Digital Signal Processing
Mini Project Report

Song Genre Classification

Submitted to:
Dr. Raghavendra Bobbi

Submitted by:

Chetan Giridhar Vashisht
12EC31

Sharang Kulkarni
12EC85

Dhulipati Krishna Harshith
12EC37

ABSTRACT

In this project, we investigate the impact of machine learning algorithms in the development of automatic music classification models aiming to capture genres distinctions. The study of genres as bodies of musical items aggregated according to subjective and local criteria requires corresponding inductive models of such a notion.

This process can be thus modeled as an example-driven learning task. We use the concept of Mel-Frequency Cepstrum Coefficients to map songs to a less complicated representation.

TABLE OF CONTENTS

1. Introduction

Song recognition has always been a challenging task. a lot of research has been put into the field and a lot of fruitful results can be found. We attempt to classify songs based on their genre. Each genre of songs has a characteristic set of traits, and in this project we attempt to exploit this in order to classify the songs. The classification is into distinct genres. The ten generes selected are Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae and Rock.

2. Approach

In order to classify the songs into their respective generes, representative Mel Frequency Cepstral Coefficients(MFCCs) of each song are extracted. The result is passed to the machine learning algorithm "K-Nearest Neighbours".

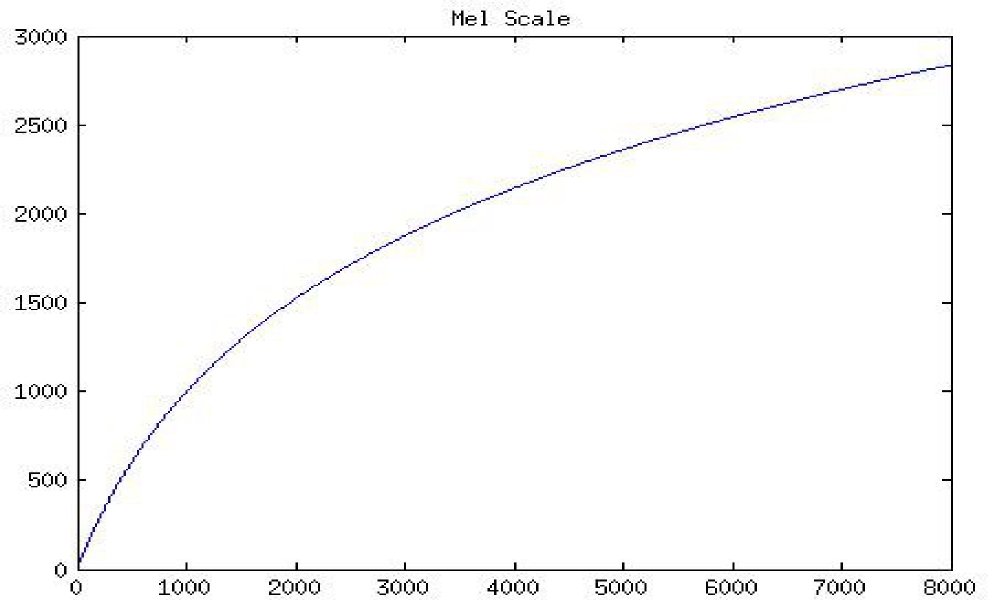
3. Data Aquisition

In order to obtain enough songs of a given genre, we downloaded from "Gaztan's Music Collection". The package consists one hundred songs of each of ten different genres. Each song is of length thirty seconds. A very good collection of songs and it can be used in the future for other projects as well.
We are using all of the available songs collection, 100 songs of each genre and from 10 distinct genres. 70% of the songs are used to train the machine learing algorithms and the reamining 30% is used in testing of the data. 700 songs for training and 300 songs involved in testing.

4. Mel Frequency Cepstrum Coefficients

4.1 Mel Scale

Our ears are more sensitive to frequencies below 1kHz and less sensitive to the fequencies above the same. As an approximation, our reception is linear below 1kHz and logrithamic above that. This is termed as the Mel Scale. In order to obtain the correct coefficients, it is necessary to convert to the Mel scale. Hence the Mel scale is a strong way to decide the genre of each song.



4.2 Frame blocking

The input speech signal is segmented into frames of 20ms with optional overlap of 1/3~1/2 of the frame size. Usually the frame size (in terms of sample points) is equal to power of two in order to facilitate the use of FFT. If this is not the case, we need to do zero padding to the nearest length of power of two. If the sample rate is 16 kHz and the frame size is 320 sample points, then the frame duration is 320/16000 = 0.02 sec = 20 ms. Additional, if the overlap is 160 points, then the frame rate is 16000/(320-160) = 100 frames per second.

4.3 Sampling

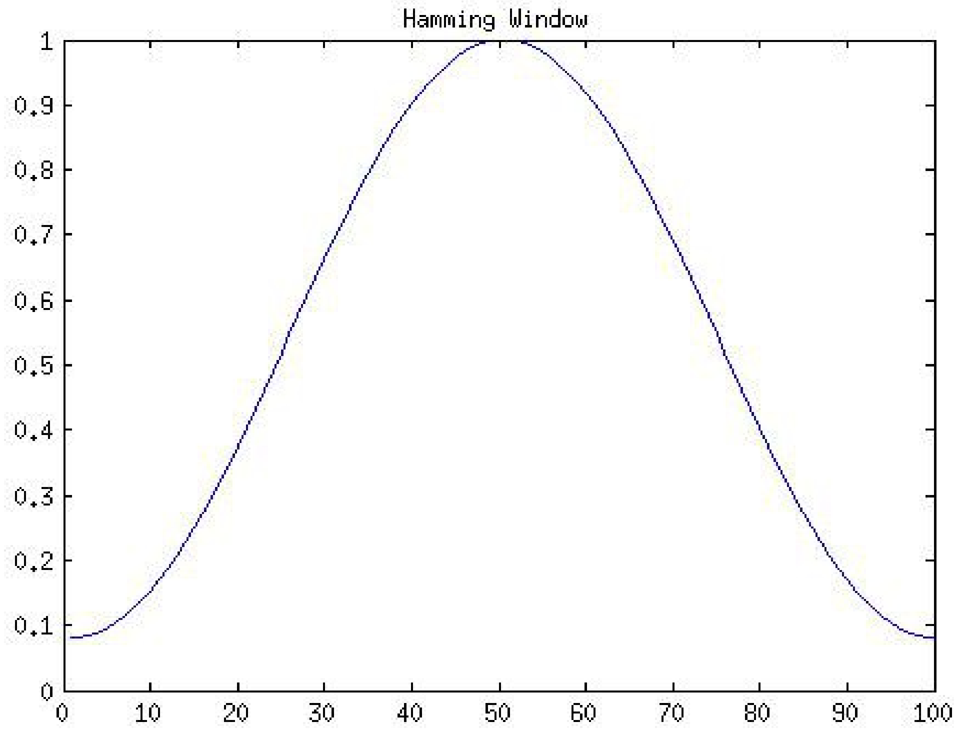
The songs obtained from the collection each has a length of thirty seconds. We aim to sample the middle fifteen seconds. This gives us the best representation of the song. The whole interval is sampled into twenty milisecond intervals. We operate on each of these twenty milisecond intervals. The MFCCs of each of these is generated. This process returns twenty fundamental frequencies on the Mel scale. The last five frequencies are highly unimportant, so they are rejected.

This leaves us with fifteen Mel Frequency coefficients. The MFCCs of every single interval is collected in an array. This is now a 7500 cross 15 two dimentional array. The average and covariance matrix of both are calculated. The covariance matrix yields a fifteen by fifteen 2D matrix, this is instead stored as a one by 225 row matrix. The mean is concatinated in the begining of the array thus returning a 240 by one row matrix for each song.

4.4 Hamming Window

For each 20ms time frame, we have to generate a hamming window to smoothen the edges with the approprite size of the window and use the proper transfer function to make sure that no data is lost. The inbuilt MTALAB function hamming(.) is used to generate the hamming window. The following is a typical hamming window of length N and parameter alpha:

h = (1-alpha)-alpha*cos(2*pi*n/(N-1))

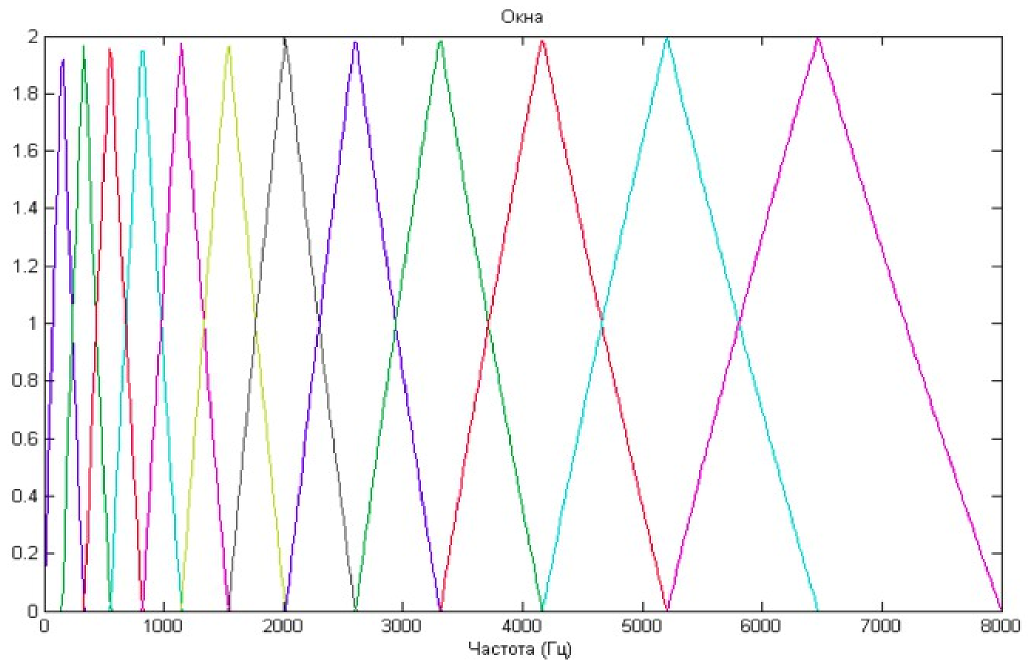


4.5 Discrete Fourier Transform(FFT)

Spectral analysis shows that different timbres in speech signals corresponds to different energy distribution over frequencies. Therefore we usually perform FFT to obtain the magnitude frequency response of each frame. When we perform FFT on a frame, we assume that the signal within a frame is periodic, and continuous when wrapping around. Hence we multiply each frame by a Hamming window to increase its continuity at the first and last points.

We discrete fourier transform of the signal using the inbuilt MATLAB function fft(.). We require only the magnitude spectrum.

4.6 Generating Triangular Bandpass Filters



Mel-frequency is proportional to the logarithm of the linear frequency, reflecting similar effects in the human's subjective aural perception. We multiply the magnitude frequency response by a set of 20 triangular bandpass filters to get the log energy of each triangular bandpass filter. The positions of these filters are equally spaced along the Mel frequency, which is related to the common linear frequency f by the following equation:

melFreq=1125*log(1+linFreq/700)

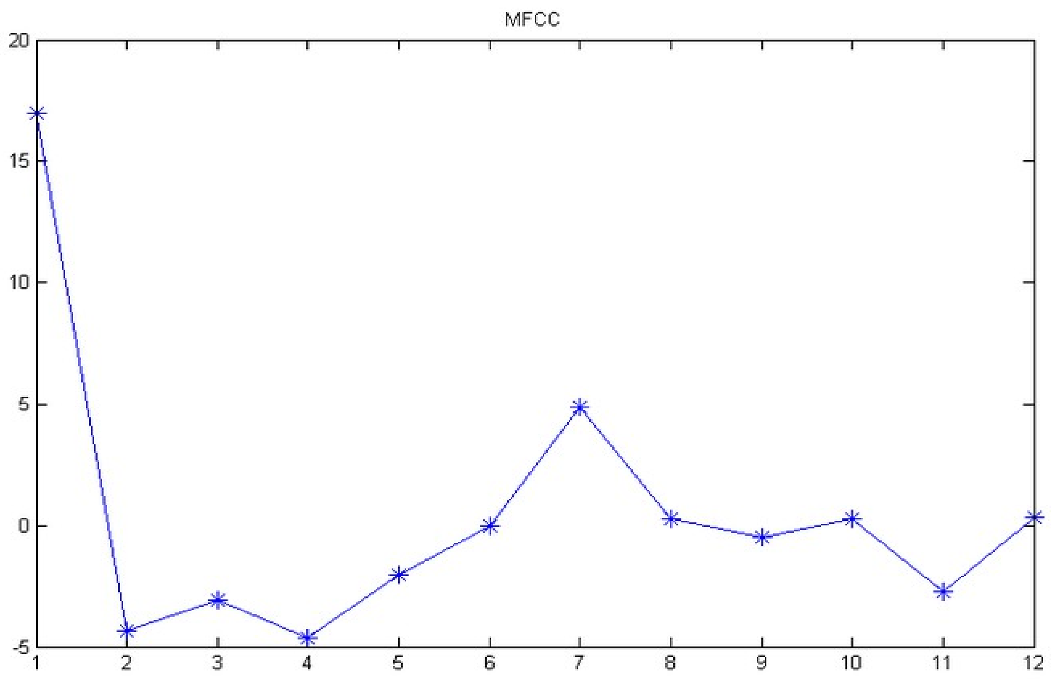
The mel scale is converted back to the linear scale using the following relation

linFreq=700*(exp(melFreq/1125)-1)

The following line shows how the 'i'th Triangular filters are generated. filterNum reprents the number of desired filters (20 in this case), fLow represents the lower frequency limit (0 Hz) and fHigh represents the higher frequency limit(fs/2).

f(i) = mel2linFreq(lin2melFreq(fLow) + (i-1)*(lin2melFreq(fHigh)-lin2melFreq(fLow))/(filterNum+1))

4.7 Discrete Cosine Transform (DCT)



In this step, we apply DCT on the Ek obtained from the triangular bandpass filters to have L (here 20) mel-scale cepstral coefficients. The formula for DCT is shown next.

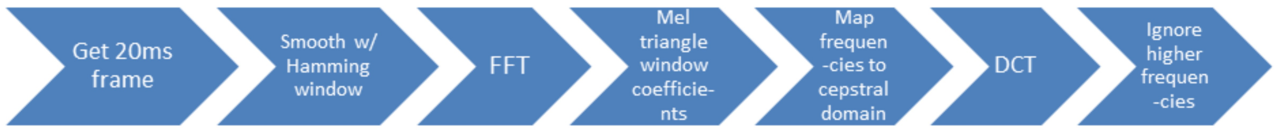
Cm = S(k-1,N)*cos[m*(k-0.5)*p/N]*Ek, m=1,2, ..., L

where N is the number of triangular bandpass filters, L is the number of mel-scale cepstral coefficients. Since we have performed FFT, DCT transforms the frequency domain into a time-like domain called quefrency domain. The obtained features are similar to cepstrum, thus it is referred to as the mel-scale cepstral coefficients, or MFCC. The discrete cosine transform is completed using the inbuilt MATLAB function DCT(.).

Average and Covariance:

The output vector of the DCT step has 20 elements. The last five of them hardly contribute. Hence they are ignored. We store only the first fifteen of them. The MFCC vectors of all the intervals are collected into one 2D vector and it is a 7500 cross 15 vector, as mentioned earlier. We take the average and covariance matrix of this vector collection. The mean is 1 cross 15 rectangular vector and the covariance matrix is a 15 cross 15 symmetric square matrix. The dollowing must be converted into a format which the machine learing algorithm finds comportable to read. So we change it to a 240 cross 1 row matrix. Hence we obtain the MFCC vector for a song.

The following MFCC code has been written as a function. The input to it is a song and the output is a 240 cross 1 row vector containing the average and covariance matrix of the MFCCs generated.


Get 20ms frame
Smooth w/ Hamming window
FFT
Mel triangle window coefficie-nts
Map frequen -cies to cepstral domain
DCT
Ignore higher frequen -cies

5. Classification Technique

5.1 Kullback-Liebler (KL) Divergence

The fundamental calculation in our k-NN training is to figure out the distance between two songs. We compute this via the Kullback-Leibler divergence. Consider p(x) and q(x) to be the two multivariate Gaussian distributions with mean and covariance corresponding to those derived from the MFCC matrix for each song. Then, we have the following:

$$2KL(p||q) = \log \frac{|\Sigma_q|}{|\Sigma_p|} + Tr(\Sigma_q^{-1}\Sigma_p) + (\mu_p - \mu_q)^T \Sigma_q^{-1}(\mu_p - \mu_q) - d$$

However, since KL divergence is not symmetric but the distance should be symmetric, we have:

$$D_{KL}(p, q) = KL(p||q) + KL(q||p)$$

5.2 k-Nearest Neighbors (k-NN)

The machine learning technique we applied is the k-nearest neighbors (k-NN) because existing literature has shown it is effective considering its ease of implementation.
In k-NN classification, 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)

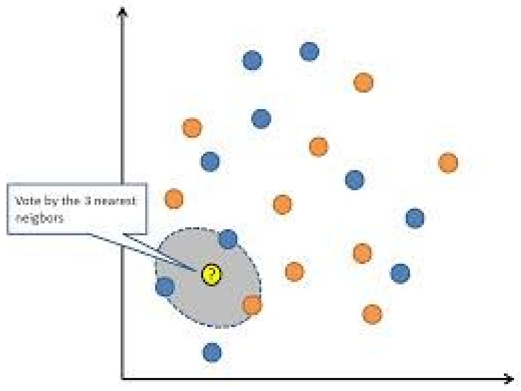


Illustration of k-nearest neighbours

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct.
For each song in the test data set:
1. Find the KL divergence of the test song from every song in the training set. Store the training label of the training song and the KL divergence in an array.
2. Sort the array in ascending order, so that the nearest neighbours are indexed earlier.
3. Select the first 'k' rows from the array and place it in a new array.
4. Find the count of each label in the new array.
5. The label with the highest count is the label which will be assigned to the test song.

5.3 Cross-Validation:

A 4-fold cross-validation model is implemented along with the k-NN method to obtain the value of 'k' which produces the best performance.
The model returned the value 10 as the optimal value.

6. RESULTS

Actual ->	Blues	Classical	Country	Disco	Hiphop	Jazz	Metal	Pop	Reggae	Rock
Predicted										
Blues	30	0	0	0	0	0	0	0	0	0
Classical	0	30	0	0	0	0	0	0	0	0
Country	0	0	30	7	0	0	0	0	0	0
Disco	0	0	0	23	0	0	2	0	0	0
Hiphop	0	0	0	0	30	0	0	0	0	0
Jazz	0	0	0	0	0	30	0	0	0	0
Metal	0	0	0	0	0	0	20	0	0	0
Pop	0	0	0	0	0	0	8	30	2	0
Reggae	0	0	0	0	0	0	0	0	28	0
Rock	0	0	0	0	0	0	0	0	0	30
Accuracy	100%	100%	100%	76%	100%	100%	66%	100%	93%	100%

Primary analysis:

Out of 10 genres, the songs from 7 genres were perfectly predicted by our k-nearest neighbours algorithm. On the whole, the system has an accuracy of 93%, which is commendable.

Secondary analysis:

Disco has been misclassified as a 'Country' song 7 times out of 30, and Metal has been misclassified as a 'Pop' song 8 times out of 30. This can be attributed to a possible similarities in the MFCC coefficients produced by the respective genres. These misclassifications represent a fallacy in the k-NN system.

7. FUTURE WORK

We plan to implement more complicated Machine Learning algorithms such as K-Means Clustering and Neural Networks on the dataset. The problem with these algorithms is that they get stuck in local optima and cannot reach the global maxima. The lazy algorithm k-nearest neighbours seemed to give satisfying results. We plan to combat the local optima problem by considering multiple representative points for each genre to improve the learning.

8. REFERENCES

[1] Project based on a paper “Music Genre Classification” by Michael Haggblade, Yang Hong and Kenny Kao.

[2] Help to generate MFCCs taken from here:
http://mirlab.org/jang/books/audiosignalprocessing/speechFeatureMfcc.asp?title=12-2%20MFCC

[3] Images from http://sysmagazine.com/posts/140828/

[4] Description to understand MFCCs at www.wikipedia.com/Mel-frequency_cepstrum

[5] www.stackoverflow.com/

[6] “Classification of Musical Genre: A Machine Learning Approach” by Roberto Basili, Alfredo Serafini, Armando Stellato.

[7] “A Comparative Study on Content-Based Music Genre Classification” by Tao Li, Mitsunori Ogihara and Qi Li.