Incorporating K-Means and Neural Networks into Music Genre Recognition

Yigit UNALLAR
International Computer Institute
Ege University
Izmir, Turkey
yigitunallar@gmail.com

Keywords—NN; K-Means; Librosa; Python; Feature Extraction; Supervised Learning; Unsupervised Learning

I. Introduction

When we listen to music, we do not simply perceive the acoustics of the sound in a temporal pattern, but also its relationship to other sounds, songs, artists, cultures and emotions. Owing to the complex, culturally-defined distribution of acoustic and temporal patterns amongst these relationships, it is unlikely that a general audio similarity metric will be suitable as a music similarity metric. Hence, we are unlikely to be able to emulate human perception of the similarity of songs without making reference to some audio futures [1]. The success of music classification systems, demonstrates that this difficulty can be overcome by learning the complex relationships between audio features and the metadata classes to be predicted [2]. Previous works in music feature selection confirmed the suggestion that too large feature sets not only slow down the classifiers, but also diminish the obtained quality [3]. Although designed for different objectives (classification, clustering), many music information retrieval systems follow the same recipe with minor variations, adopting a three-step strategy: some features extracted from a music audio, then these features are summarized to generate a music representation, and finally a machine learning model is trained over the summarized representations for the task of interest [4]. This paper evaluates the performance of both unsupervised and supervised learning models in classifying audio clips of different genres. In this regard, both K-means method and NN approach are implemented to identify each genre accurately. To learn from the audio clips, a wrapper library is utilized to convert music files into low level audio data and extract features that will be fed into learning algorithms as parameters. In the following sections, we describe an audio analysis library written in Python, K-means and NN implementations for learning challenge and finally share both methods' performance evaluation and test results.

II. Librosa

Librosa is a library written in Python programming language for music and audio analysis [5]. In simple terms, it provides building blocks necessary to create information regarding a given file. In this study, we have an audio dataset comprised of 4 different genres: “classical”, “pop”, “rock” and “metal” and each genre includes 10 music files from a diversified group of artists and bands. In order for the machine learning models to classify the music files, we have to devise a way to extract some features and let the algorithms work on this low-level data set to train parameters. To keep music analysis simple, below computational methods were used to convert music files into vector form:

- **Mfcc:** Computes Mel-frequency cepstral coefficients,
- **Chroma-Stft:** Computes a chromagram from a waveform or power spectrogram,
- **Mel:** Computes a Mel-scaled power spectrogram,
- **Contrast:** Computes spectral contrast,
- **Tonnetz:** Computes the tonal centroid features.

III. Tensorflow

Tensorflow is an open-source library for numerical computation, specializing in machine learning applications [6]. In this study, we will use simple 2 layers NN for classification task and implement predefined functions in Tensorflow Python library. As we are dealing with multiclass classification problem, we will employ *sigmoid* function in the first hidden layer and *tanh* in the second layer to provide non-linearity. Once the NN structure is ready to train, we will implement *gradient descent optimizer* to minimize the cost function.

IV. K-MEANS

K-means clustering is a type of unsupervised learning, which is used when we have unlabeled data (i.e. data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the feature similarity. Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically. Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid, feature weights can be used to qualitatively interpret what kind of group each cluster represents. This method is simpler to implement compared to NN, but lacks precision due to unlabeled data, and similarity algorithm. We set K=4 as we have 4 different music genres in the dataset and have not employed label data during learning process but performance analysis.

V. IMPLEMENTATION

The devised solution and the algorithms within the scope of this paper work have been tested and implemented in Python 3.5 and run on MacBook Pro Late 2013 computer with MacOS Sierra installed. The given hardware sports dual-core Intel CPU clocks at 2.6 GHz, and 8 GB of RAM. The process of feature extraction from sound clips has been done by three helper methods. First, "*loadMusic*" obtains each music files in the given path, loads them and appends into an array. Next, "*parseAudio*" function, which reads every single sound files individually, extracts their futures by calling the third function "*featureExtraction*" and labels each file by looking at the subfolder name. To train and test our algorithm, 80/20 rule has been implemented to divide our dataset into training and test sets. Lastly, "*oneHotEncoder*" function has been defined to encode each label into binary form, thus facilitating both training and performance evaluation. Next, we defined "*epoch*" value, which indicates the number of how many times iterative training will be run, the number of classes which stands for 4 music genres, the unit sizes in each hidden layer which empirically defined as 280 and 300 respectively, and learning rate which is set to 0.1. As for K-means implementation, we kept using the same helper methods already defined to extract music features and fed them into K-means function implemented in *Scikit-learn* Python library [7]. We have not utilized "*oneHotEncoder*" function intentionally, as we do not need label data during clustering. However, we came across with another problem when K-means outputs the clustering results. In "*parseAudio*" function, we encoded each genre type into numbers (ie. pop = 1) and when the outputs returned, K-means attach each cluster a different number thus evaluating the performance harder. To eliminate this problem, we brought in the heuristic methods and human factor to find out which genre-number label corresponds to which cluster number. This solution will be further discussed in the following section.

VI. TEST RESULTS

During the training process, we have implemented NN algorithm to obtain the best result and we will be sharing the highest scores so far in this section. Training the NN took approximately 30 minutes and analyzing the songs and extracting features took up most of it. We input 32 x193 matrix into NN as we have 32 songs in 4 different genres as training data. Trained NN has been tested on test data which comprises 8 songs in 4 different genres. As per the f-score calculation we obtained %87.5 accuracy in genre recognition. Figure 1 depicts the cost function output over 5000 iterations.

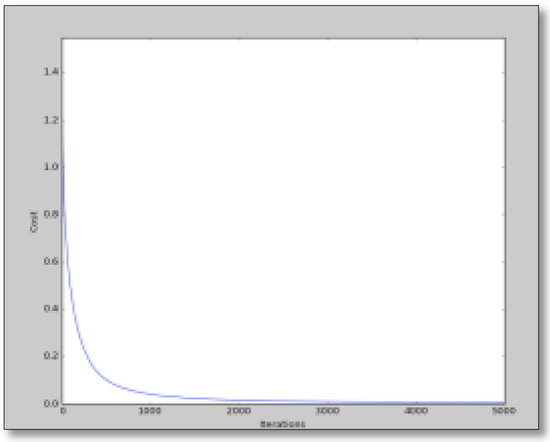

Figure 1. NN Cost Function

Table 1. NN Test Results

Pop=1, Classical=2, Metal=3, Rock=0	
True Labels	1,1,2,2,3,3,0,0
Predicted Labels	1,2,2,2,3,3,0,0

As shown in Table 1, our NN predicted the second song mistakenly. The rest of the predictions has been done correctly. This result has shown us that, our feature extraction methods might be altered to better distinguish pop songs from classical musics.

As for K-means algorithm, we used the same training set as an input which is 32 x 193 matrix. Training period took less time compared to NN's. The test results in Table 2 shows that, K-means very poorly performed compared to NN's performance.

Table 2. K-means Test Results

True Labels	1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0
Predicted Labels	2 2 0 2 3 2 3 0 2 2 2 2 2 0 2 2 2 3 1 1 3 0 1 1 0 0 2 2 0 0 0 0

As depicted above, in order to evaluate the performance of K-means algorithm, we have to determine genre-number labels and cluster number pairs. Assuming that 0 corresponds to cluster number 0, we can see that K-means has done well in “Rock” genre. Another heuristic approach might be implemented in “Classical” genre, which apparently corresponds to cluster number 2. As per the genres “Pop” and “Metal”, we are not able to easily distinguish corresponding clustering numbers but predicting that clustering number 3 and 1 correspond to “Pop” and “Metal” respectively. This leads us approximately %55 accuracy in K-means method.

VII. CONCLUSION AND FURTHER WORK

In this paper, we showed how to extract features from music files and train both unsupervised and supervised learning methods over this data set to recognize the genres accurately. As expected, supervised learning, specifically NN has over performed K-means, which is trained over unlabeled data. We believe that, our algorithms may perform better by altering some parameters as machine learning field, in general, evolves by conducting many trial and errors. As per K-means, we assume that some features made it more puzzling for the algorithm cluster each genre distinctly, thus performing poorly. In the future work, we aim to implement semi-supervised learning over both unlabeled and labeled data set to scale up our implementation and decrease the cost of obtaining labeled data which is costly and scarce in today’s technology world.

REFERENCES

[1] K. West, S. Cox, P. Lamere, “Incorporating Machine Learning into Music Similarity Estimation”, AMCMM’06 October 27, 2006, USA

[2] G. Tzanetakis, P. Cook, “Musical Genre Classification of Audio Signals”, IEEE Transactions on Speech and Processing, Vol. 10, No: 5, July 2002

[3] I. Vatolkin, W. Theimer, G. Rudolph, “Design and Comparison of Different Evolution Strategies for Future Selection and Consolidation in Music Classification”, 2009 IEEE Congress on Evolutionary Computation, pages 174-181.

[4] P. Zhang, X. Zheng, W. Zhang, S. Li, “A Deep Neural Network for Modeling Music”, 5th ACM on International Conference on Multimedia Retrieval Pages 379-386

[5] https://librosa.github.io/librosa/index.html

[6] https://www.tensorflow.org/

[7] http://scikit-learn.org/stable/#