EE599 Deep Learning – Homework 1

©K.M. Chugg, B. Franzke

January 9, 2019

Problem 1: Crowdsourcing a dataset of human-generated, random binary sequences

In this problem you will create an hdf5 file containing a numpy array of binary random sequences that you generate yourself. Here are the steps to follow:

1. Obtain and run the template python file provided – this is in DEBUG mode by default.
2. Experiment with the error trapping assert statements to see what they are doing, using the shape method on numpy arrays, etc.
3. Make the DEBUG flag False and enter 25 binary sequences, each of length 20. It is important that you do this by hand and without the aid of a compute random number generator or, for example, a coin.
4. Make sure that your hdf5 file has been written properly and can be read and checked to be correct.
5. Name your hdf5 file with your full name and submit it through the Canvas website with this assignment.

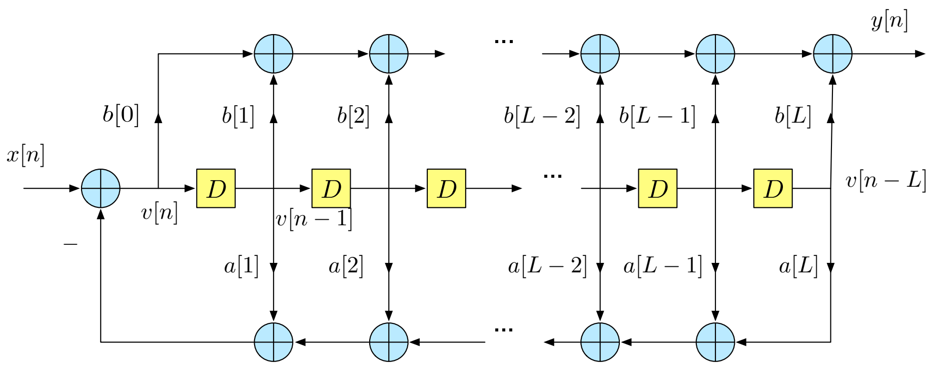
Problem 2: Using Filters in Python

This is a basic problem to build proficiency in Python. There are filter design and analysis tools in Python that are very similar to those in Matlab. In this problem you will design and plot the frequency response for a few ARMA filters. For your reference and review, Fig. 1.

1. Many methods in deep learning use a first order ($L = 1$) AR filter with unit gain at DC – *i.e.,* the gain of the frequency response at zero frequency is 1. Also by "AR" it is implied that $b[i] = 0$ for all $i > 0$. This type of filter has a single parameter α which is the pole location in the z-plane.

   (a) Specify the difference equation and Z-transform for this first order AR filter – *i.e.,* specify $b[0]$, and $a[1]$ in terms of α. Note that, in the standard form in Fig. 1 $a[0] = 1$ by default, but, just as in matlab, you need to enter for $a[0]$ in the Python routine.

   (b) Plot the magnitude of the frequency response for $\alpha = 0.9$, $\alpha = 0.5$, $\alpha = 0.1$, $\alpha = -0.5$. Use linear normalized frequency ν, which is unique on $[-1/2, +1/2]$ and has units of cycles per sample. **Hint:** `use scipy.signal.freqz`.



$$v[n] = x[n] - (a[1]v[n-1] + a[2]v[n-2] + \cdots a[L]v[n-L])$$

$$y[n] = b[0]v[n] + b[1]v[n-1] + b[2]v[n-2] + \cdots + b[L]v[n-L]$$

$$\text{state}[n] = (v[n-1], v[n-1], \ldots v[n-L])$$

implements this difference equation:

$$y[n] = \sum_{i=0}^{L} b[i]x[n-i] - \sum_{i=1}^{L} a[i]y[n-i]$$

Frequency response:

$$H(z) = \frac{b[0] + b[1]z^{-1} + b[2]z^{-2} \cdots + b[L]z^{-L}}{1 + a[1]z^{-1} + a[2]z^{-2} \cdots + a[L]z^{-L}} \qquad z = e^{j\pi\nu}$$

Figure 1: The general format and notation for an ARMA filter. The arrays of AR coefficients $a[n]$ and MA coefficients $b[n]$ define the filter. The order of teh filter is the number of delay elements in thsi diagram, L.

(a) If the time constant is defined as the number of samples required for the impulse response to decay to 20 % of its value at $n = 0$, give the time constant for $\alpha = 0.9$, $\alpha = 0.5$, $\alpha = 0.1$.

(b) Design an $L = 4$ Butterworth filter with bandwidth of $\nu_0 = 0.25$. Provide the AR and MA coefficients and plot the magnitude of the frequency response.

(c) Create a numpy array of length 300 comprising iid realizations of a standard normal distribution. Filter this sequence using the Butterworth filter and plot the input and output signals on the same plot. **Hint:** Use `scipy.signal.lfilter`.