Song Recommendation System Using XGBoost

Course Project: CSL 7550 - Machine Learning - 1

Abhijit S Iyer
Department of CSE
Indian Institute of Technology, Jodhpur
Jodhpur, India
M21CS001

Jayesh Budhwani
Department of CSE
Indian Institute of Technology, Jodhpur
Jodhpur, India
M21CS007

***Abstract*—This report provides a sneak peek into the Introduction to XGBoost and then present an algorithm implemented for 'Music Recommendation System', using the Extreme Gradient Boost Algorithm (XGBoost). The XGBoost algorithm is based on decision trees, which uses a gradient boosting framework. For our project, we have used the XGBoost algorithm to predict whether the user has played a certain song for more than once in a specific time interval. In such a case, i.e. if the user has listened to a specific song for more than once after the first listening event, the chances of similar songs being suggested to the user increase manifold.**

I. Introduction

Music is food for the soul is what a general saying goes by. But the right kind of music is of utmost importance, since different people have different tastes in music, and the right kind of music needs to be listened to in case someone wants to fulfill their musical desires. Therefore, as an organization/ company that has taken a foot in the direction of building a music player application that would suggest songs to users based on their interest, it becomes extremely crucial for them to recommend the right kind of songs to listeners. Thus, XGBoost algorithm comes to the rescue, which can be tweaked in such a way that based on various parameters that a song holds, a song can be suggested to the user with the help of classification of the parameters. The reason why XGBoost algorithm was chosen over other algorithms is that it is a highly refined and efficient version of the decision tree classifier algorithm, since is uses tree pruning, parallel processing and regularization of data values to avoid any bias.

II. Problem Statement

We framed our task to implement a recommendation system that would be based on XGBoost Algorithm. The datasets used were taken from Kaggle (KKBox Music Recommendation Datasets). The algorithm would take in various parameters of the datasets available to come to a conclusion on the list of songs that can be suggested to a user.

III. Algorithms Used

To facilitate our purpose, we have experimented with two search algorithms:

A. XGBoost Algorithm

Extreme Gradient Boosting Algorithm, also known shortly as the XGBoost Algorithm, is a highly refined and boosted version of the Gradient Boosting Algorithm; which in turn employs gradient descent algorithm on decision trees to minimize the loss in prediction. It was developed as a research project in The University of Washington, and was presented in the SIGKDD conference in 2016. The ML world often calls the XGBoost algorithm as 'Gradient Boost on steroids'. It makes use of hardware and software optimization techniques to yield superior results in efficient time by making use of as less hardware resources as possible. They work on the principle of boosting weak decision tree classifiers using the gradient descent architecture. It makes use of tree pruning, parallelization and hardware optimization techniques to produce best results.

XGBoost uses the following system optimization techniques[1]:

- **Parallelization**: XGBoost uses parallelization in the process of sequentially building the tree. It interchagnes the execution order of loops executed (inner loop for calculating features, outer loop for ennumerating the leaf nodes of a tree) thus enabling the improvement of computation time.
- **Tree Pruning**: XGBoost employs a greedy strategy in determining the depth at which the tree should be pruned. It depends on the negative loss criterion at the point of split.
- **Hardware Optimization**: By making use of cache awareness by allocating internal buffers in each thread to store gradient statistics, XGBoost makes use of efficient hardware resources. It also makes use of 'out-of-core' computing to optimize available disk space.

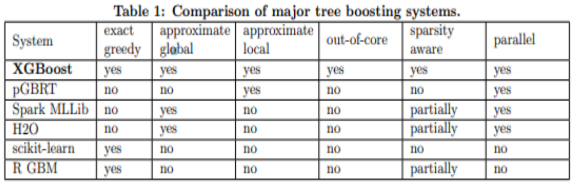
Along with system optimization techniques, XGBoost also uses the following algorithmic optimization techniques:

- **Regularization**: Reguarization is performed on the model during training to reduce changes of the model being overfitted to the taining data. This ensures that the model is less biased towards the train dataset.
- **Sparsity Awareness**: XGBoost employs a sparsity awareness strategy where it admits sparse features for inputs by automatically 'learning' best missing value depending on

training loss and also handles different types of sparsity points that may be present in the dataset efficiently.

- **Weighted Quantile Sketch**: By making use of weighted quantile sketch, XGBoost manages to apply the distributed weighted quantile sketch algoeithm to efficiently find the optimal split points among weighted datasets.
- **Cross Validation**: XGBoost comes with a built in cross-validation method at each iteration, taking away the need to explicitly program this search and to specify the exact number of boosting iterations required in a single run. A comparison is shown below[2].

Table 1: Comparison of major tree boosting systems.



System	exact greedy	approximate global	approximate local	out-of-core	sparsity aware	parallel
XGBoost	yes	yes	yes	yes	yes	yes
pGBRT	no	no	yes	no	no	yes
Spark MLLib	no	yes	no	no	partially	yes
H2O	no	yes	no	no	partially	yes
scikit-learn	yes	no	no	no	no	no
R GBM	yes	no	no	no	partially	no

B. Random Forest Classifier Algorithm

The Random Forest Classifier is an ensemble method of learning models, which is primarily used for classification purposes. It makes use of the principle of selecting the final classification output as those trees among the ensemble that in majority select a certain class. Random forests in general have the habit of overfitting the training datset. They are better than the decision tree classifier, but are not as good as gradient boosting method or the XGBoost algorithm. In our case, we have used the Random Classifier to compare our results with the XGBoost algorithm, and also to show that the XGBoost algorithm can perform much better than the Random Forest Classifier, which can prove the claim that XGBoost is indeed a superior algorithm when it comes to recommendation systems.

IV. Methodology

This Recommendation system is done using the dataset provided on Kaggle. The recommendation system is based on XGBoost algorithm that is used for Regression as well as Classification. In this system we have implemented a XGBoost and Random forest classifier (for comparasion). For XGBoost we have tried to implement a local search algorithm that uses randomly generated parameters to create a model and test that model on test data after fitting the model on training data. By using such local seach we have tried to get the best parameters that can provide best accuracy over the test data. The local search work by selecting randomly learning rate, max depth, min child weight, and number of estimators that are all hyper parameters used by XGBoost classifier. Then we use these randomly generated parameters to fit a model on train data and then predict its accuracy on test data and store the accuracy and parameter. Finally we search for the best accuracy and used those respective parameters for creating a final model.

Also we have tried to compare the accuracy of XGBoost with Random Forest Classifier in this we have used a Random Forest Classifier provided by sklearn library.

V. Results

We tried experimenting with a sample space containing learning rate 0.1, and 0.01, maximum depth 25, 30, 35, and 40, min child weight 5, 10, and 20, number of estimators 150, 250, 300. Based on above data we have shown below some generated models and their accuracy in table 2.

Rate	Depth	Child Depth	Estimators	Accuracy
0.01	35	20	150	70.20%
0.01	25	20	150	70.16%
0.1	30	5	250	70.56%
0.01	35	10	150	70.27%

Also the comparison between XGBoost and Random Forest Classifier is as shown in table 3.

Algorithm	Accuracy
XGBoost	70.27
Random Forest Classifier	68

VI. Observations

- As we tried Local search if we increase the number of sample values the time increases.
- The Dataset contains very large number of samples and thus for quick implementation we have used only 0.9% of the sample which is giving accuracy of 70% if the size of samples is increased then accuracy will increase.
- We have also observed from the correlation between all the features and also observed that the samples have missing values which are replaced by unknown and nan are replaced by 0 matrix that.
- We tried using Grid search, Genetic Algorithm for parameter tuning but based on our requirement we found local search a good fit for the task. For future work we can use grid search with cross validation for parameter tuning.

References

[1] Medium. 2021. XGBoost Algorithm: Long May She Reign!. [online] Available at: ¡https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d¿ [Accessed 29 November 2021].

[2] T. Chen and C. Guestrin, “XGBoost,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.