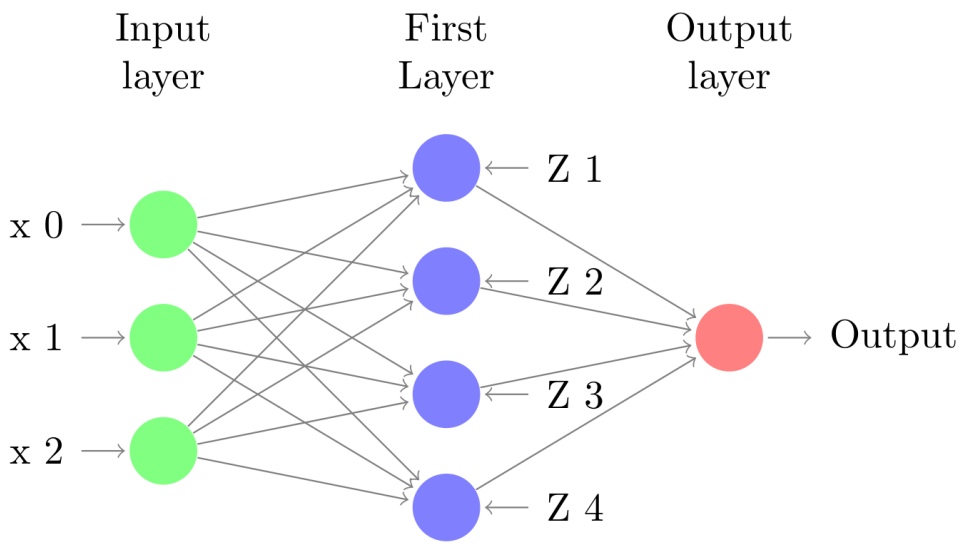
UUN: S1525701

Task 3

Task 3.3

To classify Polygon A I used neural network with the following structure:



For input we have a point (x_1,x_2) and x_0 bias. In our case x_0 is always 1.
Neurons Z1-Z4 in the first layer represent neurons for decision regions given by the 4 lines of our polygon.

Not drawn on the diagram are the weights associated with each layer- W(L,i, j) is the weight of neuron “i” at level L, coming from neuron “j” at level L-1.

Finding the weights

To find the weights I first found the inequalities for the 4 lines of Polygon A.

In order to model them with a step function neuron I presented the inequalities in the form:

$$w_b + w_1x_1 + w_2x_2 \geq 0$$

Using this inequality I take w_b as the bias weight (being multiplied by $x_0 = 1$) and w_1 and w_2 as the corresponding weight for the coordinates of a points (x_1,x_2).

Finally, the weights to combine the output of all 4 neurons represents a simple AND function with 4 inputs – with weights [-3,5, 1, 1, 1, 1].

Task 3.10

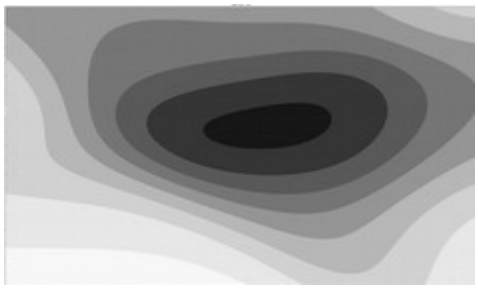
Description

Experiment was conducted by adjusting the weight vectors, which form the described shape for hNN_AB by applying multipliers.

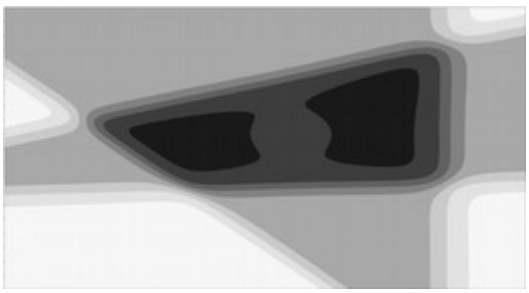
Example: $[w_0, w_1, w_2] \rightarrow [w_0, w_1, w_2]$**multiplier*

We multiply the weights responsible for the decision region of Polygon A by *multiplierA* and the decision region of Polygon B by *multiplierB*.

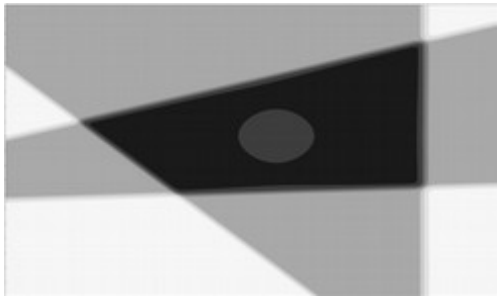
Experiments are done with different values for both multipliers and picture representation of the results can be seen below.



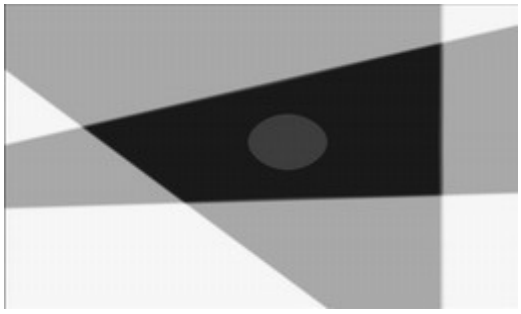
multiplierA=1, multiplierB=1



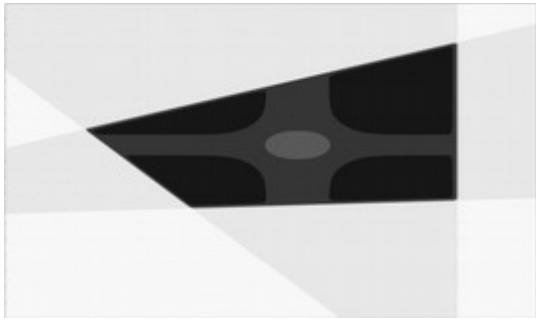
multiplierA=5, multiplierB=1



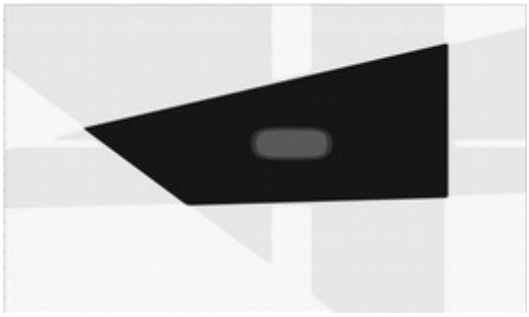
multiplierA=20, multiplierB=1



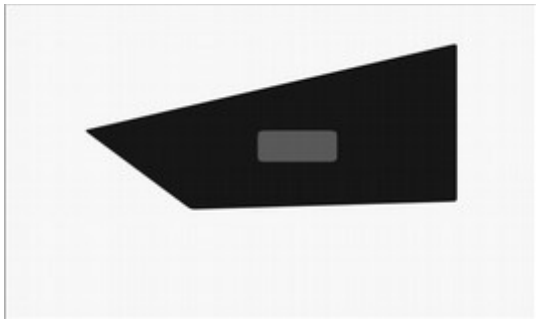
multiplierA=50, multiplierB=1



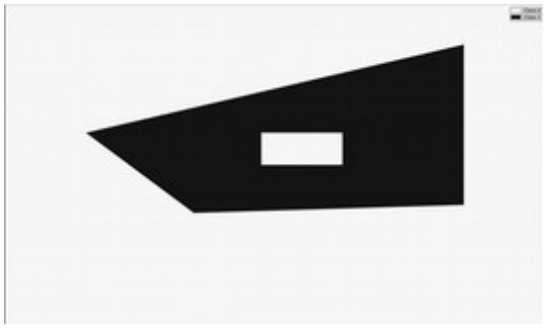
multiplierA=50, multiplierB=5



multiplierA=50, multiplierB=10



multiplierA=50, multiplierB=30



Classification of hNN_AB function

Discussion

We can see that with increase in weights the structure becomes very close to the structure constructed by *hNN_AB* network. With increase of the multipliers the decision boundaries become close to the linear classification of the step function. With lower multipliers we can see that the data is classified by smoother non-linear boundaries. This is due to the nature of the sigmoid function. It has a horizontal asymptotes at 0 and 1, which means that for very small and very big numbers the sigmoid's output approximates the step function's output very closely but the two would never output the same result.

The main difference we can observe in our results is that Polygon B is classified as a different class than the outer region of Polygon A, when we use the sigmoid function.

Our classification problem is linearly separable and contains only 2 classes so the step activation function performs better in this case. However, if we had a multilayer neuron network and we wanted to learn the weights by back propagation that would be impossible with the step function − because it is not differentiable.

Besides the fact that inner polygon and the space out of Polygon A should be the same class, the sigmoid function, with adequately adjusted weights, did a good job at classifying the data.