The Movie Lens project

Simon Cox

24-12-2019

Before we start

Dear reader, thank you for taking time to read my work. I know that if you also participate in the data science course, then it is not out of free will, but still. I tried to write it all down as structured as possible to make it easy for you to read, enjoy!

Introduction

This document contains the Movie Lens project which is part of the **edX Data Science Professional Programm** capstone project. The goal of this project is to create a movie recommendation system based on the MovieLens dataset. This recommendation system will consist of a trained machine learning algorithm. The quality of this recommendation system will be assessed by means of the residual mean squared error or RMSE. The deliverables for this project will be this R markdown document in .Rmd and .pdf format and the complete R script in .R format. All files are available in this github repository: https://github.com/SimonBCox/data-science-capstone.git

The MovieLens dataset is split in a training set (`edx`) and a validation set (`validation`). The `edx` dataset will be used to train the recommendation system and the `validation` set will be used to calculate the final RMSE. Both sets are generated using a script provided by the course instructors. This provided script will not be shown in this document.

Method

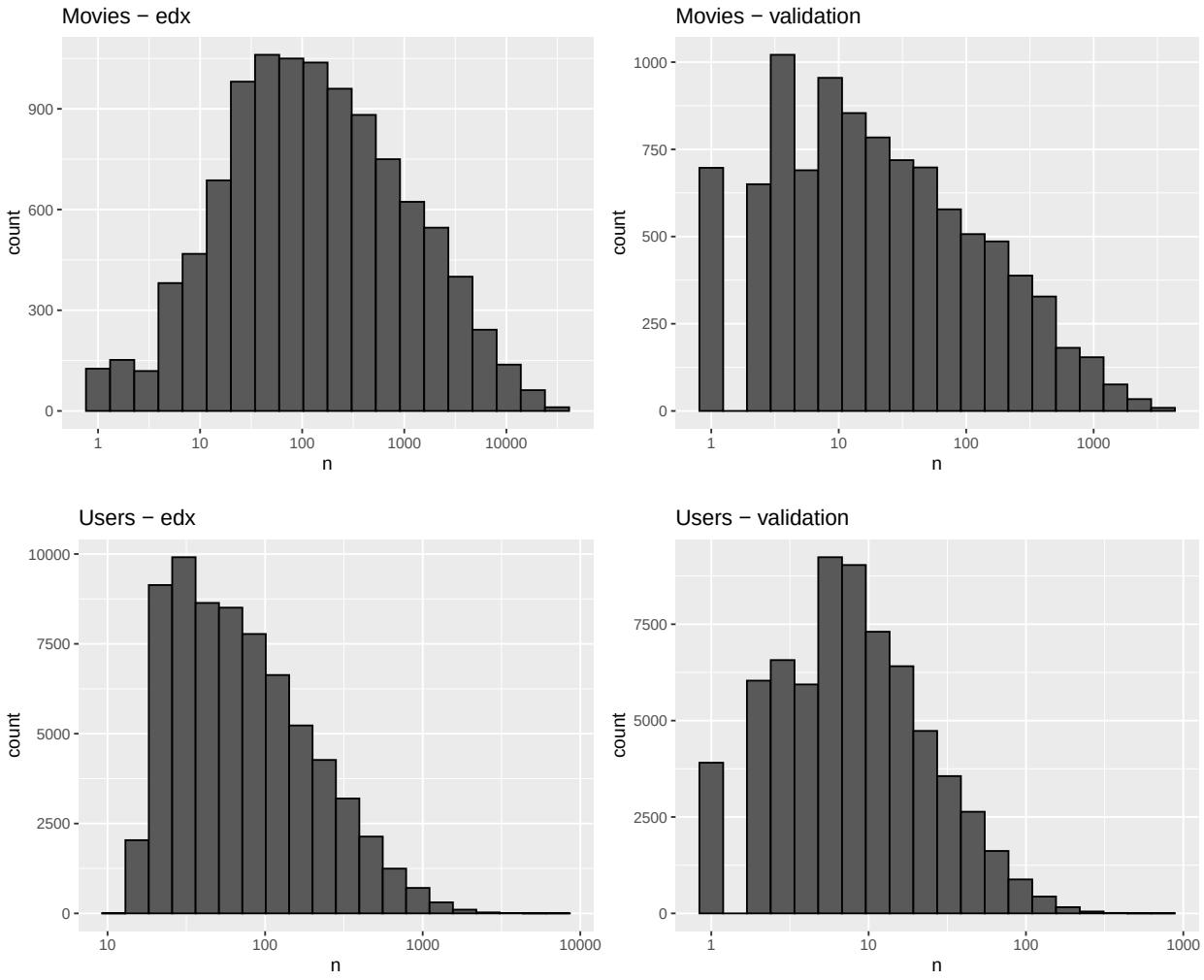
Let us start with a short description of the `edx` and `validation` datasets. Both sets are of class data.frame. The `edx` set contains 9000055 rows and 6 columns. The table below shows the column names and the number of unique entries for each column of the `edx` set.

```
##                userId movieId rating timestamp title genres
## unique entries  69878   10677     10   6519590 10676    797
```

The `validation` dataset is about 9 times smaller than the `edx` dataset and contains 999999 rows and also 6 columns. Below the number of unique entries for each column are shown.

```
##                userId movieId rating timestamp title genres
## unique entries  68534    9809     10    944830  9808    773
```

On the next page the distributions for the movieId column and userId column are presented for both sets. The images show for each unique entry in the movieId column how many times it has been rated and for each unique entry in the userId column how many ratings it has given. We can see a different distribution for the moviesId's in the `edx` and `validation` set. The `validation` set seems to have more movieId's with just a few ratings. With regard to the usersId's both sets seem to have a similar distribution.



The `edx` dataset is very large which results in a memory error when using the `lm()` function. To avoid memory errors, the procedure and techniques are used as described in the data science book Chapter 33 Large datasets.

Similar to the approach in Chapter 33, the recommendation system will be build step-by-step. For this project the recommendation system wil be build in four consecutive steps:

1. Movie effect model
2. Movie and user effect model
3. Movie, user and genres effect model
4. Regularized movie, user and genre effect model

With each (sub)model we will calculate the RMSE and present it in order to show the difference each step makes.
The RMSE is calculated using the `calculate_RMSE` function, which requires `true_ratings` and `predicted_ratings` as input.

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

1. Movie effect model

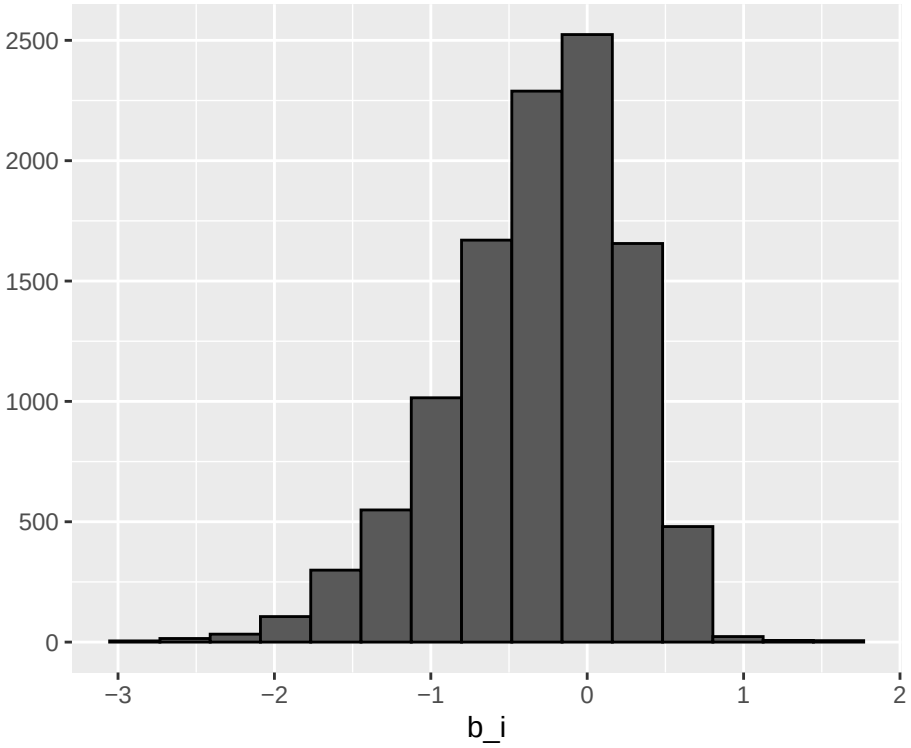
Before we start including any effects we need to calculate the average rating (`mu`). This value of `mu` will be used in all four models.

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

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

Now we can include the movie effect in the recommendation system by a movie specific bias (`b_i`). `b_i` is calculated for each unique movie as the average of each rating minus `mu` and is stored in `movie_avgs`. The distribution of `b_i` is substantial as shown in the figure below.

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



Now the movie effect is known we can predict the ratings for the `validation` set. The predicted ratings will be stored in `predicted_ratings`, which server as input for the `calculate_RMSE` function.

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>% # add column b_i
  mutate(pred = mu + b_i) %>%
  pull(pred)

calculate_RMSE(predicted_ratings, validation$rating)
```

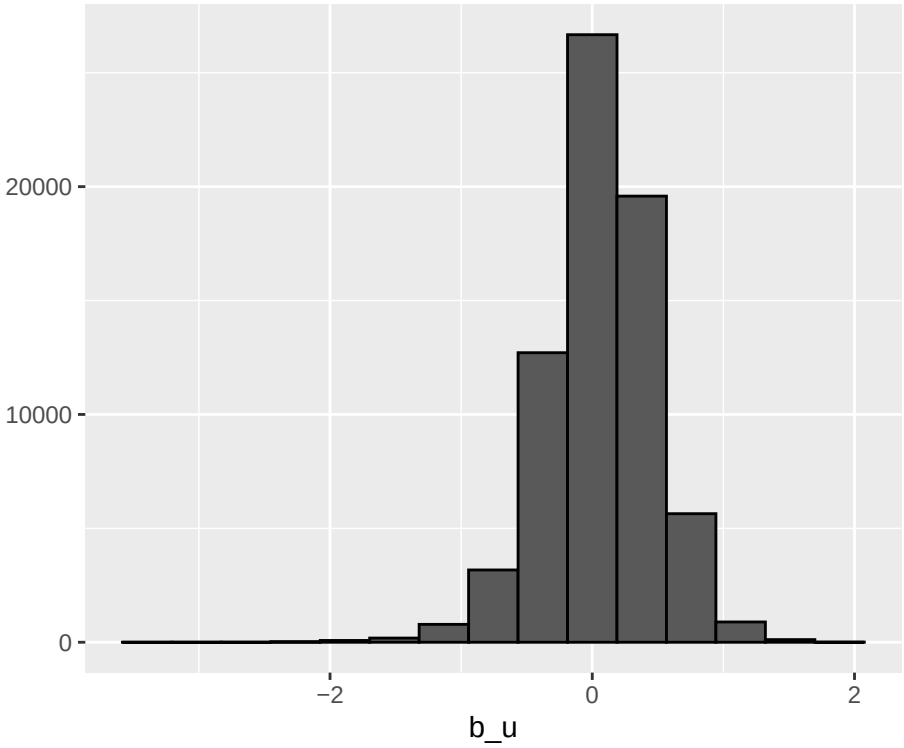
```
## [1] 0.9439087
```

As we can see the RMSE is higher then 0.9, which is too high. So, let us continue and improve the model.

2. Movie and user effect model

In this second step the user effect is added to the recommendation system by means of a user specific bias (`b_u`). The user specific bias is calculated in a similar way as `b_i` and will be stored in `user_avgs`. The distribution of `b_u` is also substantial as shown in the figure below.

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%  # add column b_i
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```



With the movie and user effects now known we can predict the ratings for the `validation` set and run the `calculate_RMSE` function.

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%  # add column b_i
  left_join(user_avgs, by='userId') %>%    # add column b_u
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

calculate_RMSE(predicted_ratings, validation$rating)
```

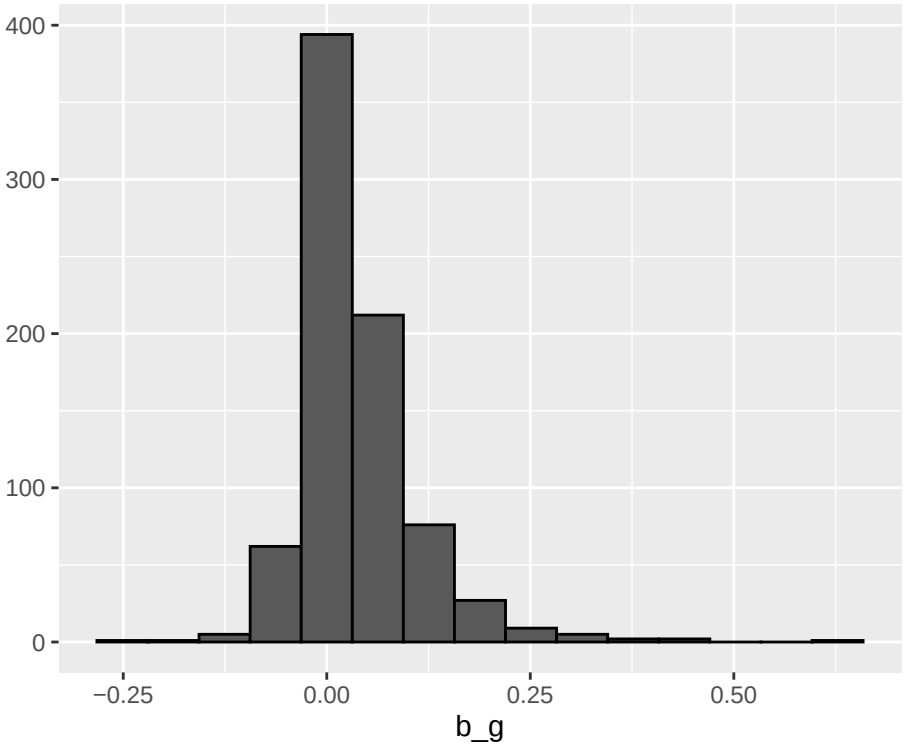
```
## [1] 0.8653488
```

As we can see the RMSE decreases to below 0.9, which is good but we have two more steps that hopefully improve the model a bit more.

3. Movie, user and genres effect model

In this thrid step we add the genres effect to the recommendation system by means of a genres specific bias (`b_g`). Again, the calculation procedure for `b_g` is similar to that of `b_i` and `b_u` and the values will be stored in `genres_avgs`. The distribution of `b_g` is less substantial than we have seen with `b_i` and `b_u`, see the figure below.

```
genres_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%  # add column b_i
  left_join(user_avgs, by='userId') %>%    # add column b_u
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
```



Now we have all the ingredients to predict the ratings for the `validation` set based on the movie, user and genre effects.

```
## [1] 0.8649469
```

Again the RMSE decreases. Now below 0.865. Let us continue to the final step to see how low we can get.

4. Regularized movie, user and genre effect model

In this final step the model is regularized with a penalty term (λ). This is done to make the model more robust and reduce the influence of outliers. To do this we first create the function `calculate_RMUGEM`, which takes λ as input and calculates the RMSE using the regularized movie, user and gerne effects model or RMUGEM.

```
calculate_RMUGEM <- function(l){
    b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

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

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

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

  return(calculate_RMSE(predicted_ratings, validation$rating))
}
```

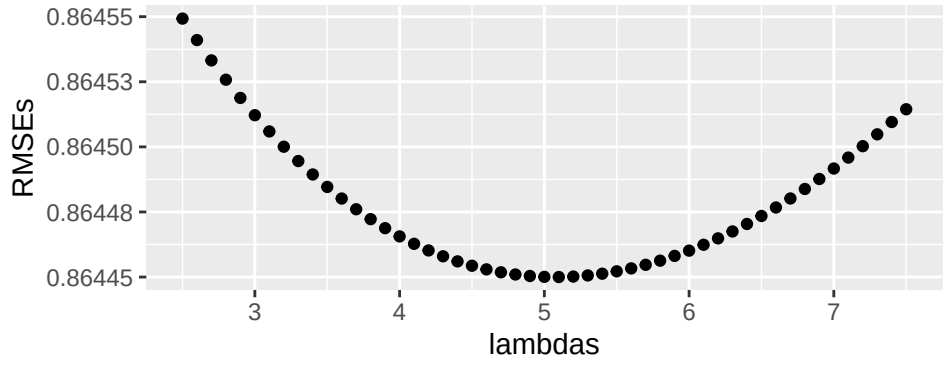
To determine the best value for λ we create an array containing 50 values for λ and strore them in `lambdas`.

```
lambdas <- seq(2.5, 7.5, 0.1)
```

Then we use `sapply` to perform the `calculate_RMUGEM` for each λ. The resulting RMSE's are stored in `RMSEs`.

```
RMSEs <- sapply(lambdas, calculate_RMUGEM)
```

We can plot the RMSE's against the lambdas and see that a lambda of $\lambda = 5.1$ provides the minimum RMSE of 0.86445.



Results

In the table below the RMSE's for all four consecutive models are presented. It shows that for each effect that we add the RMSE reduces. And finally, an even smaller RMSE is reached by regularizing the effects. A benefit of this approach is that the calculation time for this model is rather low. Downloading the MovieLens dataset almost consumes the most amount of time.

Method	RMSE
Movie effect model	0.94391
Movie and user effect model	0.86535
Movie, user and genre effect model	0.86495
Regularized movie, user and genre effect model	0.86445

Conclusion

A movie recommendation model was created based on the MovieLens dataset. The final model incorporates regularized movie, user and genre effects and is trained with the `edx` dataset. The quality of the model is assessed by predicting ratings for the `validation` set and calculating the RMSE. With the final model a RMSE of 0.86445 is achieved.

Although the model is built step-by-step, it only uses one type of machine learning algorithm. This is mainly due to the fact that the `edx` dataset was that large that using the `train()` function resulted in a memory error. A smaller partition of the `edx` dataset has been created to enable the use of the `train()` function. However, the calculation times was still disproportionately high in comparison to the results. Therefore, this path has been abandoned.

Concerning future work, I don't think I will continue working on the MovieLens dataset. But I will definetly continue working on other datasets which are more related to my field of expertice, which is structural engineering.

Concluding remark

Thank you for reading this report. I hope you liked it. Have a nice day!

Kind regards,

Simon Cox