Assignment: Homework 5

What to Hand In

1. Your code with appropriate comments and a readme file. You are responsible for making us understand and know how to run your code.
2. A report with answers, figures, and explanations.
3. Any sound files requested in the problems.

How to Hand It In

1. Put all of your files in one folder. Compress this folder and name it <firstname>_<lastname>_HW5.zip. For example, "Zhiyao_Duan_HW5.zip".
2. Submit via blackboard to the appropriate entry.

When to Hand It In

It is due immediately at the beginning of the day (i.e., 0:00AM) specified on the course calendar. **Late assignments will receive a grade deduction of 20% of the full grade each day.**

Problems (10 points in total)

In this homework we will work on two audio machine learning problems: 1) keyword matching based on Dynamic Time Warping, and 2) sound classification model training and evaluation. Both tasks will be based on the MFCC feature representation of audio.

1. (1 points) Try out and visualize the MFCC features
   Record several typical phonemes, such as "ah", "i", "ts", etc. with your voice. Save the wave file into the folder "`data/p1/`" with sampling rate of 16K Hz and mono channel. Crop the waveform to remove silence and normalize the volume (e.g., normalize the overall RMS to a certain value). Then write code in "`p1.m`" to read the waveforms and extract their MFCC features. A simple implementation of the MFCC feature extraction code is already given as "`my_mfcc.m`". Please read and understand it before you use it. Then plot the feature (using "`imagesc`" function) and compare them. Can you see differences in this feature visualization between different phonemes?

Now choose 2 of the following 3 problems to solve.

2. (4.5 points) Dynamic Time Warping (DTW). Finish the code in the given function "`my_dtw.m`" and then test it in keyword matching.
   a. (2.5 points) Implement the DTW algorithm.
      Here are the reference steps.
      (1) (0.5 points) Calculate the local distance matrix using Euclidean distance, where each entry is the distance between two elements from the two input sequences.
      (2) (1 point) Calculate the accumulated distance matrix, where each entry is the sum of its local distance value and the minimum of the 3 neighboring accumulated matrix values (refer to lecture slides).

(3) (1 point) Backward tracing to find the warping path. From the end point of the accumulated distance matrix, move backward until reaching the beginning point. Each move should select the minimum neighboring entry.
(4) Calculate the accumulated cost by summing up all of the local distance values along the warping path.

b. (1 points) Here is a toy example of two sequences, `x1` and `x2`. Both are simple 1D signals, and we want to measure their similarity/distance. If we calculate their distance as the sum of the pair-wise absolute differences over time, what is the distance? If we calculate their distance using the DTW accumulated cost, what is the distance? Write your code in "`p2_b.m`" to implement this, and then plot the local distance matrix and global distance matrix (using "`imagesc`" function), with the warping path laid on top of it (using "`plot`" function).

```
x1 = [1,2,3,4,5,6,7,8,9,1,2,3,4,5];
x2 = [1,1,1,2,3,5,5,6,8,9,1,4,3,5];
```

c. (1 points) Now let's try some real examples. Record several keywords using your own voice, such as "Apple", "Strawberry", "Banana", etc. Save the waveform of each keyword as a separate file into the folder "`data/p2/`" with a sample rate of 16 kHz and mono channel. Then pick one keyword (such as "strawberry") from them and record and save the file again. Crop the waveforms to just reserve the non-silent frames before saving. Then write your code in "`p2_c.m`" to calculate the MFCC feature for each of the wave files, and apply DTW algorithm to calculate the accumulated cost for "strawberry - strawberry", "strawberry - banana", "strawberry - apple". Visualize the local distance matrix with the warping path, and compare. Is the accumulated cost for the same pair of keyword the minimum one?

3. (4.5 points) Sound Classification. This problem is a simple machine learning task based on a tiny animal sound dataset. You will use the MFCC feature and a built-in classifier in MATLAB.
   a. (1.5 points) Prepare the training data. In the folder "`data/p3/train/`", there are four animal categories, each with 12 wave files as the training data for that animal sound. Write your code in "`p3_train.m`" to read all of the wave files and extract the MFCC features. (You may find the "`dir`" function useful.) Then concatenate all of the feature matrices into a matrix `x_train`, with the size of `[n_frame*dim]`, where dim is the feature dimension, and `n_frame` is the total number of frames of all the sound files. Create the category label at the frame level as `t_train`, with the size of `[n_frame*1]`, where all frames from the same waveform have the same label.

      Hint:
      1) Removing the silent (or low-volume) frames can help to improve the classification results later.

2) In the MFCC feature, the first dimension represents the signal energy, based on which you can set the threshold.
3) You can use numeric values for each animal category, such as "1" for "cat", "2" for "dog", etc.

b. (1 point) Train the classifier. Continue your code in "`p3_train.m`". Feed your training data and category labels into the classifier. You can use the "`fitcecoc`" function to implement the training process. The output is the trained classifier model. Save your trained model into disk as "`my_model.mat`" (using the "`save`" function).

c. (1 point) Evaluation. In the folder "`data/p3/test`", there are several sound files for the four animals as a test set. Write your code in "`p3_test.m`", where you apply the same operation as in Step a to prepare the test data as `x_test`, `t_test`. Then load your trained model (using the "`load`" function) and input your test data `x_test` using the "`predict`" function, to get the output labels `y_test`. Now you have the model output `y_test` and the ground-truth label `t_test`. Now finish the code in the given function "`my_evaluation.m`", where you write an evaluation code to generate the confusion matrix and print it out. Please make some conclusions on the confusion matrix.

d. (1 point) Live test. Write a real-time audio recording script using the built-in "`audioDeviceReader`" object in MATLAB. (For older MATLAB version refer to "`dsp.AudioRecorder`" as similar function.) A sample code for live recording is given as "`p3_test_live.m`", where you can start with. To complete the real-time sound classification task based on this code, you should load the trained model, and at each coming frame apply MFCC feature extraction, and feed the feature into model. When the recorder runs, mimic the animal sound using your voice, and display the classification result for each frame in real-time. Which animal is easier to mimic?

Hint:
1) Skip the live frames with weak volume.
2) From the captured live frame, you must apply exactly the same process and parameter settings for feature extraction as those in "`my_mfcc.m`" before feeding the frame into the classification model. "`my_mfcc.m`", however, cannot be directly applied in live test, as it contains the function of STFT and performs feature extraction for a batch of frames. Therefore, you need to understand the feature extraction process and implement/integrate the useful part of "my_mfcc.m" into "`p3_test_live.m`", to be able to extract features in each incoming audio frame.

4. (4.5 points) Predict the reverb decay factor using linear regression. Reverberation is the process of multi-path propagation of a sound from its source to one or more receivers. A reverberation is created when a sound or signal is reflected by the surfaces of objects and then decay as the surfaces absorbed the sound in the space. This problem is a straightforward machine learning task based on a tiny reverberation signals dataset. You will use the MFCC feature and the linear regression algorithm.

   a. (0.5 points) Get familiar with reverberation. Generate reverberated signals for "`violin.wav`" (using `reverberator` in Audio Toolbox of Matlab) with DecayFactor in 0:0.02:0.98 and 0.05:0.1:0.95 and save them into audio files. You can listen to some of them to understand how the decay factor affects the reverberation effects.
   Hint: The generated signal has two channels, please use the first channel only. Normalize the audio signal to avoid audio clipping when writing into files and specify the value of DecayFactor in the filenames.

   b. (1.5 points) Implement the linear regression algorithm. In this step, we are going to use 0:0.04:0.96 as the training set and 0.05:0.1:0.95 as the test set. The training set and test set should have no overlap files. X matrix represents the features for all training samples, and the Y matrix represents the labels for all samples, which is the decay factors in our case.
   Here are the steps:
   (1) For each file, extract the MFCC feature for all frames and calculate the mean and variance. Concatenate the mean vector, variance vector and an element 1 to form a feature vector. The reason to include 1 is to calculate bias.
   (2) Concatenate these feature rows to get the X matrix.
   (3) Compute the regression coefficients from X and Y.

   c. (1 point) Evaluate the results. Here are the steps:
   (1) Use the calculated weight vector to predict the decay factors in the test set. Compute the mean square error (MSE) metric between predicted Y_hat and ground truth Y to see how your trained model fits the test data.
   (2) Change the training set to be 0:0.02:0.98, repeat part b, and evaluate the mean square error (MSE) metric. Does it get improved? Explain the reason.

   d. (1 point) Robustness to other factors. Change the other factors (e.g., PreDelay, Diffusion, WetDryMix, etc.) of the reverberator to nondefault values and apply it to re-reverberate the signals in the test set to create a new test set. Use your trained weight vector in part c2 to predict the decay factors and evaluate the result. Are you satisfied with the result? If not, can you think of a way to improve the robustness? Explain it.

   e. (0.5 points) We used the same file in training and testing. What if we use a different file in testing? Does your model still work? If not, what do you think is the problem?