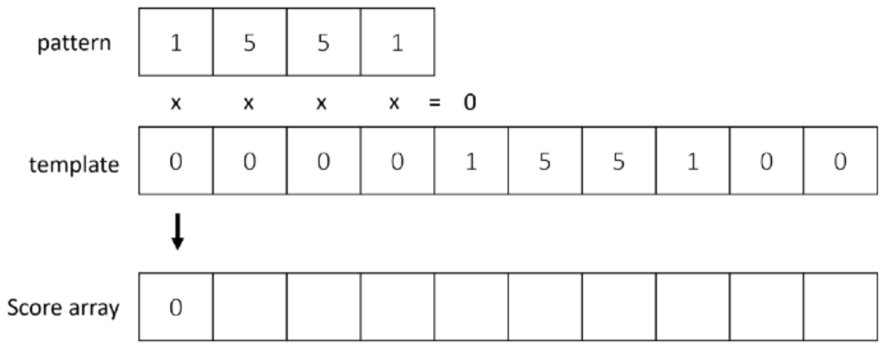
1D Spatial Cross-correlation

Basic concept

1D cross correlation involves vector (elementwise) multiplication of a pattern and template. To find the location of best match for the pattern, each location is scored. The base idea (a simplistic approach) can be seen below:



Let's call the pattern array ***f***, and the template array ***g***. The pattern has length ***n***, and we will denote the offset position in the template we are currently scoring as ***m***. In the diagram above, ***n*** = 4 and ***m*** = 0. Currently, the cross-correlation score for the pattern and template at a particular offset position ***m*** (commonly referred to as the *lag*) is:

$$corr_{f,g}[m] = \sum_{i=1}^{n} f_i \cdot g_{i+m}$$

Question 1.

a). Write a function which accepts a pattern and template, and returns a cross-correlation score array using the equation above. A score should be calculated for each location where the pattern and template fully overlap.

<u>Test inputs</u>

Pattern: [1, 2, 2, 1]

Template: [0, 1, 1, 0, 1, 2, 2, 1, 0, 0, 0]

Expected return value: [4, 4, 5, 8, 10, 8, 4, 1]

b). Write a function which accepts a score array, and returns the array location (the index) and value of the largest element in the array. This is the best *lag* i.e. the location in the template where the pattern displayed best match.

<u>Test inputs</u>

score array: [4, 4, 5, 8, 10, 8, 4, 1]

Expected return value: 4, 10

Normalisation

For some inputs, this scoring method would not always return the best match location. As pointed out today in class, irregular high values in our array can result in poor localisation for the best match location (*lag*) in the template.

Today in class, mean shifting was discussed as a possible method to normalise arrays. While this does the job in some situations, it inherently biases values in arrays which were initially low or high. Values in the pattern and template which are near the mean take values near zero, and so do not contribute to the score for that *lag,* while values close the extremes have greater impact on the score. For this reason, a different normalisation method has been chosen for this class.

Our chosen normalisation method relates to the energy of the pattern and template. The approach is to calculate the array 'energy' for the pattern, and the slice of the template we are scoring, and normalising by this value. Consider the below:

pattern	0	1	1	0						
template	0	1	1	1	1	0	0	1	1	0

Lag locations 0, 1, 2, and 6 in the template would all produce the same un-normalized score. This said, if we were to calculate the array energy for the slice of template we were scoring then normalise by this value, *lag* 6 would become the clear best match.

Consider *lag* 0. Our template slice is [0, 1, 1, 1], with a total sum of 3. If we were to use this sum as our measure of array energy, we could normalise our score to be (0 + 1 + 1 + 0) / 3 = 0.666. If *lag* = 6 and we are therefore scoring the actual best match location, our template slice is [0, 1, 1, 0] with sum 2. The score for this *lag* would be (0 + 1 + 1 + 0) / 2 = 1. We have now found a method which returns the highest score for the true best *lag* by using array energy.

A more robust measure of array energy is the following:

$$\sigma_f \sigma_g = \sqrt{\sum_{i=1}^{n} f_i^2 \cdot \sum_{i=1}^{n} g_i^2}$$

Be careful here – *g* refers to the **slice** of the template we are looking at. If we are at *lag* 4 and the pattern array length (*n*) is 4, our template slice is from elements 4-7 inclusive (we can write this array slice in python as arr[4:8]).

The energy equation above can be expressed as squaring each array value and summing the result for both the pattern and the template slice, then multiplying these individual energy measures and taking the square root of this product.

Putting it all together, the normalised cross-correlation score for *lag* m is:

$$corr_{f,g}[m] = \frac{\sum_{i=1}^{n} f_i \cdot g_{i+m}}{\sigma_f \sigma_g} \ where \ \sigma_f \sigma_g = \sqrt{\sum_{i=1}^{n} f_i^2 \cdot \sum_{i=1}^{n} g_i^2}$$

$$when\ calculating\ array\ energy, g\ refers\ to\ the\ template\ slice\ not\ the\ whole\ template$$

Question 2.

a) Write a function which calculates array energy normalisation factor of two arrays. The input to this function is two arrays, where one is the pattern, and one is a slice of the template. Make sure you have a statement which checks the input arrays are the same length!

Test inputs

Array 1: [0, 1, 1, 2]

Array 2: [1, 1, 0, 2]

Expected return value: 6

Edge matches

It is possible the best *lag* involves only part of the pattern and template being matched. These situations become more apparent in 2D scenarios. Zero padding or allowing overhangs can be used in these scenarios.

Zero padding has the easier implementation because the cross-correlation equation will not change. If we attempt overhangs, you must take a slice of the pattern being used for array energy calculation and array multiplication, which can sometimes use confusing loops to calculate where to start and end the pattern and template slice. It results in lower space usage, so you are more than welcome to use this method if you wish. Both are acceptable.

Question 3.

a) Write a function which zero-pads a template array. The inputs to the function are the size of the padding (equal to the pattern length - 1), and the array to be padded.

<u>Test inputs</u>

Padding size: 3

Array to pad: [1, 1, 0, 2]

Expected return value: [0, 0, 0, 1, 1, 0, 2, 0, 0, 0]

Submission question

At this point, the last task to do for 1D spatial cross-correlation is the speaker and microphone distance question. The signal files are available in the ‘files’ tab of our team (files -> data -> 1_pattern_finding -> sensor1Data/sensor2Data.

2 Signal Offset

Code for submission: *script to find offset*
Non-code submission: *offset time, run time, sensor distance*

You will be provided with two signal files, you know these signals have come from the same source and are just offset by some time. Using cross correlation find the offset time. Knowing that this signal propagates at 333m/s, what is the distance between the two sensors, *x* (refer Figure 1)?

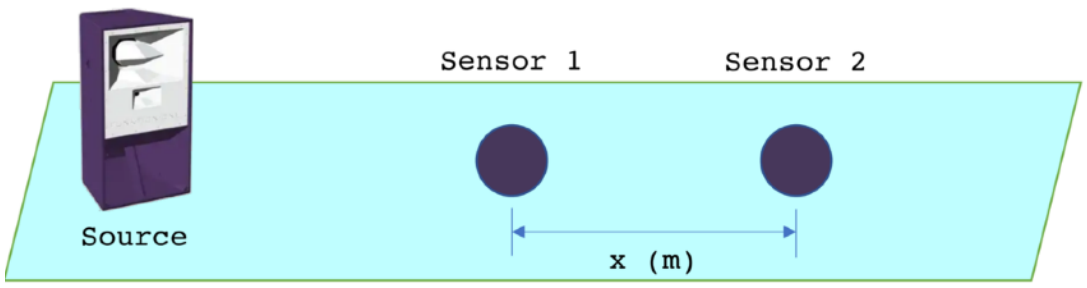


Figure 1 Source and sensor arrangement