Yahoo Music Recommendation

Group – Music Recommender

Member - Amit Singh [10465317]

Project Report
Under the Guidance of
Prof. Rensheng Wang
By
Amit Singh

Stevens Institute of Technology

TABLE OF CONTENTS

1. Introduction

Nowadays, almost all Music applications are trying to recommend different music to different users based on their listening records. Recommendation systems play a vital role in ever-increasing consumer-facing databases because they offer a solution to the Paradox of Choice among users.

- Content-based filtering uses item features to recommend items like what the user likes, based on previous actions or explicit feedback.
- Collaborative filtering uses similarities between users and items simultaneously to provide recommendations. These models can recommend an item to user A based on the interests of a similar user. Neighborhood methods and Latent Factor Methods are two widely used collaborative-based filtering methods.

1.1 Motivation

As we know, the music data around us is huge. The number of songs available exceeds the listening capacity of a 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 customers discover music by giving quality recommendations.

1.2 Project Definition

The project aims to design and implement Yahoo Music Recommendation project aims to build out a recommendation engine that can predict three tracks that a user will like based on a wide range of features. It then processes the input data using various algorithms. Then the algorithm predicts the recommendations for songs. We tried to determine how only a few songs are selected and recommended from such extensive data. The project is implemented using the Python Language.

1.3 Scope of Work

The Tasks that we had in hand are:

- Understanding Data Set

Yahoo music data set is very large and contains many parameters like the Users, Tracks, Album, Artist, Genre, and Ratings. A specific ID denotes every parameter in the dataset.

- Data Processing
  The data we get online is usually not fit to use, and Yahoo Music data was unsorted, so we cannot use it directly. To solve this problem, we pre-process the data so that it can be used.

- Training Algorithm
  We pass the data we get after data processing through the training algorithm to train the algorithm and get the training output.

- Testing Dataset
  The testing dataset contains six tracks for each user to be tested.

- Observation
  After the data is tested, observations are made, and output is predicted.

2. Data Set Description

Yahoo Music Dataset consists of a hierarchy of data like user id, track id, album, artist, and genre.
Track -> Album -> Artist -> Genre

Yahoo Music Dataset consists of 120000 rows, having 20,000 user data with six tracks per user. We need to select three tracks labeled as "1" and recommend them to each user, and we label the remaining three songs as "0".

2.1 Training data

Training data contains the user id and items the user had already rated. The rating could be for a track, album, artist, or genre. This Training data will be used to train various algorithms and predict the user's songs.

- testTrack_hierarchy.txt – This file contain data in following format as:
  <UserID>|<trackID>|<Album>|<Artist>|<Genre 1>|…|<Genre n>

  For example, below row in the records is in following format:

199810|74139|277282|271146|113360|173467|173655|192976

199810 (user ID) | 74139 (track ID) |277282 (album ID) | 271146 (artist ID) | 113360 (genre 1) | 173467 (genre 2) | 173655 (genre 3) | 192976 (genre4)

- trainIdx2_matrix.txt – This file contain data in following format as:
  userID | itemID | score

  For example, the row in the records is in following format:
  199808|248969|90
  199808 (userID) | 248969 (itemID) | 90 (score)
- trainItem.data - the training set
- testItem.data- the test set

2.2 Testing Data

Based on the training, the algorithm was provided below the test data file, which includes the IDs of a song to predict. The algorithm will predict the ratings for the 6 tracks and sort them according to rating value. The 3 highest rating tracks will be recommended for the user and given 1 as a result, and the other 3 tracks will be given 0 value as a result.

submission.csv - submission file in the correct format:
TrackID | Predictor

2.3 Data Pre-Processing

Raw data which we get is not always beneficial for training. We need to process the given data and generate data for machine learning training.

We generated file in following below format. Where UserID and trackID are ID but Album, Artist, and Genre 1-21 are corresponding rating.
<UserID>|<trackID>|<Album>|<Artist>|<Genre 1>|…|<Genre 21>

3. Training Algorithm (Method Used)

The crucial part of any Machine Learning Project is to select the Algorithm for Data Processing & Training. We started with a fundamental approach to using only Album, Artist, and Genre information. We implemented one after one procedure from weights, matrix factorization, and various classifications to Ensemble Algorithms to use multiple algorithms to solve the given problem.

This section provides insights into the different algorithms used in the project. The programming was done in Python, using notebooks, and PySpark was used for extensive data processing. A brief introduction to the algorithms and their implementations is detailed below:

3.1 Method 1 – Using Album, Artist, Genre [1-21] Ratings

Album, Artist, and 21 Genre ratings were considered for this algorithm. Sorted score per users and set '1' to top 3 tracks and '0' to last 3 tracks.

- Summation of album, artist, and 21 genre rating
  Sum (Artist + Album + Genre1-21) -> 0.8433

- Maximum of album, artist, and 21 genre rating
  Max (Artist, Album, Genre1-21) -> 0.8121

- Average of album, artist, and 21 genre rating
  Mean (Artist, Album, Genre1-21) -> 0.8438

3.2 Method 2 – Using Album, Artist, Genre [1-21] Ratings with weights

Album, Artist, and 21 Genre ratings were considered for this algorithm and assigned different weights to the album, artist, and genre. Sorted score per users and set '1' to top 3 tracks and '0' to last 3 tracks.

- Weighted Average of Artist, Album, and Summation of Genre
  Mean (Artist * 0.4, Album * 0.3, Sum (Genre [1-21]) * 0.3) -> 0.86986 %

- Weighted Average of Artist, Album, and Summation of Genre

Mean (Artist * 0.7, Album * 0.2, Sum (Genre [1-21]) * 0.1) -> 0.87022 %

- Weighted Average of Artist, Album, and Average of Genre
  Mean (Artist * 0.8, Album * 0.15, Mean (Genre [1-21]) * 0.05) -> 0.84293 %

- Weighted Average of Artist, Album, and Summation of Genre
  Mean (Artist * 0.8, Album * 0.15, Sum (Genre [1-21]) * 0.05) -> 0.86938 %

- Weighted Sum of Artist, Album, and Average of Genre
  Sum (Artist * 0.8, Album * 0.15, Mean (Genre [1-21]) * 0.05) -> 0.87050

3.3 Method 3 – Matrix Factorization (Alternating Least Squares)

Matrix factorization algorithms are some of the most successful examples of latent factor models, which work by attempting to explain and predict ratings by characterizing both users and the items they are rating using numerous factors. If an item and user share a highly correlated factor, a recommendation may be made.

Matrix factorization attempts to estimate the rating matrix as the product of two lower-rank matrices: q_i for the items and p_U for the users. Any user's approximate rating of any item is then given as $\hat{r}_{ui} = q_i^T p_U$**.** Both the p and q matrices are unknown. ALS approaches the problem by alternating fixing one of the matrices; once each matrix is completely fixed, the other matrix is recomputed by solving a least-squares problem. The benefit is ALS is the potential for high parallelization. The elements of the matrices can be calculated independently of each other, and thus these calculations can be spread across large amounts of computing power.

Implementation:

- PySpark was used for parallelization in this implementation, and the ALS model built into PySpark was used.
- Parameter used: rank & maxIter = 20.
- After extraction, 26 entries were missing, because for some user id's only five tracks were present instead of six.
- 26 missing entries are assigned as '0'.
- This produced lowest score for given training data.

3.5 Method 4 – Various Classification Methods

- Album, Artist, Genre [1-21] Ratings are used
- Training data is used, which has correct prediction. Data contain 6000 entries for 1000 users.
- The following methods were used:
    - Logistic Regression
    - Decision Tree
    - Random Forest
    - Gradient Boosted Decision Tree

3.5.1 Logistic Regression:

It is a supervised machine learning algorithm. Logistic Regression is a statistical model used to perform a binary classification task using the log function.
Implementation:

- Used built-in Logistic Regression model in PySpark.
- Used Logistic Regression to predict '1' if the user likes the track and '0' otherwise.
- It produced a better result than the ALS model.

3.5.2 Decision Tree Classification:

Decision Trees are a non-parametric supervised learning method used for classification and Regression. The algorithm continuously splits the data according to a particular parameter. In a classification tree, the outcome is categorical.
Implementation:

- Used built-in Decision Trees Classifier in PySpark.
- If users liked a track or not, they were assigned a '1' and '0', respectively.

3.5.3 Random Forest Classification:

Random forest is a supervised machine learning algorithm used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in Regression. Random Forest Algorithm can handle the data set containing continuous variables as in the case of Regression and categorical variables as in the case of classification. It performs better results for classification problems. Random forest is the most accurate ensemble classifier and works efficiently on massive datasets. It can accurately predict the missing data, even when large portions of data are missing without preprocessing. It combines bagging

and random feature selection. Here, the goal is to predict if a user will like a track '1' or not '0'. Let's say there are ten decision trees in a random forest. If 8 of those trees predict that the user will like the track and 2 of those trees predict that the user will not like the track, then the prediction will be a 1 (likes the track).

Implementation:

- For this project, I used a built-in Random Forest Classifier in PySpark.
- This produced a better result than the Decision Trees model.

3.5.4 Gradient Boosted Tree Classification:

Gradient Boosting machine learning algorithm. Gradient boosting works by building weak prediction models sequentially where each model tries to predict the error left over by the previous model. Each predictor corrects its predecessor's error, and the weights of the training instances are not tweaked. Instead, each predictor is trained using the residual errors of its predecessor as labels.
Implementation:

- For this project, I used a built-in Gradient Boosting Classifier in PySpark.
- Predict '1' if the user likes the track and '0' otherwise.
- CrossValidator with GBTClassifier used for better performance.

3.6 Method 5 – Ensemble

The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions. In learning models, noise, variance, and bias are significant sources of error. The ensemble methods in machine learning help minimize these error-causing factors, ensuring the accuracy and stability of machine learning algorithms. We have seen a significant improvement in the score using Ensemble Learning. It calculates a vector of weights for each solution vector. Least Squares Solution $a_{LS} = (S^TS)^{-1}S^T$ x. S is the matrix of solution vectors S^T x is the accuracy of each model needed to calculate the weights.

This method is helpful because various algorithms have pros and cons. Therefore, a technique can sometimes have corner cases that it cannot predict correctly. In this scenario, if we ensemble different outputs, one method can eliminate the corner cases of other methods and help each other improve the result in the music recommender system Ensemble algorithm given best accuracy. Ensemble produced the best score compared to the above methods.

4. Result

The below table provides the information on prediction results and the score achieved

Observation

- I tried different weighted methods. It shows that the Artist rating has more weightage than Album, and Album weightage than Genre. Score changes according to the weight assigned to the feature.
- Also, I tried different classification algorithms. Random forest produced a better result in the classifier algorithm.
- Matrix Factorization has given the least score.
- The best score generated by the Ensemble learning algorithm with 0.87752.

Methods	**Score**
Ensemble	0.87752
Random forest	0.87683
Gradient Boosted Tree Classification	0.87677
Sum (Artist * 0.8 + Album * 0.15 + Mean (Genre [1-21]) * 0.05	0.87050
Mean (Artist * 0.7, Album * 0.2, Sum (Genre [1-21]) * 0.1) 0.87022	0.87022
Logistic Regression	0.86977
Mean (Artist * 0.8 + Album * 0.15 + Sum (Genre1-21) * 0.05)	0.86938
Decision Tree Classification	0.85731
Mean (Artist, Album, Genre [1-21])	0.84338
Sum (Artist + Album + Genre [1-21])	0.84291
Max (Artist, Album, Genre [1-21])	0.81215
Matrix Factorization	0.61088

5. Conclusion

This project explored the realm of recommendation engines by utilizing many different algorithms to predict if a user would like a track or not with the Yahoo music dataset. We made our hands dirty with Data Pre-processing, found different approaches to improve the prediction, and wrote an entirely new algorithm. Leader Board also helped us have healthy competition among our class groups which motivated us to achieve higher scores. Overall, the project was an excellent way to obtain practical experience with recommendation engine algorithms.