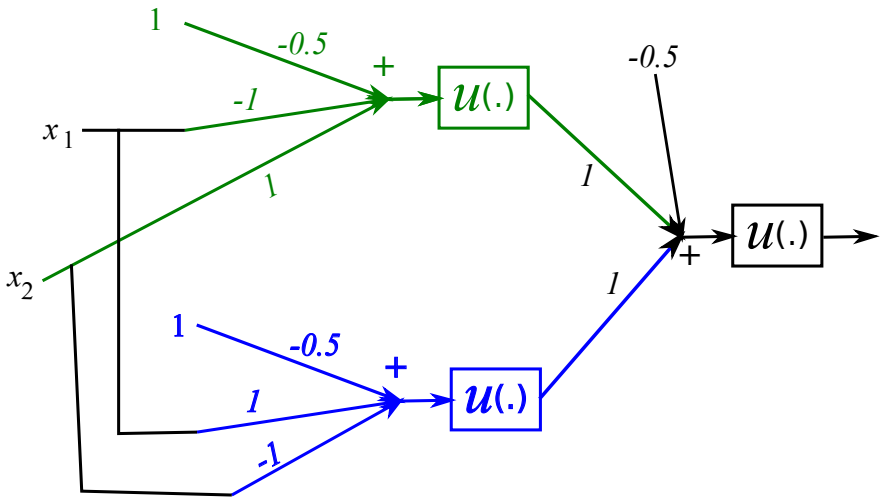
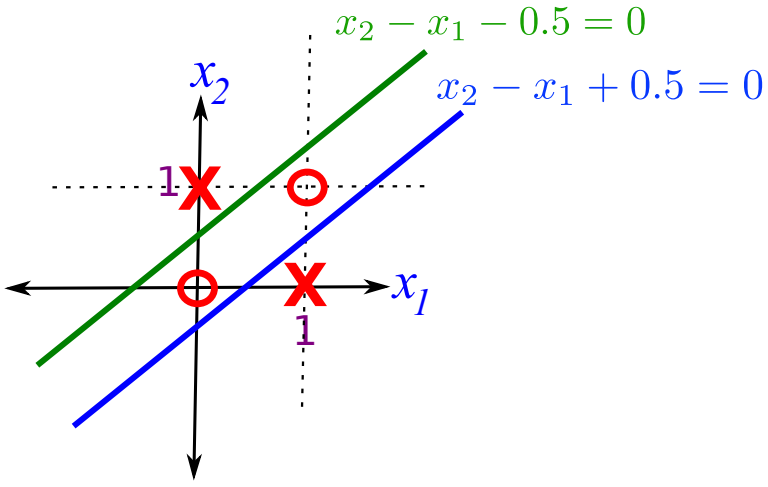
ECE/CS 559 - Neural Networks Lecture Notes #5
Introduction to Multilayer networks

Erdem Koyuncu

- Why use multiple layers?
- Sometimes that are not separable by a single layer may be separated my multiple layers.
- One example was the XOR operation. XOR cannot be done via a single layer (by one perceptron), but we showed a 2-layer network that can do XOR.



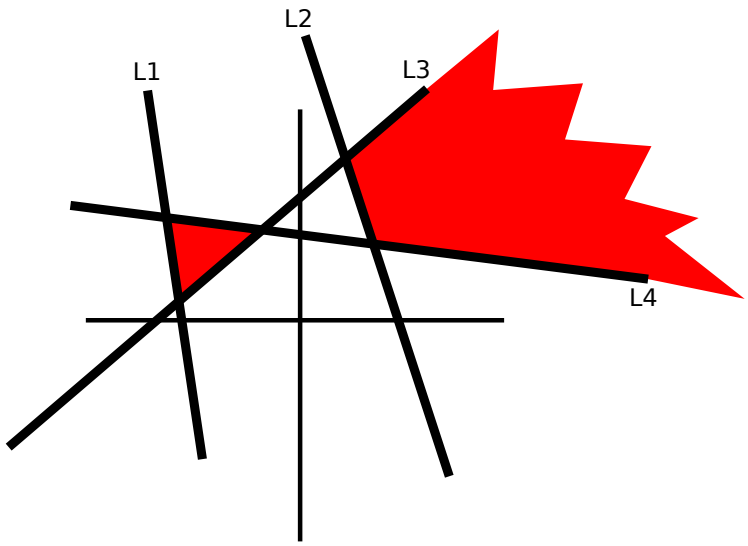
The geometry for this network looks like as follows:



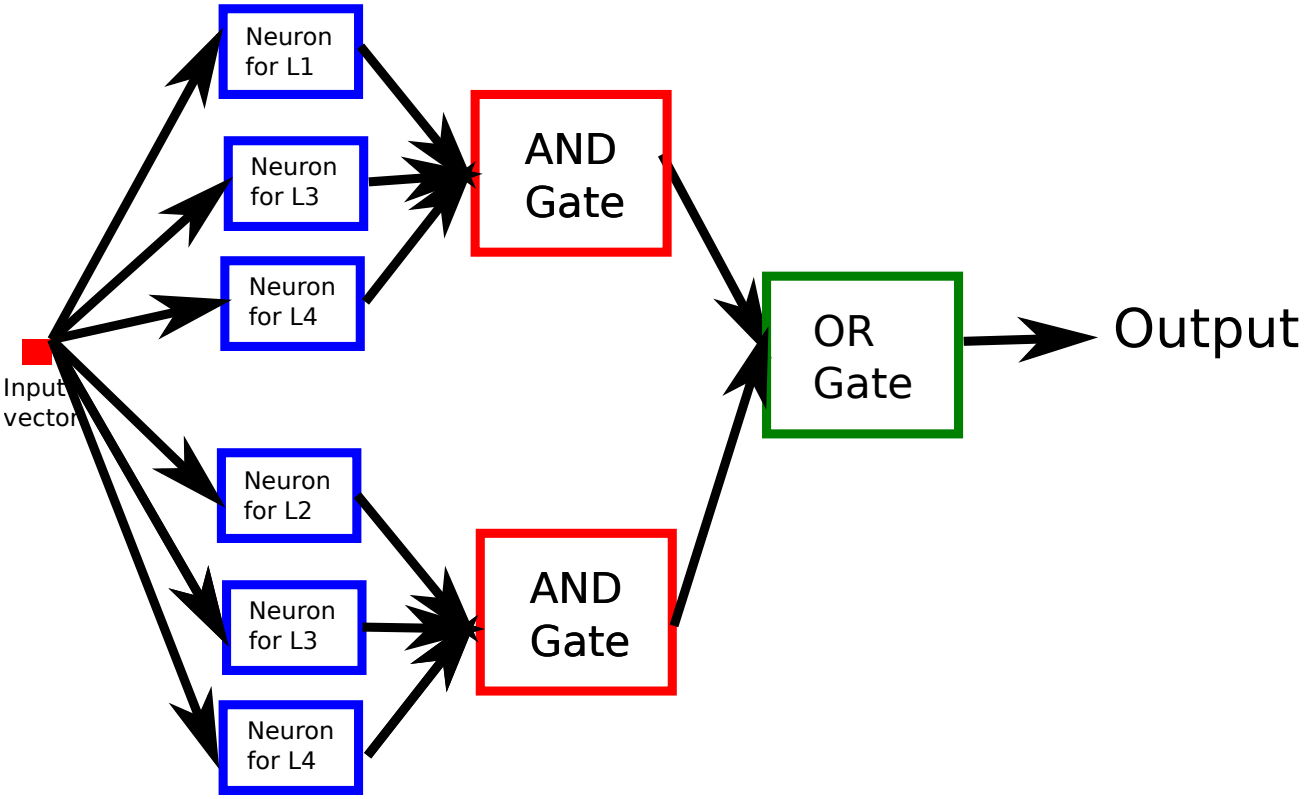
- In the first layer, the green top neuron induces the green separating line. For this neuron, over the green line, the output is 1, and below the green line, the output is 0. This way, we isolate one of the regions where we want an eventual output of 1. On the other hand, in the first layer, the blue

bottom neuron induces the blue separating line. For this neuron, below the blue line, the output is 1, and above the blue line, the output is 0. This way, we isolate the remaining region where we want an eventual output of 1. We want an eventual output of 1 if outputs from any one of the neurons in the first layer equals 1, which is implemented through the OR operation via the neuron in the second layer.

- Consider, in general, the following geometry

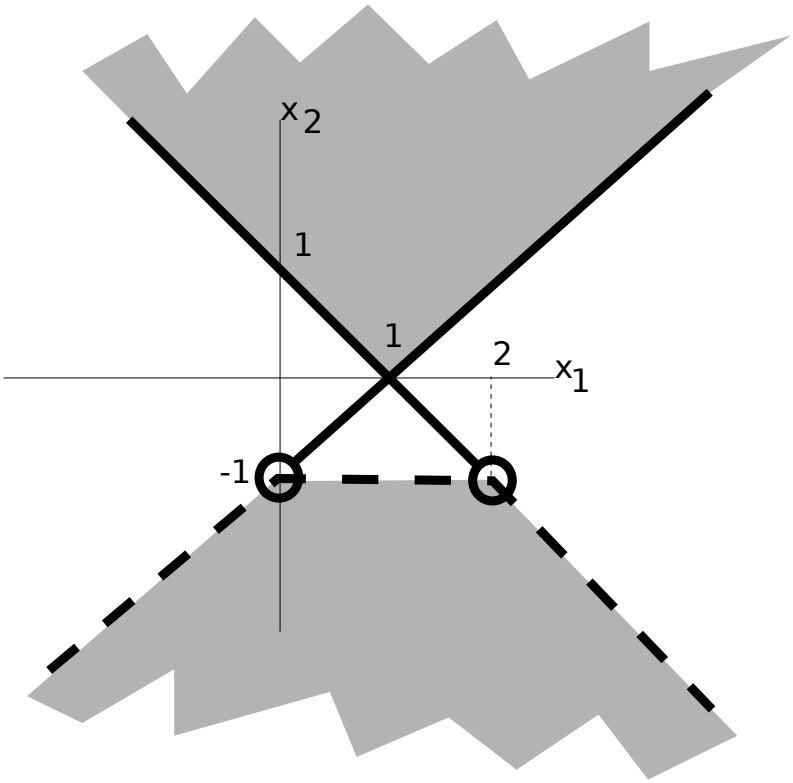


We want an output of 1 if the input vector falls in anywhere inside the red region, otherwise we want an output of 0. One can then consider a 3-layer systematic network to solve these kind of problems:



The first layer is the line formation layer, where we form the lines that separates the different regions. Consider the top three neurons in the first layer. We pick the weights of the L1 neuron such that the output of the neuron is 1 if and only if the input lies to the "right" of the L1 line. We pick the weights of the L3 neuron such that the output of the neuron is 1 if and only if the input lies on the "top" of the L3 line. We pick the weights of the L4 neuron such that the output of the neuron is 1 if and only if the input lies on the "bottom" of the L4 line. This way, the output of the first AND gate in the second layer is 1 if and only if the input lies in the leftmost triangular red region. This handles the first region where we want an output of 1. In a similar vein, we design the remaining weights such that the output of the second AND gate in the second layer is 1 if and only if the input lies in the rightmost unbounded red region. The final OR gate ensures that our network works as intended.

- Hence, we can classify linearly separable patterns using 3 layers of neurons only.
- The boundary conditions (i.e. what happens at the separator lines) may require special care.
- Example: Design a neural network that outputs a 1 in the shaded area and 0 otherwise. Note that all boundary points in the bottom region is excluded.



- Note that the top region is

$$A = \{(x_1, x_2) : x_1 - x_2 - 1 \leq 0 \text{ and } -x_1 - x_2 + 1 \leq 0\} \tag{1}$$

or equivalently (to make them compatible with our $u(\cdot)$ neurons),

$$A = \{(x_1, x_2) : -x_1 + x_2 + 1 \geq 0 \text{ and } x_1 + x_2 - 1 \geq 0\}. \tag{2}$$

The bottom region is

$$B = \{(x_1, x_2) : x_1 - x_2 - 1 > 0 \text{ and } -x_1 - x_2 + 1 > 0 \text{ and } x_2 + 1 < 0\} \tag{3}$$

Now due to strict inequalities, we cannot immediately make these compatible with our neurons. But, we can use the following trick. The region is equal to

$$B = \{(x_1, x_2) : \text{NOT}(x_1 - x_2 - 1 \leq 0) \text{ and } \text{NOT}(-x_1 - x_2 + 1 \leq 0) \text{ and } \text{NOT}(x_2 + 1 \geq 0)\} \quad (4)$$

and using negation once again, we obtain

$$B = \{(x_1, x_2) : \text{NOT}(-x_1 + x_2 + 1 \geq 0) \text{ and } \text{NOT}(x_1 + x_2 - 1 \geq 0) \text{ and } \text{NOT}(x_2 + 1 \geq 0)\} \quad (5)$$

- From these representations, the neurons in the first layer should provide the outputs:

$$y_1 = u(-x_1 + x_2 + 1) \quad (6)$$
$$y_2 = u(x_2 + x_2 - 1) \quad (7)$$
$$y_3 = u(x_2 + 1). \quad (8)$$

The neurons in the second layer should provide the outputs:

$$z_1 = u(y_1 + y_2 - \frac{3}{2}) \qquad \text{(this is the usual AND gate)} \quad (9)$$
$$z_2 = u(-y_1 - y_2 - y_3 + \frac{1}{2}) \qquad \text{(this gate ANDs the NOTs of all inputs)} \quad (10)$$

The neuron in the final layer is a mere OR:

$$output = u(z_1 + z_2 - \frac{1}{2}) \quad (11)$$

Note that $z_1 = 1$ if and only if the input vectors belong to A, and $z_2 = 1$ if and only if the input vectors belong to B.