HarvardX: PH125.9x Data Science
MovieLens Prediction Project

Jananya Sivakumar

Overview

HarvardX: Data Science course deals with the different ways to explore the data given, visualize the data using R packages and various modelling approaches employing Machine Learning techniques. The objective of the MovieLens project is to build a recommendation system that suggests the audience the movie of their preference.

Introduction

One of the most widespread applications of Machine Learning techniques is in recommender systems, which suggest relevant products/services to the users. The generation of data these recommendation engines can be analyzed to take tactical and strategic decisions. The purpose of the MovieLens project is to create a recommendation system where Machine Learning algorithms and models are implemented to predict the preference of the problem.

Aim of the Project

The aim of this project is to train a Machine Learning algorithm to predict the movie ratings in the provided validation set using the user ratings from the inputs of the provided subset. To evaluate the closeness of our predictions to the true values, RMSE (Root Mean Square Error) values in the validation set. RMSE is a measure of accuracy that compares the different models of a particular dataset (lower RMSE is better).

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

The criteria of this algorithm are for the RMSE to be lower than 0.8775.

Dataset

In this project, MovieLens dataset is used to create the recommendation system, using the 10M version of the dataset. https://grouplens.org/datasets/movielens/10m/

```
#Installing the required packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us
.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
ct.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.
us.r-project.org")
```

```
#Attaching the required packages
library(tidyverse)
library(caret)
library(data.table)
library(reshape2)
library(lubridate)
library(dplyr)
library(Rcpp)
library(ggplot2)
library(ggthemes)

#Loading the dataset
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/rati
ngs.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(movieId), tit
le = as.character(title), genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

Methods

Data Preparation

Preparation of data is carried out by generating the validation and trainig datasets to design and test the algorithms. 10% of the given MovieLens dataset is the validation set that is the final hold-out test set. Training, developing and selection of the modelling algorithms is done using the training dataset.

```
#Validation set: 10% of the data
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
ist = 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)
```

```
#Train set
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list =
FALSE)
train_set <- edx[-test_index,]
temp <- edx[test_index,]

# Make sure userId and movieId in train set are also in edx set
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
rm(test_index, temp, removed)
```

Data Exploration

Analysis of the data is done by exploring the given data.

Structure

The internal structure of the R object is displayed along with the dimention of the dataframe. The names of the column heads in the dataframe are also printed. Displaying the first 6 rows present in the input data frame and the summary of the data. The standard deviation of the rating and the summary of the column including the minimum, maximum values and the quartile medians of the data is displayed.

```
#Structure
str(edx)
dim(edx)
names(edx)

#Viewing the data
head(edx)
summary(edx)
sd(edx$rating)
summary(edx$rating
```

Computation

Computation measure (mean, median, min, max, etc..) or a function for each factor variable in a vector. It creates a subset and apply functions to each of the subset. The total observation of the data is calculated. Using R package dplyr, the data is grouped by genres and summarized.

```
#Sum and Mean
tapply(edx$rating, edx$genres, sum)
tapply(edx$rating, edx$genres, mean)

#Total Observation
tot_observation <- length(edx$rating) + length(validation$rating)
tot_observation

#Grouping of data
edx %>% group_by(genres) %>% summarise(n=n()) %>% head()
tibble(count = str_count(edx$genres, fixed("|")), genres = edx$genres) %>% gr
oup_by(count, genres) %>% summarise(n = n()) %>% arrange(-count) %>% head()
```

Data Visualisation

Creating Plots

```
edx %>% mutate(year = year(as_datetime(timestamp, origin="1970-01-01"))) %>%
  ggplot(aes(x=year)) +
  geom_histogram(color = "white") +
  ggtitle("Rating Distribution Per Year") +
  xlab("Year") +
  ylab("Number of Ratings") +
  scale_y_continuous(labels = comma) +
  theme_economist()


edx %>% group_by(rating) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=rating, y=count)) +
  geom_line() +
  geom_point() +
  scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x))) +
  ggtitle("Rating Distribution", subtitle = "Higher ratings are prevalent.")
+
  xlab("Rating") +
  ylab("Count") +
  theme_economist()
```

Modelling Approaches

Average Ratings Model

```
mu <- mean(edx$rating)
mu
model_1 <- RMSE(validation$rating, mu)
naive_rmse
rmse_results <- data_frame(Model = "Average", RMSE = model_1)
rmse_results
```

MOVIE AND USERS MODEL

```
#Cross validation to determine lambda

lambdas <- seq(0, 8, 0.1)
rmses_2 <- matrix(nrow=10,ncol=length(lambdas))
# perform 10-fold cross validation to determine the optimal lambda
for(k in 1:10) {
  train_set <- edx[cv_splits[[k]],]
  test_set <- edx[-cv_splits[[k]],]

  # Make sure userId and movieId in test set are also in the train set
  test_final <- test_set %>%
    semi_join(train_set, by = "movieId") %>%
    semi_join(train_set, by = "userId")

  # Add rows removed from validation set back into edx set
  removed <- anti_join(test_set, test_final)
  train_final <- rbind(train_set, removed)

  mu <- mean(train_final$rating)

  rmses_2[k,] <- sapply(lambdas, function(l){
    b_i <- train_final %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu)/(n()+l))
    b_u <- train_final %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - mu)/(n()+l))
    predicted_ratings <-
      test_final %>%
      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, test_final$rating))
  })
}

rmses_2
rmses_2_cv <- colMeans(rmses_2)
rmses_2_cv
qplot(lambdas,rmses_2_cv)
lambda <- lambdas[which.min(rmses_2_cv)]

mu <- mean(edx$rating)
b_i_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
b_u_reg <- edx %>%
  left_join(b_i_reg, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
predicted_ratings_6 <-
  validation %>%
  left_join(b_i_reg, by = "movieId") %>%
  left_join(b_u_reg, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
model_6_rmse <- RMSE(predicted_ratings_6, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Regularized Movie + User Effect M
odel",
                                     RMSE = model_6_rmse))
rmse_results
```

MOVIE + USER + WEEK

```
lambda = 0.75
week_avg_reg <- test_set %>%
  group_by(week) %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  summarise(n=n(), w_ui_reg = sum(rating - mu - b_i - b_u) / (lambda + n) )

w_ui_reg <- test_set %>%
  mutate(date = as_datetime(timestamp),
         week = round_date(date, unit="week")) %>%
  left_join(movie_avg, by="movieId") %>%
  left_join(user_avg, by="userId") %>%
  left_join(week_avg_reg, by="week") %>%
  .$w_ui_reg
# Now there's only 1 NA when using week effect. Replace NA with mu
w_ui_reg <- replace_na(w_ui_reg, mu)
y_hat <- mu + b_i + b_u + w_ui_reg
model_6_rmse <- rmse(test_set$rating, y_hat)



lambdas <- seq(0, 5, 0.25)
rmses <- sapply(lambdas, function(lambda){
  movie_avg_reg <- train_set %>%
    group_by(movieId) %>%
    summarise(n=n(), b_i_reg = sum(rating - mu) / (lambda + n) )

  user_avg_reg <- train_set %>%
    group_by(userId) %>%
    left_join(movie_avg, by="movieId") %>%
    summarise(n=n(), b_u_reg = sum(rating - mu - b_i) / (lambda + n) )

  b_i_reg <- test_set %>%
    left_join(movie_avg_reg, by="movieId") %>%
    .$b_i_reg

  b_u_reg <- test_set %>%
    left_join(user_avg_reg, by="userId") %>%
    .$b_u_reg

  y_hat <- mu + b_i_reg + b_u_reg

  return(rmse(y_hat, test_set$rating))
})
```

Results

```
rmse_results <- data.frame(Method = c("Simple Model",
                                      "Movie + User Effect",
                                      "Regularized Movie + User + Week Effect
"),
                           RMSE = c(model_1, model_2, model_3))
rmse_results %>% knitr::kable()
```

Conclusion

MovieLens is a classical dataset for recommendation system. In this project, the “Just the average” model only gives a RMSE of 1.0612, whreas the regularization of the different models has proved successful.