Movie Recommendation by Collaborative Filtering using the Netflix Data

Nitin Pathania

December 8, 2020

Abstract

A recommendation model, in simple terms, is an algorithm which aims to provide the most relevant and relatable information to a user depending on the behaviour of the user. Companies like Netflix and Google have a huge database of the behaviours of data collected to be able to perform state-of-the-art recommendations so that they can display the most relevant content or services to the users to increase the engagement.These days whether you look at a video on YouTube, a movie on Netflix or a product on Amazon, you are going to get recommendations for more things to view, like or buy. You can thank the advent of machine learning algorithms and recommender systems for this development.Recommender systems are far-reaching in scope, so we are going to zero in on an important approach called **collaborative filtering**, which filters information by using the interactions and data collected by the system from other users. It is based on the idea that people who agreed in their evaluation of certain items are likely to agree again in the future.

1 Project Overview

1.1 Motivation

Considering Data Engineering as an integral part of data world, Spark plays a big role in big data processing. The project on Netflix recommendation program was one of the top problems of last decade which made lot of noise in the tech world. Knowing the fact that the dataset has 3.25 million records makes it a good data engineering problem, using spark cluster and python ways. People may be interested in a product but would often not comb through multitudes of information just to find what they need. In other words, the sales of several companies are dependent on their ability to present the user preferences in their eyesight. This project intends to predict what the users' opinion is on Netflix movies and as such recommend movies the user may have never seen.

1.2 Problem Statement

We all know recommendations system play a big part in online business specially e-commerce and streaming services. Companies need to recommend the products which would attract users and compel them to buy. Looking at the customer side, they want to get recommendations based on their taste and liking. The dataset consists of ratings given by users to different movies, also the details of the movies. It consists of a total 3.25 million records for training and 100,000 records for testing. In this case, collaborative filtering techniques like item-item filter and user-user filter to compare and use the best among them. Along with-it using cosine distance similarity between the records to give the best possible recommendations. Finally applying Alternating Least Squares algorithm to get the results. By doing this project, I got an idea about the entire workflow of a recommendation system.

2 Documentation of approach

Importing Libraries

We had to install libraries sklearn, matplotlib and seaborn explicitly via pip.

Problem 1

Step 1

Data uploads and pre-processing:

1. Create an Amazon S3 bucket and upload the data. The data consist of one training, one testing set and a movie data that has the titles and the year of the release.

2. SSH into the master node of the EMR and set up your Jupyter Notebook with pyspark.

3. Load the data from the S3 and set up an schema to hold the data.

4. Create a dataframe to hold the data.

5. We map attempt to cast the latitude and longitude to floats.

Step 2

Now that we have all the data set up, we will get a view our data is distributed.

First, we will look at a graph of the counts of movies in the training dataset

Next,lets look at a graph of the counts of users in the test set based on their rating.

It seems that most of the users in our testing dataset did not rate the movies they watched.

Finally, we will take a look at how many movies users in the training dataset seem to be watching.

Well, now that we have seen that we will start the actual work.

1. We will implement the ALS model to predict the ratings of users in the testing dataset.

2. To get the best parameters that can predict for the model, I varied the rank data and the maxiter parameters.

The best model with a rank of 0.8, maxIter of 10 and a regParam of 0.09 yielded an accuracy score of 0.849.

Problem 2

Step 1

I would be attempting to build a better model ie a model whose rate prediction is more likely to be true. But first- some digging.

There are 1821 different movies and 28978 different users in our train dataset, while there are 1701 movies and 27555 different users in the train data.

- To determine the best model for this system, I will be finding the estimated average overlap of items for users and the estimated average overlap of users.

- To find these, I will be creating a dataframe whose row index is the user, and column is the movies and value represent the rate for that movie by the user or null if there is no rate present.

To find the average for a user with id 1664010, simply select the row that corresponds to this user and sum up all the values in that row and divide by the count of non-zeros.

We will do this for 4 more users and average them and this will be our estimated average overlap of item rated by our 5 selected users. We will do this for the items in our movie user rating Dataframe and from this we will find our estimated average overlap of users for items.

As expected, the overlap of users that rated items is higher than the overlap of items rated by users. This is because, often a user's rate on an item is influenced mostly by personal preference. ie a user that likes the cartoons is likely to rate the movie Tom and Jerry relatively high. Likewise, in the inverse, a user that hates cartoons is likely to rate the cartoon with a lower rate. the average overall for both is these users is like to be fairly even, However, the overlap of items rated by users will have a combination of very high and very low rates and this is likely to result in an overall high average

The Selection

I selected the Matrix factorization approach using SVD simply because of its ability to scale across the large data.

The similarity measure- the user-user or item-item matrix for our dataset with more than 1000 distinct users would require a lot of computation and memory. This approach considering the size of our data would not be the best option.

I will be normalizing my dataset because, not only does our testing data contains several nans as a result of not being rated, but also, every time a new user is entered, without any prior rated information, without normalizing the model most likely cannot predict any movie to this new user.

Finally, my SVD approach is because this approach allows for the dataset to be scaled and appropriately reduces the large dimensions to a smaller set that is easily scalable through.

Normalization

To normalize my data, I found the mean of each user by movie row and subtracted the mean value for each rating.

Problem 3

Step 1

To ensure my mode, will be predicting correcting, I further split my training data and created a testing set from this.

Using the SVD model simply consist of factorizing the matrix into smaller dimensions of based on relatedness and using this relatedness to predict the rate of new ones.

I will be using the SVD model found in the Scipy library

The data is pivoted before it is passed in the model.

Once we have the predicted ratings, it is important that we add the mean value back to it.

Once that is done we can view our predictions and check how well they performed. The accuracy score above was received from our split, train data set. Now let's try this on our actual testing.

Finally i tried to predict new movies for a user based on the model.

3 System Configuration

Configuring the system consisted of setting up a Standard Amazon EMR cluster m5.2xlarge, 3 slave nodes, 1 master node and Amazon S3 bucket for storing the data.

4 Big data application/Dataset

4.1 Datasets description

This dataset is a subset of the data provided as part of the Netflix Prize. TrainingRatings.txt and TestingRatings.txt are respectively the training set and test set.Each of them has lines having the format: MovieID,UserID,Rating.Each row represents a rating of a movie by some user. The dataset contains 1821 movies and 28978 users in all. Ratings are integers from 1 to 5. The training set has 3.25 million ratings, and the test set has 100,000. Use the training set to predict the ratings provided in the test set. Note that your predicted ratings do not have to be integers. You must look to achieve best performance on the Mean Absolute Error and the Root Mean Squared Error metrics.

4.2 Implementation/Execution

The project is done in Jupyter notebook created on AWS EMR Cluster.The implementation of project starts with creating an EMR cluster on AWS:

1. Choose Create cluster.

2. On the Create Cluster - Quick Options page, accept the default values except for the following fields: • Enter a Cluster name. • In Software configuration select the applications as spark. • Under Security and access, choose the EC2 key pair that you created in Create an Amazon EC2 Key Pair.

3. Choose Create cluster. Create EMR notebook using EMR Cluster: EMR Notebook: EMR Notebooks, a managed environment, based on Jupyter Notebooks that allows data scientists, analysts, and developers to prepare and visualize data, collaborate with peers, build applications, and perform interactive analysis using EMR clusters. EMR Notebooks is pre-configured for Spark. It supports Spark magic kernels allowing you to interactively run Spark jobs on EMR clusters written in languages such as PySpark, Spark SQL, Spark R, and Scala.

Steps to create EMR notebook:

1. Choose Notebooks, Create notebook.

2. Enter a Notebook name and an optional Notebook description.

3. select Choose, select a cluster from the list (cluster created above), and then Choose cluster. Only clusters that meet the requirements are listed.

4. For Security groups, choose Use default security groups.

5. For AWS Service Role, leave the default or choose a custom role from the list. For more information, see Service Role for EMR Notebooks.

6. For Notebook location choose the location in Amazon S3 where the notebook file is saved. Open the Notebook created and select pyspark in jupyter notebook.

4.3 Results/Discussion

We were able to get the mean absolute error as 0.019038633968441944 and mean squared error as 0.1384366719810287 at the end.

5 Final Conclusion/Lessons learned/Future work

Factorizing the data seems to make the data easily manageable, however, our testing data is predominantly 0 and as such scaling will be relatively easier. It shows significantly as the trained test data seemed to have had a relatively worse performance than the test data. Also, normalizing seemed to have made it easier to predict for user who only have a 0 ratings. Recommendation algorithms provide an effective form of targeted marketing by creating a personalized shopping experience for each customer. For large retailers like Amazon, Netflix a good recommendation algorithm is scalable over very large customer bases and product catalogs, requires only sub second processing time to generate online recommendations, is able to react immediately to changes in a user's data, and makes compelling recommendations for all users regardless of the number of purchases and ratings.

References

[1] https://towardsdatascience.com/tensorflow-for-recommendation-model-part-1-19f6b6dc207d

[2] https://medium.com/@tomar.ankur287/item-item-collaborative-filtering-recommender-system-in-python-cf3c945fae1e

[3] https://spark.apache.org/docs/2.4.7/sql-programming-guide.html

[4] https://medium.com/sfu-cspmp/recommendation-systems-user-based-collaborative-filtering-using-n-nearest-neighbors-bf7361dc24e0
https://spark.apache.org/docs/2.4.7/sql-programming-guide.html

[5] https://www.netflixprize.com/