MovieLens Project

Introduction/Overview/Executive summary

This is the MovieLens report for Data Science (Capstone) course. It consists of several sections including introduction/overview/executive summary, methods/analysis, results and conclusion. All the analysis is conducted on R statistic/Rstudio platform. The codes are displayed and discussed in result section. The data for this analysis in obtained from the cloud and the link is "http://files.grouplens.org/datasets/movielens/ml-10m.zip. Linear regression model with various predictors are used in this analysis. Regularized Movie & User Model was found to have the lowest RMSE.

Methods/Analysis

There are two datasets associated with this project, ratings.dat and movies.dat. The rating dataset has 10,000,054 records and consists of four columns: UserId, movieId, rating and timestamp. The second dataset, movies.dat has 10,681 records and only three columns: movieId, title and genres. These two datasets are joined before the analysis.

Three different models are built and assessed. The analysis stops after the third model as it has reached RMSE less than 0.8649 which is the maximum points in the RMSE section.

The joined main dataset is divided into two subsets: training and testing dataset using createDataPartition function. A validation dataset is extracted from the testing dataset where the records exisit in the training dataset. This validation dataset will be used to test the RMSE of the model. To avoid any influence of the data integrity, the validation dataset is removed from the training dataset.

First, the average rating is calculated for entire training dataset. For first model, only movie bias is considered as rating might be rated higher or lower based on the movie title. The movie bias is calculated by taking the average of the different between the rating and the average rating. The predicted values (validation) for first model is adding the average rating of entire training dataset with the movie bias. Then RMSE value is calculated using validation rating and the predicted values.

For second model, both movie and user biases are considered. Movie bias is as described above. User bias is calculated by taking the average of after substracting the rating by movie title and specific user with the average rating of the entire training dataset and movie bias. The predicted values (validation) for first model is adding the average rating of entire training dataset with the movie bias. Then RMSE value is calculated using validation rating and the predicted values.

The RMSE improves using second model which is having movie and user biases.

The movie sample sizes are different. The highest ratings are given for one movie is 31,362 whereas some movies are only given one rating. This might affect the prediction. Regularization is introduced in the model to add penalty for the larger sample movie. The regularization equation is to a new variable, lambda to the denominator while calculating the bias. Interative loop is used to calculate the best lambda for the equation. The best lambda is defined as the value when RMSE is the lowest.

Results

Below are the codes to load all necessary libraries:

Loading libraries

```r
pkgTest <- function(x)
{
  if (!require(x,character.only = TRUE))
  {
    install.packages(x,dep=TRUE)
    if(!require(x,character.only = TRUE)) stop("Package not found")
  }
}

pkgTest("tidyverse")
```

```
## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```r
pkgTest("lubridate")
```

```
## Loading required package: lubridate

##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
pkgTest("stringr")
pkgTest("tidytext")
```

```
## Loading required package: tidytext
```

```r
pkgTest("caret")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

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

```r
pkgTest("tinytex")
```

```
## Loading required package: tinytex
```

Aquiring MovieLen dataset

Below codes are to acquire movielens dataset:

```r
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")
```

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

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

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year

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

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

```r
# 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")
```

Creating training (edx) and testing (temp) dataset

```r
set.seed(1)
test_index <- createDataPartition(y=movielens$rating, times=1, p=0.1, list=FALSE)
edx  <- movielens[-test_index,]
temp <- movielens[test_index,]
```

Creating Validation dataset and revised Training dataset (edx)

```r
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Removing validation dataset from training dataset
edx        <- anti_join(edx, validation)
```

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

Model 1 (Only Considering movieId)

```r
# Calculating average rating for entire training dataset
FinalResult01  <- validation
meanRating_edx <- mean(edx$rating)
FinalResult01$meanRating_edx <- mean(edx$rating)

# Calculating movidId bias on the training set and conduct the prediction
meanMovieBias_edx       <- edx %>% group_by(movieId) %>%
  summarize(biasMovie01 = sum(rating - meanRating_edx)/(n()))
FinalResult01    <- merge(FinalResult01, meanMovieBias_edx, by="movieId", all.x=TRUE)
FinalResult01$PredRating01 <- FinalResult01$meanRating_edx + FinalResult01$biasMovie01
rmse_01                 <- RMSE(FinalResult01$rating,FinalResult01$PredRating01)
print(paste0("RMSE for Model 1 = ", rmse_01))
```

```
## [1] "RMSE for Model 1 = 0.943704611124392"
```

Model 2 (Only Considering movieId and userId)

```r
# Calculating average rating for entire training dataset
FinalResult02  <- validation
meanRating_edx <- mean(edx$rating)
FinalResult02$meanRating_edx <- mean(edx$rating)

# Calculating movidId & userId biases on the training set and conduct the prediction
edx2 <- merge(edx, meanMovieBias_edx, by="movieId", all.x=TRUE)
meanUserBias_edx       <- edx2 %>% group_by(userId) %>%
  summarize(biasUser02 = sum(rating - meanRating_edx - biasMovie01)/(n()))
FinalResult02    <- merge(FinalResult01, meanUserBias_edx, by="userId", all.x=TRUE)
FinalResult02$PredRating02 <- FinalResult02$meanRating_edx +
  FinalResult02$biasMovie01 + FinalResult02$biasUser02
rmse_02                 <- RMSE(FinalResult02$rating,FinalResult02$PredRating02)
print(paste0("RMSE for Model 1 = ", rmse_02))
```

```
## [1] "RMSE for Model 1 = 0.865532933550825"
```

Model 3 (Only Considering movieId and userId but with regularization)

```
FinalResult03  <- validation

lambdas  <- seq(0, 10, 1)
for (ilambdas in lambdas){
  meanMovieBiasRegularized_edx       <- edx %>% group_by(movieId) %>%
    summarize(biasMovieRegularized03 = sum(rating - meanRating_edx)/(n()+ilambdas))
  FinalResult03a <- merge(FinalResult03, meanMovieBiasRegularized_edx, by="movieId", all.x=TRUE)

  edx2 <- merge(edx, meanMovieBiasRegularized_edx, by="movieId", all.x=TRUE)
  meanUserBiasRegularized_edx <- edx2 %>% group_by(userId) %>%
    summarize(biasUserRegularized03=sum(rating-meanRating_edx-biasMovieRegularized03)/(n()+ilambdas))
  FinalResult03b <- merge(FinalResult03a, meanUserBiasRegularized_edx, by="userId", all.x=TRUE)

  FinalResult03b$PredRating03 <- rep(meanRating_edx, nrow(FinalResult03a)) +
    FinalResult03b$biasMovieRegularized03 + FinalResult03b$biasUserRegularized03

  rmse                  <- RMSE(FinalResult03b$rating, FinalResult03b$PredRating03)
  print(paste0("Trying lambda = ", ilambdas, " and RMSE = ", rmse))

  if (ilambdas == 0){
    lambdasChoosen = ilambdas
    rmseChoosen = rmse
  } else {
    if (rmseChoosen > rmse){
      lambdasChoosen = ilambdas
      rmseChoosen = rmse
    }
  }
}
```

```
## [1] "Trying lambda = 0 and RMSE = 0.865532933550825"
## [1] "Trying lambda = 1 and RMSE = 0.865313952922557"
## [1] "Trying lambda = 2 and RMSE = 0.865168549749048"
## [1] "Trying lambda = 3 and RMSE = 0.865072282981788"
## [1] "Trying lambda = 4 and RMSE = 0.865014584369877"
## [1] "Trying lambda = 5 and RMSE = 0.864988414585793"
## [1] "Trying lambda = 6 and RMSE = 0.864988516991341"
## [1] "Trying lambda = 7 and RMSE = 0.865010776755441"
## [1] "Trying lambda = 8 and RMSE = 0.865051888127315"
## [1] "Trying lambda = 9 and RMSE = 0.865109149297303"
## [1] "Trying lambda = 10 and RMSE = 0.865180323691184"
```

```
print(paste("Lambda that produces the lowest RMSE : ", lambdasChoosen, sep=""))
```

```
## [1] "Lambda that produces the lowest RMSE : 5"
```

```
print(paste("RMSE : ", rmseChoosen, sep=""))
```

```
## [1] "RMSE : 0.864988414585793"
```

```
# Rerun the analysis using the lowert RMSE lambda
meanMovieBiasRegularized_edx       <- edx %>% group_by(movieId) %>%
  summarize(biasMovieRegularized03 = sum(rating - meanRating_edx)/(n()+lambdasChoosen))
FinalResult03a <- merge(FinalResult03, meanMovieBiasRegularized_edx, by="movieId", all.x=TRUE)

edx2 <- merge(edx, meanMovieBiasRegularized_edx, by="movieId", all.x=TRUE)
meanUserBiasRegularized_edx       <- edx2 %>% group_by(userId) %>%
  summarize(biasUserRegularized03 = sum(rating - meanRating_edx
                                        - biasMovieRegularized03)/(n()+lambdasChoosen))
FinalResult03b <- merge(FinalResult03a, meanUserBiasRegularized_edx, by="userId", all.x=TRUE)

FinalResult03b$PredRating03 <- meanRating_edx + FinalResult03b$biasMovieRegularized03 +
                                 FinalResult03b$biasUserRegularized03
rmse_03 = rmseChoosen
```

Conclusion

Displaying all three models and the choosen model

```
results <- data.frame(methods=c("Movie Model","Movie & User Model",
                       "Regularized Movie & User Model"),rmse = c(rmse_01, rmse_02, rmse_03))
ChoosenModel <- results[results$rmse == min(results$rmse),]
print(paste("Model choosen is : ", ChoosenModel$methods, " and RMSE = ",
            ChoosenModel$rmse, sep=""))
```

```
## [1] "Model choosen is : Regularized Movie & User Model and RMSE = 0.864988414585793"
```