Lab 2: Signals and Systems

Convolutions and Correlations

Introduction

The central theme of this second session are *correlators* and their applications to *time-delay analysis* and template matching. Let x and y be discrete signals, each consisting of N samples. The correlation of x and y is defined as (using C/Java style indexing, so the first index is 0!):

$$(x \star y)[d] \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} x[n]\, y[n+d] \quad (\text{for } 0 \leq d < N).$$

Note that in the above formula, the index $n + d$ may be "out of bounds". This is a standard problem in signal processing, which is often solved by considering signals *circular*. In other words, the above formula should actually be read as:

$$(x \star y)[d] \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} x[n]\, y[(n+d) \bmod N] \quad (\text{for } 0 \leq d < N).$$

In all exercises of this second lab session we adopt the convention that signals are circular, and leave out the **mod** operator in our formulas. However, if you implement a *correlator* (which is a routine that computes the correlation of two signals) then you have to make sure that you do not address arrays out of bounds. Moreover, you need to incorporate that indexing in Matlab/Octave starts from 1, which is a real nuisance for this particular operation.

1 Time delay and correlators

(a) Implement yourself the function `circorr(x,y)` which returns the circular correlation of the signals `x` and `y`.

(b) Create a discrete random signal `x=rand(1,100)` and plot it. Now plot the circular correlation of `x` and itself. What is your conclusion?

(c) Write a function `rotate(sig, n)` that rotates the signal `sig` by `n` samples. For example, the output of `rotate(1:10,3)` would be the signal `[8 9 10 1 2 3 4 5 6 7]`. Let `y` be `rotate(x,50)`. Plot the circular correlation of `x` and `y`. What is your conclusion?

(d) A `Pearson correlator` is a special correlator. It yields normalized values from the interval [-1,1]. The formula for the Pearson correlation of `x` and `y` is (again, we assume the signals to be circular):

$$(x \circ y)[d] \stackrel{\text{def}}{=} \frac{\sum_{n=0}^{N-1}(x[n]-\overline{x})\,(y[n+d]-\overline{y})}{\sqrt{\sum_{n=0}^{N-1}(x[n]-\overline{x})^2}\sqrt{\sum_{n=0}^{N-1}(y[n]-\overline{y})^2}} \quad (\text{for } 0 \leq d < N).$$

In this formula, the notation $\overline{x}$ denotes the mean value of the signal x. Implement yourself the function `circpearson(x,y)` which returns the circular pearson correlation of the signals `x` and `y`. Compare the results and the speed of your implementation with the function `pearson` that can be found on Nestor. The function `pearson` uses a computational trick that you will learn to understand later in the course lectures. Note: if the denominator of the Pearson correlator is zero, then we define $(x \circ y)[d] = 0$, to avoid division by zero.

(e) Explain why a Pearson correlator is in most applications more useful than a standard correlator. Give an example that supports your explanation.

(f) Create two signals using the function `sinusoid2samples` that you made in the first lab session: one is a sine, and the other is a cosine (choose for both signals the same frequency, typically 5 to 10 Hz). Sample the signals with a sampling rate of 100 samples per seconds. Plot the Pearson correlation of the two signals. What do you observe? Vary the amplitudes of the signals. Do you see a noticable effect? Explain.

(g) Repeat the parts (b) and (c) using the circular Pearson correlator. Also, perform robustness analysis by successively adding (more) noise (use the random generator) to `x` while `y` remains constant. How much noise can you add, while still detecting (reasonably) correct delays?

2 A more general Pearson correlator

The Pearson correlator from exercise 1 assumes that both signals have the same length (i.e. same number of samples). Now, write a Pearson correlator where the lengths of the signals `x` and `y` may differ. Let us assume that the length of `y` is only 10 samples, while `x` consists of 1000 samples. We still assume that `x` is circular. Now, the expression $\overline{\mathrm{x}}$ is no longer a constant, but a (circular) running average of only 10 samples (the length of `y`).

(a) Write a Matlab/Octave script that computes the Pearson correlation of two signals `x` and `y` that may differ in lengths. Matlab/Octave has a built-in function `conv(x,y)` that computes the convolution (not the correlation!) of two signals that may have different lengths. Use this function in your implementation.

(b) Now make the signal $x[n] = \delta[n-50] + \delta[n-100])$ (use 150 samples), and the masks $h_1[n] = \delta[n-10] + \delta[n-60]$ (use 70 samples) and $h_2[n] = \delta[n-10] + \delta[n-59]$ (use again 70 samples). Use the Pearson correlator to compute $x \circ h_1$ and $x \circ h_2$. What is your conclusion?

(c) The mask h_2 of the previous exercise fits almost perfectly the signal x, but it is off by only one sample. write a function `patternMatch(x,h)` that tries to match the pattern/mask `h` with the signal `x` using an adaptation of the direct Pearson correlator which is more robust. The output of this function must be a vector of (modified) correlation values.

(d) On Nestor, you can find the files `handel.wav` and `hallelujah.wav`. The first file is a short selection from the *Messiah* by Händel. The latter file is a crop from the first file, containing the choir singing "Hallelujah". Both files are mono (i.e. 1 channel), sampled at 8kHz. Try to detect occurrences of "Hallelujah" in the file `handel.wav`. Why does it (sometimes/not) work?

(e) On Nestor, you can also find the files `furelise8kHZ.wav` and `cropelise8kHz.wav`. The first file is a selection from *für Elise* (Beethoven), and the second file is a crop from it. Answer the same questions as in part (d) for these files.

3 Correlation in 2D (Images)

Download the files `page.pgm` and `maskM.pgm` from Nestor. The first file contains a scanned page of an old book (by Ubbo Emmius) and the second is a very small image (a mask) containing only the letter `m` (it was cropped from the page). Use (Pearson) correlation to count the number of occurrences of the letter `m` in the page. Besides correlation, you are allowed to use any FIR-filter that suits you. Investigate the stability of your approach by adding successively noise to the page. Make a plot of recognition rate versus noise level.