Signal Processing Algorithms & Architecture

Dr. Anirban Dasgupta
Department of Electronics & Electrical Engineering
Indian Institute of Technology Guwahati

Lec 2: Correlation and Convolution

Hello everyone, we are in module 1, and this is the second lecture where we will continue with the topic of correlation and convolution. So, let us get started. I am Dr. Anirban Dasgupta, Assistant Professor in the EEE Department at IIT Guwahati. And we will start this lecture with the concept of time domain signal processing. So this concept was introduced in the first lecture, and here in this module, we will cover the different time domain techniques or algorithms.

So, first, we will cover. Time shifting and scaling; then we will cover amplitude shifting and scaling. We will cover modulation, we will cover correlation, and the next convolution. Then we will go for zero crossing detection, peak detection, and envelope detection.

Smoothing and pattern matching, so we will start with time shifting and scaling. So, here you can see that there is a diagram of a signal, which is a section of a sine wave, and we have done a time shifting and a time scaling. So what is time shifting? Time shifting refers to shifting the signal. in time either to the left, which is advancing, or to the right, which is delaying. So suppose we have the signal $x(n)$, the shifted version is represented.

By $x[n - n_0]$, where n_0 is an integer. So, if n_0 is negative, this means that we are advancing the signal and if it is positive, then we are delaying the signal. Time scaling is something different for discrete time signals as compared to an analog or continuous-time signal. So, in an analog continuous-time signal, the signal is either Compressed or elongated, but in a discrete time signal, of course, this happens.

There is one more thing that is happening, which is technically a change in the sampling rate. Because, for example, if I have a signal $x[n]$, then $x[2n]$ is basically me downsampling the signal because I am picking every even point of the signal. So, in general, I will say that this is modeled by

$$x[a\,n]$$

where $a \neq 1$. And if $a < 1$, Then this is called upsampling.

And if $a > 1$, then this is called downsampling. Now, since we are talking about time domain algorithms, this course is a course on signal processing algorithms. So, I would like to talk about time complexity, and this time complexity is of order $O(n)$ for a signal of length. Because you are operating or running the loop for every point of the signal. The next algorithm is amplitude shifting and scaling.

So these are very simple algorithms; I am going a bit fast. So, amplitude shifting is now shifting the signal vertically. So, in time, we were shifting it horizontally. Here, this is given mathematically by-

$$y[n] = x[n] + c$$

Now, here we can see that this is my original signal and by shifting the amplitude by a factor, say 0.5, it is shifted above the line along the y-axis. Now, amplitude scaling is similarly shrinking or expanding the signal; it is kind of having a signal gain, like when my audio is of low volume, means I want to increase the volume, so I can perform amplitude scaling. And this is just multiplying by a scalar, and this algorithm will also have a time complexity of $O(n)$. The third algorithm, or the third kind of problem, is signal modulation, and to put it simply, it is just multiplying two signals, and since we are multiplying them pointwise, like

$$x_1[n] \cdot x_2[n]$$

means, say, $y[0]$ will be $x_1[0] \cdot x_2[0]$. And $y[1]$ will be $x_1[1] \cdot x_2[1]$. So the signal lengths typically must be the same. If they are not the same, then some kind of zero padding is done. Zero padding means adding some extra zeros so that the dot product of this vector can be taken.

And this modulation process is very common in communication systems which It is used to transmit signals over long distances. So, here if you see, this is a baseband. Signal, or this is called a message signal, and this is typically the carrier signal. So, these two signals are modulated, which means they are multiplied pointwise. This is also called amplitude.

Modulation will also have an order of n complexity for two signals having length n. Now, coming to correlation, which is a very important concept in signal processing. What is correlation? In English, we say that when we have some synonyms, we try. To

correlate, correlate situations, something like that. So in the context of signals, it is a measure of similarity.

So if there are two signals, correlation is a measure of how similar the signals are. And this is very useful in radar and sonar systems to find the location of the target. So what is done is there is a transmitted signal and there is a reflected signal. The lag is actually telling about the distance or where the target is. It is also used in operations such as template matching in image processing.

So correlation is typically of two types. One is cross-correlation, where we are finding similarities between two different signals and autocorrelation, which means within the same signal, like the delayed version of a signal, how much they are correlated. Like, say I am speaking, and in another audio, I am also speaking, but in this one, I am saying. The same word "hello," but there is a delay in recording. So, by using autocorrelation, I can match how much there is a lag.

So, what is the importance of correlation? So, first is signal comparison. Like I said, I want to compare two signals. Also estimating the time delay between two signals or between one's own signal and its delayed version. Correlation is also useful in noise reduction and system analysis. In signal comparison, pattern recognition, and template matching, it is very useful.

In time delay estimation, as I mentioned in radar and sonar, this is used for target localization by observing the delay of the transmitted and received signals. Noise reduction is supposed to be applied when there is a weak signal and the noise is very high. So, if it is correlating with a specific pattern or an already useful signal that is present. With me, then I can extract the weak signal from a noisy environment, considering. that the noise is uncorrelated with the weak signal.

Then in system analysis it is also very useful, as we can find the cross-correlation between the input and output of systems. So let us define these terms. The cross-correlation of two signals, say f(n) and g(n), is given by

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[m+n]$$

So this is cross-correlation, and in the same way, autocorrelation of a signal f of n is given.

$$(f \star f)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot f[m+n]$$

By the sum of f[m], f[m+n], and this symbol, the star symbol is used to define the correlation operation. And again, since this is an algorithm, if we have two signals of lengths n and m, we are finding the correlation. The naive method gives you a time complexity of $O(n \cdot m)$. Now this can be improved by other methods. Like the fast fourier transform, which we will discuss later, correlation, whether it is a linear operation or not.

It is a question. So for correlation to be linear, two properties must be satisfied. First is additivity, and the second is homogeneity. So, additivity means that if I add two Signals f and g are then correlated with h. This should be equal to if I individually correlate. these two signals and then add them.

Homogeneity means that if I multiply one signal with a scalar A and then correlating with G, it should be equal to first correlating these. two signals and then multiplying by the scalar. So if these two satisfy, then we can Say that correlation is a linear operation, and you can easily verify that these two. Operations are easily satisfied, and hence correlation is a linear operation.

- **Additivity:**
  $$(f + g) \star h = f \star h + g \star h$$
- **Homogeneity:**
  $$(af) \star g = a(f \star g),$$

where a is a scalar.

The next concept is the sliding window concept, which is typically used in operations like correlation and convolution, where there is one signal, are slided over the other signal.

So what is the length of output sequence? So in modulation, amplitude scaling, or time shifting, we saw that the input length and the output length are the same. Even for modulation, if two signals are of length n, then the output is also of length n. But this is not the case in correlation. So suppose my input signal $x[n]$ has a length N and y[n] has a length M, then the output signal length, say, that is defined by the symbol L; this is given by

$$N + M - 1.$$

And why is this so? So the maximum overlap will occur when y is fully shifted over n; these are the two signals.

So this is $x[n]$ of length N and $y[n]$ of length M. So let us observe this point. Now this will shift; say I am doing the sliding window operation, so n is shifted over m. And this is the maximum case in which I will get a non-zero output. So, if I shift this up to this, I will get M values and then plus N minus 1 values.

So, the total output of non-zero points will be $m + n - 1$. Let us take an example of autocorrelation, and I will take a very small signal, which is -1, 2, and 1. So, if I go by that formula, first of all, what should the output sequence length be? So, as I said, it is $n + m - 1$, and here n and m are both 3 because it is an autocorrelation. So it will be

$3 + 3 - 1$, which is 5. And if we do the math, these are the five points.

The five points will be n equal to negative 2, negative 1, 0, 1, and 2. And why is that? Because if you try to plug in those values, say n equals negative 3. So negative 3, if I put, say n equals negative 3, you will never come across a point where you get f of 0. So if for f of 0, where m equals 0, this is f of minus 3, and f of minus 3 does not exist. So, at most at $f(0)$ we will get $f(-2)$, and hence n will start from -2 and go up to 2.

And if we use this formula, we get that this is the sequence: negative 1, 0, 6, 0, and negative 1. my 0 index which is 6. Now 0 index will occur when both the signals are overlapping with each other. We also observe one more thing: the output sequence is symmetric, and this is true for any autocorrelation operation. And this is a pictorial representation of what is happening.

So this is n equal to 0 where the two signals typically It is the same signal because its autocorrelation is overlapping each other. And then these are the shifted versions of the signals. So this is n equal to 1, This is n equals 2, this is n equals negative 1, and this is n equals negative 2. So, this symmetric property is for real-valued signals where my autocorrelation sequence will be symmetric. And this is because if I shift the signal by k or negative k, it will yield the similarity measure, as multiplication is commutative.

And for complex-valued signals, this autocorrelation sequence will be Hermitian symmetric, which means it is complex. conjugate. So let us take an example of cross-correlation. So what happens in cross-correlation? Let us say, for the sake of simplicity, that I have taken the signals to be of the same length, which is 4. But they can be of any different lengths as well.

And first, let us find the output sequence length. So it is $4 + 4 - 1$, which is now 7. If I plug in the values of n, I see that when n is negative 3, I will get the value of f at 0. So, this is the formula where this is $f(3)\ g(0)$. At n equals negative 2, we get a value of

$$f(2) \cdot g(1) + f(1) \cdot g(0).$$

And if we carefully observe, there will be There are 7 such values, but let us solve this problem graphically because it is more intuitive. So here we have all 7 values, and this is point 0 where the signals overlap each other. Other, the 0 indexes match, and then these are the positive shifts. These are the negative shifts. So, in this manner, we get this output sequence with the 0 being at the value of 8.

So, one challenge. In this correlation, or even convolution, handling the boundary conditions is essential, and because of this, the Signal boundaries get distorted. So the most common solution to this is zero padding, which means that technically, what you are doing is that I just multiplied 6 and 2. But essentially You have to do a vector operation. Why a vector operation? That we will understand when we study. The architectures are like a dot product of two vectors.

So if I say dot product, these Two vectors should be of the same length. So how can we make that? By zero padding. So now, by zero. Padding these two vectors to become equal in length allows us to easily take the dot product, which is basically the So how many zeros do we need to pad? This is the length.

Of the kernel and the number of overlapping points. Like in this case, I will see that the kernel length is 5 and there is only one overlap. So, I have to pad four zeros. In this case, my kernel length is 4 with one overlapping point. So, I have to pad three zeros. This method can often still result in some artifacts at the boundary.

So, a better method is to use edge replication instead of zero padding. So, here, as you can see, the edges are replicated, and the edge values are repeated. Now, this is somewhat better compared to zero padding, but this can also sometimes. Add some artificial discontinuity at the borders.

So, these are now typically solved by some. Post-processing methods include some filtering or other techniques, which we can discuss in later lectures. What is the other

method of computing correlation? The other method of computing. Correlation can be interpreted as the inner product of two signals. And what this inner product measures is that it is a projection of one signal onto the other. Let us take the same two signals, and now we will find the correlation using matrix multiplication.

So the first signal we write down as it is and then pad these many zeros. Why? Because now we know that the signal output length would be 7, and in the second Column, we just rotate it 1 down and put a 0 here, and similarly, we do it for 4 values. and these are the 4 values of the other signal. Now if we multiply we get this signal as the output. And if you verify it, this is the same as the output we got by the formula as well as using the Graphical method, so all three methods are leading to the same result.

And this is one more takeaway since this can be computed as a matrix multiplication. You can say that this operation is a linear operation. Another interesting property is that this. Cross-correlation is not commutative in nature. Why is this so? Because, say I want to find The cross-correlation of two signals f and g is considered for a specific value of m, say at m equals 0.

I get the value as $f(0) \cdot g(0) + f(1) \cdot g(1) + f(2) \cdot g(2)$ Now this may be the same for the first case but For m equals 1, we see that for the first case, f is correlated with g: it is f(0) g(1), f(1) g(2), and f(2) g(3). While g correlated with f gives me g(0) f(1), g(1) f(2), and g(2) f(3). And now these two will not be the same. It may not be the same because these values will be different.

So f(0) g(1) is not equal to g(0) f(1), and the sum might not be the same. Even if it is the same that may be a pure coincidence for some specific value of m. So, in general, the cross-Correlation operation is not commutative. Now, coming to convolution. So convolution is very similar to correlation with a small change that here. I have n minus k, and this symbol indicates linear convolution, and this n is indicating the shift of the output signal relative to the input.

So, what about the complexity since correlation and convolution are of similar nature, having a similar number of operations. So, you can say that the naive way of computing convolution is O(n · m). This is one example of 1D convolution. So first, let's say we take one signal, say f of n and g of n; we flip g of n and we get g of minus n. Now we will shift the flipped signal g of n across f of n and shift, multiply, and add those.

Shift, multiply, add; shift, multiply, add, and each stage will give me the output values. So let us take a real example with the numeric values: I have

$$f = [1, 2, 3], \text{and}$$

$$g = [0, 1, 0.5]$$

So the output length is typically, so I can say technically, although it is given 0, 1, and 0.5; everything beyond is 0. So this length is basically 2. So it will be 3; it will be of length 2. So 3 plus 2 minus 1 is 4. And now, if we use the formula and find out, we will get all the values. So there are some interesting properties of convolution; unlike correlation, convolution is commutative in nature. So f convolved with g is equal to g convolved with f, and the order does not really affect the result.

The second is associativity, which means that if I convolve the result of g and h with f, this is the same as convolving f and g first and then convolving the result with h of n, and the third property, which is a property of linearity is the distributivity, where if I convolve f of n with g of n plus h of n, I will get this as a sum of individual convolutions. And this is what I was talking about: that convolution is a linear operation very much like correlation. And you can verify that the additivity and the homogeneity properties are satisfied.

So thank you so much. Wishing you a great day ahead. So we will come back for more lectures.