Report - Music Genre Classification

ECEN-5322 - Fall 2015

Krishna Chaitanya Sripada, Siva Palakurthi, Soumyatha Gavvala

December 15, 2015

1 Introduction

The rapid growth of the digital media in the past decade has made it easier for man to associate with his interests at a faster and effortless ways. Man has been finding new ways to make music, find, store it and organize it. Finding music was difficult without the name or lyrical content and this lead to the necessity of searching music by its content. Music Information Retrieval (MIR), which is extraction of information about the music from the raw data is a small but fast growing field of research in the similar application. MIR deals with analyzing, creating and categorizing music.

Our project's aim is to classify a vast collection of musical tracks by their content into different genres. We have taken 729 tracks of known labels as the training data on the basis of which we build the algorithm to categorize unknown test music files. These files were acquired from the website of the 5th International Conference on Music Information Retrieval. These 729 songs belong to six genres.

- Classical: 320 songs
- Electronics: 115 songs
- Jazz/blues: 26 songs
- Metal/punk: 45 songs
- Rock/pop: 101 songs
- World: 122 songs

For the classification of the raw training data given to us, we are using MATLAB as the platform for our algorithms. We are provided with a Music Analysis (MA) toolbox to process the songs which are Pulse Code Modulated and sample them into data that is easier to handle on MATLAB.

The issue we face is the large size of the data and the classification into genres based on the content which requires the analysis of features of the songs like bandwidth and spectral roll-off. When the songs are converted into matrices, we get a data set of high dimensions and so in order to make it easier to work on the data, we reduce the dimensionality of the data.

Then we classify them into groups according to their labels based on the distances between the songs. The distance between two songs of the same genre would be less than that of two songs of different genres. On this basis we cluster the training set into groups thus building an algorithm to classify unknown music files.

2 Distance in song-space

For the categorization of these 729 songs into their respective labels, we need an appropriate basis to differentiate them. This we do by measuring the distance between the songs. The distance between two songs of the same genre would be less than that of two songs of different genres. The distance in question is the L2 or Euclidean distance, which is the difference in the sum of squared terms between two given vectors.

One of the method we have implemented is the calculation of Frobenius norm of the MFCC values. Frobenius norm is also called as Euclidean norm but is not L2 norm. The Frobenius norm of different songs was compared to measure the distance between them.

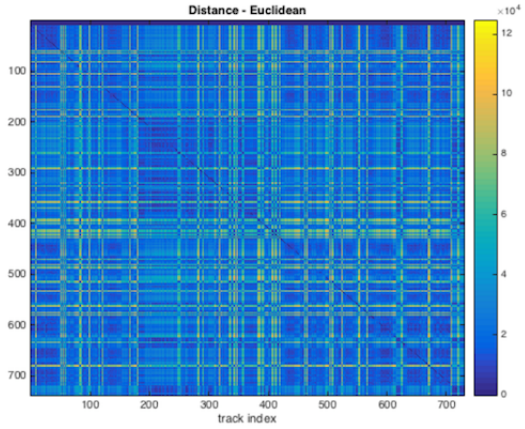

Figure 1: Graphical representation of the distance matrices we used. Dark blue regions represent songs which are close to one another while the yellow regions represent a larger distance between songs.

3 Dimension reduction

For the analysis of the training data we need to sample the given songs. We get hundreds of thousands of samples and the dimension of a song in its raw form is huge and impractical to work with directly. So we used several dimensionality reduction techniques in order to reduce the data to dimensions that allowed us to work with the hundreds of songs in the training data set.

3.1 MFCC

Mel Frequency Cepstral Coefficients (MFCCs) [2], are a pre-processing technique that is frequently used in speech recognition and has been adapted for music information retrieval. The raw data when sampled has dimensions in terms of thousands and it is a strenuous

procedure to analyze data of such high dimensions. Due to the high number of samples, the data in each sample is not sufficient for analysis. Hence we requires some pre-processing of the data in order to extract useful information.

From the MA toolbox provided to us, we make use of the MFCC function to obtain MFC coefficients. These coefficients extract information from songs that is similar to how the human ear hears music that is the content. We first transform the songs into their MFCC coefficients, effectively reducing the dimension of each song while retaining useful information.

3.2 Fast Fourier Transform (FFT)

Fast Fourier transform(FFT) is used to compute discrete fourier transform of a sequence. It basically converts audio signal in its original form to representation in the frequency domain. We computed FFT representations of all songs using matlab fft function which is a vector of size 512 (which is configurable).

The diagrams below are FFT representations of two audio samples from training data.

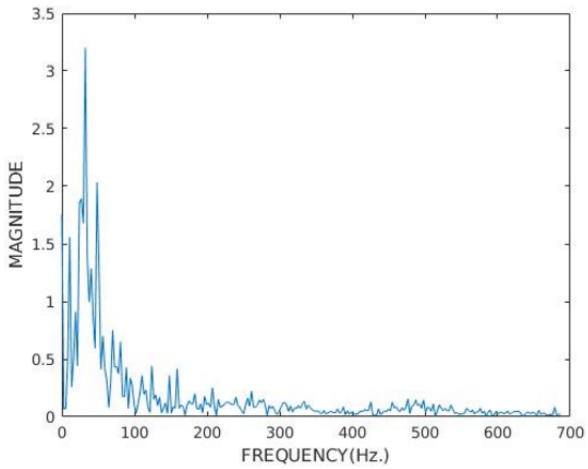

Figure 2: FFT representation of sample1

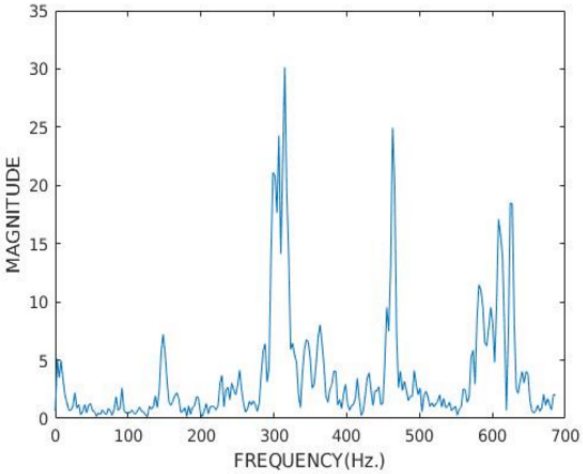

Figure 3: FFT representation of sample2

3.3 Principal Component Analysis

Principal Component Analysis is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

PCA [1] also called the K-L method searches for k n-dimensional orthogonal vectors that can best be used to represent the data where k $\leq$n. The original data are thus projected onto a much smaller space resulting in dimensionality reduction. For our experiments, the results of MFCC are passed through the PCA technique which even further reduced the dimension of the songs. The implementation is provided at the end of this document.

3.4 Random Projections

The random projection module implements a simple and computationally efficient way to reduce the dimensionality of data by trading a controlled amount of accuracy for faster

processing times and smaller model sizes [3].

The fastness of Random Projections is because of the theoretical concept of JL Lemma. In mathematics, the Johnson-Lindenstrauss lemma is a result concerning low-distortion embeddings of points from high-dimensional into low-dimensional Euclidean space. The lemma states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved. The map used for the embedding is at least Lipschitz, and can even be taken to be an orthogonal projection. Using Johnson-Lindenstrauss, we found $N > 77$. We chose $N = 79$ in order to make it consistent with PCA output.

4 Statistical learning

Once the data had been sufficiently reduced in dimension, we used several statistical learning methods to classify songs according to their genre. Specifically, we examined the techniques of k-nearest neighbors, neural networks and random forests.

4.1 Linear Discriminant Analysis

This technique involves finding linear combination of features that separates two or more classes of objects. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification. Each song is represented as a vector which is the mean of the distribution represented by the MFCC coefficients. LDA is closely related to PCA as both approaches look for linear combinations of variables that best explain the data. fitcdiscr which is the discriminant analysis classifier in MATLAB was used for classifying the data. 90% of the data from each genre was used as the training data and 10% of the data from each genre was used as the testing data.

4.2 K-Nearest Neighbors

The kNN algorithm [5] simply computes the k nearest neighbors to the song which is to be classified, where $k \in N$ is a natural number. The song is then classified into a genre based on the genre of its nearest neighbors. To avoid ties, k is usually chosen to be an odd number.

This is one of the simplest algorithms and fastest available and can be easily applied to any of our distance matrices by simply sorting the matrix to find the nearest neighbors.

4.3 Neural Networks

This technique was explored because of the neural network's ability to model complex functions and detect patterns in large data sets. Among the various neural network types available, we used the most simple and common feedforward network of perceptrons using the tangent-sigmoid transfer function.

To train the neural network, target vectors corresponding to the training data are needed. For this purpose, we created a target vector of size 6 x 729 (since we have 6 classes and 729 music files) and each column has a bit 1 set if the song belongs to that class otherwise it is set to 0. The next step for training was to select a training algorithm. The algorithm used to train the

networks was the Levenberg-Marquardt backpropagation algorithm. Since neural networks tend to suffer from overfitting, the data was split into three subsets. 80% of the data was used for training, 10% as the cross-validation data and 10% as the testing data. The algorithm used the training subset to find a gradient based descent and used the validation subset to confirm that a descent had occurred. The training algorithm performed this supervised gradient descent until some specified performance goal was met or until the validation subset error failed to descend. If the data failed to descend, the network was reinitialized to a new starting point and a new gradient descent was attempted.

Genre	**Target Vector**
Classical	[1 0 0 0 0 0]
Electronic	[0 1 0 0 0 0]
Jazz/Blues	[0 0 1 0 0 0]
Metal/Punk	[0 0 0 1 0 0]
Rock/Pop	[0 0 0 0 1 0]
World	[0 0 0 0 0 1]

Table 1: Neural Networks: Target Vector representation for the 6 classes

4.4 Random Forests

A random forest is a classifier based on decision trees. The algorithm involves creating a large number of decision trees with controlled variation. Points are then classified using the mode of a vote of all the decision trees about which class the point belongs to. This method is advantageous because random forests provide accurate classification while remaining robust against overfitting. However, in order to classify songs, it is necessary to represent each song as a vector of values. In order to do this, the samples of MFCC coefficients were averaged, providing the mean of the distribution represented by the MFCC coefficients. This vector was used as input to the random forest classifier.

The random forest implementation is available in Matlab [6] which creates an ensemble of bagged decision trees. It generates in-bag samples by oversampling classes with large misclassification costs and under sampling classes with small misclassification costs. Consequently, out-of-bag samples have fewer observations from classes with large misclassification costs and more observations from classes with small misclassification costs. We can avoid large estimated out-of-bag error variances by setting a more balanced misclassification cost matrix or a less skewed prior probability vector.

5 Experiments

For all of the classification methods explored, performance was evaluated by performing 10 iterations. This resulted in 10 confusion matrices. The element-wise mean was computed to indicate the overall performance of the classifiers described below.

In order to visualize the high-dimensional data, metric multidimensional scaling (MDS) was used which is a technique that projects data onto a small number of dimensions while trying to maintain the pairwise distances between each point. The MDS plots for the mean of the

MFCC coefficients and for the Gaussian mixture model are shown in figures 4 and 5, respectively. Although a lot of separation between genres is lost when projecting onto only two dimensions, these figures show that there is definitive clustering that is occurring for each genre.

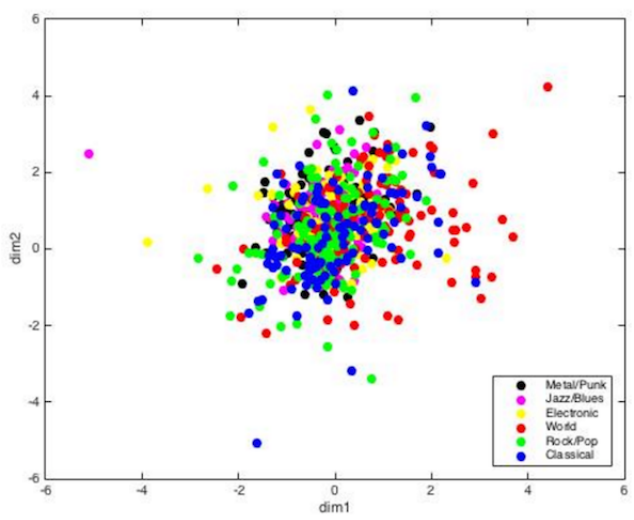

Figure 4: Mean of MFCC distribution

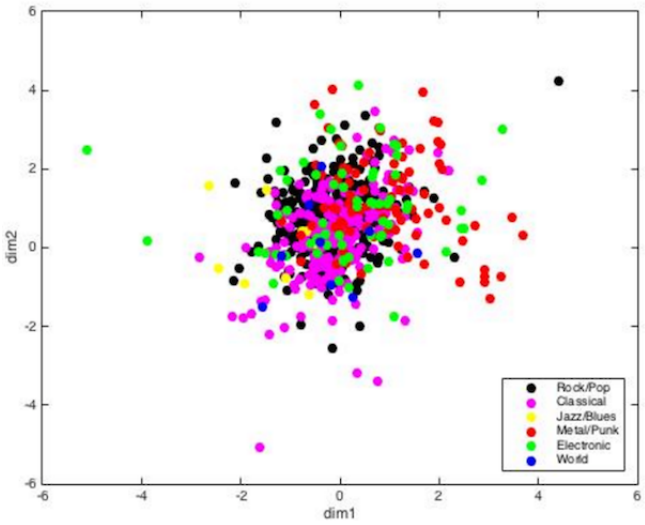

Figure 5: Mixture of Gaussian

5.1 MFCC Parameters

MFCC coefficients are a set of DCT decorrelated parameters, which are computed through a transformation of the logarithmically compressed filter-output energies, derived through a perceptually spaced triangular filter bank that processes the Discrete Fourier Transformed (DFT) speech signal.

MFCC Parameters can be considered as the extracted features as MFCC's in a way represent the Timbre of the songs. This way of representing the song is usually a lossy transformation but, computational complexity and robustness are two primary reasons to allow losing some in information. Increasing the accuracy of the parametric representation by increasing the number of parameters leads to an increase of complexity and eventually does not lead to a better result due to robustness issues. The greater the number of parameters in a model, the greater should be the training sequence. Thus, we take the mean values of all the MFCC's corresponding to one song and cluster them using the Gaussian Mixture Model.

5.2 Mean of MFCC Coefficients

Representing a song by the mean of its MFCC's had decreased the complexity and computational time rather than taking all the MFCC values of a song. This mean of MFCC's was given as input to all the classifiers that we have designed, specifically allowing the usage of Random forests. The most expensive computation involved calculating the MFCC coefficients for every song, which took under 10 minutes. The remainder of the calculations were done within a couple more minutes, making this algorithm very reasonable as a method for large-scale classification problems involving music.

Ultimately, the choice of music representation that was made use in this project are:

1. Mean of MFCC distribution
2. Random Projections using Johnson Lindenstrauss method

5.3 Gaussian Mixture Models

Feature selection is followed by a classification algorithm to generate the data for classification and here were are using Gaussian Mixture Model [4]. The GMM with expectation maximization is a feature modelling and classification algorithm widely used in the speech based pattern recognition, since it can smoothly approximate a wide variety of density distributions. Gaussian Mixture Models are often used for data clustering. Usually, fitted GMMs cluster by assigning query data points to the multivariate normal components that maximize the component posterior probability given the data. Moreover, GMM clustering can accommodate clusters that have different sizes and correlation structures within them. Because of this, GMM clustering can be more appropriate to use than, e.g, k-means clustering.

We specify the number of clusters before fitting the model, which specifies the number of components in the GMM. Also, the type of covariance matrix, diagonal or full, is specified. Geometrically, the covariance structure determines the shape of a confidence ellipsoid drawn over a subpopulation or a cluster. Basically, all the mean values of MFCC's of each song are taken in consideration and clustered using the GMM. This enables us to visualize the basic clustering before using any classifiers on the data that we obtained from representing the music into coefficients of better dimensions.

5.4 Linear Discriminant Analysis

The purpose of linear discriminant analysis is to classify the object into one or more groups based on the features that best describe the data. These groups are the label information that we want the classifier to predict.

The fitcdiscr method in Matlab helps facilitate performing the linear discriminant analysis classification step. The mean of the MFCC distribution forms the feature vector and the labels for these feature vectors are supplied as parameters to the fitcdiscr method. This method generates a linear discriminant analysis model that is then used to predict the testing data labels.

The training data supplied to the classifier was the results obtaining from Random Projects using Johnson-Lindenstrauss theorem and the mean of the MFCC distribution. The results of the confusion matrix are shown in tables 2 and 3 respectively.

	Classical	Electronic	Jazz/Blues	Metal/Punk	Rock/Pop	World
Classical	**0.7013**	0.026	0	0	0.026	0.2468
Electronic	0.0476	**0.6667**	0	0	0.2381	0.0476
Jazz/Blues	0.1429	0.1429	**0.4286**	0	0.2857	0
Metal/Punk	0	0	0	**0.7333**	0.2667	0
Rock/Pop	0	0.25	0.05	0.1	**0.6**	0
World	0.2727	0.1818	0	0.0909	0.2727	**0.1818**

Table 2: Linear Discriminant Analysis: Element-wise mean of confusion matrix for all 6 genres using mean of MFCC distribution

	Classical	Electronic	Jazz/Blues	Metal/Punk	Rock/Pop	World
Classical	**0.7167**	0.0333	0.0333	0	0.0333	0.1833
Electronic	0	**0.5556**	0	0.0556	0.1111	0.2778
Jazz/Blues	0.1667	0.1667	**0.6667**	0	0	0
Metal/Punk	0	0	0	**0.5833**	0.4167	0
Rock/Pop	0.0417	0.2083	0	0.1667	**0.5417**	0.0417
World	0.2188	0.2813	0	0	0.25	**0.25**

Table 3: Linear Discriminant Analysis: Element-wise mean of confusion matrix for all 6 genres using mean of Random Projection coefficients

5.5 K-Nearest Neighbors

KNN algorithm is used either for classification or regression. When used as a classifier, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

We used classifyKNN_D_Multi function which is part of matlab's Audio analysis Library for classifying songs. The best results we got were at k = 9. The results of the classifier are shown in Table 4 and Table 5.

	Classical	Electronic	Jazz/Blues	Metal/Punk	Rock/Pop	World
Classical	**1**	0	0	0	0	0
Electronic	0.0909	**0.6818**	0	0	0.1364	0.0909
Jazz/Blues	0	0	**1**	0	0	0
Metal/Punk	0	0	0	**1**	0	0
Rock/Pop	0	0.1111	0	0	**0.7778**	0.1111
World	0.3333	0	0	0	0.0833	**0.5833**

Table 4: KNN: Element-wise mean of confusion matrix for all 6 genres using mean of MFCC coefficients

	Classical	Electronic	Jazz/Blues	Metal/Punk	Rock/Pop	World
Classical	**1**	0	0	0	0	0
Electronic	0.0667	**0.6**	0.1333	0.0667	0.1333	0
Jazz/Blues	0	0	**1**	0	0	0
Metal/Punk	0	0	0	**1**	0	0
Rock/Pop	0	0.2727	0	0.1818	**0.3636**	0.1818
World	0.3333	0.1667	0	0	0.0833	**0.4167**

Table 5: KNN: Element-wise mean of confusion matrix for all 6 genres using mean of Random Projection coefficients

5.6 Neural Networks

The type of neural network that was chosen to perform the music classification was a feedforward back-propagation network. Within this type of network there are several parameters that affect the performance of the network. The number of hidden layers in the network are chosen to be 10 and the algorithm used to train this network was the Levenberg-Marquardt back-propagation algorithm. Over several iterations of the classification technique, we noticed that the default hidden layer size of 10 gave the best results. This hidden layer needs to be chosen in such a way that the trade off between modeling ability and training complexity is balanced.

The transfer function selected was the tangent-sigmoid transfer function. This transfer function was chosen for its fast computation time as well as its ability to model many complex functions very well.

The training data supplied to the classifier was the results obtaining from Random Projects using Johnson-Lindenstrauss theorem, Fast Fourier transforms and the mean of the MFCC distribution. The results of the confusion matrix are shown in tables 6, 7 and 8 respectively.

	Classical	Electronic	Jazz/Blues	Metal/Punk	Rock/Pop	World
Classical	**0.9875**	0	0	0	0	0.0125
Electronic	0.0435	**0.7652**	0	0	0.087	0.0957
Jazz/Blues	0.1923	0.1154	**0.3462**	0	0.0768	0.2692
Metal/Punk	0.0222	0.0667	0	**0.6667**	0.2222	0.0222
Rock/Pop	0.0594	0.1089	0.0099	0.0297	**0.7129**	0.0891
World	0.0656	0.082	0	0	0.0328	**0.8197**

Table 6: Neural Net: Element-wise mean of confusion matrix for all 6 genres using mean of MFCC distribution

	Classical	Electronic	Jazz/Blues	Metal/Punk	Rock/Pop	World
Classical	**0.9938**	0	0	0	0	0.0063
Electronic	0.1565	**0.7042**	0.0087	0	0.0783	0.0435
Jazz/Blues	0.3846	0.1538	**0.4231**	0	0.0385	0
Metal/Punk	0.0444	0	0	**0.5333**	0.4222	0
Rock/Pop	0.1881	0.1089	0	0.0495	**0.6634**	0
World	0.5328	0.1393	0	0	0.041	**0.2869**

Table 7: Neural Net: Element-wise mean of confusion matrix for all 6 genres using FFT coefficients

	Classical	Electronic	Jazz/Blues	Metal/Punk	Rock/Pop	World
Classical	**0.9813**	0	0	0	0	0.0188
Electronic	0.0957	**0.6522**	0	0	0.0957	0.1478
Jazz/Blues	0	0.1154	**0.5769**	0	0.1923	0.1154
Metal/Punk	0.0222	0.0667	0	**0.6**	0.2667	0.0444
Rock/Pop	0.0792	0.1386	0	0.0297	**0.7228**	0.0396
World	0.2459	0.0738	0	0.0082	0.0656	**0.6066**

Table 8: Neural Net: Element-wise mean of confusion matrix for all 6 genres using mean of Random Projection coefficients

The performance and the error histogram for this classification mentioned in Figures 6 and 7 shows how well the classifier works on the data. From the error histogram, we see that most of the instances run for training are around the zero error line which shows that the classifier performs well with minimal errors. Also more of the errors are bordered between -1 and 1 showing the error variation is very less. From Figure 7, we see that the mean square error gradually decreases for a higher epoch value.

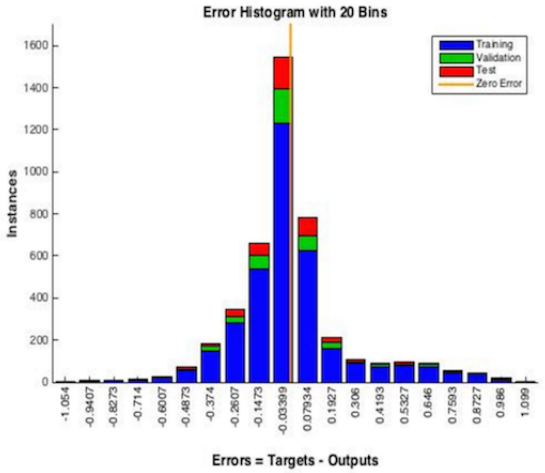

Figure 6: The error histogram on running the data on the neural networks classifier

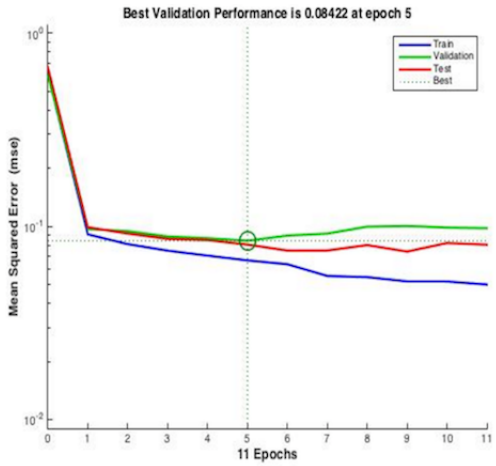

Figure 7: The performance of the classifier on the data

5.7 Random Forests

This classifier was used because of its robustness towards overfitting the training data. Random Forests stems from the idea of decision tree classification but instead we use an ensemble to such decision trees to perform classification.

Matlab provides a TreeBagger method that helps perform this classification. For our experiments, we looped through different values of the number of trees parameter and figured out that the classifier works best for the value '27'.

The training data supplied to the classifier was the results obtaining from Random Projects using Johnson-Lindenstrauss theorem and the mean of the MFCC distribution. The results of the confusion matrix are shown in tables 9 and 10 respectively.

	Classical	**Electronic**	**Jazz/Blues**	**Metal/Punk**	**Rock/Pop**	**World**
Classical	**0.9762**	0	0	0	0	0.0238
Electronic	0.2	**0.6**	0	0	0.1	0.1
Jazz/Blues	0.1667	0.1667	**0.5**	0	0.1667	0
Metal/Punk	0	0	0	**0.5**	0.5	0
Rock/Pop	0.0476	0.0476	0	0.0476	**0.7619**	0.0952
World	0.3667	0.2	0	0	0.1	**0.3333**

Table 9: Random Forest: Element-wise mean of confusion matrix for all 6 genres using mean of MFCC distribution

	Classical	**Electronic**	**Jazz/Blues**	**Metal/Punk**	**Rock/Pop**	**World**
Classical	**0.9277**	0.012	0	0	0.0602	0
Electronic	0.1034	**0.6897**	0	0	0.1034	0.1034
Jazz/Blues	0.4	0	**0.4**	0	0.2	0
Metal/Punk	0	0.1	0	**0.6**	0.2	0.1
Rock/Pop	0.0588	0.0588	0	0.1176	**0.7647**	0
World	0.3214	0.25	0.0357	0	0	**0.2929**

Table 10: Random Forest: Element-wise mean of confusion matrix for all 6 genres using mean of Random Projection coefficients

6 Discussion

Analysis of the Results

We represented songs using two methods (MFCC and FFT) and performed 2 Dimensionality reduction techniques(PCA and Random Projections). We also used mean representations of MFCC and FFT values.

These values are used to train classifier models. We used 4 classifier techniques - kNN, Random Forests, Neural Networks and LDA. Mean values of MFCC distribution performed better than FFT representations and Dimensionality reduction techniques. Though LDA and Random Forests performed better with mean accuracies around 79%, their accuracies are lower than kNN and Neural Nets. We observed that for every classifier, the best classification

accuracy occurred in classifying classical songs. Based on the data set that we trained on, this makes sense since the number of classical songs was far more than any other genre, and in whole comprised about 44% of the data set. Based on this, it is logical for a classifier to be biased toward classical music simply because the training set contained a large number of classical songs. However, although this bias allows highly-accurate classification of classical music, it also causes many songs that were not classical to be incorrectly-identified as classical. In fact, for every classifier that we tested, the largest error for every genre came from incorrectly classifying points as classical when they were not.

The genres that had the lower classification accuracy were jazz/blues and world music. The former was most likely classified poorly simply because there were fewer jazz/blues songs in the training set than any other genre; in total jazz/blues songs comprised less than 4% of the data set. This small sample size is apparently insufficient to make a distinct classification region around jazz/blues and songs of this genre were frequently classified incorrectly. The

world music, however, was the second-largest genre in the training set, but still the classifiers we tested still performed relatively poorly. The reason for this is likely due to the definition of the world genre encompasses songs that didn't belong to any of the other genres. We noticed from Figures 4 and 5 that the world genre is clustered to a much lesser extent than any of the other genres, although it is notable that the genre seems most concentrated in the region where all other genres begin to overlap, supporting the idea of it these songs correspond to the category that the classifier's didn't know which class they belong to. For both the world and jazz/blue genres, however, there were some classifiers that did notably well.

In conclusion, KNN classifier performed very well when mean values are used. It can classify Classical, Jazz and Metal songs with 100% accuracy, while performing poorly in other sections(as seen in Table 5).

Neural Networks performed better than all methods in terms of overall accuracy though it didn't resulted in 100% accuracy in any category, when Mean of MFCC values are used(as seen in Table 6). World Category recorded its best accuracy greater than 81% using Neural Nets.

7 Future Work

We believe that adding spectral features like bandwidth, spectral roll-off, spectral flux and loudness will differentiate clearly between genres and improves accuracy significantly. Adding a pre-processing stage and removing any outliers will also improve our classification models, as they won't be affected by errands. While trying various classification models, we found that some classifiers perform better in some genres, so we believe combining multiple classifiers by giving one's output to another will have positive impact on accuracy.

References

[1] `http://lvdmaaten.github.io/drtoolbox/`

[2] `http://www.cic.unb.br/~lamar/te073/Aulas/mfcc.pdf`

[3] `https://en.wikipedia.org/wiki/Random_projection`

[4] http://www.mathworks.com/help/stats/clustering-using-gaussian-mixture-models.html

[5] http://www.mathworks.com/matlabcentral/fileexchange/45831-matlab-audio-analysis-library

[6] http://www.mathworks.com/help/stats/treebagger.html