MovieLens

Sesha

9/3/2019

MovieLens Project - Introduction and Overview

Movie Lens Project Dataset is approx. 9 million records about movie reviews/ratings provided by users for vast set of movies over hundered genres. The main objective of this project is to generate algorithm to predict movie ratings based on existing dataset. Though there are several methods to generate predictive machine learning algorithm, we are using an efficient algorithm to predict movie ratings.

The key metric used to measure efficiency would be RMSE Root Mean Square Error Value , which will be used to estimate efficiency of this algorithm.

MovieLens Project comes with a 2 separate datasets 1) Movies Datasets consists of movie ids, movie titles(their years), movie genre 2) Ratings Datasets to provide the ratings provided by various users for movies to hold ratingid,movieid,ratings

Methods and Analysis

This is regarding MovieLens Project. In this Project , an algorithm has been developed to predict ratings for random sample of movies.This algorithm provides efficiency of predicted ratings upto 87%. I have analysed algorithm using different methods. Initially the dataset is downloaded from location and dataset is split into testset and trainset. Since algorithm needs to be trained on exact same set of movies data , data has been cleaned to use same set of movies and users between test and training sets. In order to close predictions bias factors b_i and b_u have been considered as biases for movies and users respectively Lamda is being used to sample the data for sample set of values to maximize efficiency of this algorithm.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(tinytex)) install.packages("tinytex")
```

```
## Loading required package: tinytex
```

```
#install.packages("tinytex")
#tinytex::install_tinytex()

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




# Validation set will be 10% of MovieLens data
```

```r
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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```r
edx <- rbind(edx, removed)

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

lambdas <- seq(0, 10, 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 <-
    validation %>%
    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, validation$rating))
})

summary(rmses)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8648  0.8648  0.8649  0.8649  0.8650  0.8653
```

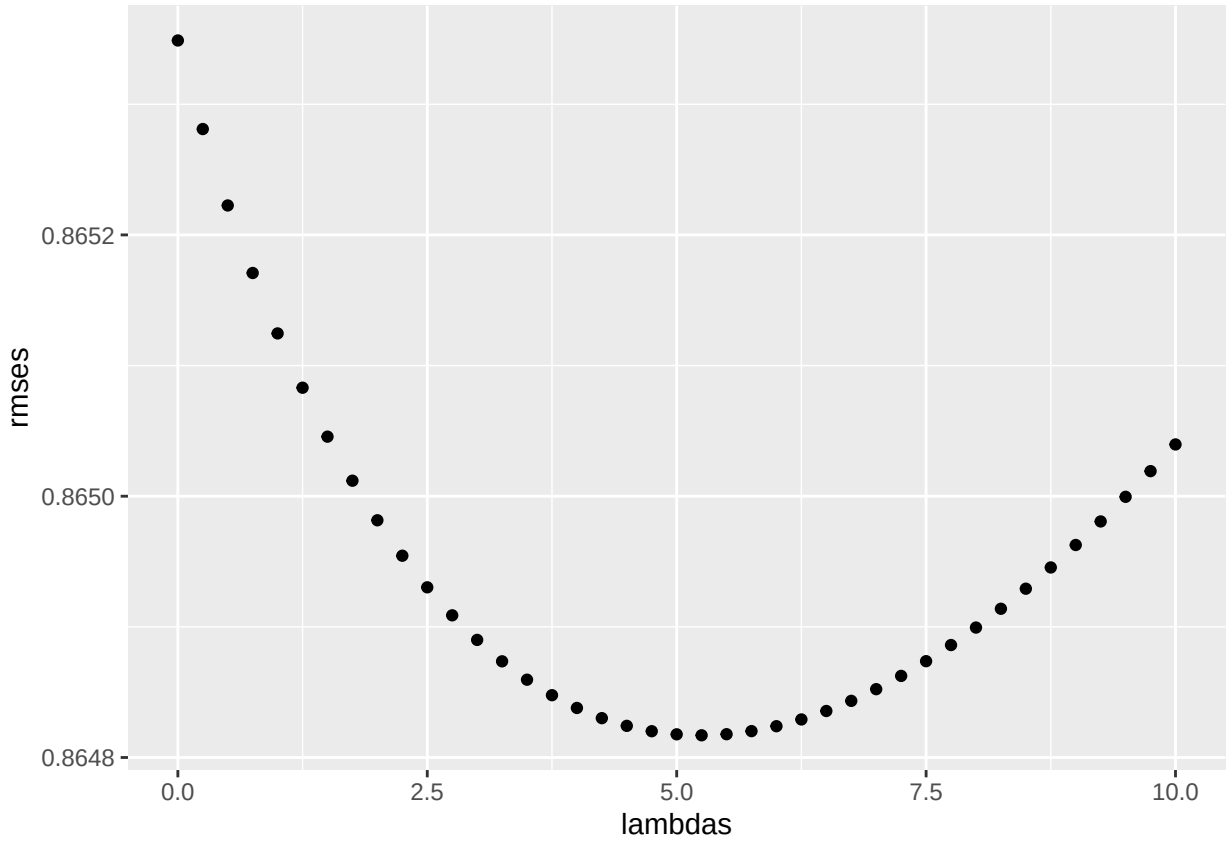
Results

As per analysis in understanding movielens algorithm , noticed that using linear regression it didnt fetch expected results due to bias values. However noticed better RMSEs with inclusion of movie bias on the base model and RMSE is around 0.986.

However if User effect is included along with Movie effect the result here is 0.908 with regularization it is 0.965.

With varying sample sets between 0 and 10, noticed that RMSE is improved and came down 0.864

Here is plot of used samples Vs RMSEs This plot helps to identify the lowest recorded RMSE for sample of lamdas.



Minimum RMSE noticed

```
min(rmses)
```

```
## [1] 0.864817
```

Lambda at which minimum rmse noticed

```
lambdas[which.min(rmses)]
```

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

Conclusion

Brief Summary: This algorithm provides an RMSE of 0.864 by considering bias for movies and users. This algorothm provides an increased efficiency by including bias values aswellas sampling data across different set of values so as to understand the best performing value/lambda.

Limitations and Future work: However there are few limitations to this algorithm in terms of data cleansing and usage of well balanced data. As we are not completely using entire dataset here some good amount data can be analysed further and set for these types of algorithms.

Also it will be a good idea to study genre bias values across users and include this factor in this algorithm.

Well balanced data and over sampling Future work is to identify more appropriate ways to organise data meaningfully and there by use the data cleansing techniques to sort and group based on userids/genres, which can further increase efficiency of algorithm.