EE513 Final Project: Speaker Recognition

Author: Daniel Palmer

Date: 5/6/20

**Abstract**

For the duration of this project students worked with samples of various people saying the word “tyra”. There were 5 different people, each giving 50 samples in total, and it was our goal to create a matlab script that would be able to differentiate the two speakers using what we have learned thus far. The data presented in this paper is a result of applying mel cepstrum feature extraction to a windowed set of data and training that data in order to determine the classifier for my data set. This classifier would be and important part for my testing phase to determine the confusion matrix which was then used to determine the performance of my script.

**1. Feature Selection**

The feature selection was one of the first steps in completing this script. For this part of the project I used the Mel cepstrum feature extraction, which was used in parallel with my window function and the trim function provided by the professor. For each segment of the tapered signal, the MFCC was calculated and stored in an accumulated vector. After looping through the whole segment, the deltas and the delta deltas were computed. The next step was selecting the features that are the most unique to the signal, and that was done by piping the ff1,deltas,and delta deltas into a feature vector for the first and second half of the signal. These respective plots are shown in figures 1 and 2. For the second figure, I actually additionally chose feature 14 and 20 to represent it to make things a little more accurate. In total I used 7 features in my final vector to represent the signal.

  
Figure 1. First Half Feature Selection

  
Figure 2. Second Half Feature Selection

**2. Classifier Structure**

Since I used 7 features, the means and inverse covariance matrices in my classifier ended up being 7x7.

d11=(c1tst(:,kt)-m1)'\*invc1\*(c1tst(:,kt)-m1);   
d12=(c1tst(:,kt)-m2)'\*invc2\*(c1tst(:,kt)-m2);

In this situation, if d11 were smaller than d12, then we would accumulate the correct confusion matrix by 1. If d12 were smaller than d11 we would accumulate the incorrect confusion matrix by 1. This would then repeat for the second class.

**3. Resampling method**

The resampling method used was bootstrapping, and I used 20 samples. This method seemed to give better results for me, and if I hadn’t done this last minute I may have tried to do a hybrid to see the results of that.

**4. Performance summary**

Overall the script performed well, and the speakers were recognized for the most part above 90 percent with all the cases I tested. For most tests I ran the accuracy had a mean of .95 with a standard deviation of .05. The case I ran with the classifier ended up having a mean accuracy of .94 and a standard deviation of 0. The example tables below show the overall performance summary of just a random test case, and then a test case with the classifier in the folder with this code.

Figure 3. Performance Summary Random Test

****Figure 4. Performance Summary W/ Classifier

**Issues/Bugs**

One issue I noticed early on in testing was with speaker file 2 and 5. This issue traced back to the trim function were it kept saying that it was trying to trim out of bounds. When looking into the issue I noticed it was trying to index the beginning of the signal around -499, which was the issue at hand. This calculation happened within what seemed like a last ditch effort loop for when the trim function may have gone haywire. At the point in the script the trim index was set to one, and it was going through a calculation that ended up subtracting 500 from it. Instead of trying to debug the issue, I noticed it only happened with speakers 2 and 5, so I just avoided using those files and moved on.