Movie Rating Prediction Project: Movie Lens Dataset

HarvardX - PH125.9x Data Science: Capstone Course

Mauricio Rabelo Soares

14 agosto, 2022

Introduction

A recommendation system, is a subclass of information filtering system that provide suggestions for items that are most pertinent to a particular user. Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, or what movies to watch. [1] The goal of this project is to create a movie recommendation system, which the RMSE < 0.86490, using all the tools we have learn throughout the multi-part course in HarvardX's Data Science Professional Certificate series. As presented in the Machine Learning course [2] [3], a recommendation systems for a movie use ***ratings*** $(\boldsymbol{y_{u,i}})$ that ***users*** $(\boldsymbol{u})$ have given to some items, or ***movies*** $(\boldsymbol{i})$, to make specific recommendations to specific user. In this model, movies for which a high rating is predicted for a given user are then recommended to that user. For this challenge we going to use a subset of dataset provided by the GroupLens. In our case the subset of this data is provided by MovieLens 10M movie ratings. This a stable benchmark dataset with **10 million ratings** applied to **10,000 movies** by **72,000 users**.

Methods

The methods section explains the process and techniques used in this project, in the first part, then explains the data cleaning used to extract and clean the data, in the second part. In the third part of this section we present the data exploration and visualization of the data to gain some insights. The fourth, and last part of this section, we show the modeling approach used in this project.

The process and techniques used

Recommendation systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer. The recommendations system build in this project are made using the same process and techniques used by the winners of the Netflix challenges[4], which is presented in Chapter 34 Large datasets at Irizarry (2019) [5], and some new approaches[Qiu et al. (2021)][6][7]

The process begin with the download, cleaning and tidy the data. After the data cleaning we explore the data to gain some insight trough visualization and selected table. This insights is the basis of the modelling approach of this project, that starts building the simplest possible recommendation system: predict the same

[1]https://en.wikipedia.org/wiki/Recommender_system

[2]HarvardX - PH125.8x Data Science: Machine Learning https://www.edx.org/course/data-science-machine-learning

[3]34.7 Recommendation systems - https://rafalab.github.io/dsbook/large-datasets.html#recommendation-systems

[4]https://en.wikipedia.org/wiki/Netflix_Prize

[5]https://rafalab.github.io/dsbook/large-datasets.html#fnref112

[6]https://github.com/Airborne737/MovieLens_Harvard

[7]https://rpubs.com/tarashnot/recommender_comparison

rating for all movies regardless of user, and ends with the `recosystem`. The `recosystem` is a recommendation system that use parallel matrix factorization, the product of two matrices of lower dimensions, $P_{n \times k}$ and $Q_{n \times k}$ were: P - user matrix; Q - movie matrix.

Data cleaning

For this project we going to use a subset of dataset provided by the GroupLens. In our case the subset of this data is provided by MovieLens 10M movie ratings. This a stable benchmark dataset with **10 million ratings** applied to **10,000 movies** by **72,000 users**. This version of the dataset was released in January of 2009[8]. The code below load the library that we need, then download the file from GroupLens site and read the file. After read the file the code create the data frame `movielens` and remove the temporary file.

```r
# Load library
library(caret)
library(data.table)
library(fields)
library(tidyverse)
library(knitr)
library(kableExtra)
library(grid)
library(ggplot2)
library(lattice)
library(gridExtra)
library(recosystem)
```

```r
options(timeout=100)
#create tempfile and download
dl <- tempfile()
download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

#read the file and give name to columns
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")

#create data frame movielens
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")

#remove temporary files
rm(dl, ratings, movies)
```

Data exploration and visualization

The exploration and visualization of the data provide insightful information about the users, the movies and ratings. The first 5 rows of the dataset that we use in this project is presented in table 1. The columns `movieId`, `userId` and `rating` are the variable of interest.

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

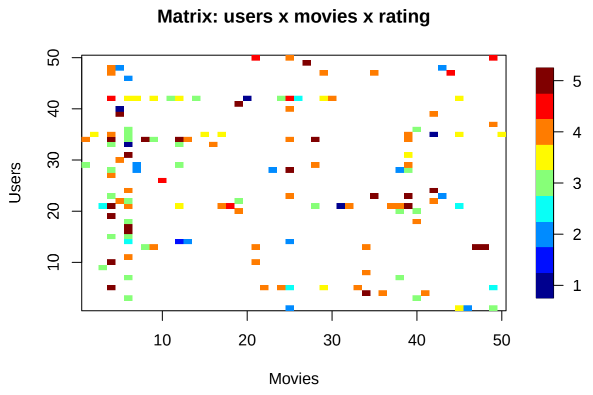
```r
knitr::kable(head(movielens %>% as_tibble(),5),
             caption = "The first 5 rows of the dataset, movielens",
             align = "cccccc",
             position = "h") %>%
  kable_styling(latex_options = "scale_down")
```

Table 1: The first 5 rows of the dataset, movielens

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi

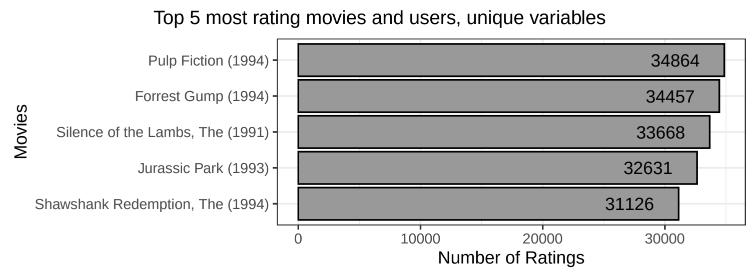
The variables, `movieId`, `userId` and `rating`, that is going to be used to build the model is presented as a matrix in the figure 1. The figure have the movies (`moveId`) in the x axis, the users in the y axis (`userId`), and the respective rating (`rating`). The matrix, extract from a sample of 50 users and 50 movies, provide some insights about the behavior of some users, the preference for some movies, and the sparsity of the matrix. The goal of this project is to fill the blank spaces with a rate.

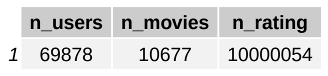
```r
#matrix users x movies x rating
set.seed(1, sample.kind="Rounding")
users <- sample(unique(movielens$userId), 50)
movielens %>% filter(userId %in% users) %>%
  select(userId, movieId, rating) %>%
  spread(movieId, rating) %>% select(sample(ncol(.), 50)) %>%
  as.matrix() %>% t(.) %>%
  image.plot(1:50, 1:50,. ,
             nlevel=9,
             xlab="Movies",
             ylab="Users",
             main= "Matrix: users x movies x rating",
             out.width = "50%")
```



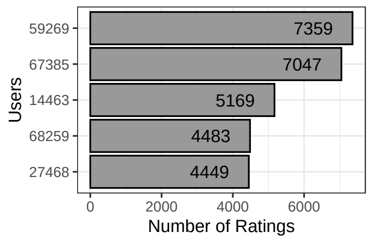
To see a potential flaw in the data we make a slice of the top 5 most rated movies and most active users. The unique users that provided ratings, the unique movies that were rated and the unique rating provided by a unique user to a unique movie, are presented to illustrate the possible rating matrix $users \times movies = 10677 \times 69878 = 746087406$ and the realized rating matrix 10000054, or 1.34% of points of the matrix is filled. The extremes values confirm that some users are much more actives than others, the most active user rated more than 50% of the total unique movies, and some movies have been rated for more than 1/3 of the total unique users.

```r
# Unique users, movies, rating
p0 <- tableGrob(movielens %>% summarize(n_users = n_distinct(userId),
                                        n_movies = n_distinct(movieId),
                                        n_rating = length(rating)))

# Top 5 movies
p1 <- movielens %>%
  group_by(title) %>%
  summarize(count = n()) %>%
  arrange(-count) %>%
  top_n(5, count) %>%
  ggplot(aes(count, reorder(title, count))) +
  geom_bar(color = "black", fill = "#999999", stat = "identity") +
  geom_text(aes(label=count), position=position_dodge(width=0.9), hjust=1.5) +
  xlab("Number of Ratings") +
  ylab("Movies") +
  theme_bw()

# Top 5 users
p2 <- movielens %>%
  group_by(userId) %>%
  summarize(count = n()) %>%
  arrange(-count) %>%
  top_n(5, count) %>%
  ggplot(aes(count, reorder(userId, count))) +
  geom_bar(color = "black", fill = "#999999", stat = "identity") +
  geom_text(aes(label=count), position=position_dodge(width=0.9), hjust=1.5) +
  xlab("Number of Ratings") +
  ylab("Users") +
  theme_bw()

#Top 5 most rating movies and users, unique variables
gridExtra::grid.arrange(p1,
               arrangeGrob (p0, p2, ncol = 2),
               nrow = 2,
               top = "Top 5 most rating movies and users, unique variables")
```



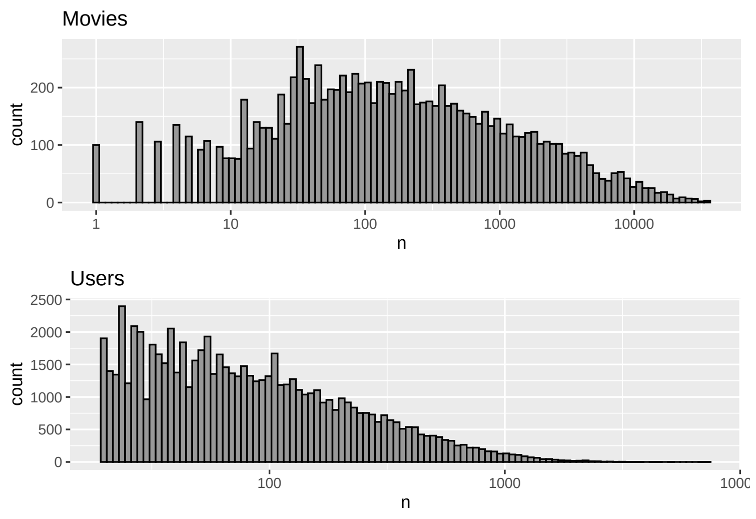


	n_users	n_movies	n_rating
1	69878	10677	10000054



The dataset distribution presented trough histograms provide some insights about the general proprieties of the data. As showed in the slice before some movies get rated more than others, and some user are more active than others.

```r
p3 <- movielens %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 100, color = "black", fill="#999999") +
  scale_x_log10() +
  ggtitle("Movies")
p4 <- movielens %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 100, color = "black", fill="#999999") +
  scale_x_log10() +
  ggtitle("Users")
gridExtra::grid.arrange(p3, p4, nrow = 2)
```



Modeling approach

The recommendation system is better as its error has decreased, for this project the error is the typical error we make when predicting a movie rating $(\hat{y}_{u,i} - y_{u,i})$. The loss function used to evaluate the models is based on the residual mean squared error ($RMSE$) on a test set. The definition of $RMSE$ includes N , the number of user/movie combination, and the sum occurring over all these combination. In this case the Loss function is:

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

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

The model which $RMSE = 0$ is a perfect model prediction, or without errors. For this project the reported $RMSE < 0.86490$ is the goal. The $RMSE > 1$ means our error is larger than one star, which means a bad model.

Machine Learning

The machine learning decisions are based on algorithms build with data, so for this project the dataset MovieLens 10M are going to be used to train and test the model. The train_set and test_set are build trough function `createDataPartition` as presented:

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

# Matching userId and movieId in both train and test sets
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Adding back rows into train set
removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)

#remove temporary file
rm(test_index, temp, removed)
```

Naive approach

The simplest model to recommend a movie to any user, is the model that predict the same rate μ for all movies regardless of user. In the naive approach the variation of the differences is random, and the independent error $\epsilon_{u,i}$ centered at 0. The model looks like:

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

The average of all ratings is the estimate that minimize the $RMSE$, and if we fill the blank cells in the matrix with the μ we obtain the $\varepsilon_{u,i}$.

```r
#Average of all ratings
mu <- mean(train_set$rating)
mu
```

```
## [1] 3.512465
```

```r
# Model 1 - Naive_rmse
naive_rmse <- RMSE(test_set$rating, mu)
results <- tibble(Method = "Model 1: Naive RMSE", RMSE = naive_rmse)
knitr::kable(results,
             caption = "RMSE by approach",
             align = "cc",
             position = "h")
```

Table 2: RMSE by approach

Method	RMSE
Model 1: Naive RMSE	1.061202

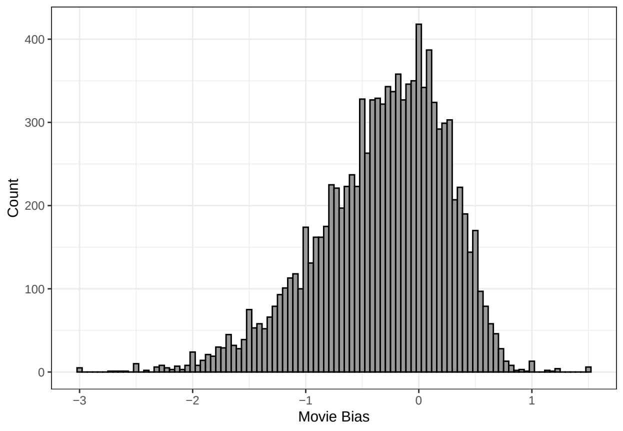
Bias approach

The data exploration and visualization trough Matrix: users x movies x rating presented a pictured that some user are more active than others, and some movies are better rated than others, so let's modeling this effects. In the formula below the b_i and b_u are respectively the movie and user effect:

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$$

The histogram presented below show the right skew distribution around zero for the movie effects, wich measure the mean of $Y_{u,i} - \hat{\mu}$ for each movie i.

```r
# Movie bias
movies_bias <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Movie bias distribution
movies_bias %>% ggplot(aes(b_i)) +
  geom_histogram(color = "black", fill = "#999999", bins = 100) +
  xlab("Movie Bias") +
  ylab("Count") +
  theme_bw()
```



In this approach, using $y_{u,i} = \mu + b_i$, our prediction improves by 0.11:

```
# Model 2 - Movie Bias RMSE
predicted_ratings <- mu + test_set %>%
  left_join(movies_bias, by = "movieId") %>%
  pull(b_i)
m_bias_rmse <- RMSE(predicted_ratings, test_set$rating)
results <- bind_rows(results, tibble(Method = "Model 2: Mean + movie bias",
                                     RMSE = m_bias_rmse))
knitr::kable(results,
             caption = "RMSE by approach",
             align = "cc",
             position = "h")
```

Table 3: RMSE by approach

Method	RMSE
Model 1: Naive RMSE	1.0612018
Model 2: Mean + movie bias	0.9439087

The user effects measure the variability across all users, in special the more actives users since they affect the average rate of more movies. As we saw in the colors of the matrix some users are more demanding than other. To fit the model we add the b_u term. The code of the improved model is:

```
# User bias
user_bias <- train_set %>%
  left_join(movies_bias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Model 3 - Movies and User Bias RMSE
predicted_ratings <- test_set %>%
  left_join(movies_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
movie_user_bias_rmse <- RMSE(predicted_ratings, test_set$rating)
results <- bind_rows(results,
                      tibble(Method = "Model 3: Mean + movie bias + user effect",
                             RMSE = movie_user_bias_rmse))
knitr::kable(results,
             caption = "RMSE by approach",
             align = "cc",
             position = "h")
```

Table 4: RMSE by approach

Method	RMSE
Model 1: Naive RMSE	1.0612018
Model 2: Mean + movie bias	0.9439087
Model 3: Mean + movie bias + user effect	0.8653488

Regularization approach

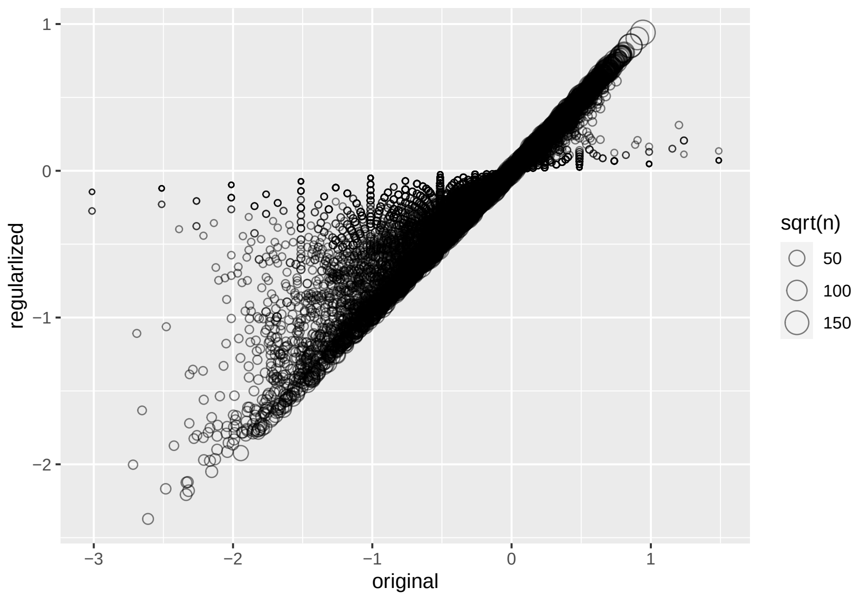
The movies that had one rate, or few rates, produce some noisy in the prediction system since larger errors can increase our $RMSE$. The regularization approach penalize large estimates that are formed using small samples, to constrain the total variability of the effect sizes.

The penalized least squares equation of the full model is presented bellow and the λ is the tuning parameter that allow us to control the variability of the effect size. When our sample size is very large, we have a stable estimate, and the penalty term, or λ, is ignored.

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

To present the effect of penalty term let's choose a $\lambda = 20$ and compare the original values and regularized value. The plot show that as small the sample are more close to 0 will be, or closer to mean value, which means a smaller $RMSE$ for that sample.

```
# Model 4 Regularization and RMSE
lambda <- 20
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

#Regularized x original
tibble(original = movie_avgs$b_i,
       regularlized = movie_reg_avgs$b_i,
       n = movie_reg_avgs$n_i) %>%
  ggplot(aes(original, regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```

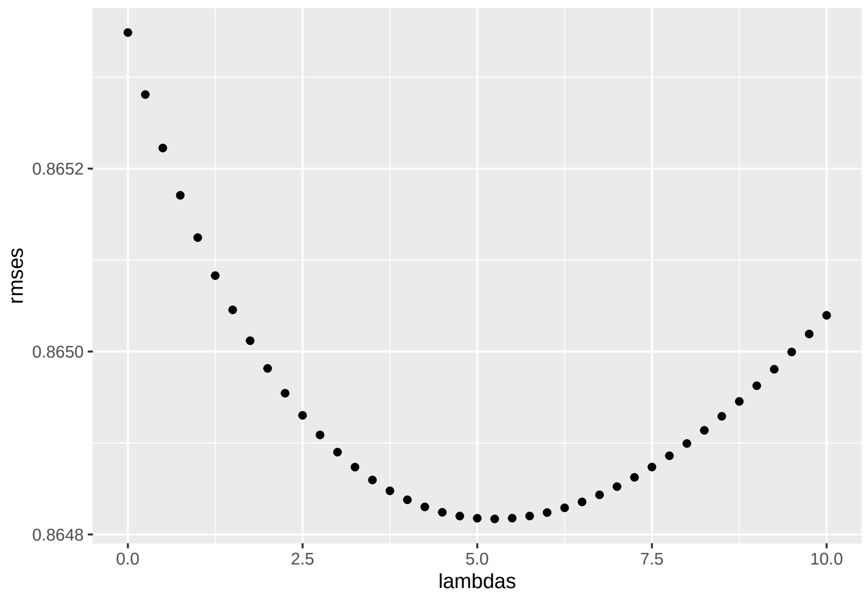


The values of b_i and b_u that minimize the full model and the code to calculate the $RMSE$ is presented below.

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

$$\hat{b}_u(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} \left(Y_{u,i} - \hat{b}_i(\lambda) - \hat{\mu}\right)$$

```r
#minimize the full model
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(x){
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+x))
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+x))
  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))
})

# Lambdas plot and best parameter
qplot(lambdas, rmses, color = I("#000000"))
```



```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

```
# Model 4 - Regularization RMSE
results <- bind_rows(results,
                     tibble(Method = "Model 4: Regularized movie and user effects",
                            RMSE = min(rmses)))
knitr::kable(results,
             caption = "RMSE by approach",
             align = "cc",
             position = "h")
```

Table 5: RMSE by approach

Method	RMSE
Model 1: Naive RMSE	1.0612018
Model 2: Mean + movie bias	0.9439087
Model 3: Mean + movie bias + user effect	0.8653488
Model 4: Regularized movie and user effects	0.8648170

Finally we achieve our goal, the $RMSE$ is lower than 0.86490. But can we do better?

Matrix Factorization approach

A matrix Factorization is the last approach to solve recommendation system problem presented in this project. The main idea is that there is structure in the data, the factors as gender and actor, could be grouped and form a simplification of information. We can approximate the matrix $R_{m \times n}$, that represent the relevant structure, by the product of two matrices of lower dimension: $P_{n \times k}$, or the latent factors of users, and $Q_{n \times k}$, that represent a group of movies.

$$R \approx PQ'$$

Let p_u be the u-th row of P, and q_v be the v-th row of Q, then the rating given by user u on movie v would be predicted as $p_u q_v'$. A typical solution for P and Q is given by the following optimization problem (Chin, Zhuang, et al. 2015a, 2015b):

$$\min_{P,Q} \sum_{(u,v)\in R} \left[f(p_u, q_v; r_{u,v}) + \mu_P ||p_u||_1 + \mu_Q ||q_v||_1 + \frac{\lambda_P}{2} ||p_u||_2^2 + \frac{\lambda_Q}{2} ||q_v||_2^2 \right]$$

where (u,v) are locations of observed entries in R, $r_{u,v}$ is the observed rating, f is the loss function, and μ_P,μ_Q,λP,λQ are penalty parameters to avoid overfitting.

The `recosystem` had the best accuracy in comparative analysis, and going to be our choice.[9] The algorithm use formatted data to tune the parameters by `$tune`, train by `$train` and test by `$predict`. The tune parameter adjust is: the number of latent factors `dim`, regularization cost for user and movie factors `costp_l2` and `costq_l2`, the step size in gradient descent `lrate` and the number of iterations `niter`.

```r
# Model 5 Matrix Factorization using recosystem
set.seed(1, sample.kind="Rounding")
train_reco <- with(train_set, data_memory(user_index = userId, item_index = movieId, rating = rating))
test_reco <- with(test_set, data_memory(user_index = userId, item_index = movieId, rating = rating))
r <- Reco()

# Recosystem tuning
opts <- r$tune(train_reco, opts = list(dim = c(20, 30),
                                        costp_l2 = c(0.01, 0.1),
                                        costq_l2 = c(0.01, 0.1),
                                        lrate = c(0.01, 0.1),
                                        nthread = 4,
                                        niter = 10))

# Recosystem training
r$train(train_reco, opts = c(opts$min, nthread = 4, niter = 30))
```

```
## iter      tr_rmse          obj
##    0       0.9728   1.2003e+07
##    1       0.8743   9.9030e+06
##    2       0.8400   9.1844e+06
##    3       0.8174   8.7560e+06
##    4       0.8017   8.4789e+06
##    5       0.7896   8.2803e+06
##    6       0.7797   8.1236e+06
##    7       0.7714   8.0007e+06
##    8       0.7644   7.9047e+06
##    9       0.7585   7.8241e+06
##   10       0.7533   7.7566e+06
##   11       0.7486   7.6979e+06
##   12       0.7444   7.6461e+06
##   13       0.7407   7.6048e+06
##   14       0.7374   7.5638e+06
##   15       0.7343   7.5319e+06
```

[9]https://rpubs.com/tarashnot/recommender_comparison

```
##   16        0.7314   7.4993e+06
##   17        0.7288   7.4727e+06
##   18        0.7262   7.4459e+06
##   19        0.7240   7.4238e+06
##   20        0.7219   7.4038e+06
##   21        0.7200   7.3842e+06
##   22        0.7181   7.3665e+06
##   23        0.7164   7.3500e+06
##   24        0.7148   7.3358e+06
##   25        0.7132   7.3202e+06
##   26        0.7118   7.3070e+06
##   27        0.7104   7.2956e+06
##   28        0.7092   7.2845e+06
##   29        0.7080   7.2748e+06
```

```
# Recosystem prediction
results_reco <- r$predict(test_reco, out_memory())

# Model 5 RMSE
factorization_rmse <- RMSE(results_reco, test_set$rating)
results <- bind_rows(results, tibble(Method = "Model 5: Matrix factorization using recosystem", RMSE = f
knitr::kable(results,
             caption = "RMSE by approach",
             align = "cc",
             position = "h")
```

Table 6: RMSE by approach

Method	RMSE
Model 1: Naive RMSE	1.0612018
Model 2: Mean + movie bias	0.9439087
Model 3: Mean + movie bias + user effect	0.8653488
Model 4: Regularized movie and user effects	0.8648170
Model 5: Matrix factorization using recosystem	0.7807834

The model improved substantially and we achieve the best result so far, we have our choice to train and test the dataset.

Results

This section presents the modeling results and discusses the model performance of the `movielens` data frame. The code below provided by HarvardX and create the `edx` and `validation` datasets that will be used to train and test our final algorithm.

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################

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

```r
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")

# if using R 4.0 or later:
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")
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)

################################################
#End of inital code supplied by HarvardX course
################################################


# Final validation
set.seed(1, sample.kind="Rounding")
edx_reco <- with(edx, data_memory(user_index = userId, item_index = movieId, rating = rating))
validation_reco <- with(validation, data_memory(user_index = userId, item_index = movieId, rating = rat
r <- Reco()

para_reco <- r$tune(edx_reco, opts = list(dim = c(20, 30),
                                          costp_l2 = c(0.01, 0.1),
                                          costq_l2 = c(0.01, 0.1),
                                          lrate = c(0.01, 0.1),
                                          nthread = 4,
                                          niter = 10))

r$train(edx_reco, opts = c(para_reco$min, nthread = 4, niter = 30))
```

```
## iter      tr_rmse          obj
##    0       0.9726   1.2017e+07
##    1       0.8730   9.8870e+06
##    2       0.8393   9.1783e+06
##    3       0.8167   8.7508e+06
##    4       0.8013   8.4699e+06
##    5       0.7899   8.2787e+06
##    6       0.7805   8.1290e+06
##    7       0.7727   8.0094e+06
##    8       0.7659   7.9136e+06
##    9       0.7601   7.8353e+06
##   10       0.7550   7.7663e+06
##   11       0.7504   7.7093e+06
##   12       0.7462   7.6576e+06
##   13       0.7424   7.6143e+06
##   14       0.7390   7.5759e+06
##   15       0.7359   7.5431e+06
##   16       0.7329   7.5105e+06
##   17       0.7303   7.4839e+06
##   18       0.7278   7.4588e+06
##   19       0.7255   7.4337e+06
##   20       0.7234   7.4144e+06
##   21       0.7214   7.3942e+06
##   22       0.7195   7.3771e+06
##   23       0.7178   7.3592e+06
##   24       0.7162   7.3456e+06
##   25       0.7146   7.3300e+06
##   26       0.7131   7.3171e+06
##   27       0.7118   7.3069e+06
##   28       0.7105   7.2938e+06
##   29       0.7093   7.2839e+06
```

```
# Final prediction
final_reco <- r$predict(validation_reco, out_memory())

# Final RMSE
final_rmse <- RMSE(final_reco, validation$rating)
final_rmse
```

```
## [1] 0.7812026
```

The final results of matrix factorization is a $RMSE$ of 0.7809.

Conclusion

The recommendation system, developed in this project, provide suggestions for items that are most pertinent to a particular user. The goal of this project was achieved and the $RMSE$ of matrix factorization of 0.7809 is lower than our target 0.86490. A larger dataset maybe could provide a even better model since we have more data to train, and is one of the limitations of this project. The addition of more variables could be a good outlook for future works since can capture more structures and latent factors in the data.

References

bnwicks - https://github.com/bnwicks/Capstone/blob/master/MovieLens.R
Airborne737 - https://www.rpubs.com/Airborne737/movielens

R Markdown: Gerando relatórios usando o R (Parte 1) - https://www.youtube.com/watch?v=obxa5VH4WvY

R Markdown: Gerando relatórios usando o R (Parte 2) - https://www.youtube.com/watch?v=tcNx0QbDPBo

Irizarry, Rafael A. 2019. *Introduction to Data Science*. Chapman; Hall/CRC. https://doi.org/10.1201/9780429341830.

Qiu, Yixuan, David Cortes, Chih-Jen Lin, Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang, and other contributors. See file AUTHORS for details. 2021. “Recosystem: Recommender System Using Matrix Factorization.” https://CRAN.R-project.org/package=recosystem.