Movielens Project

Kumar Abhishek

04/04/2020

Introduction

The goal of this project is to apply the machine learning techniques learnt in the Edx Data Science cetrification program to predict the ratings of movies on the movielens data set available at https://grouplens.org/datasets/movielens/10m/, ensuring that the Root Mean Square Error (RSME) of the producted ratings is accepatbly low.

Method

The data is extracted and compiled in a tidy format named movielens. The movielens data is then split into a training set named "edx" and a validation named "validation", for assessing the acuurancy of trained model through calculation of RMSE.

Creation of edx and validation sets:

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

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

```
## Warning: package 'tidyverse' was built under R version 3.6.1
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

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

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'tidyr' was built under R version 3.6.1
```

```
## Warning: package 'readr' was built under R version 3.6.1
```

```
## Warning: package 'purrr' was built under R version 3.6.1
```

```
## Warning: package 'dplyr' was built under R version 3.6.1
```

```
## Warning: package 'forcats' was built under R version 3.6.1
```

```
## -- 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
```

```
## Warning: package 'caret' was built under R version 3.6.1
```

```
## 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
```

```
## Warning: package 'data.table' was built under R version 3.6.1
```

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

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

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

```
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")
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
 edx <- movielens[-test_index,]
 temp <- movielens[test_index,]
 validation <- temp %>%
      semi_join(edx, by = "movieId") %>%
      semi_join(edx, by = "userId")
 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)
```

Splitting edx set into test and train sets:

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
index<-createDataPartition(edx$rating,times=1,p=0.5,list=FALSE)
train<-edx%>%slice(-index)
test<-edx%>%slice(index)
test<-test%>%semi_join(train,by="movieId")%>%semi_join(train,by="userId")
```

Defining RMSE function:

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

Training a model using "Regularization" method for values of lambda ranging from 0 to 10, for identifying the lambda which minimizes RMSE on test set. Estimating movie bias parameter b_i and user bias parameter b_u and then using these to predict the ratings for test set.

```
 lambdas <- seq(0, 10, 0.25)
 rmses <- sapply(lambdas, function(l){
     mu <- mean(train$rating)
     b_i <- train %>%
         group_by(movieId) %>%
         summarize(b_i = sum(rating - mu)/(n()+l))
     b_u <- train %>%
         left_join(b_i, by="movieId") %>%
         group_by(userId) %>%
         summarize(b_u = sum(rating - b_i - mu)/(n()+l))
     predicted_ratings <-
         test %>%
         left_join(b_i, by = "movieId") %>%
         left_join(b_u, by = "userId") %>%
         mutate(pred = mu + b_i + b_u) %>%
```

```
        .$pred
    return(RMSE(predicted_ratings, test$rating))
})
lambda <- lambdas[which.min(rmses)]
lambda
```

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

So, lambda=5 minimizesRMSE on test set.Now using this value of lambda and the mean rating mu_edx for the edx set, movie parameter b_i_edx and user parameter b_u_edx can be estimated.

```
mu_edx <- mean(edx$rating)
b_i_edx <- edx %>%
    group_by(movieId) %>%
    summarize(b_i_edx = sum(rating - mu_edx)/(n()+lambda))
b_u_edx <- edx %>%
    left_join(b_i_edx, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_edx = sum(rating - b_i_edx - mu_edx)/(n()+lambda))
```

Using the estimated b_i_edx and b_u_edx parameters along with lambda = 5 on the validation set to predict ratings predicted_ratings_v.

```
predicted_ratings_v <-
    validation %>%
    left_join(b_i_edx, by = "movieId") %>%
    left_join(b_u_edx, by = "userId") %>%
    mutate(pred_v = mu_edx + b_i_edx + b_u_edx) %>%
    .$pred_v
```

Now calculating the RMSE for the predicted ratings on validation set w.r.t the actual ratings on the validation set

```
RMSE(predicted_ratings_v, validation$rating)
```

```
## [1] 0.8648177
```

Result

The regularization method gives an RMSE of 0.8648177 on the validation data set.

Conculsion

The regularization technique provides an RMSE which is accetable as per the requirements of this project. However, it should be considered here that the model overlooks similarities in the rating patterns for different categories of movies and audiences. It is possible to decrease the RSME further by using the matrix factorization technique based on Singular Value Decomposition (SVD) method. SVD helps in identifying and including all factors which significantly affect the predicted values, into the fitted model.