Prelab 4

Prelab 4 - Autocorrelation

Summary

In this prelab, you get familiarized with two common tasks in speech signal analysis: voicing determination and autocorrelation.

Downloads

- test_vector.wav

Part 1 - Voiced/Unvoiced Detector

Voiced/unvoiced signal classification is an incredibly well-studied field with a number of vetted solutions such as Rabiner's pattern recognition approach or Bachu's zero-crossing rate approach. Pitch shifting (next lab) does not require highly-accurate voiced/unvoiced detection however, so we will use a much simpler technique.

The energy of a signal can be a useful surrogate for voiced/unvoiced classification. Put simply, if a signal has enough energy, we assume it is voiced and continue our pitch analysis. The energy of a discrete-time signal is given as follows:

$$E_s = \sum_{n=-\infty}^{\infty} |x(n)|^2 \tag{1}$$

> **Assignment 1**
> Using the given test speech signal and the test code given below, determine a useful threshold for E_s and classify frames as voiced (return 1) or unvoiced (return 0). The test code will plot the results for you.

COPY

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import read, write

FRAME_SIZE = 2048

def ece420ProcessFrame(frame):
    isVoiced = 0

    #### YOUR CODE HERE ####

    return isVoiced

################# GIVEN CODE BELOW #####################

Fs, data = read('test_vector.wav')

numFrames = int(len(data) / FRAME_SIZE)
framesVoiced = np.zeros(numFrames)

for i in range(numFrames):
    frame = data[i * FRAME_SIZE : (i + 1) * FRAME_SIZE]
    framesVoiced[i] = ece420ProcessFrame(frame.astype(float))

plt.figure()
plt.stem(framesVoiced)
plt.show()
```

Part 2 - Autocorrelation

Autocorrelation is the process of circularly convolving a signal with itself. That is, for a real signal, the discrete autocorrelation is given as:

$$R_{xx}[l] = x[n] \circledast \tilde{x}[-n], \tag{2}$$

where $\tilde{x}[-n]$ is the complex conjugate of the time reversal of $x[n]$. The output $R_{xx}[l]$ measures how self-similar a signal is if shifted by some lag l. If normalized to 1 at zero lag, this can be written equivalently as:

$$R_{xx}[l] = \frac{\sum_{n=0}^{N-1} x[n]x[n-l]}{\sum_{n=0}^{N-1} x[n]^2} \tag{3}$$

For a periodic signal, the lag l that maximizes $R_{xx}[l]$ indicates the frequency of the signal. In other words, the signal takes l samples before repeating itself. This algorithm, combined with some additional modifications to prevent harmonics from being detected, comprises the most well-known frequency estimator for speech and music.

> **Assignment 2**
> Calculate and plot the autocorrelation of the test signal `tune` using the test code below. You may not use `np.correlate()` or other such functions.

> **Question**
> Indicate the value of lag l that maximizes $R_{xx}[l]$. What is the signal frequency that corresponds to this lag?

Python test code

COPY

```
import numpy as np
import matplotlib.pyplot as plt

fs = 8000        # Sampling Rate is 8000
duration = 1     # 1 sec
t = np.linspace(0,duration,duration*fs)
freq = 10        # Tune Frequency is 10 Hz
tune = np.sin(2*np.pi*freq*t)

# Add some Gaussian noise
tune += np.random.normal(0, 0.5, duration * fs)

plt.figure()
plt.plot()
```

```
# Start a new figure for your autocorrelation plot
plt.figure()

# Your code here

# Only call plt.show() at the very end of the script
plt.show()
```

Grading

Prelab 4 will be graded as follows:

- Assignment 1 –> 1 point

  > *A plot of the voiced/unvoiced detector (1 point)*

- Assignment 2 –> 1 point

  > *A plot of the autocorrelation result (0.5 point)*
  > *Short answer question (0.5 point)*