EE599 Deep Learning – Homework 1

January 22, 2020

Problem 1: Working with data in hd5 format

In this problem you will create an hd5 file containing a numpy array of binary random sequences that you generate yourself. The hd5 format is useful because it allows one to store multiple data objects in a single file with object names – e.g., you can store 'regressor' and 'label' datasets where the former is a nupy array of floats and the latter is a numpy integer array. Also, hd5 allows for fast, non-sequential access to data in the dataset without reading the entire dataset – e.g., one can access `x[ idxs ]` where `idxs = [ 2, 234, 512]` efficiently. The access property is useful when pulling a batch of data from a large dataset in the process of training.

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 hd5 file has been written properly and can be read and checked to be correct.
5. Name your hd5 file with your full name and submit it through the Canvas website with this assignment.

Problem 2: A simple feedforward (MLP) neural network

An MLP has two input nodes, one hidden layer, and two outputs. The activation for the hidden layer is ReLu. The output layer is linear (*i.e.,* identity activation). The two sets of weights and biases are given by

$$\mathbf{W}_1 = \begin{bmatrix} 1 & -2 \\ 3 & 4 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mathbf{W}_2 = \begin{bmatrix} 2 & 2 \\ 3 & -3 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$$

What is the output activation when the input is $\mathbf{x} = [\ +1\ \ -1\ ]^{\mathrm{t}}$?

Note: the ReLu activation is $h(x) = x$ for $x > 0$ and $h(x) = 0$ for $x \leq 0$.

Problem 3: Convolutions and correlations

Convolutions and correlations are covered in a typical signals and systems class in your undergraduate degree. These are used in Convolutional Neural Networks (CNNs), so this is a brief review problem. For a one-dimensional, discrete time (index) signals, convolution is defined as

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

and a correlation is defined as

$$r[n] = x[n] \star h[n] = \sum_{k=-\infty}^{\infty} x[k]h[k+n]$$

For two-dimensional signals (e.g., images and image filters) convolution is

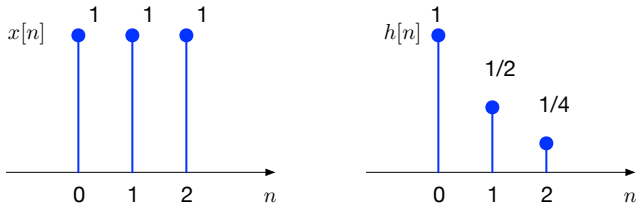
$$y[i][j] = x[i][j] * h[i][j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[m][n]h[i-m][j-n]$$

and correlation is

$$r[i][j] = x[i][j] \star h[i][j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[m][n]h[m+i][n+j]$$

1. Show that $y[n] = x[n] * h[n] = x[n] \star h[-n]$ – i.e., that correlation is convolution with the reflected (reversed) signal. State and show the relationship between convolution and correlation in the 2D case.

2. Find $y[n] = x[n] * h[n]$ for the signals shown below. Plot the result in python using a stem-plot.



3. Find $r[i][j] = x[i][j] \star h[i][j]$ for the signals shown below. Plot a heat-map of the result using matplotlib.pyplot.matshow.

⋱	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋰
…	0	0	0	0	0	0	0	…
…	0	$x[-2][1]=1$	$x[-1][1]=1$	$x[0][1]=1$	$x[1][1]=1$	$x[2][1]=1$	0	…
…	0	$x[-2][0]=1$	$x[-1][0]=1$	$x[0][0]=1$	$x[1][0]=1$	$x[2][0]=1$	0	…
…	0	$x[-2][-1]=1$	$x[-1][-1]=1$	$x[0][-1]=1$	$x[1][-1]=1$	$x[2][-1]=1$	0	…
…	0	0	0	0	0	0	0	…
⋰	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

and

$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\iddots$
$\dots$	0	0	0	0	0	$\dots$
$\dots$	0	$h[-1][1] = 1/4$	$h[0][1] = 1/2$	$h[1][1] = 1/4$	0	$\dots$
$\dots$	0	$h[-1][0] = 1/2$	$h[0][0] = 1$	$h[1][0] = 1/2$	0	$\dots$
$\dots$	0	$h[-1][-1] = 1/4$	$h[0][-1] = 1/2$	$h[1][-1] = 1/4$	0	$\dots$
$\dots$	0	0	0	0	0	$\dots$
$\iddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Note: In both cases, all values at indices not shown are assumed to be zero and you only need to show the output over the region where it is non-zero.

Problem 4: 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.

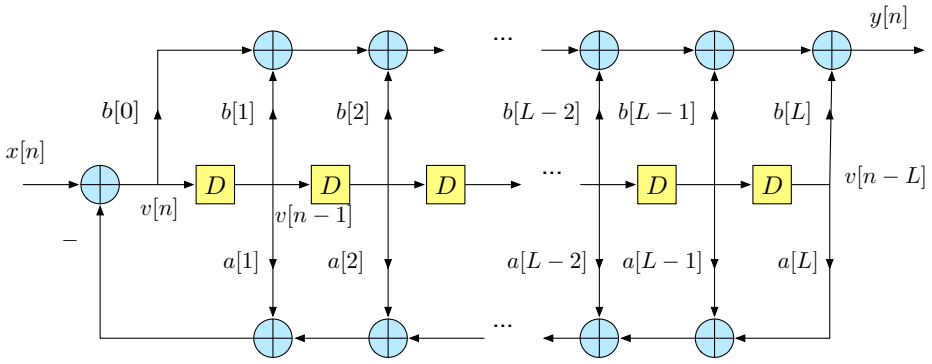
      i. **Note:** enough information has been given above to fully specify this simple filter. If you do not see that, this is a brief review of signals and systems and you can get help from the TAs.

   (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`.

   (c) 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 $\alpha = 0.9$, $\alpha = 0.5$, $\alpha = 0.1$.

   Python has most of the matlab functionality for filter design. These sub-problems are aimed at getting you experience using those tools.

   (a) Design an $L = 4$ Butterworth filter with bandwidth of $\nu_0 = 0.25$ – here normalized linear frequency is denoted by ν in cycles/sample – i.e., the frequency response is periodic in ν with period one. Provide the AR and MA coefficients and plot the magnitude of the frequency response.



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

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

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

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:

$$H(z) = \frac{b[0] + b[1]z^{-1} + b[2]z^{-2} \cdots + b[L]z^{-L}}{1 + a[1]z^{-1} + a[2]z^{-2} \cdots + a[L]z^{-L}} \qquad z = e^{j2\pi\nu}$$

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.

(b) 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`.