Homework 4 Gowajee

Instructions

Answer **all** the questions and upload your answers to courseville. Answers can be in Thai or English. Answers can be either typed or handwritten and scanned. No need to submit your code.

Academic integrity and plagiarism

You are expected to do this homework yourself. **Copying other people's codes or solutions will be treated as cheating on an exam.** You are encouraged to discuss the problems and concepts in the homework with your peers, but they are not allowed to show you their written solutions. Afterwards, you must write the code and solution yourself. If you did so, write down the name of the person you discussed with.

MFCC

In this section, we will study the steps necessary to construct Aurora advanced front end MFCC features. As we learned in class, Mel-Frequency Cepstral Coefficients (MFCCs) are a popular representation of speech signal due to their ability to capture the spectral envelope of individual frames of speech in a compact representation. In this section, we will study the various steps involved in converting a waveform into MFCC feature vectors. Specifically, we will implement the speech recognition front-end feature extraction algorithm as defined by the Aurora Advanced Front End (AFE) standard using Octave. For additional information on a specific Scilab command,run *help <command>* from within a Octave window.

1. Audio waveforms are commonly stored in the WAVE audio file format (.wav) using pulse-coded modulation (PCM), where the audio signal is represented digitally by sampling the waveform at a fixed sampling rate and linearly quantizing the resulting values.

   Load *gas_station.wav* using the function *wavread*. What is the length of this wave file in milliseconds?

2. The information of the speech signal is in the fluctuation of the signal, and we can ignore the DC offset. The AFE standard applies the following filter to remove the DC offset from the waveform:

   $s_{of}[n] = s_{in}[n] - s_{in}[n-1] + 0.999 s_{of}[n-1]$

   However, since we are processing the entire waveform at once instead of in real-time as it is being recorded, we can directly compute the DC offset of the entire utterance. What is the DC offset of this utterance? Remove the DC offset from the waveform.

3. At a 16 kHz sampling rate, the AFE standard computes MFCC coefficients from 25ms frames of windowed speech with a frame shift of 10ms. To utilize all samples in the waveform, the last frame may be padded with zeros. What is the frame length and shift in samples? For this waveform, how many frames are there?

4. One of the features included in the AFE standard is the log frame energy, defined as:

   $logE = max(-50, ln(\Sigma_{i=0}^{FrameLength-1} s_{of}[frameStart + i]^2))$

   where s_{of} is the offset compensated waveform within the frame. Note that $logE$ is floored at -50 to support flat line waveforms. Compute $logE$ of frame 50 (one-indexed).

5. Recall that for periodic speech signals, the source at the glottis falls proportionally to $1/f^2$ at higher frequencies. The decrease in energy is partially compensated by a high frequency boost factor of f from the radiation characteristics at the lips. To compensate AFE applies the following filter:

   $s_{pe}[n] = s_{of}[n] - 0.97 s_{of}[n-1]$

   where s_{pe} is the signal after pre-emphasis and $s_{of}[-1] = 0$. Plot the magnitude spectrum at frame 50 before and after the pre-emphasis. Is this a highpass or lowpass filter?

6. To compute the spectral representation, we first apply a Hamming window to each frame to smooth out the signal at the frame boundaries. We use the Fast Fourier Transform (FFT) to compute the discrete Fourier transform and obtain a frequency representation of the signal. Since FFTs with lengths that are powers of two can be computed more efficiently, we will zero pad the windowed frame to 512 samples and compute a 512-point FFT. Hint, use the command *hamming* and *fft*.

   When computing the Fourier transform of a signal with real values, the resulting spectrum generally consists of complex values. Instead of representing the energy at various frequencies using real and imaginary components, it is often more useful to consider the magnitude and phase of the resulting spectrum. For speech signals, we usually only use the magnitude information.

   For real-valued input signals, the magnitude of the resulting Fourier transform exhibits symmetry about $\omega = 0$. Thus, only the first 257 points of the FFT are needed to represent the signal. What frequencies do spectrum[1] and spectrum[257] correspond to? Plot the magnitude spectrum at frame 50 and label the x-axis in Hz.

7. The Mel scale is an approximation of the human auditory system's frequency response. Instead of viewing the frequency axis of the power spectrum on a linear scale, we can compute the energy within a set of 23 logarithmically scaled frequency bands in order to characterize the spectral envelope. The energy within each frequency channel is obtained via

a weighted sum of the linear frequency spectrum magnitudes over the frequency range of the band, roughly corresponding to a triangular filter. Note that this weighted average is computed on the magnitude spectrum for each filter bank instead of the full complex value for computation efficiency. To help you visualize the Mel-frequency filter banks, run the following in Octave:

load('mel_filters.mat')
*plot((0:256)'/256*8000, mel_filters)*
title('Mel filter banks')
xlabel('frequency (Hz)')
ylabel('weight')

What is the minimum and maximum frequency covered by this set of filter banks? Apply the filter banks to the spectrum at frame 50 and compute the energy within each band.

8. The Mel-Frequency Spectral Coefficients (MFSCs) are defined as the log of the filter bank energies. Again, the resulting values are floored at -50. From the MFSCs, we can calculate the Mel-Frequency Cepstral Coefficients (MFCC) by performing the following Discrete Cosine Transform (DCT)

   $C_s = \Sigma_{j=1}^{23} MFSC_j cos(\frac{\pi s}{23}(j - \frac{1}{2}))$

   Notice how the equation does not involve any imaginary numbers. The DCT is also more computation efficient than the FFT. Since the cosine terms can be pre-computed, the DCT requires $O(n)$ operations, while the FFT requires $O(nlog(n))$

9. The Aurora AFE uses the log energy and the first 13 MFCC coefficients as features for each frame. Assemble the above steps and implement a Scilab function *compute_mfcc* that takes an input wave file path, computes a matrix of feature vectors across all frames, and return the matrix. The feature matrix should have each column correspond to $[logE, C_0, ..., C_{12}]$ for each frame. So the output matrix would be of size 14x(number of frames)

   To help verify your implementation. Compare against the MFCC provided in *gas_station_mfcc.mat*

10. Which MFCC dimension is logE most similar to? Explain. Based the derivation of C_1, what can we tell about the range of C_1 for vowels and fricatives? Verify this against the computed MFCC coefficients for the phrase “gas station”. Hint, look at the DCT equation for C_1 and C_0.

11. In this specification, give the number of dimensions in one frame for the following features

    (a) time signal

    (b) Spectrogram

(c) MFSC

(d) MFCC

Cepstral Mean Normalization (CMN)

We have learned in class that the effect of microphones or room echoes can be modeled as a LTI system with a impulse response $h[n]$. If $s[n]$ is the speech signal that comes out of the mouth, the recorded signal, would be $s[n] * h[n]$. The echo effect is usually called “Reverberation” by speech scientists. The effect of $h[n]$ on the recorded signal is usually called “Reverberation noise” or “Convolutive noise.” In this section, we will learn one way to remove convolutive noise in MFCC.

Take a look at the file *gas_station_echo.wav.* gas_station_echo.wav is a simulated recording where the microphone is placed in bigger room with more echos. Compare the two wav file by listening. Can you hear a big difference? Look at the spectrogram of the original and corrupted version, can you notice any difference?

1. Compute the Euclidean distance between MFCCs generated from gas_station and gas_station_echo. You can treat all the frames together as one big vector. The number is quite big even though you barely hear or notice any difference between the spectrograms.

2. We learned that convolution in the time becomes addition in the cepstral domain.

   $s[n] * h[n] \leftrightarrow \tilde{s}[n] + \tilde{h}[n]$

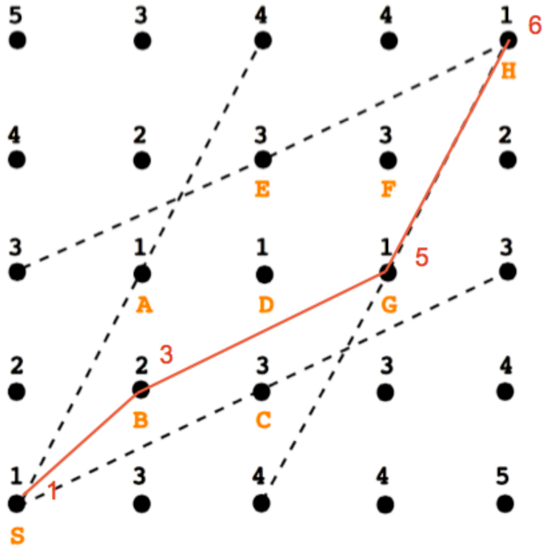
   where $\tilde{s}[n]$ and $\tilde{h}[n]$ are the corresponding cepstral transform. If we can know $\tilde{h}[n]$, it can be easily removed by subtraction. However, it is hard to estimate $\tilde{h}[n]$ individually, since s[n] also has convolutive effects from the vocal tract which we do not want to remove. If we assume that $\tilde{h}[n]$ is constant over time, we can remove the effect of $\tilde{h}[n]$ by removing the DC offset (the mean) of each MFCC dimension calculated over the whole recording. Calculate the Euclidean distance if you remove the mean of MFCC from each recording. Is it smaller of bigger than the previous question? The method is called Cepstral Mean Normalization (CMN).

   Note that CMN not only removes the effect of the microphone from the MFCC features, but also the DC offset in the original features. Thus, if two utterances only differ in the offset of their MFCC features, CMN will remove this sole distinguishing feature as well. Fortunately from a speech recognition perspective, this situation is unlikely to occur.

DTW theory

Calculate the DTW path of the following template and test pairs. Describe the best path and the distance at each point of the path.

In class, we can find the best warp as shown below:



So the answer would be
Ans: (1,1) - 1, (2,2) - 3, (4,3) - 5, (5,5) - 6

1. Template : [1 5 7 9 3 4 2 2 1], Test : [2 8 6 9 7 3 4 1]. Use type II local constraint.

2. Also apply a global constraint of $|x - y| < 3$

3. Now the features are two dimensional. Use Euclidean distance for the distance of each frame. Use type II local constraint with no additional global constraint.

   Template : [(1,-1) (3,3) (8,7) (4,-5) (5,4)]

   Test : [(2,0) (2,4) (4,2) (4,-4) (6,3)]

4. DTW can be considered as a shortest path search on graphs. Is DTW as we did in class a breadth first search or depth first search?

Keyphrase detection with DTW

In this section we will implement a simple keyphrase detector using DTW. Keyphrase detection is a method for detection whether a keyphrase is spoken in a long utterance (or an infinite stream). This has been used in continuous listening devices such as “Google Home” and “Amazon Alexa.”

First, download the starter code and data from CourseVille. Which has the following files

- **compute_dtw.m** A blank function file for computing dtw distance.

- **slidedtw.m** Given an input template this function performs keyphrase detection on a longer file by doing multiple DTW matchings. A DTW is computed every 25 frames.
- **computeall.m** This function calls *slidedtw.m* on all test utterances.
- **computeRoC.m** This function uses the output from *computeall.sce* to compute the error rates.
- **data.mat** A data file containing MFCCs for the template and test utterances.

We have learned in class a way to match one word with another using DTW. However, the method is slightly different when we use it to detect a phrase in an infinite stream. This can be done by computing multiple DTWs using different starting points in the test utterance. In other words, we compute $DTW(X, Y[1]), DTW(X, Y[1+N], DTW(X, Y[1+2N])$ where $Y[x]$ is the test utterance starting from the x^{th} frame. If $DTW(X, Y[x])$ is smaller than a certain threshold, the template exists around that location.

1. Do

   load('data.mat')

   You will notice several variables.

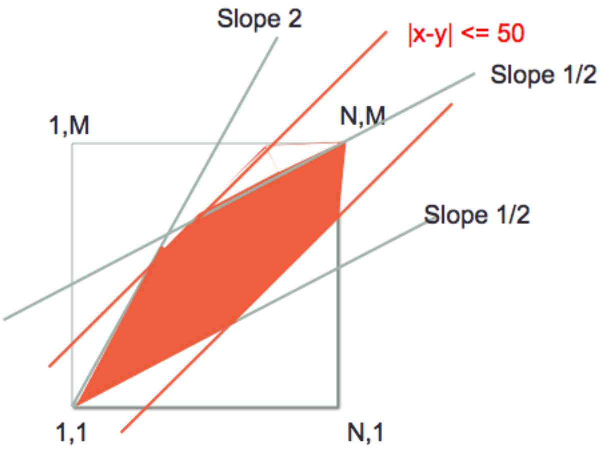
   - **templates** A cell containing the MFCCs of the template (Gowajee). A cell is a special array which can be access by using {} instead of (). See the template by typing *template{1,1}*. Notice how the data is 14x62. 14 is the feature dimension, and 16 is the number of frames in the template. The MFCC is normalized using MVN and whiten using PCA.
   - **mfccs_yes** A cell containing MFCCs from utterances that contains Gowajee. There are 35 utterances from 35 speakers. You can access the 20th utterance by typing *mfccs_yes{1,20}*
   - **mfccs_no** A cell containing MFCCs from utterances that does not contain Gowajee. These are collected from the same 35 speakers.
   - **templates_nocmn, mfccs_yes_nocmn, mfccs_no_nocmn** MFCCs without using CMN.

2. Complete *compute_dtw.sce*. This should compute a DTW with type II local constraint. $|x - y| \leq 50$ global constraint. However, we will not require the end point to be (N, M), where N is the size of the template, and M is the size of test. The end point can be anywhere along the line $x = N$, so that the test utterance can have extra frames that do not need to be matched. In other words, the DTW distance between template X and test utterance Y is

   $$DTW(X,Y) = min_{N-50 \leq m \leq N+50} d(N,m) \tag{1}$$

   where $d(x, y)$ is the shortest distance from point $(1, 1)$ to (x, y)

   See the picture below for the constraints.



3. Test your code by running

```
    dist = compute_dtw(templates{1},templates{1})
 dist  =

   0
    dist = compute_dtw(templates{1},mfccs_yes{29})
 dist  =

   1596.5334
```

4. Now we will evaluate the performance of the keyphrase detector. The keyphrase detector works by computing a DTW distance every 25 frames, and compare each distance against a threshold. If the distance is smaller than the threshold, we say there is Gowajee in the utterance.

   Run

   ```
   [yesd,nod] = computeall(templates{1,1},mfccs_yes,mfccs_no);
   %This step can take 10 minutes
   %yesd and nod contain dtw distances from utterances
   %  that have and do not have Gowajee, respectively.
   [FP,TP] = computeRoC(yesd,nod);
   plot(FP*100,TP*100);
   xlabel('False Alarm Rate (%)');
   ylabel('True Positive Rate (%)');
   ```

   The plot generated is called a "Receiver operating characteristic (RoC) curve." The RoC curve illustrates the performance of a binary classifier (Is there Gowajee? yes or no) as the threshold (the DTW distance) is varied.

   The x-axis is the False Alarm Rate or how often is a 'no' classified incorrectly as a 'yes.' The y-axis is the True Positive Rate or how often is a 'yes' classified correctly as a 'yes.'

   The plot usually starts at the origin (0,0), which is where the threshold is so low that every utterance is classified as 'no.' As we increase the threshold, more utterances are classified as 'yes,' but it also increases the false alarm rate. Finally at (1,1), the threshold is so high.

   What is the false alarm rate when the true positive rate is 50%?

5. Repeat the process for MFCC features without CMN. In other words, run

   ```
   [yesd_nocmn,nod_nocmn] =
   computeall(templates_nocmn{1,1},mfccs_yes_nocmn,mfccs_no_nocmn);
   ```

   Is the performance better or worse than with CMN?