MovieLens Project Submission

Juan Carlos Cortez Villarreal

Preface

This is my submission for the first part of the Data Science: Capstone! course provided by HarvardX in association with edX.org. The objective of this document is to explain the procedure and present results from an approach to the MovieLens Movie Recommendation System Project.

1. Introduction

The MovieLens dataset contains millions of ratings of movies collected by GroupLens Research. This information allows us to train machine learning algorithms and produce a movie recommendation system. We can see the data as a matrix of user vs movies, and each element a rating. The idea then would be to train a machine learning algorithm to predict other ratings so that suggestions can be made from those predictions. The objective of this submission is to design and code an algorithm that gets an RMSE lower than 0.87750.

2. Analysis

In order to start our analysis, we first need to get the dataset, then we'll be able to get some insights on the data.

2.1. Download and parse the dataset

We first download the dataset.

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

Then we parse the ratings dataframe by just reading it from the ratings.dat file and adding the column names

```
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))
```

And we parse the movies dataframe, by reading it from the movies.dat file, adding the column names and casting the variables to the appropriate type.

```
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))
```

Finally we join both ratings and movies dataframes to get all the variables together

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

2.2. Split data into train and validation sets

We set the seed and partition the dataset in a 90/10 division between a train set called edx, and a temporal set

```
set.seed(1) # if using R 3.6.0: set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

Then we define the validation set as the entries in the temporal set that have it's users and movies also in the train set. After that, we put the rest of the entries in the train set

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

Finally, we delete the files and variables that we are no longer using

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

2.3. Analyze the edx dataset

In this section, we get some information about the data in the edx dataset.

2.3.1. About the dimensions

```
print(paste("edx dataset has", dim(edx)[1], "rows and", dim(edx)[2], "columns."))
```

```
## [1] "edx dataset has 9000055 rows and 6 columns."
```

2.3.2. About the number of movies given a certain rating

```
print(paste(sum(edx$rating==0), "movies were given a rating of 0."))
```

```
## [1] "0 movies were given a rating of 0."
```

```
print(paste(sum(edx$rating==3), "movies were given a rating of 3."))
```

```
## [1] "2121240 movies were given a rating of 3."
```

2.3.3. About the number of movies

```
print(paste("There are", length(unique(edx$movieId)), "unique movies in the edx dataset"))
```

```
## [1] "There are 10677 unique movies in the edx dataset"
```

2.3.4. About the number of users

```
print(paste("There are", length(unique(edx$userId)), "unique users in the edx dataset"))
```

```
## [1] "There are 69878 unique users in the edx dataset"
```

2.3.5. About the number of movies in certain categories

```
categories <- c("Drama", "Comedy", "Thriller", "Romance")
for (cat in categories) {
  print(paste("There are", sum(grepl(cat, edx$genres, fixed=TRUE)),
              "movies in the", cat, "category."))
}
```

```
## [1] "There are 3910127 movies in the Drama category."
## [1] "There are 3540930 movies in the Comedy category."
## [1] "There are 2325899 movies in the Thriller category."
## [1] "There are 1712100 movies in the Romance category."
```

2.3.6. The most rated movies

```
edx %>% group_by(title) %>%
  summarize(n = n()) %>% arrange(desc(n)) %>% top_n(5)
```

```
## Selecting by n
```

```
## # A tibble: 5 x 2
##   title                                n
##   <chr>                            <int>
## 1 Pulp Fiction (1994)              31362
## 2 Forrest Gump (1994)              31079
## 3 Silence of the Lambs, The (1991) 30382
## 4 Jurassic Park (1993)             29360
## 5 Shawshank Redemption, The (1994) 28015
```

2.3.7. The most given ratings

```
edx %>% group_by(rating) %>%
  summarize(n = n()) %>% arrange(desc(n))
```

```
## # A tibble: 10 x 2
##    rating       n
##     <dbl>   <int>
##  1    4   2588430
##  2    3   2121240
```

```
##  3    5   1390114
##  4    3.5  791624
##  5    2    711422
##  6    4.5  526736
##  7    1    345679
##  8    2.5  333010
##  9    1.5  106426
## 10    0.5   85374
```

2.3.8. We verify that ratings of full star (no fraction) are given much more than half star ratings

```
edx %>% mutate(startype = ifelse(rating %% 1 > 0, "Half-star", "Full-star")) %>%
  group_by(startype) %>%
  summarize(n = n()) %>% arrange(desc(n))
```

```
## # A tibble: 2 x 2
##   startype        n
##   <chr>       <int>
## 1 Full-star 7156885
## 2 Half-star 1843170
```

3. Method

There are many approaches to the implementation of the desired recommendation system. We'll pick the Penalized Least Squares because it offers good enough results without becoming too complex for the scope of this document.

The basic idea is to minimize the following equation:

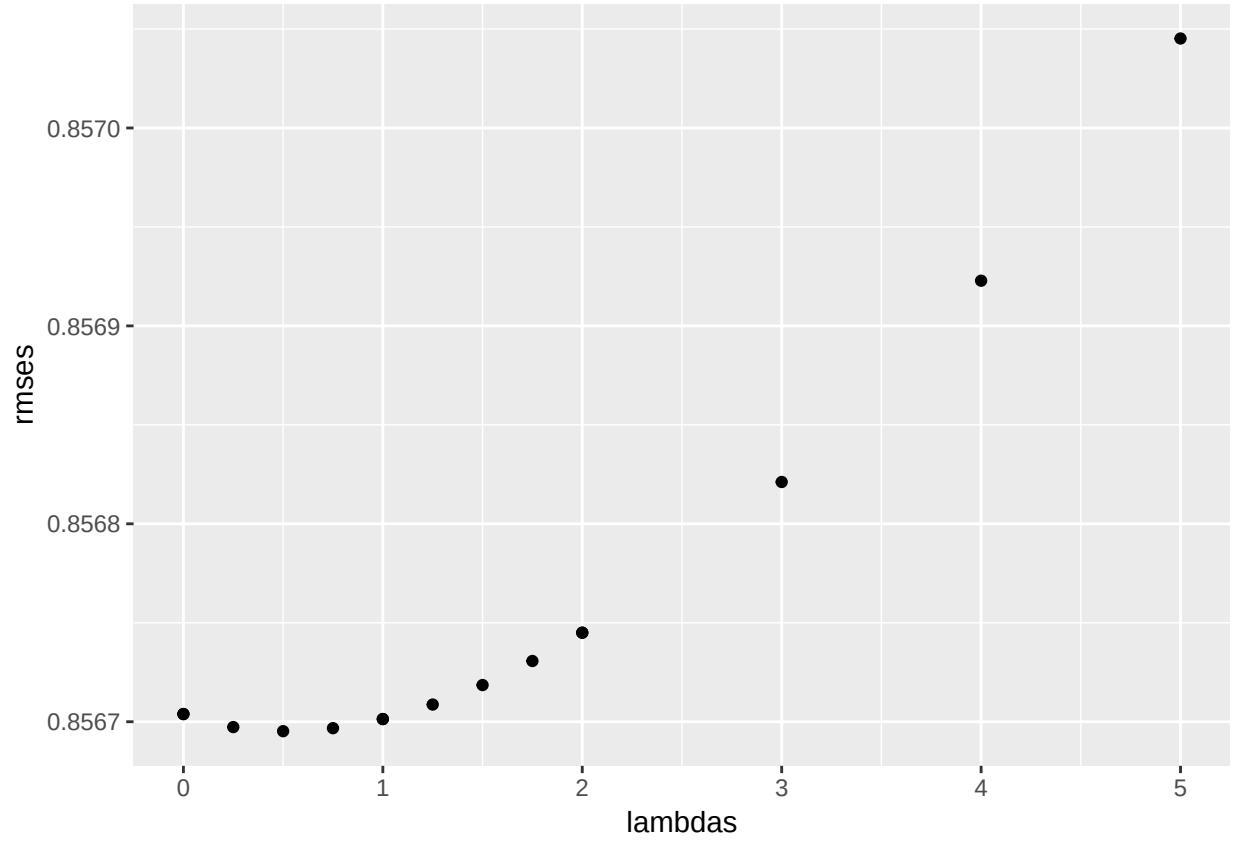
$$\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_i-b_u\right)^2+\lambda\left(\sum_i b_i^2+\sum_u b_u^2\right)$$

where
$y_{u,i}$ is our estimation of the rating for user u and movie i
μ is the average rating of the dataset
b_i is the movie effect or bias
b_u is the user effect or bias
λ is the penalty factor

Since λ is a tuning factor, we implement a cross-validation code to choose the optimal value
Note: lambda choices have been reduced after finding the value, to make the code run faster in subsequent runs

```
lambdas <- c( seq(0, 5, 1), seq(0, 2, 0.25) )

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

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
```

```
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <-
    edx %>%
    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, edx$rating))
})

qplot(lambdas, rmses)
```




```
print(paste("Lambda", lambdas[which.min(rmses)], "offers the lowest RMSE."))
```

```
## [1] "Lambda 0.5 offers the lowest RMSE."
```

4. Results

With the λ value optimized at 0.5, we can now run the algorithm on the validation set and find our results for the RMSE.

```
lambda <- 0.5
mu <- mean(edx$rating)

b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

validation_RMSE <- RMSE(predicted_ratings, validation$rating)
print(paste("Our algorithm has an RMSE of", validation_RMSE))
```

```
## [1] "Our algorithm has an RMSE of 0.86522255159723"
```

5. Conclusion

We followed the course's approach and managed to optimize lambda, which allowed us to reach an RMSE below 0.87750. We accomplished the objective of this submission.

For future work, other ways to improve on the solution would be to include the genre effect and add matrix factorization to include similar tendencies between groups of movies.

Bibliography

Irizarry, Rafael A. "Data Analysis and Prediction Algorithms with R." Introduction to Data Science, 22 Apr. 2019, rafalab.github.io/dsbook/.