Assignment 7: Neural Nets

Assignment written by Rett Bull, Pomona College with changes
by Nathan Shelly and Sara Sood, Northwestern University.

In this assignment, we will experiment with a few simple neural networks. The idea is to obtain preliminary answers to questions like "How quickly can a network be trained?" "How reliable (or repeatable) is the training?" and "How accurate is the resulting network?"

For this assignment, you will do some work at the python interpreter (the IDLE Shell window). Save a summary of what you do (python commands and all) into a file called `a7.py`. You will be submitting this file at the end.

You will be using the `neural.py` file that was included with this assignment. At the top of your a7.py file, be sure to include the following: `from neural import *`. The asterisk means import everything from the `neural.py` file.

1. Construct a network with two hidden nodes and train it on the XOR data below. Notice that we have complete information about the function we are trying to approximate - an unusual situation for a neural network.
   - How quickly (after how many training iterations) do the weights converge? Converging means the change in error over 100 iterations (the default `print_interval`) becomes negligible (for the purposes of this assignment negligible means the change in error is less than .0005: $\delta_w < .0005$. *Hint:* you may need to increase the iterations of the network by passing a number greater than the default 1000 for the `iters` argument of `train`.
   - How well does the resulting network perform - What is the final error?
   - Try training several different networks to see the variation in the convergence rate. You may also want to pass in a smaller `print_interval` to `train` to get finer information about the changes in the error.

   Table 1: XOR

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

2. Repeat part 1 with a network with eight hidden nodes. Does the convergence go faster? Is the resulting network a better approximation to the XOR function? Notice that "faster" can mean "fewer iterations" or "less clock time." A network with more nodes will have more weights to adjust and will take more clock time for each training cycle, but it may require fewer cycles and the total time may be shorter.

3. See what happens when you repeat part 1 using a network with just one hidden node. The XOR function cannot be computed with a single node.

4. Table 2 contains a sampling of voter opinions. The idea is to deduce a voters party affiliations from their views on the importance of various topics. Six voters were asked to rate the importance of five issues on a scale from 0 to 1 and to identify themselves as Democrat (0) or Republican (1). Write a

neural network and train it on the data in Table 2. Then try it on the samples from Table 3 or other cases of your own creation. Can you explain the conclusions drawn by the network?

Table 2: A sampling of voter opinions taken from Dave Reed of Creighton University

Budget	Defense	Crime	Environment	Social Security	Party
0.9	0.6	0.8	0.3	0.1	1.0
0.8	0.8	0.4	0.6	0.4	1.0
0.7	0.2	0.4	0.6	0.3	1.0
0.5	0.5	0.8	0.4	0.8	0.0
0.3	0.1	0.6	0.8	0.8	0.0
0.6	0.3	0.4	0.3	0.6	0.0

Table 3: Some test cases for the voter network

Budget	Defense	Crime	Environment	Social Security	Party
1.0	1.0	1.0	0.1	0.1	?
0.5	0.2	0.1	0.7	0.7	?
0.8	0.3	0.3	0.3	0.8	?
0.8	0.3	0.3	0.8	0.3	?
0.9	0.8	0.8	0.3	0.6	?

Submitting Your Code

Copy and paste all code from the interpreter into your a7.py file. **Comment out all lines of code**. Upload that file to Canvas.

Optional Advanced Extra Work

Check out data sets here.

Try to write code to parse a dataset (they are csv files) into data that you can use to train a neural network.