

University of Burgundy

Masters in Computer Vision and Robotics

SSI Pattern Recognition Project Report

AUDIO GENRE CLASSIFICATION

By

Vamshi Kodipaka
Bhargav Shah
Hardik sinh Parmar

Supervisor:

Dr. Desire SIDIBE

29th May 2019



CONTENTS

1. Introduction
2. Problem Statement
3. Audio Understanding: Pipeline
4. Basic of Audio Processing In MATLAB
5. PROJECT-1: MFCC Feature Extraction
 - a. What is Mel-Frequency Cepstrum and MFC Coefficients
 - b. MFCC Features- Extraction:: Matlab inbuilt
 - c. Math behind MFCC :: Three types of Spectral Analysis
 - i. Conventional Cepstral Method
 - ii. Linear Predictive Coding based
 - iii. Mel-Frequency Cepstral based
5. PROJECT-1: Feature Extraction for 4 Music files and 729 music files (Outputs)
6. Difficulty of Dimension Reductionality in Project-1
7. PROJECT-2: Music Genre Classification with GTZAN Dataset
 - :: With all Pipeline Implementation
9. Future Work(To-do-list)
10. References

INTRODUCTION

Musical genres are categorical descriptions with definite patterns that are used to describe audio files. 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 since years. Google says, Audio Classification can be used for audio scene understanding which in turn is important so that an artificial agent is able to understand and better interact with its environment.

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 project, we will explore different methods for music classification. Specifically, we explore the **classification of music** according to its genre. Problems we encountered during the project and modifications in the project for the project understanding were explained. Results obtained from each method and the comparison of these methods will be included.

Many algorithms in machine learning works on image processing, similarly we find the need of music classification in-order to distinguish one music file to the other and **categorize the new test file** (if given) into which of the trained music class it belongs to.

PROBLEM STATEMENT

There was ISMIR2004 Audio Description Contest[1] (Genre/ Artist ID Classification and Artist Similarity) in which there is a sub-contest Genre Classification. It is the 5th International Conference on Music Information Retrieval. In 2018, this project was given to LAAS-CNRS Team[2] to work on this dataset for Music Genre Classification. So, we take this as this challenge as our project on Pattern Recognition this Semester.

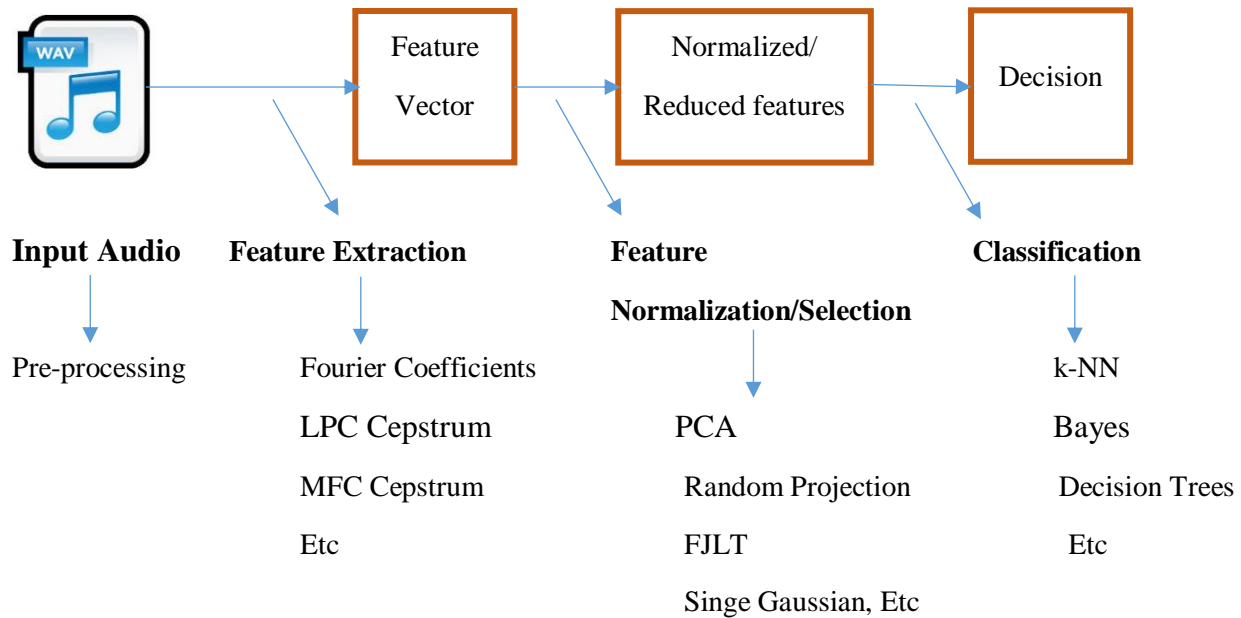
The dataset that we have chosen/observed in this contest has 729 tracks of the genres Classical (320), Electronic(114), Jazz/Blues(26), Metal/Punk(45), Rock/pop(102), World(122). Where each of the above songs is sampled at 11025 Hz mono.

We have the above dataset (<https://zenodo.org/record/1302992#.XHHZC6JKjIV>) for training and testing a genre classification algorithm so that we will be 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.

Any Machine Learning algorithm has 3 implementation steps:

1. Feature Extraction
2. Feature Selection/Normalization
3. Feature Classification

AUDIO UNDERSTANDING: PIPELINE



Idea for .mp3 to .wav conversions:

We have .mp3 files and we can convert to .wav files. Converting to wave files can be done reducing the sampling rate and changing bit rate.

We either use .mp3 or .wav file as per MATLAB compatibility.

(The commands related to these conversions are explained in a 'a1.txt' file)

Since we have 729 audio mp3 files, we first take one .mp3 file and try to extract features for it. Then we will try to display the extracted features as interest points. Then we will try to take for whole music dataset.

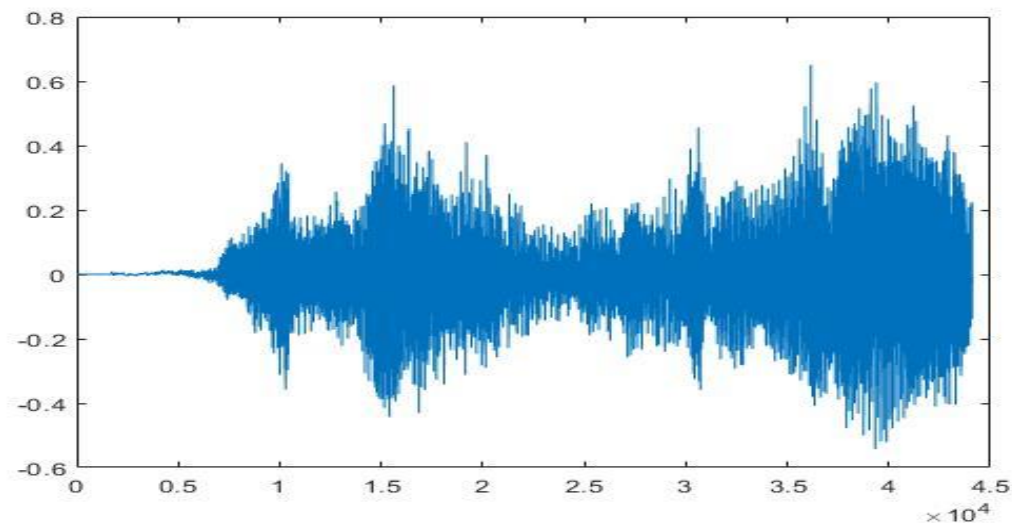
BASICS OF AUDIO PROCESSING

1. To load .mp3 file: `[y,Fs]=audioread('artist_1_album_1_track_1.mp3')`

Here, y gives amplitude and Fs gives frequency.

2. Plot .mp3 with amplitude (vs) frequency:

`plot(y(1:44100,1))`



3. Viewing the information of the .mp3 file:

```
info=audioinfo('artist_1_album_1_track_1.mp3')
```

```
>> info=audioinfo('artist_1_album_1_track_1.mp3')

info =

  struct with fields:

    Filename: 'D:\AudioGenreClassifier-master\AudioGenreClassifier-master\artist_1_album_1_track_1.mp3'
    CompressionMethod: 'MP3'
    NumChannels: 2
    SampleRate: 44100
    TotalSamples: 1687360
    Duration: 38.2621
    Title: []
    Comment: []
    Artist: []
    BitRate: 128
```

4. To play .mp3 using sound command:

```
sound(y(1:441000),1,Fs)
```

Workspace	
Name ▲	Value
Fs	44100
y	1686000x2 double

5. To play .mp3 using audioplayer object in MATLAB:

`p =audioplayer(y,Fs)`
`play(p) ::` plays loaded music fil
`pause(p) ::` pauses music file
`stop(p) ::` stops music file
`start(p) ::` starts from point of stop
`clear(p) ::` clears 'p' object's music file.

Note:

We can plot Amplitude vs frequency graphs for various type of music files (like rock, pop, instrumental etc.) among the data set. But we don't know which music file is of which type/class among them for now. And our task is to classify. But before that, we have to extract features.

Note: All the basic commands related to audio operations are attached in a **a1.txt** file along with this pdf.

```
Command Window
>> p=audioplayer(y,Fs)

p =

audioplayer with properties:

    SampleRate: 44100
    BitsPerSample: 16
    NumberOfChannels: 2
    DeviceID: -1
    CurrentSample: 1
    TotalSamples: 1686000
    Running: 'off'
    StartFcn: []
    StopFcn: []
    TimerFcn: []
    TimerPeriod: 0.0500
    Tag: ''
    UserData: []
    Type: 'audioplayer'

>> play(p)
>> pause(p)
>> stop(p)
fx >> |
```

FEATURE EXTRACTION: MFCC

a. What is Mel-Frequency Cepstrum and MFC Coefficients?

Mel refers to 'Melody'. "Cepstrum" means the Inverse Fourier transform (IFT) of the logarithm of the estimated spectrum of a signal.

Definition: In sound processing, the **Mel-Frequency Cepstrum (MFC)** is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

History of MFCC:

Paul Mermelstein is typically credited with the development of the MFC. Mermelstein credits Bridle and Brown for the idea:

Bridle and Brown used a set of 19 weighted spectrum-shape coefficients given by the cosine transform of the outputs of a set of non-uniformly spaced bandpass filters. The filter spacing is chosen to be logarithmic above 1 kHz and the filter bandwidths are increased there as well. We will, therefore, call these the mel-based cepstral parameters.

Davis and Mermelstein have commented that the spectral basis functions of the cosine transform in the MFC are very similar to the principal components of the log spectra, which were applied to speech representation and recognition much earlier by Pols and his colleagues. (Source: Wikipedia)

The difference between the Cepstrum and the mel-frequency cepstrum:

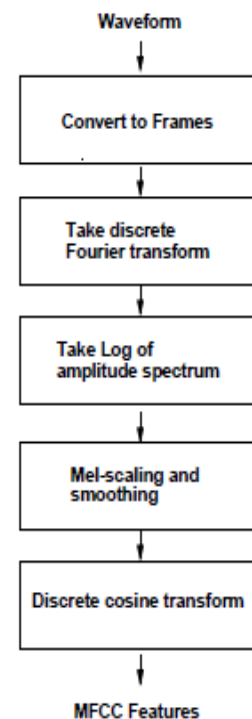
In the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

MFCCs are commonly derived as follows:

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

There can be variations on this process, for example: differences in the shape or spacing of the windows used to map the scale, or addition of dynamics features such as "delta" and "delta-delta" (first- and second-order frame-to-frame difference) coefficients.

Also, the European Telecommunications Standards Institute in the early 2000s defined a standardized MFCC algorithm to be used in mobile phones.



Process to create MFCC features

Noise Sensitivity:

MFCC values are not very robust in the presence of additive noise, and so it is common to normalize their values in speech recognition systems to lessen the influence of noise. Some researchers propose modifications to the basic MFCC algorithm to improve robustness, such as by raising the log-mel-amplitudes to a suitable power (around 2 or 3) before taking the DCT (Discrete Cosine Transform), which reduces the influence of low-energy components

Use: Mel-Frequency Cepstral Coefficients are the standard preprocessing technique in Speech Processing and developed for automatic speech recognition and proven to be most useful for music information retrieval tool.

Preprocessing:

Sonograms are similar to MFCC. But they are based on psychoacoustic models which are slightly more complex than those used in MFCC. So we use MFCC as feature extraction. It is a non-linear frequency scale. The important aspects of MFCC model are:

1. Non-linear perception of loudness in decibels
2. To some extent special masking effects using Discrete Cosine Transform
(Here, DCT is substitute of IFT)

b. MFCC Feature Extraction:

This method captures overall spectral shape, which carries all the important about the the instrumentation of its timbres, the quality of the singers voice and production effects. It drastically reduces amount of data for the computational model of music similarities

Working with 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. We then 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.

Finally we apply discrete Fourier transform and hence achieve MFCC features. We have chosen number of cepstral coefficients to be 10 in order to reduce number of datapoints and hence the complexity.

Calculation of Mel-Frequency:

Note: Mel-Scale is approximately linear for low-frequency ($f < 500\text{Hz}$) and logarithmic for high frequencies.

$$M(\text{mel_freq}) = 1127 * \log(1 + f/700)$$

where f is frequency in linear scale
and M is frequency in mel scale.

This modulation of log acts as a weight vector like in
($y = w^T X - t$) in a classical regression model

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 have a $10 \times L$ matrix, where L depends on the length of song. L on average was found to be 3000. So we have reduced the data points from about a million to about 10×3000)

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.

We used Matlab's inbuilt MFCC functions in MA Toolbox (Malcolm Slaney's Auditory Toolbox) for Feature Extraction.

- a. `ma_mfcc.m`
- b. `mfcc_coeff.m`

So, these functions does the required operations shown in this flow-chart (Process to create MFCC).

c. Math Behind MFCC Feature Extraction:

Origin of mfcc has have two primitive analysis: CSA and LPCC and then we see MFCC. Let $x[n]$ be the input signal (complex spectrum),

i. From conventional spectral analysis:

Then we can say $C[n]$ real cepstrum of the signal is:

$$C[n] = \frac{x[n] + x[-n]}{2}$$

As we said earlier, real cepstrum of a signal $x[n]$ and

Calculating Cepstral Coefficients of $C[n]$ cepstrum:

=====*****=====

$$\text{Inverse FT : } C(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log(|X(e^{jw})|) e^{jwn} dw$$

$$C(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log(|X(e^{jw})|) \cdot [\cos(jwn) + \sin(jwn)] dw$$

Since odd parts: sin terms becomes zero is 0

Therefore,

$$C(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log(|X(e^{jw})|) \cdot \cos(jwn) dw$$

Now:
$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log(|X(e^{jw})|) e^{jwn} dw \quad \text{----- (eq. 1)}$$

We know that, if is a complex signal: $x = a + jb$;

$$\mathbf{x} = \mathbf{a} + \mathbf{j}\mathbf{b}; = |\mathbf{x}| \cdot (e^{j\theta})$$

Then x[n] in eq.1 becomes: $x[n] = \log(|X(e^{jw})|) + \arg(X(e^{jw}))$

$$\mathbf{x}[\mathbf{n}] = \frac{1}{2\pi} \int_{-\pi}^{\pi} [\mathbf{log}(|X(e^{jw})|) + \mathbf{j} \mathbf{arg}(X(e^{jw}))] \cdot [\mathbf{cos}(jwn) + \mathbf{sin}(jwn)] d\mathbf{w}$$

Positive cepstrum:

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log(|X(e^{jw})|) \cdot [\cos(jwn)] dw - \frac{1}{2\pi} \int_{-\pi}^{\pi} j \arg(X(e^{jw})) \cdot [\sin(jwn)] dw$$

Negative cepstrum:

$$x[-n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log(|X(e^{jw})|) \cdot [\cos(jwn)] dw + \frac{1}{2\pi} \int_{-\pi}^{\pi} j \arg(X(e^{jw})) \cdot [\sin(jwn)] dw$$

Adding x[n] and x[-n] gives C[n] as mentioned initially

=====*****=====

ii. Linear Predictive Coding Cepstrum(LPC Cpestrum):

Let $x[n]$ is a signal, $a_0, a_1, a_2, \dots, a_p$ are the LPC coefficients of p th order, then DFT of $x[n]$, we get LPC Spectrum----- (Statement-1)

If instead of DFT of signal, if we use signal $x[n]$ and take the DFT, I will get $x[k]$ which is called a real signal spectrum and Statement-1 gives us the LPC Cepstrum.

Calculating Cepstral Coefficients of LPC Cepstrum:

To calculate c_0, c_1, \dots, c_p (cepstral coefficients), we have:

The LPC vector is defined by $[a_0, a_1, a_2, \dots, a_p]$ and the CC vector is defined by $[c_0, c_1, c_2, \dots, c_{p-1}]$

LPC Cepstrum (c_m)	
$c_0 = \log G^2$	
$c_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \quad 1 \leq m \leq p$	$G = e^{c_0/2}$
$c_m = \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \quad m > p$	$a_m = c_m - \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \quad 1 \leq m \leq p$

Here, G is LPC Gain =====*****=====

iii. Mel-Frequency-Cesptrum:

1. Compute FFT power spectrum of speech signal
2. Apply mel-space filter bank to the power spectrum to get energies
3. Compute DCT of log filter-bank energies to get uncorrelated MFCCs.

So, instead of taking whole spectrum we take 80 points of the spectrum and find out the cepstral coefficient. We can treat that is a signal pass through the inverse DFT and cepstral coefficient will be generated. So, information is reduced.

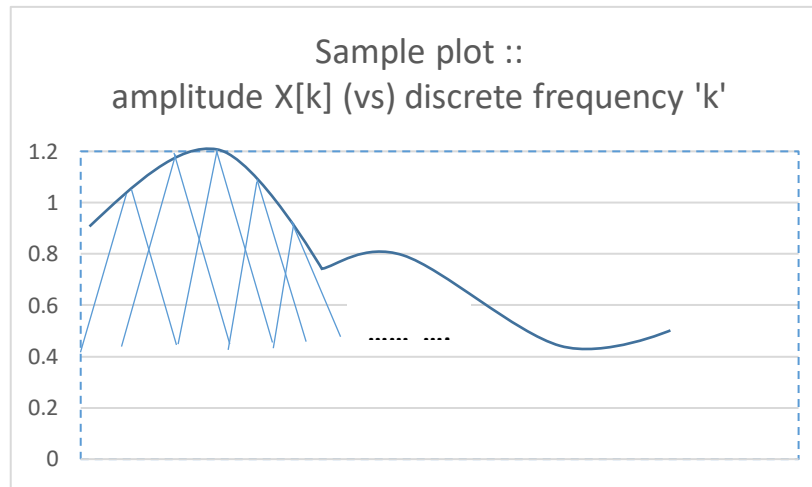
Now the problem is that if it is a linear filter this does not matched with human perception (human frequency perception)

Let $x[n]$ is framed through the entire signal ,

DFT

has a window size fixed

$$x[n] \rightarrow X[k] \quad \text{-- } |X[k]|$$



$$k = N/2; \quad f_{\max} > F_s/2; \quad F_s = 8\text{KHz}; \quad f_s \text{ becomes } 4\text{KHz}$$

$$\text{If triangular train filter} = 100\text{Hz} \rightarrow F_s = 4000/100 = 40\text{points}$$

Now suppose, this triangular filter is applied to 8KHz = 80points will be generated.

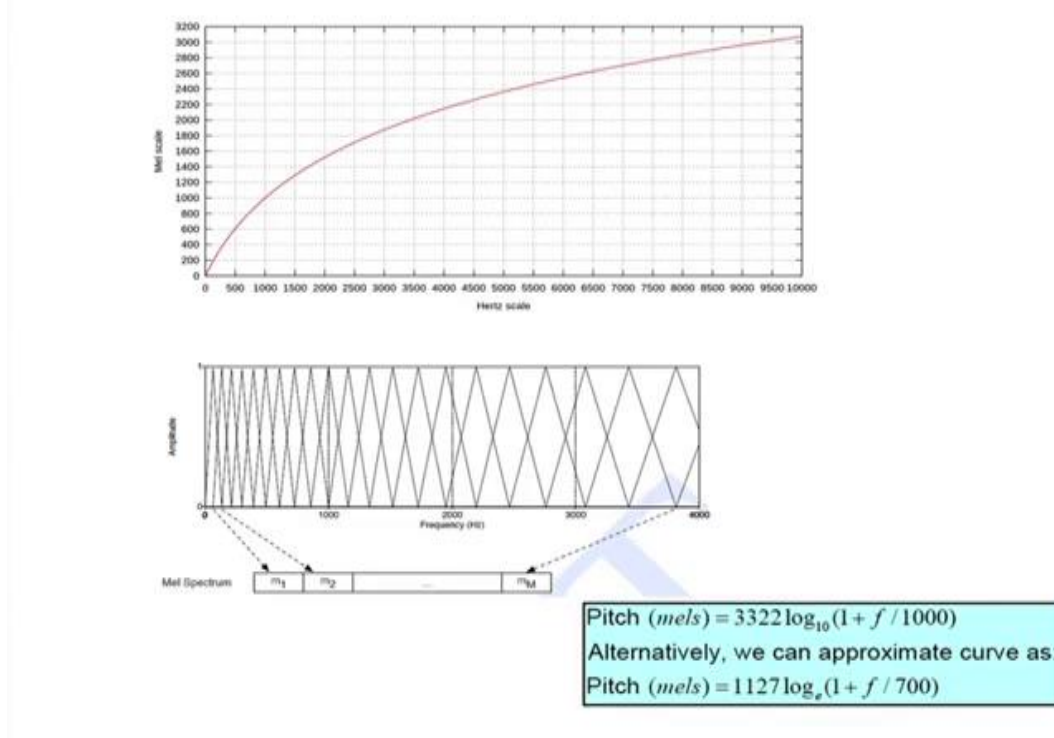
$$\text{To convert into } k \text{ to } f, \text{ we have: } f = 2\pi k/N;$$

We say, the signal the frequency perceived by human being is not in linear scale this is in Mel scale. So, instead of taking the linear filter now we convert the filter bandwidth as per the Mel scale what is the Mel scale you know this using this equation described previously. So, the bandwidth of the filter is depends on the Mel scale. So, instead of uniform bandwidth filter, we take Mel scale filter.

So, what is human perception in frequency lower frequency our resolution is very high; that means, lower frequency we can perceive linearly along the physical frequency, but at the higher frequency we have a some low resolution; that means, the band of frequency the frequency band we perceive as a same frequency is larger. So, we can say at the high frequency region the bandwidth of the filter will be large. So that means, who do not require that much of course, resolution. So, half estimation is sufficient for high frequency.

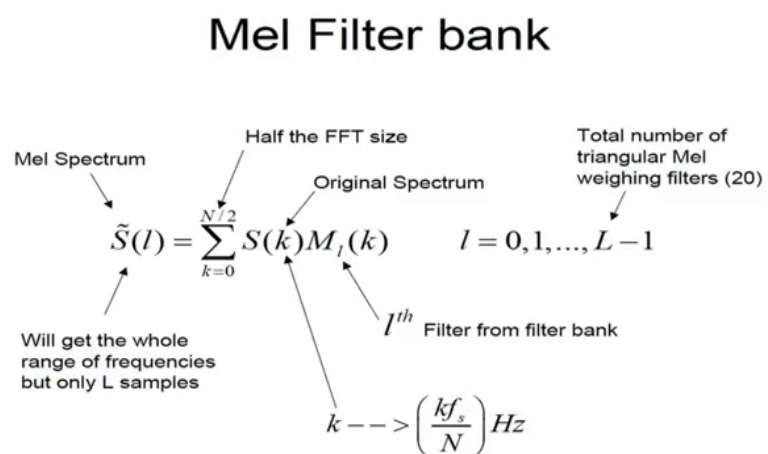
So, that is why we take the bandwidth of the filter will be larger. So, bandwidth defined by the Mel scale. So, if we take the Mel scale filter then we take the Mel scale filter if you find out the dividends 4 kilo hertz we take 20 filter which will be sufficient to cover the 4 kilohertz frequency or entire frequency range.

So, I can get $m_1, m_2 \dots m_{20}$ because every filter give me a single point bandwidth single bandwidth. So, every filter has a single bandwidth which is m_1, m_2, m_3, m_4 , we can find out 20 Mel point.



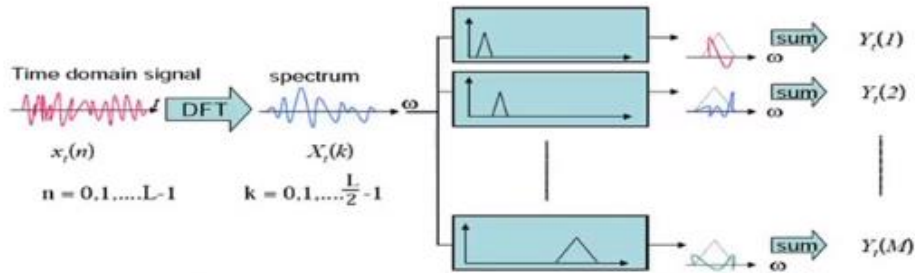
So, since filters are Mel scale filter are designed in Mel scale then we can say we have Locked the LP or the frequency spectra in Mel scale instead of hertz

So, what we get instead of hertz, we get Mel scale in here and here is let $X_{\text{cap}}[k]$ instead of $x[k]$. We take X_{cap} average(average energy). So, I instead of spectrum I get Mel scale spectrum once we get the Mel scale spectrum, if we analyze cepstral coefficient, then this is called Mel scale cepstral coefficient. So, since my spectrum is frequency warped in Mel scale that is why it is called Mel frequency cepstral coefficient.



So, what I am actually doing any mathematics I am designing the filters I design I will explain in the next class first I designed and described the equation.

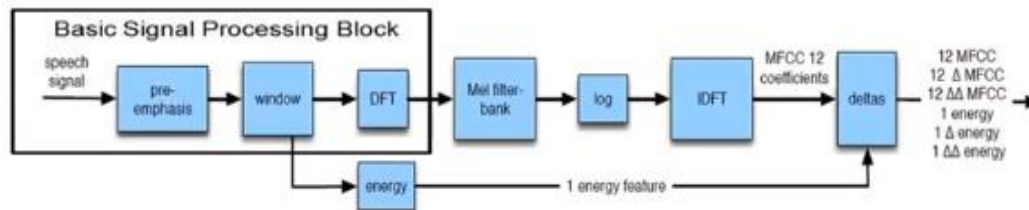
Mel Filter bank



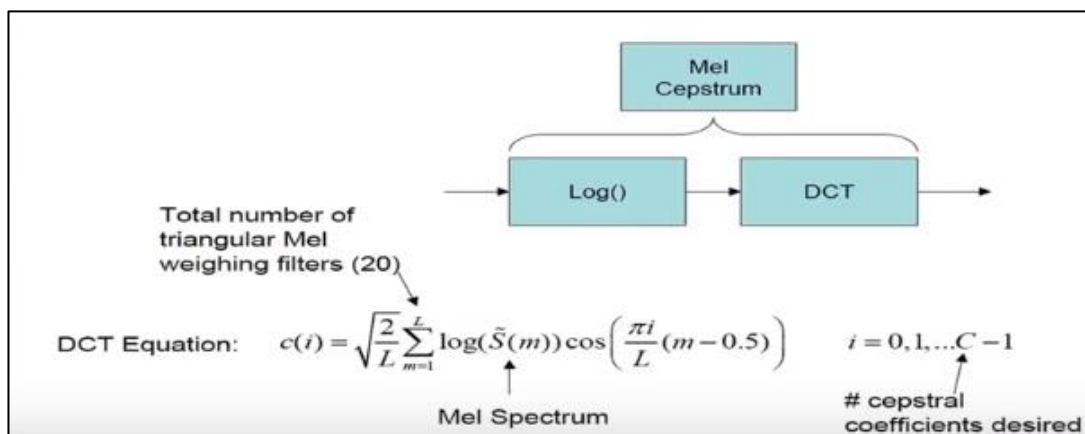
So, we calculate $X_{cap} L$; L is the number of filter

$S_{cap} L$ using Mel filter bank over the Mel filter, we are calculating Mel spectrum and once cepstral coefficient is analyzed based on the Mel shaped spectrum then is called **Mel frequency cepstral coefficient**

Block diagram of Extracting a sequence of 39-dimensional MFCC feature vectors



Instead of IFT we can take DCT:



FEATURE EXTRACTION FOR 4 MUSIC FILES

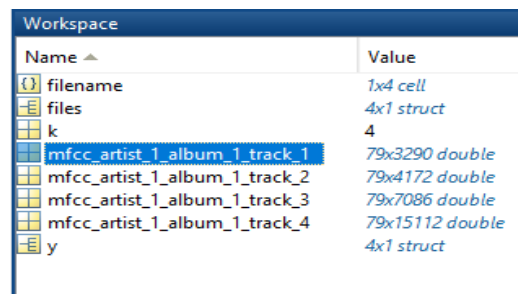
ATTACHMENTS ALONG THIS PDF DOCUMENT:

1. **main.m**: Main file (to RUN)
2. **ma_mfcc.m**: to perform the mel-frequency-cepstrum analysis.
3. **mfcc_coefficients.m**: to find the mfcc coefficients for all the given no. of .wav files
4. **a1.txt**: describes the basic commands on audio operations.
5. **Four sample music files(in tracks folder)**:

We have extracted features [mfcc_coefficients,DCT_coefficients] for all 729 files.

We have attached 4 sample music files out of 729 files, to prove that we have extracted the required features (mfcc,DCT)

6. **mfccResults.mat and mfccResults.xls in (mfccResult)** : Contains results of mfcc_coefficients and DCT matrix
- 7.
8. Output figures

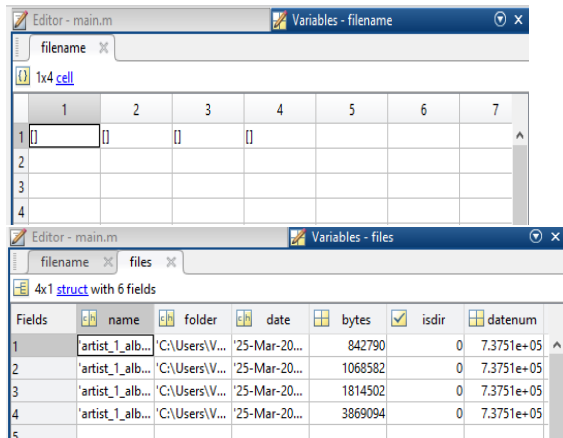


Name	Value
filename	1x4 cell
files	4x1 struct
k	4
mfcc_artist_1_album_1_track_1	79x3290 double
mfcc_artist_1_album_1_track_2	79x4172 double
mfcc_artist_1_album_1_track_3	79x7086 double
mfcc_artist_1_album_1_track_4	79x15112 double
y	4x1 struct

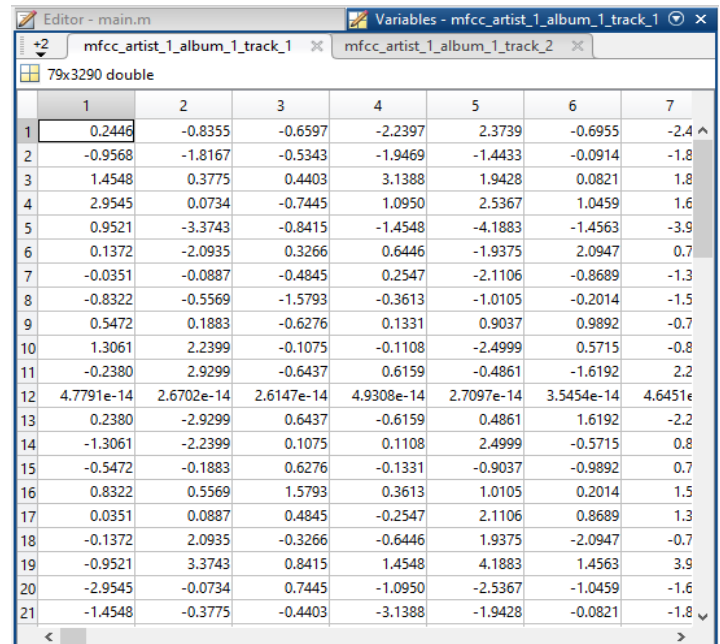
OUTPUTS:

(each cell has its own purpose:: See the workspace)

- A) **Filename cell is to read file name:**
B) **Files cell is to read the details of .wav files**



Fields	name	folder	date	bytes	isdir	datenum
1	artist_1_alb...	C:\Users\V...	'25-Mar-20...	842790	0	7.3751e+05
2	'artist_1_alb...	C:\Users\V...	'25-Mar-20...	1068582	0	7.3751e+05
3	'artist_1_alb...	C:\Users\V...	'25-Mar-20...	1814502	0	7.3751e+05
4	'artist_1_alb...	C:\Users\V...	'25-Mar-20...	3869094	0	7.3751e+05



	1	2	3	4	5	6	7
1	0.2446	-0.8355	-0.6597	-2.2397	2.3739	-0.6955	-2.4
2	-0.9568	-1.8167	-0.5343	-1.9469	-1.4433	-0.0914	-1.8
3	1.4548	0.3775	0.4403	3.1388	1.9428	0.0821	1.8
4	2.9545	0.0734	-0.7445	1.0950	2.5367	1.0459	1.6
5	0.9521	-3.3743	-0.8415	-1.4548	-4.1883	-1.4563	-3.9
6	0.1372	-2.0935	0.3266	0.6446	-1.9375	2.0947	0.7
7	-0.0351	-0.0887	-0.4845	0.2547	-2.1106	-0.8689	-1.3
8	-0.8322	-0.5569	-1.5793	-0.3613	-1.0105	-0.2014	-1.5
9	0.5472	0.1883	-0.6276	0.1331	0.9037	0.9892	-0.7
10	1.3061	2.2399	-0.1075	-0.1108	-2.4999	0.5715	-0.8
11	-0.2380	2.9299	-0.6437	0.6159	-0.4861	-1.6192	2.2
12	4.7791e-14	2.6702e-14	2.6147e-14	4.9308e-14	2.7097e-14	3.5454e-14	4.6451e
13	0.2380	-2.9299	0.6437	-0.6159	0.4861	1.6192	-2.2
14	-1.3061	-2.2399	0.1075	0.1108	2.4999	-0.5715	0.8
15	-0.5472	-0.1883	0.6276	-0.1331	-0.9037	-0.9892	0.7
16	0.8322	0.5569	1.5793	0.3613	1.0105	0.2014	1.5
17	0.0351	0.0887	0.4845	-0.2547	2.1106	0.8689	1.3
18	-0.1372	2.0935	-0.3266	-0.6446	1.9375	-2.0947	-0.7
19	-0.9521	3.3743	0.8415	1.4548	4.1883	1.4563	3.9
20	-2.9545	-0.0734	0.7445	-1.0950	-2.5367	-1.0459	-1.6
21	-1.4548	-0.3775	-0.4403	-3.1388	-1.9428	-0.0821	-1.8

- C) **k describes the no. of files:**

Editor - main.m

filename files k

1x1 double

	1	2	3
1	4		
2			

D) The wave is converted into mel-scale

All four cells are generated in mel-scale:

1. 'mfcc_artist_1_album_1_track_1'
2. 'mfcc_artist_1_album_1_track_2'
3. 'mfcc_artist_1_album_1_track_3'
4. 'mfcc_artist_1_album_1_track_4'

E) y cell is the output feature vector:

Editor - main.m

Variables - y

+2 mfcc_artist_1_album_1_track_3 mfcc_artist_1_album_1_track_4 y files

4x1 struct with 6 fields

Fields	name	folder	date	bytes	isdir	datenum
1	'artist_1_alb...	'C:\Users\V...	'25-Mar-20...	3869094	0	7.3751e+05
2	'artist_1_alb...	'C:\Users\V...	'25-Mar-20...	1814502	0	7.3751e+05
3	'artist_1_alb...	'C:\Users\V...	'25-Mar-20...	1068582	0	7.3751e+05
4	'artist_1_alb...	'C:\Users\V...	'25-Mar-20...	842790	0	7.3751e+05
5						

(For 729 music files feature extraction Output: See next page (Outputs are clearly stated))

Also note that we have implemented Project-2 on GTZAN Dataset which will be further explained in the successive pages

FEATURE EXTRACTION OUTPUTS FOR 729 MUSIC FILES

(Replace 4 with 729 in the main file. We have worked on PCA where we are getting crap results)

Workspace	
Name ^	Value
distance	743x729 double
euclideanDistance	1x79 double
features	729x79 double
filename	1x729 cell
files	729x1 struct
frobeniusNorm	0
GMMModel	1x1 gmdistributio
groupNames	729x1 cell
groups	729x1 double
i	745
j	745
k	6
matrix1	79x79 double
matrix2	79x79 double
mfcc_artist_100_album_1_track_1	79x79 double
mfcc_artist_100_album_1_track_2	79x79 double
mfcc_artist_100_album_1_track_3	79x79 double
mfcc_artist_100_album_1_track_4	79x79 double
mfcc_artist_100_album_2_track_1	79x79 double
mfcc_artist_100_album_2_track_2	79x79 double
mfcc_artist_100_album_2_track_3	79x79 double
mfcc_artist_100_album_3_track_1	79x79 double
mfcc_artist_100_album_3_track_2	79x79 double
mfcc_artist_100_album_3_track_3	79x79 double
mfcc_artist_100_album_4_track_1	79x79 double
mfcc_artist_100_album_4_track_2	79x79 double
mfcc_artist_100_album_4_track_3	79x79 double
mfcc_artist_100_album_4_track_4	79x79 double
mfcc_artist_100_album_5_track_1	79x79 double
mfcc_artist_100_album_5_track_2	79x79 double
mfcc_artist_100_album_5_track_3	79x79 double
mfcc_artist_101_album_1_track_1	79x79 double
mfcc_artist_101_album_1_track_2	79x79 double
mfcc_artist_101_album_1_track_3	79x79 double
mfcc_artist_102_album_1_track_1	79x79 double
mfcc_artist_102_album_1_track_2	79x79 double

Workspace	
Name ^	Value
mfcc_artist_96_album_1_track_2	79x79 double
mfcc_artist_97_album_1_track_1	79x79 double
mfcc_artist_97_album_1_track_2	79x79 double
mfcc_artist_97_album_1_track_3	79x79 double
mfcc_artist_98_album_1_track_1	79x79 double
mfcc_artist_98_album_1_track_2	79x79 double
mfcc_artist_98_album_1_track_3	79x79 double
mfcc_artist_99_album_1_track_1	79x79 double
mfcc_artist_99_album_1_track_2	79x79 double
mfcc_artist_99_album_1_track_3	79x79 double
mfcc_artist_99_album_1_track_4	79x79 double
mfcc_artist_9_album_1_track_1	79x79 double
mfcc_artist_9_album_1_track_2	79x79 double
mfcc_artist_9_album_1_track_3	79x79 double
mfcc_artist_9_album_1_track_4	79x79 double
mfcc_artist_9_album_1_track_5	79x79 double
mfcc_artist_9_album_1_track_6	79x79 double
mfcc_artist_9_album_2_track_1	79x79 double
mfcc_artist_9_album_2_track_10	79x79 double
mfcc_artist_9_album_2_track_11	79x79 double
mfcc_artist_9_album_2_track_2	79x79 double
mfcc_artist_9_album_2_track_3	79x79 double
mfcc_artist_9_album_2_track_4	79x79 double
mfcc_artist_9_album_2_track_5	79x79 double
mfcc_artist_9_album_2_track_6	79x79 double
mfcc_artist_9_album_2_track_7	79x79 double
mfcc_artist_9_album_2_track_8	79x79 double
mfcc_artist_9_album_2_track_9	79x79 double
mfcc_artist_9_album_3_track_1	79x79 double
mfcc_artist_9_album_3_track_2	79x79 double
mfcc_artist_9_album_3_track_3	79x79 double
mfcc_artist_9_album_3_track_4	79x79 double
mfcc_artist_9_album_3_track_5	79x79 double
mfcc_artist_9_album_3_track_6	79x79 double
values	1x729 double
variables	745x1 cell

Fig: Features Extracted for all 729 files

***** Note we are facing Problem of Dimension Reductionality on 729 files**

We also have Spectral plots for one music file:

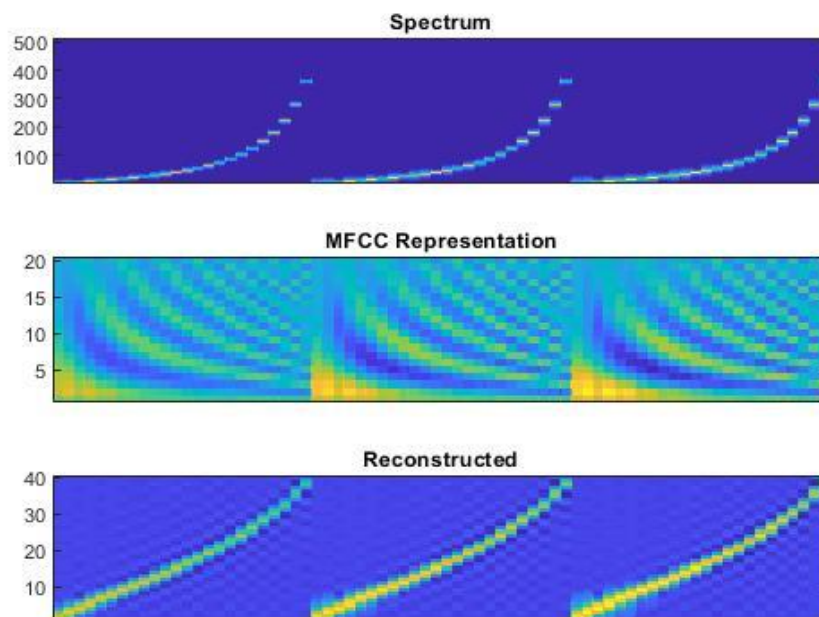


Fig: Spectral Plot Output

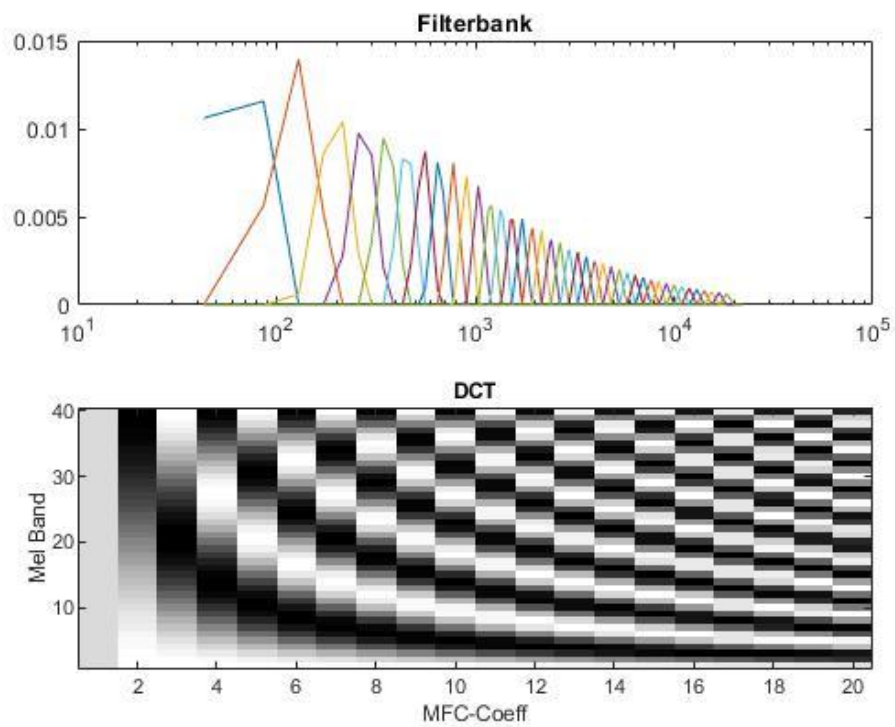


Fig:MFC-Coefficients for a single music file

Difficulty with Dimension Reduction in Project-1:

Since we have 729 files of each great then ~1min 15sec, we faced difficulty in the PCA Dimension Reduction of all the features sampled at 11025 frequency components which on 64-bit processor taking 4hours for each simulation. So, we were unable to solve the errors in the given time. Also, we faced few problems in DCT,MFCC reductions.

PROJECT-2

IMPLEMENTATION WITH GTZAN DATASET

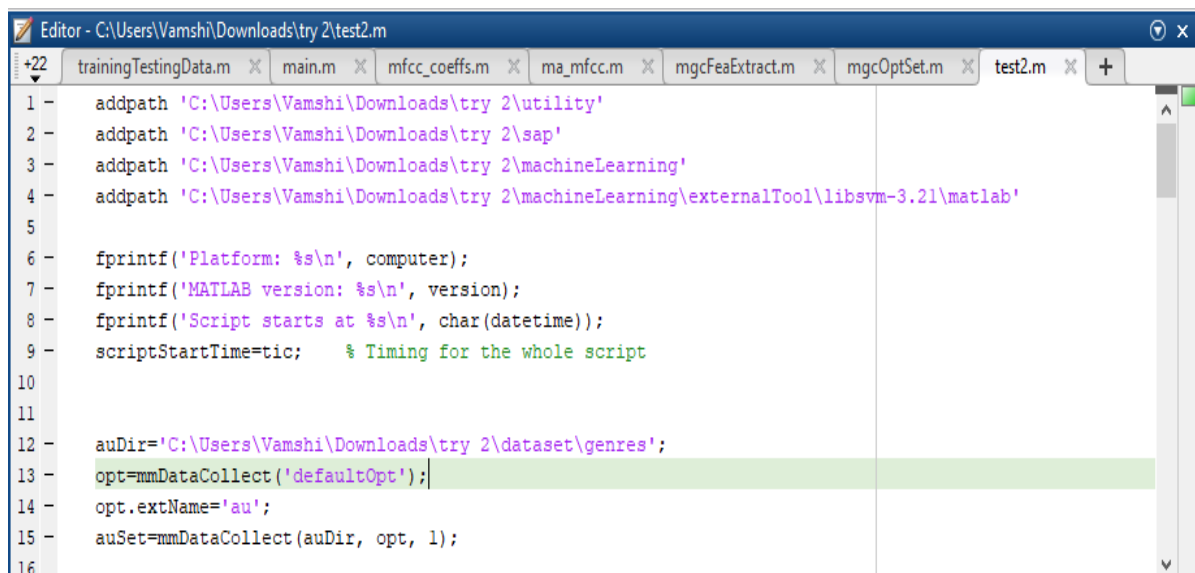
So, We have taken other dataset GTZAN Dataset, you can observe the dataset and tutorial for the project here[8]: <http://mirilab.org/jang/books/audioSignalProcessing/appNote/musicGenreClassification/html/goTutorial.html>

This time for this project, we use matlab Toolboxes already available:
<http://mirilab.org/jang/matlab/toolbox/>

All the Toolboxes are downloaded from this Tutorial link.

GTZAN Dataset has 1000music files with 10 classes and each file is of short duration (~10sec or ~20sec) which we felt easier to extract features. It will take atleast 15mins to run and extraxt features for all 1000files.

Working on Project-2:

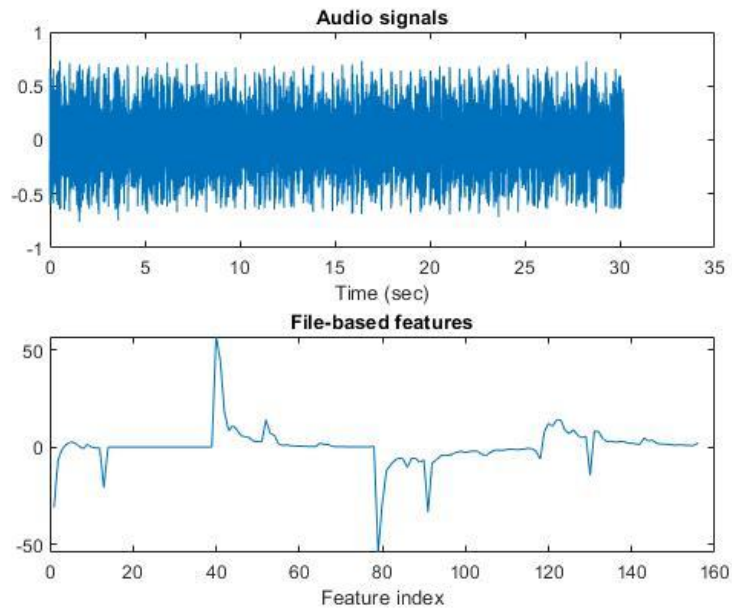


```
Editor - C:\Users\Vamshi\Downloads\try 2\test2.m
+22  trainingTestingData.m  main.m  mfcc_coeffs.m  ma_mfcc.m  mgcFeaExtract.m  mgcOptSet.m  test2.m  +
1 -  addpath 'C:\Users\Vamshi\Downloads\try 2\utility'
2 -  addpath 'C:\Users\Vamshi\Downloads\try 2\sap'
3 -  addpath 'C:\Users\Vamshi\Downloads\try 2\machineLearning'
4 -  addpath 'C:\Users\Vamshi\Downloads\try 2\machineLearning\externalTool\libsvm-3.21\matlab'
5
6 -  fprintf('Platform: %s\n', computer);
7 -  fprintf('MATLAB version: %s\n', version);
8 -  fprintf('Script starts at %s\n', char(datetime));
9 -  scriptStartTime=tic;    % Timing for the whole script
10
11
12 -  auDir='C:\Users\Vamshi\Downloads\try 2\dataset\genres';
13 -  opt=mmDataCollect('defaultOpt');
14 -  opt.extName='au';
15 -  auSet=mmDataCollect(auDir, opt, 1);
16
```

1. Install in the current working directory, all the MIR Toolboxes[Utility Toolbox, Machine Learning, Speech and Audio Processing Toolbox, Automatic Speech Recognition using HTK Toolbox] and Libraries mentioned in the link. Then start using the Tutorial Link

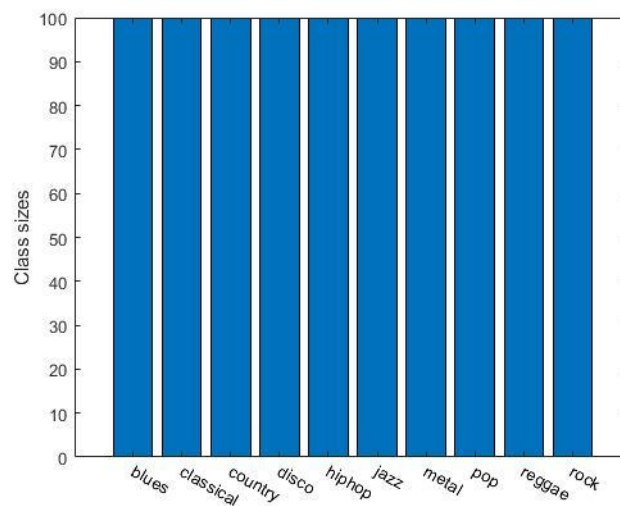
2. **Feature Extraction:** MFCC Feature Extraction (Refer In test2.m):

```
opt.auFeaFcn=@mgcFeaExtract; % Function for feature extraction
```



Data Visualization:

It is necessary to Visual the Data Features as and when the data is trained. This graph below shows the classes trained 100%-Full class Size.



```
figure;
```



```
[classSize, classLabel]=dsClassSize(ds, 1);
```

Here we visualized: 156 features, 1000 instances, 10 classes

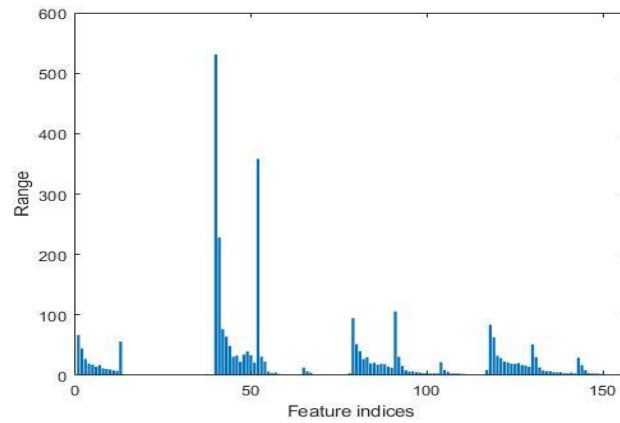
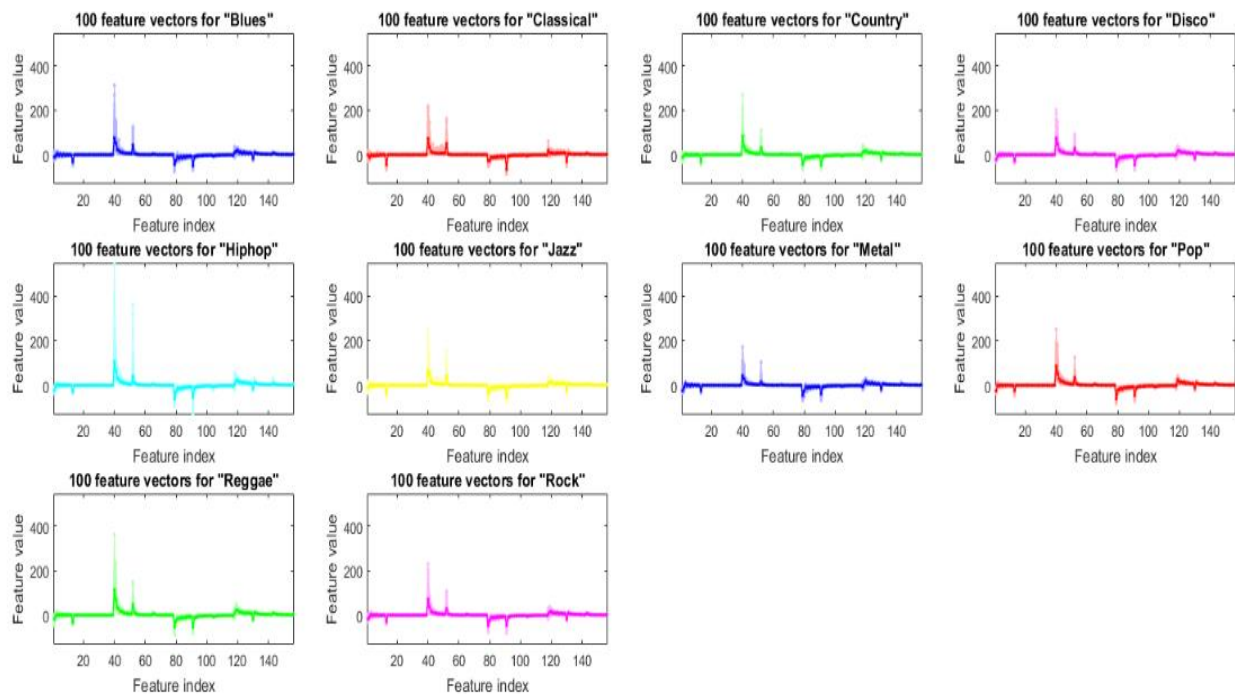


Fig: Plotting Range of features of the Dataset

```
figure; dsFeaVecPlot(ds); figEnlarge;
```



Plot the feature Vectors in Each Class

3. Dimension Reduction(PCA):

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.

PCA Algorithm is finely explained in shortly here:

Get the required data → Subtract the mean → Calculate the covariance matrix → Calculate the eigenvectors and eigenvalues of the covariance matrix → Choosing components and forming a feature vector → Deriving the new data set

We considered dimensionality reduction via PCA (principal component analysis). First, we plotted the cumulative variance given the descending eigenvalues of PCA. For Maths on PCA[9] & [10].

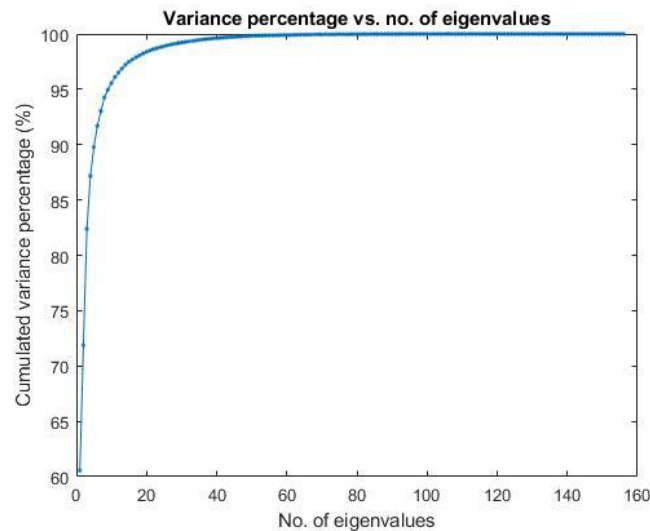


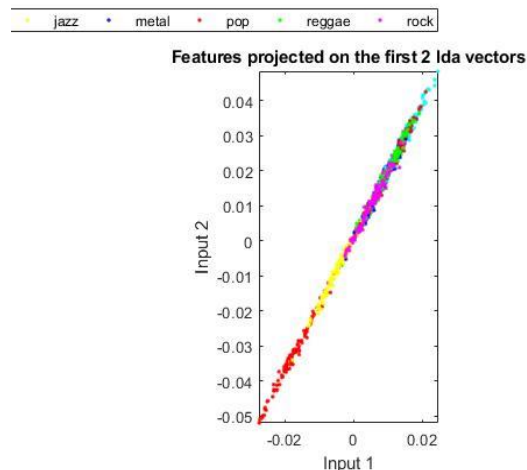
Fig: We observe the cumulative Variance and fix threshold: 95%

Reduce the dimensionality to 10 to keep 95% cumulative variance via PCA.

However, our experiment indicates that if we use PCA for dimensionality reduction, the accuracy will be lower. As a result, we shall keep all the features for further exploration.

In order to visualize the distribution of the dataset, we can project the original dataset into 2-D space. This can be achieved by LDA (linear discriminant analysis):

Apparently the separation among classes is not obvious. This indicates that either the features or LDA are not very effective.



4. Classification(K-NN):

K-NN is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space.

In *k-NN classification*, the output is a class membership. An object is classified by a plurality 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.

k -NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.

Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

The neighbors are taken from a set of objects for which the class (for k -NN classification) or the object property value (for k -NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. So, we used the most straightforward KNNC (k-nearest neighbor classifier). For more about K-NN refer[11].

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
 - 3.1 Calculate the distance between the query example and the current example from the data.
 - 3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

K-NN Results: Recognition Rates are as follows:

```
rr=57.6% for original ds
```

```
rr=64.9% for ds after input normalization
```

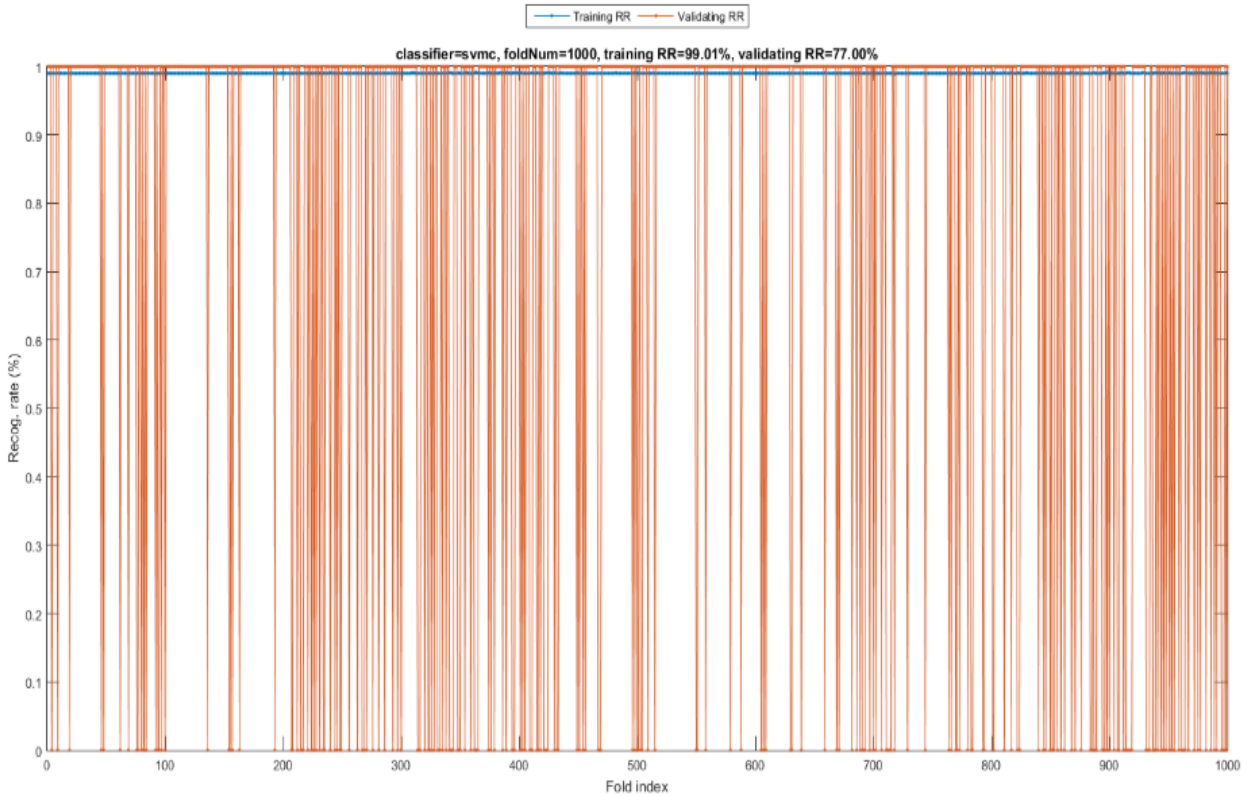
KNNC does not give satisfactory result. So we shall try other potentially better classifiers, such as SVM. Before try SVM, here we use a function [mgcOptSet.m](#) to put all the MGC-related options in a single file.

5. Classification by SVM:

SVM are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging

to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. For more info of SVM refer[12].



SVM Outputs:

Fold = 100/1000

Fold = 200/1000

Fold = 300/1000

Fold = 400/1000

Fold = 500/1000

Fold = 600/1000

Fold = 700/1000

Fold = 800/1000

Fold = 900/1000

Fold = 1000/1000

Training RR=99.01%, Validating RR=77.00%, classifier=svmc, no. of folds=1000

Time for cross-validation = 490.748 sec

The recognition rate is 77%, indicating SVM is a much more effective classifier.

We actually plot the confusion matrix between computed classes and desired classes:

	blues	classic	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	83.00% (83)	0	2.00% (2)	2.00% (2)	1.00% (1)	2.00% (2)	5.00% (5)	0	0	5.00% (5)
classical	0	94.00% (94)	0	0	2.00% (2)	2.00% (2)	0	0	0	2.00% (2)
country	4.00% (4)	0	70.00% (70)	5.00% (5)	0	1.00% (1)	0	6.00% (6)	2.00% (2)	12.00% (12)
disco	2.00% (2)	0	3.00% (3)	66.00% (66)	7.00% (7)	1.00% (1)	2.00% (2)	4.00% (4)	6.00% (6)	9.00% (9)
hiphop	3.00% (3)	0	0	4.00% (4)	74.00% (74)	0	2.00% (2)	1.00% (1)	14.00% (14)	2.00% (2)
jazz	0	5.00% (5)	1.00% (1)	1.00% (1)	0	90.00% (90)	0	0	0	3.00% (3)
metal	6.00% (6)	0	2.00% (2)	3.00% (3)	1.00% (1)	0	83.00% (83)	0	0	5.00% (5)
pop	0	0	9.00% (9)	4.00% (4)	2.00% (2)	0	0	80.00% (80)	2.00% (2)	3.00% (3)
reggae	5.00% (5)	0	4.00% (4)	6.00% (6)	8.00% (8)	0	0	4.00% (4)	71.00% (71)	2.00% (2)
rock	4.00% (4)	0	12.00% (12)	11.00% (11)	0	3.00% (3)	7.00% (7)	1.00% (1)	3.00% (3)	59.00% (59)

FUTURE WORK:

1. We wish to solve the problem of PCA in Project-1
 2. Use a K-NN Algorithm and Classify all the 729 files
 3. Use a test music file and categorize that one to which class it belongs
 4. Create a GUI and make the project-1 more user-friendly.
- Is all possible cases, we look further to work/research intern on project-1 with LAAS-CNRS.

REFERENCES:

- [1] ISMIR2004 Audio Description Contest: http://ismir2004.ismir.net/genre_contest/
- [2] 5TH International Conference on Digital Libraries for Musicology, Paris 2018
<http://www.iremusc.cnrs.fr/fr/evenements/5th-international-conference-digital-libraries-musicology-paris-2018>
- [3] “Computational Models of Music Similarities and their Application in Music Information Retrieval”- A Desseration submitted to Prof. Dipl.-Ing. Dr. techn, Gerhand Widmer, Institute fur Computational Perception – Johannes Kepler Universitat Linz & TU Wien
- [4] Elias Pampal’s MA Toolbox- For MALTAB Functions: <http://www.pampalk.at/ma/>

- [5] NPTEL Digital Speech Processing Videos: Mel Frequency Cepstral Coefficients by Prof. S.K.Das Mandal, IIT Kharagpur, India. (For Math behind MFCC)
 Youtube Link-1: <https://www.youtube.com/watch?v=E9LGj9s9sbw>
 Youtube Link-2: <https://www.youtube.com/watch?v=KzevshgDv8g&t=406s>
- [6] “Mel Frequency Cepstral Coefficients for Music Modelling” by Beth Logan, Cambridge Research Laboratory, Compaq Computer Corporation, Cambridge MA 02142
- [7] Wikipedia: mfcc, cepstrum, history of mfcc.
- [8] Jyh-Shing Roger Jang GTZAN Project Tutorial:
<http://mirllab.org/jang/books/audioSignalProcessing/appNote/voicedSoundDetect4audioMusic/html/goTutorial.html>
- [9] PCA -1(for Mathematics): <http://www.cs.tau.ac.il/~rshamir/abdbm/pres/17/PCA.pdf>
- [10] PCA -2: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
- [11] K-NN classifier Description:<https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- [12] SVM: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
- [13] “Beth Logan, Mel Frequency Cepstral Coefficients for Music Modeling.
 The Cambridge Research Laboratory, Compaq Computer Corporation”.
- [14] “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”.
- [15] “Elias Pampalk Islands of Music Analysis, Organization, and Visualization of Music Archives”.
- [16] “Jayme Garcia, Arnal Barbedo and Amauri Lopes Automatic Genre Classification of Musical Signals”.
- [17] “Nir Ailon And Bernard Chazelle: The fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors”
<https://www.cs.princeton.edu/chazelle/pubs/FJLT-sicomp09.pdf>
- [18] “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