Movie recomendation report

wasu wisanuyothin

17/6/2563

1. Introduction

Nowadays, most of streaming services always recomend movie, video clip or music for you. The recomendation system is created to make user find more relevent stuff and have more convenient using sites. The recomendation system also help the company target the product to each consumer easier, hence lead to increased in sales and profit.

In 2006, netflix created a challenge that will give 1 million dollars for people who make a better recomendation system. Movie recomendation system create from the user data that rate movies that they watched. The ratings range between 0 and 5. 0 means the user don't like the movie and 5 means the user like this movie. The goal of the system is to recomend the movie that the system think the user will like by using previous movie ratings.

In this report, the movie recomendation system is created from MovieLens 10M dataset. The goal of this project is to reduce the root mean square error(RMSE) lower than 0.86490. RMSE is the error from prediction rating compare to the actual rating of user, more detail of RMSE claculation will be in 1.2 model evaluation section.

1.1 Dataset

The dataset that we use is MovieLens 10M dataset.GroupLens research lab had generate database with over 20 million ratings for over 27,000 movies by more than 138,000 users. MovieLens 10M dataset is a subset from this database, consisted of 10 million ratings applied to 10,000 movies by 72,000 users.[1]

1.2 Model evaluation

To evaluate our model, we have to comapre the predicted rating with the actual rating. The diffrences between predicted raating and actual rating is called loss function. One of the most common loss function is RMSE. RMSE or root mean square error is defined by this formula:

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

where N is the number of ratings, $y_{u,i}$ is the rating of movie i by user u and $\hat{y}_{u,i}$ is the prediction of movie i by user u. The goal of this project model is to minimize RMSE to lower than 0.86490.

[1] https://grouplens.org/datasets/movielens/10m/

1.3 process of study

4 main steps of this project is:
1. Data preparation: this step included downloading data, seperate data into edx and validation set.
2. Data visualization: view the structure of data and find correlation between variables.
3. Model creation: create model to predict rating from data
4. evaluation: Evaluate model using RMSE
5. Summarize and report: find the best model from evaluation and report

2. Method and Analysis

2.1 Data preparation

In this part we will download MovieLens 10M dataset and seperate into 2 dataset. The first dataset is named edx, accounted for 90 percents of the data in MovieLens 10M dataset. edx dataset is use to visualize, analysis and build prediction model. The second dataset is called validation, accounted for another 10 percents in MovieLens 10M dataset. validation dataset can't be use in any other thing accept from calculating the RMSE in the final model. We will treat validation dataset as an unknown data in real life situation that our model have to be used with. In this section, the code that we used to create dataset as we had describe are shown below:

```
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"))

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")

# 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)
```

To tune and evaluate model during model creation process, we will divide edx dataset into train and test set by 80 percents and 20 percents accordingly.

```
#Seperate edx data into train and test set
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,] #train
temp2 <- edx[test_index,] #test

# Make sure userId and movieId in test set are also in train set
test_set <- temp2 %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from test set back into train set
removed <- anti_join(temp2, test_set)
train_set <- rbind(train_set, removed)
```

2.2 Data visualization

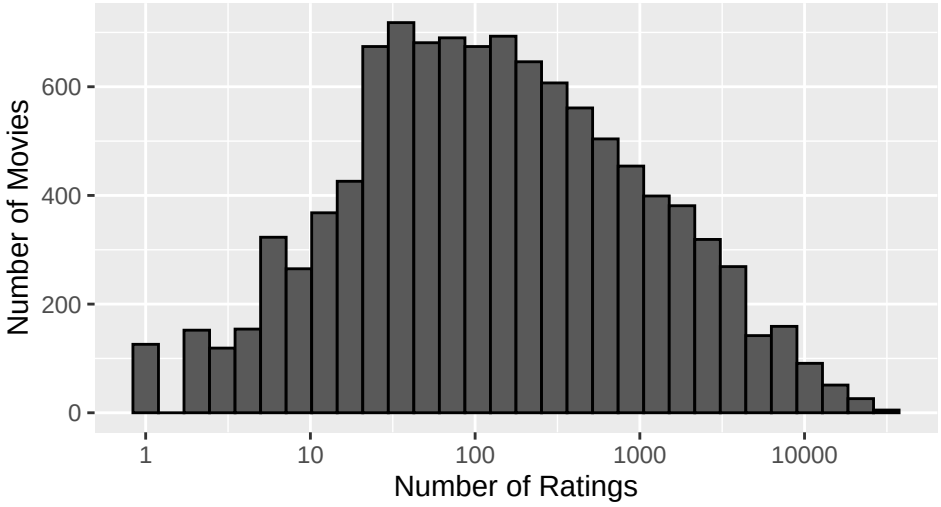
First, we will look at the structue of the data. The data in edx consisted of 9,000,055 observation and 6 column. The columns are userId, movieId, rating, timestamp, title and genres, as shown below.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

Next we will look at the number of rating among all the movies. As we can see from figure 1, some movie have more than 10,000 ratings and some movies got only one rating. As we known, some movie are blockbuster and a lot of people have watched it, so it will have a large amount of number of rating. On the other hand, some movies are indy movies or didn't get famous so lower people had watch it, leading to lower number of

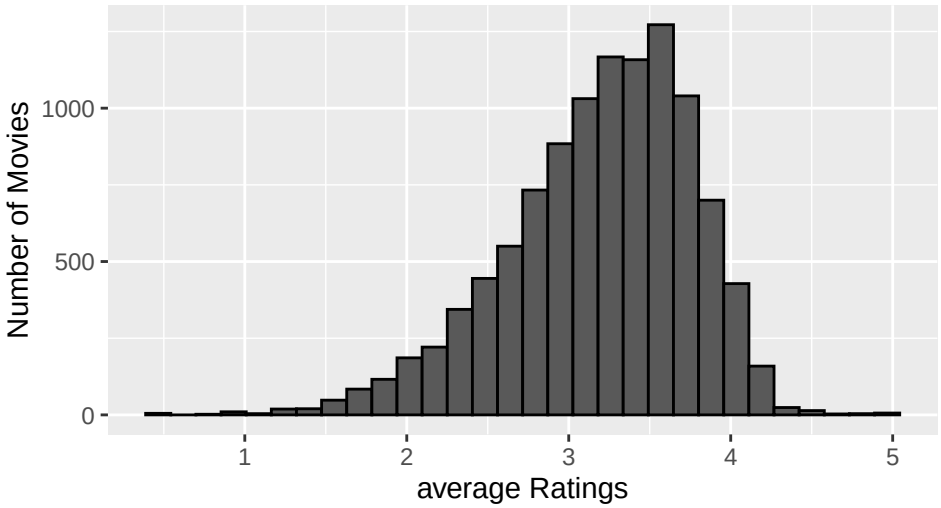
rating.

Fig 1. Distribution of number of rating across all movies



From our experiences, we know that some movies are good and some movies are bad. if we plot the average rating across all users across all the movies, we can see that the plot look like normal distribution. The difference of average rating in difference movie will be use in so called movie effect in the model creation section.

Fig 2. Distribution of average rating across all movies



The number of rating from users are distributed in a right skewed distribution as shown in figure 3. This means that some user are more active than another. we can see that some people have rate more than a thousand of movie while majority of people rate less than 100 movies. In figure 4, we can also see that some people are more harsh than others and rate lower score across all movie. In the opposite, some might rate movie higher than it should get. This varation cause user effect that will be used in model creation section.

Fig 3. Distribution of number of movie rated across all u

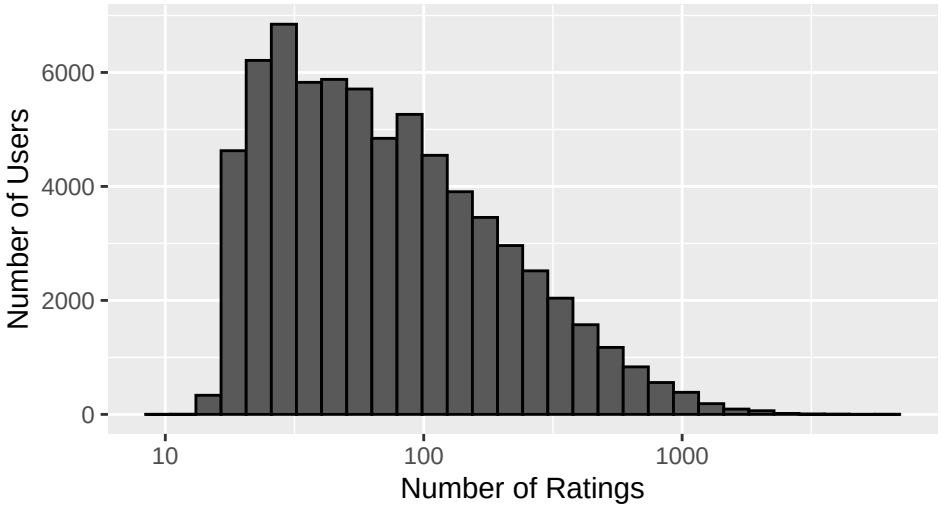
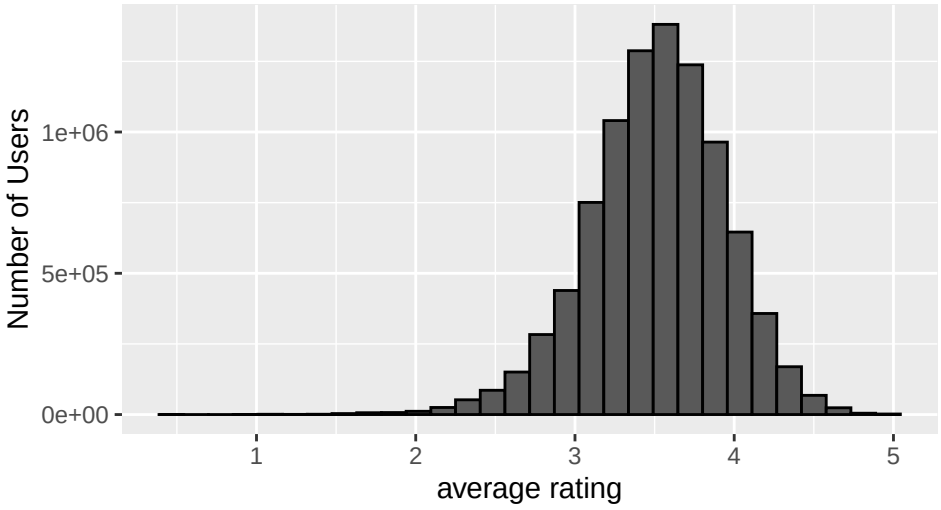


Fig 4. Distribution of rating across all users



2.3 Model creation

2.3.1 Mean model

The first model that we are going to use is calculating the mean of all rating. We will take an average of all the rating in our data and will predict the average rating for every movie that we are going to predict.

2.3.2 Linear model

This model will calculate the prediction rating from linear equation. From mean model, we know that every movie ratings are deviated from the average rating. We start by creating the first formula like this:

$$\hat{y}_{u,i} = \mu + \epsilon_{u,i}$$

Where $\hat{y}_{u,i}$ is prediction rating, μ is rating average and $\epsilon_{u,i}$ is error distribution.
From figure 2 and our experience, we know that some movie are good and some are bad. This reason that

we called movie effect have to explain some of the error distribution in the first formula. We defind b_i as the movie effect. Our linear formula can be written like this:

$$\hat{y}_{u,i} = \mu + b_i + \epsilon_{u,i}$$

To calculate b_i as quick and as easy as possible, we will find b_i from this equation:

$$\hat{b}_i = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{\mu})$$

Another factor that can explain the error is user effect. From figure 4, we can see that some user give a very high rating for all movies and some user tend to give low rating for all the movie that they rated. The new linear formula after adding user effect is:

$$\hat{y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

The calculation of b_u is:

$$\hat{b}_u = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{b}_i - \hat{\mu})$$

2.3.3 Regularization

From our linear model, the highest and lowest prediction rating are the movies that have very low number of rating. As we can see from below.

Top 5 best predicted rating movies

```
## # A tibble: 5 x 3
##   title                                   b_i     n
##   <chr>                                 <dbl> <int>
## 1 Hellhounds on My Trail (1999)          1.49     1
## 2 Shanghai Express (1932)                1.49     1
## 3 Shadows of Forgotten Ancestors (1964)  1.49     1
## 4 Fighting Elegy (Kenka erejii) (1966)   1.49     1
## 5 Sun Alley (Sonnenallee) (1999)         1.49     1
```

Top 5 worst predicted rating movies

```
## # A tibble: 5 x 3
##   title                                b_i     n
##   <chr>                              <dbl> <int>
## 1 Besotted (2001)                    -3.01     2
## 2 Hi-Line, The (1999)                -3.01     1
## 3 Grief (1993)                       -3.01     1
## 4 Accused (Anklaget) (2005)          -3.01     1
## 5 Confessions of a Superhero (2007) -3.01     1
```

Because of this reason, we have to use regularization to reduce this error. The prediction of movie that have very few user ratings can't be trust. So we will penalize the prediction of small sample size movies by using λ that will be the penalizing term. The new formula to calculate b_i and b_u are:

$$\hat{b}_i(\lambda) = \frac{1}{n_i + \lambda}\sum_{i=1}^{n_i}(y_i - \hat{\mu})$$

Where n_i is the number of ratings made for movie i.

$$\hat{b}_u(\lambda) = \frac{1}{n_u + \lambda} \sum_{i=1}^{n_u} (y_i - \hat{b}_i - \hat{\mu})$$

Where n_u is number of rating from user u. λ is a tuning parameter. When n is very low compare to λ, λ will bring $\hat{b}_u$ and $\hat{b}_i$ closer to zero. To chose the best value for λ, we will perfrom cross validation and choose the λ that has lowest RMSE.

2.3.4 Matrix factorization

As stated by the winner of netflix chllenge, matrix factorization have been the best way to reduce RMSE. Matrix factorization take accounted for the idea that similar groups of movies and similar groups of users have similar rating pattern of rating. The idea of matrix factorization is to create a matrix with i movies as the column, u users as the row and data that stored is residue that come from rating subtracting by row and column mean. So we will get matrix $R_{u \times i}$. Then we will make matrix R into 2 lower dimension matrix, $P_{k \times u}$ and $Q_{k \times i}$. The formula of matrix R, P and Q is:

$$R_{u \times i} = P^t_{k \times u} \times Q_{k \times i}$$

The recosystem package help making the matrix less memory consuming and help decomposing the matrix. The process of solving the matrices P and Q is referred to as model training, and the selection of penalty parameters is called parameter tuning.[2]

3. Model cretion and result of evaluation

3.1 Mean model

The code to calculate mean and find RMSE are shown below.

```
# Overall rating for all movie across all user
mu_hat <- mean(train_set$rating)
RMSE(mu_hat,test_set$rating)
```

```
##   method     RMSE
## 1   Mean 1.060704
```

We can see that the RMSE is about 1. That means our prediction rating is miss by 1 star out of 5 stars from actual rating from user. RMSE is high as we expected by using only the mean to predict all the movies.

3.2 linear model

The first equation that we will build model on is using only movie effect, the equation is shown below. The code that bulid and evaluated is shown below.

$$\hat{y}_{u,i} = \mu + b_i + \epsilon_{u,i}$$

[2]https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html

```r
#Movie effect
movie_avg <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating-mu_hat))
RMSE(test_set %>%
       left_join(movie_avg, by = "movieId") %>%
       mutate(pred = mu_hat + b_i) %>%
       pull(pred),test_set$rating)
```

```
##         method      RMSE
## 1         Mean 1.0607045
## 2 Movie effect 0.9437144
```

We can see that after we applied movie effect in to the equation, RMSE has reduce from around 1 to around 0.9. Another effect that we can add to this model is the user effect that we had describe before. By adding the user effect the prediction model will have this formula:

$$\hat{y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

```r
#user effect
user_avg <- train_set %>%
  left_join(movie_avg,by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u=mean(rating-mu_hat-b_i))
RMSE(test_set %>%
       left_join(movie_avg, by = "movieId") %>%
       left_join(user_avg, by = "userId") %>%
       mutate(pred = mu_hat + b_i + b_u) %>%
       pull(pred),test_set$rating)
```

```
##                  method      RMSE
## 1                  Mean 1.0607045
## 2         Movie effect 0.9437144
## 3 Movie + User effects 0.8661625
```

We can see that RMSE had reduced from 0.94 to 0.86 after applying user effect with the movie effect. But we still can't reach our goal of RMSE lower than 0.86490.

3.3 Regularization

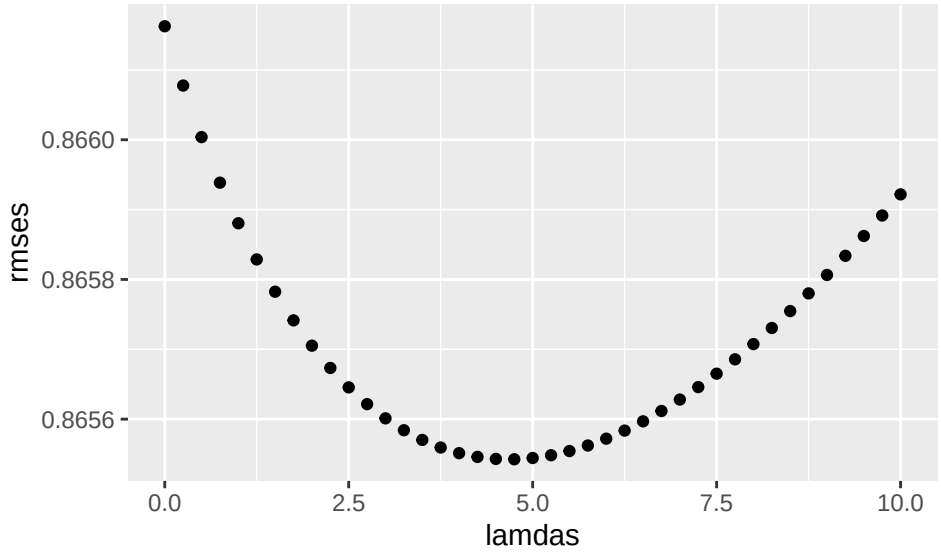
As we have mention before in section 2.3.3, the small amount of sample size can lead to error in our prediction model. To make a better RMSE and eliminate this problem, we use λ for a tuning parameter to regularize the prediction that come from low sample size. So our movie and user effect when we applied regularization become:

$$\hat{b}_i(\lambda) = \frac{1}{n_i + \lambda} \sum_{i=1}^{n_i} (y_i - \hat{\mu})$$

$$\hat{b}_u(\lambda) = \frac{1}{n_u + \lambda} \sum_{i=1}^{n_u} (y_i - \hat{b}_i - \hat{\mu})$$

To build this model, we have to select the best λ first by using cross validation as shown below. Then we will apply λ that has the lowest RMSE to the model:

```r
#Cross validation to choose lampda for regularization
lamdas <- seq(0,10,0.25)
rmses <- sapply(lamdas,function(l){

  mu <- mean(train_set$rating)

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

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

  predicted_ratings <- test_set %>%
    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_set$rating))
})
qplot(lamdas,rmses)
lamdas[which.min(rmses)]

#Apply lampda that has lowest rmse
b_i <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_hat)/(n()+lamdas[which.min(rmses)]))
b_u <- train_set %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu_hat)/(n()+lamdas[which.min(rmses)]))
predicted_ratings <-
  test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  .$pred
RMSE(predicted_ratings, test_set$rating)
```




```
##                 method      RMSE
## 1                 Mean 1.0607045
## 2          Movie effect 0.9437144
## 3 Movie + User effects 0.8661625
## 4       Regularization 0.8655426
```

As we can see from the RMSE of regularization model, RMSE have reduce from 0.866 to 0.865 from previous model.

3.4 matrix factorization

In matrix factorization, we have to turn train set into user-movie matrix. But using normal way like using spread() and as.matrix() will consume a lot of memory that normal computer can't handle. To avoid this problem, recosystem package have function that store the data into harddisk. The usage of recosystem consisted with the following steps:
1. Turn data into the format that can be use with function in recosystem by data_memory()
2. Create model object using Reco()
3. Tuning can be done by tune(). The parameter that we tuning are parameter for L1,L2 regularization, learning rate and iterations
4. Train the model using train() with the best tuning parameter
5. Use predict() to compute the prediction value
The code that use to create matrix factorization are shown below:

```
library(recosystem)

#convert train and test set to input of recosystem
reco_train <- with(train_set, data_memory(user_index = userId,
                                          item_index = movieId,
                                          rating = rating))
reco_test <- with(test_set, data_memory(user_index = userId,
                                        item_index = movieId,
                                        rating = rating))
```

```
#create model object
r_train <- Reco()

#tune pararmeter by cross validation
opts_train <- r_train$tune(reco_train,opts = list(dim      = c(10L, 20L, 30L, 40L),
                                    costp_l1 = c(0, 0.1),
                                    costp_l2 = c(0.01, 0.1),
                                    costq_l1 = c(0, 0.1),
                                    costq_l2 = c(0.01, 0.1),
                                    lrate    = c(0.01, 0.1, 0.2),
                                    nthread  = 4,
                                    niter = 10)
)

#train model using best tuning parameter
r_train$train(reco_train, opts = c(opts_train$min, nthread = 4, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       1.0006  1.0146e+007
##    1       0.8795  8.0974e+006
##    2       0.8448  7.4974e+006
##    3       0.8214  7.1275e+006
##    4       0.8037  6.8682e+006
##    5       0.7895  6.6837e+006
##    6       0.7774  6.5379e+006
##    7       0.7670  6.4155e+006
##    8       0.7581  6.3209e+006
##    9       0.7500  6.2375e+006
##   10       0.7430  6.1678e+006
##   11       0.7367  6.1068e+006
##   12       0.7308  6.0550e+006
##   13       0.7255  6.0095e+006
##   14       0.7206  5.9677e+006
##   15       0.7160  5.9300e+006
##   16       0.7118  5.8958e+006
##   17       0.7079  5.8645e+006
##   18       0.7042  5.8382e+006
##   19       0.7008  5.8122e+006
```

```
#make prediction for validation set
y_hat_train <-  r_train$predict(reco_test, out_memory())

#calculate rmse
rbind(results,data.frame(method="Matrix factorization",RMSE=RMSE(test_set$rating, y_hat_train)))
```

```
##                  method      RMSE
## 1                  Mean 1.0607045
## 2         Movie effect 0.9437144
## 3 Movie + User effects 0.8661625
## 4       Regularization 0.8655426
## 5 Matrix factorization 0.7884045
```

we can see that using matrix factorizaiton give us the lowest RMSE in all of our model. RMSE has reduce significantly from 0.865 to 0.788.

After we known that matrix factorization is the best performing model, we will create new matrix factorization with entire edx dataset to have more data and improve our model. We then calculate RMSE of the new matrix factorization model against validation set.

```
library(recosystem)
#convert edx to input of recosystem
reco_edx <-  with(edx, data_memory(user_index = userId,
                                   item_index = movieId,
                                   rating = rating))

#create model object
r <- Reco()

#tune pararmeter by cross validation
opts <- r$tune(reco_edx,opts = list(dim      = c(10L, 20L, 30L, 40L),
                                    costp_l1 = c(0, 0.1),
                                    costp_l2 = c(0.01, 0.1),
                                    costq_l1 = c(0, 0.1),
                                    costq_l2 = c(0.01, 0.1),
                                    lrate    = c(0.01, 0.1, 0.2),
                                    nthread  = 4,
                                    niter = 10)
)

#train model using best tuning parameter
r$train(reco_edx, opts = c(opts$min, nthread = 4, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9756  1.2102e+007
##    1       0.8714  9.8746e+006
##    2       0.8359  9.1494e+006
##    3       0.8121  8.7048e+006
##    4       0.7955  8.4199e+006
##    5       0.7827  8.2124e+006
##    6       0.7722  8.0538e+006
##    7       0.7637  7.9309e+006
##    8       0.7565  7.8317e+006
##    9       0.7502  7.7536e+006
##   10       0.7446  7.6850e+006
##   11       0.7396  7.6249e+006
##   12       0.7349  7.5764e+006
##   13       0.7306  7.5291e+006
##   14       0.7265  7.4860e+006
##   15       0.7227  7.4487e+006
##   16       0.7192  7.4159e+006
##   17       0.7160  7.3860e+006
##   18       0.7129  7.3568e+006
##   19       0.7101  7.3324e+006
```

```
#make prediction for validation set
reco_val  <-  with(validation, data_memory(user_index = userId,
                                           item_index = movieId,
                                           rating = rating))
```

```r
y_hat <-  r$predict(reco_val, out_memory())

#calculate rmse
data.frame(method="Matrix factorization",RMSE=RMSE(validation$rating, y_hat))
```

```
##                  method      RMSE
## 1 Matrix factorization 0.7798942
```

Using matrix factorization model to predict rating in validation set has RMSE of 0.780, which is lower than our goal 0.86490 . This model also significantly reduce RMSE compare to others previous model.

4. Summary

In this study, we have built 4 model. The first model is mean model. In this model. we predict all rating the same as average of rating in train set. The RMSE in this model is 1.0607045, which is the worst that we get across all model. The second model is linear model. We use movie effect that come from the fact that some movie are good and some are bad to explain the error that happen in the first model. We reduced the RMSE to 0.9437144. Then we use user effect that some user are more harsh and give rating lower than it should be and some user give rating hiher than it should be. After applying the user effect, RMSE had reduced to 0.8661625. When we select the best and worst prediction rating from the last model, we can see that these movies have small number of rating. The thrid model was built to account for this error. We use λ to penalize the small sample size and choose the best λ via cross validation. The RMSE that we got from regulariztion model is 0.8655426. For the final model, we use matrix factoriztion. We convert data into user-movie matrix and decompose it to lower dimension matrix using function from recosystem. The RMSE that we got from this model is substantially lower than other model with value of 0.7802432.

4.1 Limitation

The first limitation is in the calculation of matrix factoriztion. This process take a lot of cpu and memory of computer. It also take a large amount of time just to run it one time. Therefore, this model might not be efficient enough if we have to run it many times or often.
The second limitation is due to lack of predictor that we use in creating model. The genre of movies and temporal effect haven't been accounted for in this study.

4.2 Future work

Other common model that were seen to improve movie recomendation model are Gradient Boosted Decision Trees. This method have been use by winner of netflix challenge. Other effect that we can improve our linear model is to incoporate temporal effect and different kinds of genre to make a better prediction.

Reference

1. Irizarry R. Introduction to data science.
2. Qiu Y. recosystem: Recommender System Using Parallel Matrix Factorization [Internet]. Cran.r-project.org. 2020 [cited 17 June 2020]. Available from: https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html