EEC 201 Final Project

Bishane Sagal and Karmvir Thind

March 2024

1 Introduction

The goal of this project is to develop a speaker recognition program. Using 11 given speech samples and 18 speech samples from our colleagues as training data we aim to write a program that will match input test data to the correct student. We experienced difficulty reading in the eighth sample of the test data so our model selection omitted this sample, this may have impacted our results.

2 Approach

Our approach to the problem can be summarized into three steps:

1. Perform feature extraction on the audio data
2. Develop/Choose an algorithm to learn from the features and fit a model
3. Classify new audio data using the developed model

2.1 Feature Extraction

For feature extraction we made use of the Mel Frequency Cepstrum Coefficients (MFCCs). We followed a the mechanical procedure for producing the MFCCs: Compute the Short Time Fourier Transform of the input data (STFT), convert the magnitude of the STFT (to eliminate complex numbers) to mel-scale using appropriately spaced filter banks, take the log of the mel-scale STFTs, and perform the Discrete Cosine Transform (DCT) on these values to produce the MFCCs.

2.2 Algorithm Selection

We tested various models available through Python's ScikitLearn library and also developed our own implementation of the recommended Linde-Buzo-Gray (LBG) algorithm. Ultimately we landed on the LBG algorithm as it produced

the best results. We tested various models using for loops to sweep through various parameters and hyperparameters. Using the classification output we tuned the model to achieve our results of 100% accuracy for the given speech data and an average of 77.7% for the EEC 201 class data.

2.3 Model Prediction

Assessment of our model was done using features of the pre-built models from Python's Scikitlearn or in the case of our LBG algorithm, we displayed which speaker we were making a selection for, which speaker in the codebook we classified the speaker with, returned a list where the index corresponds to the speaker in the test data and the value to the speaker classification match in the codebook, and finally a calculated decimal value equal to the percentage of the number of speakers classified correctly to total speakers in the test dataset.

3 Tests

3.1 Test 1

Auditory performance rate was 6 out of 8 students guessed correctly from the test dataset for a total recognition rate of 75%. This was performed by listening to the training dataset, having a third party randomly play a speech file, and then guessing which speaker from the training dataset produced the audio. During this test, the subject was able to listen to the training files as many times as desired. This gives the human subject an edge in that the program only "listens" to each audio file once.

3.2 Test 2

The sampling rate of our load function is 22050, giving us 256 samples in 0.0116 milliseconds.

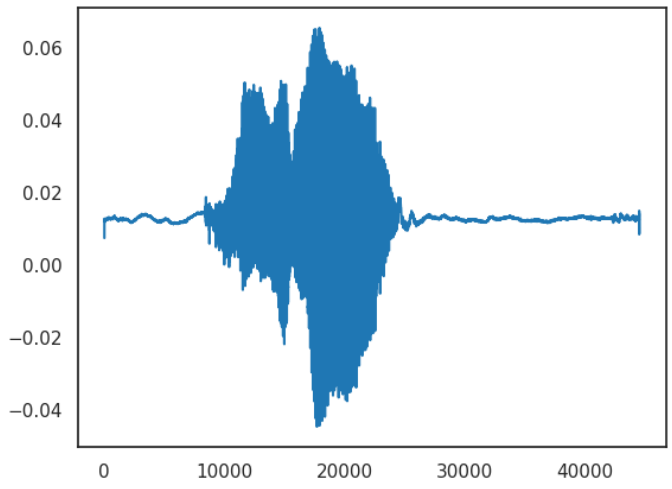

Figure 3.1

Periodogram with N = 128

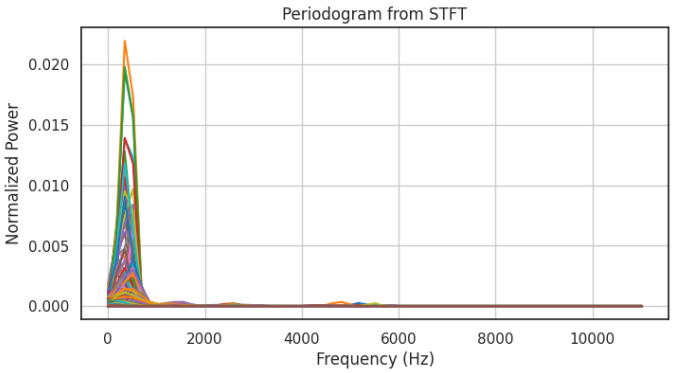

Figure 3.2

Periodogram with N = 256

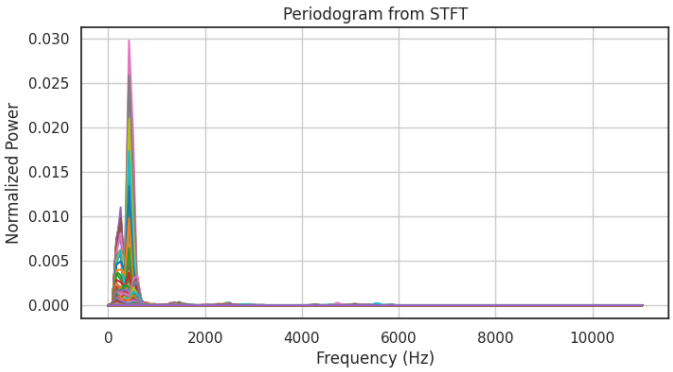

Figure 3.3

Periodogram with N = 512

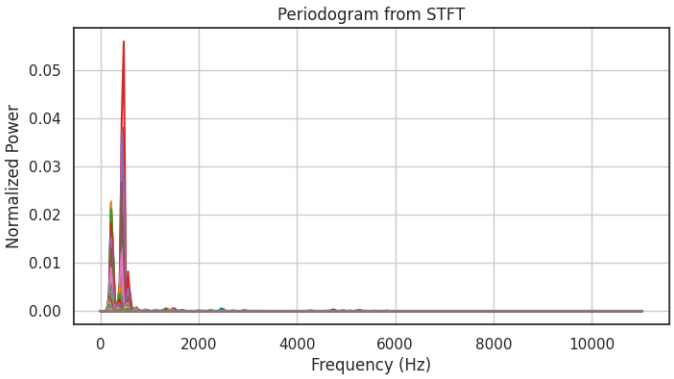

Figure 3.4

3.3 Test 3

Ideal Mel Spaced Filter Banks

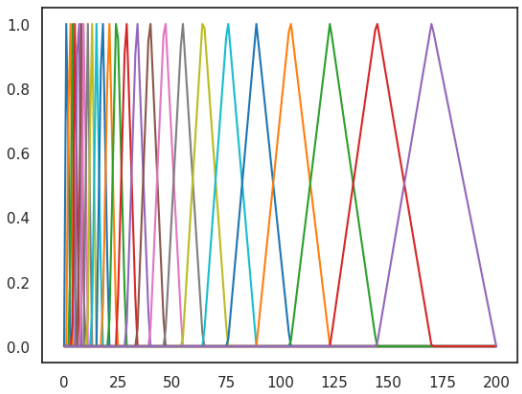

Figure 3.5

Spectrum of an audiofile with no mel frequency wrapping

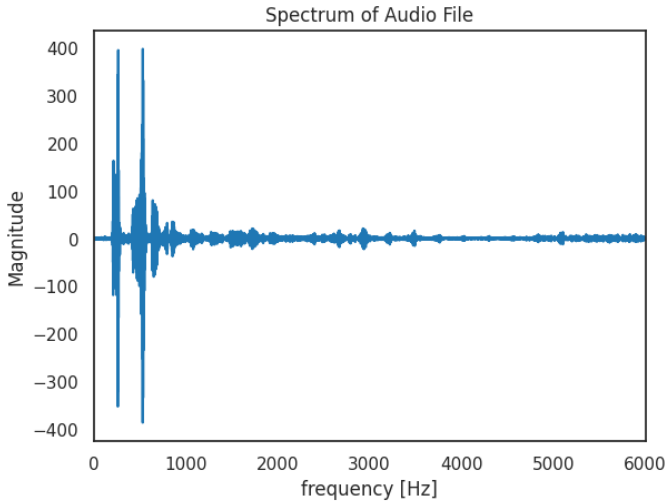

Figure 3.6

The Mel Frequency wrapping reduces the bandwidth of our audiofile spectrum by placing a larger weighting emphasis on the lower frequency components. The auditory resolution of humans is the same in narrow low frequency bands and wide high frequency bands and this is exactly what the mel frequency wrapping achieves. The magnitude of higher frequency components (approximately above 500 Hz) on the audio-file spectrum are dramatically reduced.

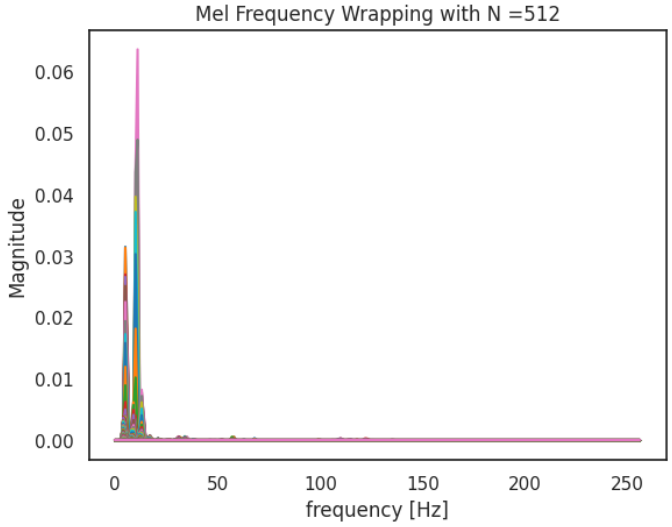

Figure 3.7

3.4 Identifying Points of High Energy

As seen below, we have high intensity points in the range of 0.5 to 1 seconds. The highest energy points are with the 128-512 frequency band. The intensity plotting is in accordance with our raw audio file coefficients because our largest amplitudes lie around the 20000 sample which corresponds to approximately 1 second (sample 22050).

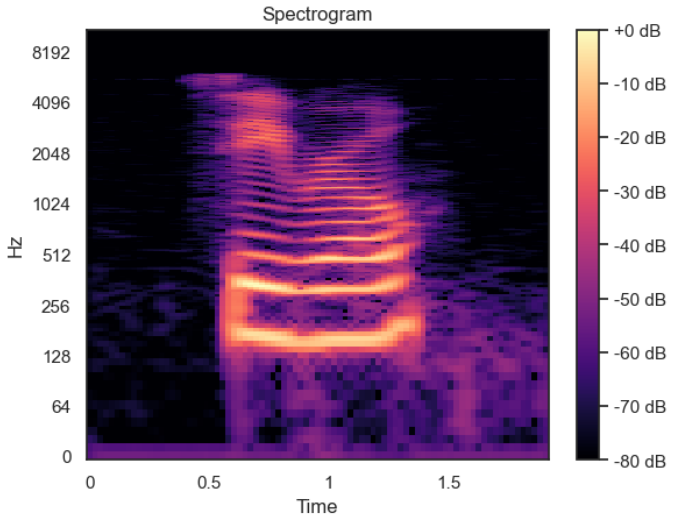

Figure 3.8

3.5 Test 5 - MFCC Plot

Training Data MFCC vectors of two speakers plotted in a 2-D plane. The MFCC vectors of speaker three form two loosely spanned clusters (left side/bottom right). The MFCC vectors of speaker four does not appear in distinct clusters but there is some weighting on the left side of the graph.

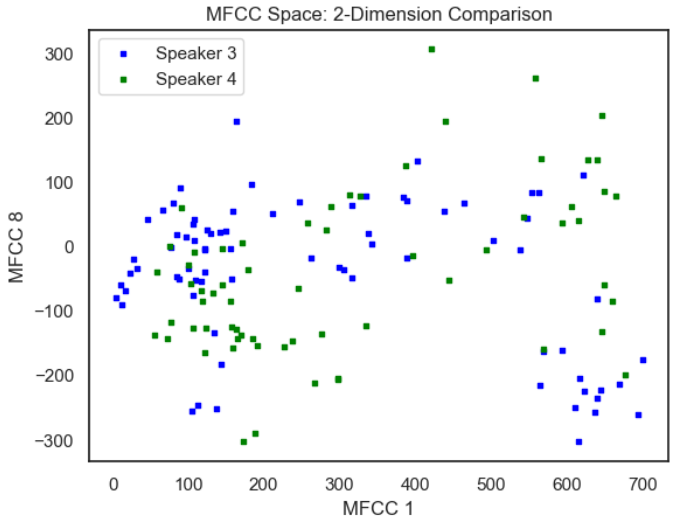

Figure 3.9

Training Data MFCC vectors plotted in a 2-D plane.

3.6 Test 6 - VQ Codewords

Epsilon value for these code words were 0.01 and 512 centroids were used for these codewords. The Euclidean distance measurement method was used for our clustering data points. The additional parameters for the codeword generation can be seen in Test 7.

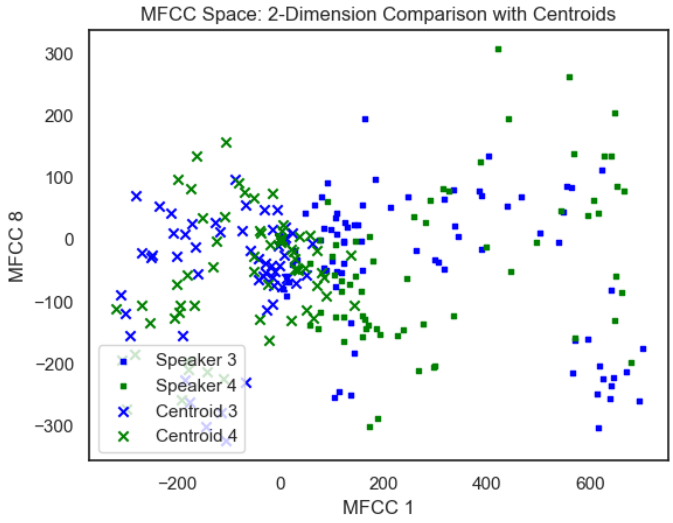

Figure 3.10

3.7 Test 7 - Given Speech Data

Our system demonstrated a recognition accuracy rate of 100%. Our results are tabulated in Fig. 3.11. We implemented two separate LBG algorithms, three separate MFCC calculation algorithms, and various hyperparameters to achieve this rate. We opted to use the Python librosa package's 'mfcc' method to calculate the MFCC coefficient as it yielded the best results. Our final model had the following parameters:

1. Centroids: 512
2. Epsilon: 0.01
3. FFT Length: 1440
4. Window Length: 1440
5. Overlap: 480
6. Window: Hamming
7. Number of Mel Bands: 100
8. Number of MFCC Coefficients: 12
9. DCT: Type 1

Test 7 Results	
Speaker	Prediction
1	1
2	2
3	3
4	4
5	5
6	6
7	7

Figure 3.11

3.8 Test 8 - Notch Filters

We created three notch filters with fc's of 200 Hz, 800 Hz, and 100 Hz with stop band widths of 25 Hz, 200 Hz, and 250 Hz respectively. Our system tested with an accuracy of 100% which was rather unexpected. The system seems to be robust. Perhaps if different rejection frequencies and/or bandwidths were selected the system performance would degrade.

Test 8 Results		
Notch Filter Freq (Hz)	Bandwidth (Hz)	Accuracy
200	50	100%
800	200	100%
1000	250	100%

Figure 3.12

3.9 Test 9 - Replacement

We replaced speaker 1 and speaker 4 with audio samples of Karmvir and Bishane respectively. Karmvir provided a 'Hello' audio sample and Bishane provided a 'Zero' audio sample. We found the system performed well and retained an accuracy of 100%.

Test 9 Results	
Speaker	Prediction
1 (Karmvir)	1
2	2
3	3
4 (Bishane)	4
5	5
6	6
7	7

Figure 3.13

3.10 Test 10 - EEC 201 Student Data

Finally, we tested our system on the audio data of the EEC 201 class. First we trained and tested using only the 'Twelve' speech samples. Then we trained and tested using only the 'Zero' speech sampled. Lastly, we trained and tested using all speech samples. We found the recognition rate for 'Twelve' was quite high at 94.4%. Our system performance degraded significantly and began hallucinating for 'Zero' with an accuracy rate of only 61.1%. For all the speech samples are performance remained the same with a total recognition rate of 77.7% and demonstrated the same behavior: good recognition for 'Twelve' and poor recognition for 'Zero.' This was unexpected as our model was initially trained and developed to perform well on the given speech data which was the word 'Zero.' This was perhaps a result of over fitting to the 'Zero' speech in the given data with poor generalization, or a result of the audio in the class data collected in a less controlled environment. Different hyper parameters in the feature extraction for MFCCs or parameters in the LBG algorithm may have yielded better results. Ultimately the system performed quite well edging out the human performance by approximately 2%. It should be noted that the system was only allowed to 'listen' to each sample once whereas the we could listen to the training samples as many times as we needed. The system also

performed this well on a total of 36 samples whereas we only had to recognize a small set of 8 samples.

Test 8 Results	
Test	Accuracy
Twelve Only	94.4%
Zero Only	61.1%
Both	77.7%

Figure 3.14

4 Unique Efforts

Karmvir: My unique efforts in this project lied primarily within the speech pre-processing portion. Iterating through the corresponding audio files and converting these into amplitude coefficients. Additionally, manually computing the MFCC and cepstrum coefficients for each data was implemented through a function. The manual computation of the MFCC was modeled by Haythem Fayeks who wrote an article explaining speech processing for machine learning. The parameters for finding the MFCC and Cepstrum coefficients was hard coded within the function. In the scope of the project report, Test 1 through 5 were executed by me and plotted accordingly within the report.The implementation of the notch filter function was created by me as well using the scipy signal library in python. Both partners contributed to debugging and interfacing each portion of the code to the entirety of the project. The dimensions and type of variables required a team effort to correctly manipulate the data and implement correctly within the project. After Bishane had implemented the LBG algorithm, we each ran our own individual tests on each data set, modifying the MFCC/LBG algorithm parameters to find the most accurate result.

Bishane: My unique effort in this project was writing the code for the model selection. I implemented two separate LBG algorithms. One was built from scratch and the other was built referencing a report by Orchisama Das of Stanford on speaker recognition. The first model utilized dictionaries to organize data and the second utilized a 3-Dimensional numpy array. I converted both models into a Python class to make the training and prediction processes easier to execute. After testing these models using for loops to sweep through an extensive set of hyperparameters for the MFCC feature extractions and LBG parameters the second model yielded better results and was easier to modify

and troubleshoot. I also discovered a major flaw in our testing. After testing different models for multiple days I discovered Python's 'glob' module imported the speech files in alphabetic/numeric order, so speaker 10 and 11 were directly after speaker 1 and before speaker 2. This explained great confusion and frustration from having low recognition rate initially. In the private sector this would be labeled "a waste of budget" and "grounds for termination" but luckily in this academic setting it was an invaluable lesson with no severe consequences. Lesson learned: always check the data you are importing to ensure it is exactly what you expect. Our older models may have converged and we just had not realized it. In the report I focused on detailing the results of the LBG section of the test. I executed and recorded the results of tests 6 through 10. Overall my partner and I tested the models independently and both produced high recognition rate models. Ultimately, the model I generated generalized better and yielded better results overall.