Capstone Project 1: Movielens

Majid Sedighi

10/12/2020

1- Introduction

The main goal of this project is to create a movie recommendation system for the movielens dataset. This is similar to a competition held by Netflix where they challenge was to design an algorithm to accurately predict the ratings that users would give to movies based on the dataset provided by Netflix. To describe the goal of the project more specifically, first we need to get familiar with the dataset that is provided.

The Movie lens dataset is a set of movie reviews collected and made available by GroupLens, a research lab in the Department of Computer Science and Engineering at the University of Minnesota. The data sets were collected over various periods of time, and the complete latest version includes $27,000,000$ ratings for $58,000$ movies by $280,000$ users. The version that is used for this project is called movielens 10M, and it contains 10000054 ratings applied to 10681 movies by 71567 users (according to documentation provided by GroupLens). Each user is represented by an id, and all users selected had rated at least 20 movies. The main task for the algorithm designed in this project is to predict how a movie would be rated by a user based on the information available. The criteria for validating the achievement of this goal is the root-mean-square error (RMSE) between the predicted values and the observed values for ratings over a subset of data defined as the final validation set. The goal is to minimize this value using different methods and techniques taught throughout the course.

The algorithm used in this project to achieve the said goal builds on top of the attempts made during the course. it relies on regularizing various predictors and combinations of them. Throughout the course, the best results were achieved when the average of all the ratings were used as the baseline value, and two regularized bias terms were added, first for each movie and then for each user. This study continues this approach further by adding two more bias terms for the genre of the movie and for the date that the review is written for each movie. These terms are explained further in the following sections.

2- Analysis

In this section the algorithm is described in detail.

first, we need to acquire the data and perform the preprocessing steps to make it ready for the analysis. the script for the first part of this section is provided by the course, and it consists of downloading the database and splitting it to the training set and the hold-out final validation set. A point that has been considered in this part is to remove the entries in the validation set for the movies and users that do not have any data in the training set. These removed entries from the validation set are added back to the training set. Another point to consider is that the validation set defined here is only for the final test on the RMSE value, and it is not to be used during the development of the algorithm for purposes such as parameter tuning and tests.

here are some of the characteristics of the training data set as defined by the script.

```
## [1] "dimensions of edx"

## [1] 9000055       6

##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy

## [1] "dimensions of validation"

## [1] 999999      6

##    userId movieId rating timestamp
## 1:      1     231      5 838983392
## 2:      1     480      5 838983653
## 3:      1     586      5 838984068
## 4:      2     151      3 868246450
## 5:      2     858      2 868245645
## 6:      2    1544      3 868245920
##                                                       title
## 1:                                     Dumb & Dumber (1994)
## 2:                                     Jurassic Park (1993)
## 3:                                        Home Alone (1990)
## 4:                                           Rob Roy (1995)
## 5:                                    Godfather, The (1972)
## 6: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                   genres
## 1:                                Comedy
## 2:       Action|Adventure|Sci-Fi|Thriller
## 3:                       Children|Comedy
## 4:               Action|Drama|Romance|War
## 5:                           Crime|Drama
## 6: Action|Adventure|Horror|Sci-Fi|Thriller
```

As mentioned above, the final validation set cannot be used during the analysis, therefore the Edx set needs to be split again to training and test set. However, before splitting the data another additional step is necessary for the analysis. The dates of review submissions is included in the data set as “timestamp”, with values being the time passed since January 1, 1970 in seconds. We need the dates in the actual year of the review. Therefore we add a column to both Edx and validation called “date”, where we calculate the year that the review was written.

```r
org <- as.Date('1970-1-1')
set <- edx %>% mutate(date = as.Date(timestamp/86400,org)) %>%
  mutate(date=format(date, format="%Y"))
holdout <- validation %>% mutate(date = as.Date(timestamp/86400,org)) %>%
  mutate(date=format(date, format="%Y"))
```

Which results in our dataset mutating into this format:

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres date
## 1:                Comedy|Romance 1996
## 2:         Action|Crime|Thriller 1996
## 3:  Action|Drama|Sci-Fi|Thriller 1996
## 4:       Action|Adventure|Sci-Fi 1996
## 5: Action|Adventure|Drama|Sci-Fi 1996
## 6:       Children|Comedy|Fantasy 1996
```

Now we split the Edx again into training and test set for optimizing the hyper parameters, specifically the Lambda parameter for regularization. We choose the same proportion of data for the test set as the first split.

```r
set.seed(2, sample.kind="Rounding")
```

```
## Warning in set.seed(2, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
test_index <- createDataPartition(y = set$rating, times = 1, p = 0.1, list = FALSE)
train_set <- set[-test_index,]
temp <- set[test_index,]

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

# Add rows removed from test set back into training set
removed <- anti_join(temp, test_set)
```

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

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

The model chosen for the calculation is as follows:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_{d,i} + \epsilon_{u,i}$$

where $Y_{u,i}$ is the predicted rating for user u and movie i,

μ is the average of all ratings across all movies and users,

b_i is the regularized bias term for each movie,

b_u is the regularized bias term for each user,

b_g is the regularized bias term for each group of genres,

$b_{d,i}$ is the bias term for the date of the review in year d for movie i,

and $\epsilon_{u,i}$ is the error for user u and movie i.

Each bias term is regularized by the factor of λ, following this equation:

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

where n_i is a number of ratings b for movie i.

The process for calculation of each term is described here. The code in this section is for demonstration purpose and is not evaluated. An important point to consider when making the calculations is that when splitting the data we made sure the there are no movies nor users in the validation and test set that are not represented in the training set. However, we did not make sure that every unique group of genres and every combination of movie and review date in validation and test set are also represented in the training set. This can result in bias terms that are not calculated from the training set and are needed for making the predictions on validation and test set. In order to avoid producing NAs we consider these biases to be zero when making predictions. This is a logical assumption, since it makes sense to consider that the predictors for which we have no entries in the training set have no effect on our prediction.

The bias terms that have been used in the course already are calculated in the same manner as was in the course:

```r
mu <- mean(train_set$rating)
b_i <- train_set %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+1))
b_u <- train_set %>% group_by(userId) %>% summarize(b_u = sum(rating - b_i - mu)/(n()+1))
```

The new bias terms b_g and $b_{d,i}$ are an attempt to expand on the context of the course to achieve better accuracy. Here is a brief explanation on how each are calculated.

The dataset provided includes the column genres, where each movie has been indicated with multiple genres. there are 797 unique combinations of genres present in the edx dataset. we define a regularized bias term for each of these values using the following code:

```r
b_g <- train_set %>% group_by(genres) %>%
  summarize(b_g =sum(rating - b_i - b_u - mu)/(n()+1))
```

Note that the b_i, b_u, and μ values have to be subtracted beforehand, since we would have them calculated before hand.

The last bias term $b_{d,i}$, uses the date column that was created before, where we have the date of the reviews in year. The goal here is to calculate the bias term for each movie as well as the year that the review was written, therefore we need two levels of grouping the data, and this code will provide it:

```r
b_d_i <- train_set %>% group_by(movieId,date) %>%
  summarize(b_d_i =sum(rating - b_i - b_u - b_g - mu)/(n()+1))
```

Once we have all of the bias terms calculate, we need to add them to the test set in their corresponding positions to be able to predict the ratings for the entries in that set.

```r
prediction <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_d_i, by = c("movieId"="movieId","date"="date"))
```

As mentioned before, the calculation of $b_{d,i}$ in this manner can result in NAs. Therefore we add this line to be able to get a prediction for such values.

```r
prediction$b_d_i[is.na(prediction$b_d_i)] <- 0
```

Note that it is technically possible to also get NAs for b_g on the prediction set. However, in practice this does not happen, and the code is not included in order to accelerate the calculations.

Now we can make predictions by simply adding the terms in the equation:

```r
predicted_ratings <- prediction %>% mutate(pred = mu + b_i + b_u + b_g + b_d_i) %>% .$pred
```

Finally, we can estimate the accuracy of our model using the $RMSE$ metric. This value is calculated as below:

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

Where N is the number of entries, and $\hat{y}_{u,i}$ and $y_{u,i})^2$ are the predicted and the real values of the ratings, respectively.

For this algorithm, we define a function to handle the calculation of RMSE:

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

In practice, we would like to run this model multiple times in order to fine tune the algorithm for the value of λ. Therefore we define a function to calculate the $RMSE$ given with λ as input:

```r
model <- function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>% group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  ts <- train_set %>% left_join(b_i, by="movieId")
  b_u <- ts %>% group_by(userId) %>% summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  ts <- ts %>% left_join(b_u, by = "userId")
  b_g <- ts %>% group_by(genres) %>% summarize(b_g =sum(rating - b_i - b_u - mu)/(n()+l))
  ts <- ts %>% left_join(b_g, by = "genres")
  b_d_i <- ts %>% group_by(movieId,date) %>%
    summarize(b_d_i =sum(rating - b_i - b_u - b_g - mu)/(n()+l))

  prediction <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_d_i, by = c("movieId"="movieId","date"="date"))

  prediction$b_d_i[is.na(prediction$b_d_i)] <- 0
  predicted_ratings <- prediction %>%
    mutate(pred = mu + b_i + b_u + b_g + b_d_i) %>% .$pred
  return(RMSE(predicted_ratings, test_set$rating))
}
```

3- Results

In this section we run our calculations by feeding the datasets to thhe functions we defined in the last section in order to find the optimal λ for this dataset. In this stage of the calculation, we are not using the final validation set, therefore we carry out using the training and test set that was the result of the splitting of edx set.

for each value of λ, the bias terms are calculated and the $RMSE$ for the test set (not to be confused the final validation set) is saved in a vector. For this algorithm we first run the function over a sequence of integers between 1 to 10, to get rough estimation of the optimized λ parameter. Then we run it again around the integer with the minimum $RMSE$ to fine tune the λ parameter with a margin of 0.1.

```
lambdas <- seq(1, 10, 1)
```

The first stage of calculation shows the following results:

```
result <- data.frame(Lambda=lambdas, RMSE=rmses)
print(result)
```

```
##    Lambda      RMSE
## 1       1 0.8621291
## 2       2 0.8616523
## 3       3 0.8613516
## 4       4 0.8611488
## 5       5 0.8610118
## 6       6 0.8609234
## 7       7 0.8608727
## 8       8 0.8608524
## 9       9 0.8608568
## 10     10 0.8608820
```

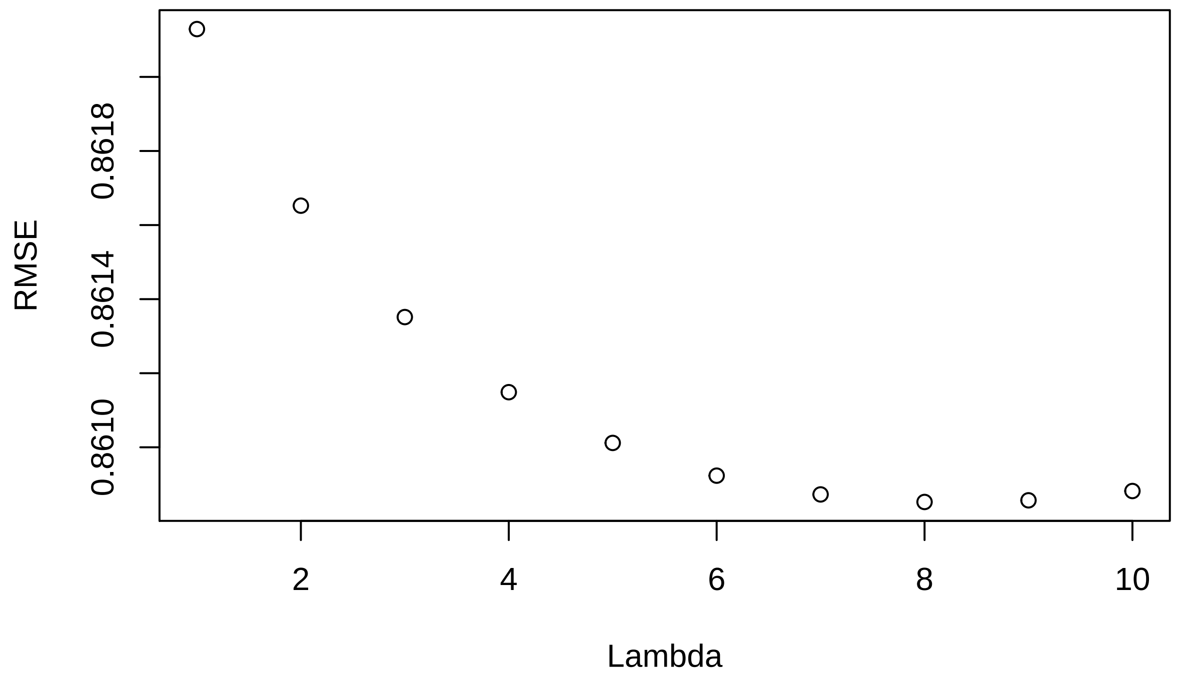
```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 8
```

```
min(rmses)
```

```
## [1] 0.8608524
```

```
plot(result)
```



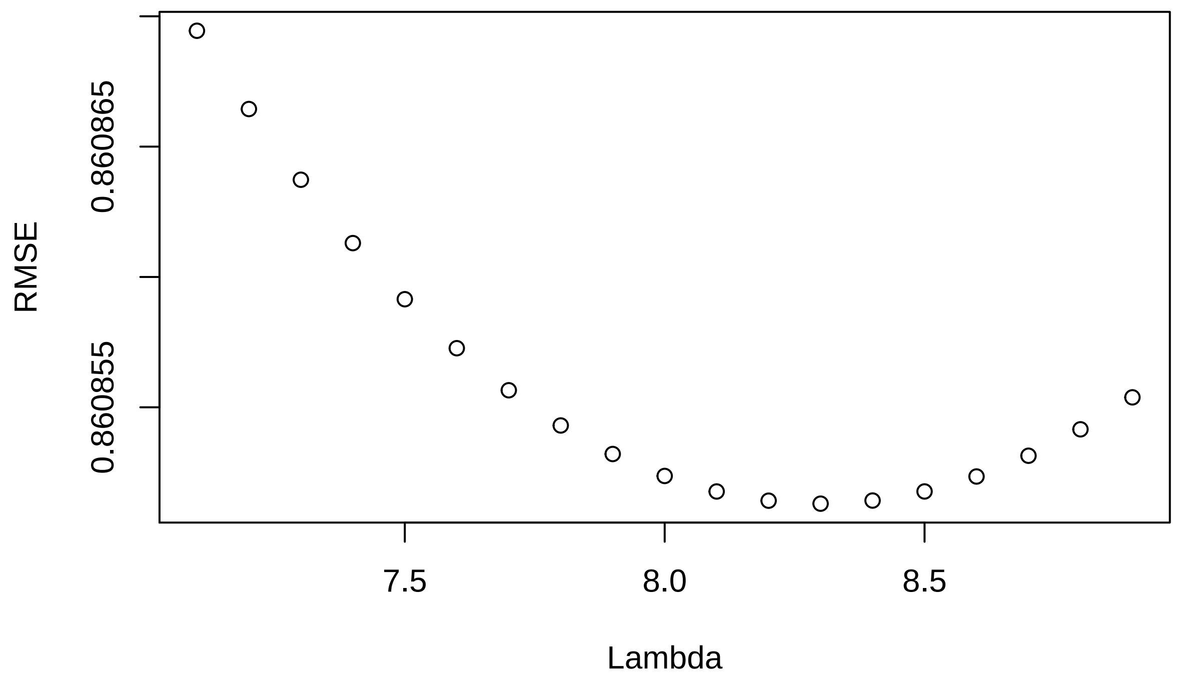
now that we found the 8 to result the lowest $RMSE$, we run another calculation around this value with more precision:

```
lambdas <- seq(lambda-0.9, lambda+0.9, 0.1)
```

```
result <- data.frame(Lambda=lambdas, RMSE=rmses)
print(result)
```

```
##    Lambda      RMSE
## 1     7.1 0.8608694
## 2     7.2 0.8608664
## 3     7.3 0.8608637
## 4     7.4 0.8608613
## 5     7.5 0.8608591
## 6     7.6 0.8608573
## 7     7.7 0.8608557
## 8     7.8 0.8608543
## 9     7.9 0.8608532
## 10    8.0 0.8608524
## 11    8.1 0.8608518
## 12    8.2 0.8608514
## 13    8.3 0.8608513
## 14    8.4 0.8608514
## 15    8.5 0.8608518
## 16    8.6 0.8608523
## 17    8.7 0.8608531
## 18    8.8 0.8608542
## 19    8.9 0.8608554
```

```r
plot(result)
```



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

```
## [1] 8.3
```

```r
min(rmses)
```

```
## [1] 0.8608513
```

Based on the second stage, we find the 8.3 to be the best value for this dataset. Now we can train for the whole edx dataset and test on our final validation set to acquire our final result for $RMSE$.

```r
final_RMSE
```

```
## [1] 0.8597354
```

The final result for $RMSE$ is 0.8597354, which is below the value that was the goal given by the assignement.

4- Conclusion

In this study, an attempt was made to expand on the analysis that was carried out during the course by the means of adding two more bias terms for date of review and genre of the movies. Although these terms improved the accuracy of predictions and achieved the goal for the value of the $RMSE$, this is not in any way a complete approach for analyzing this data. In the following, some means to improve this results is discussed.

An essential characteristic of such data is the interaction between the users. Users that have similar taste in movies tend to rate movies similarly. A good approach would be to find similar users to each user (using correlation matrix or any other metric), and consider how they rated a movie for making prediction. This approach was tested for this study but did not yield good results. This could be due to the limited number of entries in the dataset, since for every prediction there was very few similar user ratings available, if any. This approach could work for a much larger dataset, if the speed of calculations is not an issue (calculating all the correlations and creating similar user data for each user is computationally expensive). Same correlations can be studied between movies to explore similariies between movies.

In order to improve the results given here an approach could be to include the date of review in months of year instead of only year used in this study. This could pick up a more precise temporal effect for reviews and improve accuracy. Again due to the limitations of the dataset size this approach will result in a significant number of NAs in the prediction of the corresponding bias term, and in result defeats the purpose.