SMAI Project 17: Music Genre Classification (Team 25)

Sri Kiranmayee J 201501155
Aaron Jacob Varghese 201501158
Amal Santhosh 201501208
Aniruddha Vivek Patil 201501146

Abstract

The purpose of this project was to implement the Music Genre Classification paper published by et al. The code was written in python (with SkLearn, Keras and Scikit libraries) and the results were fairly close to the results shown in the paper. There were also more classifiers that we used than what the paper mentions.Additonally, we also present a Recommneder System to suggest similar songs for the specific song provided in the user interface we built.

1. Introduction

For the purpose of implementing this paper, we followed the exact steps as explained in the paper. We started with the feature extraction that requires us to extract the Mel Frequency Cepstral Coefficients (MFCCs) in order to replicate what humans perceive as the change in frequencies. Then, using these coefficients, we used different classifiers like KNN, K-Means, SVM and a neural network to classify the songs into genres. In addition to replicating the paper, we also implemented random forests and gaussian mixture models for classification. We also made a UI so that one can upload a custom song to and it will try to guess the genre. It will also recommend tracks that are related to the uploaded track. Compared to this paper that classifies between 4 genres, we tried and got decent results for classification between 5 genres.

2. Implementation

The implementation was done in python.

2.1. Feature extraction

For the purpose of extracting feature arrays from each song, we used 25 MFCC Coefficients using about 70% of the song. The separated portion of the song was divided into windows of 0.2 seconds with a step size of 0.1 seconds. Then DFT is applied to each of the frames and the corresponding periodogram power spectral estimates are found. We then proceed to compute the Mel filterbank, a set of 42 triangular filters which are applied to the earlier power spectrum. The filterbank consists of 42 vectors of length 512 each (assuming 512 point DFT). Applying log and then applying DCT to the vectors leaves us with 42 coefficients out of which the top 25 are taken. These are the MFCC coefficients we use for our classifiers.

2.2. Neural nets for classification

The Neural Nets were implemented using the Keras library. The feature vectors were first compressed by combining the mean vector and the top half of the covariance matrix, giving a feature vector of length 225. The neural net architecture consisted of an input layer of size (1,225), 10 hidden layers and an output layer which outputs an array of length 4. The training labels were provided as (1,0,0,0) for Classical, (0,1,0,0) for Jazz, (0,0,1,1) for Metal and (0,0,0,1) for Pop.

2.3. SVM for classification

SVM classifiers conventionally provides a means of classification between the data into two classes. Here, the idea was to extend this make the data fall into multiple classes (i.e., 4 in this case corresponding to the 4 genres). We used the multi-class nu-SVM which basically differs from the c-SVM as the former has the capability to control the number of support vectors.

2.4. KNN for classification

We found the k-NN implementation to be pretty straightforward by following the concepts we had learned about this machine learning technique from class. We used two distance metrics: KL Divergence and Hellinger's distance with the motive of comparing the performance of both. We tweaked the parameter k (the number of nearest neighbors) quite a bit to make sure we get the optimal results.

2.5. Kmeans for classification

We implemented a custom k-Means algorithm to accommodate our feature set, using KL divergence between two songs as the distance metric and tolerance as 0.0001. We used various other distance metrics also to compare and contrast the clustering performance.

2.6. Random forests for classification

The Random Forest classifier was a technique that was not mentioned in the research paper we referred to but implemented our own custom version and tested by tweaking the number of decision trees in the forest to attain the optimal accuracy of classification we could.

2.7. Gaussian Mixture Model(GMM) for classification

Gaussian Mixture Model is another classification technique we implemented in addition to the ones specified in the paper. The idea was to observe the performance of the variety of the classification techniques.

2.8. Recommendation System

We developed a recommendation system which specifies the genre of the audio file input to it using the KL divergence distance function to find the closest music files in our existing dataset. We developed the User Interface using React.JS.

3. Results

Most of our results matched that given in the paper. The results of our attempt to classify 5 genres were decent.

3.1. 4 Genres

Below are the accuracies we obtained for each classification technique. For KNN and Kmeans using KL divergence:

Method	Classical	Pop	Jazz	Metal	Overall
SVM	0.894	0.928	0.628	0.95	0.823
KNN	0.947	0.928	0.571	0.95	0.813
KMeans	0.631	0.785	0.771	1	0.794

For Random Forest, we observed an overall accuracy of 87% for 100 trees. For GMM, we observed an overall accuracy of 35%. For Neural Net, we observed an overall accuracy of 94.6 %.

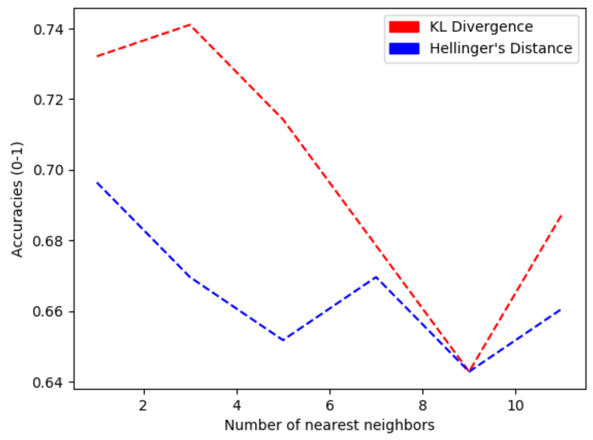
3.2. 5 Genres

Below are the accuracies we obtained for each classification technique. For KNN and Kmeans using KL divergence:

Method	Classical	Pop	Jazz	Metal	Reggae	Overall
SVM	0.842	0.821	0.451	0.95	0.6	0.714
KNN	1	0.857	0.428	0.95	0.6	0.741
KMeans	0.894	0.928	0.1	0.628	0.95	0.823

For Random Forest, we observed an overall accuracy of 80% for 100 trees. For Neural Net, we observed an overall accuracy of 85 %.

For KNN we observed that using 3 neighbors produced the best results as is evident from the following graph.



4. Challenges Faced

The most challenging aspect of our efforts have been the task of feature extraction using the concept of Mel Frequency Cepstral Coefficients (MFCC). After this, the implementation of each of the classification techniques we used presented issues of their own. The duration to run certain techniques and acquire the results and then to retry by changing certain parameters was time consuming. The user interface we decided to present the project which required learning a new technology called ReactJS with an Express server.