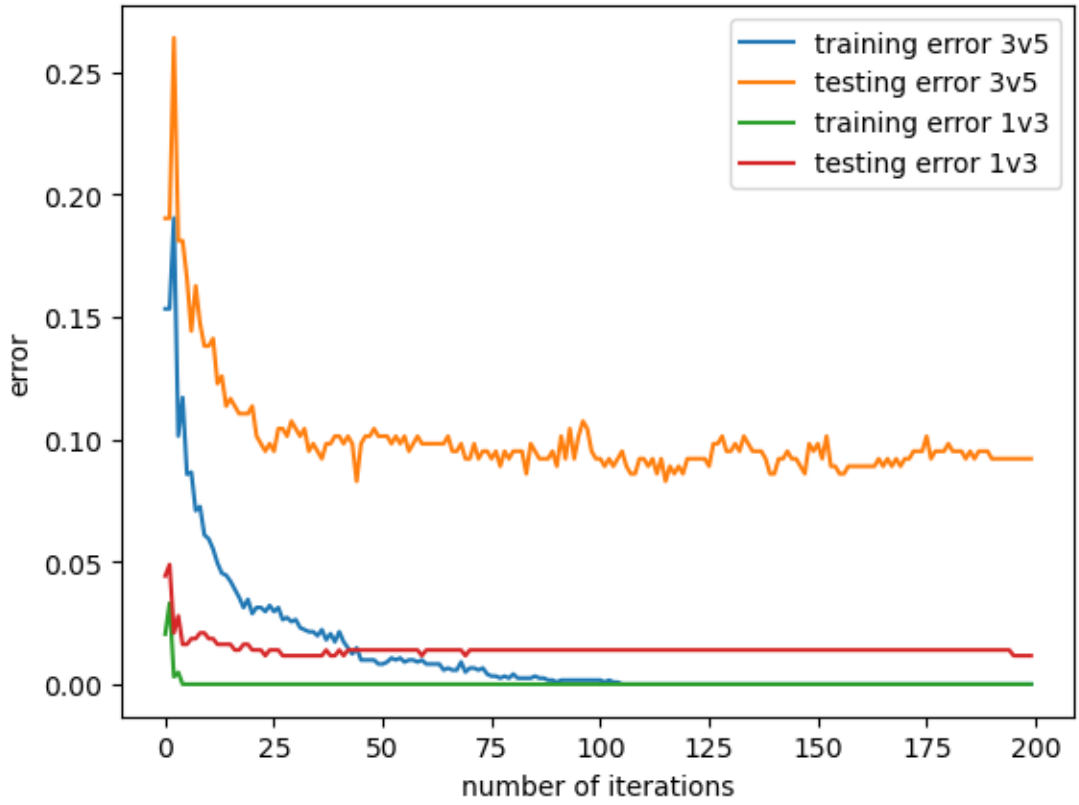
417T Homework 5

Matthew DeSantis

2023-11-24

1

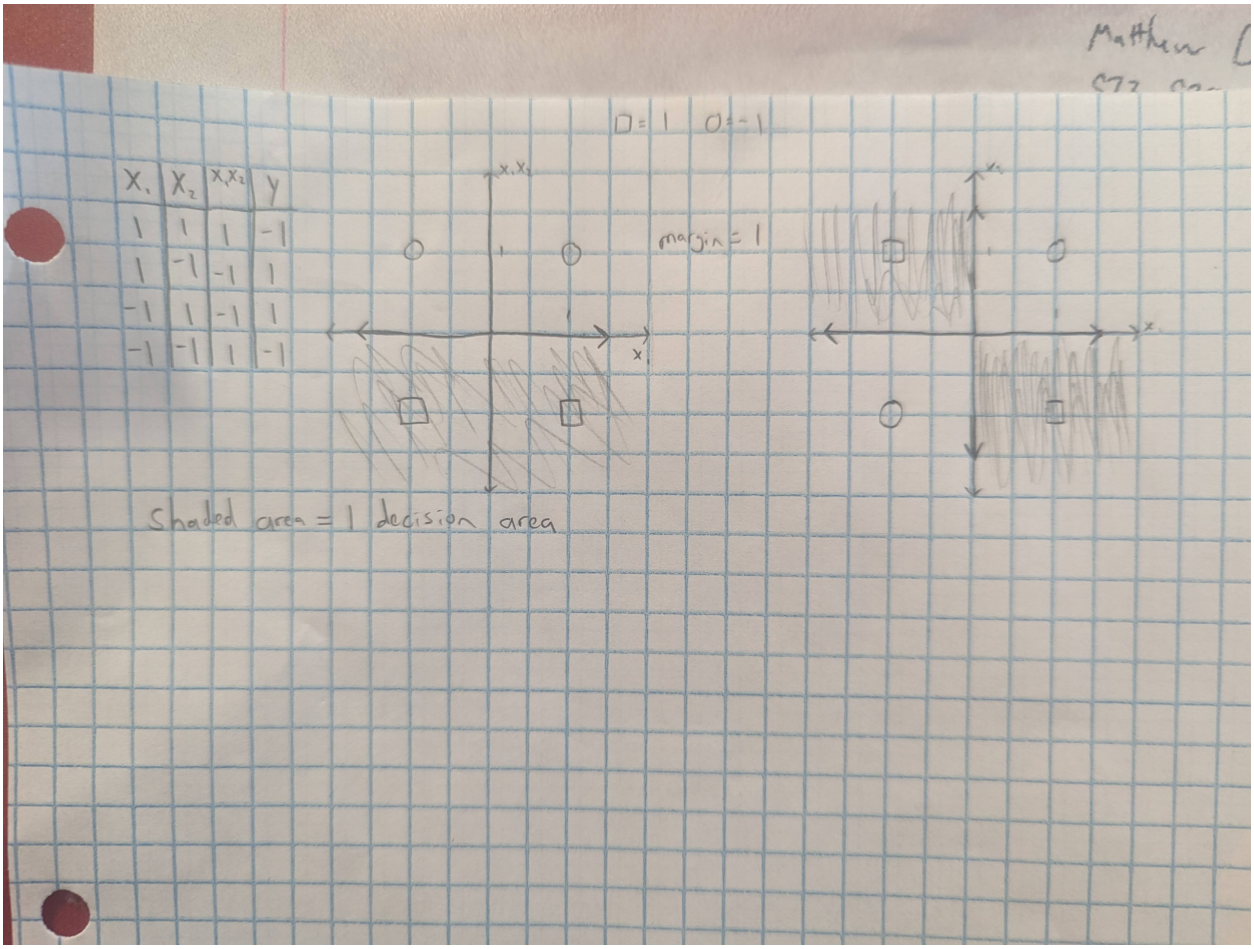


It appears that Adaboost does well, but it is prone to overfitting, especially with the more difficult 3v5 problem. While the training error continues to fall, the testing error more or less stabilizes around 35 iterations for the 3v5 problem and 5 or so for the 1v3 problem. It's important not to take the extremely low testing error at face value, as it isn't representative of how the model will really perform in the wild.

2

8

3



X_1	X_2	X_1X_2	y
1	1	1	-1
1	-1	-1	1
-1	1	-1	1
-1	-1	1	-1

Figure 1: 3

4

$$d^2 = (\phi(\vec{x}_i) - \phi(\vec{x}_j))^2$$

$$d^2 = \phi(\vec{x}_i)^2 - 2\phi(\vec{x}_i)\phi(\vec{x}_j) + \phi(\vec{x}_j)^2$$

$$d^2 = K(\vec{x}_i, \vec{x}_i) - 2K(\vec{x}_i, \vec{x}_j) + K(\vec{x}_j, \vec{x}_j)$$

5

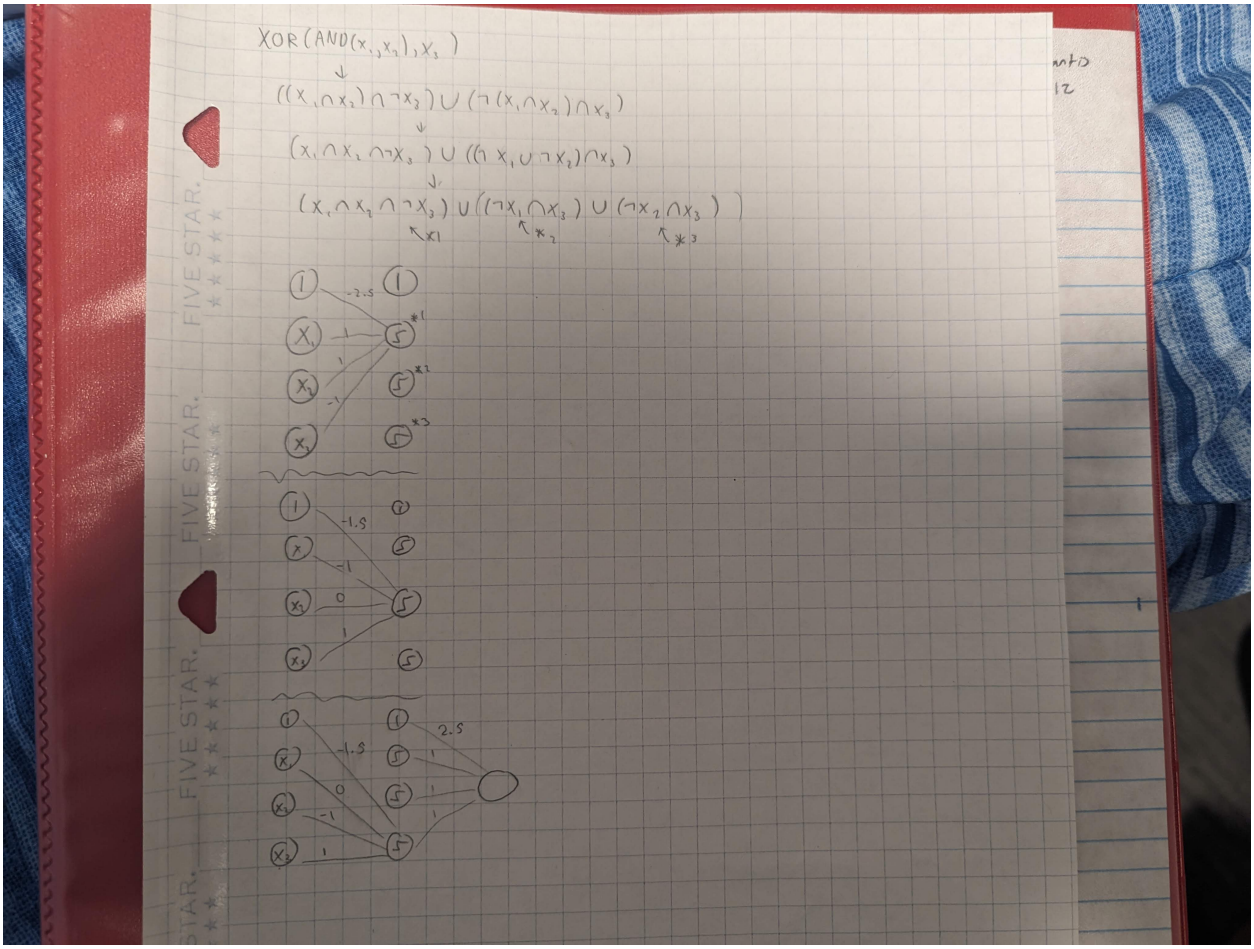

XOR(AND(x1,x2),x3)
((x1∩x2)∩¬x3)∪(¬(x1∩x2)∩x3)
(x1∩x2∩¬x3)∪((x1∪¬x2)∩x3)
(x1∩x2∩¬x3)∪((¬x1∩x3)∪(¬x2∩x3))
*1
*2
*3
-2.5
-1.5
2.5

Figure 2: 5

6

(a)

$$\nabla \frac{1}{N} \sum_{n=1}^{N} (\tanh(\vec{w}^T \vec{x}_n) - y_n)^2$$

$$\frac{2}{N} \sum_{n=1}^{N} (\tanh(\vec{w}^T \vec{x}_n) - y_n) \nabla (\tanh(\vec{w}^T \vec{x}_n) - y_n)$$

$$\frac{2}{N} \sum_{n=1}^{N} (\tanh(\vec{w}^T \vec{x}_n) - y_n) (\nabla \tanh(\vec{w}^T \vec{x}_n) - \nabla y_n)$$

$$\frac{2}{N} \sum_{n=1}^{N} (\tanh(\vec{w}^T \vec{x}_n) - y_n) (1 - \tanh^2(\vec{w}^T \vec{x}_n)) \vec{x}_n$$

$\tanh^2(x)$ approaches 1 as x approaches infinity, so the gradient would approach 0 as the weights approached infinity. With a 0 gradient, no updates would be made, so it's a bad idea to initialize the weights to be very large when using the sigmoid function as your activation function.

(b)

Because $tanh(x)$ and $tanh^2(x)$ both return 0 at 0, in a neural network with at least one hidden layer, the output of each layer (and thus the input of each layer) will be 0, resulting in deadlocked gradients which won't budge.