Correlation of discrete systems

Definitions

The **correlation** (or cross-correlation) of two finite-energy discrete signals $x[n]$ and $y[n]$ is defined as:

$$r_{xy}[l] = \sum_{-\infty}^{\infty} x[n]y[n-l]$$

$$r_{xy}[l] = \sum_{-\infty}^{\infty} x[n+l]y[n]$$

This equation is similar to convolution, up to a reflection of the signal (the second term has $y[n-l]$ instead of $y[l-n]$). Thus correlation = convolution with a reflected signal, and vice-versa.

The argument l represents the delay of $y[n]$ compared to $x[n]$

The **auto-correlation** of a signal $x[n]$ is defined as the correlation of the signal with itself

$$r_{xx}[l] = \sum_{\infty}^{\infty} x[n]x[n-l] = \sum_{\infty}^{\infty} x[n]x[n+l]$$

Properties of correlation and auto-correlation

1. Symmetry:

   $$r_{xy}[l] = r_{yx}[-l]$$

   Proof: at whiteboard

2. Auto-correlation is an **even** signal:

   $$r_{xx}[l] = r_{xx}[-l]$$

   Proof: based on first property

3. Auto-correlation is maximum for $l = 0$

   $$r_{xx}[0] \geq r_{xx}[l], \forall l$$

   No proof

Interpretation

Graphical interpretation: on whiteboard

Usages of correlation and auto-correlation

Besides significant theoretical importance in theory, there are some straightforward applications of correlation in practice. A few examples are given below

1. Searching for a certain part in a large signal

When the two signals have both positive and negative values (roughly of similar length), the correlation signal will have a large value when the positive and negative areas match, and small values when they don't match. This can be used to locate a certain part within a large signal

```
%matplotlib inline
import matplotlib.pyplot as plt, numpy as np
x1 = np.array([1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1
x2 = np.hstack((np.random.randn(800), x1, np.random.randn(300)))
corr = np.correlate(x2, x1)
plt.figure(figsize=(18,6));
plt.subplot(1,3,1); plt.stem(x1); plt.title ('Signal to look for');plt.axis([0,
plt.subplot(1,3,2); plt.stem(x2); plt.title ('Signal to search in');
plt.subplot(1,3,3); plt.stem(corr); plt.title ('Correlation signal');
```

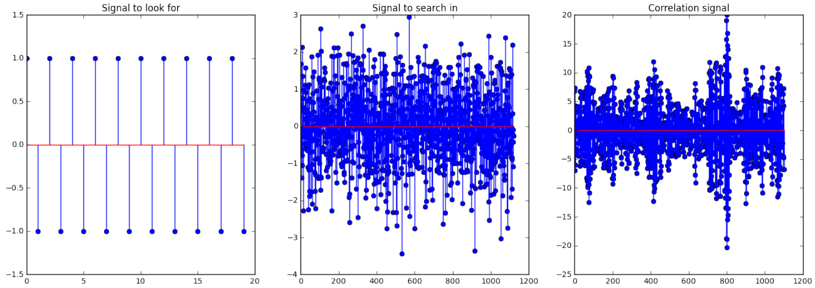

Figure 1

2. Estimating the delay of a signal

In radar-like systems, a signal pulse is sent from an emitter, gets reflected from a target and is received back with a lot of random noise added. We would like to estimate the delay of the received pulse. The correlation signal between the original block and the received signal will have a maximum when the original block overlaps with the block in the received signal. The position of the maximum indicates the delay.

```
%matplotlib inline
import matplotlib.pyplot as plt, numpy as np
x1 = np.array([1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1])
x2 = np.hstack((np.zeros(800), x1, np.zeros(300))) + 0.8*np.random.randn(800+30
corr = np.correlate(x2, x1)
plt.figure(figsize=(18,6));
plt.subplot(1,3,1); plt.stem(x1); plt.title ('Signal pulse sent');plt.axis([0,
plt.subplot(1,3,2); plt.stem(x2); plt.title ('Received signal, buried in noise'
plt.subplot(1,3,3); plt.stem(corr); plt.title ('Correlation signal');
```