Movielens project report

Irene Viola

12/10/2020

1. Introduction

This project is based on the MovieLens assessment of the Capston final part of the Data Science course. Here the MovieLens 10M dataset has been used (http://grouplens.org/dataset/movielens/10m/), that consists in 10,000,000 movies of different genres that have been rated by different users. This leads to a very large variation in ratings for each movie, not only because of the users' preference, but because of the number of ratings given to each movie by different users. The aim of this project is to use machine learning to predict the rating that a user will give to a movie based on a training set and test set, and estimate the accuracy of the algorithm using RMSE.

2. Dataset

Since this dataset is very large, built-in machine learning algorithms using *caret* package are too heavy and use too much resources for laptops to run in a reasonable time. For this reason, a machine learning algorithm for prediction based on a linear model would be the best solution. To evaluate the accuracy by using RMSE has been used the *RMSE()* function from the *DescTools* package. The dataset has been splitted, as in the assessment in the capstone section, in training and test sets with a proportion of 90%-10% respectively. This is completed in the first steps of the script. The training set (called "edx") has 9,000,061 entries with 6 columns. The test set (called "validation") has 999,993 entries and 6 columns. The column information is shown below for the validation dataset. The columns information are shown for validation and edx.

```
glimpse(validation)
```

```
## Rows: 999,993
## Columns: 6
## $ userId    <int> 1, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, ...
## $ movieId   <dbl> 588, 1210, 1544, 151, 1288, 5299, 380, 435, 480, 477, 508...
## $ rating    <dbl> 5.0, 4.0, 3.0, 4.5, 3.0, 3.0, 3.0, 3.0, 5.0, 3.0, 3.0, 4....
## $ timestamp <int> 838983339, 868245644, 868245920, 1133571026, 1133571035, ...
## $ title     <chr> "Aladdin (1992)", "Star Wars: Episode VI - Return of the ...
## $ genres    <chr> "Adventure|Animation|Children|Comedy|Musical", "Action|Ad...
```

```
glimpse(edx)
```

```
## Rows: 9,000,061
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 231, 292, 316, 329, 355, 356, 362, 364, 370, 37...
```

```
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983392, 838983421, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Dumb & Dumber (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Comedy", "Act...
```

3. Models construction and development

3.1 Start algorithm

The simplest model to consider is to perform the average across all user and movie:

$$Y_{u,m} = \mu \tag{1}$$

Here $Y_{u,m}$ is the predicted rating of the user u and movie m and μ is the average rating of all the entries, resulting in 3.512 (`mean(edx$rating)`).

```
mu <- mean(edx$rating)
RMSE(validation$rating, mu)
```

```
## [1] 1.060651
```

3.2 Independent error term correction for Movie and User

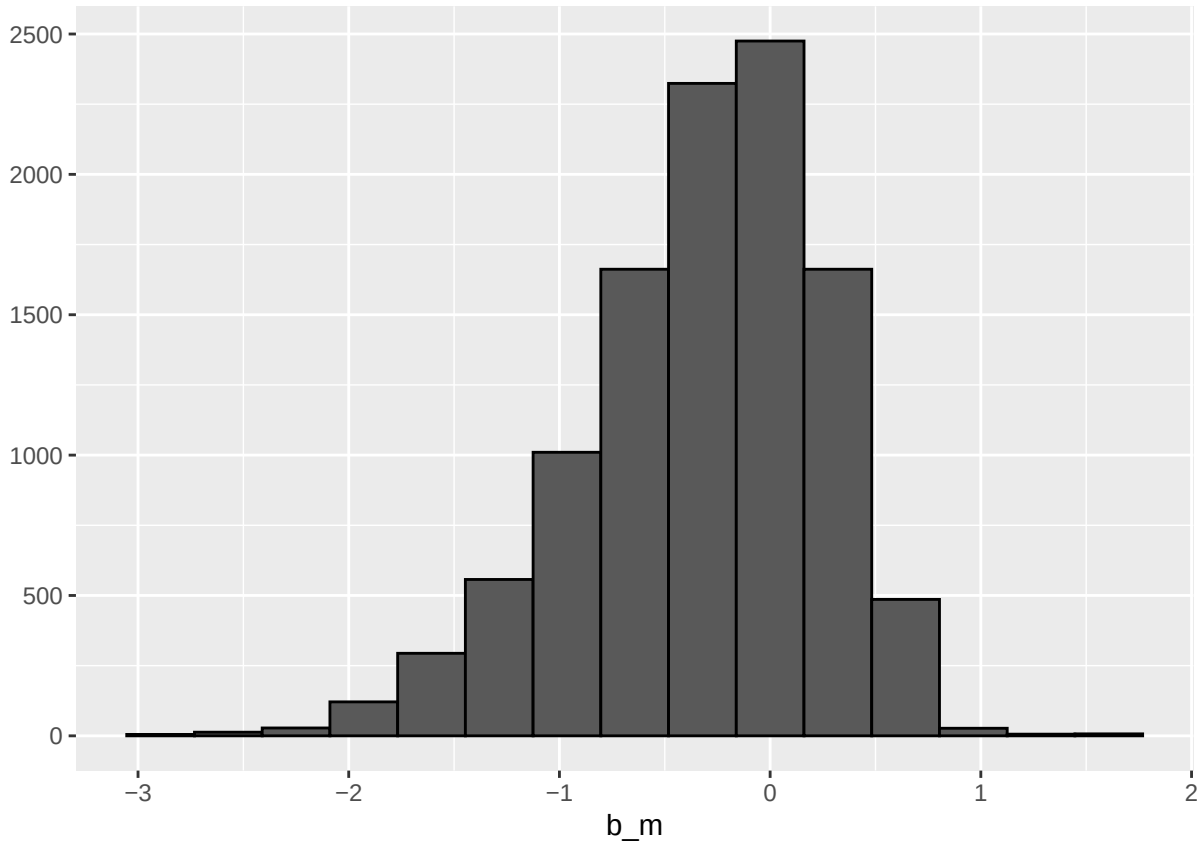
In this first model there are few errors that haven't been taken into consideration. In order to improve the model independent error term $b_{u,m}$ must be considered. These express rating differences for users and movies since the singular taste can affect the number of ratings and the rating itself for each movie, plus popular movies have been rated more respect to less known ones. First consider the movie bias term b_m. This term averages the rankings for any movie m to smooth the "popular/niche" effect on the movies. The new model is:

$$Y_{u,m} = \mu + b_m \tag{2}$$

```
# First calculate the movie effect b_m
b_m <- edx %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu), .groups ='drop')

# predict ratings with mu and b_m
p_ratings_M <- validation %>%
  left_join(b_m, by='movieId') %>%
  mutate(pred = mu + b_m) %>%
  pull(pred)

#Then calculate the RMSE for the p_rating_M
RMSE_M<- RMSE(validation$rating, p_ratings_M)
```



Second step to improve the model is to consider the user bias term b_u. Adding this term, the "love/hate" effect due to extreme rating and preferences of users is going to be minimidez. The updated model is:

$$Y_{u,m} = \mu + b_m + b_u \tag{3}$$

```
#First calculate the user effect b_u
b_u <- edx %>%
       left_join(b_m, by='movieId') %>%
       group_by(userId) %>%
       summarize(b_u = mean(rating - mu - b_m), .groups ='drop')
#Make a new prediction considering the user effect and the movie effect
# predict new ratings with movie and user bias
p_ratings_M_U <- validation %>%
                 left_join(b_m, by='movieId') %>%
                 left_join(b_u, by='userId') %>%
                 mutate(pred = mu + b_m + b_u) %>%
                 pull(pred)

#Then calculate RMSE for user movie effect ratings prediction
RMSE_M_U<- RMSE(validation$rating, p_ratings_M_U)
```

3.3 Regularization of the method for movie and user

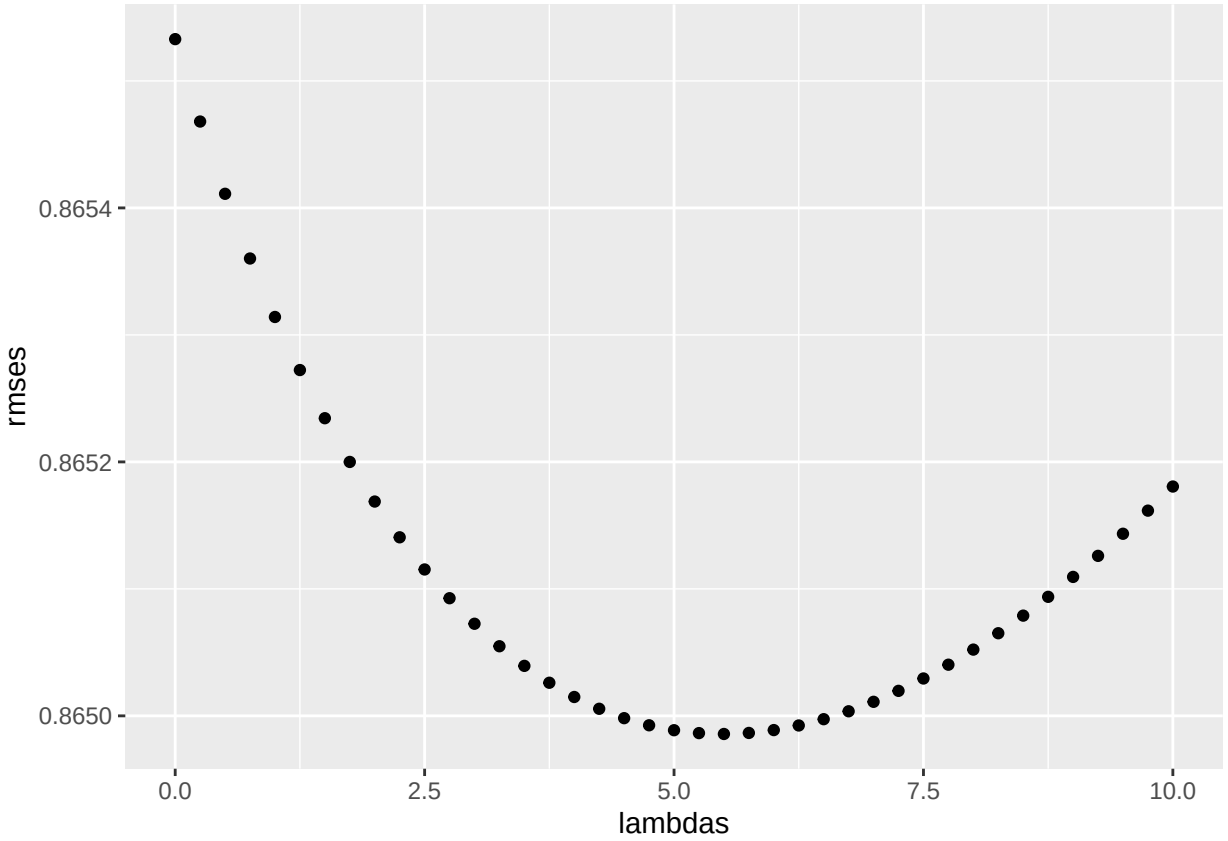
Regularization reduces the effect of large errors in the predictions. Regularization also correct incorrect estimates on small sample sizes. Here this has been used to reduce the effect that extreme rating will have

on b_m term and to reduce anomalies that affect the term b_u due to the ratings of users. This method, in cases where is not possible to predict an interval, acts as confidence intervals, like in this case where the prediction is a single number. Considering the regularization, the update model became:

$$\frac{1}{N}\sum_{u,m}(Y_{u,m} - \mu - b_m - b_u)^2 + \lambda(\sum_m b_m^2 + \sum_u b_u^2) \tag{4}$$

The first term of the equation is previous LSE, the last is penalty with large bias term. The equation in this way will minimize the biases by using λ. To find the value of λ that minimize the bias, has been tested the sequence `lambda <- seq(from=0, to=10, by=0.25)`.

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



The lambdas plot shows the RMSE associated to each λ of the sequence, to find out which is the best λ:

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

```
## [1] 5.5
```

3.4 Regularized model on user and movie bias

Using the best λ, the final model with regularization is:

```
#best lambda
lambda <- lambdas[which.min(rmses)]
#Movie effect regularised
b_m <- edx %>%
       group_by(movieId) %>%
       summarize(b_m = sum(rating - mu)/(n()+lambda), .groups ='drop')
#User effect regularised
b_u <- edx %>%
       left_join(b_m, by="movieId") %>%
       group_by(userId) %>%
       summarize(b_u = sum(rating - b_m - mu)/(n()+lambda), .groups ='drop')
#Rating prediction on validation set using regularised terms
pr_ratings_U_M2 <- validation %>%
                     left_join(b_m, by = "movieId") %>%
                     left_join(b_u, by = "userId") %>%
                     mutate(pred = mu + b_m + b_u) %>%
                     pull(pred)
# RMSE predictions
RMSE_U_M2 <- RMSE(validation$rating, pr_ratings_U_M2)
```

3.5 Genre correction

Another bias that can be considered to make the prediction more accurate is the independent error given by the genre. By adding to the model the term b_g, the effect due to the genre preference of the user will be taken in account and minimized.

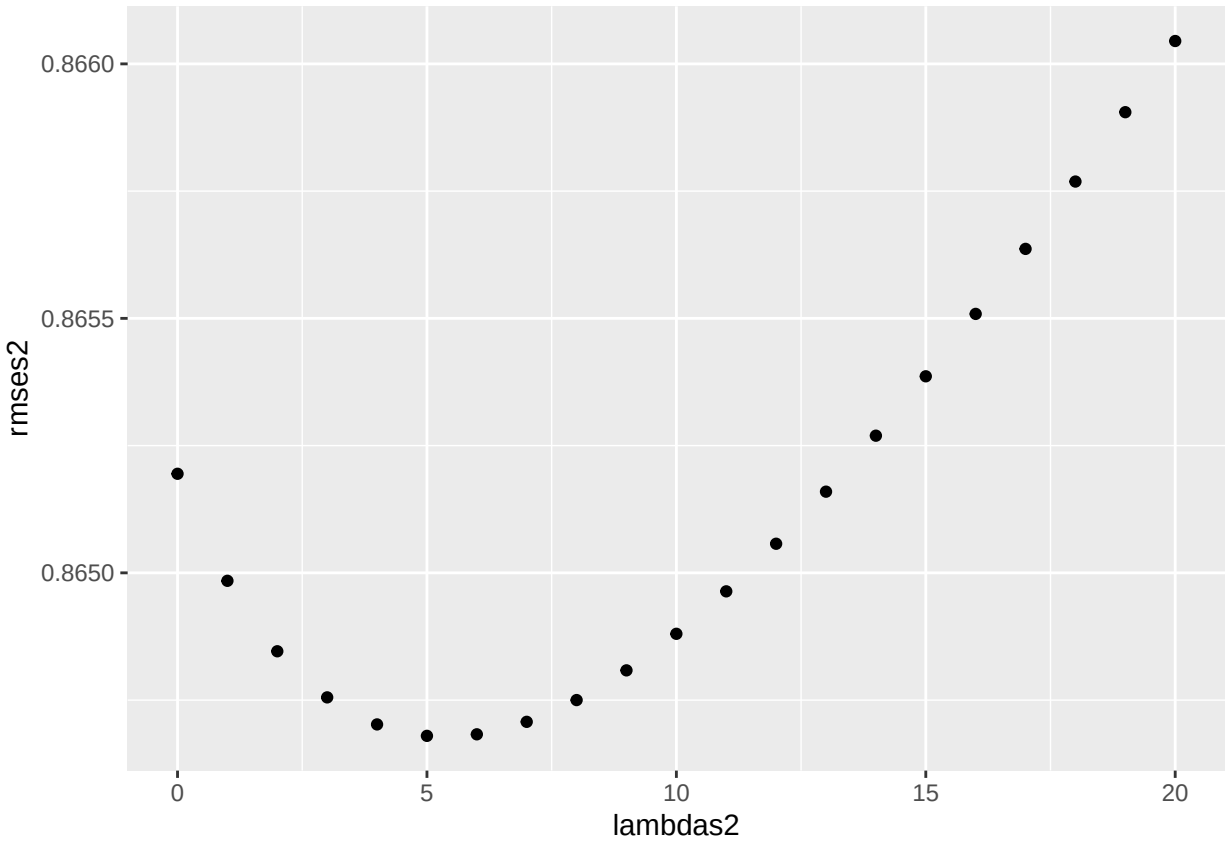
$$Y_{u,m,g} = \mu + b_m + b_u + b_g \tag{5}$$

Adding a new bias, result in a new regularization equation:

$$\frac{1}{N}\sum_{u,m,g}(Y_{u,m,g} - \mu - b_m - b_u - b_g)^2 + \lambda(\sum_m b_m^2 + \sum_u b_u^2 + \sum_g b_g^2) \tag{6}$$

The first term of the equation is previous LSE, the last is penalty with large bias term. The equation in this way will minimize the biases by using λ. However, adding a new bias term, a new λ is needed.

```
qplot(lambdas2, rmses2)
```



The plot here is specular to the previous one and the best λ that minimize the bias is different.

```
lambdas[which.min(rmses2)]
```

```
## [1] 1.25
```

3.6 Regularized model on movie, user and genre bias

Using the best λ, the final model with regularization for all the three bias is:

```
#best lambda
lambda2 <- lambdas[which.min(rmses)]
#Regularised movie effect using lambda2
b_m2 <- edx %>%
        group_by(movieId) %>%
        summarize(b_m = sum(rating - mu)/(n()+lambda2), n_i = n(), .groups ='drop')
#Regularised user effect using lambda2
b_u2 <- edx %>%
        left_join(b_m2, by='movieId') %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - mu - b_m)/(n()+lambda2), n_u = n(), .groups ='drop')
#Regularised genre effect using lambda2
b_g2 <- edx %>%
        left_join(b_m2, by='movieId') %>%
        left_join(b_u2, by='userId') %>%
        group_by(genres) %>%
```

```
        summarize(b_g = sum(rating - mu - b_m - b_u)/(n()+lambda2), n_g = n(), .groups ='drop')
#Rating prediction on validation set using regularised terms with lambda2
pr_ratings_U_M_G2 <- validation %>%
                  left_join(b_m2, by='movieId') %>%
                  left_join(b_u2, by='userId') %>%
                  left_join(b_g2, by = 'genres') %>%
                  mutate(pred = mu + b_m + b_u + b_g) %>%
                  pull(pred)
# RMSE predictions FINAL
RMSE_U_M_G2 <- RMSE(validation$rating,pr_ratings_U_M_G2)
```

4. Conclusions

It is possible to see clear improvements to the RMSE as the model was corrected with movie, user, genre bias terms and regularization on λ.

Models	RMSE
Average	1.06095
Movie effect	0.94502
Movie + user effects	0.86440
Regularized movie + user effect	0.86381
Regularized movie + user + genre effect	0.86379

The model considering the regularization of movies, users and genre bias is the more efficient one based on RMSE. However, is possible to see that the difference between the two regularized model is very low, meaning that the preferences in genre of the singular users doesn't give such a large bias as expected. This model is very efficient to run on every kind of laptop that supports R packages and the MovieLens database or other large database. Since it is a linear model it permits to successfully predict movie rating on such a large database without an excessive strain on the computer. This prediction could also be done by using matrices and vectors, indeed the stress on the computer would be quite high and could take a lot of time and resources. Another approach could be to create a small matrix from the edx partition, in order to have the same size of the validation set; in this case the stress on the machine would be less than using the entire edx set, but it could lead to overestimation or underestimation problems depending on the movies that are included, the users and the genres.