MovieLens Recommendation Sysstem

Fréderic Van der Cruyssen

3/15/2021

Introduction

In this capstone project we want to predict user movie rating based on a large dataframe "the Movielens 10M database". We start off with creating the datasets and doing some exploration of the data. Next, we will build a simple linear model and further refine it towards an accurate final model to predict user movie ratings.

Creating datasets

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

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

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- 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.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
```

```
library(tidyverse)
library(DescTools)
```

```
##
## Attaching package: 'DescTools'
```

```
## The following object is masked from 'package:data.table':
##
##     %like%
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
library(tinytex)
knitr::opts_chunk$set(echo = FALSE)

# 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")

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")

# We create a test set based on 10% of the total edx set.
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# We verify if movieId and UserId are also present in both validation and test set.

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

# Removed rows are back added to the edx set to continue calculations.

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)
```

Overview of dataset

We arrange the dataset by number of ratings en number of movies with a signle rating. We can see that there are 10.000 movies with 10 million ratings. The most rated movie is Pulp Fiction.

```
# Most rated films
edx %>% group_by(title) %>%
  summarize(n_ratings = n()) %>%
  arrange(desc(n_ratings))
```

```
## # A tibble: 10,676 x 2
##    title                                                     n_ratings
##    <chr>                                                         <int>
##  1 Pulp Fiction (1994)                                           31383
##  2 Forrest Gump (1994)                                           30981
##  3 Silence of the Lambs, The (1991)                              30271
##  4 Jurassic Park (1993)                                          29364
##  5 Shawshank Redemption, The (1994)                              28025
##  6 Braveheart (1995)                                             26310
##  7 Fugitive, The (1993)                                          26130
##  8 Terminator 2: Judgment Day (1991)                             26083
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)  25754
## 10 Apollo 13 (1995)                                              24377
## # ... with 10,666 more rows
```

```
# Number of movies rated once
edx %>% group_by(title) %>%
  summarize(n_ratings = n()) %>%
```

```
  filter(n_ratings==1) %>%
  count() %>% pull()
```

```
## [1] 123
```

Overview of columns

We have split the dataset in a training and test set. The test set consists of 999.988 rows and 6 columns.

```
glimpse(validation)
```

```
## Rows: 999,990
## Columns: 6
## $ userId    <int> 1, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, ...
## $ movieId   <dbl> 480, 539, 590, 780, 1276, 5505, 21, 110, 153, 165, 329, 7...
## $ rating    <dbl> 5.0, 3.0, 5.0, 3.0, 3.5, 2.0, 3.0, 5.0, 5.0, 5.0, 5.0, 3....
## $ timestamp <int> 838983653, 868246262, 868245608, 868244698, 1133571205, 1...
## $ title     <chr> "Jurassic Park (1993)", "Sleepless in Seattle (1993)", "D...
## $ genres    <chr> "Action|Adventure|Sci-Fi|Thriller", "Comedy|Drama|Romance...
```

Basic model

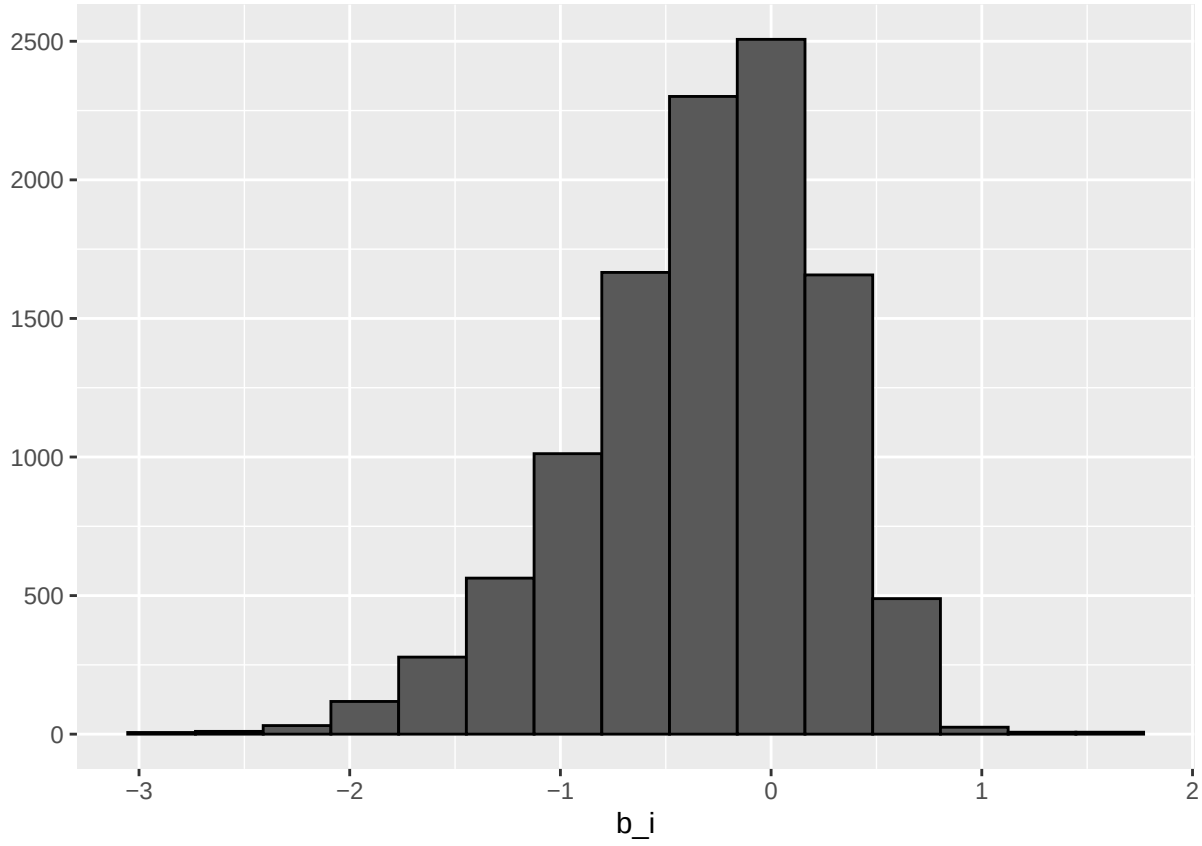
We start with a simple model by using the average method and calculate RMSE.

```
## [1] 1.06057
```

Improving the model

Next we'll try to improve the model by adding an error factor b_i to the equation and calculate RMSE and the qplot for the distribution of b_i's.

```
## [1] 0.9433053
```



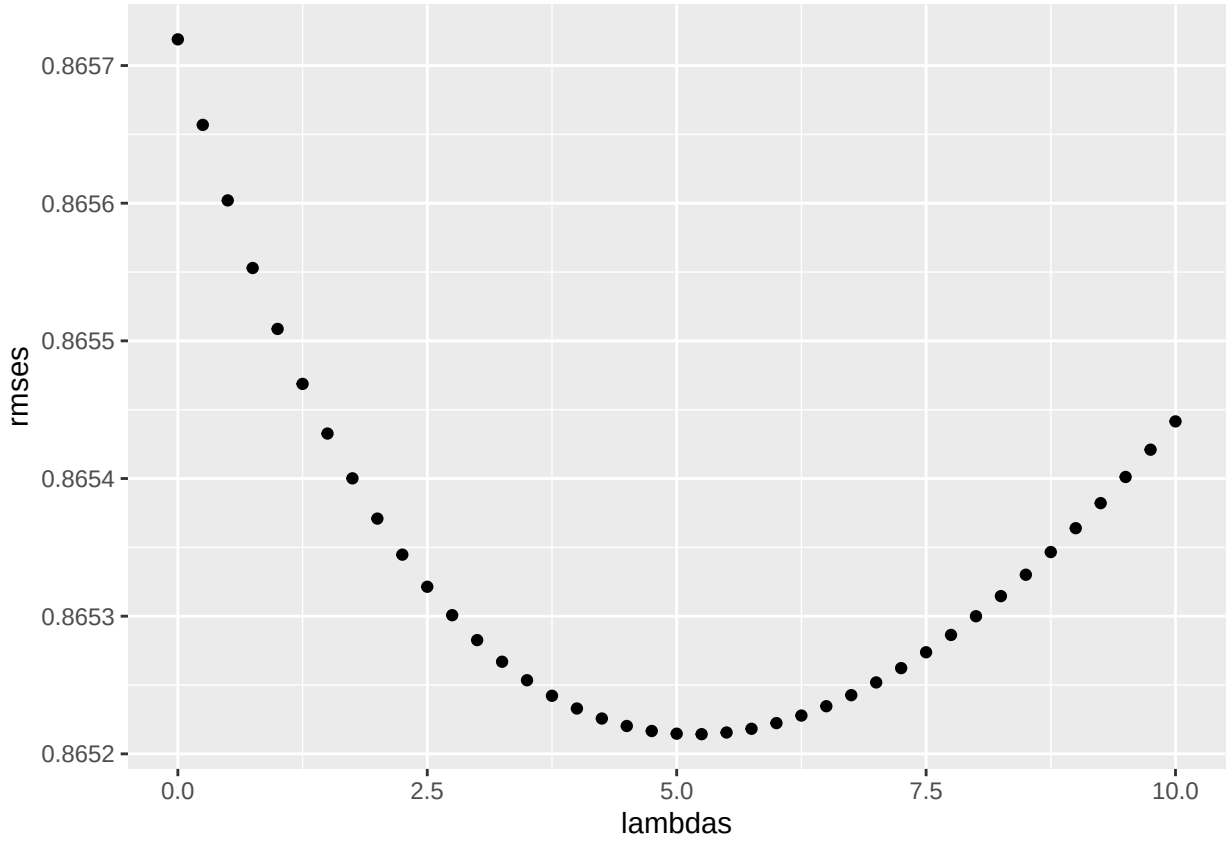
Further refinements

We refine the model by adding extreme user ratings, b_u, into the equation. We rerun the analysis and calculate RMSE.

```
## [1] 0.865719
```

Regularization

We want to remove large errors in the prediction model by using regularization.




```
## [1] 0.8652143
```

Final model

The final model RMSE now becomes:

```
## [1] 0.8652143
```

Conclusion

We summarize the RMSE for each model built:

We can see incremental improvements to the RMSE as we supplant our model with bias terms and regularization.

Method	RMSE
Average	1.060219
Movie effect	0.9439066
Movie and user effects	0.8656791
Final model	0.8651767