Capstone

Nelson Levy K F Macedo

08/03/2022

Introduction

This report is part of the EDX Course HarvardX: PH129.9X Data Science: Capstone. In this part, the objective is create a movie recommendations system using the MovieLens dataset.

The initial code provided downloads the data and creates a train set (edx) and a test set (validation). The code provided can be retrieved in the **Create Train and Final Hold-out Test Sets** section of the course.

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes

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

```
## Carregando pacotes exigidos: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     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")
```

```
## Carregando pacotes exigidos: caret
```

```
## Carregando pacotes exigidos: lattice
```

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

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

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

```
## Carregando pacotes exigidos: 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
```

```r
library(tidyverse)
library(caret)
library(data.table)

# 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 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                           title = as.character(title),
#                                           genres = as.character(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")

# 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
```

```
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)
```

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

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

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

Initial Analysis (descriptive)

The edx dataset has 6 variables (userId, movieId, rating, timestamp, title and genres) and 9.000.055 observations.

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
##  - attr(*, ".internal.selfref")=<externalptr>
```

In the edx dataset there are 10.677 different movies evaluated by 69.878 different users:

```
n_distinct(edx$movieId)
```

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

```
n_distinct(edx$userId)
```

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

In the differente genres of movies, the most common genres are Drama (3910127 observations), Comedy (3540930), Action(2560545) and Thriller(2325899):

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

genres = c("Drama", "Comedy", "Thriller", "Romance")
sapply(genres, function(genre) {
    sum(str_detect(edx$genres, genre))
})
```

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

The movie with the greatest number of ratings is "Pulp Fiction"

```r
edx %>% group_by(movieId, title) %>%
    summarize(count = n()) %>%
    arrange(desc(count))
```

```
## `summarise()` has grouped output by 'movieId'. You can override using the
## `.groups` argument.
```

```
## # 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
```

Method

For the challenge porpoused, the approach used is the penalized least squares. The method was choosed because has been tested in similar cases with very good results (Irizarry, 2022). To do so, I used the mean of ratings for the movie and then penalize the bias for movies and bias for users (details ahead), summarized like this:

$Y\ u, i\ = mean + bmovie + buser$

To calculate the RMSE a function was created:

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

The first part of the approach is to determine the optimal lambda. Here, the lambda is a tunning factor that optimizes the Root Mean Square Error Loss (RMSE). For possible lambdas, was tried all values from 0 to 5, increasing 0.1 per attempt as follow:

```
#Testing the best lambda (tuning parameter)
possible_lambdas <- seq(0, 5, 0.1)

#Creates a function to test each possible lambda
rmses <- sapply(possible_lambdas, function (l){

  #Calculate mean of ratings in the training set
  mean_rating<- mean(edx$rating)

  #Bias of movies
  b_movies <- edx %>%
    group_by(movieId) %>%
    summarize(b_movies = sum(rating - mean_rating)/(n()+l))

  #Bias of users
  b_users <- edx %>%
    left_join(b_movies, by= "movieId") %>%
    group_by(userId) %>%
    summarize(b_users = sum(rating - b_movies - mean_rating)/(n()+l))

  #predict ratings with each lambda
  predicted_ratings <- edx %>%
    left_join(b_movies, by = "movieId") %>%
    left_join(b_users, by = "userId") %>%
    mutate(pred = mean_rating+ b_movies + b_users) %>%
    .$pred

  #returns RMSE for each lambda
  return(RMSE(predicted_ratings, edx$rating))
})

# Uses the minimal RMSE as lambda
  lambda <- possible_lambdas[which.min(rmses)]
```

Determined the optimal value for lambda was 0.5, giving a RMSE = 0.8566952, the next step is calculate the mean and the biases for movies and users, predicting the ratings for the users:

```
pred_y <- sapply(lambda, function (l){

  #Calculate mean rating
  mean_rating<- mean(edx$rating)

  #Movie bias with optimal lambda
  b_movies <- edx %>%
    group_by(movieId) %>%
    summarize(b_movies = sum(rating - mean_rating)/(n()+l))

  #User bias with optimal lambda
  b_users <- edx %>%
    left_join(b_movies, by="movieId") %>%
```

```
    group_by(userId) %>%
    summarize(b_users = sum(rating - b_movies - mean_rating)/(n()+l))

  #Predict ratings on test set (validation)
  predicted_ratings <- validation %>%
    left_join(b_movies, by = "movieId" ) %>%
    left_join(b_users, by = "userId") %>%
    mutate(pred = mean_rating + b_movies + b_users) %>%
    .$pred

  return(predicted_ratings)
})
```

For the last, let's calculate the RMSE in the validation set:

```
RMSE_final <- RMSE(validation$rating, pred_y)
```

Results

With the predictions ready, the results were exported to a .csv file named "predictions.csv":

```
write.csv(validation %>%
  select(userId, movieId) %>%
  mutate(rating = pred_y), "predictions.csv", na = "", row.names=FALSE)
```

With the method applied, there was possible to determine a lambda factor that optmized the RMSE method. Applying the knowledge provided by the course, it was able to estimate a rating some person would give to a movie, based on the mean of that movie by other users, plus some penalyzes from the movie bias and the users bias, penalyzing the movies with a low amount of rates.

The results can be exporte in .csv format so it can be used in other platforms, providing integration with other formats and informatic languages.

Conclusion

The objective was to create a recommendations system for movies in a dataset from MovieLens. The approach used was the penalized least squares, already known by data scientists as a valid method for similar analysis. The optimal lambda result in RMSE = 0.8566952 and the final RMSE = 0.8652226, considered a enough result for the porpouses of the project.

References

Irizary, R. (2022) Introduction to Data Science, Data analysis and prediction algorithms with R. Available in https://rafalab.github.io/dsbook/.