Music Genre Classification

Sudeep Kulkarniirlekar (106996755)
Jeet Baru (107189037)

December 9, 2016

Contents

1 Introduction

Musical genres are categorical descriptions that are used to describe music. They are commonly used to structure the increasing amounts of music available in digital form on the Web and are important for music information retrieval. Genre categorization for audio has traditionally been performed manually. A particular musical genre is characterized by statistical properties related to the instrumentation, rhythmic structure and form of its members. The goal of music classification is to use a database of known music in order to classify the genre of an unknown piece of music automatically rather than doing it manually with increased accuracy.

In this report, we explore different methods for music classification. Specifically, we explore the classification of music according to its genre. Results obtained from each method and the comparison of these methods are included.

In our first approach we simply used the code given in this toolbox and checked the accuracy by using the same classifier that we have used in the subsequent approaches for our reference. After that we have taken several approaches to the problem, analyzed result of each approach. All of our approaches have the same combination of kNN classifier and cross validation and were based on different measures of dimension reduction for classification.

2 Problem Statement

To design and implement a content based music information retrieval in MATLAB. For this implementation we applied the techniques learned in the class for dimension reduction techniques like random projection and Principle Component Analysis in our project. The dataset that has been provided to us had 728 tracks of the genres Classical (320), Electronic (114), Jazz/Blues(26), Metal/Punk(45), Rock/pop(102), World(122). Where each of the above songs is sampled at 11025 Hz mono. We have used the above dataset for training and testing a genre classification algorithm so that we are able to classify any query track in any of the above genre with high accuracy. Hence for each track we have about a million samples therefore reducing dimension for classification becomes critical.

3 Approach

- In this approach feature extraction is done using ma_FeatureExtraction function from [4]. After feature extraction the similarity matrix is obtained by running ComputeSimilarities function using ma toolbox. The training and testing data used for k-nearest neighbour classifier is obtained randomly by using k-fold cross validation. After cross validation we have used the kNN classifier to classify the test data points into the respective genre.

- Feature extraction for this approach was done by using ma_mfcc[4]. Following the feature extraction we reduced the dimensionality by using the random projection. By

using this dimension reduction technique and MFCC we managed to reduce the samples of each song from averaging 1 million to some 20 samples. Now we have 20x729 matrix which has the information of all the 729 songs. The K-L divergence technique was used to compute the distance of each song from each other song. As a result of K-L divergence technique we get a 729x729 matrix. Which we passed through the combination of k-fold cross validation and kNN classifier to classify the test points into the respective genre.

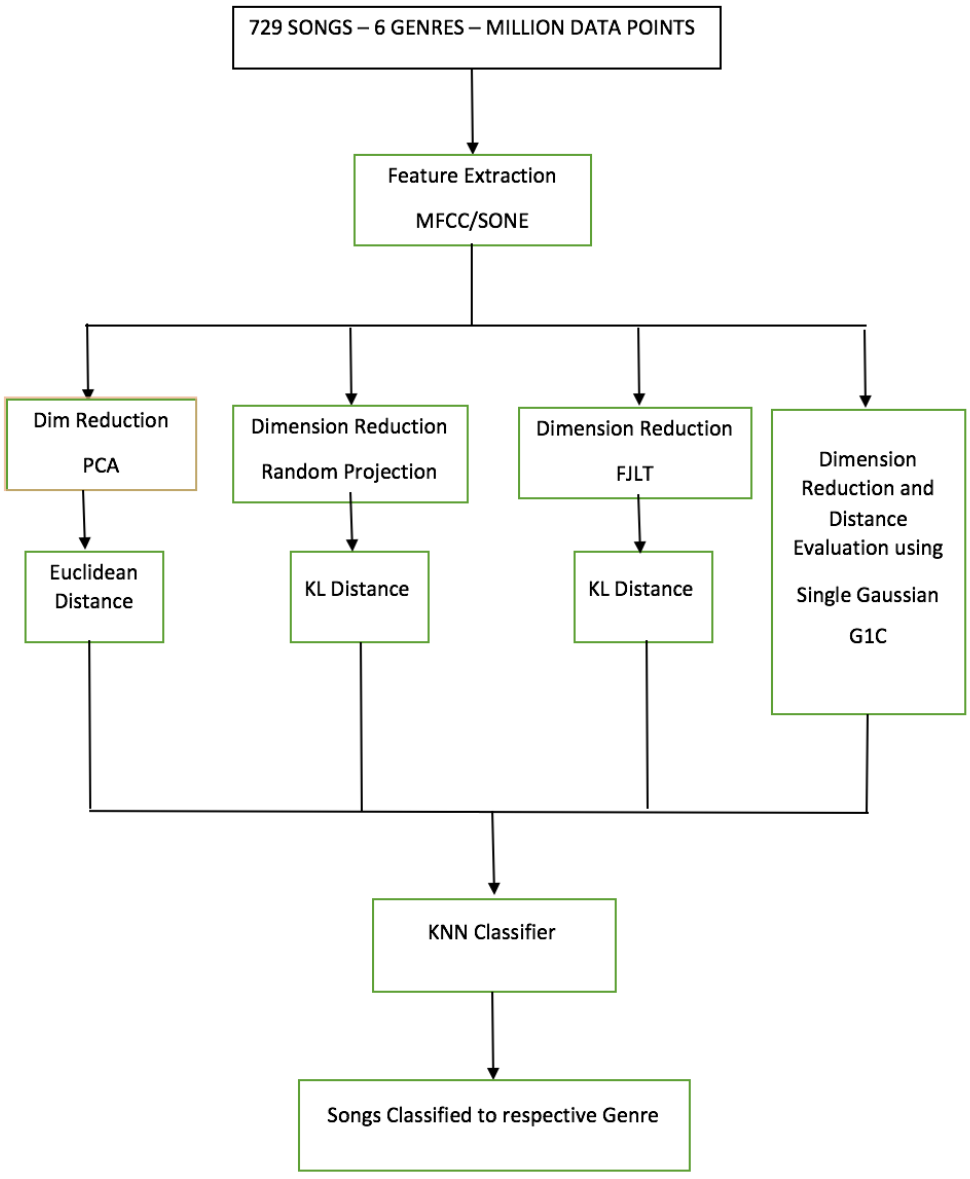

Figure 1:Flowchart defining 1st two approaches

- Like in previous approaches, we extracted features using ma_mfcc function in ma toolbox[4]. This reduced the dimension to some extent, but further dimension reduction was done by using FJLT. Further a similarity matrix was generated using K-L divergence and songs are classified using kNN classifier by using the k-fold cross validation technique.

- In the above approaches before dimension reduction, since the output of the mfcc depended on length of each song, we sequentially selected equal number of samples from all the songs so that we have equal number of datapoints for each song. In this approach instead of sequentially selecting the samples we used PCA, which would not only reduce the dimension, but also give us equal number of datapoints for each song. Using PCA the dimension was reduced to 40. Hence Distance matrix was computed using Euclidean Distance and was passed to kNN classifier for classification.

4 Feature Extraction

4.1 MFCC

This method captures overall spectral shape, which carries all the important information about the instrumentation of its timbres, the quality of singers voice and production effects. It transforms the raw data into interesting information. It drastically reduces the overall amount of data for computational model of music similarities. The transformations roughly model some characteristics of the human auditory system [2].

Working of MFCC: Firstly the given audio waveform is converted into small frame of size decided by the user. We decided the frame size to be 1024 samples in order to achieve a power spectrum of the given audio waveform firstly we take discrete Fourier transform and then apply take log of amplitude spectrum of each frame now in order to model the spectrum based on human auditory system we performed mel scale on it, the human auditory system perceives audio in linear scale linearly at lower frequencies and in logarithmic scale at higher frequencies [1]. Finally we apply discrete Fourier transform and hence achiever MFCC features. We have chosen number of cepstral coefficients to be 10 in order to reduce number of datapoints and hence the complexity.

$$M = 1127 * log\left(1 + \frac{f}{700}\right) \qquad .....(1)$$

Where f is frequency in linear scale and M is frequency is mel scale. From the above formula and graph we observe that mel frequency is linear at lower frequencies and logarithmic at higher frequencies.

Hence on application of MFCC we had a $10 * L$ matrix, where L depends on the length of song.

L on average was found to be 3000. So we have reduced the datapoints from about a million to about $10 * 3000$.

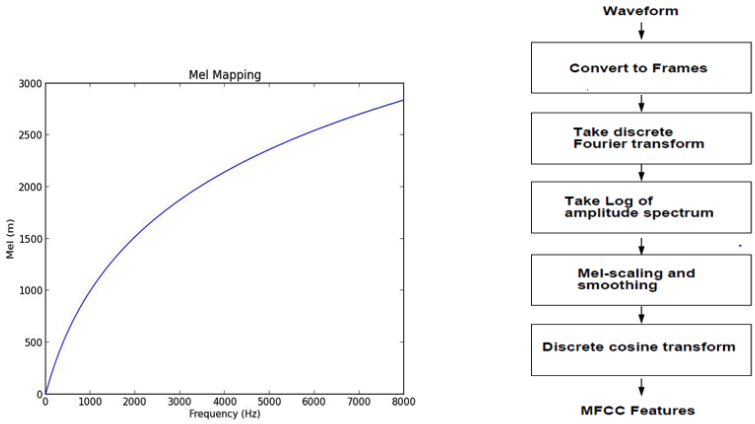

Figure 2:(a)Mel scale frequency vs linear scale (b)MFCC features

4.2 Sone

Another way of extracting featrures from the given audio file is by estimating the loudness sensation as done by human auditory system, for this we used ma_sone [4] function given in matoolbox. This function applied some auditory models to estimate how strong the loudness sensation is per frequency band. Just like MFCC the amplitude spectrum after FFT is here scaled to a bark scale. This scaling is again nonlinear and this is given by following formula

$$z = 7 * log\left(\frac{f}{650} + \sqrt{1 + \left(\frac{f}{650}\right)^2}\right) \qquad .....(2)$$

Feature extraction using sone as compared to MFCC gave lower accuracies for all our approaches, hence we used MFCC.

5 Dimension Reduction

5.1 Random Selection of Samples

In this method the features are selected sequentially and uniformly at random out of the original features, and then after further dimension reduction and generation of distance matrix we passed our data to the classifier.

For our first three appraoches as mentioned above we have used the ma_mfcc citematoolbox for feature extraction. We observed that the number samples for the smallest song in

the database, it was found to be about 4000 samples. So for uniformity we reduced the samples of each song to 4000 samples irrespective of their length. The reduction of the samples was done by dividing the samples of each song into 4000 equal groups and then taking one sample from each group. This method was very logical because we now have equal number of samples for all songs and have also preserved the time varying features of our audio file [7].

5.2 Fast Johnson-Lindenstrauss

We have tried Fast Johnson-Lindenstrauss for the dimension reduction. It is the random distribution of linear mappings from R_d to R_k. Where k is given by Johnson Lindenstrauss Lemma given below:

$$k = \left(\frac{log(d)}{\epsilon^2}\right) \qquad .....(3)$$

According the Lemma when high dimension is projected to a lower dimension subspace the distances are preserved by a factor of $1 \pm \epsilon$ In the above formula, d is the higher dimension and k is the lower dimesion, ϵ is the tolerance by which the distances are preserved. The random embedding ϕ or FJLT can be obtained by the product of the three real valued matrices. ϕ =PHD. Here the matrices P, D are random matrices and H is deterministic [6]. This dimension reduction technique was covered in class and the lemma was proved.

5.3 Random Projection

After feature extraction using ma_mfcc[4], we calculated the matrix which has all the information of all the songs. The matrix was a 4000x729 matrix. Now for dimension reduction, we used Random Projections. This technique too is a consequence of the Johnson Lindenstrauss Lemma Stated above. We used the MATLAB function (randn) to reduce the 4000x729 matrix to a 20x728matrix. Since a Gaussian Random Matrix perfectly projects points to a lower dimension in sync with the lemma.

Larger ϵ implies smaller dimension but poor classification. As a trade of we chose it to be 0.4 for both FJLT and Random Projections. Hence the lower dimension to which the data was projected was 20. So the initial million datapoints have now been reduced to mere 20.

5.4 Principle Component Analysis

It finds directions with maximum variance and which are mutually orthogonal. PCA gives the best reconstruction. It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analysing data. The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data, by reducing

the number of dimensions, without much loss of information[7]. Here for our approach we have used 40 such principle components for the dimension reduction.

5.5 G1C

This method is used to compare given distribution with a Gaussian Distribution. For this the symmetric version of KL divergence is used. To avoid re-computation for each distance computation, the inverse of the covariance matrix used. The rescaling is done for improving the results while combining the spectral distance with other information. The computation time is very less as compared to G30. ma_g1c[4] it reduces the dimension using single guassian distribution, and calculates the distance between the songs having a weighted sum of different distance metrics[3].

6 Computation of Distance Matrix

From the dimension reduction technique have the 20x729 matrix which contains the information of the songs. We computed the vector distance between every two songs for this distance computation we have used the Euclidean and Minkowski distance but we have achieved the highest accuracy using the K-L divergence method. The formula for K-L divergence :

$$D_{KL} = \sum_{j=1}^{n} P log\left(\frac{P}{Q}\right) \qquad .....(4)$$

Where P and Q are the column matrices of the two songs whose distance is being computed. Hence each songs distance is computed with every other song hence we got 729x729 similarity matrix. The similarity matrix that we get by using this method is not symmetric. Hence to make it symmetric we have added the matrix to its own transpose.

7 Statistical Learning

7.1 K-Nearest Neighbours

The classifier we have used is k-nearest neighbours, it is a classification method that uses similarity functions. This similarity is the distance between two data points in this space under appropriate metric. It stores the available cases (training) data and classifies new cases (test data) based on distance function. We have used the default distance funtion which is Euclidean distance. A case is classified by majority vote of its k-nearest neighbours. The way in which the algorithm decides of the points from the training set are similar enough to be considered when choosing the class to predict for a new observation is to pick the k-closest data points to the new observation and to take the most common data points to the new observation and take most common class among these. We have taken k-as to be 10.

The effect on the accuracy with different values of k is as shown below

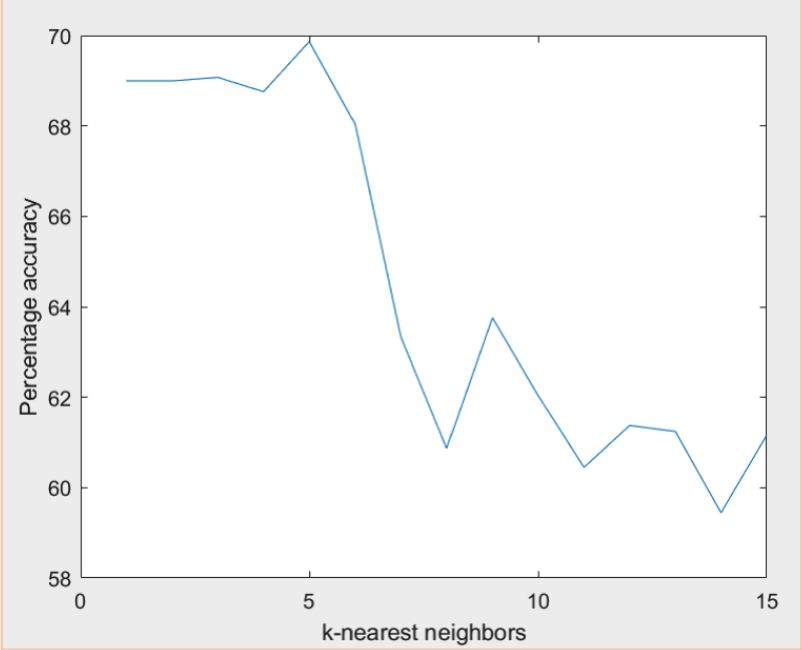

Figure 3: The variation of accuracy with k

To select optimum value of k, we performed an experiment to calculate the accuracy for range of values of k. The output of the experiment is shown above. Going by our intuition for lower values of k the accuracy must be low due to the increased effect of noise, also very high values of k would result in lower accuracies, since few genres in our dataset have very few tracks. The result of our experiment surprisingly shows higher accuracy even for lower values of k. But for our project we have decided to go with k to be 10 as the graph has the local maxima at k=10.

8 Experiments

8.1 Parameter Sweep

For our algorithm we used mfcc over sone for feature extraction since it gave better results but we had to tweak a few parameters in the ma_mfcc function. In order to decrease the no of datapoints we increased the frame size from 512 samples to 1024 samples to achieve the same objective we also decreased the number of cepstral coefficients to 10.

In order to have equal number of datapoints for each song we randomly selected equal number of features from each song. Number of features selected is equal to the total number of available features of the shortest song.

For determining the dimension to which the higher dimension data was supposed to be reduced to we used Johnson Lindenstrauss Lemma. Here we chose the tolerated error ϵ to

be 0.4, which gave us the lower dimension to be 20. For PCA we didn't use random selection of features rather selected 1st 40 directions of maximum variance for all songs.

For kNN classifier as described above we chose to classify songs based on the majority votes of 10 nearest neighbors. Value of k greater than 10 yielded lower accuracy while less than 10 although yielded higher accuracies intuitively meant to be distorted by noise.

8.2 Cross Validation

Rather than using entire data as the training data, some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on new data. This is cross validation.

We have used 5-fold cross validation to test our algorithm. This algorithm randomly generates 5 groups from the database. Of these 5 groups, 1 group goes to test data and the rest 4 groups will go to training data. Now this test data and training data is given to the kNN classifier.

- For the first iteration the 1st group will be the test data and last 4 groups will go to training data.
- For the second iteration 2nd group will be the test data and rest 4 will go to training data and so on.
- Pass this training and testing data to the classifier, so there will be 5 such iterations.
- Compute the confusion matrix in each case.
- Calculate the average confusion matrix from the 5 iterations.
- Compute the standard deviation of the entries in the confusion matrix.

8.3 Confusion Matrix

This confusion matrix (error matrix) is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning matching matrix). Each column of the matrix represents the instance in the actual class and each row represents the instances in the predicted class. Accuracy for the approaches can be calculated by the following formula:

$$Accuracy = \left(\frac{trace(A)}{(number\ of\ genre)} \right) \qquad .....(5)$$

Where A is the confusion matrix.

The results of each of the approaches have been discussed below.

8.3.1 Random projection

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	0.9748	0.1845	0.0333	0	0	0.0250
	Electronic	0.0151	0.4770	0.1300	0	0.0365	0.0254
	Jazz/Blues	0	0.0255	0.1133	0	0	0
	Rock/Pop	0.0034	0.0932	0.2067	0.6779	0.1671	0.0266
	Metal/Punk	0.0067	0.1641	0.4667	0.3221	0.7207	0.2414
	World	0	0.0557	0.05	0	0.0756	0.6817
Accuracy	60.76						

Table 1: Mean random projection

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	0.0190	0.0708	0.0745	0	0	0.0373
	Electronic	0.0146	0.0614	0.1204	0	0.0354	0.0232
	Jazz/Blues	0	0.0238	0.1768	0	0	0
	Rock/Pop	0.0077	0.0739	0.1817	0.2302	0.1189	0.0243
	Metal/Punk	0.0091	0.1069	0.2828	0.2302	0.1163	0.0999
	World	0	0.0536	0.1118	0	0.0289	0.1382

Table 2: Standard Deviation

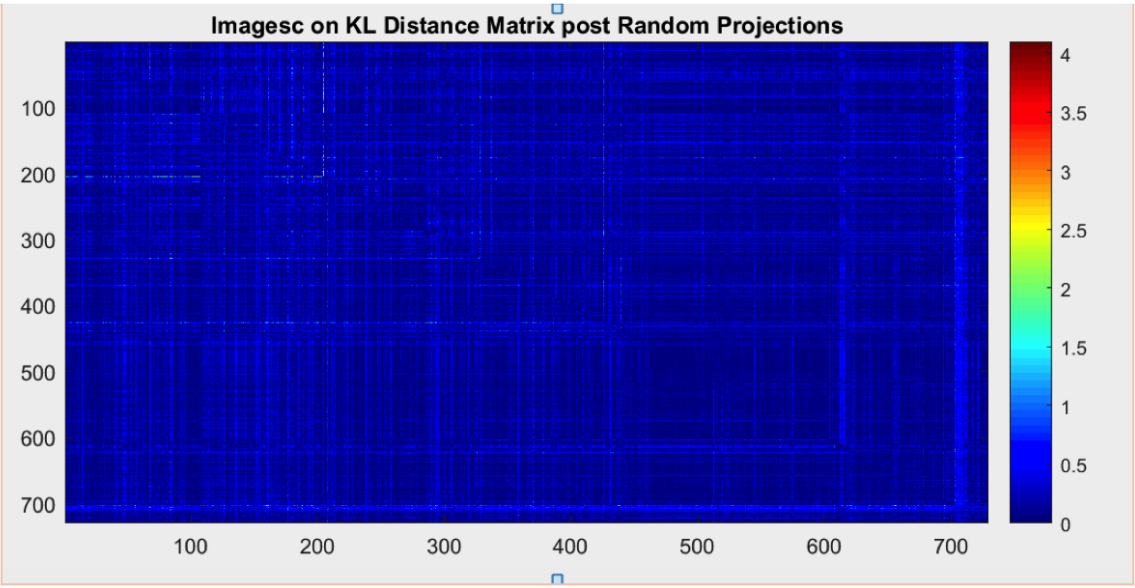

Figure 4: Imagesc on KL Distance Matrix post Random Projection

From the above figures we see that dimension reduction using Random Projection and classification using kNN classifier, does a fairly decent job giving us an accuracy of 60.76%. From the trace of the confusion matrix we observe that it classifies classical the best with

an accuracy of 98% but struggles to classify Jazz/Blues this can attributed to the fact that there are very few Jazz/Blues tracks. We also observed that the standard deviation for correct classification of genres is not large except for Jazz/blues and Rock/Pop. From the Imagesc plot we see that there is clear similarity amongst classical songs but not so much among other genres. It is the fastest method but the con of using random projections is that for every iteration we generate a new Gaussian random matrix which changes accuracies by ±10%. This is the only disadvantage of projecting the datapoints to a random subspace.

8.3.2 PCA

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	1.0000	0.0551	0	0	0	0
	Electronic	0	0.8550	0.4267	0.0200	0	0
	Jazz/Blues	0	0.0174	0.0733	0	0	0
	Rock/Pop	0	0.0648	0.4600	0.8289	0.0376	0
	Metal/Punk	0	0.0077	0.0400	0.1511	0.9450	0.1609
	World	0	0	0	0	0.0174	0.8391
Accuracy	75.69						

Table 3:Mean PCA

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	0	0.0657	0	0	0	0
	Electronic	0	0.0662	0.1738	0.0447	0	0
	Jazz/Blues	0	0.0247	0.1011	0	0	0
	Rock/Pop	0	0.0307	0.0894	0.1613	0.0360	0
	Metal/Punk	0	0.0172	0.0894	0.1469	0.0342	0.0745
	World	0	0	0	0	0.0389	0.0745

Table 4: Standard Deviation PCA

From the confusion matrices shown above we observe that the combination of PCA with kNN classifier does a pretty good job by having a normalized accuracy of 75.69%. We see that for all five iterations of cross validation accuracy of classical is 100% while the algorithm still struggles to classify Jazz/Blues. Also from the standard deviation confusion matrix we observe that the deviation in the subsequent iterations is fairly low as compared to random projections. Even though PCA is randomized technique still the accuracies over subsequent iterations of the algorithm do not change much since the components with maximal variance in the remain more or less same.

8.3.3 G1C

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	0.9568	0.0355	0.0571	0	0.0479	0.4326
	Electronic	0	0.7770	0.0833	0	0.0859	0.1453
	Jazz/Blues	0	0.0087	0.6488	0	0.0087	0
	Rock/Pop	0	0.0261	0	0.5500	0.1322	0.0080
	Metal/Punk	0	0.1084	0.1286	0.4234	0.6900	0.1245
	World	0.0432	0.0442	0.0821	0.0267	0.0354	0.2896
Accuracy	65.20						

Table 4: Mean G1C

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	0.0101	0.0204	0.1278	0	0.0332	0.0700
	Electronic	0	0.0879	0.1179	0	0.0645	0.0427
	Jazz/Blues	0	0.0194	0.3459	0	0.0194	0
	Rock/Pop	0	0.0583	0	0.11133	0.1036	0.0179
	Metal/Punk	0	0.0470	0.2167	0.1236	0.1566	0.0725
	World	0.0101	0.0046	0.1260	0.0596	0.0336	0.0706

Table 5: Standard Deviation PCA

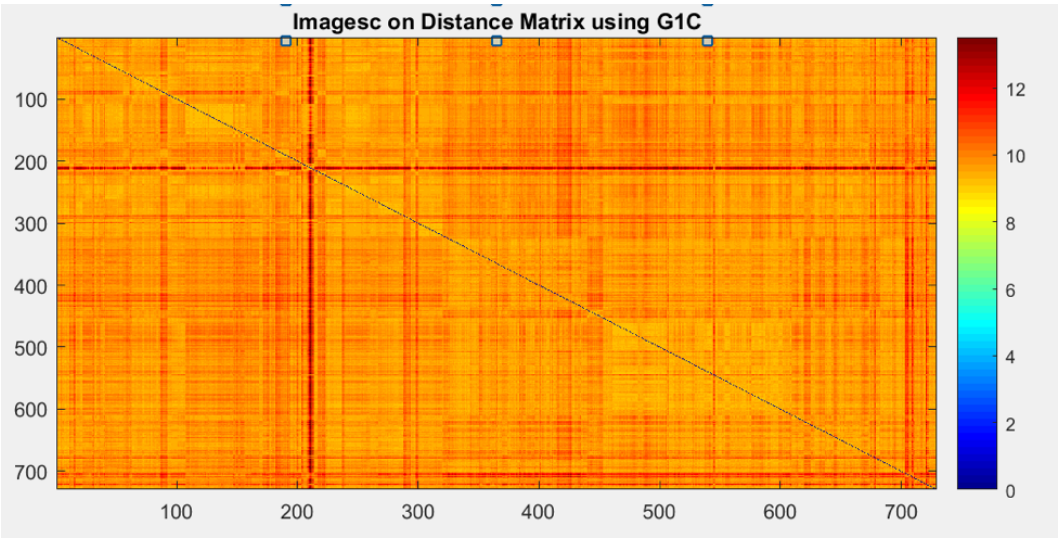

Figure 5: ImageSc on Distance matrix using G1C

In this approach we have used functions provided in the toolbox by Pampalk [4]. On multiple iterations also accuracies remain constant. Single Gaussian distribution method is fast and is very accurate as compared G30. Here we get a normalized accuracy of 65.2% we observe that the genre that is being classified with least success is World. We infer that this poor classification of World because the data being scattered. We also observed that majority of World tracks are being classified as Classical. This result is because the large no of Classical tracks.

8.3.4 FJLT

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	0.9490	0.1165	0.0400	0	0.0317	0.0602
	Electronic	0.0484	0.7413	0.2457	0.2129	0.0630	0.0267
	Jazz/Blues	0	0.0676	0.1686	0.0182	0.0348	0
	Rock/Pop	0	0.0169	0.3586	0.3992	0.0910	0.0129
	Metal/Punk	0.0026	0.0577	0.1871	0.3697	0.7795	0.1237
	World	0	0	0	0	0	0.7764
Accuracy	63.57						

Table 6: Mean FJLT

	Actual Genre						
Predicted Genre		Classical	Electronic	Jazz/Blues	Rock/Pop	Metal/Punk	World
	Classical	0.0323	0.0642	0.0894	0	0.0303	0.0491
	Electronic	0.0301	0.1130	0.1745	0.0728	0.0608	0.0254
	Jazz/Blues	0	0.0282	0.2051	0.0407	0.0478	0
	Rock/Pop	0	0.0235	0.3317	0.1766	0.0309	0.0289
	Metal/Punk	0.0058	0.0355	0.1796	0.1920	0.0807	0.0626
	World	0	0	0	0	0	0.0951

Table 7: Standard Deviation FJLT

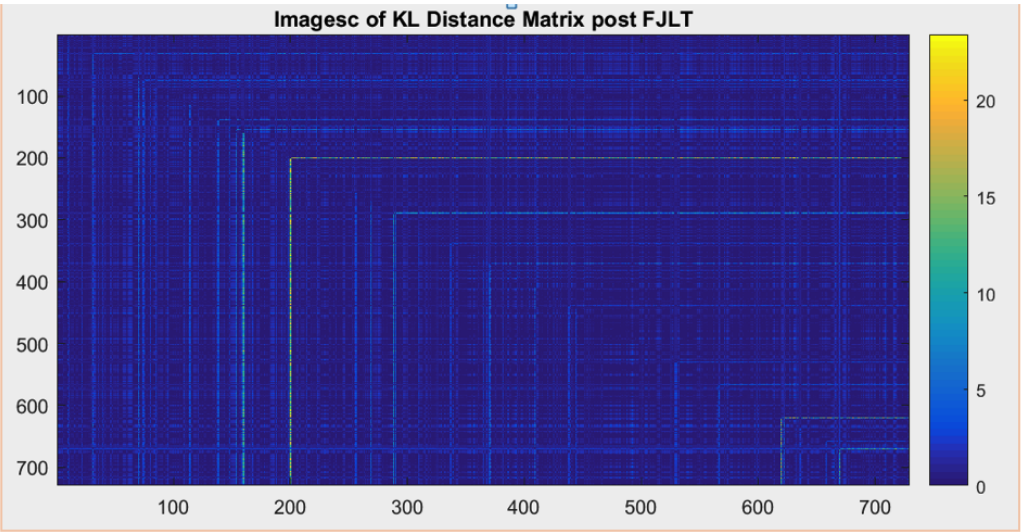

Figure 6: Imagesc on KL distance matrix post FJLT

The combination of FJLT with the kNN classifier is as accurate as any of the above approaches. FJLT is also a technique of randomized projections but multiple iterations do

not show large change in accuracies as the transformation matrix is sparse. This approach gives us a fairly good normalized accuracy of 63.57% with poor success rate of classification of Jazz/Blues genre. FJLT is slow as compared to random projection but faster than single Gaussian model.

9 Discussion

Classification of songs into their genre shall be difficult even for humans then getting an accuracy of 70% using the approaches seems to task well performed. We concentrated more on dimension reduction techniques rather than trying classification techniques other than kNN. Our course in Search Engine and Higher Dimensional Datasets concentrated heavily on these dimension reduction techniques hence we used them the results were as shown:

9.1 Analysis of Results

From the confusion matrices of mean accuracies we see that PCA performs the best while random projections is least accurate. The reason behind poor performance of random projection technique might be that we project higher dimensional data to a Gaussian randomly generated subspace. Moreover, on subsequent iterations the mean accuracy for random projection changes because for every iteration we generate a new random Gaussian matrix. Similar is the case with dimension reduction using FJLT, which is faster and has higher accuracies due to the sparse nature of transformation matrix (PHD). PCA on the other hand finds the first k directions of maximum variance in the data with the hope that components with lower variance are irrelevant and represents noise. Since these directions shall remain more or less same for subsequent iterations we see that the mean accuracies for PCA remain constant. We also tried functions given in matoolbox[4] which provided a good basis for the accuracies of our algorithms.

Given additional time we would continue to investigate the use of Neural Networks, Graph Laplacian, Self-Organizing Maps, etc. on our data. We would also like to probe the success of binary-tree approach using SVM classifier.

10 Classification of Test Data

Initially we gave all 729 songs for training and tested the songs for their genre. But we observed that due to large number of classical and world tracks in the training data the kNN classifier almost invariably classified the songs into either classical or world. Whereas no song was classified into Jazz/Blues as it had only 26 tracks in the database. To avoid this we randomly selected 26 tracks from each of the 6 genres of 729 songs for training purpose and tested the 35 songs whose genres are unknown. The result of all four approaches are shown in the table below.

Random Projections	FJLT	G1C	PCA
Classical	Classical	Electronic	World
Classical	Classical	Classical	Classical
Classical	Classical	Classical	Classical
Metal/Punk	Metal/Punk	Metal/Punk	Electronic
Metal/Punk	Metal/Punk	Rock/Pop	Rock/Pop
Jazz/Blues	Jazz/Blues	Metal/Punk	Electronic
Jazz/Blues	Jazz/Blues	Electronic	World
Jazz/Blues	Metal/Punk	Rock/Pop	Metal/Punk
Classical	Classical	Classical	Jazz/Blues
World	Electronic	Electronic	Metal/Punk
Classical	Classical	World	Classical
Rock/Pop	Rock/Pop	Jazz/Blues	Metal/Punk
World	Rock/Pop	World	Classical
Classical	Classical	Classical	Classical
Jazz/Blues	Classical	Jazz/Blues	Electronic
World	Metal/Punk	World	Classical
Classical	Classical	World	Electronic
Rock/Pop	Classical	Electronic	Classical
Classical	Classical	Classical	Classical
Classical	Classical	Classical	Classical
Rock/Pop	Electronic	Jazz/Blues	Metal/Punk
World	Electronic	Classical	Jazz/Blues
Rock/Pop	Jazz/Blues	Jazz/Blues	Rock/Pop
Classical	Classical	World	Jazz/Blues
Jazz/Blues	Rock/Pop	World	Classical
Classical	Classical	Classical	Classical
Metal/Punk	Metal/Punk	Electronic	Rock/Pop
Metal/Punk	Metal/Punk	Metal/Punk	Electronic
Metal/Punk	Metal/Punk	Metal/Punk	Metal/Punk
Classical	World	World	Classical
Classical	Classical	Jazz/Blues	Classical
World	World	World	Metal/Punk
Metal/Punk	Metal/Punk	Electronic	Electronic
Metal/Punk	Metal/Punk	Electronic	Electronic
World	World	World	Electronic

Table 8: Classification Result of Test Data

References

[1] Beth Logan, *Mel Frequency Cepstral Coefficients for Music Modeling*. The Cambridge Research Laboratory, Compaq Computer Corporation.

[2] Adam Berenzweig, Beth Logan, Daniel P.W. Ellis and Brian Whitman *A Large-Scale Evaluation of Acoustic and Subjective Music- Similarity Measures*. Columbia University, New York, New York 10027 USA.

[3] Elias Pampalk *Islands of Music Analysis, Organization, and Visualization of Music Archives*.

[4] *Elias Pampalk-MA Toolbox*. `//www.pampalk.at/ma/`.

[5] Jayme Garcia, Arnal Barbedo and Amauri Lopes *Automatic Genre Classification of Musical Signals*.

[6] Nir Ailon And Bernard Chazelle: *The fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors* `https://www.cs.princeton.edu/ chazelle/pubs/FJLT-sicomp09.pdf`

[7] Sachin Mylavarapu, Ata Kaban *Random projections versus random selection of features for classification of high dimensional data*. `http://www.cs.bham.ac.uk/ axk/Sachin_ukci13.pdf`