1.1 Pearson Correlator 1D Representation

OMG a 2D correlator equation is huge

To translate the formula into code we first need to understand it.

Input: template $h[n] \sim 3: [2, 3, 5]$
discrete input $x[n] \sim 10: [1, 2, 3, 5, 4, 420, 1, 12, 13, 15]$

Output: sequence of pearson correlations $y[n] \sim 8: [\ldots\ldots]$

Formula:

$$y[n] = \frac{\sum_{k=0}^{L-1} (x[n+k] - \bar{x})(h[k] - \bar{h})}{\sqrt{\sum_{k=0}^{L-1} (x[n+k] - \bar{x})^2} \cdot \sqrt{\sum_{k=0}^{L-1} h[k] - \bar{h})^2}}$$

Simplified:

$$y[n] = \frac{L(\sum_k x[n+k] h[k]) - (\sum_k x[n+k])(\sum_k h[k])}{\sqrt{L \sum_k x[n+k]^2 - (\sum_k x[n+k])^2} \cdot \sqrt{L \sum_k h[k]^2 - (\sum_k h[k])^2}}$$

2: [1, 42]
10: [1, 42, 1, 42, 1, 42, 10, 52, 10, 52]

sum = 42·1 + 42·42
sum1 = 42 + 42
sum2 = 1 + 42

this is a lil note about some code for my assignment.

1.2 Discrete Fourier Transform w/ Vandermonde matrix

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

DFT converts $x[n]$ from time to frequency domain

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{2\pi}{N} kn}$$

k: Frequency index
N: samples in $x[n]$ :: len(x)
$X[k]$: complex rep of Amp and Phase

$$w = e^{-j2\pi/N} = \underbrace{\cos\left(\frac{-2\pi}{N}\right)}_{a} + j\underbrace{\sin\left(\frac{-2\pi}{N}\right)}_{b}$$

(a, b) i mean this a lil nugget that you can use to make something similar to cmath
PS. I use this for 2.3

1.3 Inverse Discrete Fourier Transform w/ Vandermonde matrix

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} kn}$$

I am now unsure if this is actually difficult or simply just a diffirent Omega?

So, my guess was correct, but its not just a diffirent Omega. The second diffirence is that each $x[n]$ element is divided by N

$$w = e^{j\frac{2\pi}{N} \cdot kn} = \cos\left(\frac{2\pi}{N}\right) + j\sin\left(\frac{2\pi}{N}\right)$$

1.5 ## Z-Domain to time-Domain

z-domain ⟷ time domain

$H(z)=\sum_k h[n]z^{-k}$ — this was apparently not it :/

Useless :(

FIR Filter ~ Roots/zeros of system $X(z)$

→ FIR Filter Representation in three domains

1) TIME: $h[n]=\sum_k b_k\,\delta[n-k]$

2) FREQUENCY: $H(e^{jw})=\sum_k h[n]e^{-jwk}$
or $H(e^{jw})=H(z)\big|_{z=e^{jw}}$

3) Z-DOMAIN: $H(z)=\sum_k h[n]z^{-k}$

time ⟷ z-domain via $H(e^{jw})$

Input: M: number of roots
$\phi_k \times M$: Roots $\{z_k\}$ as angles ϕ_k; $z_k=e^{j\phi k}$

Output: - time-domain impulse response $h[n]$
constant factor = 1
- length of $h[n]$ should be $M+1$

z-transform of a filter

$$H(z)=G\prod_{k=1}^{M}(1-z_k\cdot z^{-1})=G\prod_{k=1}^{M}\left(\frac{z-z_k}{z}\right)$$

z_k = Zeros of the system
$G=1$

Examples: 1
1.047198; −1.047198

$a=0.5+0.866j$

$\rightarrow \underset{1}{(1-(0.5+0.866j)z^{-1})}\ \underset{2}{(1-(-0.5+0.866j)z^{-1})}$

$1+\ (0.5+0.866j)(-0.5+0.866j)$

$1+(-0.25\ \cancel{+0.5\times0.8j-0.5\times0.8j}\ -0.866j\times0.866j)z^{-2}$

$1-1z^{-2}$

1) convert angles(ϕ_k) into roots (z_k).

$z_k=e^{j\phi_k}=\cos(\phi_k)+j\cdot\sin(\phi_k)$

for simplicity we will keep calling these roots z_k

2) Sub into $H(z)=G\prod_{k=1}^{M}(1-z_k\cdot z^{-1})$ and expand. # this is the difficult part.
G in this assignment is 1

Lets use an example where we have 2 zeros

$$\begin{aligned}H(z)&=(1-z_1\cdot z^{-1})(1-z_2\cdot z^{-1})\\&=1-z_2z^{-1}-z_1\cdot z^{-1}+z_1\cdot z_2\cdot z^{-2}\\&=1-(z_1+z_2)z^{-1}+(z_1\cdot z_2)\cdot z^{-2}\end{aligned}$$

$\therefore h[n]=[1;-(z_1+z_2);(z_1\cdot z_2)]$
if $M=2$

Ok... 3 zeros:

$$\begin{aligned}H(z)&=(1-z_1\cdot z^{-1})(1-z_2\cdot z^{-1})(1-z_3\cdot z^{-1})\\&=[1-(z_1+z_2)z^{-1}+(z_1\cdot z_2)\cdot z^{-2}]\,[1-z_3\cdot z^{-1}]\\&=1-z_3\cdot z^{-1}-(z_1+z_2)z^{-1}+(z_1+z_2)z^{-1}\cdot z_3\cdot z^{-1}+z_1\cdot z_2\cdot z^{-2}-z_1\cdot z_2\cdot z^{-2}\cdot z_3\cdot z^{-1}\\&=1-z_3z^{-1}-z_1z^{-1}-z_2z^{-1}+z_1\cdot z_3\cdot z^{-2}+z_2\cdot z_3\cdot z^{-2}+z_1\cdot z_2\cdot z^{-2}-z_1\cdot z_2\cdot z_3\cdot z^{-3}\\&=1-\underset{a^1}{(z_1+z_2+z_3)}z^{-1}+\underset{a^2}{(z_1\cdot z_3+z_2\cdot z_3+z_1\cdot z_2)}\cdot z^{-2}-\underset{a^3}{(z_1\cdot z_2\cdot z_3)}z^{-3}\end{aligned}$$

*1 figured out a system

$$\therefore h[n]=[1;-(z_1+z_2+z_3);(\underset{B_1}{z_1\cdot z_3})+(\underset{B_2}{z_2\cdot z_3})+(\underset{B_3}{z_1\cdot z_2});-(\underset{B_1}{z_1\cdot z_2\cdot z_3})]$$

a_2

→ To translate this into code, I had the following idea of using an array for coefficients.
↳ Start with array of roots $[\cos(\phi_k)+j\sin(\phi_k);\ldots]$
↳ have an array of coefficients... coeffs = [ ]
↳ append the first "1" element

The idea of the coeff array is to have the indexes represent powers:
1: $z^0=1$ 2: z^{-1} 3: z^{-2}

I figured it out Bitch!!!
woop woop!

Lets use 10 roots:

$a_{10} = 1$ (assuming monic polynomial)

$a_9 = -(r_1 + r_2\ r_3 + \ldots + r_{10})$ (sum of single roots)

$a_8 = \sum_{i<j} r_i r_j$ (sum of products of pairs of roots)

$a_7 = (-1) \sum_{i<j<k} r_i \cdot r_j \cdot r_k$ (sum of all triplets of roots)

$a_6 =$ the pattern should be clear here.

Now for code we need 3 functions:

1) main function okay
2) compute combinations ~~difficult~~ NAH itertools.combinations (array, n-root)
3) product of combinations. easy

→ Well, this would have been great untill we came to a time complexity of $O(n^3)$.

↳ I found a new method!

Horner's Method → this algorithm is based on horner's rule in which a polynomial is written in nested form.