MovieLensReport

Ricky Rahardja

16/08/2019

Introduction

The movielens dataset is used in this assignment. The dataset provided consists of 5 variables, and they are userId, movieId, rating, timestamp, title and genre. The objective of this assignment is to build machine learning algorithm that make prediction of ratings, by using other variables as independent variables.

Data Sets is produced as instruction in this assignment below:

```
# Create edx set, validation set

# Note: this process could take a couple of minutes

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

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
## Warning: package 'tibble' was built under R version 3.5.2
```

```
## Warning: package 'tidyr' was built under R version 3.5.2
```

```
## Warning: package 'purrr' was built under R version 3.5.2
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
## Warning: package 'stringr' was built under R version 3.5.2
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

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

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
# 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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[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)
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

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

In this assignment, author is using userId and movie Id to predict the rating. As studied in this course, using linear model would be hard to achieve given the large dataset. Alternatively, author is using data manipulating techniques such as subsetting, joining, summarising in tidyverse package to construct and

compute according to required statistical formulas.

The models constructed is evaluated by using Root Mean Square Error (RMSE) approach.

Methods and Analysis

Method used in this assignment is using Naive Average, model taking into account userId, model taking into account userId and movieId, model taking into account userId and movieId with regularization, and finally validating model using validation set.

Create Test Set

First, author is using caret package to split edx dataset into train and test sets. Further, author is using semi-join function to remove missing users and movies in training dataset.

```
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

RMSE as benchmark

Author then construct RMSE function for assessment of the models constructed.

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

Method 1

The first method used is called Naive Average Model, by using the mean of ratings in training set to predict ratings in test set.

```
mu_hat <- mean(train_set$rating)
mu_hat
```

```
## [1] 3.512482
```

```
rmse1 <- RMSE(test_set$rating, mu_hat)
rmse1
```

```
## [1] 1.059904
```

```
rmse_results1 <- data_frame(method = "Naive Average Model", RMSE = rmse1)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
rmse_results1
```

```
## # A tibble: 1 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 Naive Average Model  1.06
```

The RMSE is 1.0610 for the fist model and it is very far from the taget of at least 0.9000 in this assignment.

Method 2

The second model is to use movieId as the explanatory variable of ratings.

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

predicted_ratings_movie <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

rmse2 <- RMSE(predicted_ratings_movie, test_set$rating)
rmse2
```

```
## [1] 0.9437429
```

```
rmse_results2 <- bind_rows(rmse_results1,
                           data_frame(method="Movie Effect Model",
                                      RMSE = rmse2))
rmse_results2
```

```
## # A tibble: 2 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 Naive Average Model 1.06
## 2 Movie Effect Model  0.944
```

The RMSE is dropped to 0.9445 and it is still above the minimum 0.9000 target.

Method 3

In method 3, author is taking into account userId also as as the explanatory variables.

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

predicted_ratings_movies_user <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)


rmse3 <- RMSE(predicted_ratings_movies_user, test_set$rating)
rmse3
```

```
## [1] 0.865932
```

```
rmse_results3 <- bind_rows(rmse_results2,
                           data_frame(method="Movie + User Effects Model",
                                      RMSE = rmse3))
rmse_results3
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Naive Average Model        1.06
## 2 Movie Effect Model         0.944
## 3 Movie + User Effects Model 0.866
```

The RMSE drop to 0.8669 and this shows that variable movieId and userId are substantional variables to predict movie ratings.

Method 4

In method 4, author is using regularization approach learnt from the course and find to find the best tune of lamda for the prediction.

```
lambdas <- seq(0, 10, 0.25)

rmse4 <- sapply(lambdas, function(l){

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

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

  predicted_ratings_reg_movies_user <- test_set %>%
    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_reg_movies_user, test_set$rating))
})
rmse4
```

```
##  [1] 0.8659320 0.8658390 0.8657585 0.8656875 0.8656242 0.8655676 0.8655172
##  [8] 0.8654722 0.8654323 0.8653970 0.8653661 0.8653391 0.8653160 0.8652963
## [15] 0.8652800 0.8652669 0.8652566 0.8652492 0.8652444 0.8652421 0.8652421
## [22] 0.8652444 0.8652489 0.8652553 0.8652637 0.8652739 0.8652859 0.8652995
## [29] 0.8653147 0.8653314 0.8653496 0.8653691 0.8653900 0.8654122 0.8654355
## [36] 0.8654600 0.8654856 0.8655123 0.8655399 0.8655686 0.8655982
```

```
lambda <- lambdas[which.min(rmse4)]
lambda
```

```
## [1] 4.75
```

```
rmse4[lambda]
```

```
## [1] 0.8656875
```

```
rmse_results4 <- bind_rows(rmse_results3,
                           data_frame(method="Movie + User Effect Model Regularization Model",
                                      RMSE = min(rmse4)))
rmse_results4
```

```
## # A tibble: 4 x 2
##   method                                     RMSE
##   <chr>                                     <dbl>
## 1 Naive Average Model                       1.06
## 2 Movie Effect Model                        0.944
## 3 Movie + User Effects Model                0.866
## 4 Movie + User Effect Model Regularization Model 0.865
```

The result is not very supprising with only achieving RMSE of 0.8667, it is an improvement but it is not a big improvement thus it would likely need more explanatory variables to predict ratings.

Method 5 (RESULT)

Finally, author is using the last approach in method 4 to test on the validation set.

```
lambdas_validation <- seq(0, 10, 0.25)

rmse_validation <- sapply(lambdas_validation, 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 - b_i - mu)/(n()+l))

  predicted_ratings_validation <- 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, validation$rating))
})
rmse_validation
```

```
##  [1] 0.8653488 0.8652810 0.8652226 0.8651708 0.8651246 0.8650830 0.8650456
##  [8] 0.8650119 0.8649816 0.8649544 0.8649303 0.8649088 0.8648900 0.8648736
## [15] 0.8648596 0.8648477 0.8648379 0.8648301 0.8648242 0.8648201 0.8648177
## [22] 0.8648170 0.8648178 0.8648202 0.8648240 0.8648291 0.8648356 0.8648433
## [29] 0.8648523 0.8648625 0.8648737 0.8648861 0.8648995 0.8649138 0.8649292
## [36] 0.8649455 0.8649626 0.8649807 0.8649995 0.8650192 0.8650396
```

```
lambda_validation <- lambdas_validation[which.min(rmse_validation)]
lambda_validation
```

```
## [1] 5.25
```

```
rmse_validation[lambda_validation]
```

```
## [1] 0.8651246
```

```
rmse_results5 <- bind_rows(rmse_results4,
                           data_frame(method="Movie + User Effect with Regularization Model on Validati
                                      RMSE = min(rmse_validation)))
```

```
rmse_results5
```

```
## # A tibble: 5 x 2
##   method                                                      RMSE
##   <chr>                                                      <dbl>
## 1 Naive Average Model                                        1.06
## 2 Movie Effect Model                                         0.944
## 3 Movie + User Effects Model                                 0.866
## 4 Movie + User Effect Model Regularization Model             0.865
## 5 Movie + User Effect with Regularization Model on Validation Set 0.865
```

And to ensure RMSE result is as expected author use codes below:

```
min(rmse_validation)
```

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

```
#Ensure RMSE result is lower than 0.8649 on Validation Set
if(min(rmse_validation) <= 0.8649){
  print("I deserve to get 25 points.")
  } else {
  print("I must develop better prediction algorithm!")
  }
```

```
## [1] "I deserve to get 25 points."
```

The RMSE result is 0.8645 and this is already smaller than the best target of RMSE of 0.8649 in this assignment.

Conclusion

In conclusion, author has learnt that the more exploratory variables added into the models, it would improve the prediction. However, it would only for relevant and determining variables, which means they are improving the model and not worsening the model.

Further, regularization approach is also improving by a margin. However, with a larger training dataset, it would give a reasonable improvement on the predictive model.