Harvard-capstone-Movielens-Report

Rajeev Kumar Rajesh

02/08/2020

Introduction

This capstone project on 'Movie recommendation system'.The primary goal of recommendation systems is to help users find what they want based on their preferences and previous interactions, and predicting the rating for a new item. In this document, we create a movie recommendation system using the MovieLens dataset.

Data Ingestion

```
############################################################
# Create edx set, validation set, and submission file
############################################################

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

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

```
## -- Attaching packages ---------------------------------------------------------------------- tidyverse 1
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------ tidyverse_conflic
## 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.org")
```

```
## 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
```

```
# Loading all needed libraries
library(dplyr)
library(tidyverse)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
#download file then read it

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

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

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


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

```
                                            genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
#Test and training set Data
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

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

# Make sure we don't include  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)

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

# Validation dataset can be further modified by removing rating column
validation_CM <- validation
validation <- validation %>% select(-rating)
```

Executive Summary

The purpose for this project is creating a recommender system using MovieLens dataset.

The version of movielens dataset used for this final assignment contains approximately 10 Milions of movies ratings, divided in 9 Milions for training and one Milion for validation. It is a small subset of a much larger (and famous) dataset with several millions of ratings. Into the training dataset there are approximately **69,878 users** and **10,677 different movies** divided in 20 genres such as Action, Adventure, Horror, Drama, Thriller and more.

After a initial data exploration, the recommender systems builted on this dataset are evaluated and choosen based on the RMSE - Root Mean Squared Error that should be at least lower than **0.86490**.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

```
# The RMSE function that will be used in this project is:
#  Since RMSE (root mean squre error) is used frequency so lets define a function
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
```

Exploratory Data Analysis

Lets modify the columns to suitable formats that can be further used for analysis

```
# Modify the year as a column in the edx & validation datasets
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
# Modify the genres variable in the edx & validation dataset (column separated)
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
split_valid <- validation   %>% separate_rows(genres, sep = "\\|")
split_valid_CM <- validation_CM  %>% separate_rows(genres, sep = "\\|")
```

Baseline Model: just the mean

The formula used is:

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

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0.

```
mu <- mean(edx$rating)
baseline_rmse <- RMSE(validation_CM$rating,mu)

##Create a table to store result
rmse_results <- data_frame(method = "Using mean only", RMSE = baseline_rmse)
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

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

method	RMSE
Using mean only	1.061202

Movie Effect (b_i)

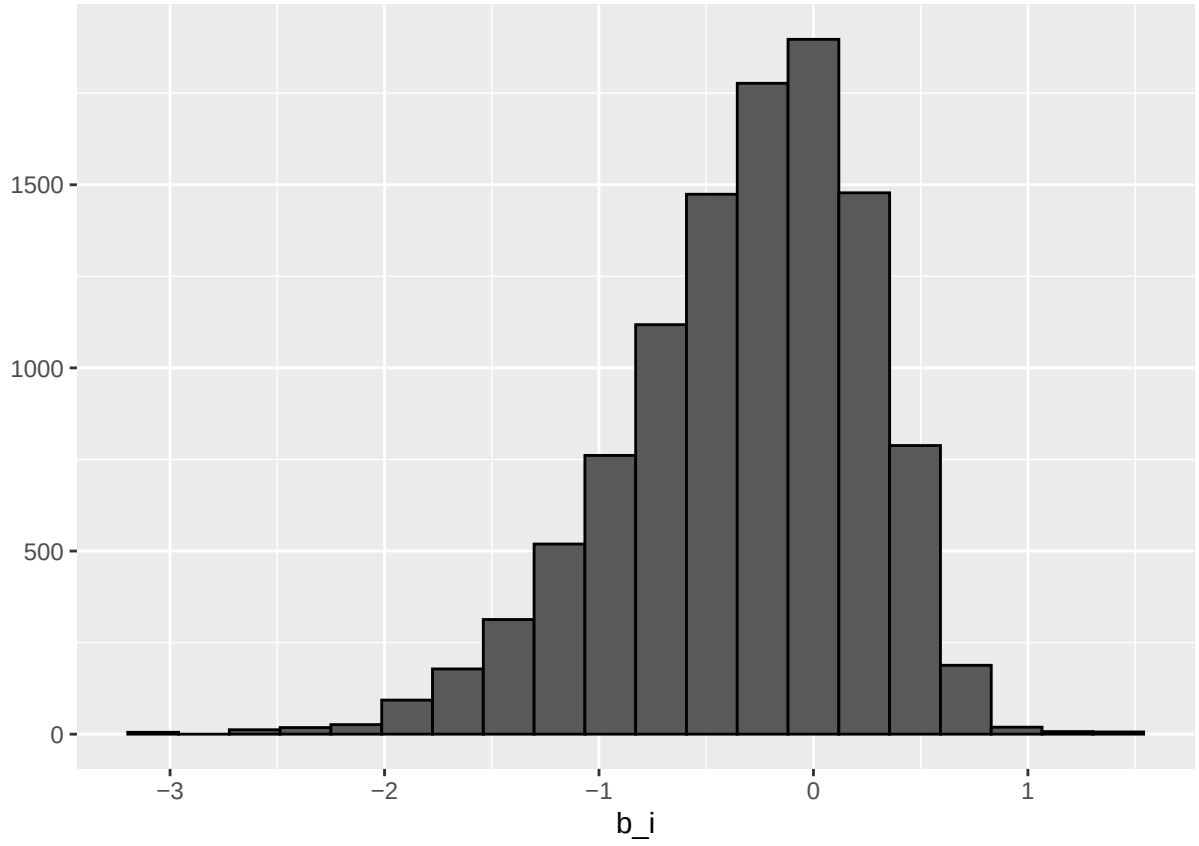
Different movies are rated differently. As shown in the exploration, the histogram is not symmetric and is skewed towards negative rating effect. The movie effect can be taken into account by taking the difference from mean rating as shown in the following chunk of code.

The formula used is:

$$Y_{u,i} = \hat{\mu} + b_i + \epsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i.

```
#different movie are rated diffrently
movie_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu), .groups = 'drop')
movie_avgs_norm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black"))
```




```
# Movie effects only
predicted_ratings_movie_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  mutate(pred = mu + b_i)
model_1_rmse <- RMSE(validation_CM$rating,predicted_ratings_movie_norm$pred)
rmse_results <- bind_rows(rmse_results,data_frame(method="Movie Effect Model",  RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087

Movie(b_i) and user effects(u_i)

Different users are different in terms of how they rate movies. Some cranky users may rate a good movie lower or some very generous users just don't care for assessment. . We can calculate it using this code.

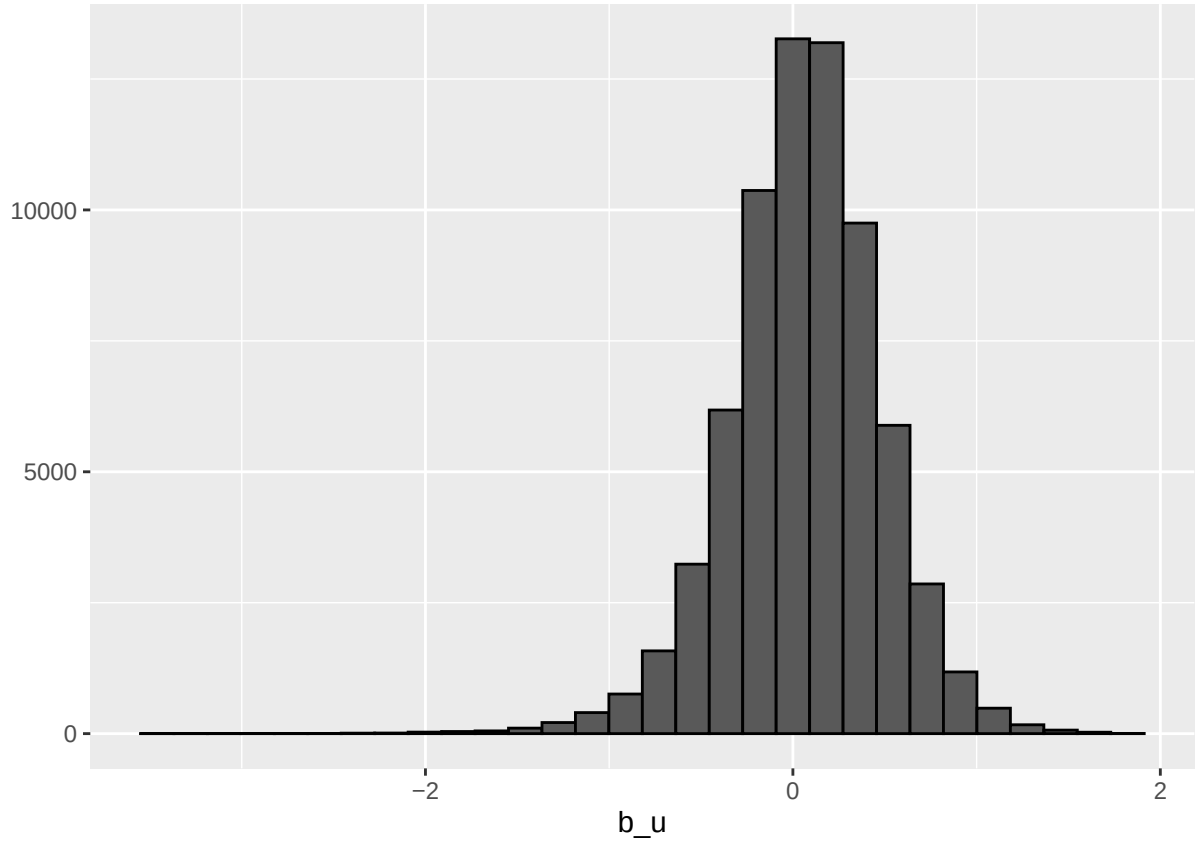
The formula used is:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + \epsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie i, i.e. the bias of movie i. The b_u is a measure for the mildness

of user u, i.e. the bias of user u.

```
user_avgs_norm <- edx %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i), .groups = 'drop')
user_avgs_norm %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```




```
# Use test set,join movie averages & user averages
# Prediction equals the mean with user effect b_u & movie effect b_i
predicted_ratings_user_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  left_join(user_avgs_norm, by='userId') %>%
  mutate(pred = mu + b_i + b_u)
# test and save rmse results
model_2_rmse <- RMSE(validation_CM$rating,predicted_ratings_user_norm$pred)
rmse_results <- bind_rows(rmse_results,data_frame(method="Movie and User Effect Model",  RMSE = model_2
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

Regularization based approach

We have noticed in our data exploration, some users are more actively participated in movie reviewing. There are also users who have rated very few movies (less than 30 movies). On the other hand, some movies are rated very few times (say 1 or 2). These are basically noisy estimates that we should not trust. Additionally, RMSE are sensitive to large errors. Large errors can increase our residual mean squared error.

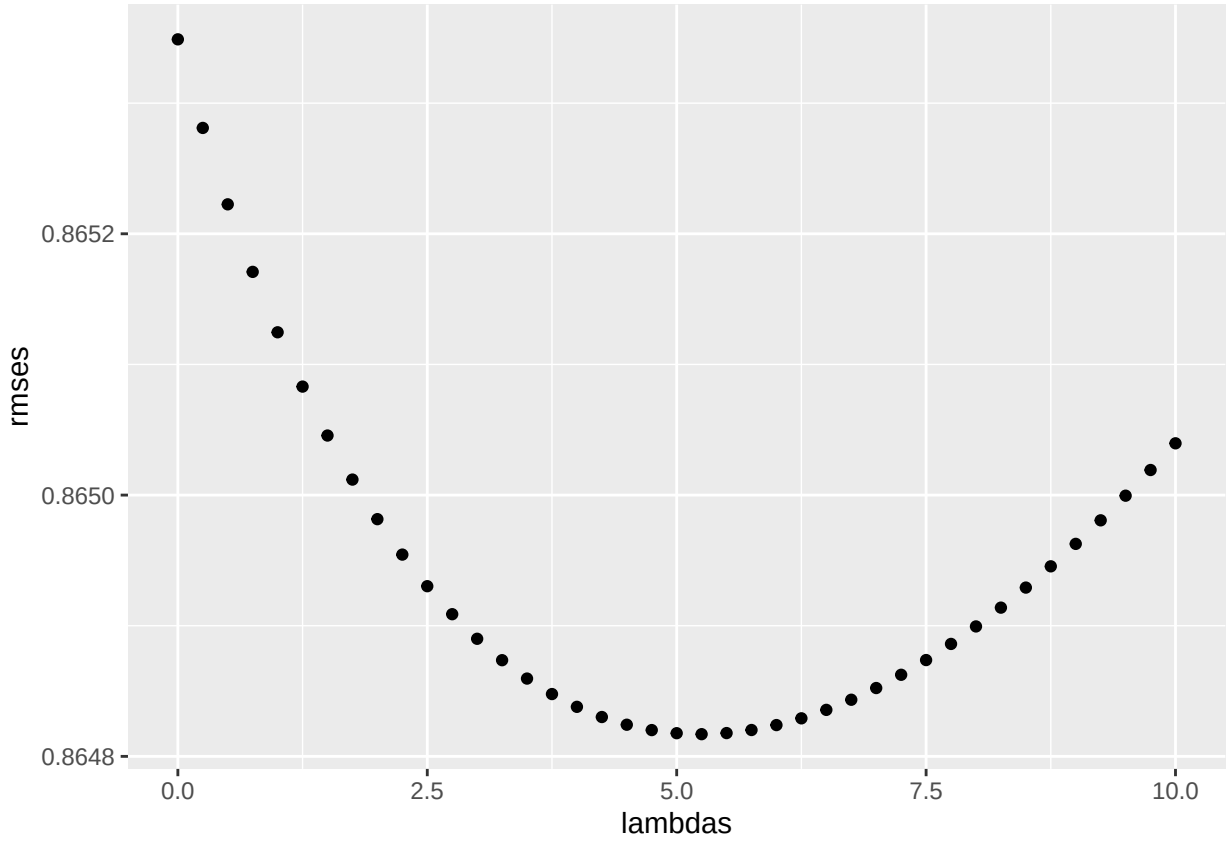
The regularization method allows us to add a penalty λ (lambda) to penalizes movies with large estimates from a small sample size. In order to optimize b_i, it necessary to use this equation:

$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i)^2+\lambda\sum_i b_i^2$$

reduced to this equation:

$$\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu})$$

```
lambdas <- seq(0, 10, 0.25)
# For each lambda,find b_i & b_u, followed by rating prediction & testing
# note:the below code could take some time
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l),.groups = 'drop')

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l), .groups = 'drop')

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

  return(RMSE(validation_CM$rating,predicted_ratings))
})

# Plot rmses vs lambdas to select the optimal lambda
qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
# Compute regularized estimates of b_i using lambda
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n(),.groups = 'drop')
# Compute regularized estimates of b_u using lambda
user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n(),.groups = 'drop')
# Predict ratings
predicted_ratings_reg <- validation %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  left_join(user_avgs_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
# Test and save results
model_3_rmse <- RMSE(validation_CM$rating,predicted_ratings_reg)
rmse_results <- bind_rows(rmse_results,data_frame(method="Regularized Movie and User Effect Model",  RM
rmse_results %>% knitr::kable(align = "c")
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

method	RMSE
Regularized Movie and User Effect Model	0.8648170

Conclusion

We started collecting and preparing the dataset for analysis, then we explored the information seeking for insights that might help during the model building.

We started the linear model with a very simple model which is just the mean of the observed ratings. From there, we added movie and user effects, that models the user behavior and movie distribution. With regularization we added a penalty value for the movies and users with few number of ratings. The linear model achieved the RMSE of 0.8648177,successfully passing the target of 0.8649.