MovieLens - Movie rating model

Andre

10 may 2019

Introduction

In this project we will create an machine learning algorithm that predict the movie rating of a movie by an specific user, this will be done using the MovieLens database.

For this project we will use the **Matrix Factorization** method.

Download data

The database used is in the GroupLens site.

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

Importing and joing the data

The datase is splited in two files, one with the ratings and another with the movies, we will extract both CSV files, adjust the columns names, join both datasets and adjust the columns types.

```r
# Reading the ratings file
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))

# Reading the movies file
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)

# Changing the column names
colnames(movies) <- c("movieId", "title", "genres")

# Adjusting the column types
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

# Creating the final dataset with both files
movielens <- left_join(ratings, movies, by = "movieId")
```

Splitting the data

To validade the quality of the models we will create 2 datasets, one for the training and another for validating the model.

```r
set.seed(1)
# Creating the splitting index
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
```

```r
# Creating the training dataset
training <- movielens[-test_index,]

temp <- movielens[test_index,]

# Creating the validation dataset
validation <- temp %>%
  semi_join(training, by = "movieId") %>%
  semi_join(training, by = "userId")

removed <- anti_join(temp, validation, by = c("userId", "movieId", "rating", "timestamp", "title", "gen
training <- rbind(training, removed)

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

Matrix Factorization

The model will use 3 parameters to create a prediction:

- Average rating for all movies and users (mu)
- Average rating for each movie (b_i)
- Average rating for each user (b_u)

With these parameters will be created the following equation

`Rating = mu + b_i + b_u`

We could use an liner regression model to calculate, but because the size of the data it would not be possible with the current machine, to decrease the computational power needed we will calculate each parameter individualy.

RMSE method

The method to evaluate the precision of the method will be Root Mean Square Error (RMSE).

This method evaluate the quality of the model with the square root of the mean square error for each prediction.

`SQUARE ROOT( AVERAGE( (TRUE VALUE - PREDICTED VALUE )^2 ) )`

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

Overall average rating

The first parameter is the overall average rating

```r
# Calculating the overall average rating
mu <- mean(training$rating)
```

Movie average rating

The second parameter is the average rating for each movie

```
# Calculating the movie average rating
movie_avgs <- training %>%
  group_by(movieId) %>%
  summarise(b_i=mean(rating - mu))
```

User average rating

The third parameter is the average rating by user already considering the movies average.

```
# Calculating the user average rating
user_avgs <- training %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u=mean(rating - mu - b_i))
```

Evaluation

To test the model we will test against the validation dataset.

```
# Calculating the ratings
predicted_rating <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

# Calulating the RMSE
rmse <- RMSE(predicted_rating, validation$rating)
rmse
```

```
## [1] 0.8655329
```

Conclusion

The Matrix Factorization method, even being an less sophisticated model, had a very good performance, with a RMSE of 0.86553, below the 0.87750 threshold.