Cross-Correlation

April 30, 2025

Zero-center and then normalize a signal

Zero-center and then normalize a signal

- Consider a signal represented numerically as a vector $\boldsymbol{x}$
- Suppose we (for reasons explained later) want to **zero-center** this signal
  - That is, we want to subtract by its mean
- Additionally, suppose we want to normalize it.

- Then the new vector would be

$$\boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} \text{ where } y_i \equiv \frac{x_i - \mu_x}{\sigma_x} \cdot \frac{1}{\sqrt{n}}$$

  - n is the dimension of the vector (eg. 1024, for signals before upsampling)
  - μ is the mean of the vector, $\mu_x = \frac{1}{n}\sum_i x_i$
  - σ_x the standard deviation is

$$\sigma_x = \sqrt{\frac{1}{n}\sum_i (x_i - \mu_x)^2}$$

- We can see that the new vector $\boldsymbol{y}$ is properly normalized:

$$\begin{aligned}\|\boldsymbol{y}\|^2 \equiv \sum_i y_i^2 &= \sum_i \left(\frac{x_i - \mu_x}{\sigma_x} \cdot \frac{1}{\sqrt{n}} \right)^2 \\ &= \frac{1}{n} \left[\sum_i (x_i - \mu_x)^2 \right] \frac{1}{\sigma_x^2} \\ &= \cancel{\frac{1}{n}} \left[\sum_i \cancel{(x_i - \mu_x)^2} \right] \left[\cancel{\frac{1}{n}} \sum_j \cancel{(x_j - \mu_x)^2} \right]^{-1} = 1\end{aligned}$$

- To summarize, to have a **zero-centered, normalized** signal, we write[1]:

```
import numpy as np
signal = (signal - np.mean(signal)) / np.std(signal) / np.sqrt(n)
```

[1] see the comment to this answer on stackexchange and this section of Wikipedia

SciPy Signal Correlate Behavior

Length of the Result

- Consider two signals of length 256:

$$\boldsymbol{x} = [x_0, x_1, ..., x_{256}]$$
$$\boldsymbol{y} = [y_0, y_1, ..., y_{256}]$$

- With the default (`full`) mode, the cross-correlation produced by `scipy.signal.correlate()` would be $\boldsymbol{z}$ of length

$$\|\boldsymbol{x}\| + \|\boldsymbol{y}\| - 1 = 511$$

- That is,

$$\boldsymbol{z} = [z_0, z_1, ...z_{510}]$$

- Next we will examine the elements of $\boldsymbol{z}$ and see how they are calculated.

Behavior

- In our case, the dimensions of the signals are the same, $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 256$,
  - so $N \equiv \max(\|\boldsymbol{x}\|, \|\boldsymbol{y}\|) = 256$
- According to SciPy's documentation, `correlate()` behaves in the following way:

$$
\begin{aligned}
z_k &= \sum_{l=0}^{\|x\|-1} x_l \; y^*_{l-k+N-1} \\
&= \sum_{l=0}^{255} x_l \; y_{l-k+255}
\end{aligned}
$$

  where in the last line the complex conjugation is dropped, because our signals are real.
- Next, let's unpack the above, remembering that *out of bound values of y is set to zero.*

Behavior

- The formula in our simplified case is

$$z_k = \sum_{l=0}^{255} x_l \; y_{l-k+255}$$

- For $k = 0$, this says we take the dot product

$$\begin{pmatrix} x_0 \\ \vdots \\ x_{255} \end{pmatrix} \cdot \begin{pmatrix} y_{255} \\ \vdots \\ 0 \end{pmatrix}$$

- For $k = 1$,

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{255} \end{pmatrix} \cdot \begin{pmatrix} y_{254} \\ y_{255} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- For $k = 255$, there is no offset (zero **lag**):

$$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{255} \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{255} \end{pmatrix}$$

- For $k = 509$:

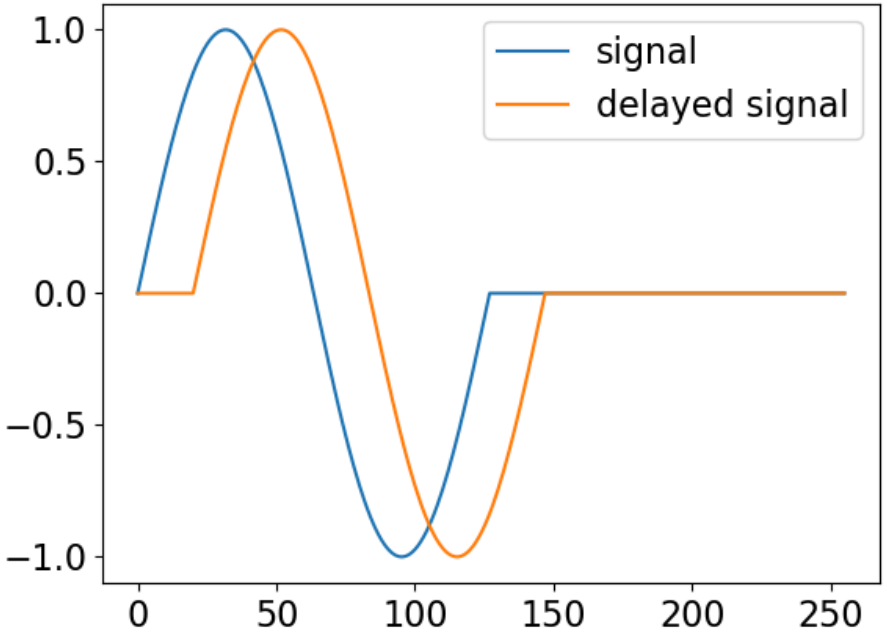
$$\begin{pmatrix} x_0 \\ \vdots \\ x_{254} \\ x_{255} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \vdots \\ y_0 \\ y_1 \end{pmatrix}$$

- Finally, for $k = 510$,

$$\begin{pmatrix} x_0 \\ \vdots \\ x_{255} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \vdots \\ y_0 \end{pmatrix}$$

Behavior

- In other words, what `correlate()` does is the following:
- Take two signals `sig1` and `sig2`
- "Slide" `sig2` by an integer number of indices (ie. lag)
- **Positive lag means "slide to the right".**
- Take the dot product of `sig1` with the *shifted* `sig2`.
- For example, we will see that a lag of -20 would produce the maximum correlation in the following figure
- This means we should shift the delayed signal to the left by 20 sample points for the two waves to align.



- In the above figure, we see that when the two vectors line up, their dot product would natually be maximal.

Zero Centered Normalized Cross Correlation

Significance of Normalization

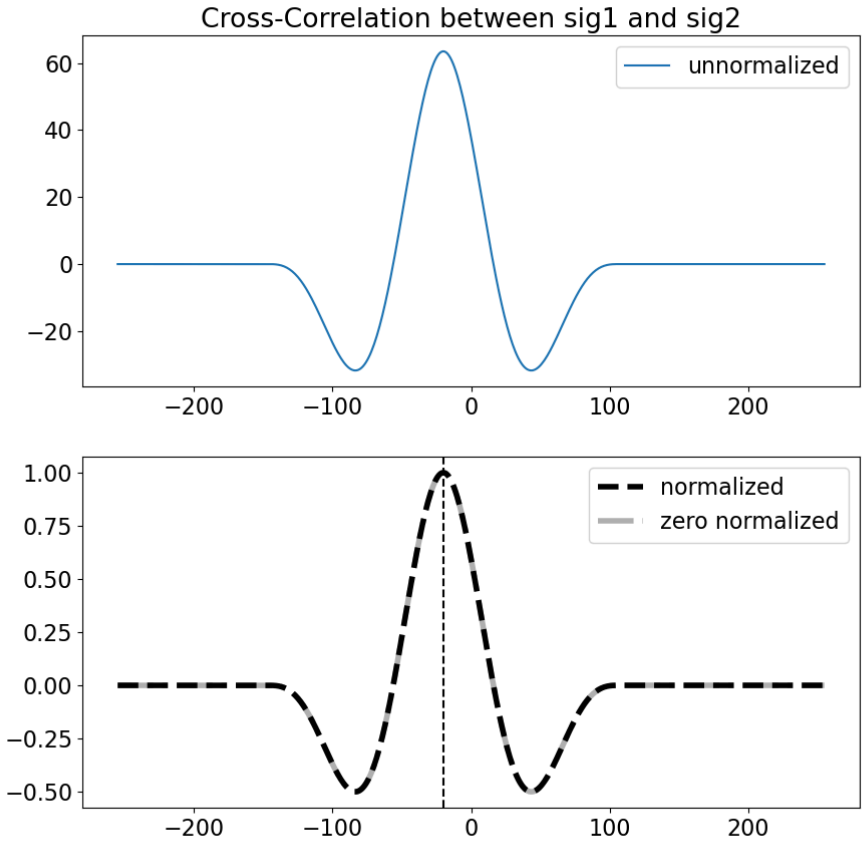
- Suppose we don't normalize the signals when we compute the cross-correlation.
- Then we don't really have a way to describe *how strong* the correlation is, since the correlation would depend on the signal amplitude and signal duration in this unnormalized case.
- As shown on the right, normalized vectors would allow us to have a bounded cross-correlation between $[-1, 1]$.
- This is due to the Cauchy-Schwarz inequality:

$$|\langle x|y\rangle|^2 \leq \langle x|x\rangle \cdot \langle y|y\rangle$$

So, when the vectors $|x\rangle$ and $|y\rangle$ are already normalized,

$$|\langle x|y\rangle|^2 \leq 1$$

,



Importance of Zero Centering

- If the waveforms have a DC offset, then the DC component could overwhelm the correlation[1]
- In this case one should compute the zero-centered normalized cross-correlation (ZNCC).
- Example: consider a sine "pulse" (`sig1`) and a delayed version of it (`sig2`)

```
signal_length = 256
pulse_length = 128
delay = 20

sig1 = np.zeros(signal_length)
sig2 = np.zeros(signal_length)
# make a single sine wave in the first half of signal 1
sig1[:pulse_length] = np.sin(np.linspace(0, 2*np.pi, pulse_length))
# for signal 2, delay the sine wave by 20 samples
sig2[delay:delay+pulse_length] = sig1[:pulse_length]
```

[1] see also [this answer on stackexchange](#).

Importance of Zero Centering

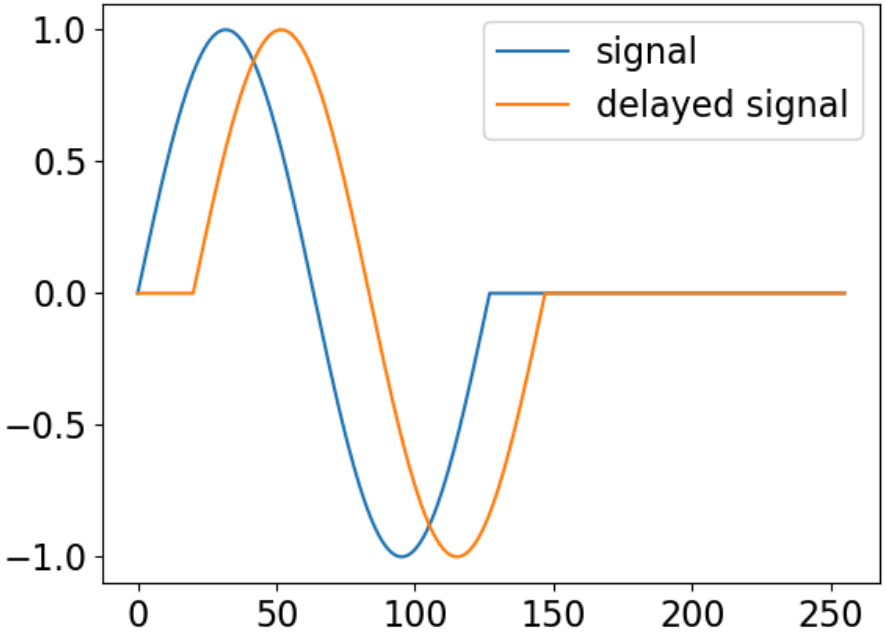


Figure 1: delay by 20 samples

- ZNCC and NCC would produce the same result in this case, since there is no DC offset
- ZNCC between `sig1` ($\boldsymbol{x}$) and `sig2` ($\boldsymbol{y}$) is

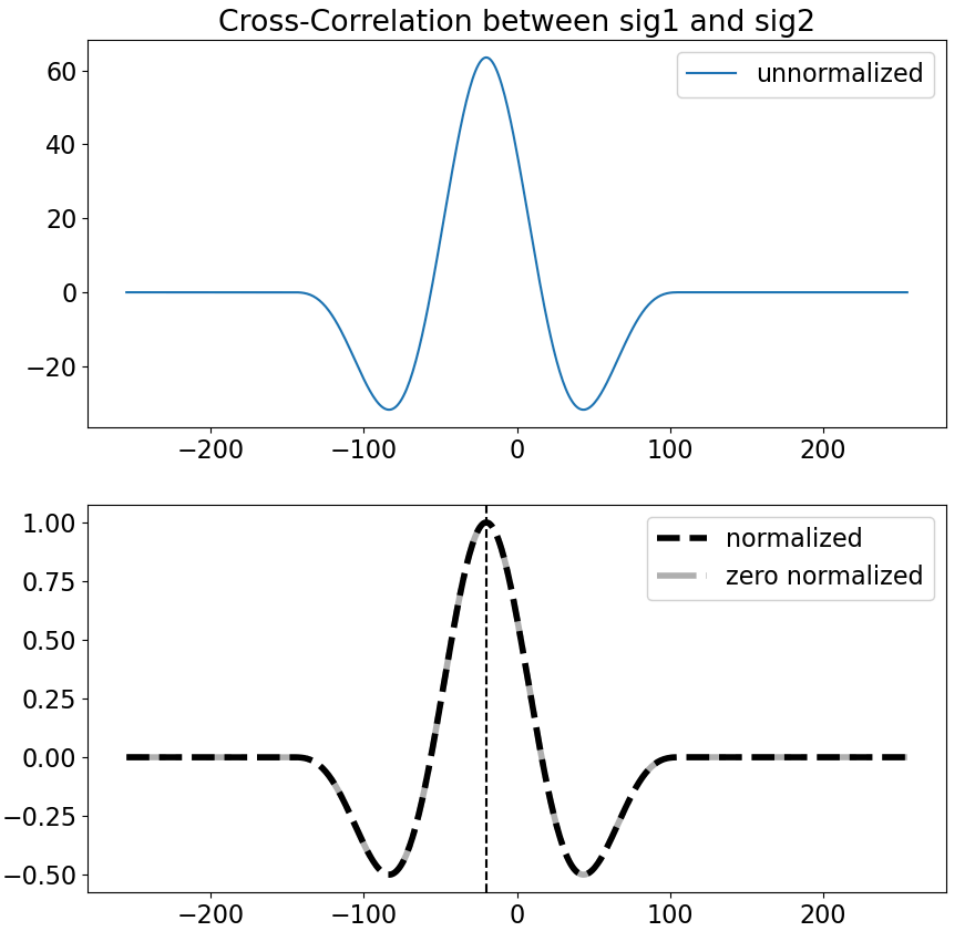
$$\frac{1}{n} \cdot \frac{1}{\sigma_x \sigma_y} \sum (x_i - \mu_x) \cdot (y_i - \mu_y)$$

  where n, σ, and μ are the signal length, standard deviation, and mean, respectively.

- Code snippet:

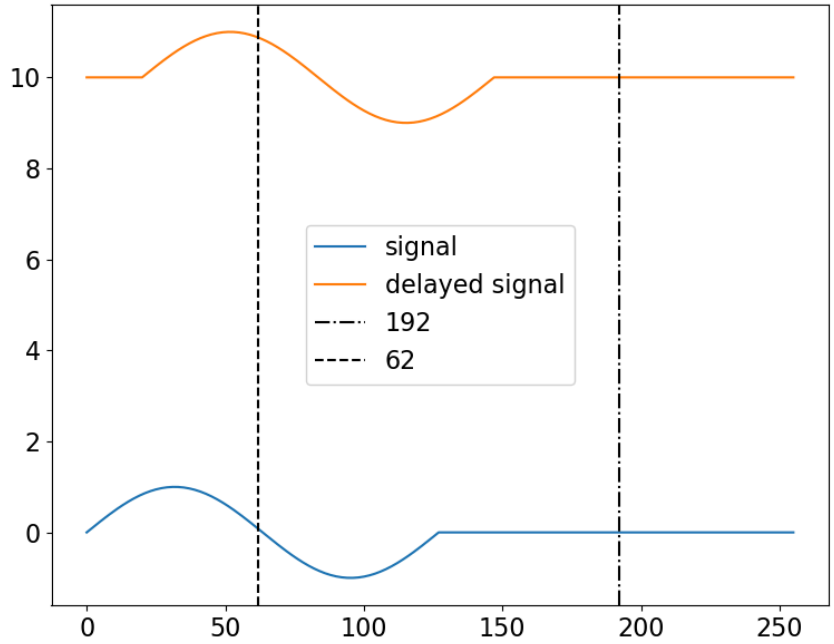
```
correlate(
  (sig1-np.mean(sig1))/np.std(sig1)
  (sig2-np.mean(sig2))/np.std(sig2)
) / signal_length
```

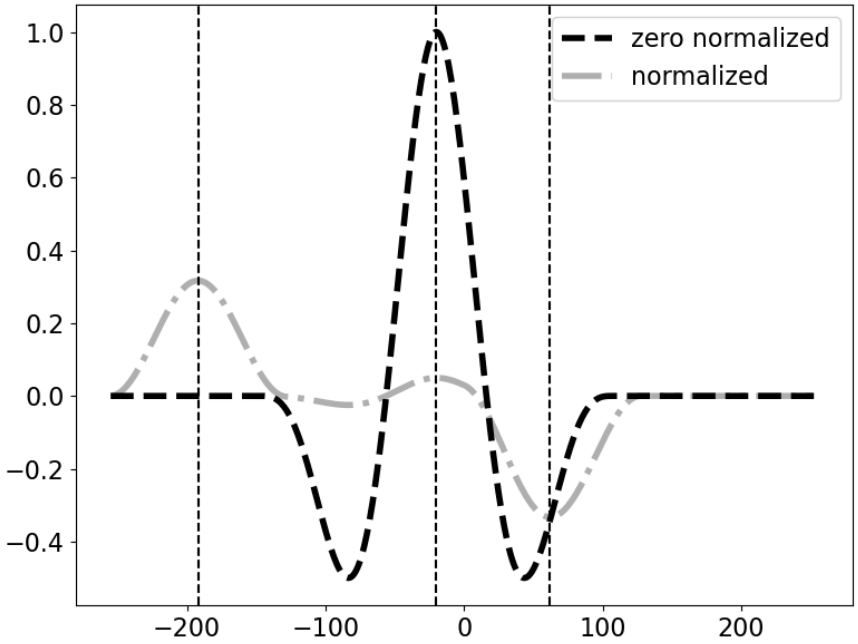
Importance of Zero Centering



- As shown, when the signals to be cross-correlated have no DC offset, then it does not matter whether one uses NCC or ZNCC.
- The peak at −20 means we should shift the delayed signal to the left by 20 samples, in order for the two signals to line up perfectly.
- Next, suppose delayed signal has a DC offset

Importance of Zero Centering

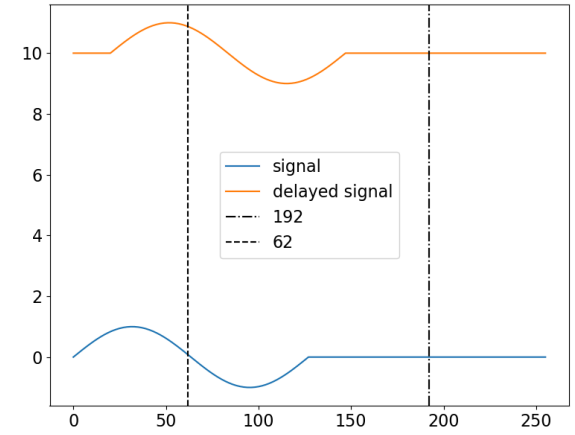


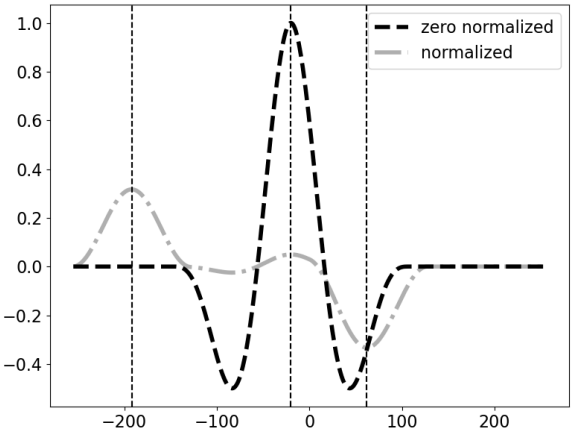


- The delayed signal now has a DC offset of 10.
- In this case, ZNCC is more desirable, as it still shows the peak at −20.

- Since what we care about is lining up the sine waves, we don't care about how the constant component of the delayed signal contributes to the correlation.

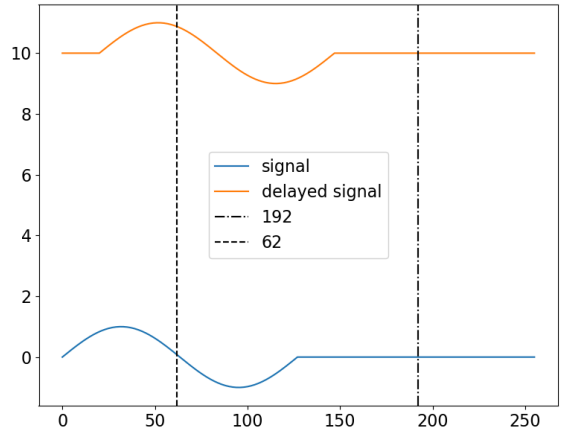
Some details of NCC

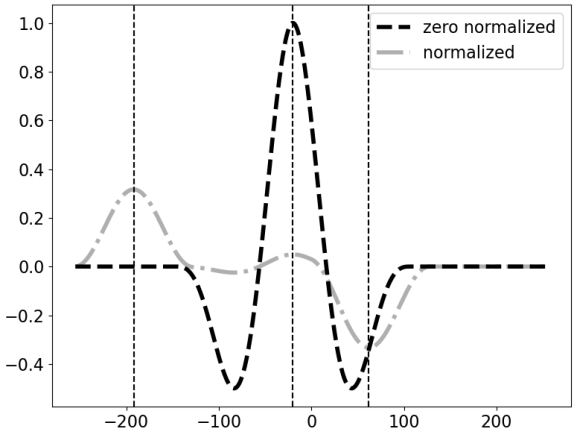




- Let's qualitatively examine the result of NCC to see how the DC offset could pollute our desired result
- At −192 samples there's a **peak**
  - This means to shift the delayed signal **to the left by 192 samples** and compute the dot product
  - This correlation score peak comes from multiplying the right-most piece of the sig2 with the left-most piece of the sig1 (i.e. the positive part of the sine)
  - The negative part of sig1 has no impact here, because out-of-range part of sig2 is zero-padded by SciPy.

Some details of NCC





- Let's qualitatively examine the result of NCC to see how the DC offset could pollute our desired result
- On the other hand, at 62 samples there's a **valley**
  - This means to shift the delayed signal **to the right by 62 samples** and compute the dot product
  - This correlation score valley comes from multiplying the left-most piece of the sig2 with the valley of the sig1 (i.e. the negative part of the sine)
  - The positive part of sig1 has no impact here, because out-of-range part of sig2 is zero-padded by SciPy.

More details

- Here's the code used for NCC

```
correlate(
  sig1/np.sqrt(np.sum(sig1**2)),
  sig2/np.sqrt(np.sum(sig2**2))
)
```

- Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the two signals, the code above amounts to an inner product:

$$\sum_i \frac{x_i}{\|\boldsymbol{x}\|} \cdot \frac{y_i}{\|\boldsymbol{y}\|} \equiv \langle \hat{\boldsymbol{x}} | \hat{\boldsymbol{y}} \rangle$$

  where $\| \cdot \|$ denotes the norm of the vector, and the $\widehat{\text{hat}}$ denotes unit vectors.

- Once again, the Cauchy-Schwarz inequality

$$\begin{aligned} |\langle \hat{\boldsymbol{x}} | \hat{\boldsymbol{y}} \rangle|^2 &\leq \langle \hat{\boldsymbol{x}} | \hat{\boldsymbol{x}} \rangle \langle \hat{\boldsymbol{y}} | \hat{\boldsymbol{y}} \rangle = 1 \\ \Rightarrow |\langle \hat{\boldsymbol{x}} | \hat{\boldsymbol{y}} \rangle|^2 &\leq 1 \end{aligned}$$

  ensures that the correlation score is bounded between $[-1, 1]$