Matrices, Least Squares, Recommendation systems and the spark between them

M Jayantakumar
B20204,Indian Institute of Technology Mandi

Abstract— **An information filtering system that provides recommendations for items most relevant to a specific user is called a recommendation system. In this article, we explore and review the progress and process of our Final Project. In addition, we discuss literature on Recommendation Systems and how we have applied it to our own music recommendation system. Throughout the course of the project development, we will discuss various methods and ideologies that contribute to creating a recommendation system. We will also build the concept from the ground up while exploring various technical and subjective learning**

I. INTRODUCTION

Music a form of art that ties people across the boundaries of space, time and culture combined with the ability to entice human emotions and memories has been an interesting topic of study for a long time now. Today in this digital era, music impregnates most spheres of daily life and social activity. Listening has become a more complex act of being in the world. The advent of Big Data and the developments in large-scale music streaming services presents us with the opportunity to connect the dots of musical behavior on biological, social, and cultural levels. Through the use of these approaches and technologies, along with access to natural listening data gathered across a variety of cultures, geographies, and economic settings as well as song level metadata, we will be able to develop robust insights into music through the development of robust frameworks. Taking the final project of CS561 as an opportunity, we in this report and this project will try exploring the domain of music recommendation systems, the inherent nature of the problem and other interesting insights that we made up along the journey through this project.

II. RECOMMENDATION SYSTEMS

A Recommendation system is a subclass of information filtering system that provide suggestions for items that are most pertinent to a particular user. Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, or what online news to read. Recommendation systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer [1]. Examples include Netflix for movie recommendations , Amazon for product recommendation , Spotify for music . Other types of recommendations might include news recommendation , Advertisement recommendation etc.

A. Problem Statement

Let us try to really understand what is the problem that we have in our hand , so that we can use the insights generated to come up with interesting ways to solve the problems. We are given a input working system on top of which recommendation systems are supposed to work to make relevant recommendations. Given such an input system the idea would be to look into the state of the system , try to gain some insight on the question of what is relevant to who and what factors decide relevancy , once we can objectify the notion of relevancy , the next question to answer would be the notion so similarity , this idea of similarity can be between users , between items and so on. Once we decide on what we mean by relevancy to a particular user and what do we mean by similarity between two entities , we can go on to defining the problem as recommending the user "k" highest elements by relevancy by looking at feature set of similar entities , with this basic idea we can try to come up with a model for this problem of recommendation.

B. Philosophies of Similarity

Constructing a similarity metric first requires us to identify what we mean by similarity , Recommendation systems comes in two major flavors depending on the philosophy of similarity they follow. Models that follow the philosophy that if two users (x and y) are similar then there is an implicit implication that the x would like whatever y liked , this class of recommendation systems are called collaborative filtering systems. The second major flavor involves asking the question of is this item similar to whatever user x likes , in this sense instead of looking into other similar users and their liking's this model only looks into content similarity and how relevant it would be for user x say. This model is called as Content based Systems.

Now, how would we go about knowing what the user likes at all. This again has two answers to it. One , lets try to extract relevance and liking's from the metadata generated by the user. Two , lets just keep it simple and ask the user itself. The former is called implicit rating based systems , while the latter is called explicit rating systems. with these clear we can try to come up with a mathematical model for the system. We in this project would look into a collaborative filtering system that would use an implicit rating system.

III. Modelling the Alternating Squares from scratch

A. Utility Matrix

Lets say that the input system consists of a $n \times m$ matrix (U) that has n users and m items in the whole system , this matrix would have an entry U_{ij} as the rating that the user i gave for the item j. Given this base and the fact that we are developing a collaborative model the problem now boils down to solving for the empty entries in the matrix U , if we can fill in the empty entries of matrix U then we would objectively know how much a user likes a item and can make a recommendation with relevance based on it.

$$\begin{bmatrix} Header & Item\ 1 & Item\ 2 & Item\ 3 & ... & Item\ m \\ User\ 1 & U_{11} & U_{12} & U_{13} & ... & U_{1m} \\ User\ 2 & U_{21} & U_{22} & U_{23} & ... & U_{2m} \\ ... & ... & ... & ... & ... & ... \\ User\ n & U_{n1} & U_{n2} & U_{n3} & ... & U_{nm} \end{bmatrix}$$

Fig. 1. Utility matrix

B. Objective of the algorithm

Given that we have formally defined our Utility matrix the objective of a recommendation system algorithm would be the fill in the matrix entries and recommend to the user the Items whose entries correspond to the k greatest values. Now comes the collaborative filtering part , since we have decided to write a collaborative system , the process that we follow to fill in the matrix entries would involve us looking into the matrix entries of similar users and trying to come up with a relevant value to fill in for the missing value.

C. The Heuristic

With a clear cut understanding of what to find out , now comes the interesting part of how we are going to do it. how are we going to fill in the matrix with relevant values. To answer this we would need some help from the world of mathematics and linear algebra , There are 100s of Heuristic's one can come up with , this particular heuristic called as the matrix factorisation method is very simple to both implement and understand and hence we will try to model our problem based on the matrix factorisation method.

Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. One matrix can be seen as the user matrix where rows represent users and columns are latent factors. The other matrix is the item matrix where rows are latent factors and columns represent items.Very much like the concept of components in PCA, the number of latent factors determines the amount of abstract information that we want to store in a lower dimension space. A matrix factorization with one latent factor is equivalent to a most popular or top popular recommender (e.g. recommends the items with the most interactions without any personalization). Increasing the number of latent factors will improve personalization, until the number of factors becomes too high, at which point the model starts to overfit. A common strategy to avoid overfitting is to add regularization terms to the objective function. Lets say the Original Rating matrix is denoted by R and the predicted rating matrix is given by R' , lets say the matrix R was factorised into two matrices U and I where U is the user latent matrix and I is the item latent matrix. Thus the objective of this matrix factorisation algorithm would be to minimise the error between R and R' , formally this can be put up as :

$$L = \underset{U,I}{\arg\min} ||R - R'|| + \alpha||U|| + \beta||I||$$

D. Alternating least squares and spark

The Spark environment comes bundled with the alternating least squares implementation of the matrix factorisation problem. Alternating Least Square (ALS) is also a matrix factorization algorithm and it runs itself in a parallel fashion. ALS is implemented in Apache Spark ML and built for a larges-scale collaborative filtering problems. ALS is doing a pretty good job at solving scalability and sparseness of the Ratings data, and it's simple and scales well to very large datasets. It uses the L2 regularization objective function rather than L1 like in SVD , ALS minimizes two loss functions alternatively; It first holds user matrix fixed and runs gradient descent with item matrix; then it holds item matrix fixed and runs gradient descent with user matrix. ALS runs its gradient descent in parallel across multiple partitions of the underlying training data from a cluster of machines.

Some important hyperparameters of the module that spark allows us to tweak includes : Max iterations , Number of latent factors and a regularization parameter.Like any other model we can have a cross validator running through various possible combinations to find out the best performing set of hyper parameters.

Algorithm 1 ALS for Matrix Prediction

```
$Y \leftarrow Y_0$                                   ▷ Initialisation , I is Y
$X \leftarrow X_0$                                   ▷ Initialisation , U is X
repeat

    for $u = 1 \ldots n$ do
        $x_u \leftarrow (\sum y_i y_i^T + \lambda I_k)^{-1} \sum r_{ui} y_i$
    end for

    for $i = 1 \ldots m$ do
        $y_i \leftarrow (\sum x_u x_u^T + \lambda I_k)^{-1} \sum r_{ui} x_u$
    end for

until L converges
```

With this now we have completed all pieces of puzzle required to move on to our project , we will be using the ALS algorithm that comes with spark to build a music recommendation system based on the million songs data set.

IV. Project Music Recommendation system

Our Idea in this project is to build a basic music recommendation system with the help of the MLlib libraries that come as part of the spark installation. We will be using the million song dataset for our project. The Million Song Dataset is a freely-available collection of audio features and metadata for a million contemporary popular music tracks.The core of the dataset is the feature analysis and metadata for one million songs, provided by The Echo Nest. The dataset does not include any audio, only the derived features[2]. The Taste Profile dataset contains real user - play counts from undisclosed partners, all songs already matched to the MSD. We will be using this dataset as part of our project.

We will be using the play counts as an implicit metric of liking for a user , It is assumed that if a user has listened a song 10 times more compared to other songs , he likes that song more than other songs . with this assumption of relevance and likability we can move on to our next question on what will be the choice of our , well the answer is simple , we will run a cross validator and let it choose the best possible hyperparameter combination , the results found are included as part of the test jupyter notebook in the code files. With the hyperparameters and rating matrix set all we need to do is run Algorithm 1 on the input matrix and try to fill the empty cells , once done we will return the top k rated songs as recommendation for the user.

This simple model was implemented and is attached as part of the code files submitted in the zip. Now that we have seen a basic model of our system let us take a look into interesting issues and extensions that we faced when developing the model.

A. The blunder of Normalisation

Initially when writing the code the results obtained were not appreciable (RMSE) , later it turned out that we did the blunder of not normalising the data leading the model to have huge bias towards extremely popular songs with large play counts in comparison to new songs. This meant our model had a weakness to exploit , this was easily solved by normalising the play count metric , so that we can be sure large play counts don't introduce bias in the system.

B. Extending it to Artist recommendation

Once we were done with the song recommendation part , we wanted to extend the idea to artist recommendation , this was achieved by including one extra dataset that had song id to artist id mapping , combining it with the song metadata dataset and the original playcount dataset and using some of pysparks SQL libraries we were able to do some joins and group by to end up with the artist playcount dataset , with this in we went on to do the same procedure as in the songs dataset to develop a parallel artist recommendation system.

C. The Application Model

The developed application is a pretty simple python application that uses a pysimplegui frontend and pyspark as the backend (to perform some dataset joins and lookups) , when run the GUI allows the user to pick between a artist recommendation system and a song recommendation system , and once chosen will show the user the models recommendation and the actual users data (from the test set) , so as to give the user a feel on how relevant and well the model works. The shared zip folder contains the screenshots of a running application.

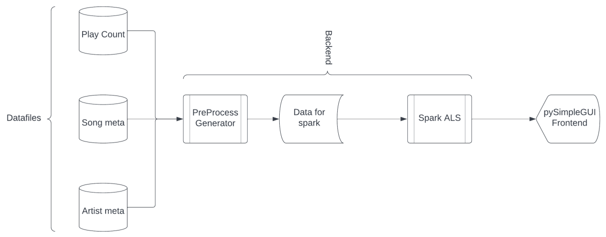

Fig. 2. Architecture of the Application

D. The War with flink

As part of the projects final weeks , we tried out implementing the same model in flink . While initial findings in the documentation about ALS was fruitful , it was later found out that a few months back the foundation decided to remove the Machine learning library of flink to a separate repository that will not be included in the normal installation , still with hope we tried our best by cloning that repository and trying to recompile the library locally , after couple of weeks of trial we were unsuccessful in running the model as there were lot of version incompatibility across the layers with un-resolvable dependency issues. While this might be a short coming , this allowed us to explore the package , the process of manually compiling it , chaining the source to make sure the dependencies trial out etc . Nevertheless the process did have a lot of things to takeaway.

V. Conclusion

The project in its due process did have a lot of technical and personal takeaways , lot of interesting ideas and procedures did pop up in my mind. Some of the interesting questions were on how to improve the rating feature set , we used a very simple play count as the metric , how can we enhance it , can we include regional and seasonal trends to get better outcomes , how can we include heterogeneous recommendation systems and how can they be optimally intertwined , what about a podcast recommendation system and so on. A lot of these questions led me to lot of interesting literature on current recommendation system trends , adding to the learning outcome of the project.Exploring various applications of big data techniques on different domains of musicology has put forth abundant opportunities to explore , but its also important for us to address questions that are not frequently raised, Its important that we don't shy away leaving any stakeholders out of discussion as they constitute a major part in the system. While Big data continues to be a hot topic of discussion , its important that we build systems that are simple and robust so that the fruits of a technology like Big data is accessible to everyone irrespective of their technical domain of expertise. Humanities, Arts and Literature when combined with Big data can give us deep insights on Human culture and emotional states of a society and its evolution with time , space and various economic stages.With the emergence of Big Data, there will be opportunities for researchers to generate 'knowledge of knowledge's sake' and to publish flashy findings that will undoubtedly attract media attention. But we urge researchers to concentrate their efforts on conducting research that will have real-world implications that will ultimately benefit individuals and society.

REFERENCES

[1] Ricci, F., Rokach, L., Shapira, B. (2022). Recommender Systems: Techniques, Applications, and Challenges. In: Ricci, F., Rokach, L., Shapira, B. (eds) Recommender Systems Handbook. Springer, New York, NY. https://doi.org/10.1007/978-1-0716-2197-4_1.

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[3] Hodgson, Thomas. (2021). Spotify and the democratisation of music. Popular Music. 40. 1-17. 10.1017/S0261143021000064.

[4] Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S. (2007). Collaborative Filtering Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds) The Adaptive Web. Lecture Notes in Computer Science, vol 4321. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-72079-9_9.

[5] Chen, J., Ying, P. & Zou, M. Improving music recommendation by incorporating social influence. Multimed Tools Appl 78, 2667–2687 (2019). https://doi.org/10.1007/s11042-018-5745-7