Movie Recommender System Using ALS, Cosine Similarity

Team 21

Abhijit Nair,
Kshitij Khurana,
Satish Kumar

Introduction

Recommender system is a very important tool in marketing. As Ecommerce is becoming more and more popular, every ecommerce website has/wants a recommendation system which targets specific users. Website like Netflix, YouTube etc. use the previously seen/liked videos by the user to recommend them similar videos. It is very interesting to see how accurate these recommendations are and the underlying motivation of this project was to learn how these systems work and to implement one of our own. There are two major ways the collaborative filtering approach can be performed - using Alternating Least Square algorithm or using Cosine similarity approach. In our project we have implemented both these approaches.

Goals

A) We will definitely accomplish

A fully functioning recommender system which can run on both the ALS approach and the cosine similarity approach and a study of performance and accuracy of both these systems.
We have achieved this, the recommender system works.

B) We will likely accomplish.

Implementing both these systems without using the MLlib library in Spark.
We have implemented the cosine similarity and the ALS algorithm without using the MLlib library.

C) We would like to ideally accomplish

Implementing a method where both the above given methodology are used to increase efficiency and to make an interactive User Interface where users can input new movies to the database and have an interactive environment.
We could not achieve this goal due to time constraints.

Challenges

- The biggest challenge was understanding the Cosine filtering Algorithm and implement it on spark RDDs.
- Another challenge was the understanding the working of Collaborative filtering and understanding the Underlying mathematics.

Cool Enhancements

- We have written a code to find the optimal number of neighbors for the neighborhood matrix, which will give us a good RMSE value.
- We used a combination of cartesian matrix and filtering to get user pairs for Cosine similarity Matrix.

Project Description:

Framework Used:

All the algorithms are implemented on SPARK

Data Profile:

The Movielens dataset has the following files:

1) Ratings
2) Movie
3) Links
4) Genome Tags
5) Genome Score

Although there five files in our dataset, but for creating our recommenders system we used rating and movies files.

Rating:

	userId	movieId	rating	timestamp
0	1	122	2.0	945544824
1	1	172	1.0	945544871
2	1	1221	5.0	945544788
3	1	1441	4.0	945544871
4	1	1609	3.0	945544824

Movies:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In our project we implemented collaborative filtering algorithms as this is one of the most commonly used algorithm in the industry. There are two types of collaborating filtering algorithms.

User-User Collaborative filtering: Here we find look alike customer to every customer and offer products which first customer's look alike has chosen in past. This is done by similarity scores between customers or users.

Item-Item Collaborative filtering: It is quite similar to previous algorithm, but instead of finding customer look alike, we try finding item look alike.

User Based Collaborating Recommenders System using Cosine Similarity:

In the algorithm, the similarities between different users in the dataset are calculated by using one of a number of similarity measures, such as Cosine similarity, Jaccard Similarity.

As mentioned above there are a number of different mathematical formulations that can be used to calculate the similarity between two users. In our approach we used cosine similarity for measuring similarity.

Cosine Similarity:

Initially we took rating dataset and calculated the cosine similarity between each of the users.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Further we calculated top k similar users
This top K similar users is our neighborhood matrix
The UserID for which we have to predict is given as an argument, and based on his K neighbors we recommend top 5 movies.
We use the below given aggregate function to calculate the ratings for our User for movies he has not seen/rated. We are also taking into consideration the fact that if other users are stingy/liberal in our calculation.

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} simil(u, u')(r_{u',i} - \bar{r_{u'}})$$

$$k = 1 / \sum_{u' \in U} |simil(u, u')|$$

The top 5 predictions are outputted.

Alternating Least Squares:

for $u = 1 \ldots n$ **do**

$$x_u = \left(\sum_{r_{ui} \in r_{u*}} y_i y_i^\intercal + \lambda I_k \right)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i$$

end for

for $i = 1 \ldots m$ **do**

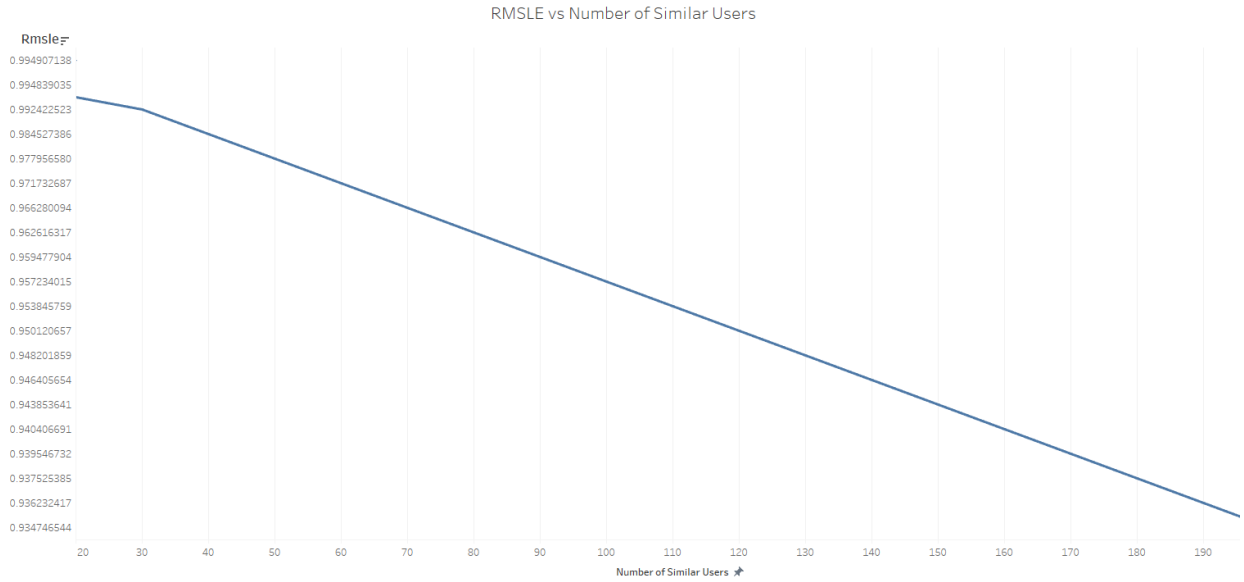
$$y_i = \left(\sum_{r_{ui} \in r_{*i}} x_u x_u^\intercal + \lambda I_k \right)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u$$

For ALS implementation we created an initial data matrix using standalone python program. Further the output of the python program is used as an input to the ALS algorithm implementation. In ALS implementation, first we randomly initialize the user matrix and movie matrix. Then, using the above formula we evaluate the user matrix and movie matrix by keeping one while evaluating the other for 10 iterations. We have evaluation the model through RMSE metric. We are calculating RMSE error for each iteration and then the final average rms value and printing in our final output.

Results and Evaluation of Models

Optimizing the Number of Neighbors for Cosine Similarity:

We can select the number of neighbors for the Neighborhood matrix, more neighbors gives us less error as seen from the below graph. We run a loop starting at 10 to 200 with a step size of 10 to see how many neighbors gave us the lowest RMSE. Higher number of neighbors reduces the RMSE but we also have to consider the computational cost.



RMSE value of Cosine similarity with 200 Neighbors: 0.9347

RMSE value of ALS algorithm with 10 iterations: 0.373068046195

Output for Cosine Similarity:

Movies recommended for User ID 3

```
[(3, [u'Hands on a Hard Body (1996)', u'Matewan (1987)', u"Antonia's Line
(Antonia) (1995)", u'Amateur (1994)', u'Love & Human Remains (1993)'])]
```

Output for ALS:

```
The movie predicted for user_id 1: for movie_id 215: Predicted rating is 0.153059
The movie predicted for user_id 2: for movie_id 100: Predicted rating is 3.088847
The movie predicted for user_id 3: for movie_id 403: Predicted rating is 1.666959
The movie predicted for user_id 4: for movie_id 436: Predicted rating is 2.504167
The movie predicted for user_id 5: for movie_id 100: Predicted rating is 1.880163
The movie predicted for user_id 6: for movie_id 178: Predicted rating is 0.630759
The movie predicted for user_id 7: for movie_id 70: Predicted rating is 2.060094
The movie predicted for user_id 8: for movie_id 355: Predicted rating is 2.816697
The movie predicted for user_id 9: for movie_id 58: Predicted rating is 1.461946
```

Conclusion and future scope

We have implemented both Cosine similarity and ALS algorithm. We would like to add the functionality of adding a new user to the dataset. Also we would have liked to implement a cleaner user interface where the user can give ratings to different movies and then get a new recommendation based on his new ratings.

Tasks Distribution:

Abhijit Nair : Project Report, Cosine Similarity Algorithm

Ksitij Khuranna: Data Prepation and exploration, Cosine Similarity Algorithm

Satish Kumar: ALS Algorithm

References:

1. https://www.youtube.com/watch?v=q97VFt56vRs&list=PLuKhJYywjDe96T2L0-zXFU5Up2jqXlWI9

2. https://www.packtpub.com/books/content/building-recommendation-engine-spark

3. http://www.infofarm.be/articles/alternating-least-squares-algorithm-recommenderlab