# Music Recommendation System
## INT248: Advance Machine Learning

Shivam Batholiya
11912024
Dept. of CSE Lovely Professional University, Phagwara, Punjab

Guide: Prof. Niharika Thakur
3rd November 2022

GitHub Project Link https://github.com/shivambatholiya/Music_Recommendation_System

## Abstract

In this project, we have designed, implemented and analysed a song recommendation system. We used Million Song Dataset provided by Kaggle to find correlations between users and songs and to learn from the previous listening history of users to provide recommendations for songs which users would prefer to listen most. In this paper, we will discuss the problems we faced, methods we have implemented, results and their analysis. We have got best results for memory based collaborative filtering algorithm. We believe that content-based model would have worked better if we would have enough memory and computational power to use the whole available metadata and training dataset.

Keywords- recommendation systems, music, Million Song Dataset, collaborative filtering, content-based

## Introduction

Rapid development of mobile devices and internet has made possible for us to access different music resources freely. The number of songs available exceeds the listening capacity of single individual. People sometimes feel difficult to choose from millions of songs. Moreover, music service providers need an efficient way to manage songs and help their costumers to discover music by giving quality recommendation. Thus, there is a strong need of a good recommendation system. Currently, there are many music streaming services, like Pandora, Spotify, etc. which are working on building high-precision commercial music recommendation systems. These companies generate revenue by helping their customers discover relevant music and charging them for the quality of their recommendation service. Thus, there is a strong thriving market for good music recommendation systems.

Music recommender system is a system which learns from the users past listening history and recommends them songs which they would probably like to hear in future. We have implemented

various algorithms to try to build an effective recommender system. We firstly implemented popularity-based model which was quite simple and intuitive. Collaborative filtering algorithms which predict (filtering) taste of a user by collecting preferences and tastes from many other users (collaborating) is also implemented. We have also done experiments on content-based models, based on latent factors and metadata.

# Dataset

We used data provided by Million Song Data Challenge hosted by Kaggle. It was released by Columbia University Laboratory for the Recognition and Organization of Speech and Audio. The data is open; meta-data, audio content analysis, etc. are available for all the songs. It is also very large and contains around 48 million (user id, song id, play count) triplets collected from histories of over one million users and metadata (280 GB) of millions of songs [7]. But the users are anonymous here and thus information about their demography and timestamps of listening events is not available. The feedback is implicit as play-count is given instead of explicit ratings. The contest was to predict one half of the listening histories of 11,000 users by training their other half and full listening history of other one million users.

Since, processing of such a large dataset is highly memory and CPU-intensive, we used validation set as our main data. It consists of 10,000,000 triplets of 10000 users. We used metadata of only 10,000 songs (around 3GB). From the huge amount of song metadata, we focus only on features that seem to be most relevant in characterizing a song. We decided that information like year, duration, hotness, danceability, etc. may distinguish a song most from other songs. To increase processing speed, we converted user and song ids from strings to integer numbers.

# Algorithms

We have implemented four different algorithms to build an efficient recommendation system.

## 1.1 Popularity based Model

It is the most basic and simple algorithm. We find the popularity of each song by looking into the training set and calculating the number of users who had listened to this song. Songs are then sorted in the descending order of their popularity. For each user, we recommend topmost popular songs except those already in his profile. This method involves no personalization, and some songs may never be listened in future.

## 1.2 Collaborative based Model

Collaborative filtering involves collecting information from many users and then making predictions based on some similarity measures between users and between items. This can be classified into user-based and item-based models.

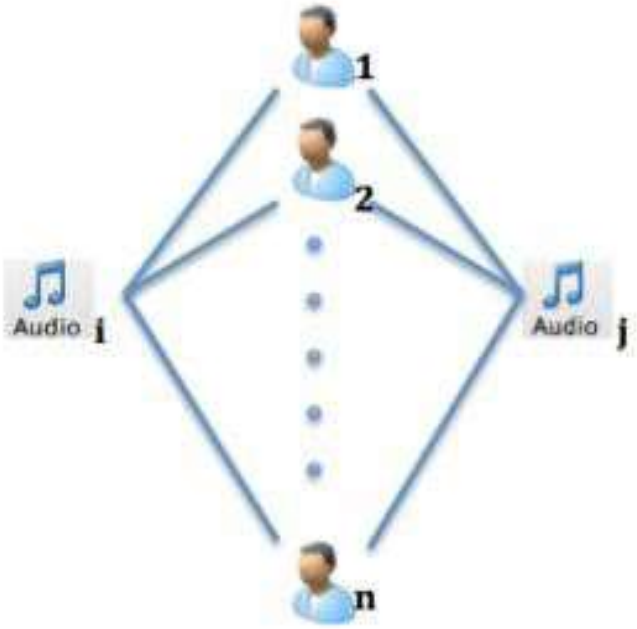


Figure 1: Item Based

In item-based model [Figure1][8], it is assumed that songs that are often listened together by some users tend to be similar and are more likely to be listened together in future also by some other user.

According to user-based similarity model [Figure2][8], users who have similar listening histories, i.e., have listened to the same songs in the past tend to have similar interests and will probably listen to the same songs in future too.

We need some similarity measure to compare between two songs or between two users. Cosine similarity weighs each of the users equally which is usually not the case. User should be weighted less if he has shown interests to many varieties of items (it shows that either she does not discern

between songs based on their quality, or just likes to explore). Likewise, user is weighted more if listens to very limited set of songs. The similarity measure,

$w_{ij} = P(\frac{i}{j})$, also has drawbacks that some songs which are listened more by users have higher similarity values not because they are similar and listened together but because they are more popular.

We have used conditional probability-based model of similarity [2] between users and between items:

$$W_{u,v} = P(v/u)^{\alpha} P(v/u)^{1-\alpha}, \alpha \epsilon (0,1)$$

$$W_{uv} = \frac{|I(u) \bigcap I(v)|}{|I(u)|^{\alpha} |I(v)|^{1-\alpha}} [2]$$

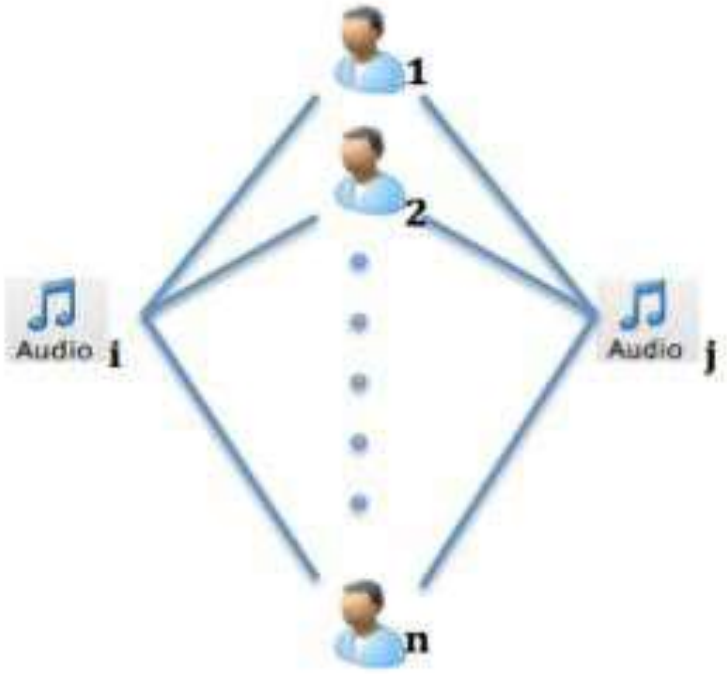


Figure 2: User Based

Different values of $\alpha$ were tested to finally come with a good similarity measure.

Then, for each new user u and song i, user-based scoring function is calculated as

$$h_{u_i}^{U} = \sum_{v \epsilon u_i} f(w_{uv})[2]$$

Similarly, item-based scoring function is

$$h_{u_i}^{I} = \sum_{j \epsilon I_i} f(w_{ij})[2]$$

Locality of scoring function is also necessary to emphasize items that are more similar. We have used exponential function to determine locality.

f(w) = $w^q, q \epsilon N[2]$

This determines how individual scoring components affects the overall scoring between two items. The similar things are emphasized more while less similar one's contribution drop down to zero.

After computing user-based and item-based lists, we used stochastic aggregation to combine them. This is done by randomly choosing one of them according to their probability distribution and then recommending top scored items from them. When the song history of a user is too small to utilize the user-based recommendation algorithm, we can offer recommendations based on song similarity, which yields better results when number of songs is smaller than that of users.

| | item1 | item2 | item3 | item4 | item5 | item6 |
|---|---|---|---|---|---|---|
| user1 | | | | | | |
| user2 | | | | | | |
| user3 | | | | | | |
| user4 | | | | | | |

Figure 3: Matrix M

We got the best results for stochastic aggregation of item-based model with values of q and $\alpha$ as 3 and 0.15, and of user-based model with values 0.3 and 5 for $\alpha$ and q respectively, giving overall mAP 0.08212. (See details about mAP in the Sec 3.5)

This method does not include any personalization. Moreover, majority of songs have too few listeners, so they are least likely to be recommended. But still, we as well as Kaggle winner got best results from this method among many others. Here, we have not used play count information

in result as they did not give good result because similarity model is biased to a few songs played multiple times and calculation noise was generated by a few very popular songs.

## 1.3 SVD Model

Listening histories are influenced by a set of factors specific to the domain (e.g., genre, artist). These factors are in general not at all obvious and we need to infer those so-called latent factors [4] from the data. Users and songs are characterized by latent factors.

Here, to handle such a large amount of data, we build a sparse matrix [6] from user-song triplets and directly operate on the matrix [Figure 3], instead of looping over millions of songs and users. We used truncated SVD for dimensionality reduction.

We used SVD algorithm in this model as follows:

Firstly, we decompose Matrix M into latent feature space that relates user to songs.

$M = U^{P}V$, where

$$M \epsilon R^{m*n}, U^{m*k}, \sum^{k*k} \text{ and } V\, k*n$$

Here, U represents user factors and V represents item factors. Then, for each user, a personalized recommendation is given by ranking following item for each song as follows-

$$W_i = U_u^T . V_i$$

Though the theory behind SVD is quite compelling, there is not enough data for the algorithm to arrive at a good prediction. The median number of songs in a user's play count history is fourteen to fifteen; this sparseness does not allow the SVD objective function to converge to a global optimum.

## 1.4 KNN Model

In this method, we utilize the available metadata. We create a space of songs according to their features from metadata and find out neighbourhood of each song. We choose some of the available features (e.g., loudness, genre, mode, etc.) which we found most relevant to distinguish a song from others. After creating the feature space, to recommend songs to the users, we look at each user's profile and suggest songs which are neighbours to the songs present in his listening history. We have taken top 50 neighbours of each song. This model is quite personalized and uses metadata. But since, we had 280GB file of metadata which takes huge amount of time in processing, we extracted features of only 3GB (10,000 songs), which is less than 2 % of total number. Due to this, we had features of only small number of songs, which gives us very small precision.

## 1.5 Evaluation Metrics

We used mean Average Precision (mAP) as our evaluation metric. The reason behind using this is that this metric was used in the Kaggle challenge which helps us to compare our results with others. Moreover, precision is much more important than recall because false positives can lead to a poor user experience. Our metric gives the average of the proportion of correct recommendations giving more weight to the top recommendations. There are three steps of computing mAP as follows: Firstly, precision at each k is calculated. It gives the proportion of correct recommendations within the top-k of the predicted rankings.

$P_k(u,r) = \frac{1}{k}\sum_{j=1}^{k} M(u,r(j))$

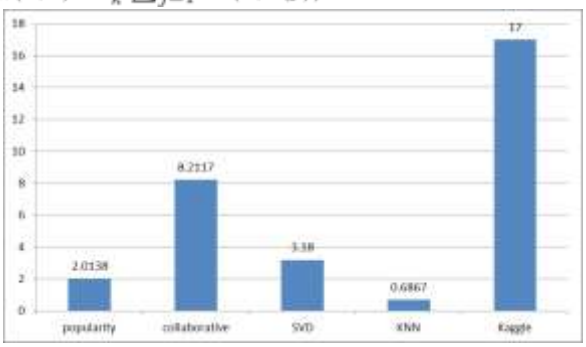


Figure 4: Results

Then, for each user, average precision at each k is evaluated.

AP$(u,r) = \frac{1}{n_u}\sum_{k=1}^{t} P_k(u,r).M(u,r(k))$

Finally, mean of all the users is taken. mAP =$\frac{1}{m}\sum_{u=1}^{m} AP(u,r_u)$

# Results

We got best results for memory based collaborative filtering algorithm. Our SVD based latent factor model gives better results than popularity-based model. It lags collaborative filtering algorithm because the matrix was too sparse which prevented objective functions to converge to global optimum. Our K-NN model did not work well and performs worse than even the popularity model. The reason behind this is that we have the features of only 10,000 songs, which is less than 3 % of the whole dataset so only some of these 10,000 songs could be recommended. The huge lack of information leads to the bad performance of this method.

# Conclusion

This is a project of our Advance Machine Learning course. We find it is very good as we got a chance to practice theories that we have learnt in the course, to do some implementation and to try to get a better understanding of a real Machine Learning problem: Music Recommendation System. There are many different approaches to this problem, and we get to know some algorithms in detail and especially the four models that we have explained in the paper. By manipulating the dataset, changing the learning set and testing set, changing some parameters of the problem and analysing the result, we earn a lot of practicing skills. We have faced a lot of problem in dealing with this huge dataset, how to explore it in a better way and we also had difficulties in some programming details. However, with lot of efforts, we have overcome all of these.

The best part of this project is the teamwork. Both of us come from different cities and thus have different cultures and ways of working. We took a bit of time to get to know each other, to adjust ourselves and to perform like a team. We become much more efficient by the time the team spirit is formed and we also enjoy more. We both find this project a nice experience and all the effort put is worthy. We have learnt a lot from this project.

In terms of research, we still have a lot to do to make our studies a better one. Music Recommendation System is such a wide, open and complicated subject that we can take some initiatives and do a lot more tests in future.

We also got to realize that building a recommendation system is not a trivial task. The fact that its large-scale dataset makes it difficult in many aspects. Firstly, recommending 500 correct songs out of 380 million for different users is not an easy task to get a high precision. That's why we didn't get any result better than 10 %. Even the Kaggle winner has only got 17 %. Secondly, the metadata includes huge information and when exploring it, it is difficult to extract relevant features for song. Thirdly, technically speaking, processing such a huge dataset is memory and CPU intensive.

All these difficulties due to the data and to the system itself make it more challenging and also more attractive. We hope that we will get other opportunities in the future to work in the domain of Machine Learning. We are certain that we can do a better job.

## Future work

- Run the algorithms on a distributed system, like Hadoop or Condor, to parallelize the computation, decrease the runtime and leverage distributed memory to run the complete MSD.
- Combine different methods and learn the weightage for each method according to the dataset
- Automatically generate relevant features

- Develop more recommendation algorithms based on different data (e.g., the how the user is feeling, social recommendation, etc)

## Acknowledgements

We would like to acknowledge the efforts of Prof. Niharika Thakur as without her constant support and guidance this project would not have been possible.

## References

[1] McFee, B., Bertin Mahieux, T., Ellis, D. P., Lanckriet, G. R. (2012, April). The million-song dataset challenge. In Proceedings of the 21st international conference companion on World Wide Web (pp. 909916).ACM.

[2] Aiolli, F. (2012). A preliminary study on a recommender system for the million songs dataset challenge. PREFERENCE LEARNING: PROBLEMS AND APPLICATIONS IN AI

[3] Koren, Yehuda. ” Recommender system utilizing collaborative filtering combining explicit and implicit feedback with both neighborhood and latent factor models.”

[4] Cremonesi, Paolo, Yehuda Koren, and Roberto Turrin. ” Performance of recommender algorithms on topn recommendation tasks.” Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010

[5] T. Bertin et al., The Million Song Dataset, Proc. of the 12th International Society for Music Information Retrieval Conference, 2011

[6] Sparse Matrices http://docs.scipy.org/doc/scipy/reference/sparse.html

[7] Mahiux Ellis 2-11 http://labrosa.ee.columbia.edu/millionsong/tasteprofile

[8] http://www.bridgewell.com/images _en