Andrew Cohen
PHYS 250
11/12/20

HW 3 Writeup

The trained perceptron in FORTRAN performed as expected. After testing the Sigmoid function on a range of alpha values, α=1.5 minimized the error of the perceptron. Examples of the tests are shown:

```
andrewscohen@linux4:~/HW3$ ./perceptron
 Trained Outputs
   3.01775010E-03
  0.997537494
  0.997992039
   2.46099755E-03
 Enter Data
 BYE TO EXIT
0,1,0
  0.449051678
 Enter Data
 BYE TO EXIT
0,0,0
  0.500000000
 Enter Data
 BYE TO EXIT
1,0,0
  0.999993920
 Enter Data
 BYE TO EXIT
1,1,0
  0.999992490
 Enter Data
 BYE TO EXIT
bye
```

```
andrewscohen@linux5:~/HW3$ ./percept
 Trained Outputs
   4.27904353E-03
  0.996508777
  0.997155190
   3.48735088E-03
 Enter Data
 BYE TO EXIT
0,1,0
  0.448834240
 Enter Data
 BYE TO EXIT
0,0,0
  0.500000000
 Enter Data
 BYE TO EXIT
1,0,0
  0.999987721
 Enter Data
 BYE TO EXIT
1,1,0
  0.999984980
```

```
andrewscohen@linux5:~/HW3$ ./percept
 Trained Outputs
   2.12992425E-03
  0.998261511
  0.998581886
   1.73751311E-03
 Enter Data
 BYE TO EXIT
0,1,0
  0.449169934
 Enter Data
 BYE TO EXIT
0,0,0
  0.500000000
 Enter Data
 BYE TO EXIT
1,0,0
  0.999997020
 Enter Data
 BYE TO EXIT
1,1,0
  0.999996305
```

Although the ANN performed well at identifying inputs that result in an output of 1, it failed to successfully identify inputs resulting in 0. A recurring problem is that the perceptron always returned the values 0.5 for the (0,0,0) input – this likely affected the other 0 outputs. This is because the sigmoid function for z=0 is 0.5. Because of this issue, I attempted to use (0,0,0) in the training set. This improved the performance of the ANN overall. To further explore this, I expanded the training set, once again including (0,0,0) but this time adding (0,0,1) back into the training. I expected to see increased performance, however, instead the ANN now failed to correctly recognize the (0,1,0) input.

<u>Training Set with (0,0,0) in place of (0,0,1)</u>

```
andrewscohen@linux5:~/HW3$ ./percept2
 Alpha is   1.50000000
 Trained Outputs
  0.500000000
  0.997406304
  0.999954224
   2.90428312E-03
 Enter Data
 BYE TO EXIT
0,0,1
  0.142030686
 Enter Data
 BYE TO EXIT
0,1,0
   1.72907822E-02
 Enter Data
 BYE TO EXIT
1,1,0
  0.999569714
 Enter Data
 BYE TO EXIT
1,0,0
  0.999992371
```

<u>Expanded Training Set</u>

```
andrewscohen@linux4:~/HW3/SIGMOID$ ./percept3
 Alpha is   1.50000000
 Trained Outputs
  0.500000000
  0.997991443
  0.998361766
   2.00746558E-03
   2.46110745E-03
 Enter Data
 BYE TO EXIT
0,0,1
   2.46109464E-03
 Enter Data
 BYE TO EXIT
0,1,0
  0.449128300
 Enter Data
 BYE TO EXIT
1,1,0
  0.999994993
 Enter Data
 BYE TO EXIT
1,0,0
  0.999995947
 Enter Data
 BYE TO EXIT
```

Andrew Cohen
PHYS 250
11/12/20

After these failed attempts I began using Tanh as my transfer function instead of the sigmoid function. The results were greatly improved. Not only was the Tanh function faster compared to the sigmoid (the CPU_TIME of the entire training cycle was measured), but the errors were also much lower and the Tanh function was able to deliver much better results for inputs with corresponding outputs of 0.

```
andrewscohen@linux5:~/HW3/TANH$ ./tanh_percept
 Trained Outputs
   1.25523784E-05
  0.997493327
  0.997493327
   1.25510514E-05
 Alpha=  0.100000001     Total Time=  0.650498986
 Enter Data
 BYE TO EXIT
0,0,0
   0.00000000
 Enter Data
 BYE TO EXIT
0,1,0
  -1.31824518E-09
 Enter Data
 BYE TO EXIT
1,0,0
  0.997493267
 Enter Data
 BYE TO EXIT
1,1,0
  0.997493267
 Enter Data
 BYE TO EXIT
```

```
andrewscohen@linux5:~/HW3/TANH$ ./tanh_percept
 Trained Outputs
   2.50329526E-06
  0.998880923
  0.998880923
   2.50324774E-06
 Alpha=  0.500000000     Total Time=   1.52316904
 Enter Data
 BYE TO EXIT
0,0,0
   0.00000000
 Enter Data
 BYE TO EXIT
0,1,0
  -3.78577170E-11
 Enter Data
 BYE TO EXIT
1,0,0
  0.998880923
 Enter Data
 BYE TO EXIT
1,1,0
  0.998880923
 Enter Data
 BYE TO EXIT
```

```
andrewscohen@linux5:~/HW3/TANH$ ./tanh_percept
 Trained Outputs
   1.66840937E-06
  0.999086499
  0.999086499
   1.66853090E-06
 Alpha=  0.750000000     Total Time=  0.675518990
 Enter Data
 BYE TO EXIT
0,0,0
   0.00000000
 Enter Data
 BYE TO EXIT
0,1,0
  -1.22810206E-10
 Enter Data
 BYE TO EXIT
1,0,0
  0.999086499
 Enter Data
 BYE TO EXIT
1,1,0
  0.999086499
 Enter Data
 BYE TO EXIT
```

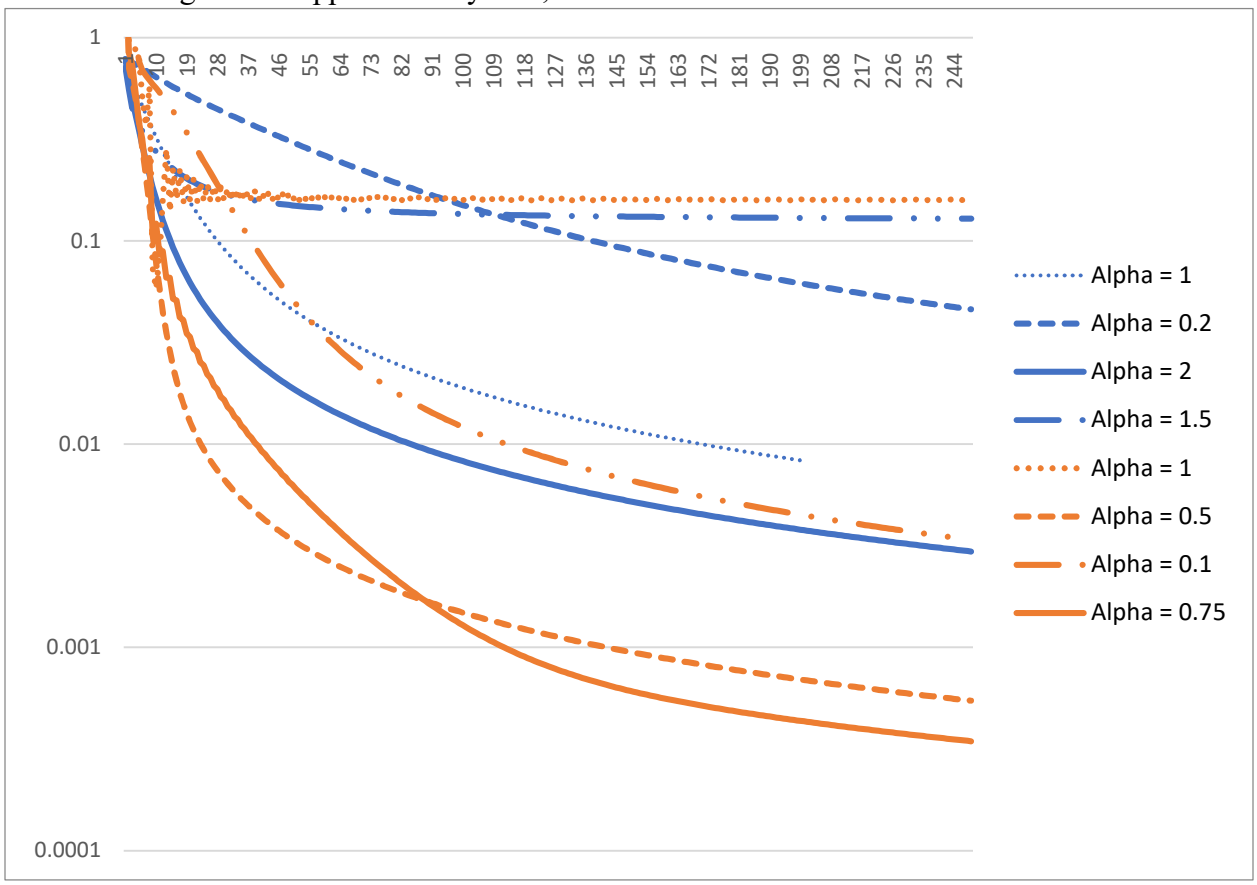
```
andrewscohen@linux5:~/HW3/TANH$ ./tanh_percept
 Trained Outputs
   2.08555616E-06
  0.998978555
  0.998978555
   2.08540723E-06
 Alpha=  0.600000024     Total Time=  0.709133029
 Enter Data
 BYE TO EXIT
0,0,0
   0.00000000
 Enter Data
 BYE TO EXIT
0,1,0
  -3.15070608E-11
 Enter Data
 BYE TO EXIT
1,0,0
  0.998978555
 Enter Data
 BYE TO EXIT
1,1,0
  0.998978555
 Enter Data
 BYE TO EXIT
```

```
andrewscohen@linux5:~/HW3/TANH$ ./tanh_percept
 Trained Outputs
   1.78759569E-06
  0.999054372
  0.999054372
   1.78760331E-06
 Alpha=  0.699999988     Total Time=  0.640735984
 Enter Data
 BYE TO EXIT
0,0,0
   0.00000000
 Enter Data
 BYE TO EXIT
0,1,0
  -1.01703732E-11
 Enter Data
 BYE TO EXIT
1,0,0
  0.999054372
 Enter Data
 BYE TO EXIT
1,1,0
  0.999054372
 Enter Data
 BYE TO EXIT
```

```
andrewscohen@linux5:~/HW3/TANH$ ./tanh_percept
 Trained Outputs
  0.126883447
  0.999473870
  0.999385536
  0.202372387
 Alpha=  0.800000012     Total Time=  0.633453012
 Enter Data
 BYE TO EXIT
0,0,0
   0.00000000
 Enter Data
 BYE TO EXIT
0,1,0
  -7.74774253E-02
 Enter Data
 BYE TO EXIT
1,0,0
  0.999207020
 Enter Data
 BYE TO EXIT
1,1,0
  0.999073923
 Enter Data
 BYE TO EXIT
```

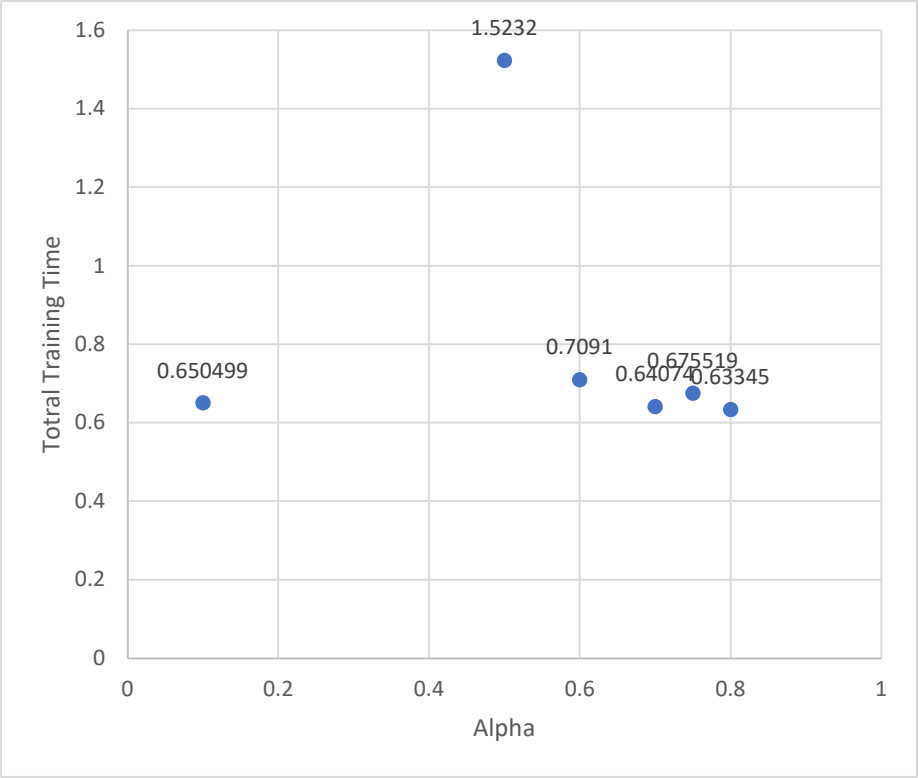
Andrew Cohen
PHYS 250
11/12/20

The results from the Tanh function were incredibly accurate and overall performed better than the Sigmoid function. The Error vs. Epoch for multiple alpha values for both functions is graphed below in order to optimize the learning rate. Based on the data, the optimum learning rate for the Sigmoid is approximately $\alpha=2$, for the Tanh it is $\alpha=0.75$.



Andrew Cohen
PHYS 250
11/12/20

Using the Tanh function, I wished to investigate the effect learning rate had on total training time. My data below suggests that there is no obvious correlation between learning rate and training time.



I attempted to use my 1-layer perceptron on the XOR training set. The result was, as predicted, a failure and an indication that more than one layer is required for this more advanced logic process.

```
andrewscohen@linux5:~/HW3$ ./XOR
 Trained Outputs
  0.363266587
  0.406083196
  0.406083196
  0.447188228
 XOR TESTAlpha=  0.750000000     Total Time=  0.599664032
 Enter Data
 BYE TO EXIT
0,0,0
   0.00000000
 Enter Data
 BYE TO EXIT
1,1,0
  0.298490494
 Enter Data
 BYE TO EXIT
0,1,0
  0.152726427
 Enter Data
 BYE TO EXIT
1,0,0
  0.152726427
 Enter Data
 BYE TO EXIT
```

Andrew Cohen
PHYS 250
11/12/20

As a final test on the Tanh code I added an automatic stopping feature. It works by halting the program if the error begins to increase in value over a certain number of epochs. This resulted in a small but meaningful reduction in error by preventing over-training.

I also attempted to code an XOR 2-layer neural net in FORTRAN. The results are promising, however, there is the occasional error in the results – specifically, the network seems to be confused by the third data point in each entry. I tested the XOR network using entries of two data point (0,1) and it performed as expected.