Recommendation System Report

Lvesselova

14 Jun 2019

Introduction

This is report relative to the creation of a Recommendation System based on the Movielens dataset. The purpose of the System is to predict movie ratings a particular user will give a specific movie. The System has been created following the main guidelines exposed in the Data Science : Machine Learning course.

The libraries used for the creation process are

```r
library(tidyverse)
library(dslabs)
library(dplyr)
library(purrr)
library(ggplot2)
library(caret)
```

The data has been loaded according to the instructions given in the Project Overview and contains 10m records relative to movies ratings. The dataset has been split into edx and validation subsets. The validation subset contains 10% of the whole dataset, whereas the edx subset represents 90% of the initial ratings data.

We will train the machine learning algorithm using edx subset and use the validation set to predict movie ratings.

RMSE function

To evaluate the algorithm, we will use the RMSE (residual mean squared error). This value should be less than 1 for the algorithm to be good.

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

The estimate that minimizes the RMSE is the least squares estimate of mu, and in our case, is the average of all ratings :

```r
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

Prediction based on the average

Let’s predict the ratings with just the average mu :

```r
first_rmse <- RMSE(validation$rating, mu)
first_rmse
```

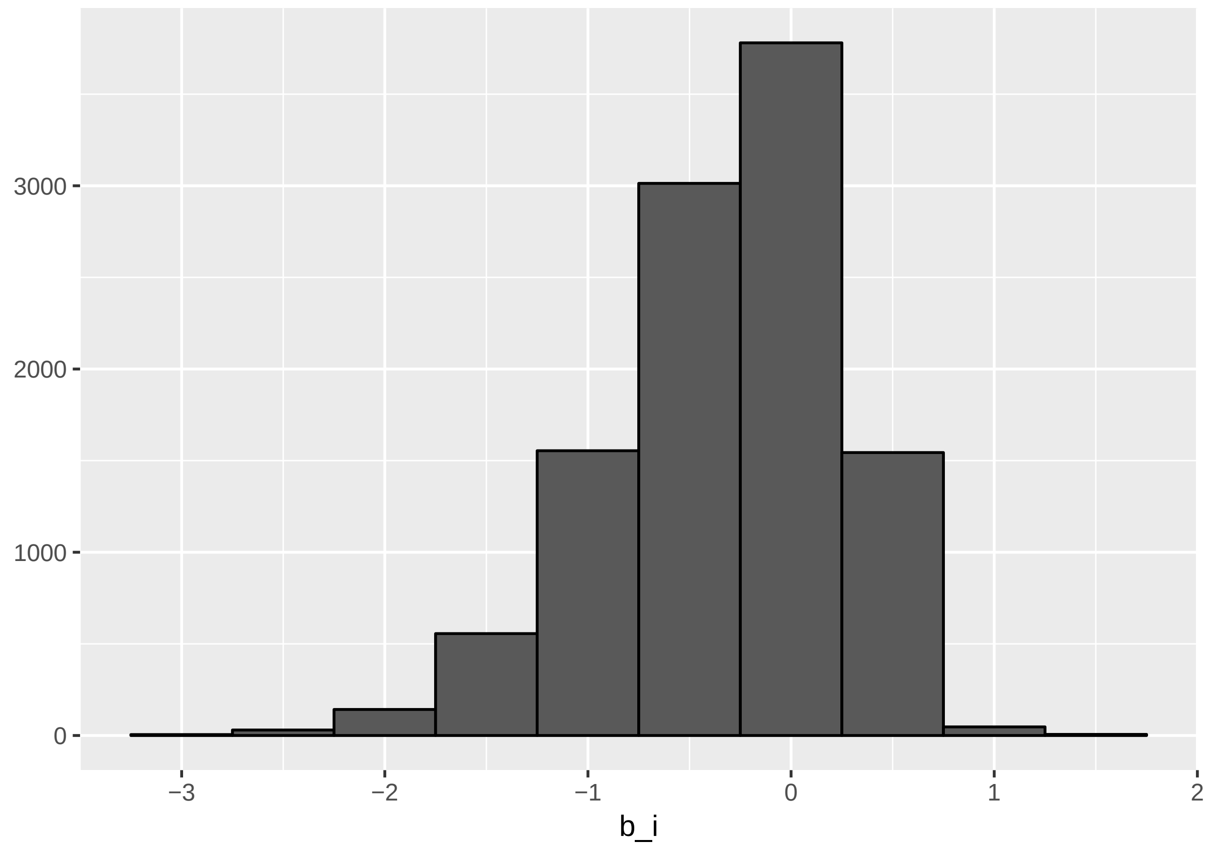
```
## [1] 1.061202
```

The first RMSE is not very good, it is greater than 1. We wil try to improve our model. For that we will consider some data caracteristics that affect the quality of the model and we will integrate them in our final model.

Movie Effects

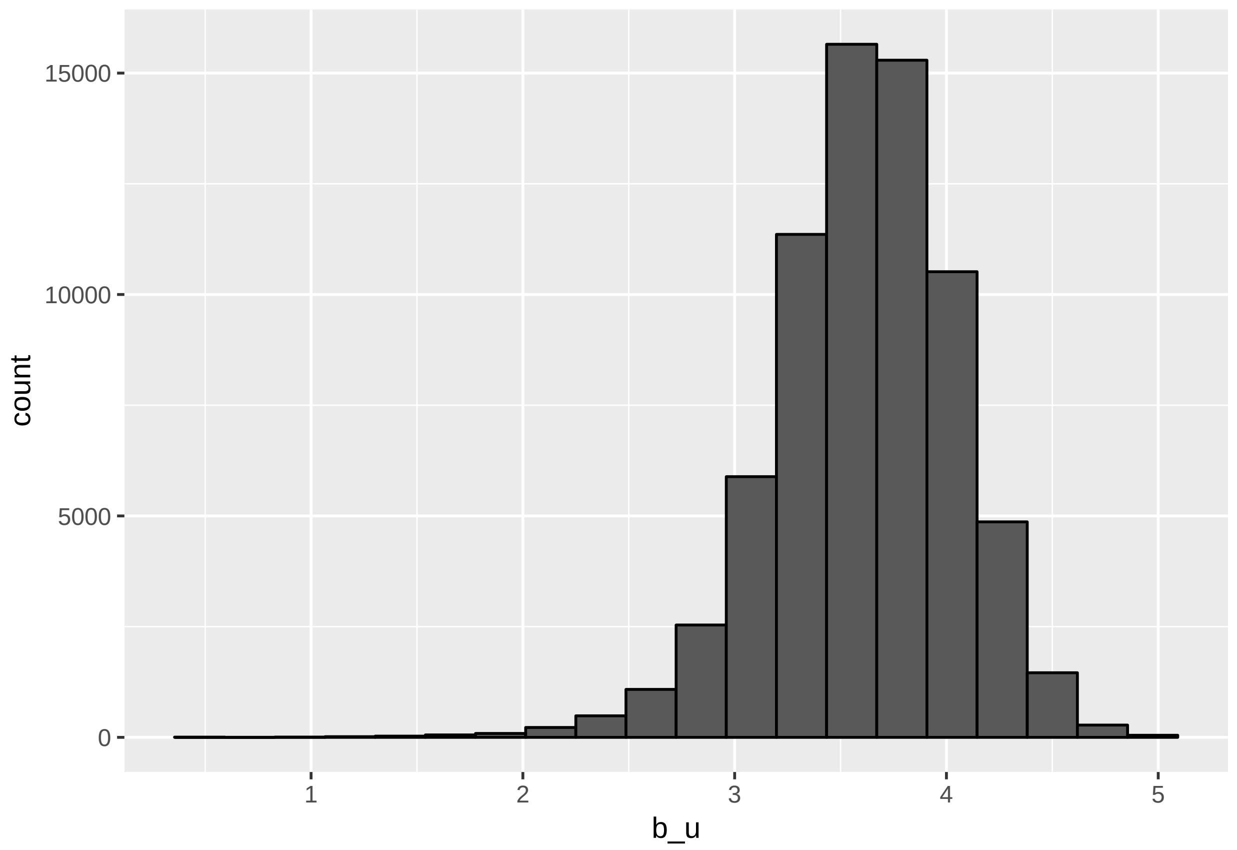
First, we observe that some movies are rated higher than others. We can calculate the movie effect b_i like this :

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

movie_avgs %>% qplot(b_i, geom="histogram", bins=10, data=., color=I("black"))
```



User Effects

Second, we observe that some users are passive, others are very active and rate almost every movie, some are difficult and give not so high rates, others easily give high rates :



The estimation of the user effect, b_u, can be computed like this :

```
user_avgs <- edx %>%
left_join(movie_avgs, by="movieId") %>%
group_by(userId) %>%
summarise(b_u = mean(rating - mu - b_i))
```

Regularization

In order to improve our model, we have to make a regularization to avoid mistakes in the model due to the small sample size effect. These mistakes are due to a small number of users, in most cases just 1, rating a movie. This increase uncertainty because of larger estimates of b_i.

Here are the lists of 10 “worst”" and 10 “best”" movies. Some of them have very few ratings and, therefore, large b_i, e.g. ‘Hellhounds on My Trail’ or ‘Hi-Line, The’ :

```
## Joining, by = "movieId"
```

```
## # A tibble: 10 x 3
##    title                                                       b_i     n
##    <chr>                                                     <dbl> <int>
##  1 Hellhounds on My Trail (1999)                              1.49     1
##  2 Satan's Tango (Sátántangó) (1994)                          1.49     2
##  3 Shadows of Forgotten Ancestors (1964)                      1.49     1
##  4 Fighting Elegy (Kenka erejii) (1966)                       1.49     1
##  5 Sun Alley (Sonnenallee) (1999)                             1.49     1
##  6 Blue Light, The (Das Blaue Licht) (1932)                   1.49     1
##  7 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko~  1.24     4
##  8 Human Condition II, The (Ningen no joken II) (1959)           1.24     4
##  9 Human Condition III, The (Ningen no joken III) (1961)         1.24     4
## 10 Constantine's Sword (2007)                                    1.24     2
```

```
## Joining, by = "movieId"
```

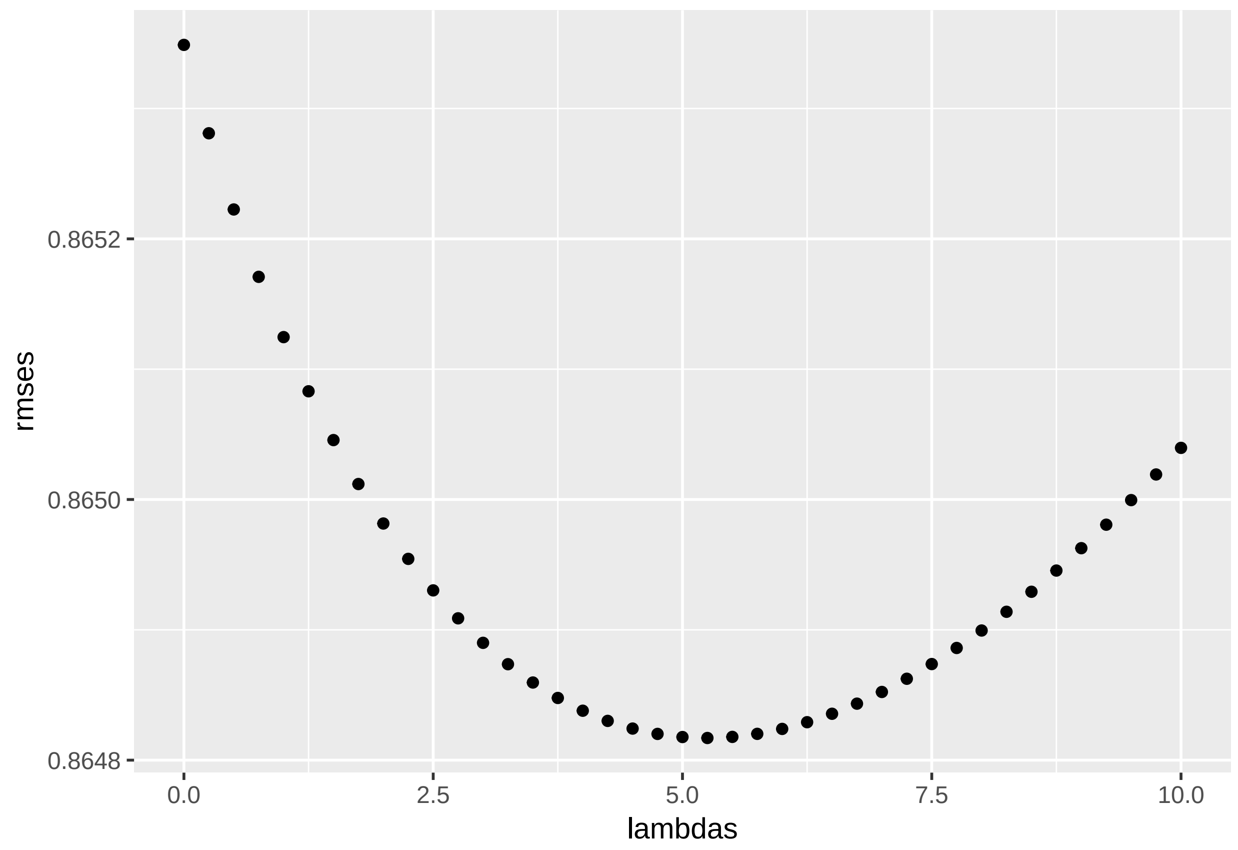
```
## # A tibble: 10 x 3
##    title                                     b_i     n
##    <chr>                                   <dbl> <int>
##  1 Besotted (2001)                         -3.01     2
##  2 Hi-Line, The (1999)                     -3.01     1
##  3 Accused (Anklaget) (2005)               -3.01     1
##  4 Confessions of a Superhero (2007)       -3.01     1
##  5 War of the Worlds 2: The Next Wave (2008) -3.01     2
##  6 SuperBabies: Baby Geniuses 2 (2004)     -2.72    56
##  7 Hip Hop Witch, Da (2000)                -2.69    14
##  8 Disaster Movie (2008)                   -2.65    32
##  9 From Justin to Kelly (2003)             -2.61   199
## 10 Criminals (1996)                        -2.51     2
```

To penalize large estimates, we use regularization. In order to regularize our estimates, we will use a penalty parameter lambda, that minimizes the uncertainty : when the sample size, n_i, is large, the penalty lamda is ignored, when n_i is small, the estimate b_i is shrunk to 0. We can use regularization for the estimate user effect b_u as well.

Finding the optimal lambdas

Here we are using the cross validation to pick the optimal lambda. We also regularize the both estimates for movie and user effects, b_i and b_u :

```r
lambdas <- seq(0,10,0.25)
mu <- mean(edx$rating)

rmses <- sapply(lambdas, function(l){

    mu <- mean(edx$rating)

    b_i <- edx %>%
        group_by(movieId) %>%
        summarise(b_i = sum(rating-mu)/(n()+l))

    b_u <- edx %>%
        left_join(b_i,by="movieId") %>%
        group_by(userId) %>%
        summarise(b_u=sum(rating-b_i-mu)/(n()+l))

predicted_ratings <-
validation %>%
left_join(b_i,by="movieId") %>%
left_join(b_u,by="userId") %>%
mutate(pred=mu+b_i+b_u) %>%
pull(pred)

return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```



We can see on the curve that for the full model the optimal lambda parameter is 5.25. Now, taking into account the penalty parameter lambda = 5.25, we will select the first 10 best and 10 worst movies with their number of ratings :

```
## # A tibble: 10 x 3
##    best_movies                                        b_i     n
##    <chr>                                            <dbl> <int>
##  1 Shawshank Redemption, The (1994)                 0.942 28015
##  2 Godfather, The (1972)                            0.903 17747
##  3 Usual Suspects, The (1995)                       0.853 21648
##  4 Schindler's List (1993)                          0.851 23193
##  5 Casablanca (1942)                                0.808 11232
##  6 Rear Window (1954)                               0.806  7935
##  7 Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)    0.802  2922
##  8 Third Man, The (1949)                            0.798  2967
##  9 Double Indemnity (1944)                          0.796  2154
## 10 Paths of Glory (1957)                            0.794  1571
```

```
## # A tibble: 10 x 3
##    worst_movies                                             b_i     n
##    <chr>                                                  <dbl> <int>
##  1 From Justin to Kelly (2003)                            -2.54   199
##  2 SuperBabies: Baby Geniuses 2 (2004)                    -2.48    56
##  3 Pokémon Heroes (2003)                                   -2.39   137
##  4 Glitter (2001)                                          -2.30   339
##  5 Gigli (2003)                                            -2.28   313
##  6 Disaster Movie (2008)                                   -2.28    32
##  7 Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie) (2002) -2.28   202
##  8 Barney's Great Adventure (1998)                         -2.27   208
##  9 Carnosaur 3: Primal Species (1996)                      -2.25    68
## 10 Son of the Mask (2005)                                  -2.14   165
```

In the listings above we can see large n sizes for the best movies. So, it looks better with the penalty lambda is applied.

Final Results

The final script calculating ratings prediction is like this :

```r
lambda <- 5.25

mu <- mean(edx$rating)

b_i <- edx %>%
         group_by(movieId) %>%
         summarise(b_i = sum(rating-mu)/(n()+lambda))

b_u <- edx %>%
         left_join(b_i,by="movieId") %>%
         group_by(userId) %>%
         summarise(b_u=sum(rating-b_i-mu)/(n()+lambda))

 predicted_ratings <-
 validation %>%
 left_join(b_i,by="movieId") %>%
 left_join(b_u,by="userId") %>%
 mutate(pred=mu+b_i+b_u) %>%
 pull(pred)


rmse_result <- RMSE(predicted_ratings, validation$rating)
rmse_result
```

```
## [1] 0.864817
```

Conclusion

For this machine learning exercice we have used a 10M ratings data subset that we split into two subsets : edx and validation. We analyzed the general properties of the data and noticed some particularities such as movie and user effects that impact the final rating prediction. We included these particularities in the model to improve its performance. Also, in order to decrease the uncertainty due to the presence of the small sample sizes in the data set, we used the regularization to constrain the variability of the small sizes effect. All these actions significantly improved the final prediction and reduced the standard error RMSE. Our final model has a very good RMSE : 0.864817