# BraedenKing-ProjectReport

May 17, 2021

Speaker Recognition using GMM model

Braeden King

1. INTRODUCTION

In this project, an approach to speaker recognition using a Gaussian Mixture Model is explored. Speaker recognition has been a useful tool in the world of forensics for a long time, employing visual and aural recognition techniques. However, "Since 1970s, computer-based or automatic speaker recognition has become mainstreams of the research. Compared to visual and aural speaker recognition, automatic method has an advantage that it enables us quantitative analysis of the speech signals and an explicit presentation of the algorithm." [1]. By using a Gaussian mixture model approach, speakers can be tested against the models to determine whether they are known to the model and which speaker they are.

2. PROCEDURE

This section describes the procedure taken to investigate using Gaussian mixture models for speaker recognition. The speaker recognition system can be broken down into five components, the Dataset, Mel-frequency Cepstrum (feature extraction), Gaussian Mixture Model, Multi-Gaussian log likelihood, and Output, as seen in Fig.1.

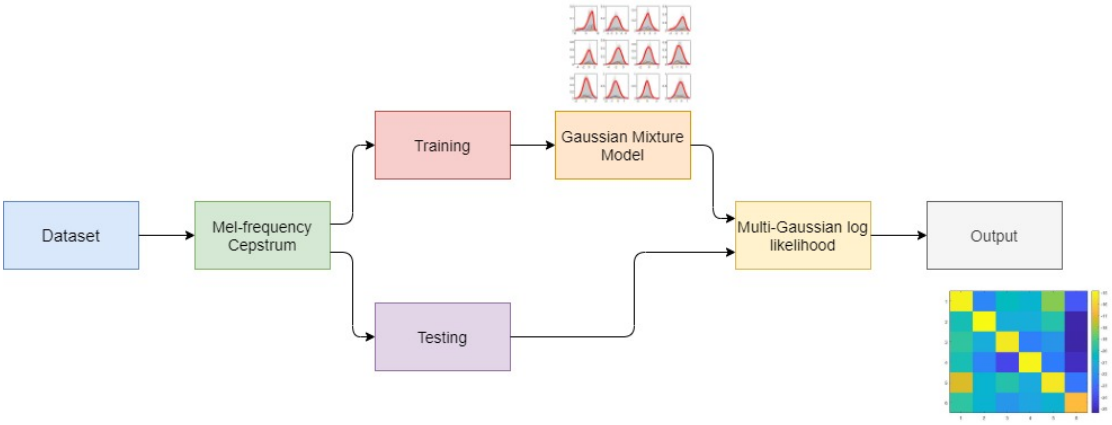



Fig. 1 Block diagram of speaker recognition system

The dataset used in this experiment was grouped into three groups. The first was the dataset used to train the models, the second was the dataset containing the same speakers as the training dataset and was used to test the model, and the last contained speakers unknown to the models that were used as a probe test. These audio samples were taken from the LibriSpeech ASR Corpus, a database of audio samples taken from audiobook readings, that was prepared by Vassil Panayotov and Daniel Povey [2]. For this experiment, there were six speakers with thirty second

audio clips as the training data and ten second clips as testing data, along with three speakers with ten second audio clips for probe testing.

The next step was to perform feature extraction of the audio clips. To do this mel-frequency cepstrum was used. This took the audio clips and windowed them into small snapshots of the whole clip. It then calculated the mel-cepstrum coefficients for each window. The feature extraction was followed by training the Gaussian Mixture Models.

To train the models the features of the training data were passed to the gmm_estimate() function provided. This function calculated the mean, standard deviation, and weights of the model. These values were then used with the features from the testing data to compute the log likelihood that the test features matched those in the model. These log likelihoods were then used to create a confusion matrix to cliassify the system.

The last step was to use the features of the test probe data to validate the system. To do this the model variables along with the features were used to calculate the log likelihood that the test probe data was of someone known to the model.

## 3. ANALYSIS

The resulting confusion matrix from the above procedure can be seen in Fig. 2. As seen in the confusion matrix, the models performed very well with only one source of confusion seen. This appeared when speaker 5 was testing against the first model and resulted in a log likelihood score of -17.16. This was still quite different from the average when testing against a speaker known to the model of -15.37. The confusion matrix of the log likelihood scores when comparing the model to an unknown speaker can be seen in Fig.3. From this matrix it can be seen that the lowest score was around -18 and has an average of -20.78. Using these mean scores along with their standard deviations of 0.607 and 2.024 respectively, the probability density functions of these scores could be plotted, as seen in Fig.4 From this plot the intercept of the two curves was found to be at -16.99, therefore -17 was decided as the threshold for matching scores. Any scores above -17 would be considered a match and any scores lower would be considered non-matches.

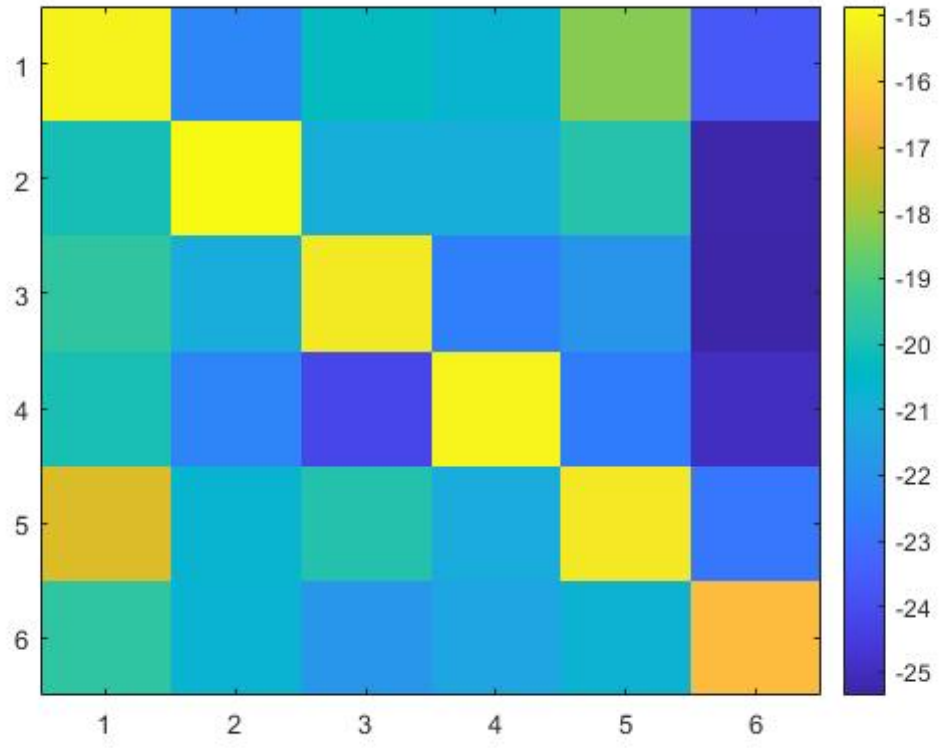



Fig. 2 Confusion matrix of speakers known to models

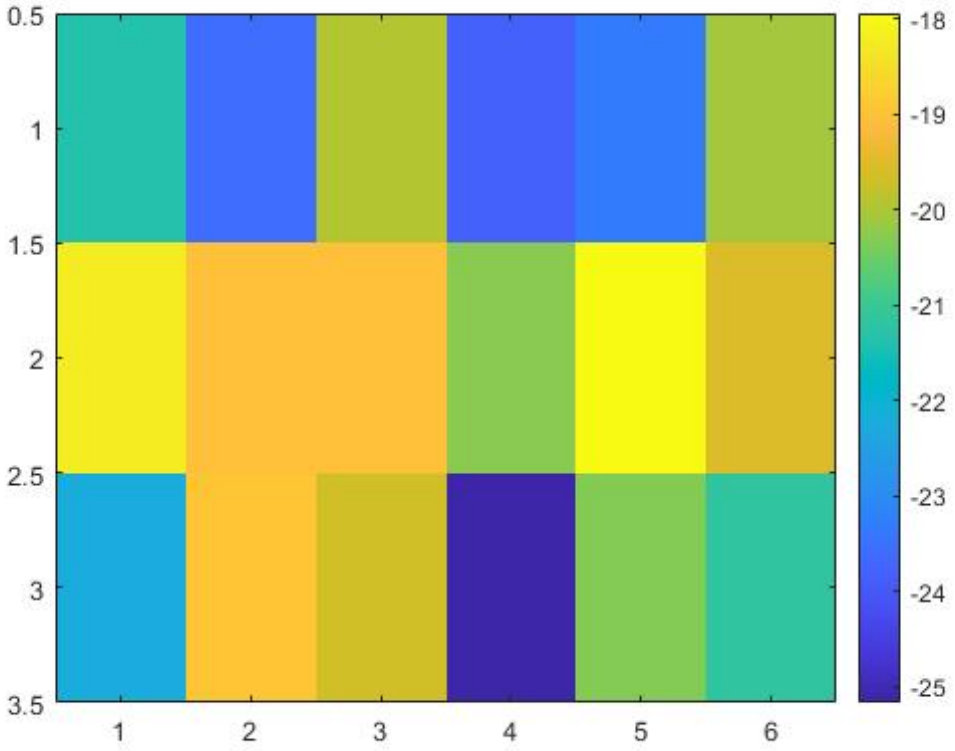



Fig. 3 Confusion matrix of speakers unknown to models

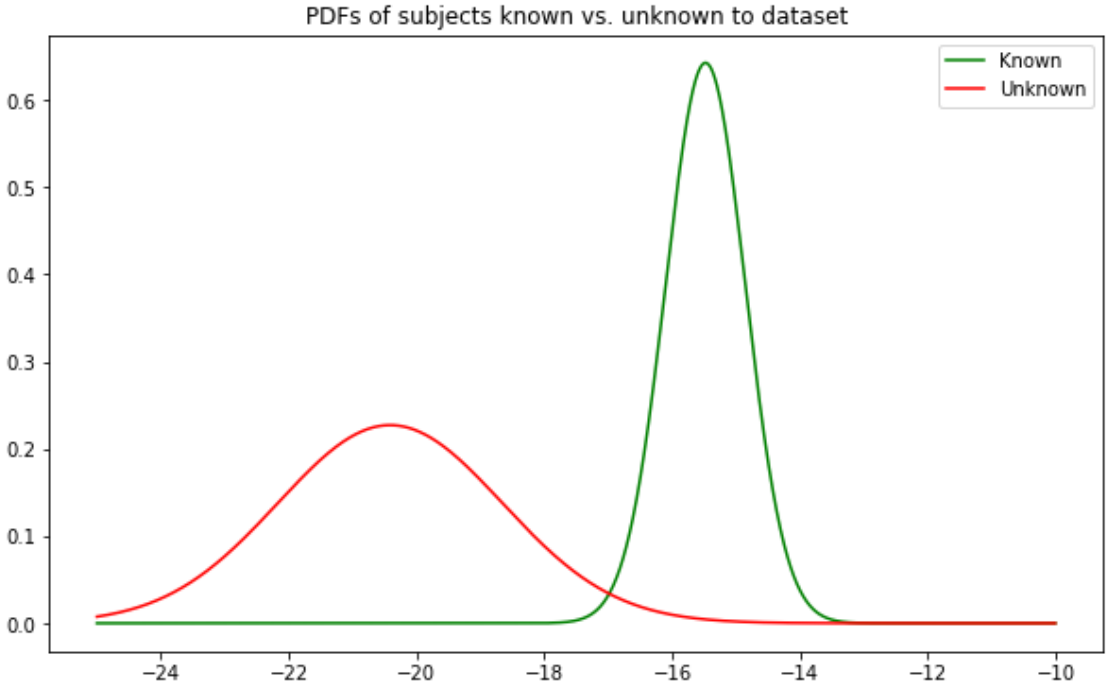



Fig. 4 Probabilty density functions of known speakers vs. unknown speakers

Based on this threshold criteria it was possible to go back and evaluate the performance of the models. When performing classification, all speakers were correctly identified, while no false non-matches were made. This allowed to calculate the false match rate (FMR) and false non-match rate (FNMR).

$$FNMR = \frac{FNM}{(FNM + TNM)} = \frac{0}{(0+30)} = 0.0\%$$

$$FMR = \frac{FM}{(FM + TM)} = \frac{0}{(0+6)} = 0.0\%$$

When performing validation, all speakers unknown to the dataset were correctly evaluated as non-matches. This meant that there were no false matches made. This allowed to calculate the false acceptance rate (FAR) and false rejection rate (FRR) of the models.

$$FAR = \frac{FA}{(FA + TR)} = \frac{0}{(0+3)} = 0.0\%$$

$$FRR = \frac{FR}{(FR + TA)} = \frac{0}{(0+6)} = 0.0\%$$

4. CONCLUSION

   This experiment showed that using Gaussian Mixture Models to perform speaker recognition can be very effective on a small sample of distinct voices. This can be seen by the small amount of overlap present between the two probability density functions. The resulting FNMR and FMR of 0% and 0% show that no misidentification happened with this dataset. The resulting FAR and FRR of 0% and 0% show that the model performed exactly as it was supposed to. This high level of accuracy could be the result of the small sample size used, or the uniqueness of speaker voices used. The next steps in continuing this experiment would be to repeat the experiment while using a larger sample size. In conclusion, using Gaussian Mixture Models to identify speakers has a high degree of success when a small and distinct dataset is used.

5. REFERENCES

[1] A. Neustein, in Forensic speaker recognition, Springer, 2014, pp. 4–5.

[2] V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 2015, pp. 5206-5210, doi: 10.1109/ICASSP.2015.7178964.

[3] S. Yanushkevich (2021). Laboratory Project #6 Speaker Recognition using GMM model in Matlab.

6. APPENDIX

[5]:
```
%project.m
%speaker recognition demo
% written by : Braeden King

GAUSSIAN_COMPS = 10;

disp('---------------------------------------');
```

```
disp('-----Speaker Recognition using GMM-----');
disp('---------------------------------------');

%------------------------Read in training audio--------------------------
[training_data1, Fs] = audioread('speech_dataset\train\422-122949-0010.flac');
training_data2 = audioread('speech_dataset\train\1272-128104-0004.flac');
training_data3 = audioread('speech_dataset\train\2078-142845-0005.flac');
training_data4 = audioread('speech_dataset\train\2902-9006-0005.flac');
training_data5 = audioread('speech_dataset\train\3170-137482-0000.flac');
training_data6 = audioread('speech_dataset\train\5338-24615-0002.flac');
disp('Completed reading training data (Press any key to continue)');
pause;

%-------------------Read in testing audio data-------------------------
testing_data1 = audioread('speech_dataset\test\422-122949-0021.flac');
testing_data2 = audioread('speech_dataset\test\1272-128104-0005.flac');
testing_data3 = audioread('speech_dataset\test\2078-142845-0014.flac');
testing_data4 = audioread('speech_dataset\test\2902-9006-0008.flac');
testing_data5 = audioread('speech_dataset\test\3170-137482-0023.flac');
testing_data6 = audioread('speech_dataset\test\5338-24615-0006.flac');
disp('Completed reading testing data (Press any key to continue)');
pause;


%---------------------Feature extraction---------------------------------
training_features1 = melcepst(training_data1, Fs);
training_features2 = melcepst(training_data2, Fs);
training_features3 = melcepst(training_data3, Fs);
training_features4 = melcepst(training_data4, Fs);
training_features5 = melcepst(training_data5, Fs);
training_features6 = melcepst(training_data6, Fs);
disp('Completed feature extraction of training data (Press any key to␣
 ↪continue)');
pause;

testing_features1 = melcepst(testing_data1, Fs);
testing_features2 = melcepst(testing_data2, Fs);
testing_features3 = melcepst(testing_data3, Fs);
testing_features4 = melcepst(testing_data4, Fs);
testing_features5 = melcepst(testing_data5, Fs);
testing_features6 = melcepst(testing_data6, Fs);
disp('Completed feature extraction of testing data (Press any key to continue)');
pause;


%---------------Training the input using GMM-----------------------------
disp('Training models with the input data');
```

```
[mu_train1, sig_train1, c_train1] = gmm_estimate(training_features1',␣
 ↪GAUSSIAN_COMPS);
disp('Completed Training Speaker 1 model (Press any key to continue)');
pause;

[mu_train2, sig_train2, c_train2] = gmm_estimate(training_features2',␣
 ↪GAUSSIAN_COMPS);
disp('Completed Training Speaker 2 model (Press any key to continue)');
pause;

[mu_train3, sig_train3, c_train3] = gmm_estimate(training_features3',␣
 ↪GAUSSIAN_COMPS);
disp('Completed Training Speaker 3 model (Press any key to continue)');
pause;

[mu_train4, sig_train4, c_train4] = gmm_estimate(training_features4',␣
 ↪GAUSSIAN_COMPS);
disp('Completed Training Speaker 4 model (Press any key to continue)');
pause;

[mu_train5, sig_train5, c_train5] = gmm_estimate(training_features5',␣
 ↪GAUSSIAN_COMPS);
disp('Completed Training Speaker 5 model (Press any key to continue)');
pause;

[mu_train6, sig_train6, c_train6] = gmm_estimate(training_features6',␣
 ↪GAUSSIAN_COMPS);
disp('Completed Training Speaker 6 model (Press any key to continue)');
pause;


disp('Completed Training ALL Models  (Press any key to continue)');
pause;

%----------------Testing the model with known voices-------------------------

%----------------Testing against the first-----------------------------
[lYM,lY] = lmultigauss(testing_features1', mu_train1,sig_train1,c_train1);
A(1,1) = mean(lY);
[lYM,lY] = lmultigauss(testing_features2', mu_train1,sig_train1,c_train1);
A(1,2) = mean(lY);
[lYM,lY] = lmultigauss(testing_features3', mu_train1,sig_train1,c_train1);
A(1,3) = mean(lY);
[lYM,lY] = lmultigauss(testing_features4', mu_train1,sig_train1,c_train1);
A(1,4) = mean(lY);
```

```
[lYM,lY] = lmultigauss(testing_features5', mu_train1,sig_train1,c_train1);
A(1,5) = mean(lY);
[lYM,lY] = lmultigauss(testing_features6', mu_train1,sig_train1,c_train1);
A(1,6) = mean(lY);

%-------------------Testing against the second model--------------------
[lYM,lY] = lmultigauss(testing_features1', mu_train2,sig_train2,c_train2);
A(2,1) = mean(lY);
[lYM,lY] = lmultigauss(testing_features2', mu_train2,sig_train2,c_train2);
A(2,2) = mean(lY);
[lYM,lY] = lmultigauss(testing_features3', mu_train2,sig_train2,c_train2);
A(2,3) = mean(lY);
[lYM,lY] = lmultigauss(testing_features4', mu_train2,sig_train2,c_train2);
A(2,4) = mean(lY);
[lYM,lY] = lmultigauss(testing_features5', mu_train2,sig_train2,c_train2);
A(2,5) = mean(lY);
[lYM,lY] = lmultigauss(testing_features6', mu_train2,sig_train2,c_train2);
A(2,6) = mean(lY);

%----------------Testing against the third model------------------------
[lYM,lY] = lmultigauss(testing_features1', mu_train3,sig_train3,c_train3);
A(3,1) = mean(lY);
[lYM,lY] = lmultigauss(testing_features2', mu_train3,sig_train3,c_train3);
A(3,2) = mean(lY);
[lYM,lY] = lmultigauss(testing_features3', mu_train3,sig_train3,c_train3);
A(3,3) = mean(lY);
[lYM,lY] = lmultigauss(testing_features4', mu_train3,sig_train3,c_train3);
A(3,4) = mean(lY);
[lYM,lY] = lmultigauss(testing_features5', mu_train3,sig_train3,c_train3);
A(3,5) = mean(lY);
[lYM,lY] = lmultigauss(testing_features6', mu_train3,sig_train3,c_train3);
A(3,6) = mean(lY);

%-----------------Testing against the fourth model----------------------
[lYM,lY] = lmultigauss(testing_features1', mu_train4,sig_train4,c_train4);
A(4,1) = mean(lY);
[lYM,lY] = lmultigauss(testing_features2', mu_train4,sig_train4,c_train4);
A(4,2) = mean(lY);
[lYM,lY] = lmultigauss(testing_features3', mu_train4,sig_train4,c_train4);
A(4,3) = mean(lY);
[lYM,lY] = lmultigauss(testing_features4', mu_train4,sig_train4,c_train4);
A(4,4) = mean(lY);
[lYM,lY] = lmultigauss(testing_features5', mu_train4,sig_train4,c_train4);
A(4,5) = mean(lY);
[lYM,lY] = lmultigauss(testing_features6', mu_train4,sig_train4,c_train4);
A(4,6) = mean(lY);
```

```
%------------------Testing against the fifth model-----------------------
[lYM,lY] = lmultigauss(testing_features1', mu_train5,sig_train5,c_train5);
A(5,1) = mean(lY);
[lYM,lY] = lmultigauss(testing_features2', mu_train5,sig_train5,c_train5);
A(5,2) = mean(lY);
[lYM,lY] = lmultigauss(testing_features3', mu_train5,sig_train5,c_train5);
A(5,3) = mean(lY);
[lYM,lY] = lmultigauss(testing_features4', mu_train5,sig_train5,c_train5);
A(5,4) = mean(lY);
[lYM,lY] = lmultigauss(testing_features5', mu_train5,sig_train5,c_train5);
A(5,5) = mean(lY);
[lYM,lY] = lmultigauss(testing_features6', mu_train5,sig_train5,c_train5);
A(5,6) = mean(lY);

%-----------------Testing against the sixth model----------------------
[lYM,lY] = lmultigauss(testing_features1', mu_train6,sig_train6,c_train6);
A(6,1) = mean(lY);
[lYM,lY] = lmultigauss(testing_features2', mu_train6,sig_train6,c_train6);
A(6,2) = mean(lY);
[lYM,lY] = lmultigauss(testing_features3', mu_train6,sig_train6,c_train6);
A(6,3) = mean(lY);
[lYM,lY] = lmultigauss(testing_features4', mu_train6,sig_train6,c_train6);
A(6,4) = mean(lY);
[lYM,lY] = lmultigauss(testing_features5', mu_train6,sig_train6,c_train6);
A(6,5) = mean(lY);
[lYM,lY] = lmultigauss(testing_features6', mu_train6,sig_train6,c_train6);
A(6,6) = mean(lY);

%---------------------Display confusion matrix------------------------
disp('Results in the form of confusion matrix for comparison');
disp('Each column i represents the test recording of Speaker i');
disp('Each row i represents the training recording of Speaker i');
disp('The diagonal elements corresponding to the same speaker');
disp('-------------------------------------------------------------------');
A
disp('-------------------------------------------------------------------');
% confusion matrix in color
figure; imagesc(A); colorbar;



disp('Validation by submitting probe sample');
%------------------Load test probe audio data-------------------------
probe_data1 = audioread('speech_dataset\test_probe\1993-147964-0005.flac');
probe_data2 = audioread('speech_dataset\test_probe\8297-275156-0003.flac');
probe_data3 = audioread('speech_dataset\test_probe\6295-244435-0008.flac');
disp('Completed reading probe testing data (Press any key to continue)');
```

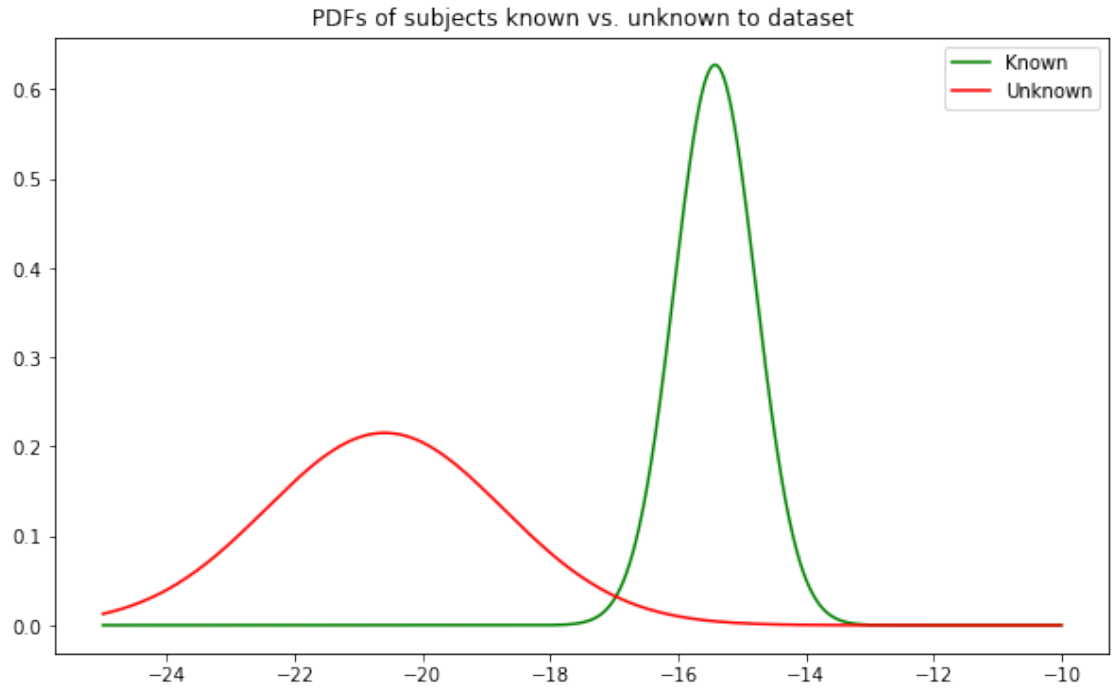
```
pause;

%------------------Test probe feature extraction-----------------------
probe_features1 = melcepst(probe_data1, Fs);
probe_features2 = melcepst(probe_data2, Fs);
probe_features3 = melcepst(probe_data3, Fs);
disp('Completed feature extraction of probe testing data (Press any key to␣
 ↪continue)');
pause;

%-----------------Test test probe against models----------------------
[lYM,lY] = lmultigauss(probe_features1', mu_train1,sig_train1,c_train1);
B(1,1) = mean(lY);
[lYM,lY] = lmultigauss(probe_features1', mu_train2,sig_train2,c_train2);
B(1,2) = mean(lY);
[lYM,lY] = lmultigauss(probe_features1', mu_train3,sig_train3,c_train3);
B(1,3) = mean(lY);
[lYM,lY] = lmultigauss(probe_features1', mu_train4,sig_train4,c_train4);
B(1,4) = mean(lY);
[lYM,lY] = lmultigauss(probe_features1', mu_train5,sig_train5,c_train5);
B(1,5) = mean(lY);
[lYM,lY] = lmultigauss(probe_features1', mu_train6,sig_train6,c_train6);
B(1,6) = mean(lY);

[lYM,lY] = lmultigauss(probe_features2', mu_train1,sig_train1,c_train1);
B(2,1) = mean(lY);
[lYM,lY] = lmultigauss(probe_features2', mu_train2,sig_train2,c_train2);
B(2,2) = mean(lY);
[lYM,lY] = lmultigauss(probe_features2', mu_train3,sig_train3,c_train3);
B(2,3) = mean(lY);
[lYM,lY] = lmultigauss(probe_features2', mu_train4,sig_train4,c_train4);
B(2,4) = mean(lY);
[lYM,lY] = lmultigauss(probe_features2', mu_train5,sig_train5,c_train5);
B(2,5) = mean(lY);
[lYM,lY] = lmultigauss(probe_features2', mu_train6,sig_train6,c_train6);
B(2,6) = mean(lY);

[lYM,lY] = lmultigauss(probe_features3', mu_train1,sig_train1,c_train1);
B(3,1) = mean(lY);
[lYM,lY] = lmultigauss(probe_features3', mu_train2,sig_train2,c_train2);
B(3,2) = mean(lY);
[lYM,lY] = lmultigauss(probe_features3', mu_train3,sig_train3,c_train3);
B(3,3) = mean(lY);
[lYM,lY] = lmultigauss(probe_features3', mu_train4,sig_train4,c_train4);
B(3,4) = mean(lY);
[lYM,lY] = lmultigauss(probe_features3', mu_train5,sig_train5,c_train5);
B(3,5) = mean(lY);
```

```
[lYM,lY] = lmultigauss(probe_features3', mu_train6,sig_train6,c_train6);
B(3,6) = mean(lY);

%---------------------Display confusion matrix-----------------------
disp('Results in the form of confusion matrix for comparison');
disp('Each column i represents the training recording of Speaker i');
disp('Each row i represents the probe test recording of Speaker i');
disp('-------------------------------------------------------------------');
B
disp('-------------------------------------------------------------------');
% confusion matrix in color
figure; imagesc(B); colorbar;

% assigning all scores of test agaisnt same subject to an array
trueScores = zeros(6,1);
for i=1:6
    trueScores(i,1) = A(i,i);
end

% compute mean and std of true scores
trueMean = mean(trueScores);
trueStd = std(trueScores);

% compute mean and std of scores from subject not in dataset
falseMean = mean(B, 'all');
falseStd = std(B, 1, 'all');

fprintf('True Mean: %.3f\n', trueMean);
fprintf('True Standard Deviation: %.3f\n', trueStd);
fprintf('False Mean: %.3f\n', falseMean);
fprintf('False Standard Deviation: %.3f\n', falseStd);

fileID = fopen('results.txt', 'w');
fprintf(fileID, '%f\n', trueMean);
fprintf(fileID, '%f\n', trueStd);
fprintf(fileID, '%f\n', falseMean);
fprintf(fileID, '%f\n', falseStd);
fclose(fileID);
```

```
  File "<ipython-input-5-20d48d0e4d3e>", line 55
    [mu_train1, sig_train1, c_train1] = gmm_estimate(training_features1',␣
 ↪GAUSSIAN_COMPS);

 ↪        ^
SyntaxError: EOL while scanning string literal
```

[4]:
```
# project.py
# Python script that plots PDFs afrer probe testing

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

results_file = open("results.txt", "r")

#-------Import results from matlab program--------
trueMean = float(results_file.readline())
trueStd = float(results_file.readline())
falseMean = float(results_file.readline())
falseStd = float(results_file.readline())
results_file.close()

#------Create normal distribution-----------
true_x = np.arange(-25, -10, 0.01);
false_x = np.arange(-25, -10, 0.01);

tProb = norm.pdf(true_x, loc=trueMean, scale=trueStd);
fProb = norm.pdf(false_x, loc=falseMean, scale=falseStd);

#----------------Plot PDFs--------------------------
plt.figure(figsize=(10,6))
plt.title('PDFs of subjects known vs. unknown to dataset');
plt.plot(true_x, tProb, 'g', label='Known')
plt.plot(false_x, fProb, 'r', label='Unknown')
idx = np.argwhere(np.diff(np.sign(tProb - fProb))).flatten()
idx = idx * 0.01 - 25
plt.legend()

print('Intercepts of PDF curves: ', idx)
```

```
Intercepts of PDF curves:  [-16.98 -12.5 ]
```


PDFs of subjects known vs. unknown to dataset
Known
Unknown
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−24
−22
−20
−18
−16
−14
−12
−10

[ ]: