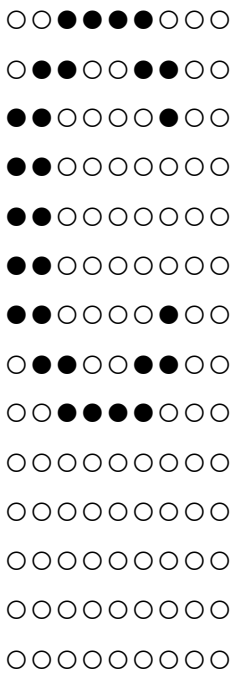
CSCE 420 Project Specification
Dr. Daugherity
Due: 11:59 P.M. Thursday, April 25, 2019

This project is to design and write a graphical character recognition program using neural networks and to explore their properties. The input will be a 9 by 14 dot pattern and the neural network will try to identify the letter. For example, if the input is



the neural net should identify the letter as C (although there may very well be less intense outputs for other similar letters such as G and O).

Use the 26 upper-case letters in the IBM EGA 9 by 14 dot matrix font (downloadable from https://farsil.github.io/ibmfonts/ in BDF format) and implement a neural net with 126 inputs (the dots) and 26 outputs A to Z (upper-case only). A minimum of two layers (columns) of neurons is required, but you will likely need more than two layers for good accuracy.

Implement the corrected back propagation algorithm posted on Piazza in C++, Java, or Python (with the usual homework constraints) and train the neural net until it is fairly accurate (meaning the largest output is the correct letter 90% or so of the time). Capture the accuracy after each back-propagation pass and plot versus pass number to show the learning. When training is done, write the weights to a file so you can read them back in for testing at a later time.

Hints: For the initial weights try random numbers between, say, -0.1 and +0.1 (but not zero). Experiment to find a useful value for α, the learning rate. You might try 0.1 or smaller, or an decreasing schedule like 0.1 divided by the pass number. Before tackling

this character-recognition problem, first debug your back-propagation code on something simple, like training a single neuron with 3 inputs (plus the constant 1 on a_0) to perform the majority function (if 2 or 3 of the inputs are 1's, output a 1; else output a 0).

Once you have trained the character recognition neural net, see how it performs when random input dots are flipped (black to white or white to black); this represents noise in scanned input. Test each of the 26 letters by progressively flipping bits at random (but without repetition) until the largest output is no longer correct; this gives you an estimate of the noise immunity of your neural net. Tabulate the number of flips each letter requires to make the output incorrect.

For the output layer use the logistic sigmoid activation function, but for all other layers use the hyperbolic tangent sigmoid activation function. Hint: Training and execution will be faster if you use table lookup or a decent approximation instead of calling the *exp* and *tanh* library functions at every step.

Note that back propagation requires both the activation function and its derivative, but the derivative can easily be calculated by

$$g'(x) = g(x)(1 - g(x))$$

for the logistic sigmoid, and

$$g'(x) = 1 - (g(x))^2$$

for the hyperbolic tangent sigmoid.

Your program should run on compute.cse.tamu.edu and must be submitted both to CSNET and also on a CD or DVD or USB jump drive (which cannot be returned).

The project report (described below) should be submitted on paper in class, along with your CD or DVD or USB jump drive. Write a report according to the outline below.

REPORT OUTLINE

The project report must be printed on a laser printer. The report should include the following sections:

1. Statement of the problem, significance, etc.
2. Restrictions and limitations
3. Explanation of your approach (*e.g.*, analysis and experiments to choose how many layers, how many neurons in each layer, learning rate and schedule, number of back-propagation iterations, etc.)
4. Sample run (screen shots)
5. Results and analysis
6. Conclusions - What did you show? What did you learn?

7. Future research (how your program could be improved or extended)
8. Instructions on how to run your program
9. Listing of the COMMENTED program
10. Bibliography - references used (including the source for the font)