Skylar Wurster

Neural Networks

Professor Morris

February 15th, 2018

Lab 1 Report

Summary: A matrix based approach to the network was taken, where each web of weights or biases formed a matrix to be used for computation. This results in a cleaner implementation with less for loops than the iterative coding solution, and also supports batch updating with multiple inputs at once. For example, instead of a single input of `double[4][1]`, we can have `double[4][n]` with n inputs. Similarly, double[n][1] expected results.

Inside of the `NeuralNets.pde` file holds the main method (setup), where the controls for the GUI are located. A number of inputs, outputs, hidden layers, and nodes per hidden layer must be specified before clicking initialize. To choose nodes per layer, separate desired sizes with a space. For instance, if your goal is for two hidden layers of size 4 and 3, set the “number of hidden layers” field to “2”, and the “nodes per hidden layer” field to “4 3”. If no hidden layers are desired, set the “hidden layers” field to “0”, and the “nodes per hidden layer” field can be ignored.

`Networks.pde` holds the neural network information. Here, you’ll find the SigmoidNetwork being used, and the MPNetwork used for homework 1. Both extend from the parent class Network. The network must first be initialized through `Initialize()`. This is used to create the network architecture and randomize weights and biases between -1 and 1.

From here, the main method will create a thread to train the network using the `Train(double[], double[])` function. This takes the input and desired output and performs weight updates based on feedforward and backpropagation.

Testing went well. By using a seeded random generator, I was able to start my weights and biases at the same numbers each time, giving enlightening results. For learning rates 0.25 to 0.5, training had a pattern. Number of epochs for convergence would increase by about 2500 to 10000 for each 0.05 subtracted from 0.5, meaning higher learning rates converged quicker. Then from 0.2 and under, convergence would happen much quicker! In general, it followed the same pattern, where 0000 through 1110 would converge quick, but 1111 would have a high error, and then eventually that would drop as well. This is visible in two "intelligence explosions", visualized through the weights' colors changing rapidly.

Additional observations include that, in general, the network converges faster with a larger hidden layer, to some limit. After some point, the network takes too long to converge, but from 16-32 nodes, the network converged in under 10000 epochs. With 64 nodes in my hidden layer training was not finished after 100000 epochs. Deep networks are similar – they take longer with larger layers, but smaller layers (say 4 nodes and 3 nodes) are just as quick as a single hidden layer. Another observation is Figure 3. Both training with momentum and without produce a similar curve, but offset on the x-axis. This may be due to momentum using some of the last update, so the weight adjustment will do nearly twice what it would without momentum. That is, momentum updating is similar to non-momentum updating, except with the learning rate doubled. This is why we see similarities at learning rate 0.1 from momentum and 0.2 from non-momentum updates.

Appendix

η	Number of Epochs
0.05	31040
0.1	17148
0.15	11690
0.2	11820
0.25	55054
0.3	42710
0.35	35711
0.4	30992
0.45	27584
0.5	25018

Figure 1: learning rate affecting convergence without momentum

η	Number of Epochs
0.05	17526
0.1	10777
0.15	45654
0.2	32749
0.25	26275
0.3	22420

0.35	19935
0.4	18275
0.45	17164
0.5	16457

Figure 2: learning rate affecting convergence with momentum and $\alpha = 0.9$

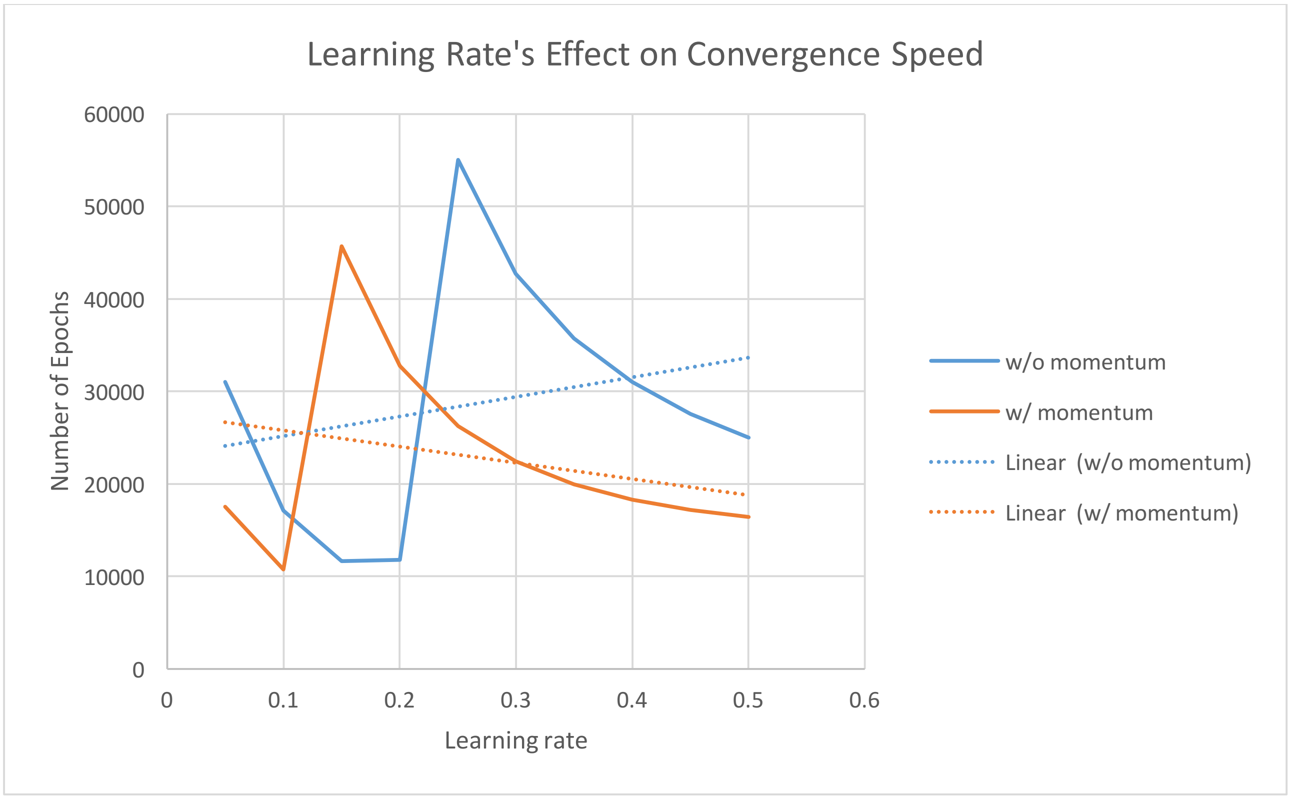
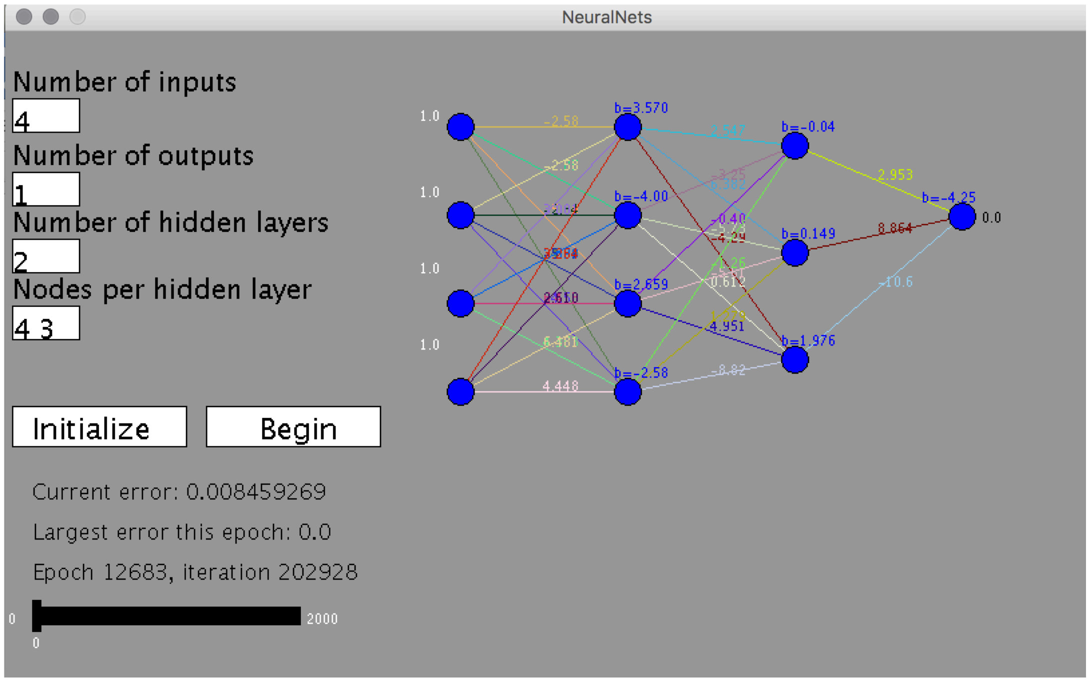


Figure 3: a graph showing how learning rate and momentum affects convergence speed

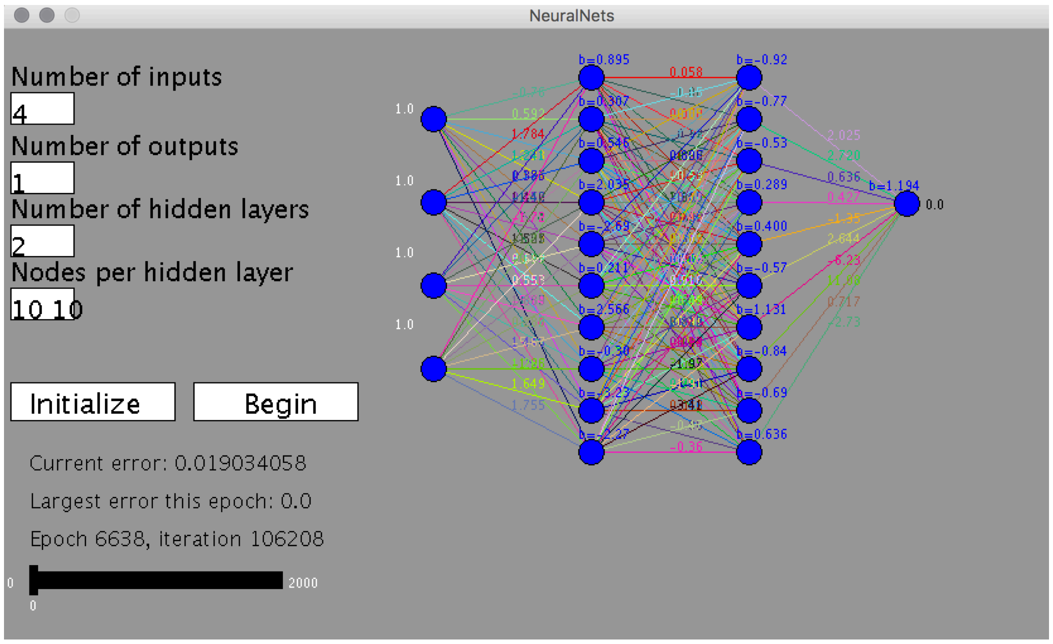

NeuralNets

Number of inputs
4
Number of outputs
1
Number of hidden layers
2
Nodes per hidden layer
4 3

Initialize   Begin

Current error: 0.008459269
Largest error this epoch: 0.0
Epoch 12683, iteration 202928

Figure 4: Example of the NeuralNets applet GUI after a deep network was trained on the project function


NeuralNets

Number of inputs
4
Number of outputs
1
Number of hidden layers
2
Nodes per hidden layer
10 10

Initialize   Begin

Current error: 0.019034058
Largest error this epoch: 0.0
Epoch 6638, iteration 106208

Figure 5: A slightly deeper network converging in 6638 epochs