EDX Capstone MovieLens

Jamien Lim

1 March 2021

Contents

1 Overview

This is project report for EDx HarvardX: PH125.9 - Data Science: Capstone with regards to the MovieLens's recommendation system. The initial manipulation of the given dataset is first prepared by HarvardX and thereafter, a series of data exploration is carried out before proceeding on to the development of an appropriate model to predict movie ratings in the validation set.

2 Introduction

Recommendation systems are widely used by many companies nowadays. For example, Netflix utilizes this system to suggest movies to the users according to their past activities. Similarly, such systems can be applied to other platforms where the system recommends the user based on the information obtained from them.

In this project, a subset of MovieLens dataset (MovieLens 10M dataset) is used to develop a model to predict the ratings of certain movies by a sample of users. The initial script for the dataset is given by HarvardX and after exploration of the dataset, a method was utilized to develop ratings prediction. In this case, minimization of Residual Mean Square Error (RMSE) loss function was chosen to be the methodology for this project. In addition, an iterative approach was used to add effect and regularization parameters to the model, further optimising it as a recommendation system. This report details the procedure, analysis and results produced by the methodology.

3 Methods

In this section, various exploration and chosen methods were used to develop the predictions of movie ratings. Firstly, the dataset is first initialized with the following code (From EDX HarvardX) to create the training and testing dataset for this project.

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################
# Note: this process could take a couple of minutes
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Warning: package 'dplyr' was built under R version 4.0.3
```

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

```
## Warning: package 'caret' was built under R version 4.0.3
```

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

library(tidyverse)
library(caret)
library(data.table)
library(stringr)
library(tidyr)

# 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)

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")

# if using R 4.0 or later:
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")
```

```r
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

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

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

# 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)
edx <- rbind(edx, removed)

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

3.1 Data exploration and creation of RMSE function

The data is first explored to look into the various aspects of the data to understand it better. In this project, RMSE is utilized to develop the model. The formula of RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - \hat{y}_{u,i})^2}$$

with $\hat{y}_{u,i}$ and $y_{u,i}$ being the predicted and actual ratings, and N, the number of possible combinations between user u and movie i.

The variable `validation` contains the true ratings from `validation$rating`, while variable `y_hat` contains the corresponding predictions.

The square root of the mean of the differences between true and predicted ratings is evaluated as and defined as follows.

```r
##############################################################
# Data exploration and creation of RMSE function
##############################################################
# Dimension of edx
dim(edx)
```

```
## [1] 9000055       6
```

```r
# Summary of edx
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
```

```
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title               genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
# Movies rating summary
edx %>% group_by(rating) %>% count()
```

```
## # A tibble: 10 x 2
## # Groups:   rating [10]
##    rating       n
##     <dbl>   <int>
##  1    0.5   85374
##  2    1    345679
##  3    1.5  106426
##  4    2    711422
##  5    2.5  333010
##  6    3   2121240
##  7    3.5  791624
##  8    4   2588430
##  9    4.5  526736
## 10    5   1390114
```

```
# Number of movies
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```
# Number of users
n_distinct(edx$userId)
```

```
## [1] 69878
```

```
# Number for different genres
genres = c("Drama", "Comedy", "Thriller", "Romance")
sapply(genres, function(g) {
  sum(str_detect(edx$genres, g))
})
```

```
##    Drama   Comedy Thriller  Romance
##  3910127  3540930  2325899  1712100
```

```
# Movie ranking in rating
edx %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        count
##      <dbl> <chr>                                                        <int>
##  1     296 Pulp Fiction (1994)                                          31362
##  2     356 Forrest Gump (1994)                                          31079
##  3     593 Silence of the Lambs, The (1991)                             30382
##  4     480 Jurassic Park (1993)                                         29360
##  5     318 Shawshank Redemption, The (1994)                             28015
##  6     110 Braveheart (1995)                                            26212
##  7     457 Fugitive, The (1993)                                         25998
##  8     589 Terminator 2: Judgment Day (1991)                            25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10     150 Apollo 13 (1995)                                             24284
## # ... with 10,667 more rows
```

```
# Distribution of rating
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>%
  arrange(desc(count))
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

```
##### Model 1 - Average Edx rating ####

# Mean rating of dataset
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

```
# RMSE Function
RMSE <- function(validation, y_hat){
  sqrt(mean((validation - y_hat)^2))
}
```

3.2 Model 1: Average rating system

This model utilizes a simple approach to tackle the recommendation system prediction model. This model is designed to use the mean of all ratings in `edx`, following the formula:

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

With that in mind, the formula is translated into the code as follows:

```r
############################################################
# Creation of Model 1
############################################################
# RMSE_1 testing
rmse_1 <- RMSE(validation$rating, mu)
rmse_1
```

```
## [1] 1.061202
```

```r
rmse_results <- data_frame(method = 'Model 1 - Average rating', RMSE = rmse_1)
```

```
## 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.
```

```r
# Results compilation
rmse_results %>% knitr::kable()
```

method	RMSE
Model 1 - Average rating	1.061202

3.3 Model 2: Movie Effect system

Model 2 adds a parameter on the rating of the movies on top of Model 1. In this way, the new model will either increase or decrease the predicted rating based on the forumla as shown:

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

In the code as follows, `avg_movies` represents the movie IDs and the movie effect paremeter is represented by b_i. With that in mind, the formula is translated into code as follows:

```r
############################################################
# Creation of Model 2
############################################################
# Movie effect is taken into account  for this model
# b_i is the mean difference between movie rating and average rating
avg_movies <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# RMSE_2 testing
pred_ratings <- mu +  validation %>%
  left_join(avg_movies, by='movieId') %>%
  pull(b_i)
rmse_2 <- RMSE(pred_ratings, validation$rating)
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method='Model 2 - Movie Effect',
                                     RMSE = rmse_2 ))

# Results compilation
rmse_results %>% knitr::kable()
```

method	RMSE
Model 1 - Average rating	1.0612018
Model 2 - Movie Effect	0.9439087

3.4 Model 3: User Effect Model

In order to obtain a better prediction result, it is important that we include a user-dependent parameter as well because it is capable of affecting the whole prediction analysis. Model 3 adds a user-dependent parameter on top of Model 2. In this way, the new model will either increase or decrease the predicted rating based on the forumla as shown:

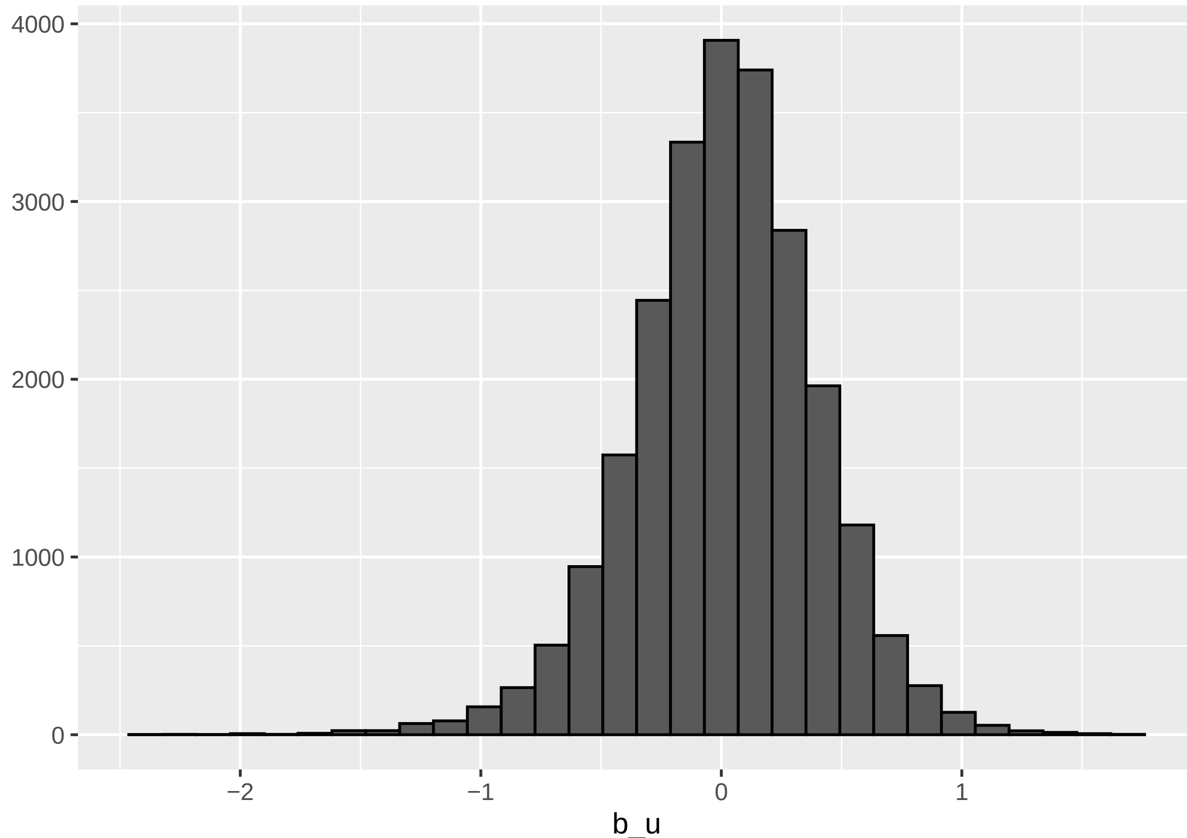
$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$

In the formula stated, b_u is the user effect parameter. On the other hand, `avg_movies` represents the b_i parameter, whereas `avg_user`represents the b_u parameters. With that in mind, the formula is translated into code as follows:

```
#############################################################
# Creation of Model 3
#############################################################
# Movie and user effects are taken into account  for this model
# Plot User Effect #
avg_user <- edx %>%
  left_join(avg_movies, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
avg_user %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```




```
avg_user <- edx %>%
  left_join(avg_movies, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# RMSE_3 testing
pred_ratings <- validation%>%
  left_join(avg_movies, by='movieId') %>%
  left_join(avg_user, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

rmse_3 <- RMSE(pred_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Model 3 - Movie + User Effect",
                                     RMSE = rmse_3))

# Results compilation
rmse_results %>% knitr::kable()
```

method	RMSE
Model 1 - Average rating	1.0612018
Model 2 - Movie Effect	0.9439087
Model 3 - Movie + User Effect	0.8653488

3.5 Model 4: Regularization of Movie + User Effect

Model 3 had a decent result in predicting the movies, however, more can be done to improve it further. It is important to remember that there are movies that had few ratings as compared to others, which can potentially affect the whole system. Therefore, to standardize the model, we reduced the unpopular movies' effect towards zero by regularizing Model 3. Regularization parameter, lambda, is added to the model and the forumla is as shown:

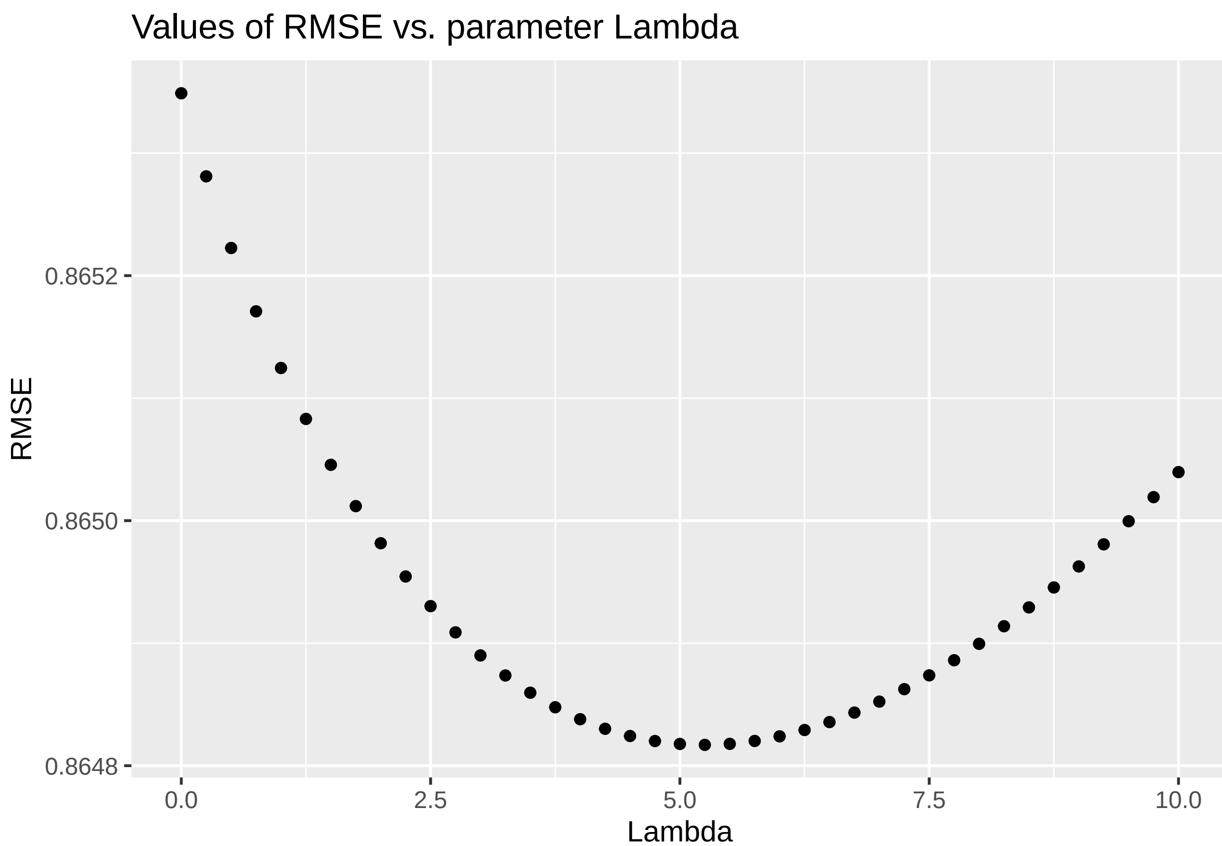
$\hat{b_i}(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$

In the formula stated, parameter lambda is optimized by testing an array of values to find out the value that minimizes RMSE the most. With that in mind, the formula is translated into code as follows:

```r
#############################################################
# Creation of Model 4 - Regularization of Movie + User Effect
#############################################################
# Movie and user effects are regularised in this model
# Fine tuning of lambda
lambdas <- seq(0, 10, 0.25)

tuning <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

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

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

  return(RMSE(pred_ratings, validation$rating))
})
```

A qplot was constructed to visualise the optimal value for lambda, with its values plotted against their corresponding RMSE. With that, we are able to obtain the minimum RMSE based on the qplot shown.

```r
# Plot of lambda tuning and find optimal lambda
qplot(lambdas, tuning, main="Values of RMSE vs. parameter Lambda",
      xlab="Lambda", ylab="RMSE")
```




```r
opt_lambda <- lambdas[which.min(tuning)]
opt_lambda
```

```
## [1] 5.25
```

```r
# RMSE_4 testing
rmse_4 <- min(tuning)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Model 4 - Regularization of Movie + User Effect",
                                     RMSE = rmse_4))
```

4 Results

In this section, RMSE of all the models defined in this project are presented and it showed that regularization of Model 3 is able to reduce the RMSE further.

```r
##### Final compilation of results #####
rmse_results %>% knitr::kable()
```

method	RMSE
Model 1 - Average rating	1.0612018
Model 2 - Movie Effect	0.9439087
Model 3 - Movie + User Effect	0.8653488
Model 4 - Regularization of Movie + User Effect	0.8648170

5 Conclusion

In conclusion, we aim to present the methodology and process in obtaining an appropriate RMSE for the HarvardX Capstone Project. With the processes defined in this report, we managed to minimize the RMSE loss function of the true and predicted ratings of the dataset.