Inf2B Coursework Report 2

s1813674

14/04/2020

—

Inf2B - Learning

—

Hiroshi Shimodaira

Task 2 – Neural Networks

Task 2.3

Our task is to find the structure and the weights of the neural network that classifies the inside of Polygon_A as Class 1 and the outside and periphery as Class 0. I was provided with the following Polygon:

Polygon_A: 1.30846 2.38478 1.3544 1.6907 2.38396 2.01798 1.96309 2.60261

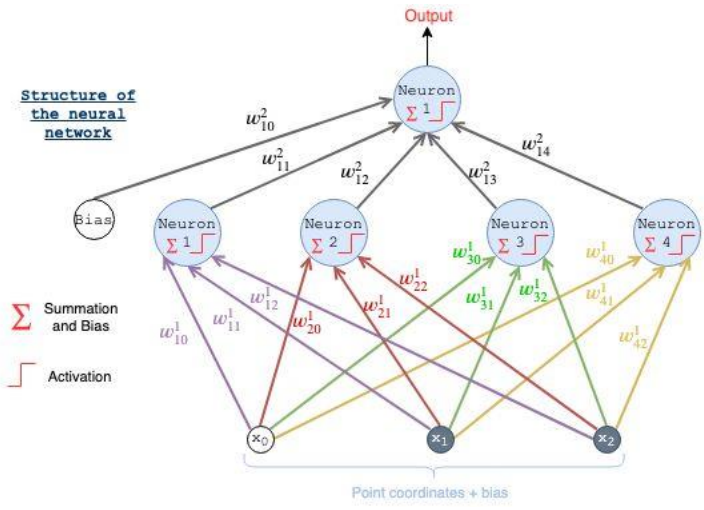
To begin, I found the decision boundaries of our neural network. Since we want the points to be classified either as inside the polygon or as outside, the decision boundaries are the lines connecting the pairs of points that form the polygon.

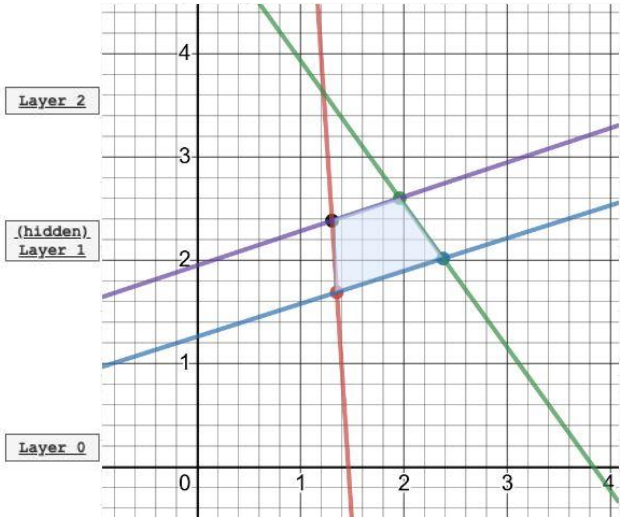
- Point 1 with Point 2 as decision boundary 1
- Point 2 with Point 3 as decision boundary 2
- Point 3 with Point 4 as decision boundary 3
- Point 4 with Point 1 as decision boundary 4

Let Point1 = (1.30846, 2.38478), Point2 = (1.3544, 1.6907), Point3 = (2.38396, 2.01798) and Point4 = (1.96309, 2.60261). We find the slope m of each line using $m = \frac{\Delta y}{\Delta x}$ and then we found the constant b in y = mx + b by plugging the coordinates of a point that lies on the line. This gives us the following equations:

- Decision boundary 1: $y = -15.10840226 * x + 22.15352002$
- Decision boundary 2: $y = \frac{8182}{25739} * x + 1.260158767$
- Decision boundary 3: $y = -1.389098772 * x + 5.329535908$
- Decision boundary 4: $y = \frac{7261}{21821} * x + 1.949386202$

Once we have that, we need to determine the structure of our neural network. Because we concluded that there were 4 decision boundaries, our neural network should look like this:





From this, we compute the weights for each neuron of the hidden layer (layer 1): (we follow lecture 11 notation and coursework notation for weights)

- For neuron 1(layer1), the decision boundary is the red line (decision boundary 1) so our equation is

$$x_2 > -15.10840226 * x_1 + 22.15352002$$
$$a(x) = -22.1535200 + 15.10840226 * x_1 + x_2$$

Since the weights are the coefficient, we get $w_{10}^1 = -22.15352002, w_{11}^1 = 15.10840226 \; and \; w_{12}^1 = 1$. We then need to normalize such that $max_i|w_{ji}^l| = 1$, so our new weights are: get $w_{10}^1 = -1, w_{11}^1 = 0.6819865306 \; and \; w_{12}^1 = 0.0451395534$

- The same way, neuron 2 is the blue line (decision boundary 2) so our equation is

$$x_2 > \frac{8182}{25739} * x_1 + 1.260158767$$
$$a(x) = -1.260158767 - \frac{8182}{25739} * x_1 + x_2$$

Since the weights are the coefficient, we get $w_{20}^1 = -1.260158767, w_{21}^1 = -\frac{8182}{25739} \; and \; w_{22}^1 = 1$. We then need to normalize such that $max_i|w_{ji}^l| = 1$, so our new weights are: get $w_{20}^1 = -1, w_{21}^1 = -0.2522566013 \; and \; w_{22}^1 = 0.7935508018$

- Neuron 3: green line (decision boundary 3)

$$x_2 < -1.389098772 * x_1 + 5.329535908$$
$$a(x) = 5.329535908 - 1.389098772 * x_1 - x_2$$

Since the weights are the coefficient, we get $w_{30}^1 = 5.329535908, w_{31}^1 = -1.389098772 \; and \; w_{32}^1 = -1$. We then need to normalize such that $max_i|w_{ji}^l| = 1$, so our new weights are: get $w_{30}^1 = 1, w_{31}^1 = -0.2606416011 \; and \; w_{32}^1 = -0.1876335984$

- Neuron 4: purple line (decision boundary 4)

$$x_2 < \frac{7261}{21821} * x_1 + 1.949386202$$
$$a(x) = 1.949386202 + \frac{7261}{21821} * x_1 - x_2$$

Since the weights are the coefficient, we get $w_{40}^1 = 1.949386202, w_{41}^1 = \frac{7261}{21821} \; and \; w_{42}^1 = -1$. We then need to normalize such that $max_i|w_{ji}^l| = 1$, so our new weights are: get $w_{40}^1 = 1, w_{41}^1 = 0.1706962183 \; and \; w_{42}^1 = -0.5129819832$

Now that we have weights for the hidden layer, we need to figure out the weights for the last layer which is our output neuron. For that, we will name z_1, z_2, z_3 *and* z_4 the outputs of Neuron 1, 2, 3 and 4 respectively after going through the step function.

Because we are designing the neural network, we expect these outputs to all be 1 for the input to be within the decision boundaries. The weight of every one of these outputs will be one because all conditions need to be met for the point to belong to the inside of the polygon. We only need to figure out the weight of the constant (also called bias). Because all our hidden layer neurons are supposed to give the right classification, we expect the output S of neuron1 in the second layer to be 1 as well. So:

$$S = z_1 + z_2 + z_3 + z_4 + z_0 \; where \; z_0 is \; the \; added \; constant \; (bias)$$
$$1 = 1 + 1 + 1 + 1 + z_0$$
$$z_0 = -3$$

We now have the weights for the last layer:

$$w_{10}^2 = -3, w_{11}^2 = 1, w_{12}^2 = 1, w_{13}^2 = 1 \; and \; w_{14}^2 = 1$$

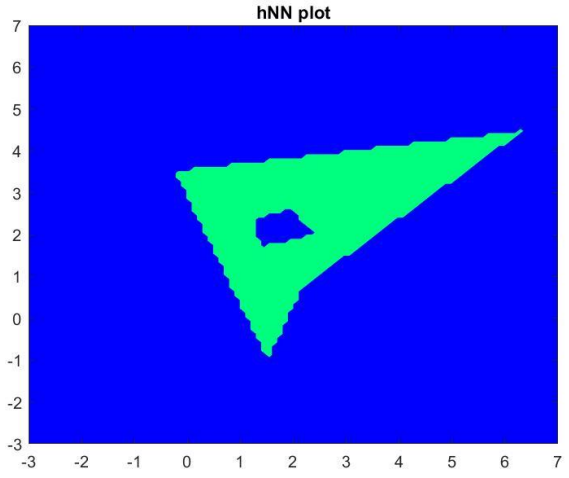
Task 2.10

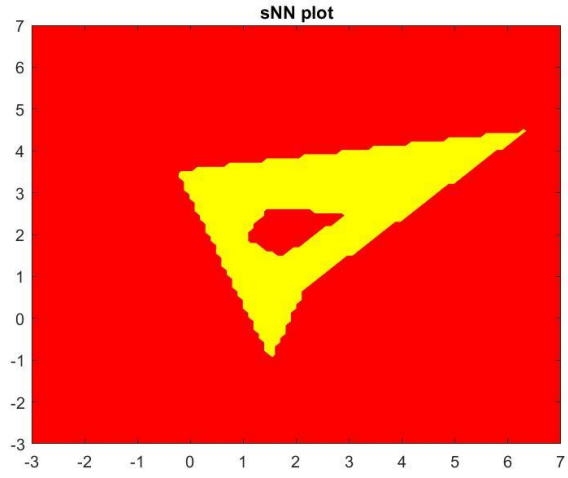
(Note: For all the plotting, I had to generate around 200 points because otherwise it would take way too long)

To get the boundaries for sNN_AB we had to squeeze the sigmoid functions into a step function. For that, because the sigmoid function is $g(a) = \frac{1}{1+\exp(-a)}$, I multiplied the weights by a very large number (10^7) in order to approximate the step function. Now because it is an approximation of the step function, it gives the same result when scaled intensively.

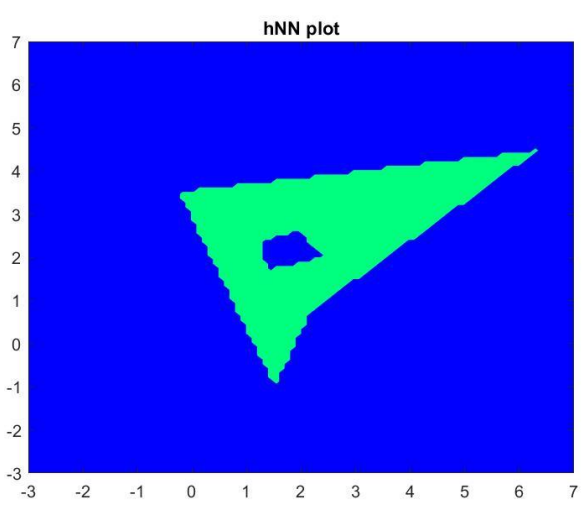
However, the less intensively we scale the weights, the fuzzier the lines for sNN get since it is only approximating the hNN network by adding a threshold (> 0.5 for class 1, <= 0.5 for class 0).

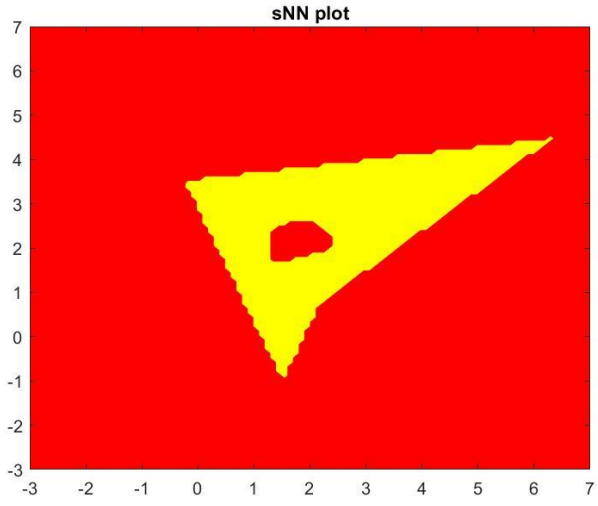
For 10^2: (on the same points)



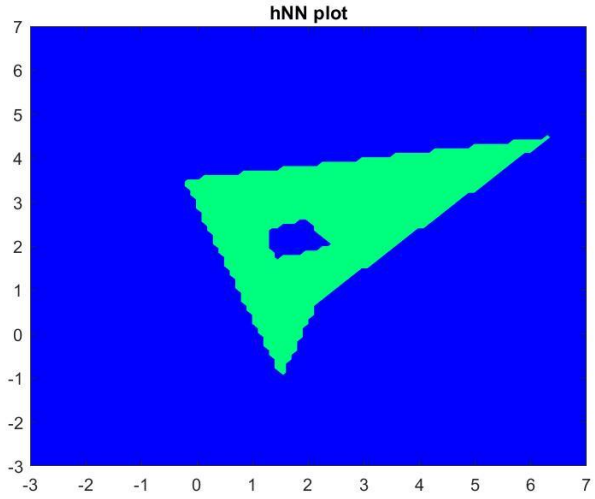


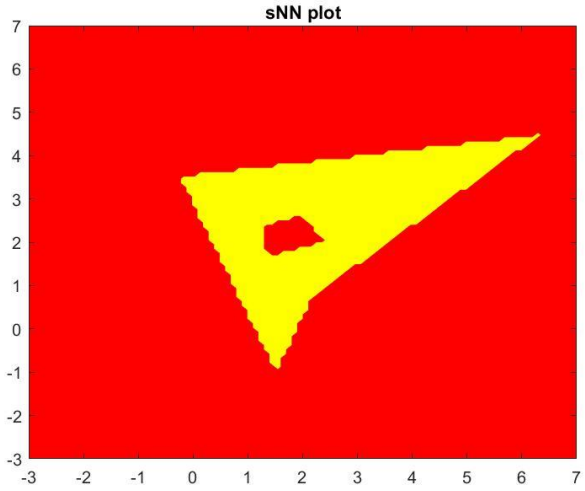
For 10^3: (on the same points)





For 10^4: (on the same points)





Hence, the bigger the multiplier the closest we are to approximating the step functions and getting decision boundaries that are our 2 polygons, this is mostly because we are using logic gates (especially OR gates). The sigmoid function on its own is not enough to get the desired decision boundary because in our case we need to use OR gates that do not work with the sigmoid values since they can give false positives/negatives.