SHRI VILE PARLE KELVANI MANDAL'S
DWARKADAS.J. SANGHVI COLLEGE OF ENGINEERING
Approved by AICTE and Affiliated to the University of Mumbai
Department of Electronics & Telecommunication Engineering

ADSP Mini Project Report

Speaker Recognition using Mel Frequency Cepstral Coefficients and Gaussian Mixture Model

Sr No	Name	SAP ID
1	Sarfaraj	60002160002
2	Ajinkeya Chitrey	60002160010
3	Sanjeet Krishna	600002160050
4	Atulya Kumar	60002160052

Name of Guide: Prof. Sunil Karamchandani

SHRI VILE PARLE KELVANI MANDAL'S

DWARKADAS.J. SANGHVI COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated to the University of Mumbai

Department of Electronics & Telecommunication Engineering

CERTIFICATE

This is to certify that Master ________________SAP ID:__________________ of **BE EXTC 1** has submitted their Mini Project Advanced Digital Signal Processing Lab for the Academic Year 2019-2020

Guide Examiner

Head of Department

I.Introduction

This project attempts to build an automatic speaker recognition. This system identifies the speaker whose voice is in the data set it is trained with. The system utilizes MFCCs (Mel-Frequency Cepstral Coefficients) and GMM (Gaussian Mixture Model). The result of this project confirms that this model works fairly well for automatic speaker recognition. This project also examines the effect of bit rate and sample rate, as well as window type, on the hit rate for the system. This project attempt to create an automatic speaker recognition system. This system is trained on the voices of eight members of the project team. It is expected that the accuracy for this system to be at least 80%.

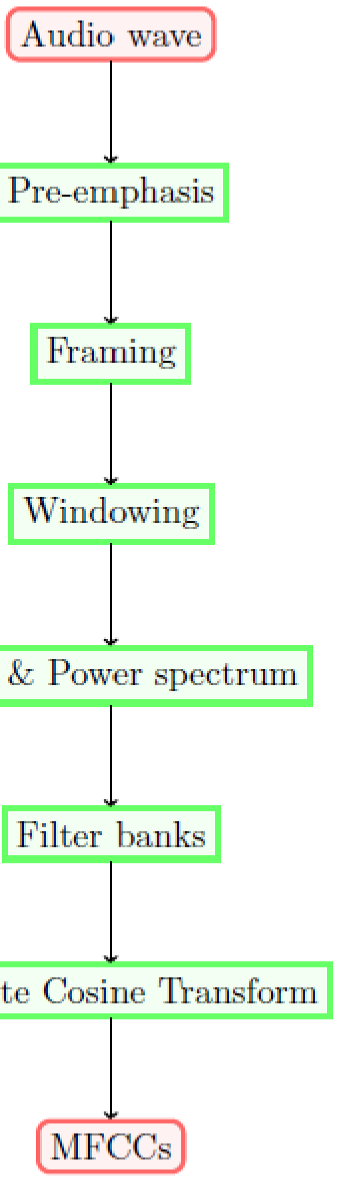
II.Problem specification

Like musical instruments, human voices differ by pitch and timbre. These features are defined by each person's larynx. Some people produce higher pitch, while some produce lower pitch, and even among people producing similar pitch, they usually have different timbres too.

These features can be mathematically represented in frequency domain. Pitches are the frequencies with high magnitude, while timbres are the ones represented by frequencies of lower magnitude. In time domain, timbres can be seen as small ripples on the signal.
Therefore, to distinguish speakers' voices, we need to analyse the voices of speakers in frequency domain, which can be computed with Fast Fourier Transform.

III.Approach

Mel-Frequency Cepstral Coefficients



The procedures to acquire MFCCs

Cepstrum

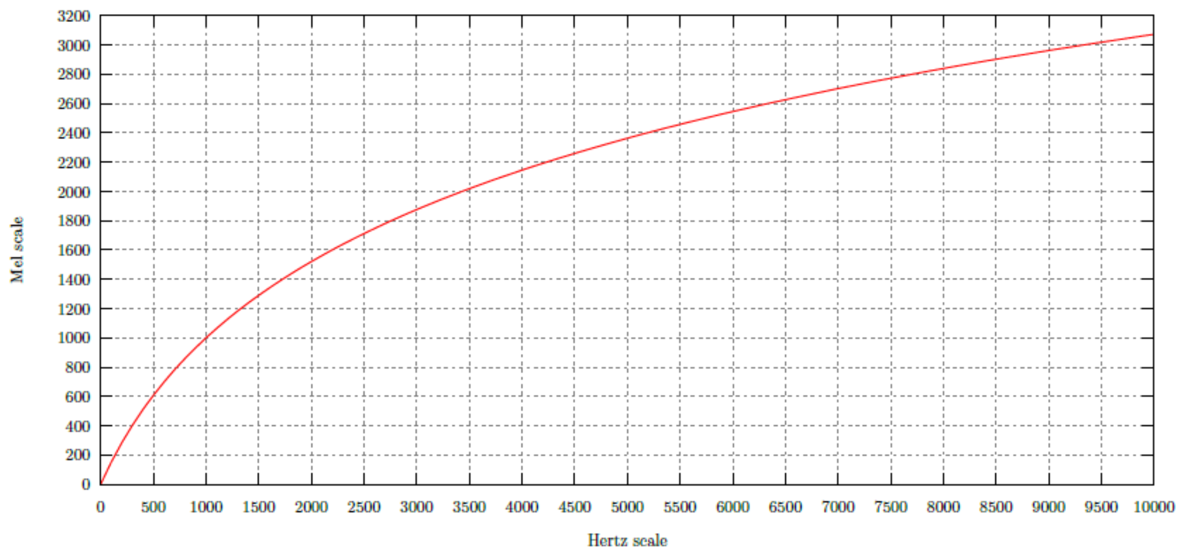
'Cepstrum' (an anagram of 'spectrum' with four first letters reverted), coined by Bogert et al, is the inverse Fourier transform of the spectrum of a signal in logarithm scale (Rabiner and Schafer 2007). Mathematically, the Cepstrum can be represented as:

$$\hat{x}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log(X(e^{j\omega}))e^{j\omega n}d\omega$$

Mel scale

Mel scale is a subjective scale, as opposed to frequency in Hertz scale, which is objective (Stevens, Volkmann, and Newman 1937). It represents how humans perceive sounds. The mathematical formula to convert from Hertz scale to Mel scale is:

$$m = 2595\log_{10}(1 + \frac{f}{700})$$



Plot of Mel scale against Hertz scale

Figure shows how frequency is mapped from Hertz scale to Mel scale. As can be seen in the figure, higher frequencies are closer in Mel scale, compared to low frequencies. This is because humans can distinguish low frequencies better than high frequencies.

As Mel scale aligns with how humans' perception of sounds, it is fitting to distinguish humans' voices, which is done quite well by human ears.

MFCCs

Mel-frequency Cepstrum coefficients are the result of a cosine transform of the real logarithm of the short-term energy spectrum expressed on a Mel-frequency scale.

a. Pre-emphasis

It is a technique to flattening the magnitude spectrum and balancing the high and the low frequency components. Pre-emphasis can increase the performance significantly up to above 10%

It is represented mathematically as:

$$y(t) = x(t) - \alpha x(t - 1)$$

Where α is the pre-emphasis coefficient.

b. Framing

Instead of using the whole signal, we divide the audio files into frames of 20-40 ms. The frame cannot be too short (too few samples for adequate information) nor too long (the signal changes too much).

c. Windowing

The framing is essentially applying a rectangular window on multiple points of the signal. Rectangular window has a quite poor performance, such as high approximation error and larger ripples in the stopband after Fourier transform. Therefore, it is important that a better window is applied on these frames.

d. FFT and Power Spectrum

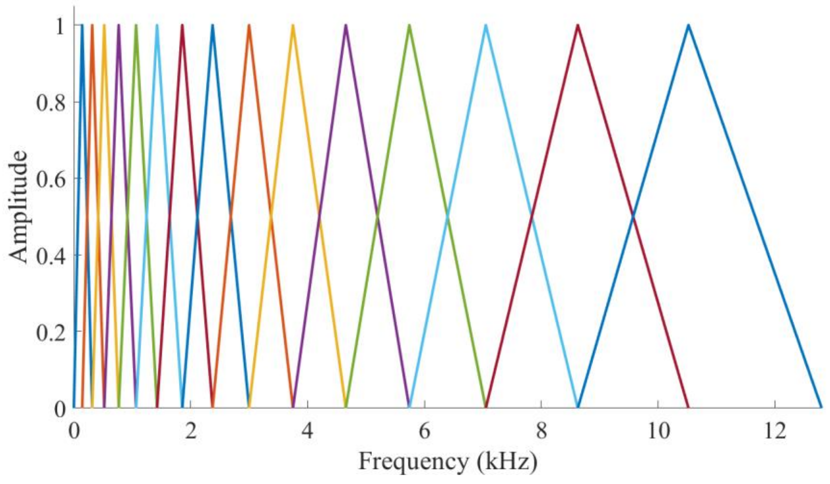
After framing and windowing, Fast Fourier Transform is applied in order to acquire the spectra for the frames. The periodogram-based spectral estimate for the speech frame $s_i(n)$ is given by the formula:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2$$

where $S_i(k)$ is the FFT of $s_i(n)$.

e. Filter Banks

In this step, the spectrum is converted to Mel-scale. Each of the triangular shape is a filter, from which the information about that range of frequencies in the spectrum is extracted. Each of these filters.



Mel filter banks is corresponding to a level in Mel scale, which is why they are less dense in higher frequency. We then take the energies of each result, then apply DCT on them, and get the MFCCs.

f. Deltas and delta-deltas

Though these are not a part of MFCC, deltas enrich the features extracted from this procedure and hence included here. Deltas and delta-deltas are also known as differential and acceleration coefficients. Deltas can be calculated as:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

where c_{t+N} to c_{t-N} are the static coefficients, i.e. the MFCCs around the time frame t.

Gaussian Mixture Model (GMM)

In Gaussian Mixture Models, data points can be clustered into several groups, each of which is modelled by a Gaussian distribution. These models are typically learned by using Maximum Likelihood Estimates (MLE) techniques.

a. Expectation-maximization (EM)

The EM algorithm is an iterative algorithm that starts from some initial estimate of θ = {π, μ, σ}, and then proceeds to iteratively update θ until convergence is detected. Each iteration consists of an E-step and an M-step.

i. Initialize θ
ii. Expectation: calculating the expectation of the component assignments C_k for each data point $x_i \in X$ given the model parameters ϕ_k, μ_k, and σ_k.
iii. Maximization: maximizing the expectations calculated in the E step with respect to the model parameters. This step consists of updating the values ϕ_k, μ_k, and σ_k.

Once the EM algorithm has run to completion, the fitted model can be used to perform various forms of inference. That is, given a new data point, the model can evaluate the probability that this data point is belong to which cluster/component.

Application

In this project, we create one GMM for each person. Each of these models consists of 15 components, corresponding to 15 features extracted from previous step (5 MFCCs, 5 deltas, and 5 delta-deltas).

IV.Implementation

MFCCs

We use python_speech_features package for the implementation of MFCCs extraction, instead of implementing everything from scratch.
Firstly, we have to read the audio file to be learnt. Both the audio content and sample rate are read.

The MFCCs are then extracted with python_speech_features.mfcc(). Afterwards, python_speech_features.mfcc() is used twice to enrich the features, which is supposed to increase performance by 20%.

We extract the MFCCs with these parameters:

• winlen– the length of the analysis window in seconds: 0.05s
• winstep– the step between successive windows in seconds: 0.05s
• numcep– the number of cepstra to return: 5
• nfilt– the number of filters in the filterbank: 30
• nfft– the FFT size: 512 (default)
• lowfreq– lowest band edge of Mel filters: 0 (default)
• highfreq– highest band edge of Mel filters: rate/2 (default)
• preemph – apply pre-emphasis filter with: 0.97 (default)
• ceplifter– apply a lifter to final cepstral coefficients: 22 (default)
• appendEnergy– True(default), meaning the zeroth cepstral coefficient is replaced with the log of the total frame energy
• winfunc– the analysis window to apply to each frame: here we try two options: no windows vs using Hamming window

Finally, those features are stacked together:

Having the function implemented in the previous part, extracting voice features for one person now can be done by iterating over a list of the person audio file.

GMM

To generate GMM for each person, we use sklearn.mixture.GMM to create a model, then fit it with features (MFCCs) extracted from previous step.

Here, we create a model with following parameters:

• n_components - number of mixture components: 15 (5 MFCCs, 5 delta, 5 delta-deltas)
• n_iter - number of EM iterations to perform: 200
• covariance_type - the type of covariance parameters to use: 'diag'
• n_init - number of initializations to perform: 3

Other default parameters are specified in the Scikit learn's.

Combining the models

The GMM can evaluate the likelihood that an audio is recorded by a person. If we compare this score of each model for an audio file, the person whose model with the most likelihood score will be identified as speaker.

V.Results

a. Metrics

From 135 recordings of each person, we use 100 recordings for training the model and 35 recordings for testing. Each model is tested with all $35 \times 8 = 280$ audio files. The accuracy of the system is defined as:

$$A = \frac{\sum n_{\text{hits}}}{\sum n_{\text{files}}}$$

The result is taken to the second digits after the decimal point. Note that since the training data set is shuffled every time it trains, the reproduced result may differ up to 0.5%.

Accuracy

i. No window

As mentioned in earlier section, we have two sets of recordings with different sampling rates and bit rates. We train the models with this data set as is, when they are all converted to low rates, and when they are all converted to high rate.

System	Mixed test	Low test	High test
Mixed	99.04%	12.60%	59.74%
Low	1.54%	88.99%	38.65%
High	68.44%	36.98%	99.18%

Accuracy for the speaker recognition system

As can be seen from the table, each of the system performs the best with the tests that are of the same bit rate and sample rate as itself. Among the three, the system with high-rate trained model performs the best with its corresponding test set (99.18%), whereas low-rate trained model system is the worst (88.99%). This is because high bit rates and sample rates preserve more data and thus can represent more nuance. High-rate trained system also appears to outperform the low-rate trained system when the test data has both high and low rate.
Note that mixed test only performs so well because the test is biased. This is a mistake on the team.

ii. Apply different windows

Due to time constraints, we only test with high-rate system. Here are the records for different windows:

Window	Accuracy
Rectangular (no windows)	99.18%
Hamming	99.78%
Bartlett	99.25%
Blackman	98.48%
Hanning	99.10%

Accuracy when applying different windows

It appears that Blackman window works the worst for framing in this case, even worse than no windows. Hamming window works the best, but not by far above the rectangular windows.

VI. Development

Dataset - https://drive.google.com/drive/folders/1kzTGzFeVPPxlAYj0nsVZlHpKJePSD0fy
Code - https://drive.google.com/file/d/1fXzI_kWUwhIjBv8Vv7Ko8H-8ziG6BZiO/

VII. Conclusion

From the result, we have following conclusions:

• The system can perform fairly well for a small set of persons.
• Data used for training should have the same specifications as the tests
• High bit rates and sample rates lead to more accuracy in the recognition
• Hamming window works the best for framing step. Blackman window, on the other hand, performs the worst.

VIII. References

1. Rabiner, L.R. and R.W. Schafer (2007). Introduction to Digital Speech Processing.

2. Stevens, S. S., J. Volkmann, and E. B. Newman (1937). "A Scale for the Measurement of the Psychological Magnitude Pitch". In: The Journal of the Acoustical Society of America

3. O'Shaughnessy, D. (1987). Speech communication: human and machine. Addison Wesley series in electrical engineering. Addison-Wesley Pub

4. Katz, Alexander and Eli Ross (n.d.). Maximum Likelihood Estimation.

5. James Lyons (2013). python speech features documentation. scikit-learn developers (2013). sklearn.mixture.GMM.