MovieLens Project

Haichen Dong

5/7/2019

```
knitr::opts_chunk$set(echo = TRUE)
```

Project Overview

This project will create a movie recommendation system by using the 10M version of MovieLens dataset (http://grouplens.org/datasets/movielens/10m/). In this project, we uses R as program language to build a machine learning algorithm and uses the data in one subset to train the model also uses the movie ratings in the validation set to predict.

Prepare Data

```
#All the Libraries needed.
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")

library(tidyverse)
library(caret)
```

```
#Build data set
destfile="U:/projects/FinalPro/Movielens.RData"
if(!file.exists(destfile)){
  dl <- tempfile()
  download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

  ratings <- read.table(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)
  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)
  save(edx, validation, file = destfile)
  rm(dl, ratings, movies, test_index, temp, movielens, removed,edx,validation)
}
load(destfile)
#Training dataset name: edx; testing dataset name: validation.
```

Build ML Model

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

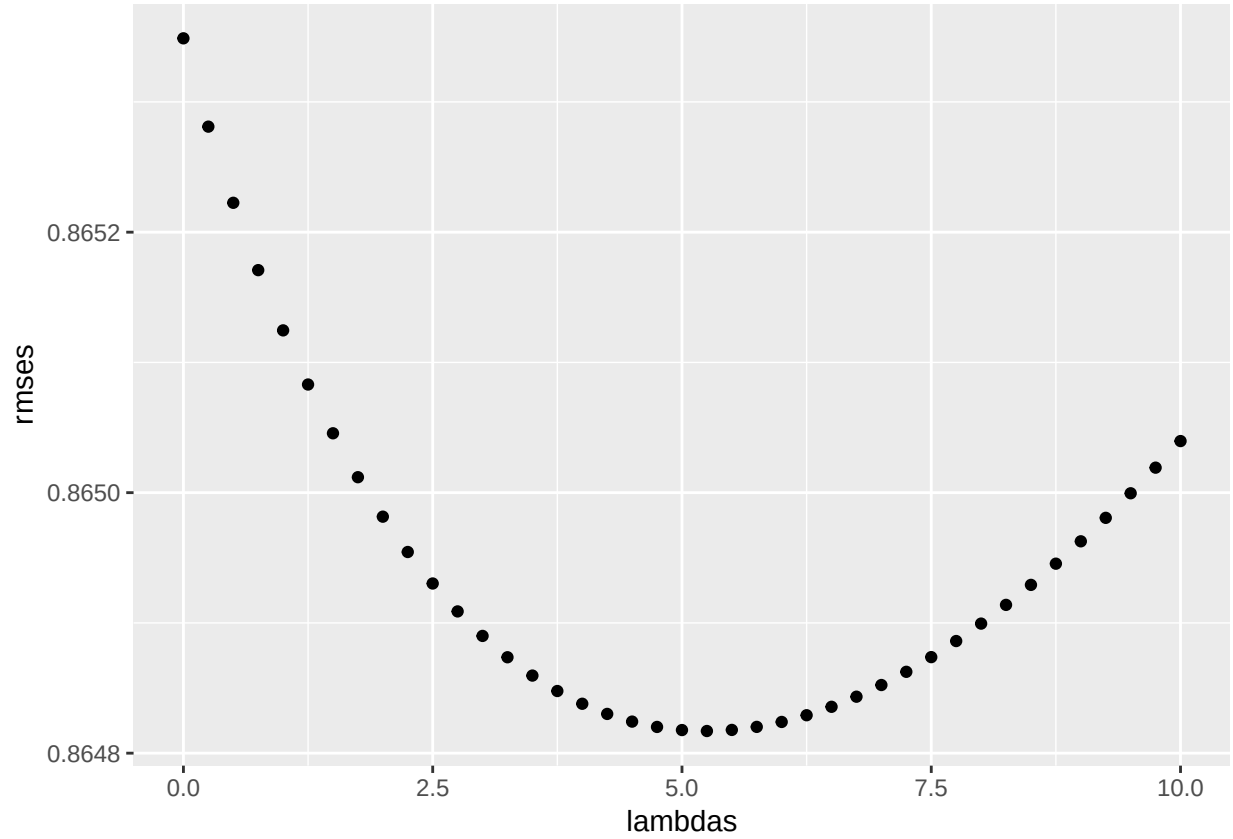
```
##### Predict the same rating for all movies regardless of user.
mu_hat <- mean(edx$rating)
#mu_hat
naive_rmse <- RMSE(validation$rating, mu_hat)
#naive_rmse
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
```

```
#  Predict by adding movie effects model
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

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

model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effect Model",
                                     RMSE = model_1_rmse))
```

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

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

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
```

```
                        tibble(method="Movie + User Effects Model",
                                   RMSE = model_2_rmse))
```

```
# Predict by using Regularization
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```



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

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

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie + User Effect Model",
                                 RMSE = min(rmses)))
```

RMSE Testing Results

```
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie + User Effect Model	0.8648170

Summary

By adding the induvial movie rate and user rate to the average rate model, Residual Mean Squared Error (RMSE) reduces from 1.0612018 to 0.8653488. Model also applies regularization to movie and user effects. By choosing lambda=5.25, RMSE reduces to 0.8648170.

Project File Link at GitHub : https://github.com/Haichen-Dong/MovieLensProject