report

Alexandre Bort

13 June 2019

About me

Welcome to my report. I'm Alexandre, a french student from ISIMA engineering school. My English isn't very good, but I will try to do my best ! Hope you will enjoy reading it!

Plan of the study

I. Introduction

Film industry didn't stopped growing the last decades. The development of the new technologies provides an easy access to films for everyone. Film producer and especially film distributors such as Netflix or even YouTube try to satisfy their users as best as possible. To do this, they use data analysis approaches. For example, that gives them the ability to improve their film suggestion system.

Here, we will try to **predict the interest that a user could have for a film**. I will minimise the `RMSE` (Root Mean Square Error) criteria as seen in the course.

We will build out study on the ***MoviesLens*** dataset provided by the edx team. The dataset is available this link. The dataset is composed by two main dataframes, one called `edx` for training and one called `validation` for testing our model.

In my work, I have tried two approaches. The first one is to predict the rating from a linear regression model. I used the `lm` function after some preprocessing (extracting date, binarizing genre columns…). However, the results weren't as expected. The RMSE was above `1` that is not acceptable. Also, I met **memory issues** that lead me to reduce the size of the dataset (edx to 200 000rows). I explore the `xgboost` library in R that implements a gradient boosted tree algorithm. Finally, I gave up this approach and try to find more efficient one.

So I come back to the lesson basics. I implemented a **linear model** that assigned the rating of a movie to the mean of all ratings minus the mean of the ratings grouped by user and movie Id (code is simpler than long sentences). Then I computed the RMSE.

II. Methods - Analysis

II.1. Data overview

Data:

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 3      1     231      5 838983392         Dumb & Dumber (1994)
## 5      1     316      5 838983392              Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 8      1     356      5 838983653           Forrest Gump (1994)
## 9      1     362      5 838984885       Jungle Book, The (1994)
##                         genres
## 1                Comedy|Romance
## 3                       Comedy
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 8      Comedy|Drama|Romance|War
## 9    Adventure|Children|Romance
```

Let's see some **statistic** about our `edx` and `evaluation` dataframe:

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18120   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35746   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35871   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53609   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000065     Length:9000065
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
summary(validation)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :8.229e+08
##  1st Qu.:18130   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35710   Median : 1831   Median :4.000   Median :1.035e+09
##  Mean   :35857   Mean   : 4123   Mean   :3.513   Mean   :1.033e+09
##  3rd Qu.:53586   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65130   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999989      Length:999989
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

II.2. Pre-processing

After a brief overview of the dataset, we can notice two things: - The `title` column contains a year between parenthesis. We can consider that the interest of customer changes across the year. We can extract them as a new column with the bellow code:

```
df <- head(edx)
# Extract year as a new column
df$year = as.numeric(substr(df$title, nchar(df$title)-4, nchar(df$title)-1))
df$title = substr(df$title, 0 , nchar(df$title)-7)
df
```

```
##   userId movieId rating timestamp                 title
## 1      1     122      5 838985046              Boomerang
## 3      1     231      5 838983392          Dumb & Dumber
## 5      1     316      5 838983392               Stargate
## 6      1     329      5 838983392 Star Trek: Generations
## 8      1     356      5 838983653           Forrest Gump
## 9      1     362      5 838984885       Jungle Book, The
##                           genres year
## 1                Comedy|Romance 1992
## 3                         Comedy 1994
## 5       Action|Adventure|Sci-Fi 1994
## 6 Action|Adventure|Drama|Sci-Fi 1994
## 8      Comedy|Drama|Romance|War 1994
## 9    Adventure|Children|Romance 1994
```

- The `genres` columns is defined in one string having several genres. The matter with this representation is that we do not distinguish the genre inside the text. One idea is to binarize the genre column in new genre columns. Here's the code:

```
# Select all genres (uniques)
list_genre <- unique(separate_rows(data = df["genres"], genres, sep = "\\|"))
nb_genres <- nrow(list_genre)
# Create empty matrix
genres_col <- as.data.frame(matrix(0, ncol = nb_genres, nrow = nrow(df)))
# Set col names
colnames(genres_col) <- as.list(list_genre)[[1]]
# Add columns to edx dataframe
df <- cbind(df, genres_col)

set_genre <- function(row){
  genres <- strsplit(row["genres"], "\\|")[[1]]
  row[genres] <- 1
  return(row)
}
df <- as.data.frame(t(apply(df, MARGIN=1, FUN=set_genre)))

# remove genre columns
df <- df[setdiff(names(df), c("genres"))]
df
```

```
##   userId movieId rating timestamp                 title year Comedy
## 1      1     122      5 838985046              Boomerang 1992      1
## 3      1     231      5 838983392          Dumb & Dumber 1994      1
## 5      1     316      5 838983392               Stargate 1994      0
## 6      1     329      5 838983392 Star Trek: Generations 1994      0
```

```
## 8      1     356      5 838983653           Forrest Gump 1994      1
## 9      1     362      5 838984885       Jungle Book, The 1994      0
##   Romance Action Adventure Sci-Fi Drama War Children
## 1       1      0         0      0     0   0        0
## 3       0      0         0      0     0   0        0
## 5       0      1         1      1     0   0        0
## 6       0      1         1      1     1   0        0
## 8       1      0         0      0     1   1        0
## 9       1      0         1      0     0   0        1
```

II.3. Memory limits

Unfortunately, all the precedent pre-processing is resource consuming. My current PC does not give me the opportunity to perform the task in a reasonable time (still processing `edx` after 15min).

The first point seems to be done in a reasonable time, but not the second. I tried this process on a little subset (200 000 rows). It seems to improve the RMSE compared to non pre-processed data. However, RMSE is bellow than the one got on the whole dataset without pre-processing.

I don't know how to improve this code. If you have any ideas, let me know in comments :-)

The next of the report assumes the data isn't pre-processed. ## II.4. Model Now we know our dataset, we can start building the model. The model I'm fitting is the following:

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

where : $\epsilon_{u,i}$ is a random error term μ is the overall mean of all ratings across all users b_i is the movie specific mean b_u is the user specific mean

We can summarize it in 3 steps: 1. Build the training - test dataset 2. Find the lambda value that minimizes the `RMSE` 3. Apply the model to the dataset

The second step is done over a range of `lambda` candidates. The following plot show the `RMSE` per `lambda` candidate.

```
qplot(lambdas, rmses)
```


We can now find the best `lambda`:

```
lambdas[which.min(rmses)]
```

```
## [1] 5
```

The first and third steps are explained in the R code.

III.Results

Now we have our model, we are ready to evaluate the model. The results are as bellow:

```
## [1] "mu: 3.512"
```

```
## [1] "lambda: 5"
```

```
## [1] "RMSE: 0.86377"
```

b_u and b_i were widely distributed as seen below $\mathrm{b}_u - 1.bb$ $\mathrm{b}_u - 2.bb$

IV. Conclusion

The model implemented relies only on the overall mean, the regularized movie effect, and the regularized user effect. It comes very close to the best possible result.

Further improvements would not only require substantially more effort, but also more computing resources, in particular enough memory for operations on large matrices (even if sparse matrices are used).

Thank you for reading :-) .