A Simple Movie Recommendation System

Introduction

In this data driven world, recommendation systems have become extremely popular and are used by several platforms such as Netflix, Amazon, Flipkart etc. to suggest items of relevance to its users.

As the name suggests, a recommendation system is literally a filtering system that aims at predicting the prefercnce (or rating) a user would give to an item.

This report starts with a brief discussion of the approach that is being used here followed by a result, conclusion and further work.

Data Preparation and Summary

Let us first start by preparing the data codes that were we have obtained from this course. The data will also be divided into two unequal sets. The edx will be used as the train set to train the algorithms on and the validation set to test the accuracy of the algorithms.The codes are given below:

```r
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")

library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)

# 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"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")),
"\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
```

```r
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


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

# Validation set will be 10% of MovieLens data
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,]

# 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)
```

Now that we have the data, let us use descriptive statistics to try and understand the data in the training set.

```r
Number_of_users <- nrow(edx$userId)
Number_of_users

## NULL

mu_hat <- mean(edx$rating)
mu_hat

## [1] 3.512465

Number_of_movies <- length(unique(edx$movieId))
Number_of_movies

## [1] 10677
```

Method

First Model

The first model is built on the assumption that all movies have the same rating. The diffrences in the ratings are explained by random variation. Let μ be the true rating for all the movies by the users and let ϵ represent the independent errors. Then the expercted value of the ratings can be given by -

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

Let us write some codes to create this model and test its accuracy by calculating the root mean squared error (RMSE).

```r
#Function to calculate the root mean squared error
RMSE <- function(True_Ratings,Predicted_Ratings){
  sqrt(mean((True_Ratings-Predicted_Ratings)^2))
}

#LSE of the mean
mu <- mean(edx$rating)
First_model <- RMSE(validation$rating,mu)
First_model

## [1] 1.061202
```

Second Model

To improve the model further, we add three add to more terms in the equation above. An effect m_i to represent the movie effect, u_i to represent the user effect and g_i to represent the genre effect,

The movie effect is added due to the fact that some movies are rated better than the others and some poorer than the others. Movie effect m_i represensts the average rating that a movie 'i' has recieved.

Similarly, the same can be said for the users. Some snob users can be cranky raters. They tend to be extra critical. There are another set of cheery users that are pleased easily. To account for these effects, we add an extra term u_i which represents the average rating given by user 'i'.

The same can also be said for genres. Some genres are rated higher than the other genres. Since the genres here are represented as combinations, the effect g_i represents the average rating of a given genre combination 'i'.

Let us add this to the equation above and test its accuracy.

$$Y_{u,i} = \mu + \epsilon_{u,i} + m_i + u_i$$

```r
#Adding Movie Effect
movie_average <- edx %>% group_by(movieId) %>% summarize(m_i = mean(rating-
mu))

predicted_ratings <- validation %>% left_join(movie_average, by = 'movieId')
%>% pull(m_i) + mu


#Adding User Effect
user_average <- edx %>% left_join(movie_average, by = 'movieId') %>%
group_by(userId) %>% summarize(u_i = mean(rating-mu-m_i))
predicted_ratings2 <- validation %>% left_join(user_average, by = 'userId')
%>% left_join(movie_average, by = 'movieId') %>% mutate(pred = mu + u_i +
m_i) %>% pull(pred)

Second_model <- RMSE(predicted_ratings2,validation$rating)
Second_model

## [1] 0.8653488
```

It can be seen that that there is a huge improvement in the result. Let us see the problems in order to find a way to further improve the model. Since we have added user effects, let's look at the top 10 movies and the number of ratings they have recieved. Can it be improved further?

Third Model

The same can also be said for genres. Some genres are rated higher than the other genres. Since the genres here are represented as combinations, the effect g_i represents the average rating of a given genre combination 'i'.

The formula would then be:

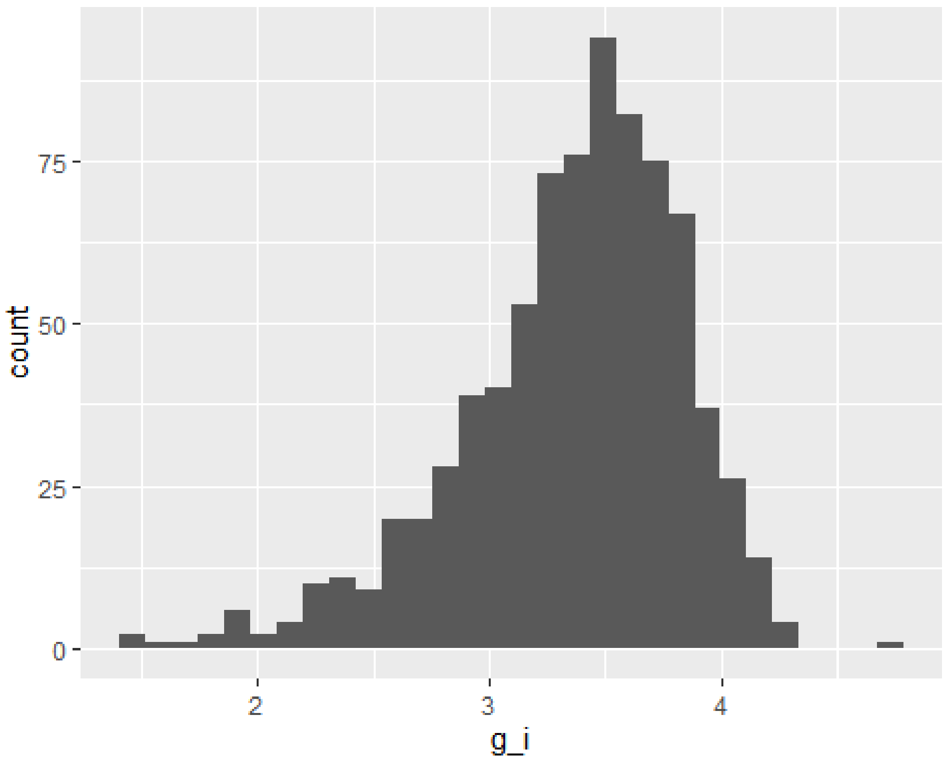
$$Y_{u,i} = \mu + \epsilon_{u,i} + m_i + u_i + g_i$$

Let us first check the significance of the genre effect before adding to the model.

```r
#Checking the significance of the genre effect
length(unique(edx$genres))

## [1] 797

edx %>% group_by(genres) %>% summarise(g_i = mean(rating)) %>%
filter(n()>=1)%>% ggplot(aes(g_i)) + geom_histogram(bins = 30)
```



```
#Adding Genre Effect
genre_average <- edx %>% left_join(user_average, by = 'userId') %>%
left_join(movie_average, by = 'movieId') %>% group_by(genres) %>%
summarize(g_ui = mean(rating-mu-m_i-u_i))
predicted_ratings3 <- validation %>% left_join(user_average, by = 'userId')
%>% left_join(movie_average, by = 'movieId') %>% left_join(genre_average, by
= 'genres') %>% mutate(pred2 = mu + u_i + m_i + g_ui) %>% pull(pred2)

Third_model <- RMSE(predicted_ratings3,validation$rating)
Third_model

## [1] 0.8649469
```

Fourth and Fifth Model

Regularisation

Let us try to find the mistakes in the current model to see if it can be improved. We will do this using only the movie effects m_i using the following codes -

```
#Regularisation

validation %>% left_join(movie_average, by = 'movieId') %>% mutate(residual =
rating - (mu+m_i)) %>% arrange(desc(abs(residual))) %>% slice(1:10) %>%
pull(title)
```

```
##  [1] "PokÃ©mon Heroes (2003)"           "Shawshank Redemption, The (1994)"
##  [3] "Shawshank Redemption, The (1994)" "Shawshank Redemption, The (1994)"
##  [5] "Godfather, The (1972)"            "Godfather, The (1972)"
##  [7] "Godfather, The (1972)"            "Usual Suspects, The (1995)"
##  [9] "Usual Suspects, The (1995)"       "Usual Suspects, The (1995)"

#top 10 and worst 10
validation %>% left_join(movie_average, by ='movieId') %>% mutate(residual =
rating - (mu+m_i)) %>% arrange(desc(abs(residual))) %>% slice(1:10) %>%
pull(title)

##  [1] "PokÃ©mon Heroes (2003)"           "Shawshank Redemption, The (1994)"
##  [3] "Shawshank Redemption, The (1994)" "Shawshank Redemption, The (1994)"
##  [5] "Godfather, The (1972)"            "Godfather, The (1972)"
##  [7] "Godfather, The (1972)"            "Usual Suspects, The (1995)"
##  [9] "Usual Suspects, The (1995)"       "Usual Suspects, The (1995)"

movie_titles <- edx %>% select(movieId,title) %>% distinct()
movie_average %>% left_join(movie_titles, by = 'movieId') %>%
arrange(desc(m_i)) %>% slice(1:10) %>% pull(title)

##  [1] "Hellhounds on My Trail (1999)"
##  [2] "Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)"
##  [3] "Shadows of Forgotten Ancestors (1964)"
##  [4] "Fighting Elegy (Kenka erejii) (1966)"
##  [5] "Sun Alley (Sonnenallee) (1999)"
##  [6] "Blue Light, The (Das Blaue Licht) (1932)"
##  [7] "Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo
peva) (1980)"
##  [8] "Human Condition II, The (Ningen no joken II) (1959)"
##  [9] "Human Condition III, The (Ningen no joken III) (1961)"
## [10] "Constantine's Sword (2007)"

movie_average %>% left_join(movie_titles, by = 'movieId') %>% arrange((m_i))
%>% slice(1:10) %>% pull(title)

##  [1] "Besotted (2001)"
##  [2] "Hi-Line, The (1999)"
##  [3] "Accused (Anklaget) (2005)"
##  [4] "Confessions of a Superhero (2007)"
##  [5] "War of the Worlds 2: The Next Wave (2008)"
##  [6] "SuperBabies: Baby Geniuses 2 (2004)"
##  [7] "Hip Hop Witch, Da (2000)"
##  [8] "Disaster Movie (2008)"
##  [9] "From Justin to Kelly (2003)"
## [10] "Criminals (1996)"
```

From the above, it can be seen that the top 10 and the top worst movies are obscure ones. Lets see the number of ratings that they have.

```
edx %>% count(movieId) %>% left_join(movie_average, by = 'movieId') %>%
left_join(movie_titles, by = 'movieId') %>% arrange(desc(m_i)) %>%
slice(1:10) %>% pull(n)

##  [1] 1 2 1 1 1 1 4 4 4 2

edx %>% count(movieId) %>% left_join(movie_average, by = 'movieId') %>%
left_join(movie_titles, by = 'movieId') %>% arrange((m_i)) %>% slice(1:10)
%>% pull(n)

##  [1]   2   1   1   1   2  56  14  32 199   2
```

As expected, the obscure movies had an insignificant number of ratings. This means - larger estimates of m_i.

To account for this we use a method called regularisation. It allows us to penalize large estimates that are formed using small sample sizes.

Penalized least squares

The idea of penalised regression is to control the total variability of the movie effects - $\sum m_i^2$. Therefore we add this to the equation to sum of squares equation -

$$\sum(y_{u,i} - \mu - m_i)^2 + \lambda \sum m_i^2$$

The second term is the penalty that gets larger when many m_i are large. The values of m_i can be obtained using the following formula -

$$m_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu)$$

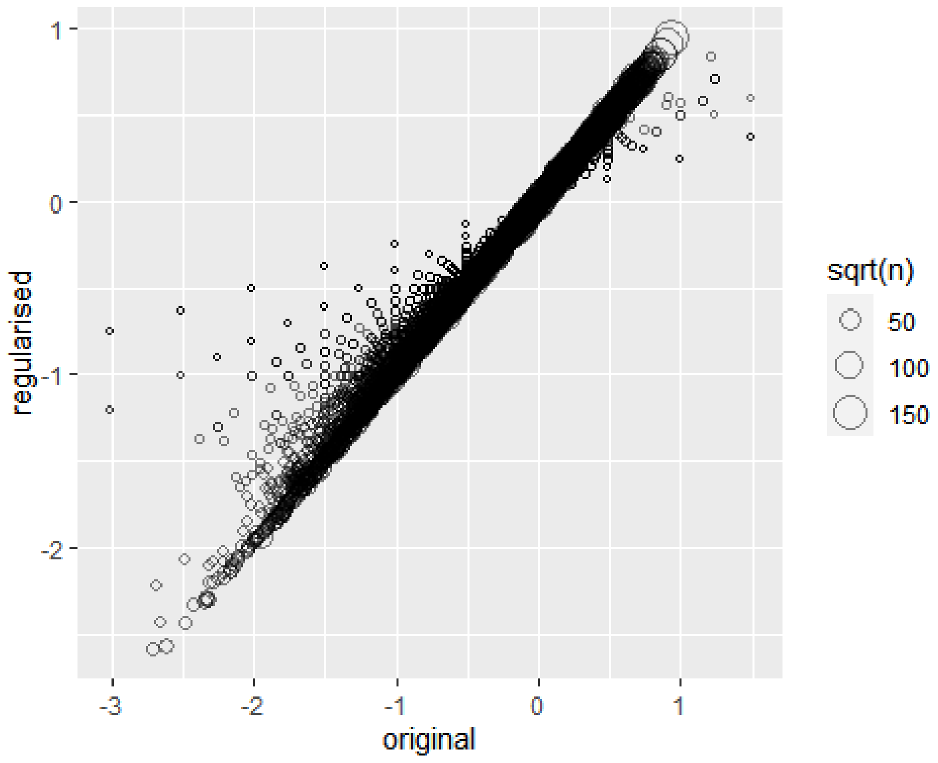
where, n_i is the number of ratings made for movie i.

Lets write a code to calculate the regularised estimates of m_i and to plot graph to visulaize how the estimates shrink. We will also look at the top 10 movies based on m_i to see how much this method actually works.

```
lambda <- 3

movie_reg_averages <- edx %>% group_by(movieId) %>% summarize(m_i =
sum(rating-mu)/(n()+lambda), n_i = n())

tibble(original = movie_average$m_i, regularised = movie_reg_averages$m_i, n
= movie_reg_averages$n_i) %>% ggplot(aes(original, regularised, size =
sqrt(n))) + geom_point(shape=1, alpha=.5)
```



```
edx %>% count(movieId) %>% left_join(movie_reg_averages, by = 'movieId') %>%
left_join(movie_titles, by = 'movieId') %>% arrange(desc(m_i)) %>%
slice(1:10) %>% pull(title)

##  [1] "Shawshank Redemption, The (1994)"
##  [2] "Godfather, The (1972)"
##  [3] "Usual Suspects, The (1995)"
##  [4] "Schindler's List (1993)"
##  [5] "More (1998)"
##  [6] "Casablanca (1942)"
##  [7] "Rear Window (1954)"
##  [8] "Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)"
##  [9] "Third Man, The (1949)"
## [10] "Double Indemnity (1944)"
```

It is visible from the list that the model seems to working a good deal. We can assume that the worst 10 list would also be satisfactory list.

Let us calculate the accuracy to quantify the improvement.

```
predicted_ratings4 <- validation %>% left_join(movie_reg_averages, by =
'movieId') %>% mutate(pred = mu + m_i) %>% pull(pred)

Fourth_model <- RMSE(predicted_ratings4, validation$rating)
Fourth_model

## [1] 0.9438538
```

The model is only a little better than the model with the user effects only.

Let us add the user effects and genre effects u_i & g_i as well.

The formula will now be -

$$\sum(y_{u,i} - \mu - m_i)^2 + \lambda(\sum m_i{}^2 + \sum u_i{}^2 + \sum g_i{}^2)$$

```
m_i <- edx %>% group_by(movieId) %>% summarise(m_i = sum(rating-
mu)/(n()+2.5))
u_i <- edx %>% left_join(m_i, by = 'movieId') %>% group_by(userId) %>%
summarise(u_i = sum(rating-m_i-mu)/(n()+2.5))
g_i <- edx %>% left_join(m_i, by = 'movieId') %>% left_join(u_i, by =
'userId') %>% group_by(genres) %>% summarise(g_i = sum(rating-m_i-u_i-
mu)/(n()+2.5))

predicted_ratings5 <- validation %>% left_join(m_i, by = 'movieId') %>%
left_join(u_i, by = 'userId') %>% left_join(g_i, by = 'genres') %>%
mutate(pred = mu + m_i + u_i + g_i) %>% pull(pred)

fifth_model <- RMSE(predicted_ratings5, validation$rating)
```

Result

The result of all the models are summarised below.

```
## # A tibble: 5 x 2
##   MODEL                           RMSE_RESULTS
##   <chr>                                  <dbl>
## 1 MEAN                                   1.06
## 2 USER_EFFECTS                           0.865
## 3 MOVIE_GENRE_EFFECTS                    0.865
## 4 REGULARISED_MOVIE                      0.944
## 5 REGULARISED_MOVIE_USER_GENRE           0.865

## [1] 0.8645493
```

We can see that the fifth model, i.e, the regularised movie, user and genere effect model is the best performing model as it has the lowest RMSE - 0.86454

Conclusion

We built a very simple model by only taking into account the user effects, movie effects and the genre effects. But what we did not take into account was the fact that groups of movies and groups of users also have a similar rating pattern. Matrix factorization is a good method to discover these patterns to implement to the model. There are famous approaches such as the content based similarity apporach and the user based similarity apporach. These can also be compared to find best models.