Movies Recommendation

Silas Liu

11/20/2021

Introduction

This project intends on modelling a recommendation system for movie ratings, based on the 10M MovieLens dataset. This is an independent set of movie ratings, for purpose of studies on machine learning techniques. This specific set contains 10 million entries, with rating from 72,000 users to 10,000 movies.

Our goal will be developing an algorithm to study the internal structure of the data and predict ratings for movies, based on available data of other users and movies ratings and minimize the residual mean squared error of the predictions.

We start by doing a data analysis on the dataset, proceeding to fitting a model with the training set and applying the model to predict on the verification set.

Analysis

The first step will be a data analysis, by making an overview of the edx dataset, which will be used to train the model.

Data Analysis

```
dim(edx)
```

```
## [1] 9000055       6
```

```
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 8
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  - attr(*, ".internal.selfref")=<externalptr>
```

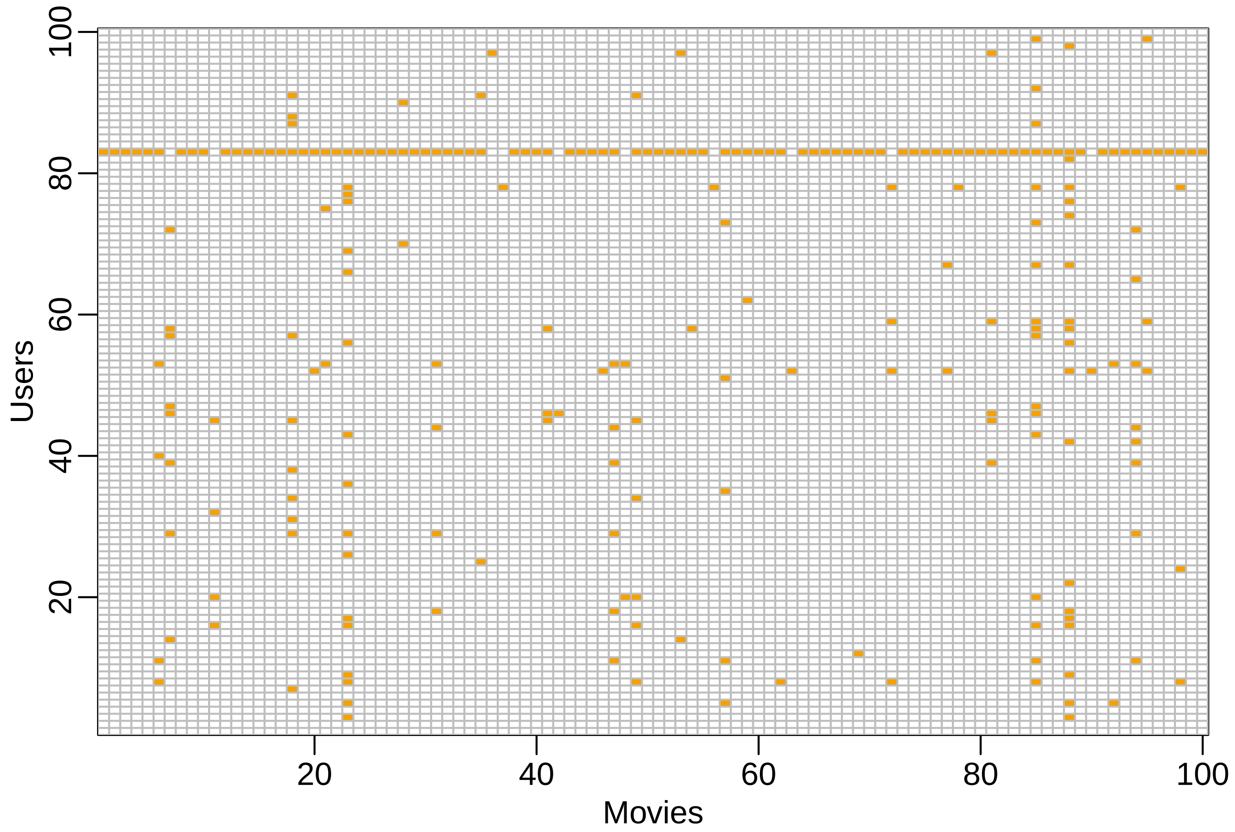
```
head(edx)
```

```
##    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
```

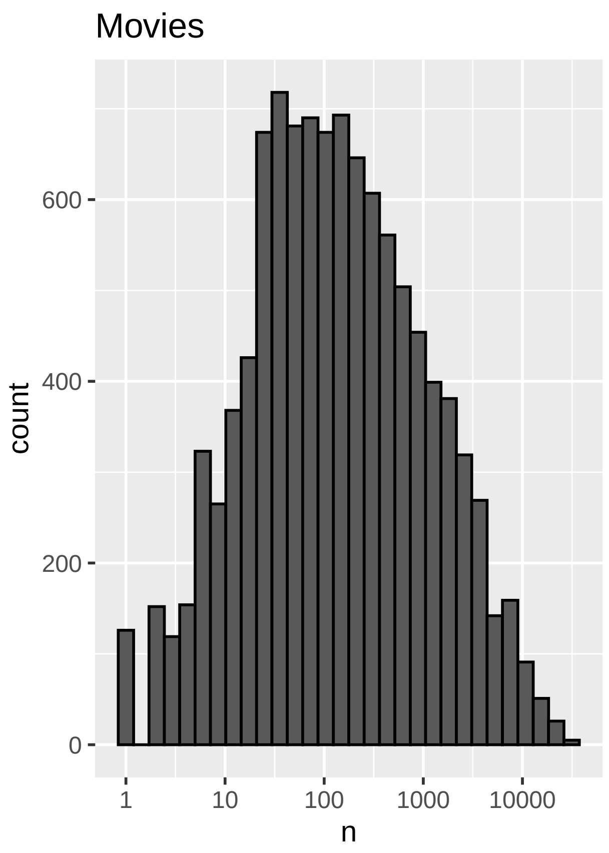
We look if there are NA values.

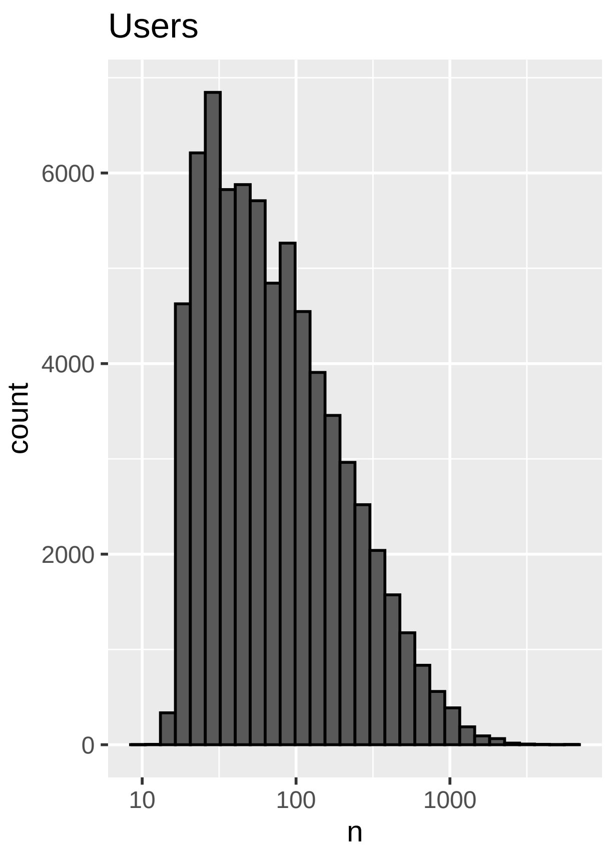
```
##   count
## 1     0
```

It can be seen that the data has no NA values. We proceed then with some insights of the data before the proper modeling. By taking a random sample of 100 movies and 100 users and showing them with colored dots, we can see how sparse the dataset of ratings is:



The following graphics show the distribution between movies and users. We can notice that some movies are more rated than others, just like there are some users who tend to rate more than others.





Model 1

For the first model, we will consider the movie and user effects and apply regularization to the data. First we calculate the mean rating over all movies and then we find the residuals as average value for each movie and user. Since the data has many outliers, movies or users with one or few ratings, regularization is applied. The equation which needs to be minimized can be written as: $\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2)$ With regularization, the estimates b_u and b_i, for user and movie, respectively, that minimize the equation are given by: $\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{i=1}^{n_i}(Y_{u,i}-\hat{\mu})$ $\hat{b}_u(\lambda)=\frac{1}{\lambda+n_u}\sum_{u=1}^{n_u}(Y_{u,i}-\hat{\mu}-\hat{b}_i)$ In order to optimize our model, we calculate the residual mean squared error (user u and movie i), which is defined by: $RMSE=\sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i}-y_{u,i})^2}$

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

Since λ is a tunning parameter of our model, we apply 10-fold cross validation to find the best value for it.

```
lambdas <- seq(0, 10, 0.25)
folds <- createFolds(edx$rating, k = 10, list = TRUE, returnTrain = FALSE)

rmses <- sapply(lambdas, function(l){
  rmse <- sapply(folds, function(i){
    train_set_star <- edx[i,]
    test_set_star <- edx[-i,] %>%
      semi_join(train_set_star, by="movieId") %>%
      semi_join(train_set_star, by="userId")

    mu <- mean(train_set_star$rating)
    b_i <- train_set_star %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu)/(n()+l))
    b_u <- train_set_star %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - mu - b_i)/(n()+l))

    predicted_ratings <- test_set_star %>%
      left_join(b_i, by="movieId") %>%
      left_join(b_u, by="userId") %>%
      mutate(pred = mu + b_i + b_u) %>%
      .$pred
    return(RMSE(test_set_star$rating, predicted_ratings))
  })
  return(mean(rmse))
})
```

With the tunned parameter, we are able to train the model from the entire training set and predict it on the verification set:

```
lambda <- lambdas[which.min(rmses)]
mu_hat <- mean(edx$rating)
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_hat)/(n()+lambda))
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu_hat - b_i)/(n()+lambda))

predicted_ratings <- validation %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  .$pred
```

Model 2

Our second model will consider the genre effect as well, in addition to the movie and user effect. In previous analysis we verified that there is correlation between the rating and movie genres. We group together each different genre or combination of genres and look for their average values, applying to the model. At the end we also apply regularization to the data. We start by tunning the λ parameter with 10-fold cross validation.

```
rmses <- sapply(lambdas, function(l){
  rmse <- sapply(folds, function(i){
    train_set_star <- edx[i,]
    test_set_star <- edx[-i,] %>%
      semi_join(train_set_star, by="movieId") %>%
      semi_join(train_set_star, by="userId")

    mu <- mean(train_set_star$rating)
    b_i <- train_set_star %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu)/(n()+l))
    b_u <- train_set_star %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - mu - b_i)/(n()+l))
    b_g <- train_set_star %>%
      left_join(b_i, by="movieId") %>%
      left_join(b_u, by="userId") %>%
      group_by(genres) %>%
      summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+l))

    predicted_ratings <- test_set_star %>%
      left_join(b_i, by="movieId") %>%
      left_join(b_u, by="userId") %>%
      left_join(b_g, by="genres") %>%
      mutate(pred = mu + b_i + b_u + b_g) %>%
      .$pred
    return(RMSE(test_set_star$rating, predicted_ratings))
  })
  return(mean(rmse))
})
```

With the tunned parameter, we are able to fit the model with the entire training set and predict it on the verification set:

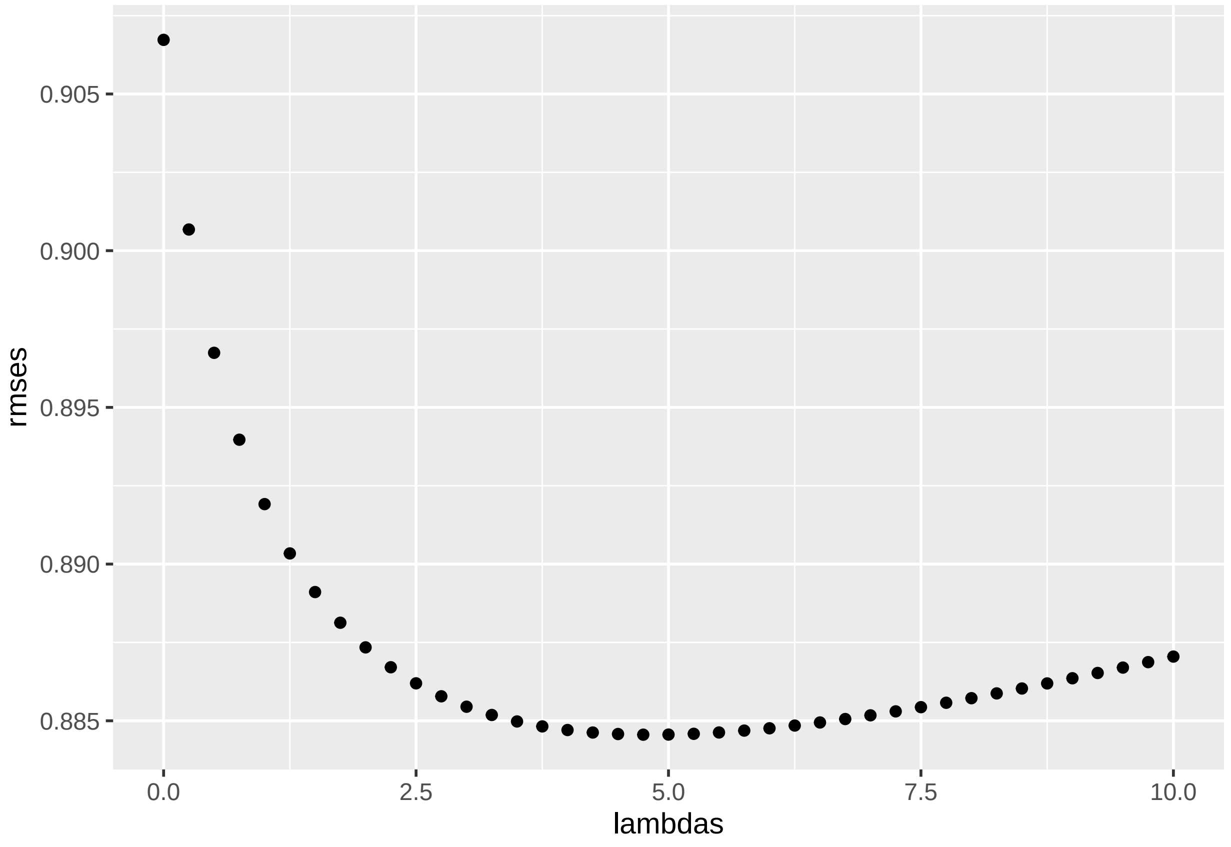
```
lambda <- lambdas[which.min(rmses)]
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_hat)/(n()+lambda))
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu_hat - b_i)/(n()+lambda))
b_g <- edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu_hat - b_i - b_u)/(n()+lambda))

predicted_ratings <- validation %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(b_g, by="genres") %>%
  mutate(pred = mu_hat + b_i + b_u + b_g) %>%
  .$pred
```

Results

This section presents the results obtained from the parameter tunning and model fitting as well as its performance. The tunning process for λ searchs for the minimum RMSE, the best value. Below we can look at the tunning for both models and their best value:

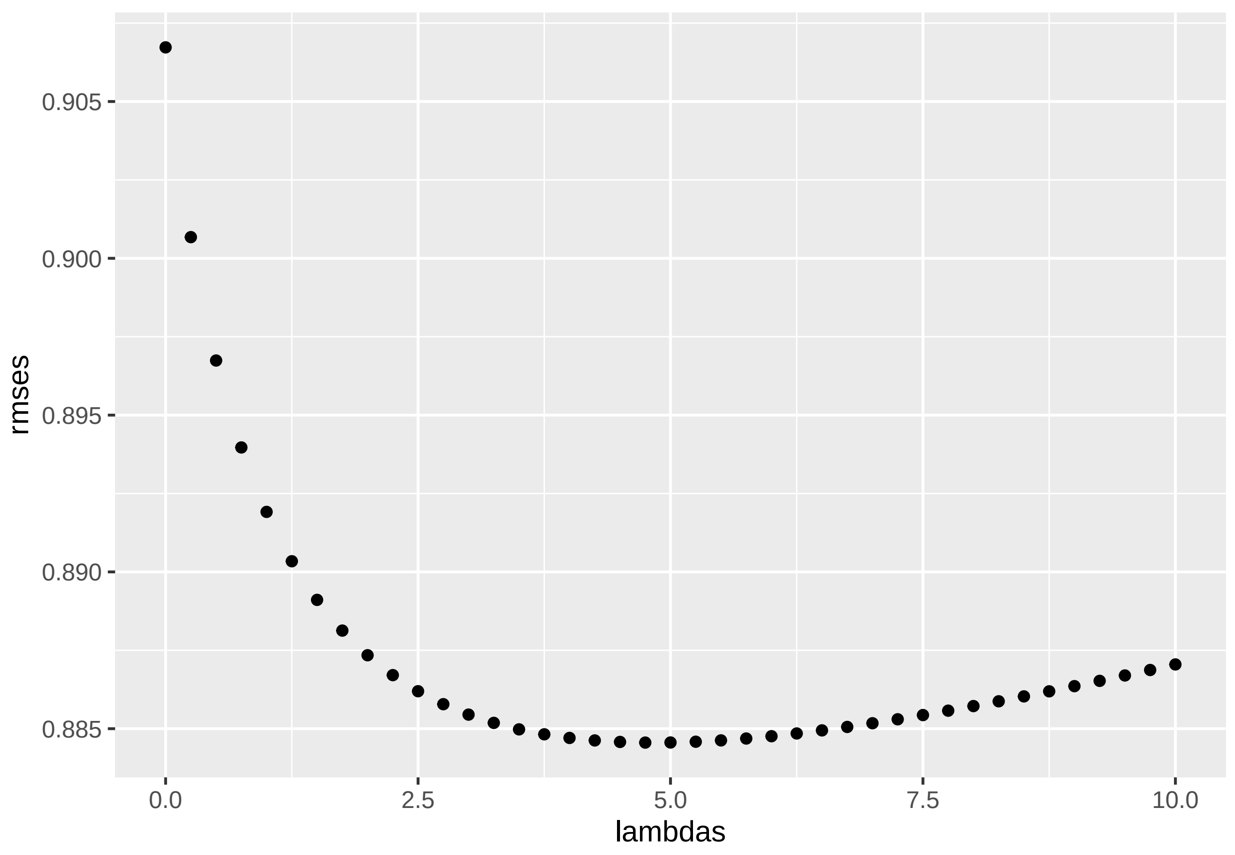
```
lambdas_1
```



```
min_lambdas_1
```

```
## [1] 4.75
```

```
lambdas_2
```



```
min_lambdas_2
```

```
## [1] 4.75
```

With the best λ, we can calculate the RMSE of each model, comparing with the validation set ratings:

method	RMSE
Model 1: Regularized Movie + User Effect	0.8648201
Model 2: Regularized Movie + User + Genre Effect	0.8644514

Conclusion

With this project we were able to construct two models for movie recommendation system, applying regularization on the data. Both used data from a training set to train the models, and fit and validate them on a validation set. For the parameter tuning process, we used only the training set with 10-fold cross validation. The first model considered only movies and users as predictors, while the second model also considered the genres as predictor.

The RMSE found on the second model was better, below 0.86490. The limitations of the final model is that predictions on new users or movies will not be possible, since they both must appear in the training set. So the model needs to be improved and retrained with new data, in a regular time frame, when there are new movies or users. For future work we suggest improvement on the model, by analyzing and better modeling the effect of the genres on the rating.