MovieLens Project

Rafael Pereira Lopez

04/01/2021

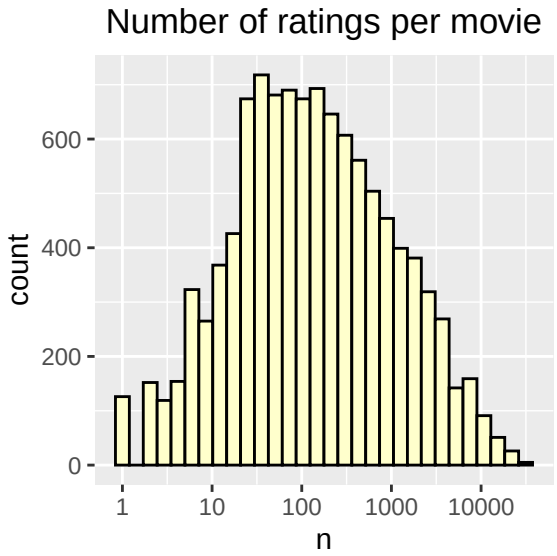
Introduction
A recommendation system uses previously available data, in this case movie ratings, to predict ratings of movies that the user hasn't seen yet, recommend movies that the user could like (good rating predicted) and avoid movies that the user could dislike (bad rating predicted).
To achieve this, machine learning techniques are used. For this the MovieLens data set and specifically the 10M version is used (it has 10 millions ratings, 100,000 tag applications applied to 10,000 movies by 72,000 users).
The predictors used by the machine learning algorithm are the movies(each movie has a set of ratings),the users(each user gives different movies different ratings) and the genres of movies (comedy,drama,action,etc.).After that, it uses regularization to penalize large estimates and generate better predictions. The desired RMSE is 0.86490.

Analysis
First the data is loaded from the MovieLens url. After that it's split in two new data sets, the **edx** data set (it will be used for training the algorithm) and the **validation** data set (which will be used to test the final algorithm's performance). The validation data set cannot be used to train the algorithm,so the edx data set is split in two new data sets, the **edx_train** and the **edx_test** (for cross validation, it will be 10% of the edx data set).
Load data:

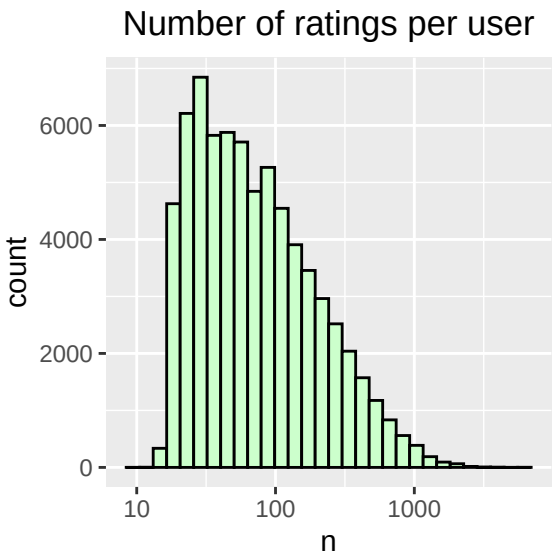
```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 4.0 or later:
```

```r
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Before splitting the edx data set, it's visible seen that some movies receive more ratings than others:



And some users are more active than others:



To split the edx data set in edx_train and edx_test:

```
#Creating subsets from edx data set
set.seed(1, sample.kind="Rounding")
temp_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train<-edx[-temp_index]
edx_test<-edx[temp_index]
#Removing users and movies of the test set not present in the training set
edx_test<-edx_test%>%semi_join(edx_train,by="userId")%>%
  semi_join(edx_train,by="movieId")
```

At first, only the average of ratings is used as a prediction:

```
avg<-mean(edx_train$rating)
rmse_avg<-RMSE(validation$rating,avg)
rmsedata<-data.frame(Method= "Simple Average",RMSE=rmse_avg)
```

The resulting RMSE is very big:

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

That is not desired. To improve it, the movie effect is added (some movies receive better ratings than others):

```
#Calculating the effect of the movies
mu<-mean(edx_train$rating)
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
#Predicting with the movie effect
test_bi<-edx_test%>%left_join(movie_avgs,by="movieId")%>%pull(b_i)
pred_me<-mu+test_bi
rmse_me<-RMSE(edx_test$rating,pred_me)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Movie Effect",RMSE=rmse_me))
```

The resulting RMSE is:

```
## [1] 0.9429615
```

It's an improvement, but could be better by adding the user effect (some users give more ratings than others):

```
user_avgs <- edx_train %>% left_join(movie_avgs,by="movieId")%>%
  group_by(userId) %>%
  summarize(u_i = mean(rating - mu-b_i))
#Predicting with the user effect
test_ui<-edx_test%>%left_join(user_avgs,by="userId")%>%pull(u_i)
pred_ue<-mu+test_bi+test_ui
rmse_ue<-RMSE(edx_test$rating,pred_ue)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Movie+User Effect",RMSE=rmse_ue))
```

The resulting RMSE is:

```
## [1] 0.8646844
```

To improve it even further, the genre effect is considered:

```
genre_avgs <- edx_train %>% left_join(movie_avgs,by="movieId")%>%left_join(user_avgs,by="userId")%>%
  group_by(genres) %>%
  summarize(g_i = mean(rating - mu-b_i-u_i))
#Predicting with the genre effect
test_gi<-edx_test%>%left_join(genre_avgs,by="genres")%>%pull(g_i)
pred_ge<-mu+test_bi+test_ui+test_gi
rmse_ge<-RMSE(edx_test$rating,pred_ge)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Movie+User+Genre Effect",RMSE=rmse_ge))
```

The resulting RMSE is:

```
## [1] 0.8643242
```

Using the validation set to obtain the RMSE of the algorithm:

```
pred_v<-validation%>%left_join(movie_avgs,by="movieId")%>%left_join(user_avgs,by="userId")%>%
  left_join(genre_avgs,by="genres")%>%mutate(pred=mu+b_i+u_i+g_i)%>%pull(pred)
pred_v[is.na(pred_v)]<-mu #Correcting NAs
pred_v[pred_v>5]<-5#Correcting superior values than the maximum rating(5)
pred_v[pred_v<0.5]<-0.5#Correcting inferior values than the minimum rating(0.5)
rmse_v<-RMSE(validation$rating,pred_v)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Validation RMSE",RMSE=rmse_v))
```

The final RMSE is:

```
## [1] 0.8652547
```

Bigger than the desired 0.86490.

To improve it even more, regularization is used. Regularization penalizes large estimates and generates better predictions.
First the movie effect is regularized. For this, the best lambda (tuning parameter) has to be found with cross validation:

```
#Finding lambda
lambdas <- seq(0, 10, 0.25)
only_sum <- edx_train %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){
  pred <-edx_test  %>%
    left_join(only_sum, by='movieId') %>%
    mutate(b_ir = s/(n_i+l)) %>%
    mutate(pred = mu + b_ir) %>%
    pull(pred)
  return(RMSE(pred, edx_test$rating))
})
lambda<-lambdas[which.min(rmses)]#Find the best lambda
```

The best lambda is:

```
## [1] 1.5
```

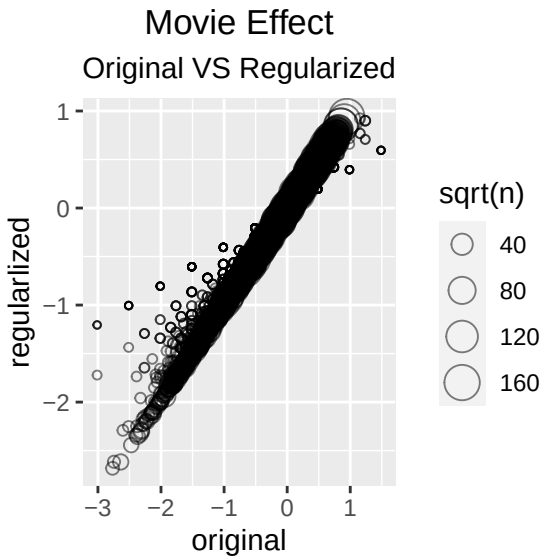
By using it:

```
movie_reg_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_ir = sum(rating - mu)/(n()+lambda), n_i = n())
test_bir<-edx_test%>%left_join(movie_reg_avgs,by="movieId")%>%pull(b_ir)
pred_mer<-mu+test_bir+test_ui+test_gi
rmse_mer<-RMSE(edx_test$rating,pred_mer)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Movie Reg+User+Genre",RMSE=rmse_mer))
```

The RMSE of Regularized movie effect is:

```
## [1] 0.8642536
```

The estimates change:



The user effect has to be regularized. A new lambda has to be found:

```
lambdas <- seq(0, 10, 0.25)
only_sumu <- edx_train  %>%left_join(movie_reg_avgs,by="movieId")%>%
  group_by(userId) %>%
  summarize(s2 = sum(rating - mu-b_ir), n_i2 = n())
rmses <- sapply(lambdas, function(l){
  pred <-edx_test  %>% left_join(movie_reg_avgs,by="movieId")%>%
    left_join(only_sumu, by='userId') %>%
    mutate(u_ir = s2/(n_i2+l)) %>%
    mutate(pred = mu + b_ir+u_ir) %>%
    pull(pred)
  return(RMSE(pred, edx_test$rating))
})
lambda<-lambdas[which.min(rmses)]
```

The best lambda is:

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

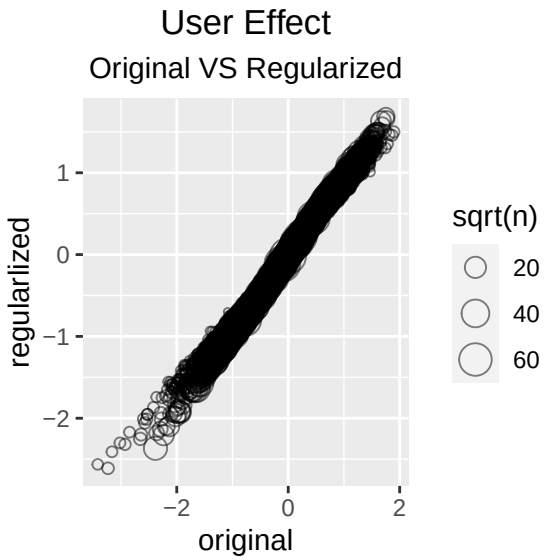
By using it:

```
user_reg_avgs <- edx_train %>% left_join(movie_reg_avgs,by="movieId")%>%
  group_by(userId) %>%
  summarize(u_ir = sum(rating - mu-b_ir)/(n()+lambda), n_i = n())
test_uir<-edx_test%>%left_join(user_reg_avgs,by="userId")%>%pull(u_ir)
pred_uer<-mu+test_bir+test_uir+test_gi
rmse_uer<-RMSE(edx_test$rating,pred_uer)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Movie Reg+User Reg+Genre",RMSE=rmse_uer))
```

The RMSE is now:

```
## [1] 0.8638584
```

The estimates change:



Finally, the genre effect is regularized. Finding the best lambda:

```
lambdas <- seq(0, 10, 0.25)
only_sumg <- edx_train  %>%left_join(movie_reg_avgs,by="movieId")%>%
  left_join(user_reg_avgs,by="userId")%>%
  group_by(genres) %>%
  summarize(s3 = sum(rating - mu-b_ir-u_ir), n_i3 = n())
rmses <- sapply(lambdas, function(l){
  pred <-edx_test  %>% left_join(movie_reg_avgs,by="movieId")%>%
    left_join(user_reg_avgs,by="userId")%>%
    left_join(only_sumg, by='genres') %>%
    mutate(g_ir = s3/(n_i3+l)) %>%
    mutate(pred = mu + b_ir+u_ir+g_ir) %>%
    pull(pred)
  return(RMSE(pred, edx_test$rating))
})
lambda<-lambdas[which.min(rmses)]
```

The best lambda is

```
## [1] 0
```

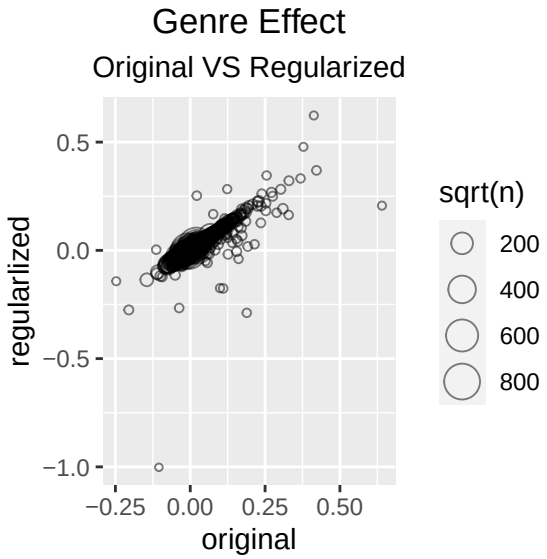
By using it:

```
genre_reg_avgs <- edx_train %>% left_join(movie_reg_avgs,by="movieId")%>%
  left_join(user_reg_avgs,by="userId")%>%
  group_by(genres) %>%
  summarize(g_ir = sum(rating - mu-b_ir-u_ir)/(n()+lambda), n_i = n())
test_gir<-edx_test%>%left_join(genre_reg_avgs,by="genres")%>%pull(g_ir)
pred_ger<-mu+test_bir+test_uir+test_gir
rmse_ger<-RMSE(edx_test$rating,pred_ger)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Movie Reg+User Reg+Genre Reg",RMSE=rmse_ger))
```

The RMSE is:

```
## [1] 0.8638564
```

The estimates change:



Using the validation set with the final model:

```
pred_vr<-validation%>%left_join(movie_reg_avgs,by="movieId")%>%left_join(user_reg_avgs,by="userId")%>%
  left_join(genre_reg_avgs,by="genres")%>%mutate(pred=mu+b_ir+u_ir+g_ir)%>%pull(pred)
pred_vr[is.na(pred_vr)]<-mu#Correcting NAs
pred_vr[pred_vr>5]<-5#Correcting superior values than the maximum rating(5)
pred_vr[pred_vr<0.5]<-0.5#Correcting inferior values than the minimum rating(0.5)
rmse_vr<-RMSE(validation$rating,pred_vr)
rmsedata<-bind_rows(rmsedata,data.frame(Method="Validation Reg RMSE",RMSE=rmse_vr))
```

The resulting RMSE is:

```
## [1] 0.8648047
```

Results
The RMSEs obtained with each model are:

```
##                          Method      RMSE
## 1                Simple Average 1.0612018
## 2                 Movie Effect 0.9429615
## 3            Movie+User Effect 0.8646844
## 4      Movie+User+Genre Effect 0.8643242
## 5              Validation RMSE 0.8652547
## 6         Movie Reg+User+Genre 0.8642536
## 7     Movie Reg+User Reg+Genre 0.8638584
## 8 Movie Reg+User Reg+Genre Reg 0.8638564
## 9          Validation Reg RMSE 0.8648047
```

The model improves a lot from the simple average to the movie effect, the same happens from the movie effect to the user effect, but doesn't change much from the user effect to the genre effect. After regularization, the performance improves. Without regularization the RMSE of the validation set is bigger than the desired value of 0.86490. After the regularization of the three predictors,the RMSE of the validation set is smaller than the desired value, fulfilling the objective.

Conclusion
The algorithm is capable of predicting the ratings with a RMSE of

```
## [1] 0.8648047
```

As desired. It uses the effects of movies, users and genres with regularization to obtain the predictions. To improve the performance of the algorithm, the date (timestamp in the data set) could be used as predictor as well(as weeks or years).