Lab 2

Some tips on normalized cross correlation

Efficient computation of normalized cross correlation

Given two signals x and y, the cross-correlation between the two can be computed through:

$$(x \star y)[d] = \sum_{n=1}^{N} x[n]y[n+d] \tag{1}$$

As discussed during the lectures, the cross-correlation of two signals can also be computed by using:

$$x \star y = \mathrm{conv}(x, \mathring{y}) \tag{2}$$

where $\mathring{y}$ is flipped (horizontally in 1D, horizontally and vertically in 2D). In MATLAB you can compute this efficiently with `conv`.

In the case of lab 2, however, we want to compute the *normalized* cross-correlation (also known as Pearson correlation). First, the mean is subtracted from the signals to make the mean of the resulting signals 0. Then, we divide by the product of the norms of both signals.

$$(x \circ y)[d] = \frac{\sum_{n=1}^{N}(x[n]-\overline{x})(y[n+d]-\overline{y})}{\sqrt{\sum_{n=1}^{N}(x[n]-\overline{x})^2}\sqrt{\sum_{n=1}^{N}(y[n]-\overline{y})^2}} \tag{3}$$

At question 2 of lab 2 you are asked to program a normalized cross correlation for signals x and y that differ in length. Suppose that x has length 100 and y has length 10. What you should do is compute the correlation for each slice of 10 elements of x with y.

Note that the numerator in (3) cannot be computed directly with $\mathrm{conv}(x-\overline{x}, \mathring{y}-\overline{y})$ since $\overline{x}$ will be different for each slice of x.

If y would be the shortest signal, the Pearson correlation can be computed through:

$$(x \circ y)[d] = \frac{\sum_{n=1}^{N}(x_d[n]-\overline{x_d})(y[n]-\overline{y})}{\sqrt{\sum_{n=1}^{N}(x_d[n]-\overline{x_d})^2}\sqrt{\sum_{n=1}^{N}(y[n]-\overline{y})^2}} \tag{4}$$

Where x_d is the slice that corresponds with alligning y with x at index d. Now N is the length of the shortest signal.

While you are figuring out your solution to the problem, keep in mind that the signals are circular (i.e. the last part of x should be the same as the first part of x). You can accomplish this easily by *prepending* the last $N-1$ elements at the front of the signal. In the case of the example here it implies that the length of x is now 109. By default, `conv` will return a vector of `length(x) + length(y) - 1`, which would be 118. A simple workaround is to use `'valid'` as a parameter for `conv`. This parameter ensures that `conv` only returns the convolution of valid alignments of x and y (so no zero-padding).

The numerator If you still want to use MATLAB's built-in `conv` function you will need to rewrite the numerator in the expression above by eliminating the brackets. Simplify the expression as much as possible. By eliminating the brackets you can use `conv` to compute the non-normalized cross correlation of two signals which you then correct to compute the normalized cross correlation.

Computing $\overline{x_d}$ Think of an efficient way to compute $\overline{x_d}$ for each slice of x. Hint: one possible solution makes use of MATLAB's function `cumsum` that computes the cumulative sum of a vector. Another valid approach is to use a running averager, in which case you can use `conv` as well. Your method of computing the means should not exceed complexity $\mathcal{O}(N \log(N))$.