MovieLens Project Report

suresh t

3/11/2020

Introduction

```
This work is the first part of HarvardX: PH125.9xData Science: Capstone. The project requires us to par
```

The rating scale ranges from 0 to 5 in Netflix rating rankings. The key objective for the project is to train a machine learning algorithm using inputs from one data set so as to predict movie ratings in another data set. The data from which the train data set and prediction data sets would be created is obtained from the following link:

https://grouplens.org/datasets/movielens/10m/

```
The MovieLens data set contains 10000054 rows, 10677 movies, 797 genres and 69878 users.
```

User ratings are the basis on which recommendation models to make specific recommendations.

Considering the large size of the dataset, an efficient model is needed to predict movie ratings that uses on an user id and a movie id. Data visualization techniques are used to get a perspective of the data. This also enables us to identify the right approach to arrive at the most appropriate model.

The best approach seems to the penalized least squares approach that is based on the mean movie rating. This mean(average) so obtained is adjusted for user-bias and movie-bias. It is pertinent to highlight that the dataset contains ratings from users who rate just a few movies and also movies with a small number of total ratings. This implication is that these tend to have more volatile ratings than users who rate lots of movies and movies with lots of ratings. To adjust for these effects, a penalty - lambda - is taken into account. The Netflix challenge decided the winner based on the residual mean squared error (RMSE) on a test set. To provide satisfying results many different models were trained for this project and the best model was chosen based on the RMSE on the validation set.

Data Wrangling

Downloading the data

Retrieving/downloading the data forms the primary step in any data analysis project. Subsequent to downloading the data, the next immediate step would be to clean or tidy the downloaded data. The data after tidying up is used in analyzing the data. In the MovieLens project, the 10 M version of the data set can be downloaded using the following set of codes: 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") # 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"))

Building the data set

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")

Create a validation data set which is 10% of MovieLens data set

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

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

#Data Analysis & Visualisation It is important to visualize data to give a perspective on the data set as well as data analytics. First, lets have a snapshot of the data in the edx data by using the str command. str(edx) The edx data set contains 9000055 rows.Each rating consists of a userId, a movieId, the rating, a timestamp and information about the movie like title and genre. hist(edx$rating, col = "red3")summary(edx$rating) The inference is as follows: Ratings range from 0.5 to 5.0. The difference in meadian an mean shows that the distribution is skewed towards higher ratings. The chart shows that whole-number ratings are more common that 0.5 ratings. Below, are the codes to view distribution of Movie ratings and distribution of users

#Distribution of Movie Ratings edx %>% group_by(movieId) %>% summarize(n = n()) %>% ggplot(aes(n)) + geom_histogram(fill = "yellow4", color = "red3", bins = 10) + scale_x_log10() + ggtitle("Movies Ratings Distribution")

#Distribution of Users edx %>% group_by(userId) %>% summarize(n = n()) %>% ggplot(aes(n)) + geom_histogram(fill = "red3", color = "black", bins = 10) + scale_x_log10() + ggtitle("Distribution of Users Ratings") Method: The most suitable approach to this project to get a RMSE of less than 0.86490 would be the Optimal Least Squares Method. The essential part in this exercise is to arrive at the optimal tuning factor 'Lambda' at which the RMSE is minimal and also below 0.86490. The following codes is used to determine the optimal tuning factor: Root Mean Square Error Loss Function RMSE <- function(true_ratings, predicted_ratings){ sqrt(mean((true_ratings - predicted_ratings)^2)) }

lambdas <- seq(0, 5, 0.25)

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

#Calculate the mean of ratings from the edx training set mu <- mean(edx$rating)

#Adjust mean by movie effect and penalize low number on ratings b_i <- edx %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+l))

adjust mean by user and movie effect and penalize low number of ratings b_u <- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>% summarize(b_u = sum(rating - b_i - mu)/(n()+l))

#predict ratings in the training set to derive optimal penalty value 'lambda' predicted_ratings <- edx %>% left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u) %>% .$pred

return(RMSE(predicted_ratings, edx$rating)) })

plot(lambdas, rmses)

lambda <- lambdas[which.min(rmses)] paste(' At a Lambda of, lambda,'the minimum RMSE is', min(rmses))

At a Lambda of 0.5 the minimal RMSE obtained is

Now, the prediction on the validation set would be done based on the Lambda obtained.

lambda <- 0.5

pred_y_lse <- sapply(lambda,function(l){

#Derive the mearn from the training set mu <- mean(edx$rating)

#Calculate movie effect with optimal lambda b_i <- edx %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+l))

#Calculate user effect with optimal lambda b_u <- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>% summarize(b_u = sum(rating - b_i - mu)/(n()+l))

#Predict ratings on validation set predicted_ratings <- validation %>% left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u) %>% .$pred #validation

return(predicted_ratings)

}) The predicted values are thus obtained.

#Conclusion: The aim of the project was to predict movie ratings from a dataset of existing movie ratings. The optimal least square method turns out to be one of the better methods to the algorithm. As can be observed, the RMSE of 0.8566952 at a Lambda of 0.5.