Lab 3 Report

Deniz Alpay
1650403
deniza@uw.edu

Aidan Johnson
1431797
johnsj96@uw.edu

1 May 2018

EE 443, Spring 2018
Design and Application of Digital Signal Processors
Department of Electrical Engineering
University of Washington

Problem 1: Mel-Frequency Cepstrum Coefficients (MFCCs)

The MFCCs were first computed as the features for the audio classification that follows. The MFCCs were computed using a 21 ms frame window with a 11 ms frameshift where a Hamming window is used to window each frame. A pre-emphasis coefficient of 0.97, 48 filter banks, and a lifter parameter of 22 were used. The resulting MFCCs had 13 coefficients and 380 frames for every ~4 seconds of audio (note each audio file was slightly longer than 4 seconds). To improve the robustness of the classification, the audio was normalized to have a similar amplitude range. The plots below include the audio signal in the time domain and MFCCs for the signal at a frequency range of 100 to 10,000 Hz.

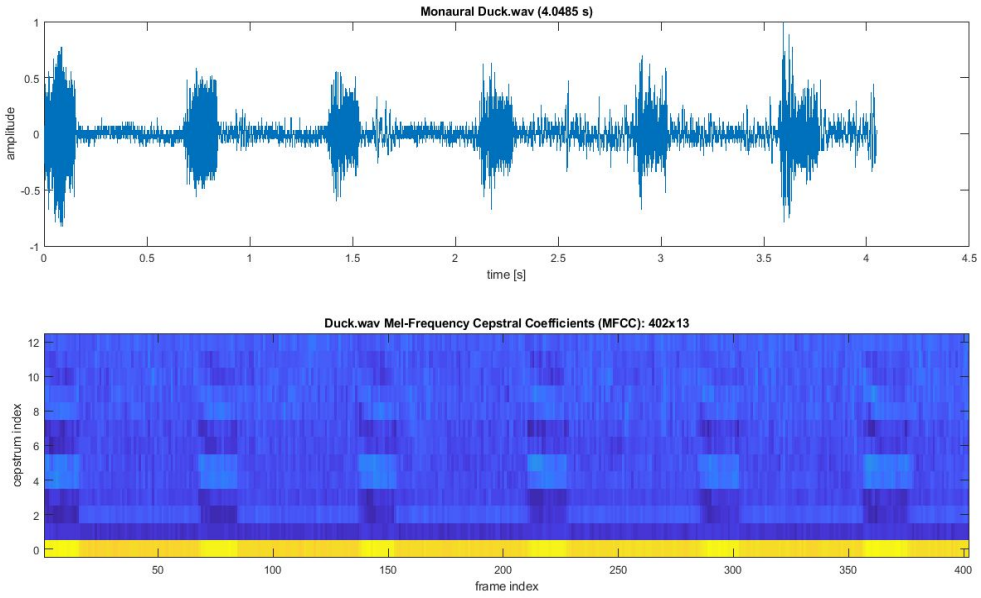

Figure 1: Top: The monaural waveform of the duck sound. Bottom: The mel-frequency cepstrum coefficients of the same waveform.

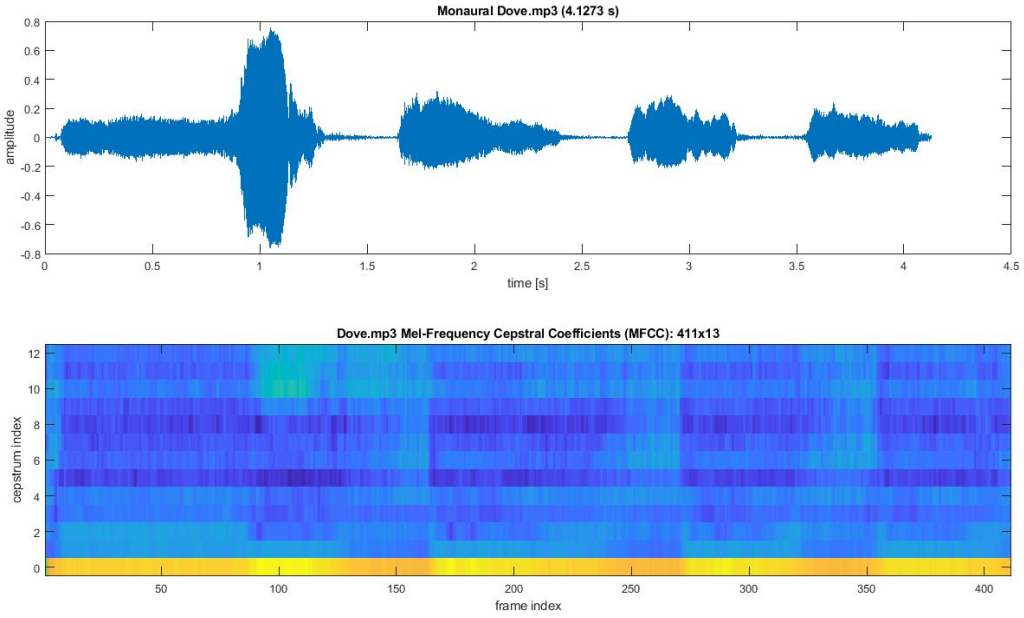

Figure 2: Top: The monaural waveform of the dove sound. Bottom: The mel-frequency cepstrum coefficients of the same waveform.

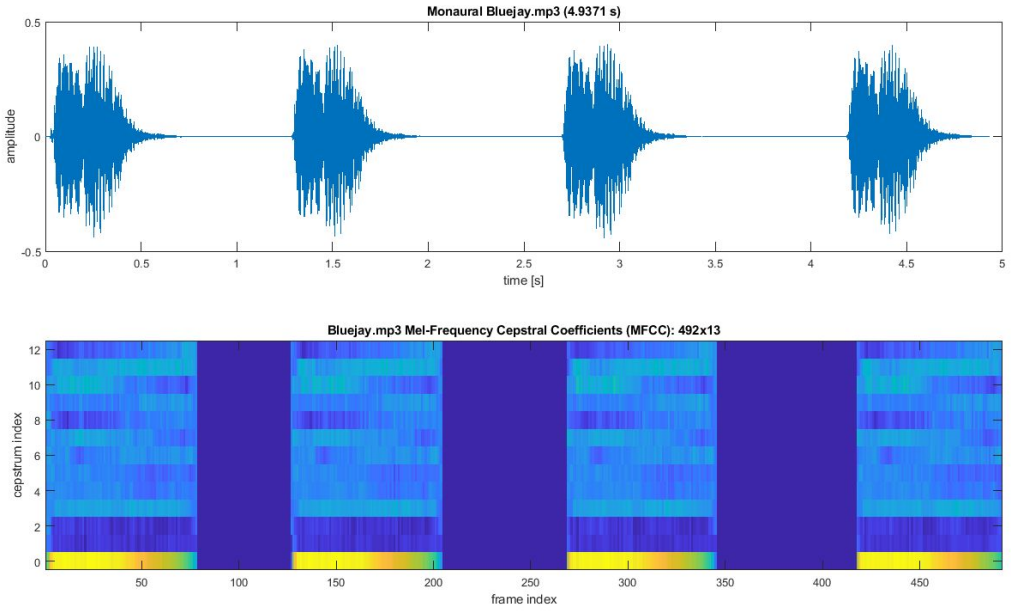

Figure 3: Top: The monaural waveform of the bluejay sound. Bottom: The mel-frequency cepstrum coefficients of the same waveform.

Problem 2: Training GMM for Bird Sound Classification

After computing the MFCC features from problem 1, the Gaussian mixture model was used to cluster the bird sounds into their respective classes. The function `fitgmmdist(X,k)` in MATLAB was used where X is a matrix consisting of the column concatenation of the MFC coefficients and k = 3 to denote the three classes we wish to cluster the data into. A MATLAB function was written to compute the MFC coefficients given the input signal file name, and optional parameters for plotting the figures (Figures 1-3). The returned MFCCs were then used to train the GMM using the `fitgmmdist` function. The model was then saved so that the model parameters could be sent to the LCDK. Due to difficulties with UART on the Department computers, despite implementing correct algorithm for sending the parameter via UART, the parameters were printed to a `.txt` file in the C-language array format using a custom function. (Perhaps insufficient administrator rights to access the COM port.) These three arrays (one for the means with a length of 39, another for the covariances with a length of 39, and the third for the weights corresponding to each of the three classes) were then applied to Problem 3. If UART had worked efficiently, the parameters could be serially sent to the LCDK.

Problem 3: Classification of Bird Sounds Using GMM

Once the GMM parameters were computed, the the GMM parameter were transferred to the LCDK to classify bird sounds received from the line input of the LCDK. The GMM parameters were transferred by formatting the parameter output in MATLAB such that they could be pasted into `main.c`. A short time Fourier Transform of 256 samples with a 12 kHz sampling rate (specified in the configuration header) was computed from the audio received from the line input. These 256 samples correspond to the 21 ms frame duration that was used to compute the MFCC. The magnitude of the spectrum was used to compute the MFC coefficients, which are then compared to the GMM parameters to evaluate which bird sound class the frame is most similar to.

When the bird sounds are played on repeat, the system evaluates each 21 ms frame to produce the probability that the frame is in each of the three bird classes. We noticed that when the volume of the audio is very low, the classifier identifies the the audio as a bluejay sound. This matches our expectations because the blue jay training audio is quiet half of the time so a lack of noise is most similar to the bluejay sound. While the classifier accurately classifies the blue jay and dove sounds, the duck is sometimes classified as the dove sound. This may be because the duck audio has more noise and doesn't have as distinct features as the other bird sounds.

However, after further troubleshooting and deliberation with the TA, it was discovered that the MFCCs generated by MATLAB are not compatible with how the MFCCs computed in the `libmfcc` library due to how they are computed. Nonetheless, our program for printing the score of the MFCCs of the sampled signal (probability of the sampled signal belonging to one of the three classes) is correctly implemented. This could be remedied by computing the MFCCs of the training data audio using the C library instead of in MATLAB. This was confirmed to be the case by predicting the predicted class for a training data sample given the MATLAB GMM of the training data. The MFCCs computed in the C library were not correctly identified

by MATLAB's `[label, score] = predict(GMModel, mfcc)` function. In short, our implementation would correctly function if correct MFCCs were given to the program.

Problem 4: Training SVM for Bird Sound Classification

In contrast to the GMM which is an unsupervised learning method that clusters the bird sound features, the SVM is supervised learning method that classifies the bird sounds by their features. In other words, in SVM the training data labels are provided, while in GMM no label is provided to the model. As a result of the SVM being a supervised learning method, SVM was trained using the MATLAB function `fitcecoc` with the same MFCCs computed in Problem 1 with labels to denote what bird each frame corresponds to.

Problem 5: Classification of Bird Sound Using SVM

After computing training the SVM, the parameters were directly hard coded into the `main.c` file to classify the bird sounds received by the line input of the LCDK. Like in Problem 2, the SVM parameters (number of classes, the bias, the number of support vectors, the support vector coefficients, and the support vectors) were printed by MATLAB so they could be manually copied to the `main.c` file. However, they differed in the format of the array. Instead of three separate arrays, all the parameters were serially aligned in an array with a length equal to the sum of two times the number of classes, and total number of support vectors multiplied by 14 plus 1. Additionally, just like the GMM classification problem, a short time Fourier transform of 256 samples on a 12 kHz sampling rate was computed. The 256 samples of each frame correspond to the 21 ms frames used in the MFCCs. The magnitude of the frame spectrum is then used to compute the MFCCs of the frame, which is then input to the SVM classifier. The SVM classifier then evaluates the MFCCs to determine which class boundaries the frame's features fall into.

With the MFCCs for a single frame, the SVM classifier was computed for each class. A non-negative classifier indicates the most likely class assignment. We then implemented the following discriminant formula for the classifier in code:

$$g(\bar{x}) = \sum_{i \in |SV|} \alpha_i \bar{x}_i{}^T \bar{x} + b,$$

where $|SV|$ is the number of support vectors, $\bar{x}$ is the MFCCs for the test/sample input, $\bar{x}_i$ is the ith support vector, b is the bias, and α_i is the ith support vector coefficient. This was computed for each of the k=3 classes (for each bias and for all the class support vectors).

We ran into trouble with computing the classifiers because of insufficient memory of the LCDK to adequately allow dynamic memory allocation with `malloc`. This also plagued the `svm.c` library. In an attempt to solve the dynamic memory instability, we implemented a version of the SVM algorithm that read the training data parameters and then computed the predicted class that mostly used static memory allocation (i.e., declared arrays). Static memory could not be only used because it requires constant parameters for assigning memory space. The number of support vectors per class is not the same and not constant, and therefore could not be

used, preventing us for using only static memory. Nonetheless, dynamic memory allocation would normally be sufficient, if not advised, if the LCDK had an appropriate size of memory.

Because of these numerous difficulties that affected all groups in this class, we could not compare the performance of the SVM model with the GM model. These difficulties also revealed an important limitation of the particular LCDK boards we have been provided. Nonetheless, this lab provided useful experience with implementing machine learning algorithms in C for hardware, and most certainly with dealing real-world engineering problem solving challenges.