MOVIE RECOMMENDER SYSTEM USING APACHE SPARK-STANDALONE CLUSTER

Akash Bachu

Introduction

It is easy to recommend things to others when people ask someone and teaching the same task to a computer in the real task. Recommender systems or Recommender Engines have been the most commonly used algorithm over the decade that predicts the rating that a user would give to an item. In this project I would be recommending movies from a Movielens dataset which has list of movies, users, ratings and links to internet movie database(IMDB) and the movie database(tmdb) webid's. Movielens is a project developed by Grouplens, a research laboratory at University of Minnesota. It has different files(movies.csv, ratings.csv, links.csv, genome.csv, tags.csv) but our focus will be on ratings, movies and links. The dataset has following features:

Number of different users : 270,896
Number of different movies : 45115
Number of Total Ratings : 26024289.

The main idea of the project is to achieve Automated content recommendation system based on user ratings by using Collaborative filtering and creating a matrix of users and items filled in with ratings for existing data.

Background

The main approach for building a recommendation is "*Alternating Least Squares for Collaborative Filtering".* Few background information on the algorithms used for building the recommendation engine will be provided.

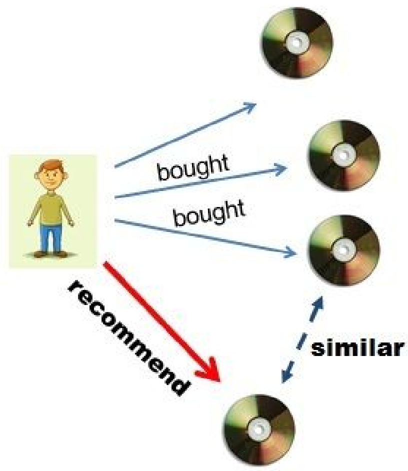
Collaborative Filtering:

Predictions are made on the interests of a user by collecting his preferences and similarities between users and products. The assumption is that a user who rates few movies on similar genres will also have similar opinions on that movies he haven't seen.

For example- if a user has given rating for 2 horror, 3 thriller and 2 adventure similar movies with genres will be recommended which he haven't seen them.

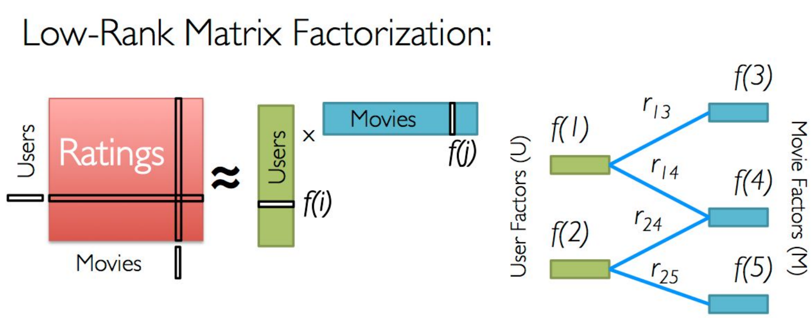
MOVIE RECOMMENDER SYSTEM USING APACHE SPARK-STANDALONE CLUSTER

Akash Bachu



Alternating Least Squares

The ALS algorithm provides Collaborative filtering between users and products to find products that customers like, based on previous ratings. ALS is Matrix Factorization algorithm which decomposes large matrix into product of matrices. ALS has few parameters which are has low dependency values on the size of dataset.



Iterate:

$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

Root Mean Square Algorithm

It is used to find the Accuracy of the model. It computes the mean value of all the differences squared between the true and predicted ratings and then proceeds to calculate the square root. We use RMS error algorithm to reduce the large error in the dataset.

MOVIE RECOMMENDER SYSTEM USING APACHE SPARK-STANDALONE CLUSTER

Akash Bachu

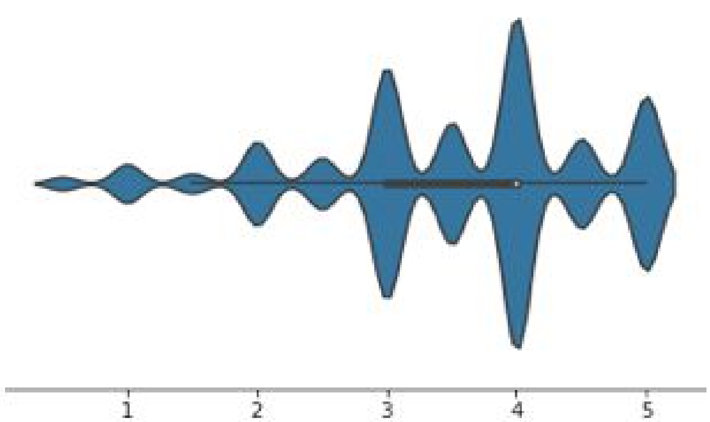
$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

Analysis, Design and Results

For this project I had used the spark standalone cluster for the analysis part. I had set up spark manually and assigned storage for each worker. For the project I had used a cluster of 1 master node and 2 Slave nodes each of 8GB memory. Jupyter notebook was used for the data preprocessing. The languages used for this project is Python(Pyspark, Pandas, Numpy, Matplotlib and seaborn). The initial step for the Analysis is to load all the necessary packages and the data. The dataset has ratings from different users from 0-5 scale including the float values. I had to count how many users had rated for each rating. One thing to be noted is that all the users did not rate for the movies so I had to omit the null values and found out the average of ratings using the average function of python pandas and it seems that the average of all the ratings is 3.52.

The two images below show the complete description of how many users had given ratings for each rating. We can also see that only 2.1% of the matrix is filled.

rating	count
5.0	3812499
4.5	2170441
4.0	6998802
3.5	3116213
3.0	5256722
2.5	1255358
2.0	1762440
1.5	403607
1.0	843310
0.5	404897



	mat_users	mat_movies	mat_ratings	matrix_size	perct_matrix
0	270896	45115	26024289	12221473040	0.212939

Building a recommender system:

MOVIE RECOMMENDER SYSTEM USING APACHE SPARK-STANDALONE CLUSTER

Akash Bachu

It's been given that only 2.1% of the matrix has been filled and rest of the users did not rate the movies. In order to solve this problem many trial and errors were done but only one gave an efficient result for the recommender model. Removing the Null and filling them with zero had not been fruitful so the empty value were filled with the average ratings to give efficient results.

User/Item	1	2	3	4	5
1	4	3	5	1	?
2	4	5	4	?	4.5
3	2	?	?	4	?
4	0.5	4	?	4.5	?

User/Item	1	2	3	4	5
1	4	3	5	1	3.52
2	4	5	4	3.52	4.5
3	2	3.52	3.52	4	3.52
4	0.5	4	3.52	4.5	3.52

This gave a good results while training the model

Initially I had used the ALS algorithm for training the model and to provide necessary recommendation by giving few ALS parameters.

userId	movieId	rating	timestamp	prediction
575	148	4.0	1012605106	3.9246416
232	463	4.0	955089443	3.9609222
452	463	2.0	976424451	2.4503825
380	463	3.0	968949106	3.0178618
534	463	4.0	973377486	3.7882247
242	463	4.0	956685706	3.7452123
30	463	4.0	945277405	3.6511815
311	463	3.0	898008246	2.934478
85	471	3.0	837512312	2.9595428
588	471	3.0	842298526	3.6811454
126	471	5.0	833287141	3.9334629
460	471	5.0	1072836030	3.9450562
350	471	3.0	1011714986	3.2347164
548	471	4.0	857407799	3.5334275
602	471	3.0	842357922	3.9636881
285	471	5.0	965092130	3.8335536

MOVIE RECOMMENDER SYSTEM USING APACHE SPARK-STANDALONE CLUSTER

Akash Bachu

These were the predicted values when I ran the fit() method to train the model and accuracy of this model is (RMSE) 61%. To improve the model I had split the dataset into training(70%) and testing(30%) model and then tested the model against the both datasets and the results were lot more better.

```
Loop 1: rmse = 0.9218727746836262
[0.9218727746836262]
Loop 2: rmse = 0.9123482476765431
[0.9218727746836262, 0.9123482476765431]
Loop 3: rmse = 0.9110819374634826
[0.9218727746836262, 0.9123482476765431, 0.9110819374634826]
Loop 4: rmse = 0.9179557259086534
[0.9218727746836262, 0.9123482476765431, 0.9110819374634826, 0.9179557259086534]
Loop 5: rmse = 0.9078096238719452
[0.9218727746836262, 0.9123482476765431, 0.9110819374634826, 0.9179557259086534, 0.9078096238719452]
[0.9218727746836262, 0.9123482476765431, 0.9110819374634826, 0.9179557259086534, 0.9078096238719452]
None
CPU times: user 95 ms, sys: 27 ms, total: 122 ms
Wall time: 53.5 s
```

In this way a better accuracy was obtained for the model(92%).

The below are few predictions on userID's

userID = 102

```
[Row(userId=102, movieId=306, timestamp=1494286154, prediction=4.91982889175415),
 Row(userId=102, movieId=44555, timestamp=1494286154, prediction=4.77939510345459),
 Row(userId=102, movieId=319, timestamp=1494286154, prediction=4.753486156463623),
 Row(userId=102, movieId=6016, timestamp=1494286154, prediction=4.67483377456665),
 Row(userId=102, movieId=1060, timestamp=1494286154, prediction=4.66062593460083),
 Row(userId=102, movieId=27773, timestamp=1494286154, prediction=4.652323246002197),
 Row(userId=102, movieId=56782, timestamp=1494286154, prediction=4.606232166290283),
 Row(userId=102, movieId=4226, timestamp=1494286154, prediction=4.60533332824707),
 Row(userId=102, movieId=69481, timestamp=1494286154, prediction=4.570173740386963),
 Row(userId=102, movieId=3000, timestamp=1494286154, prediction=4.560894012451172)]
```

userID = 482

```
[Row(userId=482, movieId=1223, timestamp=1513409519, prediction=4.812012672424316),
 Row(userId=482, movieId=1221, timestamp=1513409519, prediction=4.79148101806640 6),
 Row(userId=482, movieId=106782, timestamp=1513409519, prediction=4.743971824645996),
 Row(userId=482, movieId=318, timestamp=1513409519, prediction=4.649720191955566),
 Row(userId=482, movieId=969, timestamp=1513409519, prediction=4.627016067504883),
 Row(userId=482, movieId=35836, timestamp=1513409519, prediction=4.6183319091796875),
 Row(userId=482, movieId=2917, timestamp=1513409519, prediction=4.573208332061768),
 Row(userId=482, movieId=720, timestamp=1513409519, prediction=4.559764385223389),
 Row(userId=482, movieId=78499, timestamp=1513409519, prediction=4.5483174324035645),
 Row(userId=482, movieId=69122, timestamp=1513409519, prediction=4.545524597167969)]
```

userOD = 217570

```
[Row(userId=217570, movieId=162864, timestamp=1513407952, prediction=4.813087463378906),
 Row(userId=217570, movieId=104636, timestamp=1513407952, prediction=4.802556037902832),
 Row(userId=217570, movieId=165069, timestamp=1513407952, prediction=4.789482116699219),
 Row(userId=217570, movieId=158310, timestamp=1513407952, prediction=4.778846740722656),
 Row(userId=217570, movieId=90341, timestamp=1513407952, prediction=4.758243560791016),
 Row(userId=217570, movieId=170705, timestamp=1513407952, prediction=4.7456479072570 8),
 Row(userId=217570, movieId=139090, timestamp=1513407952, prediction=4.742129325866699),
 Row(userId=217570, movieId=136445, timestamp=1513407952, prediction=4.7405548095703125),
 Row(userId=217570, movieId=116002, timestamp=1513407952, prediction=4.735102653503418),
 Row(userId=217570, movieId=139098, timestamp=1513407952, prediction=4.703791618347168)]
```

These userID’s were randomly generated as I used seed function.

Scalability Challenges

They were two scalability challenges occurred in during the Analysis. Since the dataset is huge there was a difficulty to the analysis in the local machine. A spark cluster with 1 master node,

	mat_users	mat_movies	mat_ratings	matrix_size	perct_matrix
0	270896	45115	26024289	12221473040	0.212939

2 slave nodes were created. On the local machine, the time taken to run the full dataset was around 8hrs and sometimes and error is displayed showing that the memory is filled. With the spark cluster It took 10 min for the whole dataset to run. The other scalability issue was the missing values in the Matrix which gave me a low accuracy for the prediction model. Using the ALS algorithm helped to increase the accuracy rate of the model by filling the missing values with average value of ratings and by setting few ALS parameters. With the ALS algorithm the mean of the ratings also changed to 3.9

References

MOVIE RECOMMENDER SYSTEM USING APACHE SPARK-STANDALONE CLUSTER

Akash Bachu

https://spark.apache.org/docs/preview/ml-collaborative-filtering.html

http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/

https://www.elenacuoco.com/2016/12/22/alternating-least-squares-als-spark-ml/

https://www.davidadrian.cc/posts/2017/08/how-to-spark-cluster/

https://grouplens.org/datasets/movielens/