Project Submission: MovieLens

Bernhard Mayrhofer

January 2, 2019

Introduction

This report is part of the capstone project of the EdX course 'HarvardX: PH125.9x Data Science: Capstone'. Its goal is to demonstrate that the student acquired skills with the R programming language in the field of datascience to actually solve realworld problems.
The task is to analyse a dataset called 'Movielans' which contains millions of movieratings by users. The insights from this analysis are used to generate predictions of movies which are compared with the actual ratings to check the quality of the prediction algorithm.
One challenge of this project is the size of the dataset. It contains more than 10 million ratings which make it difficult for my copmuter to run higher-sophisticated machine learning algorithms like glm, or even neural nets on the dataset.

Summary

The report is split in three sections. First, the dataset is loaded and preformated for further analysis. Second, an exploratory datanalysis helps to understand the structure of the dataset. Finally, a machinelearning algorithm creates predictions which are then exported for a final test. A 'Penalized Root Mean Squared Error' approach was one of the few algorithms, my computer was able to apply on this large dataset. This algorithm achieved a RMSE of 0.86 and an accuracy of 36.05% on the validation set.

Penalized Least Squares Approach

Method Description

Due to the large dataset, an efficient method was needed to predict movie ratings based on an user id and a movie id. The penalized least squares approach is based on the mean movie rating. This average is adjusted for user-effects and movie-effects. Analysis shows that ratings from users who rate just a few movies and movies with a small number of total ratings 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. Onve a prediction is made, it has to be translated from a continuous number into a number from 0.5 to 5.0 in 0.5 steps. The so derived predicted values get compared with the actual valuse to calculate an accuracy value.

Step 1) Download MovieLens Data

The dataset 'movielens' gets split into a training-testset called 'edx' and a set for validation purposes called 'validation'.

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################

  # Note: this process could take a couple of minutes

  if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-pro
ject.org")
  if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.or
g")

  # MovieLens 10M dataset:
  # https://grouplens.org/datasets/movielens/10m/
  # http://files.grouplens.org/datasets/movielens/ml-10m.zip

  #To speed up data loading, the final result was already saved as 'movielens.csc'
  step <- 'load_data'#new_analysis

  if (step == 'new_analysis') {

    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/rat
ings.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(movieI
d))[movieId],
                                               title = as.character(title),
                                               genres = as.character(genres))

    movielens <- left_join(ratings, movies, by = "movieId")

    #Shortcut for testing purposes:
  } else {
    movielens <- read.csv("ml-10M100K/movielens.csv", row.names = 1)
  }

  # 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)

# Learners will develop their algorithms on the edx set
# For grading, learners will run algorithm on validation set to generate ratings

validation <- validation %>% select(-rating)
```

Step 2) Exploratory Data Analysis

This sections helps to understand the structure of the movilens dataset in order to use the insights for a better prediction of movie ratings.

Quiz

This analysis is part of the quiz in the capstone project.

Q1) How many rows and columns are there in the edx dataset?

```
#load(file='Workspace_Capstone_03_final.RData')
paste('The edx dataset has',nrow(edx),'rows and',ncol(edx),'columns.')
```

[1] “The edx dataset has 9000055 rows and 6 columns.”

Q2) How many zeros and threes were given in the edx dataset?

```
paste(sum(edx$rating == 0), 'ratings with 0 were given and',
sum(edx$rating == 3),'ratings with 3')
```

[1] “0 ratings with 0 were given and 2121240 ratings with 3”

Q3) How many different movies are in the edx dataset?

```
edx %>% summarize(n_movies = n_distinct(movieId))
```

There are 10677 movies

Q4) How many different users are in the edx dataset?

```
edx %>% summarize(n_users = n_distinct(userId))
```

There are 69878 users.

Q5) How many movie ratings are in cear of the following genres in the edx dataset?

```
drama <- edx %>% filter(str_detect(genres,"Drama"))
comedy <- edx %>% filter(str_detect(genres,"Comedy"))
thriller <- edx %>% filter(str_detect(genres,"Thriller"))
romance <- edx %>% filter(str_detect(genres,"Romance"))

paste('Drama has',nrow(drama),'movies')
paste('Comedy has',nrow(comedy),'movies')
paste('Thriller has',nrow(thriller),'movies')
paste('Romance has',nrow(romance),'movies')
```

[1] “Drama has 4151718 movies” [1] “Comedy has 2962038 movies” [1] “Thriller has 1485456 movies” [1] “Romance has 1312948 movies”

Q6) Which movie has the greatest number of ratings?

```
edx %>% group_by(title) %>% summarise(number = n()) %>%
  arrange(desc(number))
```

Pulb Fiction has the highest rating.

Q7) What are the five most given ratings in order from most to least?

```
head(sort(-table(edx$rating)),5)
```

4	3	5	3.5	2
2588430	2121240	1390114	791624	711422

Q8) True or False: In general, half star ratings are less common than whole star ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.).

```
table(edx$rating)
```

True

Data Analysis

```
str(movielens)
```

The movielens dataset has more than 10 million ratings. Each rating comes with a userId, a movieId, the rating, a timestamp and information about the movie like title and genre.

```
hist(movielens$rating,
     col = "#2E9FDF")

summary(movielens$rating)
```

Min.	1st Qu.	Median	Mean	3rd Qu.

Min.	1st Qu.	Median	Mean	3rd Qu.
0.500	3.000	4.000	3.512	4.000

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.

```
movielens$year <- as.numeric(substr(as.character(movielens$title),nchar(as.character
(movielens$title))-4,nchar(as.character(movielens$title))-1))

plot(table(movielens$year),
    col = "#2E9FDF")
```

More recent movies get more userratings. Movies earlier than 1930 get few ratings, whereas newer movies, especially in the 90s get far more ratings.

```
avg_ratings <- movielens %>% group_by(year) %>% summarise(avg_rating = mean(rating))
plot(avg_ratings,
    col = "#2E9FDF")
```

Movies from earlier decades have more volatile ratings, which can be explained by the lower frequence of movieratings. However, since the project is measured by the accuarcy, this volatility has to be taken into account.

Results

I tried a varietey of machinelearning algorithms on the testset. The challenge was to get the highest accuracy, which is measured as the number of exact matches of predicted ratings vs ratings of the validation set. The most promising algorithm was the penealized least squares approach.

Choose Optimal Penalty Rate 'Lambda'

```
#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))

  #ajdust 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,
    col = "#2E9FDF")

lambda <- lambdas[which.min(rmses)]
paste('Optimal RMSE of',min(rmses),'is achieved with Lambda',lambda)
```

The minimal RMSE of 0.856695227644159 is achieved with Lambda 0.5. Predictions will be done using this value.

Apply Lamda on Validation set for Data-Export

```r
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)

})
```

Generated predictions for validation dataset.

Export Predictions

```r
# Ratings will go into the CSV submission file below:

write.csv(validation %>% select(userId, movieId) %>% mutate(rating = pred_y_lse),
          "submission.csv", na = "", row.names=FALSE)
```

Conclusion

The aim of the project was to predict movieratings from a long list of rated movies. The size of the dataset restricted the machinlearning algorithms my computer was able to perform on the dataset. The penalized least squares approach was able to come up with ratings that are near the true ratings. However, accuracy is measured as absolute difference between the predicted value and the acutal value. The transformation from a continuous number to the actual rating did not result in a high overall accuarcy, although the prediction in terms of real numbers makes sense.