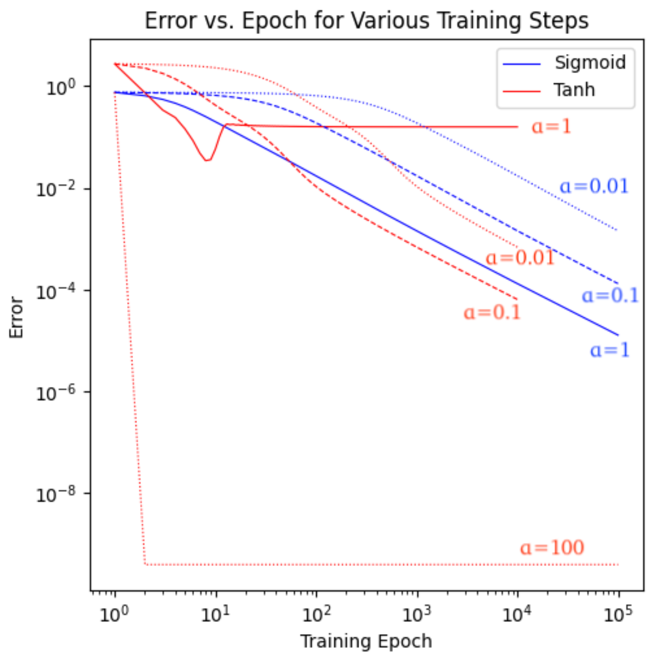
Neural Networks: Perceptron

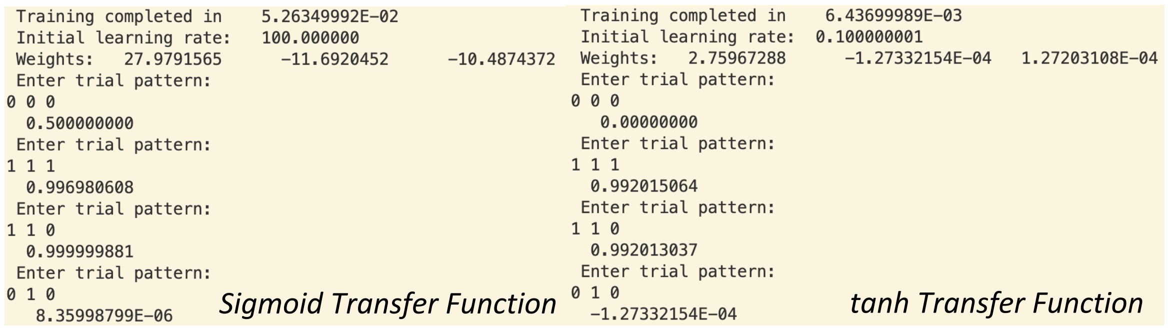
Logan Ge

1 Single Layer Perceptron

We expect the tanh transfer function to work better than the sigmoid function in the training cases we utilize for this project. When inputting the testing case (0,0,0) into our transfer function, the sigmoid function necessarily becomes $f\big(z = (0,0,0)\big) = 0.5$. the tanh transfer function is more suited for our specific pattern given that $\tanh(0) = 0$. However, before reporting the efficacy and accuracy of our perceptron, I optimize the training step with respect to epochs to present an accurate and unbiased comparison between the sigmoid and tanh transfer functions. Below we graph the error as epochs were run



It is very interesting to see that for the $\alpha = 1$ tanh function there's a dramatic decrease in error but rebounds and stabilizes much higher than the lowest value whereas all the other trials demonstrate a strictly decreasing error as epochs were run. There appears to be no clear correlation between step size vs. error optimization other than finding the sweet spot for the gradient descent function. Overall, I find that $\alpha = 100$ is the optimal training step for the sigmoid function while $\alpha = 0.1$ is the optimal training step for the tanh transfer function. I then use these values to test our perceptron



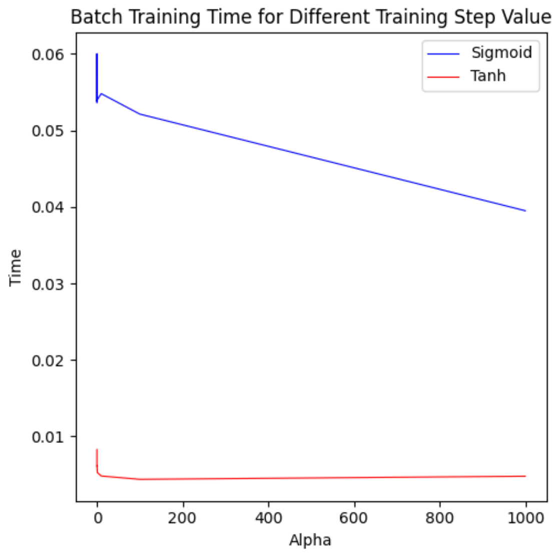
```
 Training completed in    5.26349992E-02
 Initial learning rate:   100.000000
 Weights:   27.9791565      -11.6920452      -10.4874372
 Enter trial pattern:
0 0 0
  0.500000000
 Enter trial pattern:
1 1 1
  0.996980608
 Enter trial pattern:
1 1 0
  0.999999881
 Enter trial pattern:
0 1 0
   8.35998799E-06
```

Sigmoid Transfer Function

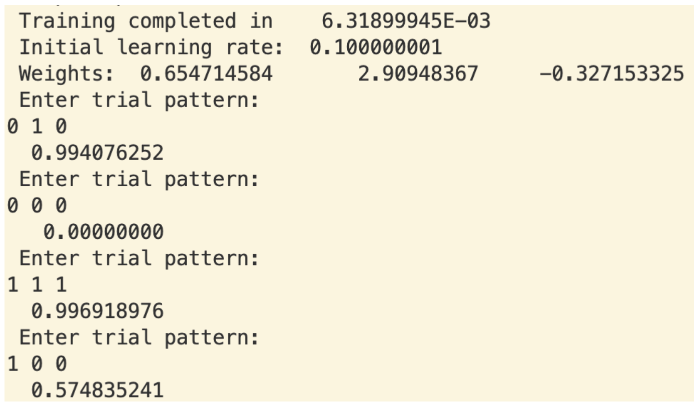
```
 Training completed in    6.43699989E-03
 Initial learning rate:  0.100000001
 Weights:   2.75967288      -1.27332154E-04   1.27203108E-04
 Enter trial pattern:
0 0 0
   0.00000000
 Enter trial pattern:
1 1 1
  0.992015064
 Enter trial pattern:
1 1 0
  0.992013037
 Enter trial pattern:
0 1 0
  -1.27332154E-04
```

tanh Transfer Function

For all the trial's except (0,0,0), the sigmoid function performs marginally better with smaller error from the expected value. However, we see that the (0,0,0) case is very far off for the sigmoid function. We thus find the tanh function to perform better. By using the CALL CPU_TIME function from Fortran, I plotted the time to complete the batch training time for various training step values for the two functions. The tanh transfer function performs significantly faster.



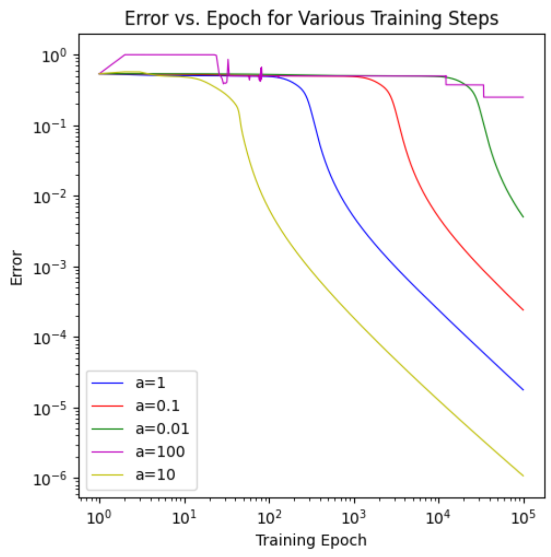
I also tested for different patters where the middle column of the input set reflects the output. The perceptron succeeds in most tests except for 1 case. It is natural to conclude that for different training sets that more optimization is required.

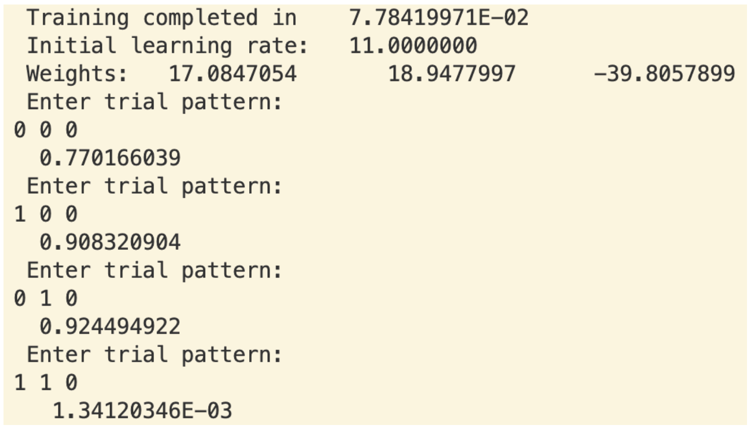


```
 Training completed in    6.31899945E-03
 Initial learning rate:  0.100000001
 Weights:  0.654714584       2.90948367     -0.327153325
 Enter trial pattern:
0 1 0
  0.994076252
 Enter trial pattern:
0 0 0
   0.00000000
 Enter trial pattern:
1 1 1
  0.996918976
 Enter trial pattern:
1 0 0
  0.574835241
```

2 2 Layer Perceptron: XOR

Following the provided link of the Python code for the two-layer perceptron, creating a 2-layer perceptron is simply adding a couple of additional lines. For each epoch, weighting the sum through the transfer function and backpropagation are performed twice (for each layer). What resulted is a decent XOR pattern recognition. As before, I start by optimizing the training step value which I found to be 11. A rough demonstration of the error drop-off at different magnitudes of alpha is demonstrated below.





```
 Training completed in    7.78419971E-02
 Initial learning rate:   11.0000000
 Weights:   17.0847054       18.9477997      -39.8057899
 Enter trial pattern:
0 0 0
  0.770166039
 Enter trial pattern:
1 0 0
  0.908320904
 Enter trial pattern:
0 1 0
  0.924494922
 Enter trial pattern:
1 1 0
   1.34120346E-03
```

No matter what training step size I adjust to, the (0,0,0) continues to prove to be a difficult input for the sigmoid function. However, the other test case seem to respond positively.