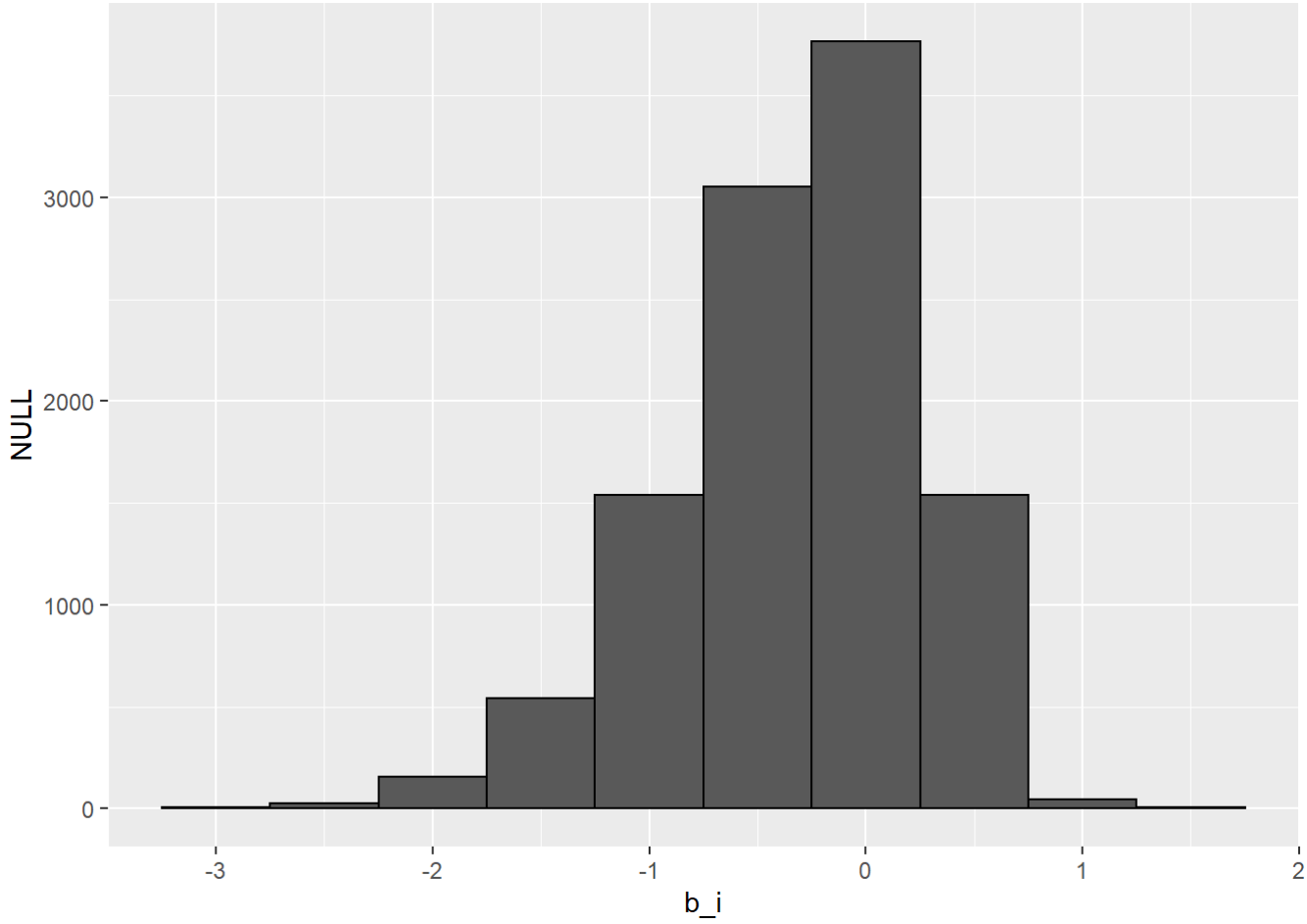
```
rmse_results <- tibble(method = "Just the average", RMSE = model_1_rmse)
```

Create the Second Model

The second model takes the movie effect into account. Estimate the movie effect b_i by taking the mean of rating - mu by movieId.

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

A histogram is plotted to see the distribution of b_i.



Predict the ratings in test set.

```
predicted_ratings <- mu + test %>% left_join(movie_avgs, by='movieId') %>% pull(b_i)
```

Output the RMSE of the second model.

```
model_2_rmse <- RMSE(predicted_ratings, test$rating)
model_2_rmse
```

```
## [1] 0.9428223
```

Put the RMSE to the table.

```
rmse_results <- bind_rows(rmse_results, data_frame(method="Movie Effect Model",RMSE=model_2_r
mse))
```

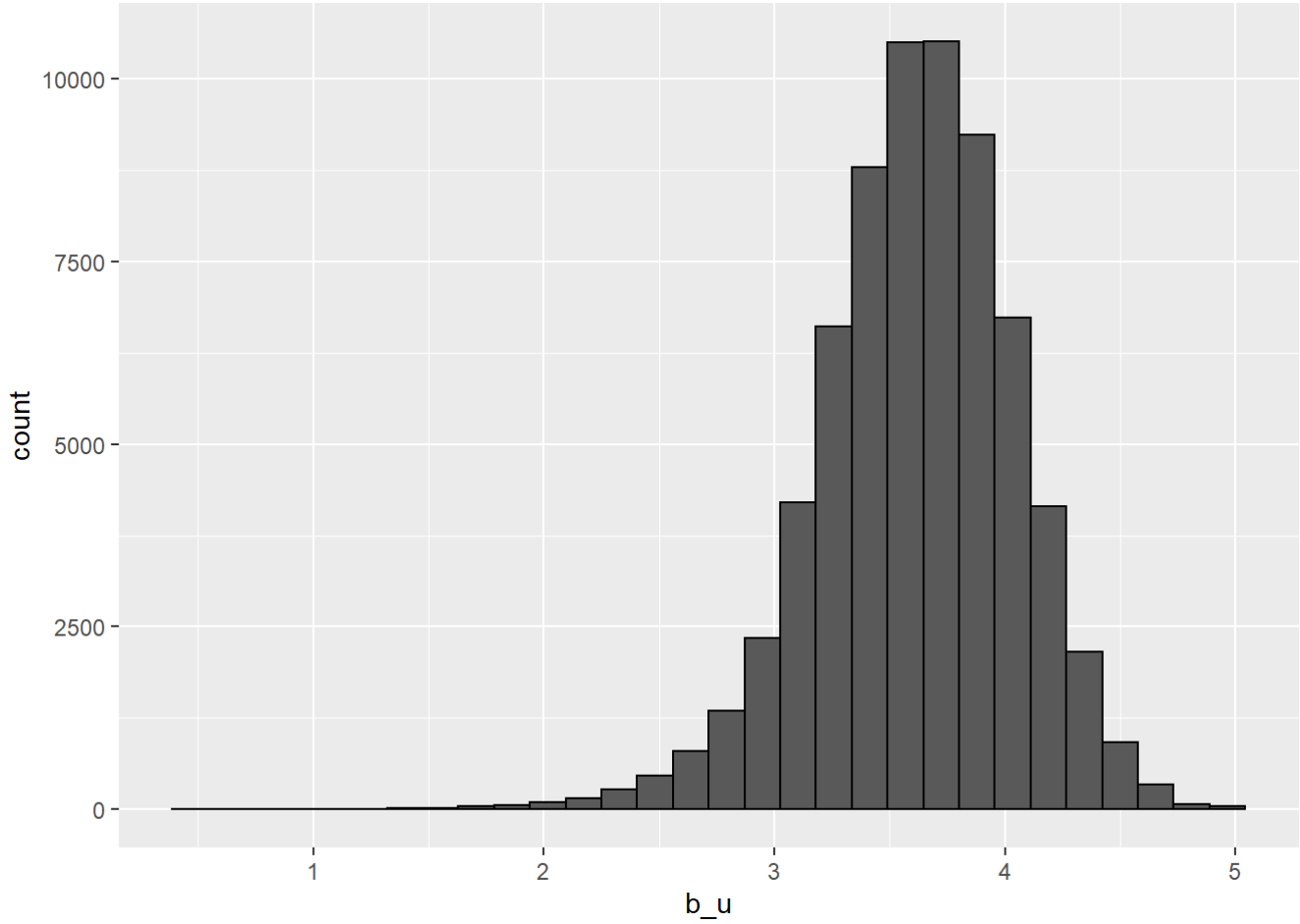
```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

Create the Third Model

The third model adds the user effect on top of the second model. Estimate the user effect b_u by taking the mean of rating - mu - b_i by movieId.

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

A histogram is plotted to see the distribution of b_u.



Predict the ratings in test set.

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

Output the RMSE of the third model.

```
model_3_rmse <- RMSE(predicted_ratings, test$rating)
model_3_rmse
```

```
## [1] 0.8641079
```

Put the RMSE to the table.

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_3_rmse))
```

Results

```
rmse_results
```

method <chr>	RMSE <dbl>
Just the average	1.0593825
Movie Effect Model	0.9428223
Movie + User Effects Model	0.8641079

3 rows

Based on the RMSE results table, the Third Model, which includes both the movie and user effect, gives the lowest RMSE value at 0.864 and provides the best estimation.

Conclusion

In the project 3 models are built. The first model, being a simple naive model, gives an initial idea that if the mean of all ratings in the training dataset is taken as the predicted value for all predictions, the RMSE is as high as more than 1. When the movie effect is introduced in the second model, the RMSE drops to 0.942. When user effect is further introduced in the third model, the prediction accuracy further improves and RMSE drops to 0.864. It is suggested to apply the Third Model in the movie recommendation system.