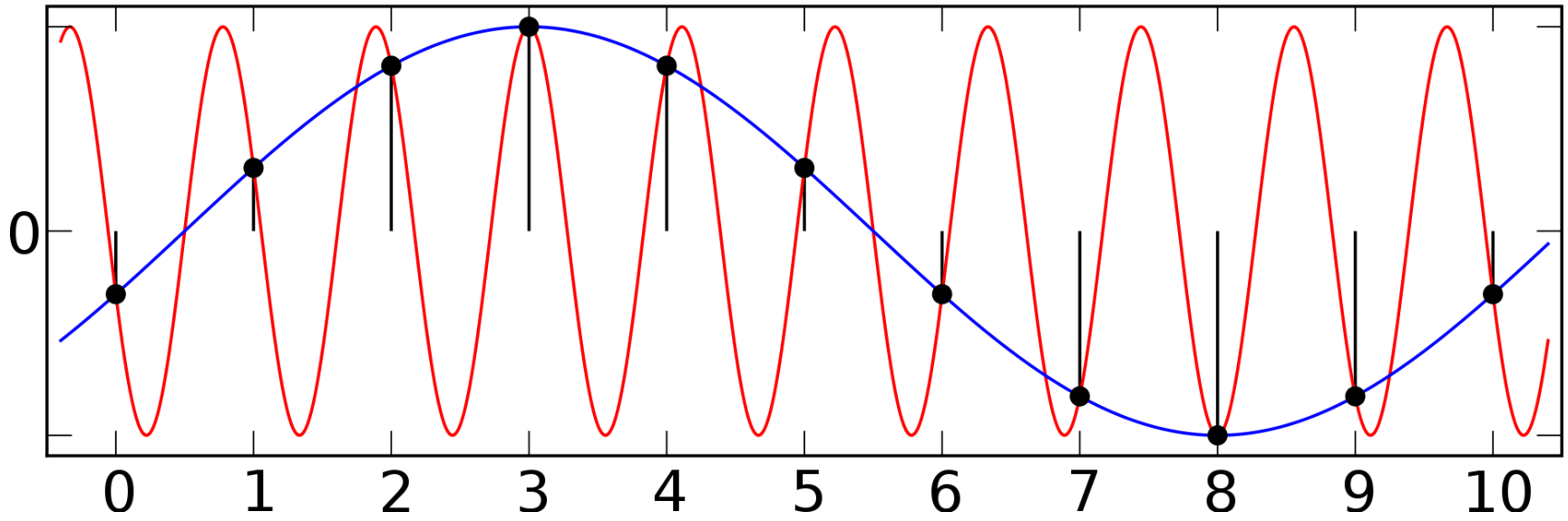
Fourier Transforms – Application: Aliasing & Correlation

CS370 Lecture 25 – March 13, 2017

Aliasing: The Math

Last time we discussed conceptually what aliasing is, and some of the effects that it can cause. Today, we'll describe it mathematically.



Aliasing - Takeaway

Our DFT coefficients F_k are sums of true continuous Fourier series coefficients c_k of increasing frequency:

$$F_k = c_k + c_{k+N} + c_{k-N} + c_{k+2N} + c_{k-2N} + \ \ldots$$

Hence, high frequencies from $c_k \notin \left[\frac{N}{2} - 1, \frac{N}{2}\right]$ in the real signal contribute to ("alias as") low frequency F_k for $k \in \left[\frac{N}{2} - 1, \frac{N}{2}\right]$.

Nyquist frequency

$\frac{N}{2T}$ is called the *Nyquist frequency.*

If the original continuous signal has frequencies such that

$$|k| > \frac{N}{2}$$

those frequencies “alias” (i.e., get added) to a lower frequency.

Once a signal is discretely sampled, there is **no way to distinguish** aliased frequencies.

Result: We have to sample at **twice** the rate of the highest occurring frequency in the original signal to avoid aliasing.

Treating Aliasing

Two (partial) solutions:

- Increase the sampling resolution to *capture* higher frequencies.
- Filter before sampling (digitizing) to *remove* (too) high frequencies, so they don't cause aliasing.

Example:

Digital cameras often have "optical lowpass filters" or "anti-aliasing filters" that physically blur the incoming light, *before* it hits the image sensor.

Correlation via FFT

Correlation

Essentially, finds similarities/relationships between different signals/data.

There are applications in pattern matching, synchronizing communications, signal detection, statistics, etc.

The Correlation Function

Consider two real, periodic data sets y_i, z_i for $i \in [0, N-1]$.

The correlation function ϕ_n is

$$\phi_n = \frac{1}{N} \sum_{i=0}^{N-1} y_{i+n} z_i$$

where n indicates the offset. Want to find n that maximizes ϕ_n.

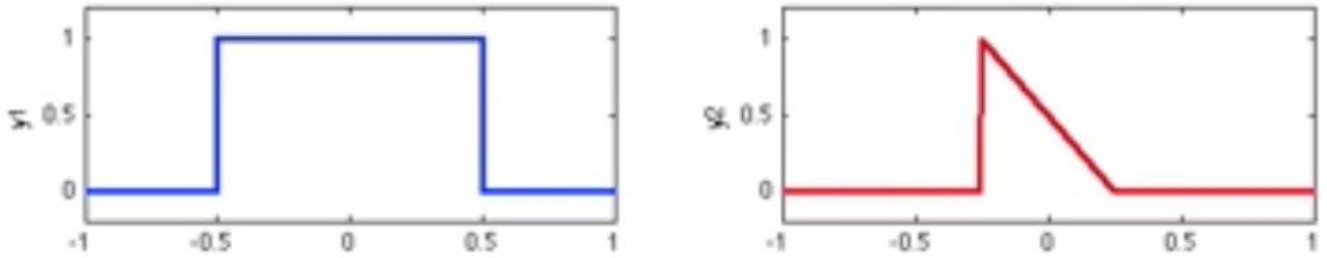
If max of ϕ_n occurs for $n = p$, then shifting by p slots gives the "best match" (i.e., maximum correlation) between the data.

Essentially: shift one vector by n, take the dot product, divide by N.

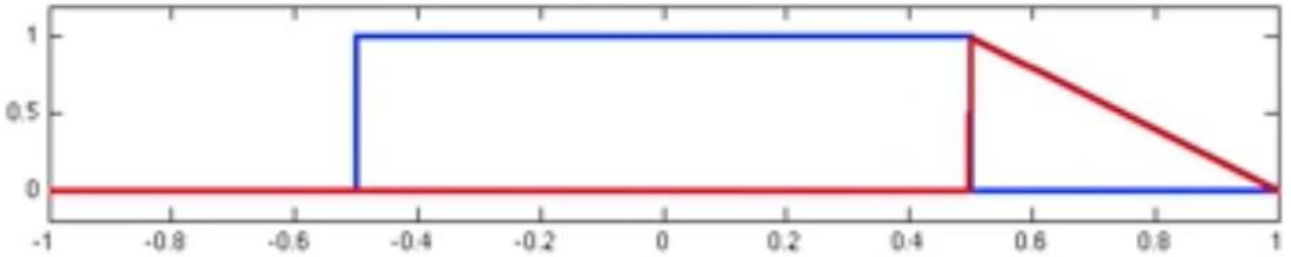
Note that the data wraps (i.e. it's periodic).

Correlation - Visual Illustration

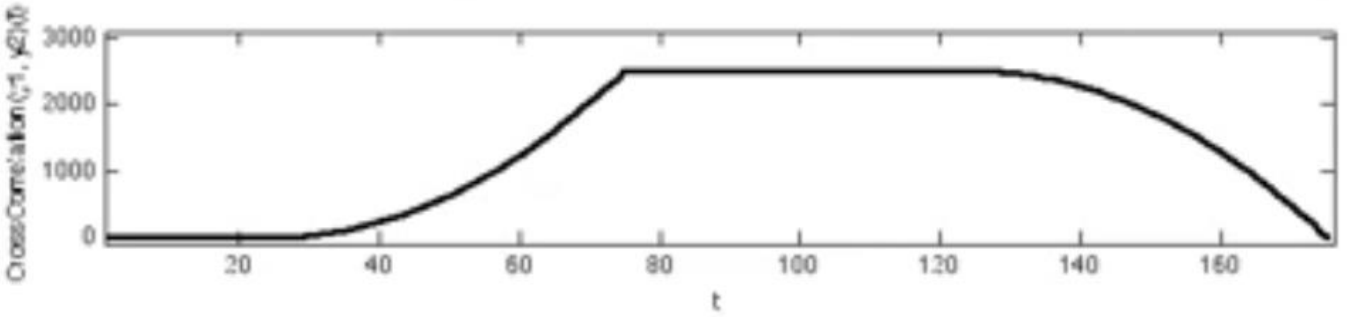
Two signals to compare:



Shift one signal with respect to the other:



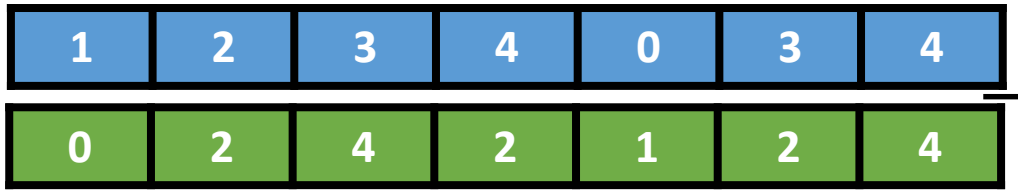
Evaluate and plot (scaled) dot product:



Correlation: Detailed Example

$$\phi_n = \frac{1}{N}\sum_{i=0}^{N-1} y_{i+n} z_i$$

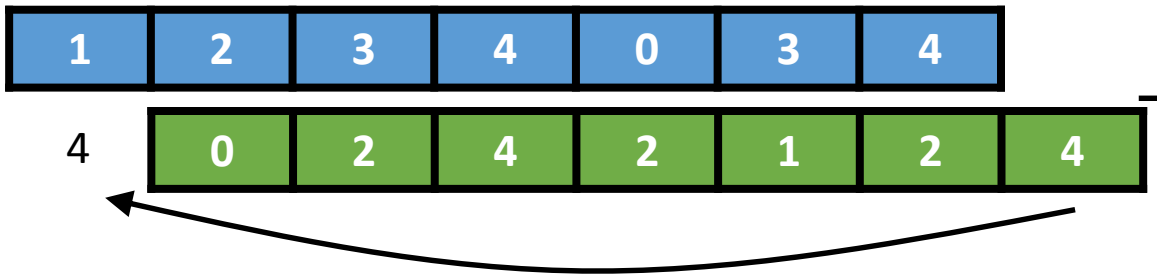
$n = 0$:



$$0 + 4 + 12 + 8 + 0 + 6 + 16 = 46$$

$$\phi_0 = \frac{46}{7} \approx 6.6$$

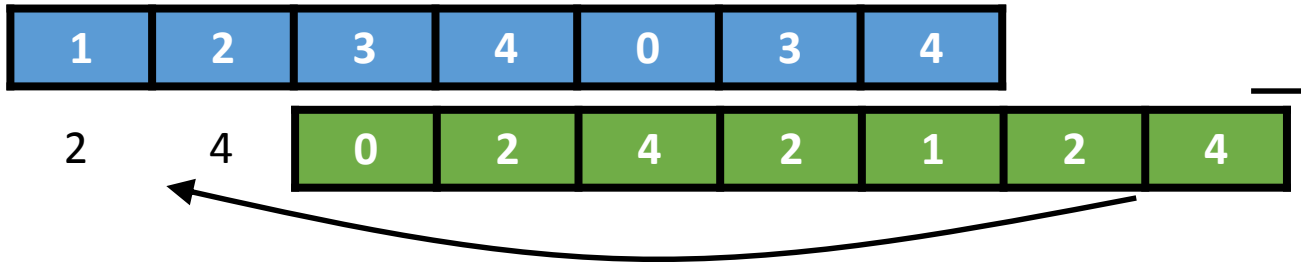
$n = 1$:



$$4 + 0 + 6 + 16 + 0 + 3 + 8 = 37$$

$$\phi_1 = \frac{37}{7} \approx 5.3$$

$n = 2$:



$$2 + 8 + 0 + 8 + 0 + 6 + 4 = 28$$

$$\phi_1 = \frac{28}{7} = 4$$

Etc.

Correlation Function – Naïve evaluation

Our definition is

$$\phi_n = \frac{1}{N}\sum_{l=0}^{N-1} y_{l+n} z_l \,.$$

To find the max, there are N possible offsets (values of n), each requiring a loop over **all N** entries of y & z.

Naïve algorithm:

Two nested ***for*** loops of length N, so direct evaluation is $O(N^2)$.

Faster Correlation

Exploiting the FFT, we can do the same thing in $O(Nlog_2N)$.

In time-domain, the correlation is:

$$\phi_n = \frac{1}{N}\sum_{l=0}^{N-1} y_{l+n} z_l \,.$$

In frequency-domain, it's simpler – no "for loop" required:

$$\Phi_n = Y_k \overline{Z_k}$$

Let's see how this can be true…

Correlation via FFT

Overall algorithm:

Given vectors y and z…

Apply forward FFT: $Y = FFT(y)$ and $Z = FFT(z)$. $\rightarrow O(Nlog_2N)$

Compute products $\Phi_k = Y_k\bar{Z}_k$ for $k = 0 \ldots N-1$. $\rightarrow O(N)$

Apply inverse FFT to get $\phi = IFFT(\Phi)$. $\rightarrow O(Nlog_2N)$

Total complexity: $O(Nlog_2N)$.