Data 612 - Project 1

Paul Perez

6/9/2020

Contents

The Recommender System

This system recommends movie's to film enthusiasts using the MovieLens `ml-latest-small` dataset. This set contains 100,836 ratings which spans 610 users, 9742 movies and 3,683 tag applications.The data is contained within 4 csv files; `links.csv`, `ratings.csv`, `movies.csv`, and `tags.csv`. To simplify the system in this project, we'll only be joining the `ratings.csv` and `movies.csv` on `movieId` field.

Data Ingestion, Selection, Manipulation

```
movies <- read.csv('https://raw.githubusercontent.com/ChefPaul/data612/master/Project%201/movies.csv')
ratings <- read.csv('https://raw.githubusercontent.com/ChefPaul/data612/master/Project%201/ratings.csv'
```

Preview of Movies & Ratings datasets

```
head(movies)
```

```
##   movieId                           title
## 1       1                Toy Story (1995)
## 2       2                  Jumanji (1995)
## 3       3         Grumpier Old Men (1995)
```

```
## 4       4           Waiting to Exhale (1995)
## 5       5 Father of the Bride Part II (1995)
## 6       6                        Heat (1995)
##                                       genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2                 Adventure|Children|Fantasy
## 3                             Comedy|Romance
## 4                       Comedy|Drama|Romance
## 5                                     Comedy
## 6                      Action|Crime|Thriller
```

```r
head(ratings)
```

```
##   userId movieId rating timestamp
## 1      1       1      4 964982703
## 2      1       3      4 964981247
## 3      1       6      4 964982224
## 4      1      47      5 964983815
## 5      1      50      5 964982931
## 6      1      70      3 964982400
```

Join Movies & Ratings datasets

```r
joined_data <- full_join(ratings, movies, by = 'movieId')
joined_data_preview <- head(joined_data, 10)
joined_data_preview %>% kable(caption = "Joined Data Preview") %>% kable_styling("striped", full_width =
```

Slice dataset to only contain UserId, Movie Title, and Movie Rating

```r
data <- joined_data[,c(1,5,3)]
data_preview <- head(data, 10)
data_preview  %>% kable(caption = "Sliced Dataset Preview") %>% kable_styling("striped", full_width = T
```

Identify Top 5 Most Rated Movies

In order to create a sparse dataset, but still ensure we have a sufficient number of ratings, we'll have to identify the Top 5 movies that were the most frequently rated.

```r
# top 5 movies
movie_freq <- table(joined_data$title) %>% as.data.frame() %>% arrange(desc(Freq))
top_5 <- movie_freq[1:5,][1]
top_5  %>% kable(caption = "Top 5 Frequently Rated Movies Preview") %>% kable_styling("striped", full_w
```

Now that we've identified the most frequently rated movies, we can filter the dataset to only contain these movies.

Table 1: Joined Data Preview

userId	movieId	rating	timestamp	title	genres
1	1	4	964982703	Toy Story (1995)	Adventure|Animation|Children|C
1	3	4	964981247	Grumpier Old Men (1995)	Comedy|Romance
1	6	4	964982224	Heat (1995)	Action|Crime|Thriller
1	47	5	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
1	50	5	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller
1	70	3	964982400	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
1	101	5	964980868	Bottle Rocket (1996)	Adventure|Comedy|Crime|Romar
1	110	4	964982176	Braveheart (1995)	Action|Drama|War
1	151	5	964984041	Rob Roy (1995)	Action|Drama|Romance|War
1	157	5	964984100	Canadian Bacon (1995)	Comedy|War

Table 2: Sliced Dataset Preview

userId	title	rating
1	Toy Story (1995)	4
1	Grumpier Old Men (1995)	4
1	Heat (1995)	4
1	Seven (a.k.a. Se7en) (1995)	5
1	Usual Suspects, The (1995)	5
1	From Dusk Till Dawn (1996)	3
1	Bottle Rocket (1996)	5
1	Braveheart (1995)	4
1	Rob Roy (1995)	5
1	Canadian Bacon (1995)	5

Table 3: Top 5 Frequently Rated Movies Preview

Var1
Forrest Gump (1994)
Shawshank Redemption, The (1994)
Pulp Fiction (1994)
Silence of the Lambs, The (1991)
Matrix, The (1999)

Table 4: Subset of Data Preview

userId	title	rating
1	Pulp Fiction (1994)	3
1	Forrest Gump (1994)	4
1	Silence of the Lambs, The (1991)	4
1	Matrix, The (1999)	5
2	Shawshank Redemption, The (1994)	3
4	Pulp Fiction (1994)	1
4	Silence of the Lambs, The (1991)	5
4	Matrix, The (1999)	1
5	Pulp Fiction (1994)	5
5	Shawshank Redemption, The (1994)	3

Table 5: User-Item Matrix Preview

userId	Forrest Gump (1994)	Matrix, The (1999)	Pulp Fiction (1994)	Shawshank Redemption, The (1994)	Silence of the Lambs, The (1991)
1	4	5	3	NA	4
2	NA	NA	NA	3	NA
4	NA	1	1	NA	5
5	NA	NA	5	3	NA
6	5	NA	2	5	4
7	5	NA	NA	NA	5

```
sub_data <- data %>% filter(title %in% c("Forrest Gump (1994)", "Shawshank Redemption, The (1994)", "Pu
sub_data_preview <- head(sub_data, 10)
sub_data_preview  %>% kable(caption = "Subset of Data Preview") %>% kable_styling("striped", full_width
```

Finally, since we have a subset of the data, we can create a user-item matrix utilizing the `spread()` function from the `tidyr` package.

```
wide_data <- sub_data %>% spread(title, rating)
wide_data_preview <- head(wide_data)
wide_data_preview %>% kable(caption = "User-Item Matrix Preview") %>% kable_styling("striped", full_wid
```

Split into Training & Test datasets using an 80/20 Ratio

We can utilize the `sample()` function to split the data into training and test sets.

```
set.seed(123)
train_sample <- sample(x = c(TRUE, FALSE), size = nrow(wide_data), replace = TRUE, prob = c(0.8, 0.2))
train_id <- as.matrix(wide_data[train_sample,])
test_id <- as.matrix(wide_data[!train_sample,])
```

While splitting, the `userId` field will remain within the matrix, but we can remove that to further build out the system and run calculations on each matrix.

Table 6: Training User-Item Matrix Preview

	Forrest Gump (1994)	Matrix, The (1999)	Pulp Fiction (1994)	Shawshank Redemption, The (1994)	Silence of the Lambs, The (1991)
1	4	5	3	NA	4
2	NA	NA	NA	3	NA
3	NA	1	1	NA	5
6	5	NA	NA	NA	5
7	3	NA	4	5	4
9	5	NA	NA	4	5

Table 7: Testing User-Item Matrix Preview

	Forrest Gump (1994)	Matrix, The (1999)	Pulp Fiction (1994)	Shawshank Redemption, The (1994)	Silence of the Lambs, The (1991)
4	NA	NA	5	3.0	NA
5	5.0	NA	2	5.0	4.0
8	3.5	0.5	1	NA	NA
11	4.0	NA	3	3.0	4.0
16	2.0	4.0	NA	NA	NA
20	4.5	4.0	4	4.5	4.5

```
train <- train_id[,2:6]
train_preview <- head(train)
train_preview %>% kable(caption = "Training User-Item Matrix Preview") %>% kable_styling("striped", full
```

```
test <- test_id[,2:6]
test_preview <- head(test)
test_preview %>% kable(caption = "Testing User-Item Matrix Preview") %>% kable_styling("striped", full_
```

Calculate the Raw Average (Training Set)

We'll need to calculate the raw average (mean) for each user-item combination in the training set.

```
raw_avg <- mean(as.matrix(train), na.rm = TRUE)
```

RMSE for Raw Average

Since we're going to have to calculate the RMSE multiple times, we can create the function below.

```
rmse <- function(m, o){
  sqrt(mean((m - o)^2, na.rm = TRUE))
}
```

Training Set RMSE

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Figure 1: RMSE Formula

```
train_rmse <- rmse(raw_avg, train)
train_rmse
```

```
## [1] 0.848821
```

Test Set RMSE

```
test_rmse <- rmse(raw_avg, test)
test_rmse
```

```
## [1] 0.959381
```

User & Item Bias from Training Set

Now that we have the mean for the training set, we can calculate each user & item bias.

```
user_bias <- rowMeans(train, na.rm = TRUE) - raw_avg
item_bias <- colMeans(train, na.rm = TRUE) - raw_avg
```

Calculating the Baseline Predictors

```
baseline_predictors <- user_bias + item_bias + raw_avg
```

Since ratings cannot be less than 1 or greater than 5, we can adjust any value that is out of range.

```
baseline_predictors[baseline_predictors < 1] <- 1
baseline_predictors[baseline_predictors > 5] <- 5
```

Calculating the RMSE for the Basline Predictors of the Training Set

```
train_baseline_rmse <- rmse(raw_avg, baseline_predictors)
```

Calculate the Raw Average (Test Set)

We'll need to calculate the raw average (mean) for each user-item combination in the test set.

```
test_avg <- mean(as.matrix(test), na.rm = TRUE)
```

User & Item Bias from Training Set

Now that we have the mean for the test set, we can calculate each user & item bias.

```
test_user_bias <- rowMeans(test, na.rm = TRUE) - test_avg
test_item_bias <- colMeans(test, na.rm = TRUE) - test_avg
```

Calculating the Baseline Predictors

```
test_baseline_predictors <- test_avg + test_item_bias + test_user_bias
```

```
## Warning in test_avg + test_item_bias + test_user_bias: longer object length is
## not a multiple of shorter object length
```

Since ratings cannot be less than 1 or greater than 5, we can adjust any value that is out of range.

```
test_baseline_predictors[test_baseline_predictors < 1] <- 1
test_baseline_predictors[test_baseline_predictors > 5] <- 5
```

Calculating the RMSE for the Basline Predictors of the Test Set

```
test_baseline_rmse <- rmse(test_avg, test_baseline_predictors)
```

Summary

```
test_eval <- (1 - (test_baseline_rmse / test_rmse)) * 100
test_eval
```

```
## [1] 24.96047
```

```
train_eval <- (1 - (train_baseline_rmse / train_rmse)) * 100
train_eval
```

```
## [1] 19.83473
```

After creating the recommender system, we can see that the test evaluation show's a 24.96% improvement while the training set's evaluation shows a 19.83% improvement.

References

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872

Source Code GitHub Repository