CS571 Lab 9

October 26, 2021

In this lab, you will be working with audio signals. Many Python libraries exist for this; for example, `librosa` is a powerful library with lots of tools. Another example is `SoundFile` (mostly for reading and writing). Google for these to see their webpages. You will need to install them in your Python environment.

Spectral analysis

1. Generate a discrete-time sinusoid with frequency $\omega_0 = \pi/16$, sampled at 8000 Hz. Generate samples for 2 seconds of the signal.

   (a) Save and write the signal as a `wav` file.

   (b) Play the `wav` file and listen to it.

   (c) Plot the signal in time domain.

   (d) Compute its Fourier transform using DFT. Plot the magnitude spectrum. Ensure that the x-axis shows discrete frequency.

2. Generate the signal $x(n) = 4\cos(\frac{\pi}{16}n) + \cos(\frac{\pi}{32}n)$, sampled at 8000 Hz. Generate the signal for 5 seconds.

   (a) Save and write the signal as a `wav` file.

   (b) Play the `wav` file and listen to it.

   (c) Plot the signal in the time domain.

   (d) Compute its Fourier transform and plot the magnitude spectrum. Ensure that the x-axis shows discrete frequency.

Short-time processing

3. In this question, you will be using the audio file `should.wav`.

   (a) Load the audio file. Listen to it. Plot the signal in the time domain. What is the sampling rate?

(b) Write a function `enframe(x,winsize,hoplength,fs,wintype)` which accepts the signal `x`, the window size (in sec) `winsize`, the hop length (in sec) `hoplength`, and the sampling rate (in Hz) `fs`. `wintype` is a either ‘`rect`’ or ‘`hamm`’, which is either a rectangular or a Hamming window. The function should return a matrix containing the windowed frames of the signal. Typical values are 30ms for `winsize` and 15ms for `hoplength`.

(c) Compute the time-dependent Fourier transform of the frames returned by the above function. Plot the magnitude spectrum. Cycle through the log magnitude spectrum for each frame.

(d) Experiment the previous question using a Hamming window and using a rectangular window. Do you note any difference in the spectra?

Audio segmentation

4. Segment the file `should.wav` into three parts using energy-based segmentation. Use the `enframe` function you developed earlier. Based on the output, cut the signal into parts. Save each part as a `wav` file and play it.