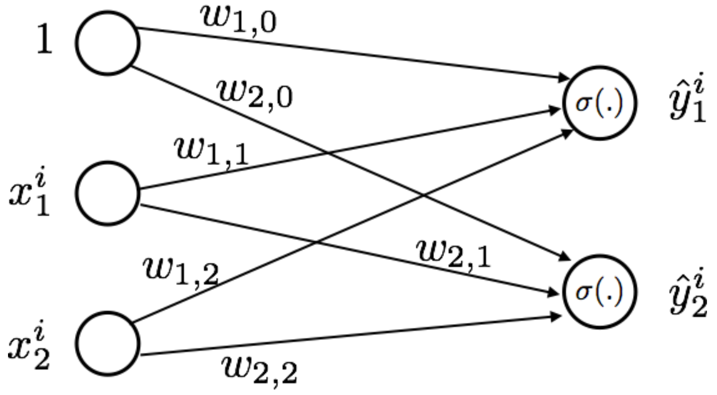
CS/ECE/ME532 Period 22 Activity

Estimated Time: 25 minutes for P1, 25 minutes for P2

1. A script is available to train two neurons using stochastic gradient descent to solve two different classification problems. The two classifier structures are shown below. Here we use a logistic activation function $\sigma(z) = (1 + e^{-z})^{-1}$. The code generates training



data and labels corresponding to two decision boundaries. Case 1 is $x_2^i = -2x_1^i + 0.2$, while case 2 is $x_2^i = 5(x_1^i)^3$.

a) Do you expect that a single neuron will be able to accurately classify data from case 1? Why or why not? Explain the impact of the bias term associated with $w_{1,0}$.

b) Do you expect that a single neuron will be able to accurately classify data from case 2? Why or why not? Explain the impact of the bias term associated with $w_{2,0}$.

c) Run SGD for one epoch. This means you cycle through all the training data one time, in random order. Repeat this five times and find the average number of errors in cases 1 and 2.

d) Run SGD over twenty epochs. This means you cycle through all the training data twenty times, in random order. Repeat this five times and find the average number of errors in cases 1 and 2.

e) Explain the differences in classification performance for the two cases that result with both one and twenty epochs.

SOLUTION:

a) Yes, a single neuron computes the decision boundary as a linear combination of the input. The nonlinear activation function does not change the shape of the decision boundary for a single neuron. The bias term $w_{1,0}$ enables the decision boundary to not include the origin.

b) No, a single neuron computes the decision boundary as a linear combination of the input, so the boundary will be a line and not approximate a cubic. The bias term shifts the boundary away from the origin, but that doesn't help with approximating the cubic decision boundary.

c) For case 1, my average error count was 168. For case 2, my average error count was 730.

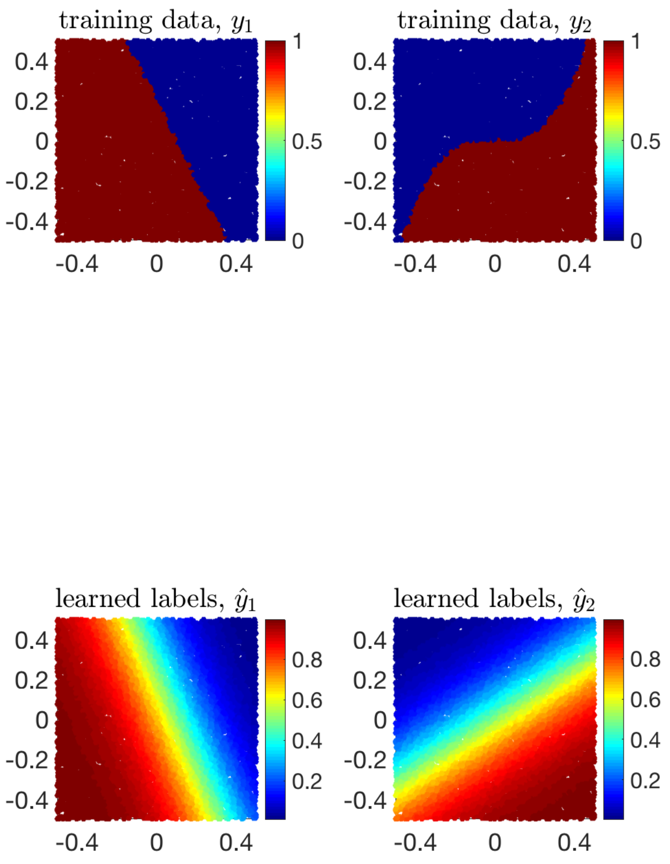


Figure 1: Training data and learned labels for case 1 and 2 classifiers trained using 1 epoch of SGD.

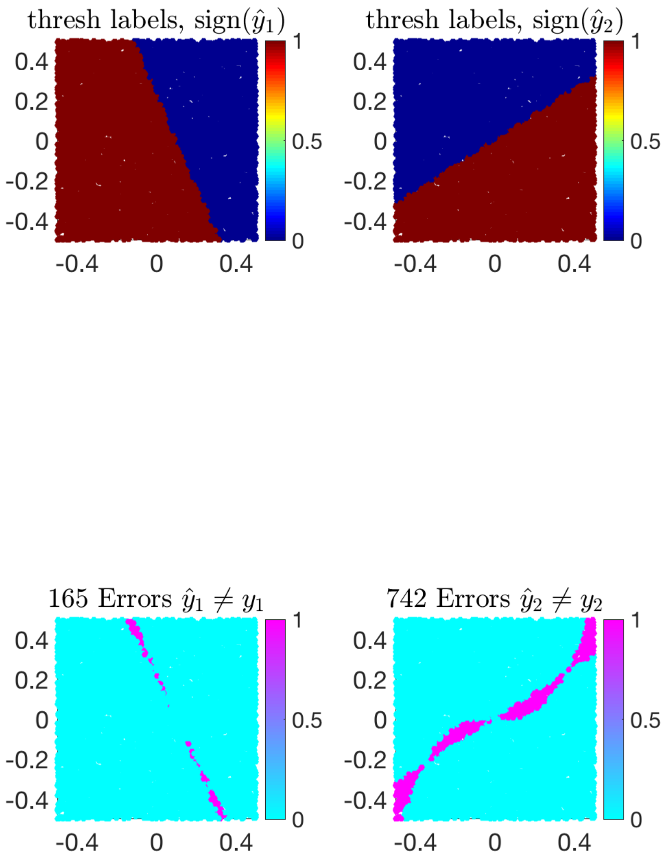


Figure 2: Thresholded labels and errors for case 1 and 2 classifiers trained using 1 epoch of SGD.

d) For case 1, my average error count was 31. For case 2, my average error count was 766.

e) Increasing from one to 20 epochs of SGD iterations produces a significant (factor of five) improvement in classification performance for case 1, but no improvement for case 2. The additional iterations help the case 1 classifier to better learn the decision boundary. In case 2 the classifier capabilities are poorly matched to the true decision boundary and additional training does not improve performance.

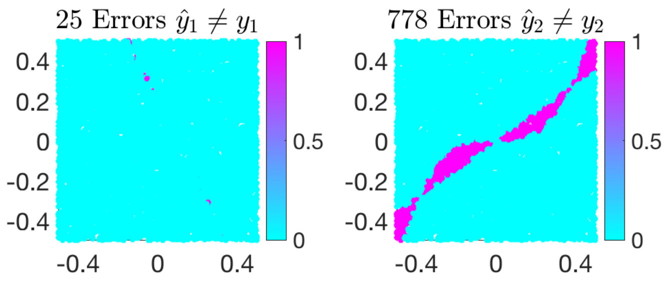
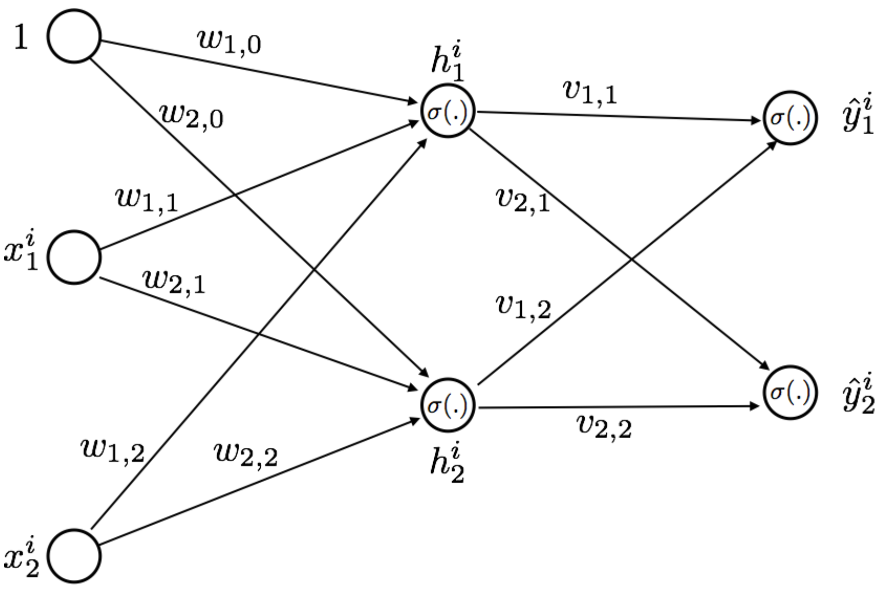


Figure 3: Errors for case 1 and 2 classifiers trained using 20 epochs of SGD.

2. This remainder of this activity uses a three-layer neural network with three input nodes and two output nodes to solve two classification problems. We will vary the number of hidden nodes. The figure below depicts the structure when there are two hidden nodes.



A second script is available that generates training data and trains the network using SGD assuming a logistic activation function $\sigma(z) = (1 + e^{-z})^{-1}$.

a) Use $M = 2$ hidden nodes and ten epochs in SGD. Run this four or five times and comment on the performance of the two classifiers and whether it varies from run to run.

b) Repeat $M = 2$ but use 100 epochs in SGD. (You may use fewer epochs if it takes more than a minute or two per run.) Run this several times and comment on the performance of the classifiers and whether it varies from run to run.

c) Recall the two-layer network results from the previous problem. How do the possible decision boundaries change when you add a hidden layer?

d) Now use $M = 3$ hidden nodes and run 100 epochs of SGD (or as many as you can compute). Does going from two to three hidden nodes affect classifier performance?

e) Repeat the previous part for $M = 4$ hidden nodes and comment on classifier performance.

SOLUTION:

a) The circular classifier (case 1) varies significantly from run to run, and generally does not approximate the actual boundary. The cubic classifier (case 2) is pretty consistent but is approximating the true cubic boundary with something that looks like a line.

b) There is less variability from run to run in case 1, but the boundary still does not approximate the actual boundary. Case 2 is consistent with a linear approximation to the true cubic decision boundary. Using more epochs leads to better convergence of SGD, which reduces the variability.

c) Adding a hidden layer allows the network to represent decision boundaries that are more complicated than linear.

d) Adding one hidden node gives the network much more flexibility in approximating the decision boundaries. It couldn't seem to generate a circular boundary with $M = 2$, but can get much closer with $M = 3$ hidden nodes. Similarly, it is better at approximating a cubic decision boundary, although this varies some from run to run.

e) Using 4 hidden nodes gives the most flexibility and the network does a much better job in approximating both a circular and cubic decision boundary. Unfortunately the number of hidden nodes is a hyperparameter that is difficult to know a priori.

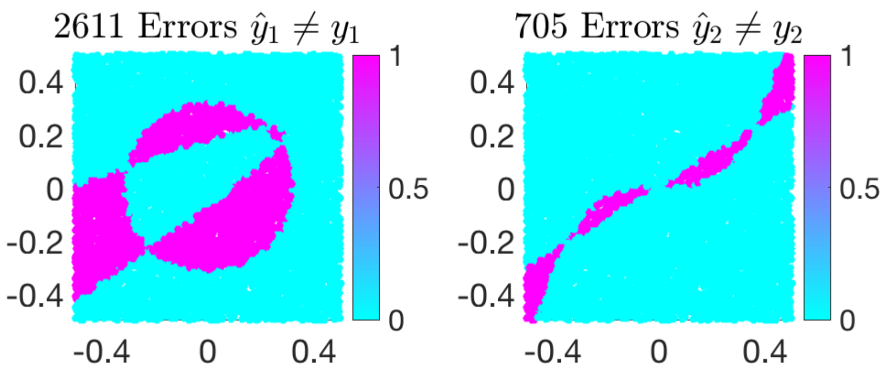


Figure 4: Two hidden nodes, 10 epochs.

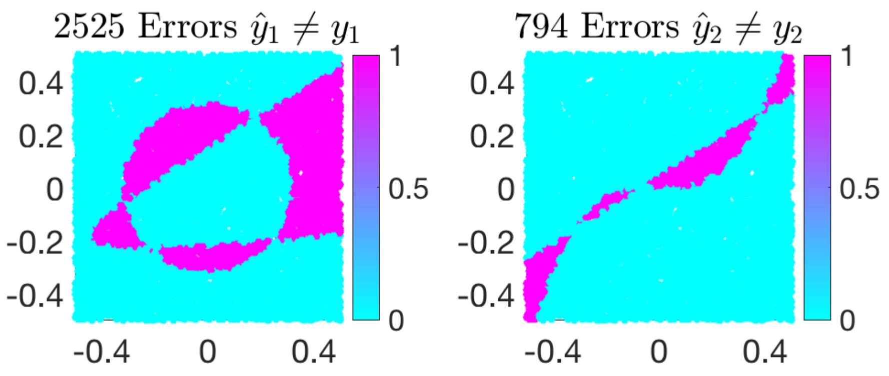


Figure 5: Two hidden nodes, 100 epochs.

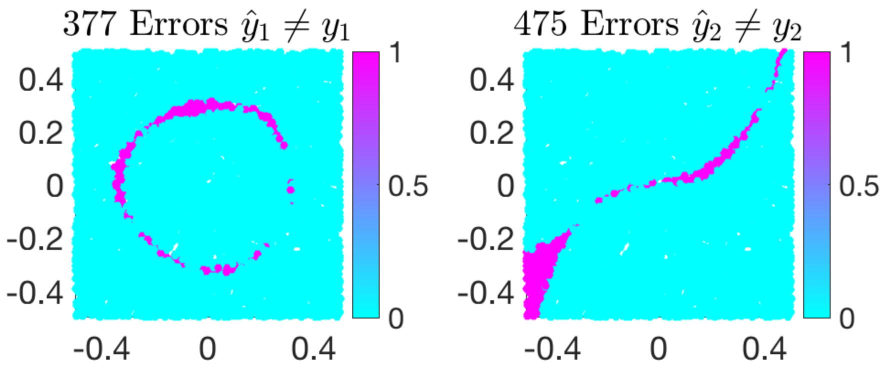


Figure 6: Three hidden nodes, 100 epochs.

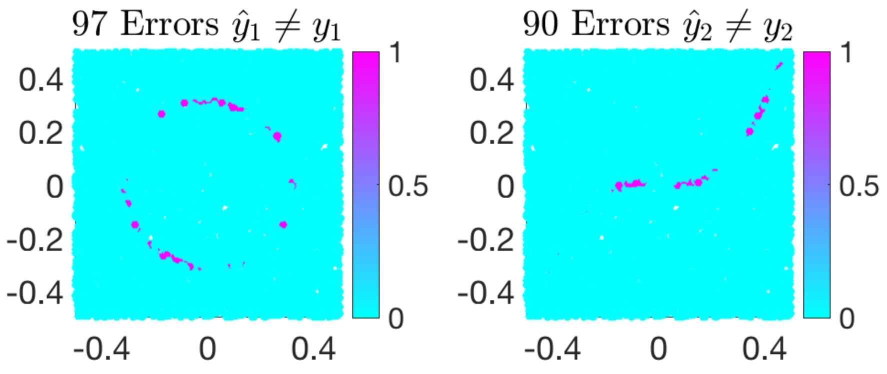


Figure 7: Four hidden nodes, 100 epochs.