Capstone-MovieLens

Ahmed Al-Jifri

2023-07-19

Introduction

The MoviesLens dataset is a dataset available online that hosts 10 million observations reflecting user interactions through movie ratings.It has ratings data from 69,878 unique users and 10,677 unique movies.The aim of this project is the development of a robust machine learning algorithm which can suggest personalized movie recommendations for users based on their historical movie ratings. To achieve this, a linear regression model coupled with Singular Value Decomposition technique (SVD) is suggested. However, since the dataset is huge, hardware/memory limitations have been imposed on the use of SVD. Therefore, a technique called "Implicitly Restarted Lanczos Bidiagonalization Algorithm" or IRLBA which approximate right and left singular vectors of svd was implemented just to showcase the idea of SVD.

The key steps in this project encompass data preprocessing to ensure cleanliness and uniformity, data exploration and visualization to get a better understanding of the dataset on hand and finally the modeling of a machine learning algorithm to achieve the goal of this project.

Additionally, the model's performance shall be tested, evaluating its accuracy and efficacy aiming to offer insights into its practical applicability. Finally, this project aims to bridge the gap between user tastes and movie recommendations, infusing the world of cinema with the precision of data science to enrich the cinematic journey of each user, one recommendation at a time.

Methodology

Firstly, the dataset provided has already been slightly preprocessed to encompass the essence of the project. the dataset has been split into two parts, a training part which hols 90% of the dataset and a test set containing the rest 10% which was done in a way to preserve both sets to have the same unique users and movies. the aim here is to build our model using the training set and use the test set only for evaluating the performance of the model.

As mentioned in the introduction we have 69,878 unique users and 10,677 unique movies, however it is evident that not all users have rated all the movies. To see this, first we need to transform the dataset on hand into a matrix with users as rows and movies as columns with ratings inside the cells. Below is basic summary and structure of the dataset "edx" which is our training dataset. it contains 6 columns with basic overview shown below:

```
load("edx.RData")
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
```

```
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  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 83
##  $ 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
```

```
rm(edx)
```

Having transformed our dataset edx into a matrix call it y, we can look at the structure below.

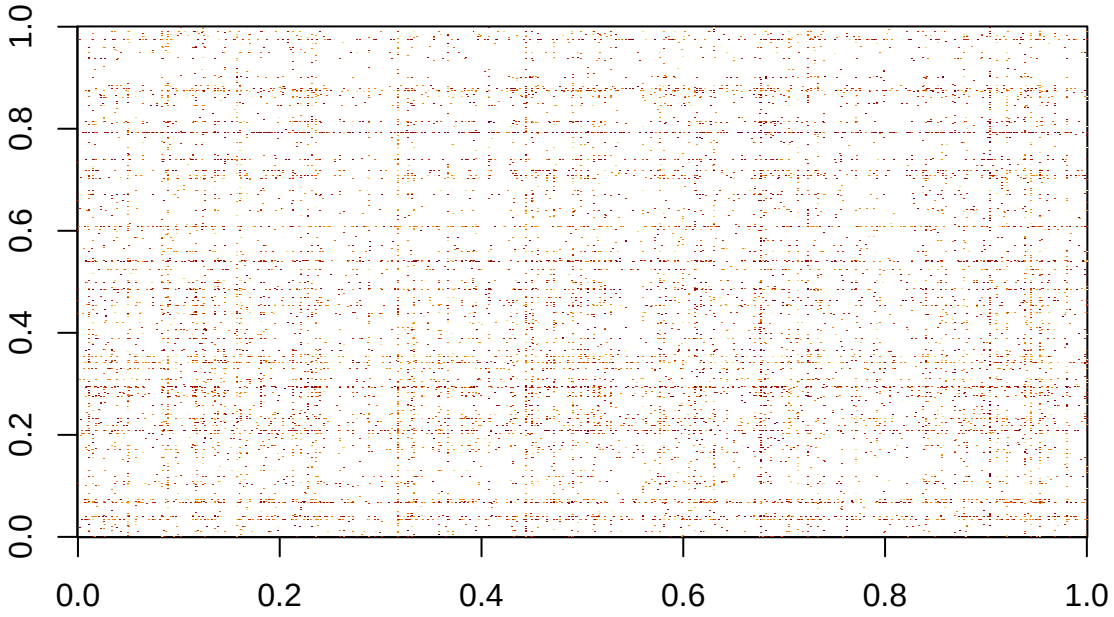
```
load("y.RData")

str(y)
```

```
##  num [1:69878, 1:10677] 5 NA NA NA NA NA NA NA NA NA ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:10677] "122" "185" "292" "316" ...
```

Nevertheless, the new matrix y mostly consists of empty cells, since as mentioned not all movies are rated by all users. a sample of y can be drawn and visualized to get an idea as shown below:

```
ind <- sample(1:nrow(y), 1000)
ind1 <- sample(1:ncol(y), 1000)
image(y[ind, ind1])
```

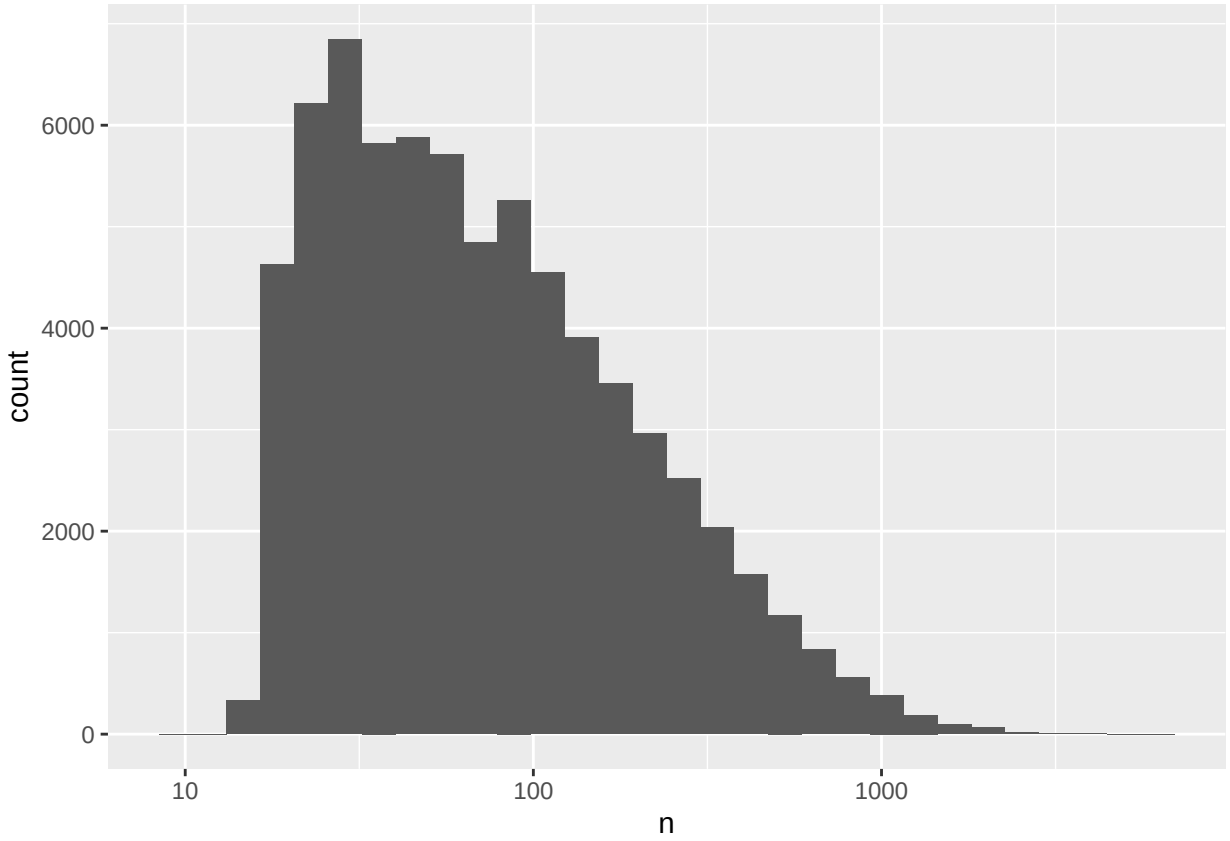



```
rm(y)
```

As seen above the white space represents empty cells and the dots represent ratings. Moreover, we can visualize how often users rate movies or how often movies get rated as shown below respectively:

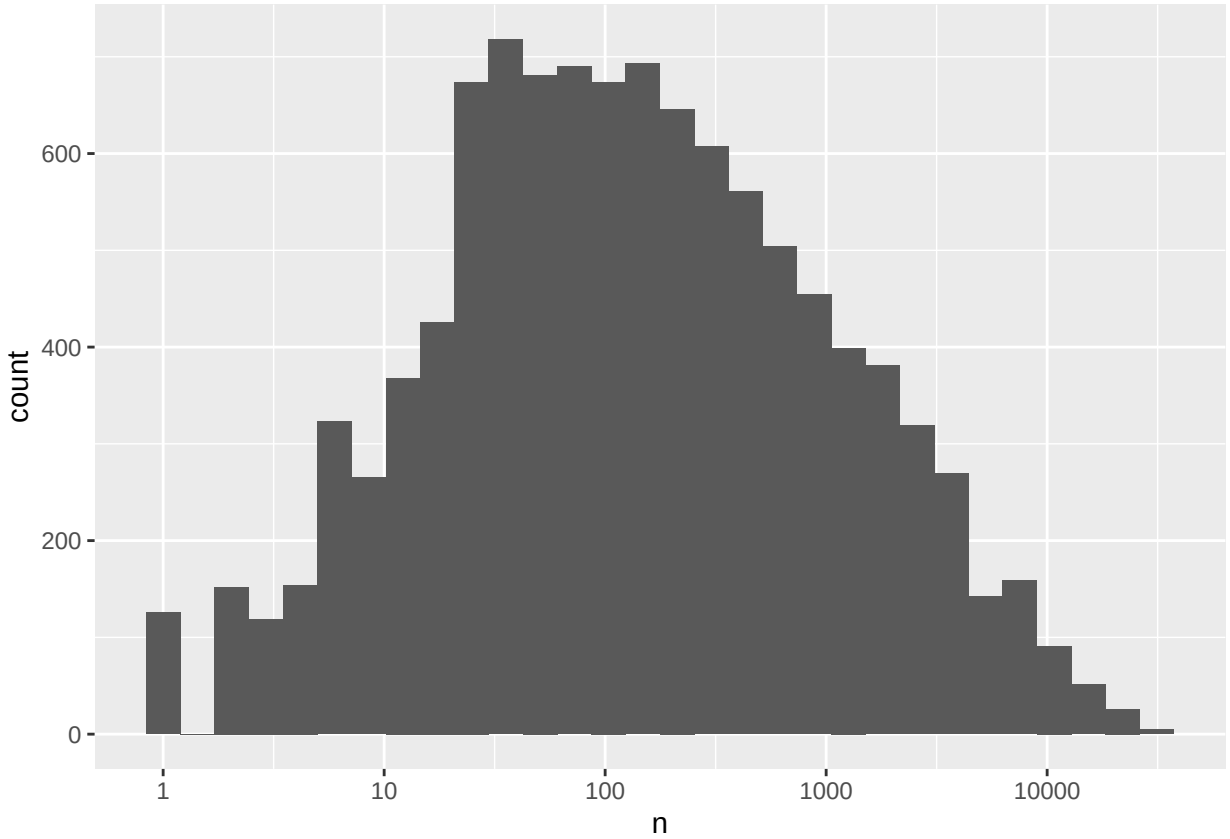
```
load("edx.RData")
##visualize count of users who rated
edx %>% group_by(userId) %>% summarise(n =n()) %>%
  ggplot(aes(n))+
  geom_histogram()+
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
##visualize count of movies rated
edx %>% group_by(movieId) %>% summarise(n =n()) %>%
  ggplot(aes(n))+
  geom_histogram()+
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Having visualized the scarcity of users rating all movies or vice versa, and the sparsity of our ratings matrix y, we can now proceed with modeling and analysis. First, the matrix y needs some preprocessing, we need to remove movie and user biases and replace Na values with -1 (assuming since a user didn't rate a movie it is not attractive for them, in opposed to replacing with 0 assuming that they are neutral).

The goal now is to fill the ratings matrix y with predicted ratings, so that a user who rated a couple of movies for example can now be recommended other movies having filled their respected row with predicted ratings. In this project a linear model (linear regression) is used to predict the ratings coupled with IRLBA which is an approximation of svd analysis. The model looks as follow:

Yij_pred = mu + bi + bu + rij, where:

Yij_pred is the predicted movie rating for user and and movie j, mu is the overall average of movie ratings, bi is the movie bias (given that there are movies rated by more users and vice versa), bu is the user bias (given that there are active users who rate more movies and vice versa), rij is the user-movie interactions matrix obtained from svd (given that captures correlation patterns between certain movies or genres)

In addition, to calculate mu, we just take the overall mean of rating column in edx dataset. On the other hand, to get bi we just group by movieId and substract the mean out of rating. Moreover, to get bu we group my userId then substract mu and bi from rating. Finally, to get rij we perform svd analysis which is of this form A = UDVt, where:

A is the matrix to be factorised U is the left singular vectors D singular values which represents vectors importance Vt (t for transposed) is the right singular vectors

After that, we use U*Vt to get rij. However, as mentioned above, svd analysis wasnt applicable due to the large data on hand and memory limitation. Therefore, a method called IRLBA was used to approximate the set of vectors U and Vt.

Results

This is the original base model we're using to predict users' ratings without the use of svd yet: Yij_pred = mu + bi + bu.

This model encompasses the overall ratings adding movie bias (movie with higher ratings get a higher bias and vice versa) and adding user bias (users who tend to rate movies higher get a higher bias and vice versa). However, his model lacks the ability to capture trends with respect to groups of movies tend to have similar ratings' patterns and groups of users tend to have similar ratings' patterns as well, which is what svd exactly provide.

Using the baseline model we get the following result:

```
##calculate overall mean of rating
mu <- mean(edx$rating)

##calculate movie bias call it bi
bi <- edx %>%
  group_by(movieId) %>%
  summarise(bi = mean(rating - mu)) %>% as.data.frame()

##calculate user bias call it bu
bu <- edx %>% left_join(bi, by = "movieId") %>%
  mutate(bu_u = rating - mu - bi) %>%
  group_by(userId) %>%
  summarise(bu = mean(bu_u)) %>% as.data.frame()
rm(edx)
load("final_holdout_test.RData")
final_holdout_test %>% left_join(bi, by = "movieId") %>%
  left_join(bu, by = "userId") %>%
  mutate(pred = mu + bi + bu) %>% summarise(RMSE(pred, rating))
```

```
##   RMSE(pred, rating)
## 1          0.8653488
```

```
rm(final_holdout_test)
```

On the other hand, adding the IRLBA analysis has the following model: Yij_pred = mu + bi + bu + rij.

It is a bit tricky to acquire the matrix rij, because of highlighted memory/hardware issues, however, it is doable with more work of chunking the matrix and work on a single chunk at a time.

First step is to have the preprocessed y matrix, having removed both movie and user biases plus handling Na values. Now we are ready to perform IRLBA to approximate right and left singular vectors U and Vt.

```
##dont run it takes ~12hours of work, matrix r is provided as a seperate file ready.
##irlba takes y the ratings matrix, and asks of how many singular values to approximate, here 800 were
rpca <- irlba(y, nv = 800, center = F, scale = F)

##rpca$u is our left singular vectors U and t(rpca$v) is our right singular vectors Vt
r <- rpca$u%*%t(rpca$v)

##align name columns of r with the same original movieIds in y matrix
colnames(r) <- colnames_y
```

After that, we are have our rij resedual matrix and we need to add it to the model described above.

```
##adding teh overall average mu and the user bias
r <- r+mu+bu$bu
rm(bu)

##adding movie bias by chunking r into two parts to preserve memory
r1 <- t(t(r[1:35000,])+bi$bi[match(as.integer(colnames(r)), bi$movieId)])
r2 <- t(t(r[35001:69878,])+bi$bi[match(as.integer(colnames(r)), bi$movieId)])
rm(r)
##now r is ready and basically it is y_hat (predicted ratings for all users and all movies)
r <- rbind(r1,r2)
rm(r1,r2, bi,colnames_y)
```

Now we are ready to test our model against the test set. Again, given the memory/hardware limitation, r had to be chunked into 6 parts and calculating mean square error for each chunk using this formula:

```
mse <- function(predicted, observed){
  sum((predicted - observed)^2)/nrow(final_holdout_test)
}
```

Note that to preserve the true essence of mse we divide each chunk by the total number of observations in final_holdout_test so that we can just sum the 6 mse parts into the total mse and take the square root of it to acquire the rmse needed.

```
mse1 = 0.1230827 ; mse2 = 0.1265856 ; mse3 = 0.1239886;
mse4 = 0.1249978; mse5 = 0.122252; mse6 = 0.1281323
##Final RMSE
sqrt(mse1+mse2+mse3+mse4+mse5+mse6)
```

```
## [1] 0.8654704
```

Nevertheless, the use of this technique did not improve results because it is an approximation after all, what we need is the actual right and left singular vectors of svd in order to capture the real interactions of users and movies which will in turn improve the results further.

Conclusion

Through data analysis and machine learning, this movie recommendation project successfully personalized suggestions for users. Techniques like linear model and svd showed promise in predicting preferences. however, due to the memory/hardware limitations, a tru svd factorisation couldn't be performed, instead and approximation called IRBLA provided approximate right and left singular vectors. Nevertheless, the use of this technique did not improve results because it is an approximation after all, what we need is the actual right and left singular vectors of svd in order to capture the real interactions of users and movies which will in turn improve the results further.