MovieLens Project

Federico J. Caballero Ferrari

2023-02-15

Contents

1 Introduction

The main goal of this project is to create a system to recommend movies as accurately as possible basing on their rating by users.

The data set MovieLens (which can be downloaded **here**) has been provided to start with specific data.

Any machine learning method could be applied in this project; however, several effects or bias have been found using the data provided. In order to select the best parameter and limit the penalization of small samples, cross-validation has been used.

1.1 Using the code provided [1]

MovieLens data initially has 6 columns:

- **userId**: user single identifier code
- **movieId**: movie single identifier code
- **rating**: movie rating by this user
- **timestamp**: rating date
- **title**: title and year of the movie
- **genres**: all the genres included in the movie

MovieLens data is split into two groups when downloaded: *edx* as train set and *final_holdout_test* as test set. That data partition was made using the *createDataPartition* function.

1.2 Data dimensions

The train set, *edx*, is 90 % of the original data. These are the dimensions:

```r
# Dimensions of edx (train dataset)
rows_edx <- nrow(edx)
col_edx <- ncol(edx)
```

As we can see, the train dataset *edx* has 9000055 rows and 6 columns.

On the other hand, the test set *final_holdout_test* is equivalent to the remaining 10 %:

```r
# Dimensions of final_holdout_test (test dataset)
rows_test <- nrow(final_holdout_test)
col_test <- ncol(final_holdout_test)
```

Which means that the test dataset *final_holdout_test* has 999999 rows and 6 columns. This dataset will only be used in order to calculate the accuracy of the algorithm.

Next, the number of unique users and movies rated in the *edx* dataset is also calculated:

```r
# Number of unique users and movies in the train dataset
edx %>% summarize(users = n_distinct(userId), movies = n_distinct(movieId))
```

[1]The initial code provided by the course for the the MovieLens project is run at this point. It is not shown in the pdf file due to problems with some characters when knitting the file; however, the code has not been modified and can be found in the R and Rmd files

```
##   users movies
## 1 69878  10677
```

```
userID_unique_train <- n_distinct(edx$userId)
movieID_unique_train <- n_distinct(edx$movieId)
```

As we can see, there are 69878 unique users and 10677 unique movies in the train dataset.

We can also run a similar calculation for the *final_holdout_test* test dataset:

```
# Number of unique users and movies in the test dataset
final_holdout_test %>% summarize(users = n_distinct(userId), movies = n_distinct(movieId))
```

```
##   users movies
## 1 68534   9809
```

```
userID_unique_test <- n_distinct(final_holdout_test$userId)
movieID_unique_test <- n_distinct(final_holdout_test$movieId)
```

Which means that there are 68534 unique users and 9809 unique movies in the test dataset.

2 Methods/analysis

The structure of the data provided can be found below:

```
# Structure of the data provided
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

As we can see, there are 2 columns with data which cannot be handled directly:

- **timestamp** (unreadable format)
- **title** (since the year of the movie is also included after the title)

For this reason, data cleaning has been carried out on these 2 columns.

2.1 Data cleaning

As stated before, the *timestamp* column has data with unreadable format, so it has been changed to datetime format through data cleaning (it requires the *lubridate* package):

```
# Convert timestamp column into date-time format (it requires the lubridate package)
if(!require(lubridate)) install.packages("lubridate")
library(lubridate)
edx <- edx %>% mutate(timestamp = as_datetime(timestamp))
```

Next, the rating year in that column is rounded as following:

```
# Create new column for rating year
edx <- edx %>% mutate(rate_year = year(timestamp))
```

Regarding the title column, the title and rating year have been separated using the regex format:

```
# Extract title in title column
edx <- edx %>% extract(title, c("title", "year"), "(.*) \\((\\d{4})\\)")
```

After performing these actions, the data structure is as shown below:

```
# New dataset structure
head(edx)
```

```
##   userId movieId rating           timestamp                    title year
## 1      1     122      5 1996-08-02 11:24:06                Boomerang 1992
## 2      1     185      5 1996-08-02 10:58:45                 Net, The 1995
## 4      1     292      5 1996-08-02 10:57:01                 Outbreak 1995
## 5      1     316      5 1996-08-02 10:56:32                 Stargate 1994
## 6      1     329      5 1996-08-02 10:56:32 Star Trek: Generations 1994
## 7      1     355      5 1996-08-02 11:14:34       Flintstones, The 1994
##                          genres rate_year
## 1                Comedy|Romance      1996
## 2         Action|Crime|Thriller      1996
## 4  Action|Drama|Sci-Fi|Thriller      1996
## 5       Action|Adventure|Sci-Fi      1996
## 6 Action|Adventure|Drama|Sci-Fi      1996
## 7       Children|Comedy|Fantasy      1996
```

2.2 Modeling approach

In order to achieve the most accurate results, several factors have been taken into account when analyzing the effect on the rating.

Even though the mean of all movies could work as a first approach, the following kinds of bias have been considered:

- Movie bias: some movies are better rated than others
- Movie year bias: movies made in different years are also rated differently
- User bias: not all users follow the same criteria when rating a movie
- Rating year bias: movies made in different years are also rated differently
- Genre bias: movies of different genres are usually rated differently. A movie can be related to more than one genre; nevertheless, for this algorithm we have considered the specific bias of each combination of genres. The reason is that, for example, the bias of a movie which has Comedy and Action cannot be explained by just combining the bias of Comedy and Action separately. As a result, for the former example the algorithm would process the bias for the combination Comedy / Action

Therefore, the rate of a movie has been estimated as following:

pred = mo + m_b + m_y_b + u_b + r_y_b + g_b

where:

- **pred**: predicted rate of the movie
- **mo**: mean of all the movies
- **m_b**: movie bias
- **m_y_b**: movie year bias

- **u_b**: user bias

- **r_y_b**: rating year bias
- **g_b**: genre bias

2.3 RMSE

The ***Residual Mean Squared Error*** **(RMSE)** is the method used to calculate the error between the predictions and the actual ratings.

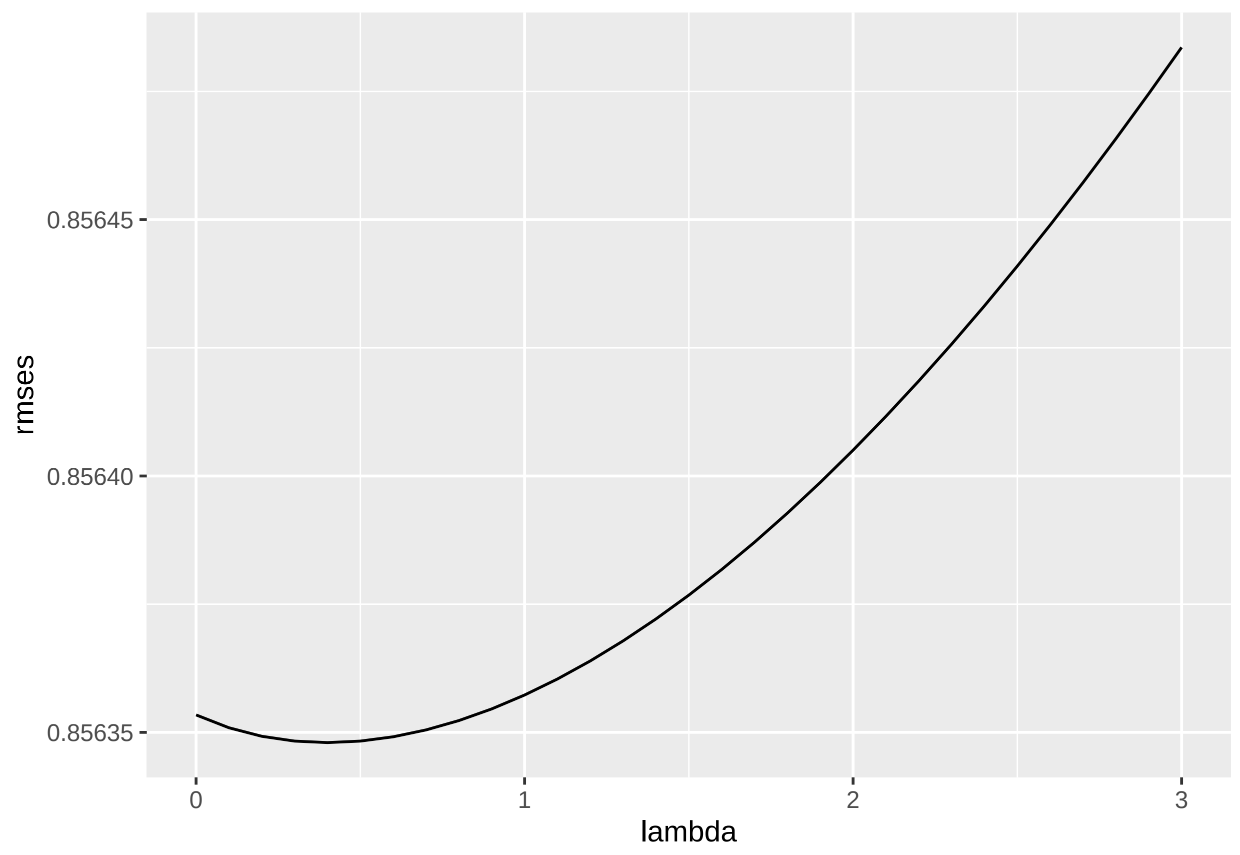
In this case, RMSE has been calculated as following:

```
# Calculate RMSE
RMSE <- function(predicted, real){
  sqrt( mean( (predicted - real)^2 ))
}
```

2.4 Small samples

There are movies which have been rated few times, which means that each rating will have a greater effect on the overall rating. Hence, these movies with small samples might lead to inaccurate predictions. For this reason, a *lambda* parameter has been used in order to penalize these samples, with values from 0 to 3 in intervals of 0.1 to choose the best *lambda* through cross-validation. Using cross-validation, the best lambda will be chosen.

```
# Note: this process might take several minutes
lambdas <- seq(0, 3, 0.1)

rmses <- sapply(lambdas, function(lambda){
  # Mean of all movies
  mo <- mean(edx$rating)

  # Movie bias
  m_b <- edx %>%
    group_by(movieId) %>%
    summarize(m_b = sum(rating - mo)/(n()+lambda))

  # Movie year bias
  m_y_b <- edx %>%
    left_join(m_b, by="movieId") %>%
    group_by(year) %>%
    summarize(m_y_b = sum(rating - mo - m_b)/(n()+lambda))

  # User bias
  u_b <- edx %>%
    left_join(m_b, by="movieId") %>%
    left_join(m_y_b, by="year") %>%
    group_by(userId) %>%
    summarize(u_b = sum(rating - mo - m_b - m_y_b)/(n()+lambda))

  # Rating year bias
  r_y_b <- edx %>%
    left_join(m_b, by="movieId") %>%
    left_join(m_y_b, by="year") %>%
    left_join(u_b, by="userId") %>%
    group_by(rate_year) %>%
    summarize(r_y_b = sum(rating - mo - m_b - m_y_b - u_b)/(n()+lambda))

  #Genre bias
  g_b <- edx %>%
    left_join(m_b, by="movieId") %>%
    left_join(m_y_b, by="year") %>%
    left_join(u_b, by="userId") %>%
    left_join(r_y_b, by="rate_year") %>%
    group_by(genres) %>%
    summarize(g_b = sum(rating - mo - m_b - m_y_b - u_b - r_y_b)/(n()+lambda))

  # Predicted values
  predicted_values <- edx %>%
    left_join(m_b, by = "movieId") %>%
    left_join(m_y_b, by = "year") %>%
    left_join(u_b, by = "userId") %>%
    left_join(r_y_b, by = "rate_year") %>%
    left_join(g_b, by = "genres") %>%
    mutate(predicted_values = mo + m_b + m_y_b + u_b + r_y_b + g_b) %>%
    pull(predicted_values )

 return(RMSE(predicted_values, edx$rating))
})

ggplot(data.frame(lambda = lambdas, rmse = rmses),
       aes(x = lambda, y = rmses)) +   geom_line()
```



```
best_lambda <- lambdas[which.min(rmses)]
```

As one can see in the upper graph, the best lambda is 0.4. Therefore, that is the value which will be used next in order to get the most accurate prediction.

2.5 Predictions with the best lambda

The 5 kinds of bias found are now calculated using the best lambda with the train dataset (*edx*):

```
# Mean of all movies
mo <- mean(edx$rating)

# Movie bias
m_b <- edx %>%
group_by(movieId) %>%
summarize(m_b = sum(rating - mo)/(n()+best_lambda))

# User bias
u_b <- edx %>%
left_join(m_b, by="movieId") %>%
group_by(userId) %>%
summarize(u_b = sum(rating - mo - m_b)/(n()+best_lambda))

# Movie year bias
```

```
m_y_b <- edx %>%
left_join(m_b, by="movieId") %>%
left_join(u_b, by="userId") %>%
group_by(year) %>%
summarize(m_y_b = sum(rating - mo - m_b - u_b)/(n()+best_lambda))

# Rating year bias
r_y_b <- edx %>%
left_join(m_b, by="movieId") %>%
left_join(u_b, by="userId") %>%
left_join(m_y_b, by="year") %>%
group_by(rate_year) %>%
summarize(r_y_b = sum(rating - mo - m_b - u_b - m_y_b)/(n()+best_lambda))

# Genre bias
g_b <- edx %>%
left_join(m_b, by="movieId") %>%
left_join(u_b, by="userId") %>%
left_join(m_y_b, by="year") %>%
left_join(r_y_b, by="rate_year") %>%
group_by(genres) %>%
summarize(g_b = sum(rating - mo - m_b - u_b - m_y_b - r_y_b)/(n()+best_lambda))
```

Next, the test set (*final_holdout_test*) is used. However, this dataset needs the same data cleaning which was applied to the *edx* train set.

```
# Predictions with test set (data cleaning performed)
predicted_values <- final_holdout_test %>%
mutate(timestamp = as_datetime(timestamp)) %>%
mutate(rate_year = year(timestamp)) %>%
extract(title, c("title", "year"), "(.*) \\((\\d{4})\\)") %>%
left_join(m_b, by = "movieId") %>%
left_join(u_b, by = "userId") %>%
left_join(m_y_b, by = "year") %>%
left_join(r_y_b, by = "rate_year") %>%
left_join(g_b, by = "genres") %>%
mutate(pred = mo + m_b + m_y_b + u_b + r_y_b + g_b) %>%
pull(pred)
```

3 Results

In order to get comparative results, the RMSE obtained by using just the mean of all movies has been calculated:

```r
# RMSE from mean of movies
rmse_mean <- RMSE(mo, final_holdout_test$rating)
rmse_mean
```

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

According to these calculations, the RMSE is 1.0612018.

We can now calculate RMSE using the predictions above and the best lambda (0.4):

```r
# RMSE with algorithm and best lambda
rmse_final <- RMSE(predicted_values, final_holdout_test$rating)
rmse_final
```

```
## [1] 0.8644945
```

The RMSE is now 0.8644945, which shows some significant improvement if we compare it to the first RMSE we obtained, from only the mean of movies.

4 Conclusion

As we can see, the first approach of only considering the mean of all movies can be clearly improved through a more complex algorithm. In this project, we have done so by taking into account the different kinds of bias and using the best lambda in order to penalize small samples. As the RMSE results prove, these additions to the algorithm have been very useful in order to increase its accuracy.

In conclusion, we can find that movie ratings are rarely random, at least in this dataset. On the contrary, factors such as the movie date, genres, rating date, former rates by the user and former rates of the movie can help us predict the rating of a movie more accurately.