Music Artist Recommender using Collaborative Filtering

Moksha Shah
Machine Learning (CS 667)
California State University, East Bay
Hayward, United States
mshah53@horizon.csueatbay.edu

Abstract— *Recently, the building of recommender systems becomes a significant research area that attractive several scientists and researchers across the world. The recommender systems are used in a variety of areas including music, books, news, search queries, and commercial products. Collaborative Filtering algorithm is one of the popular successful techniques of RS, which aims to find users closely similar to the active one in order to recommend items. Collaborative filtering (CF) with alternating least squares (ALS) algorithm is the most imperative techniques which are used for building a music recommendation engine. The ALS algorithm is one of the models of matrix factorization related CF which is considered as the values in the item list of user matrix. As there is a need to perform analysis on the ALS algorithm by selecting different parameters which can eventually help in building efficient music recommender engine. In this paper, we propose a music recommender system based on ALS using Implicit. This research focuses on the selection of parameters of ALS algorithms that can affect the performance of a building robust RS. From the results, a conclusion is drawn according to the selection of parameters of ALS algorithms which can affect the performance of building of a music recommender engine. The model evaluation is done using different metrics such as execution time, root mean squared error (RMSE) of rating prediction, and rank in which the best model was trained. The two best cases are chosen based on best parameters selection from experimental results which can lead to building good prediction rating for a music recommender.*

Keywords— *Recommender systems, Collaborative filtering, Alternating Least Squares, Big data LastFM dataset*

I. Introduction (*Heading 1*)

Big data analytics become an important trend for organizations and enterprises that are interesting in providing innovative ideas for enhancing and increasing their business performance and decision-making. Recommender systems are a group of techniques that allow filtering through large samples and information space in order to give suggestion to users when needed. Currently, they are becoming highly popular and utilized in different areas such as music, research articles, search queries, news, books, social tags, and music. Furthermore, there are other essential recommender systems basically applicable for specialist, collaborators, funny story, restaurant and hotels, dresses, monetary services, life insurance, passion associates which give online dating services and several other social media such as Twitter, LinkedIn, and Facebook.

The focus of this work is collaborative filtering system. It is well known that collaborative filtering could be described as a procedure whereby automatic prediction (i.e.filtering) about the interests of a user is made by gathering taste or preferences information from many users. The unexpressed assumption of the collaborative filtering approach can be best explained, viz., supposing a person A has similar opinion with person B on a particular issue, the assumption is that person A will be more likely to
have the same opinion as person B on a different issue X did the opinion on X of a person chosen randomly [1]. The main tools that we have used here are Implicit where we have implemented recommender system using Implicit and using Matrix Factorization analysis of our dataset is done.

The minimization of the error for the users/music pairs was chosen as the basis for the selection of the two matrices. The alternating least squares algorithm (ALS) which achieves this by randomly filling the user's matrix with values before optimizing the value of the music was used for this purpose. The value of the user's matrix is optimized with the music's matrix being kept constant (Fig. 1). Owing to a fixed set of user factors (i.e., values in the user's matrix), known ratings are employed to find the best values by optimizing the music factors, written on top of the figure. The best user factor with the fixed music factors is sleeted. This paper reports for the first time a music recommendation system based on collaborative filtering using Implicit. The performance analysis and evaluation of the proposed approach are performed on a LastFM dataset.

II. Dataset

Last.fm generates weekly "global" charts of the top 400 artists and tracks listened to by all Last.fm users.

The results differ significantly from traditional commercial music charts provided by the UK Top 40, Billboard, Soundscan, and others, which are based on radio plays or sales. Last.fm charts are less volatile, and a new album's release may continue to be reflected in play data for many months or even years after it drops out of commercial charts.

The LastFM dataset contains music listening history, where each record represents a user's interaction with a song. It includes features like:

A. ***artists.dat:*** *Contains information about artists. Each entry likely includes n artist ID and their name.*

- Format: artistID, artistName

B. ***tags.dat:*** *Contains information about tags. Each entry might include a tag ID, and the tag's name.*

- Format: tagID, tagName

C. ***user_artists.dat****: Contains information about tags. Each enttry likely consists of a user ID, artist ID, and the number of plays or interactions between the user and the artist.*

- Format: userID, artistID, weight (PlayCount)

D. ***user_friends.dat:*** *Contains information about the relationships between users. Each entry liekly indicates a friendship or connection between two users.*

- Format: userID, friendID

E. ***user_taggedartists.dat:*** *Contains data about users tagging artists with specific tags. Each entry might include the user ID, artist ID, and the associated tag ID.*

- Format: userID, artist ID, tagID

F. ***user_taggedartists-timestamps.dat:*** *Similar to user_tagedartists.dat, but eith thr addition of timestamps indicating when the tags were applied.*

- Format: userID, artistID, tagID, timestamp

III. Data preprocesing

The preprocessing pipeline for the lastFM dataset involves several key steps to convert the raw data into a suitable format for collaborative filtering using the Alternating Least Squares (ALS) algorithm. Since the dataset is sparse, the Compressed Sparse Row (CSR) matrix format is chosen to store and process the user-item interaction matrix efficiently. Below is a detailed explanation of each step.

A. *Loading the data:*

The first step involves loading the user-artists interaction data from the file user_artists.dat, which contains interactions in the form of userID, artistID, and weight-representing the number of interactions or plays- We use Pandas to read the data and store it as a DataFrame for easier manipulation.

B. *Creating the User_item interaction Matrix:*

Next, we create the user-item interaction matrix, where the rows represent users, the columns represent artists, and the values represent the interaction count. We set the index userID and artistID.

C. *Converting to CSR format:*

In this step, we convert the user-item interaction data into a Compressed Sparse Row (CSR) matrix. The CSR format is efficient for storing and manipulating sparse data, which is critical for matrix factorization tasks like ALS.

- The COO format created using scipy.sparse.coo_matrix is a simple format that stores non-zero values along with their row and column indices.

- The CSR format (Compressed Sparse Row) is more efficient for row slicing and matrix multiplication, making it ideal for large-scale collaborative filtering tasks.

The conversion process is embedded in the load_user_artists() function, where we read the user_artists.dat file, construct the interaction matrix, and convert it to CSR format.

IV. Analysis

The core analysis of the project revolves around how collaborative filtering and matrix factorization techniques can predict the missing values in the user-item matrix. Collaborative filtering leverages the principle that users who interacted with similar items in the past will likely have similar preferences in the future. ALS matrix factorization is used to extract latent factors that explain the observed interactions between users and items.

V. Recommendation

Once the ALS model is trained, the system can recommend artists to users based on the predicted interactions. Recommendations are generated by identifying the artist that a user is most likely to interact with, based on their historical preferences and the interactions of similar users. The recommendation process can be broken down into the following steps:

A. ***User-Item Interaction Matrix:*** *We first create a matrix where the rows represent users and the columns represent artistss, with the values representing interactions (e.g., play counts/ weights)*

B. ***Matrix Factorization:*** *ALS decomposes this matrix into lower-dimensional matrices representing latent factors for both users and items.*

C. ***Prediction:*** *We then predict missing entries in the matrix, which correspond to user-item interactions that have not yet occurred. These predictions are used to recommend tracks to the users.*

Recommendations typically speed up searches and make it easier for users to access content they're interested in, and surprise them with offers they would have never searched for. By using the music recommender system, the music provider can predict and then offer the appropriate artists to their users based on the characteristics of the music that has been heard previously.

VI. Collaborative Filtering

Collaborative filtering is a technique that recommends items by analyzing past behaviors of users. The key idea behind Collaborative Filtering is that similar users share similar interests, people with similar interests tend to like similar items. Hence those items are recommended to similar set of users. For example, if a person A has the same opinion as a person B on an issue. Then A is more likely to have B's opinion on different issues. Thus, recommendations are made using the ALS matrix factorization method. In this system, collaborative filtering is implemented using user-item interactions to predict preferences. There are two types of collaborative filtering:

- User-based Collaborative Filtering: Recommends items by identifying similar users.
- Item-based Collaborative Filtering: Recommends items by finding similarities between items.

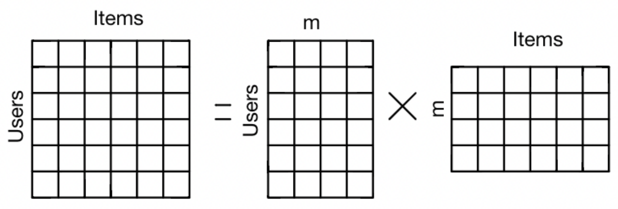
For this project, we focus on item-based collaborative filtering using ALS, which is particularly effective when the user-item matrix is sparse.

VII. Item-based collaborative filtering

In item-based collaborative filtering, we generate recommendations by analyzing the similarity between items rather than the users. The basic idea is that if a user likes a particular artist, they will likely enjoy other artists that are similar to it.
The similarity of items can be calculated using various metrices, such as cosine similarity or Pearson correlation. In ALS, this is achieved through matrix factorization, which captures latent relationships between items in a lower-dimensional space.

VIII. Matrix factorization methods

When a user gives feedback to a certain artist they heard of, this collection of feedback can be represented in the form of a matrix where each row represents each user, while each column represents different artists. The matrix will be sparse since not everyone is going to hear every artist. One strength of matrix factorization is the fact that it can incorporate implicit feedback information that is not directly given but can be derived by analyzing user behavior. Using this strength, we can estimate if a user is going to like an artist, he/she never heard to and if that estimated rating is high, we can recommend that artist to the user.



The above image does an excellent job of summarizing the core idea behind matrix factorization. Let there be matrix A with dimensionality of (m,,m) this matrix can be viewed as a dot product between two matrix with each matrices having dimensionality of (m,k) and (k,m).
The goal is to approximate the original matrix by multiplying these two matrices, thus predicting missing interactions.
Matrix factorization has several advantages:

- It reduces the complexity of large, sparse matrices.
- It captures latent features that may not be explicitly visible in the data.
- It is scalable to large datasets.

IX. Learning methods

The ALS algorithm is an iterative method for matrix factorization. It optimizes the approximation of the user-item interaction matrix by alternating between fixing the user and item latent factors and solving for the missing values. The process continues until convergence, typically based on a defined tolerance.

X. Alternating least squares (ALS)

Alternating Least Square (ALS) is also a matrix factorization algorithm and runs itself in a parallel fashion. ALS is implemented in Implicit ML and built for larger-scale collaborative filtering problems. ALS is doing a pretty good job at solving scalability and sparseness of the Ratings data, and its simple and scales well to very large datasets. Most important hyper-parameter in ALS:

- maxIter: the maximum number of iterations to run (defaults to 10)
- rank: the number of latent factors in the model (defaults to 10)
- regParam: the regularization parameter is ALS (defaults to 10)

Steps to be followed to build a recommender system: create dataframe of all distinct artists by active user. Joining both tables on left join. Selecting artists which active user is yet to listen. Adding a new column of userID of active user to remaining artists dataframe making recommendations using ALS recommender model and selecting only top 10 artists.

XI. Implicit Feedback and the implicit module

The implicit library in Python is specifically designed for collaborative filtering with implicit feedback, where user interactions such as clicks, views, plays are used instead of explicit ratings. Implicit feedback data is often more abundant and easier to obtain. The Alternating Least Squares algorithm in implicit module provides an efficient implementation for matrix factorization with implicit feedback datasets.
The LastFM dataset, which contains user listening history and music tracks, has been widely used to evaluate music recommender systems. Many state-of-the-art models have employed matrix factorization techniques to predict user preferences for unseen tracks or artists based on their past listening behavior.

XII. Original model

The original model music recommender system was developed using Apache Spark for collaborative filtering through Matrix Factorization. Apache Spark provides a scalable framework that is suitable for handling large datasets, which is essential when working with the LastFM dataset. The system primarily uses the Alternating Least Squares (ALS) algorithm for matrix factorization, implemented within Spark's MLlib.

A. Advantages of using Apache Spark for ALS:

- **Scalability:** Apache Spark is designed for large-scale data processing. The ALS algorithm in Spark is optimized for distributed computing, allowing it to scale to large datasets like the **LastFM dataset**.
- **Parallelization**: Spark automatically parallelizes operations across a distributed cluster, making it efficient even when the dataset is too large to fit into memory on a single machine.

B. *Challenged and Limitations:*

- **Cold-Start Problem**: The model struggles with the **cold-start problem**, where it cannot recommend items (artists) to users who have not interacted with many items, or for new items that have few interactions.
- **Parameter Tuning**: While Spark provides powerful tools for training collaborative filtering models, hyperparameter tuning (such as choosing the right number of latent factors, regularization, and iterations) is crucial for improving model performance and preventing overfitting.
- **Memory and Computation**: Although Spark scales well with large datasets, working with very large datasets still requires careful management of memory and computational resources, especially when using more complex models or large numbers of iterations.

XIII. Improvement Ideas

While the model provides reasonable recommendations, there are areas for potential improvement:

A. ***Hybrid Recommendation****:*

A hybrid model combining collaborative filtering and content-based filtering could be more effective. By incorporating item features (e.g., artist genres, tags, or popularity), the model could address issues like the cold-start problem (recommending items to new users or recommending new artists with little interaction history).

B. ***Incorporating Temporal Data:***

The model could be improved by including timestamp data (from the user_taggedartists-timestamps.dat file) to account for changes in user preferences over time. For example, users may prefer certain genres or artists at different points in their lives, and this temporal shift can be captured.

C. ***Personalization****:*

Implementing user-specific factors such as genre preferences or listening habits can help tailor recommendations more precisely, increasing user satisfaction.

D. ***Matrix Factorization Variants****:*

Experiment with other matrix factorization techniques like Singular Value Decomposition (SVD) or Neural Collaborative Filtering (NCF), which may provide more robust latent features and better performance compared to ALS.

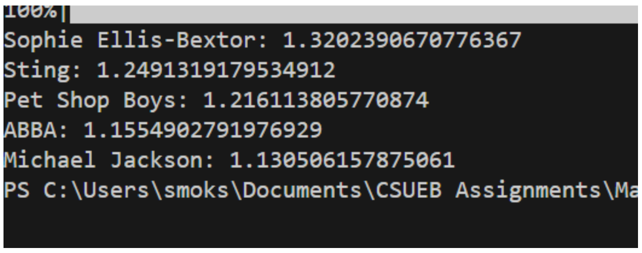
E. ***Handling Cold-Start Problem:***

To overcome the cold-start problem, the system could integrate demographic information (e.g., age, location) or content-based filtering (e.g., genre, artist bio) to provide better recommendations for new users or items.

XIV. Results and discussions

To evaluate the performance of the music recommender system, we focus on the top recommendations generated by the model using the Alternating Least Square (ALS) algorithm from the Implicit module. The model predicts user preferences for unheard artists based on their historical interaction with artists. These predictions are made by factorizing the user-item interaction matrix.

Below are the top 5 recommended artists for a specific user, along with their respective predicted scores, which indicate the strength of the recommendations. Higher scores suggest stronger relevance and higher likelihood that the user will interact with the recommended artists.



```
100%|
Sophie Ellis-Bextor: 1.3202390670776367
Sting: 1.2491319179534912
Pet Shop Boys: 1.216113805770874
ABBA: 1.1554902791976929
Michael Jackson: 1.130506157875061
PS C:\Users\smoks\Documents\CSUEB Assignments\Ma
```

Discussion of Performance:

- The ALS model, implemented through the Implicit module, effectively identifies artists that align well with the user's past listening behavior. The model captures latent preferences and provides diverse recommendations, including well-known artists like Sting and Michael Jackson, along with Sophie Ellis-Bextor and Pet Shop Boys, which may be more personalized based on the user's interaction history.
- These predicted scores can be interpreted as the likelihood that the user will interact with these artists in the future, making them highly valuable for personalized music recommendations.

XV. Future Work

A. *Scalability:*

- Further testing will be done on larger datasets to evaluate the scalability and performance of the recommender system.

B. *Deep Learning Approaches*

- Investigating neural collaborative filtering (NCF) and autoencoders for matrix factorization could lead to more advanced models that better capture latent features.

C. *Hybrid Recommender Systems*

- Future work will focus on integrating content-based filtering with collaborative filtering to improve recommendations, especially for new users and artists.

D. *Temporal Modeling*

- We plan to incorporate timestamp data into the model to capture temporal dynamics in user preferences.

XVI. Conclusions

The recommender system based on Alternating Least Squares (ALS) using the Implicit module demonstrates the effectiveness of collaborative filtering for music recommendations. The model performs well in predicting user preferences based on implicit feedback (play counts) and provides personalized recommendations for users in the LastFM dataset.

However, the system can be improved by addressing the cold-start problem, enhancing personalization, and exploring hybrid models. Further improvements could involve using timestamp data to account for temporal preferences and integrating content-based features to create a more robust recommendation engine.

References

[1] R. Zhang, Q. -d. Liu, Chun-Gui, J. -X. Wei and Huiyi-Ma, "Collaborative Filtering for Recommender Systems," 2014 Second International Conference on Advanced Cloud and Big Data, Huangshan, China, 2014, pp. 301-308, doi: 10.1109/CBD.2014.47.

[2] Simple Matrix Factorization example on the Millionsong dataset using Pyspark. (2018).Medium.Retrieved22November2018,from https://medium.com/@connectwithghosh/simple-matrix-factorization-example-on-themusiclens-dataset-using-pyspark9b7e3f567536\

[3] Yingying Zhuang, Yuezhang Chen, and Jie Zheng. Music genre classification with transformer classifier. In Proceedings of the 2020 4th international conference on digital signal processing, pages 155–159, 2020.