Movielens Project

Pamela Gipe

6/4/2019

Introduction:

This project is intended to take the entire MovieLens dataset, which consists of two parts, the Movies database (including movies titles, release dates, and genres), and the Ratings database (which includes the movie titles, movie reviewers, and movie ratings), and provide a ratings prediction algorithm that predicts ratings based on inputs from the MovieLens dataset.

Data

The MovieLens raw dataset included the movies.dat file:

(First five rows)

1::Toy Story (1995)::Adventure|Animation|Children|Comedy|Fantasy 2::Jumanji (1995)::Adventure|Children|Fantasy 3::Grumpier Old Men (1995)::Comedy|Romance 4::Waiting to Exhale (1995)::Comedy|Drama|Romance 5::Father of the Bride Part II (1995)::Comedy

Each movie entry had an ID#, the full title of the movie, the year the movie was released, and the genre or genres of the movie.

The dataset also included the ratings.dat file

(first five rows)

1::122::5::838985046 1::185::5::838983525 1::231::5::838983392 1::292::5::838983421 1::316::5::838983392

Each row in this dataset included the ID# of the movie (which tied it to the movies.dat dataset), the id# of the reviewer, the rating of the reviewer of that movie, and a timestamp.

These datasets were merged using the following code:

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

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

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

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

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.1     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
```

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

ratings <- read.table(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")
```

The data was then split into a 90% training set, and a 10% test (validation) set using the following code:

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

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, validation)
```

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

```
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Data Summary:

The Training dataset (edx) has the following key characteristics:

```r
edx %>% summarize("Number of Reviewers" = n_distinct(userId),"Number of Movies" = n_distinct(movieId))
```

```
##   Number of Reviewers Number of Movies
## 1               69878            10677
```

```r
edx %>% separate_rows(genres, sep = "\\|") %>%
                group_by(genres) %>%
                summarize(count = n()) %>%
                arrange(desc(count))
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3909401
##  2 Comedy             3541284
##  3 Action             2560649
##  4 Thriller           2325349
##  5 Adventure          1908692
##  6 Romance            1712232
##  7 Sci-Fi             1341750
##  8 Crime              1326917
##  9 Fantasy             925624
## 10 Children            737851
## 11 Horror              691407
## 12 Mystery             567865
## 13 War                 511330
## 14 Animation           467220
## 15 Musical             432960
## 16 Western             189234
## 17 Film-Noir           118394
## 18 Documentary          93252
## 19 IMAX                  8190
## 20 (no genres listed)       6
```

RMSE Analysis:

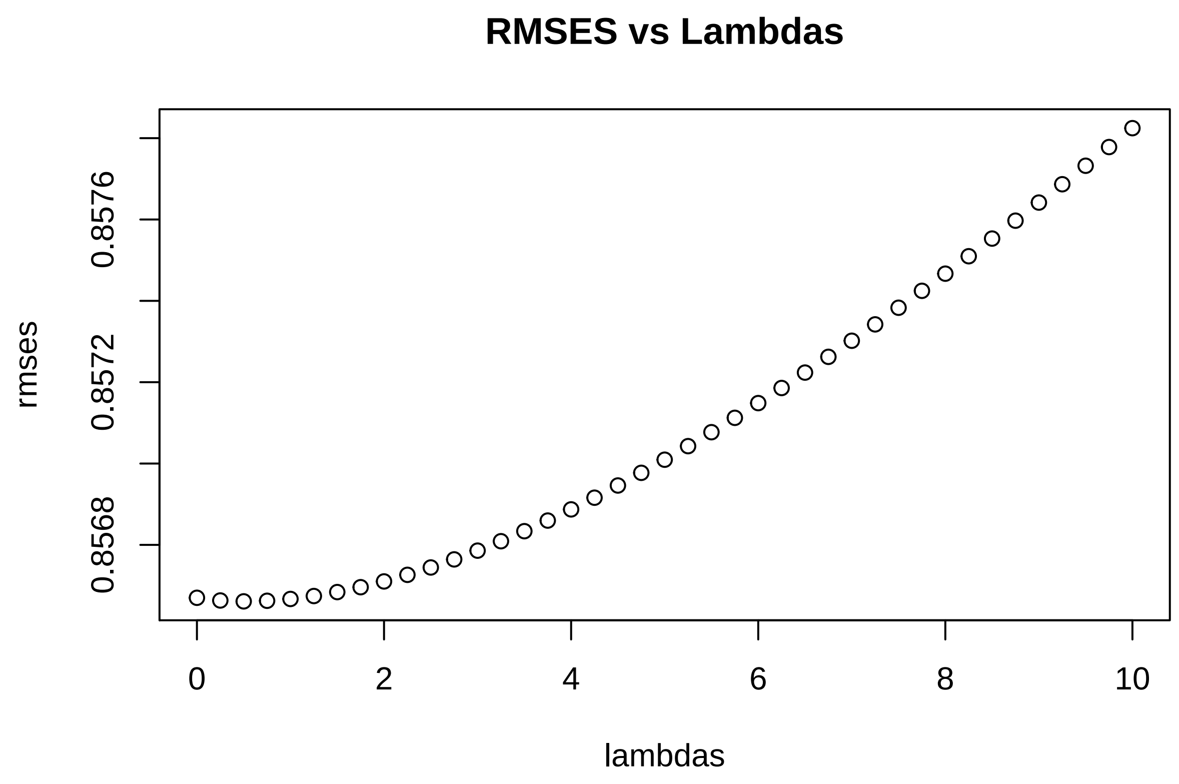
Root Mean Square Error (RMSE) is an evaluation of the relationship between data points. In general, it compares individual data points with a predicted regression line and reports how far away the data points are from the regression line. The closer the data points are to the regression line, the more accurate the predicted regression line is, and the lower the RMSE. A RMSE of 0 indicates no errors, and all data points are on the predicted regression line. The goal of this analysis is to find the lowest RMSE.

The method chosen to evaluate RMSE was a penalized least squares model. This allows for controlling the variability of the data, and includes a penalty (lambda), that when incorporated with a least squares evaluation method can reduce the errors in the prediction. Cross validation was used to derive a lambda using the training set and then apply it to the validation set to evaluate.

The following code:

```r
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>% group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    edx %>%
    left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u)
  return(RMSE(predicted_ratings, edx$rating)) })
```

Produced this chart:



Further analysis showed a lambda of **0.5** provided the lowest RMSE of **0.857**

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

```
## [1] 0.5
```

```r
min(rmses)
```

```
## [1] 0.8566611
```

Using the mean from the training set (edx), and calculating first the mean from the movie effect and then from the user effect using the optimal lambda of 0.5, predictions from the validation set (validation) were used.

```r
lambdaV <- .5
rmsesV <- sapply(lambdaV, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>% group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u)
  return(RMSE(predicted_ratings, validation$rating)) })

rmsesV
```

```
## [1] 0.8654111
```

The resulting RMSE for the validation set is **0.865**.

Conclusion:

It is informative to see how ratings are influenced by the reviewers and other factors. It would be interesting to dig further into the data and compare additional factors like release dates and genres (do summer drama movies have higher ratings than winter ones?). The size of the dataset impacts some analyses however, due to processor limitations. It can be frustrating waiting for an analysis to conclude.