Multi-layer perceptron

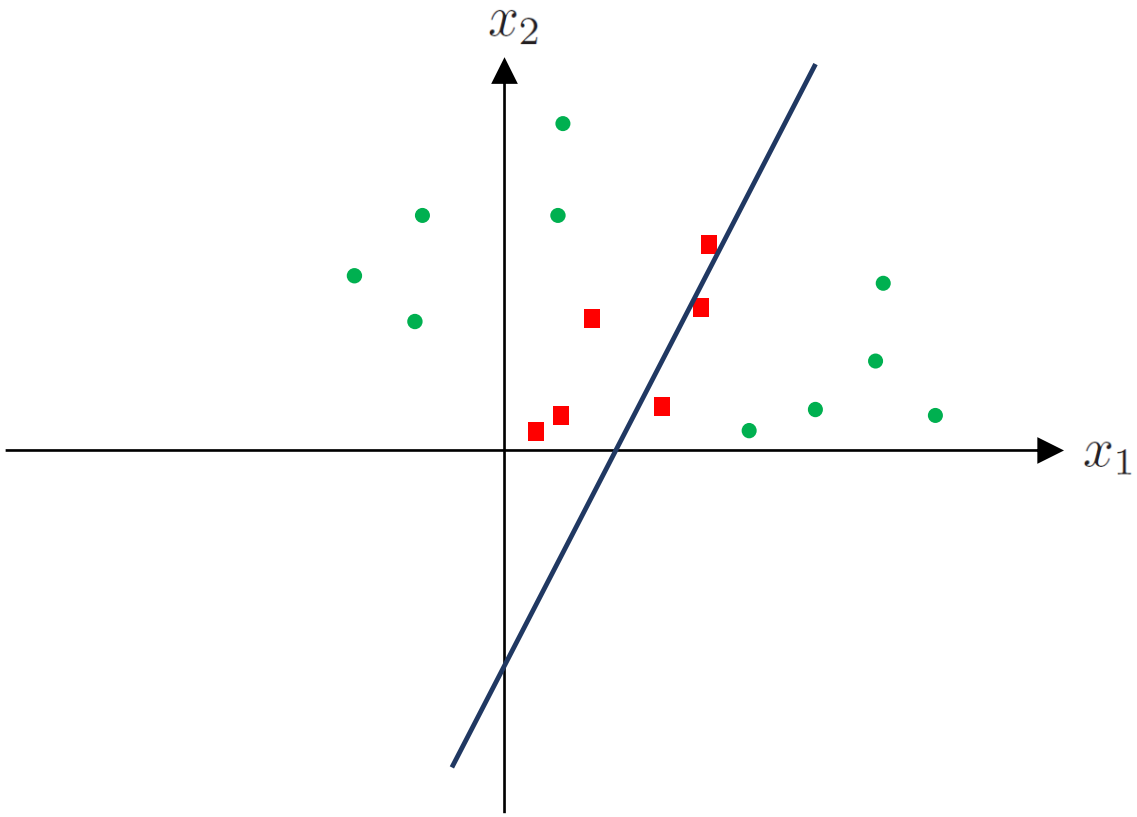
Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India

Machine Learning

Sem 3, 2018-19

Sem 3, 2018-19

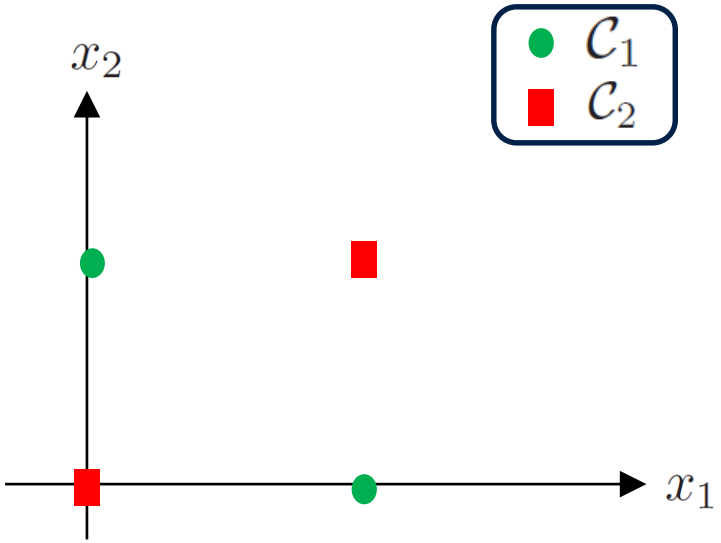
Shortcomings of single layer perceptron Learning



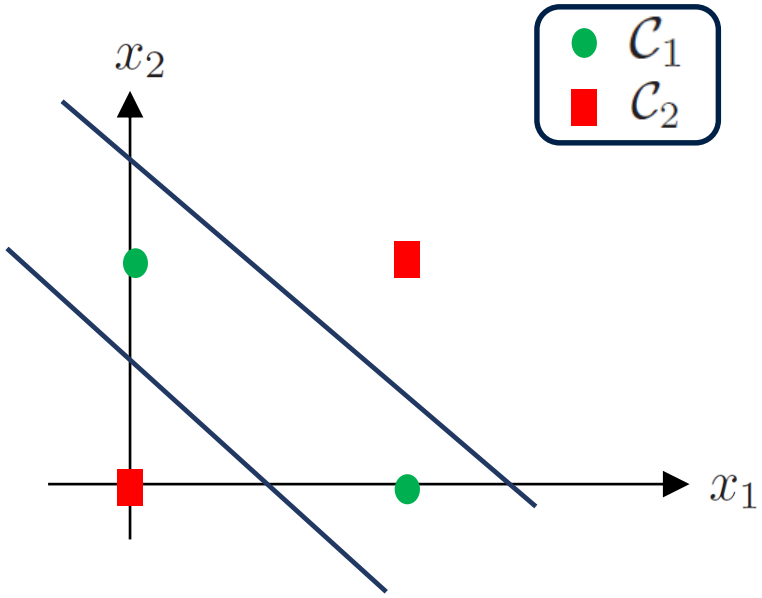
XOR function

- XOR data is not linearly separable.

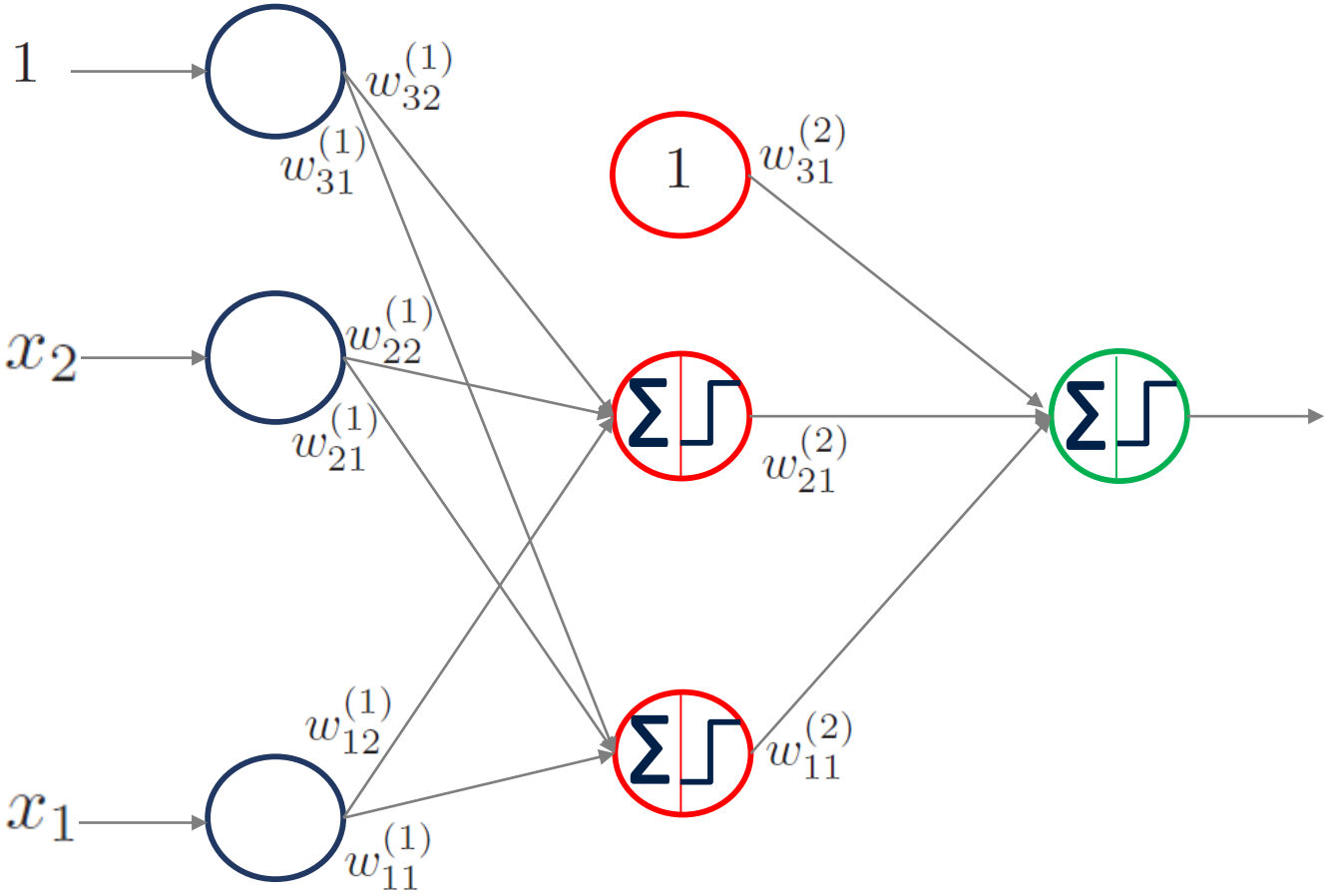
x_1	x_2	XOR	Class label
0	0	0	$\mathcal{C}_2$
0	1	1	$\mathcal{C}_1$
1	0	1	$\mathcal{C}_1$
1	1	0	$\mathcal{C}_2$



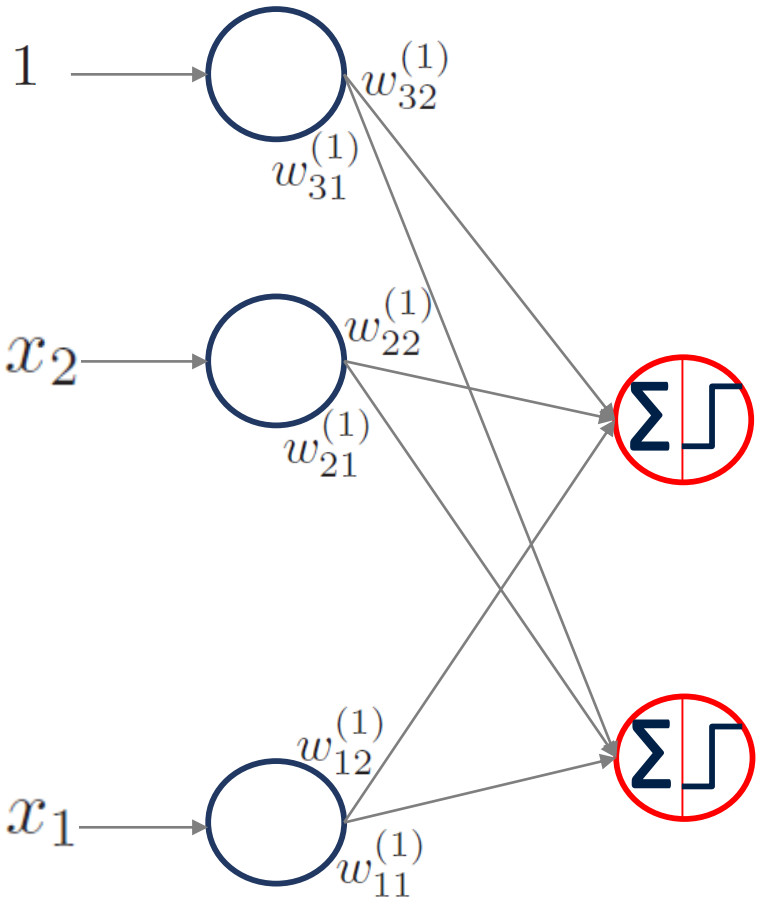
Combination of classifiers



Multi-layer perceptron



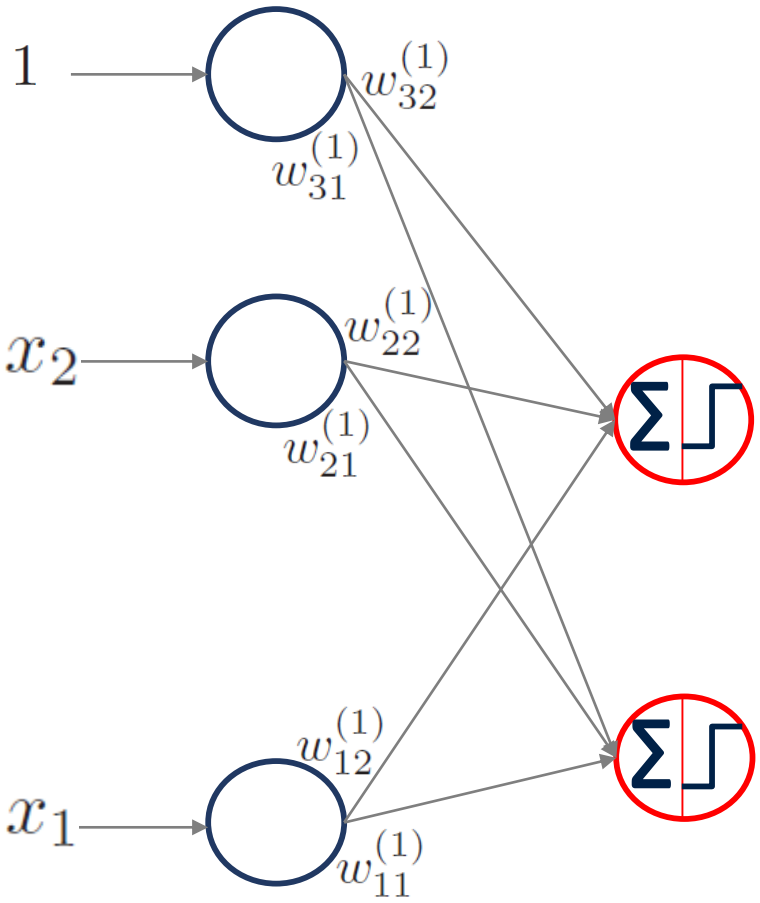
Multi-layer perceptron

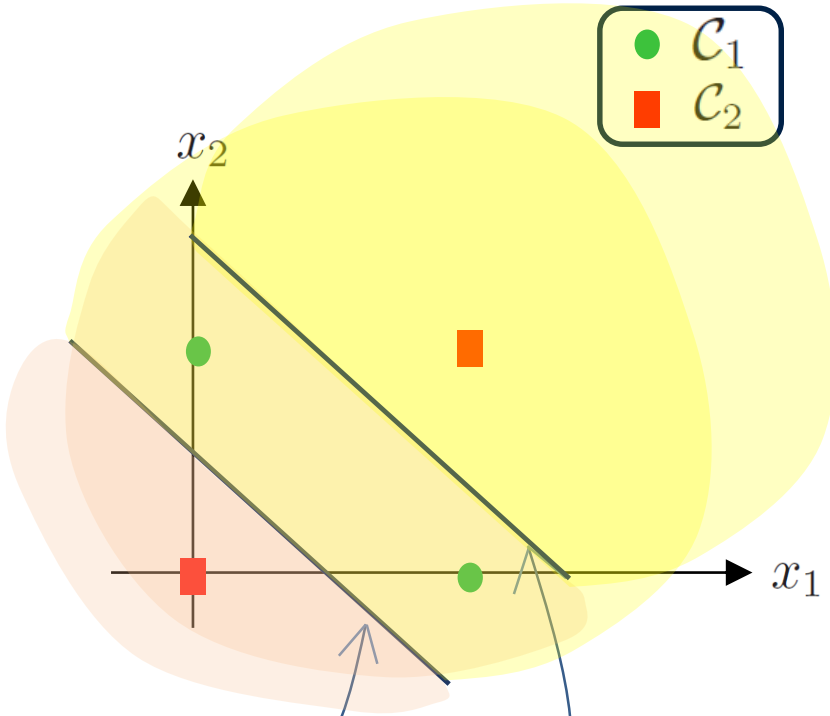


$$z_2 = \mathcal{A}(w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{32}^{(1)})$$

$$z_1 = \mathcal{A}(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)})$$

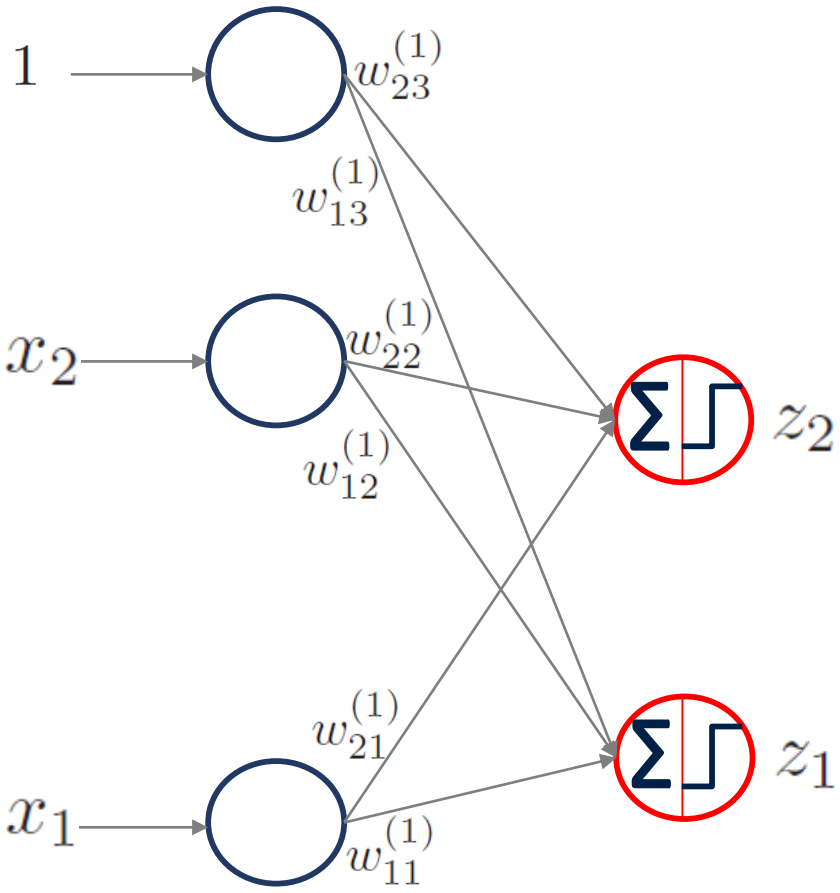
Multi-layer perceptron

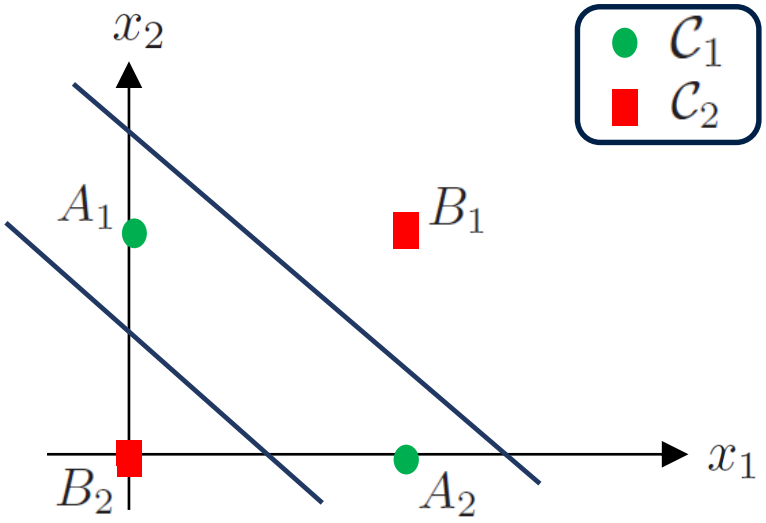


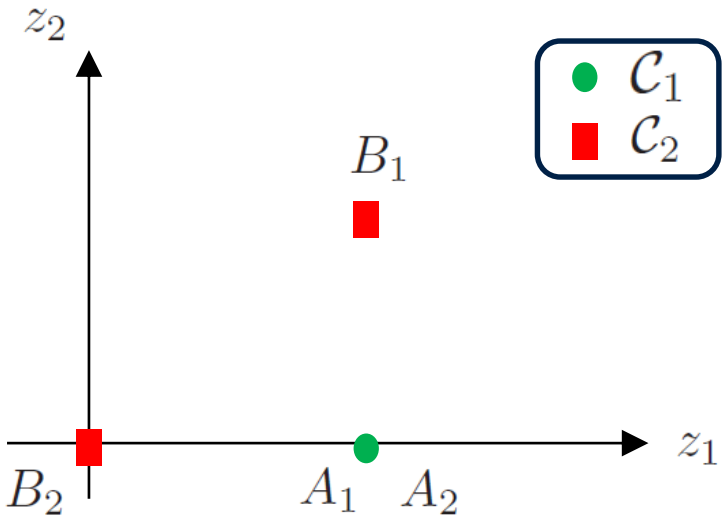


- Take $w_{11}^{(1)} = 1$, $w_{21}^{(1)} = 1$, and $w_{31}^{(1)} = -0.5$.
- Take $w_{12}^{(1)} = 1$, $w_{22}^{(1)} = 1$, and $w_{32}^{(1)} = -1.5$.

Multi-layer perceptron

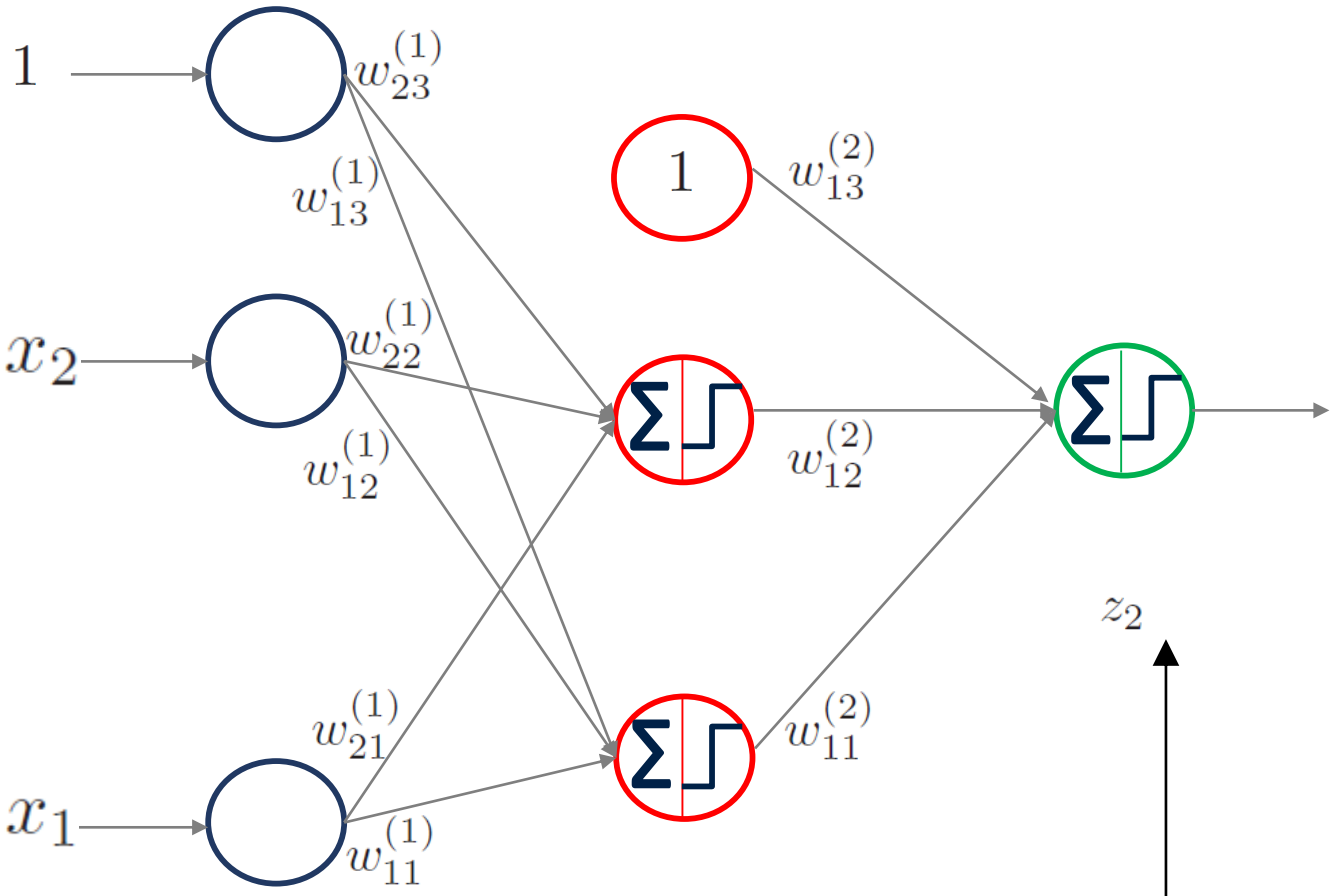


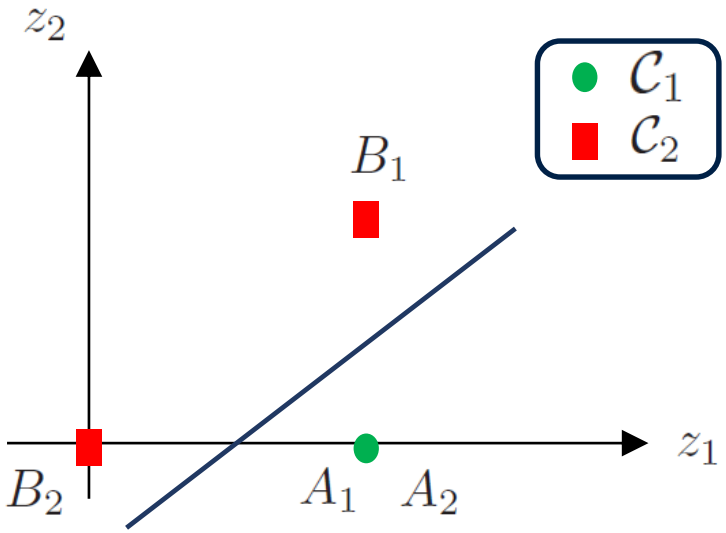




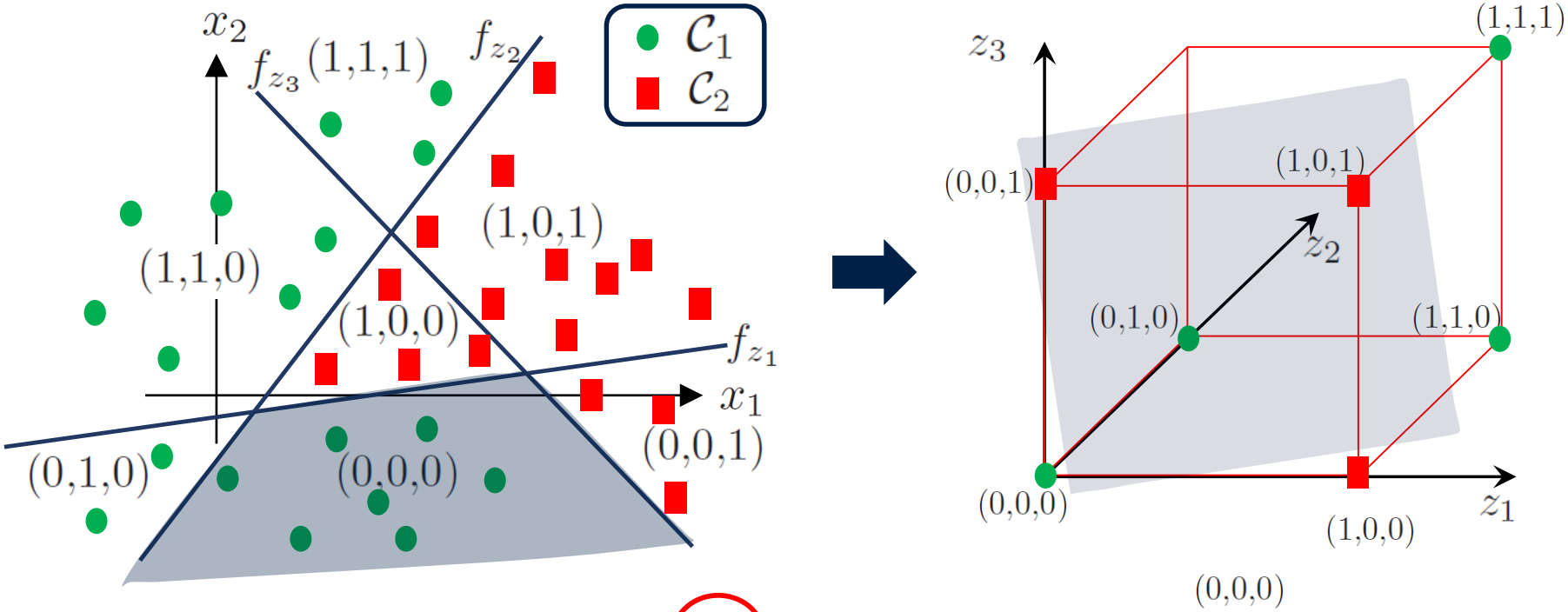
- Now what do you think you need to do?

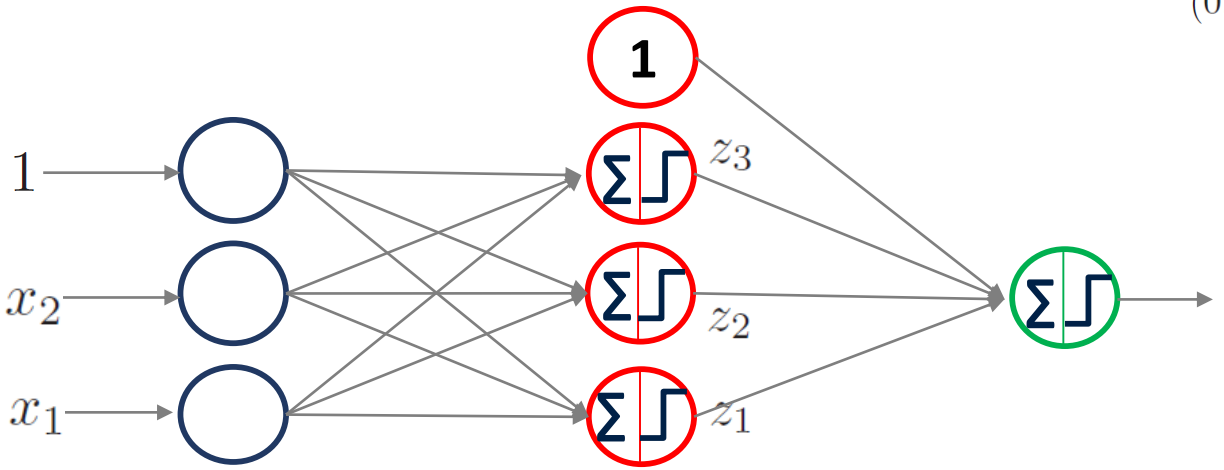
Multi-layer perceptron





Single hidden layer

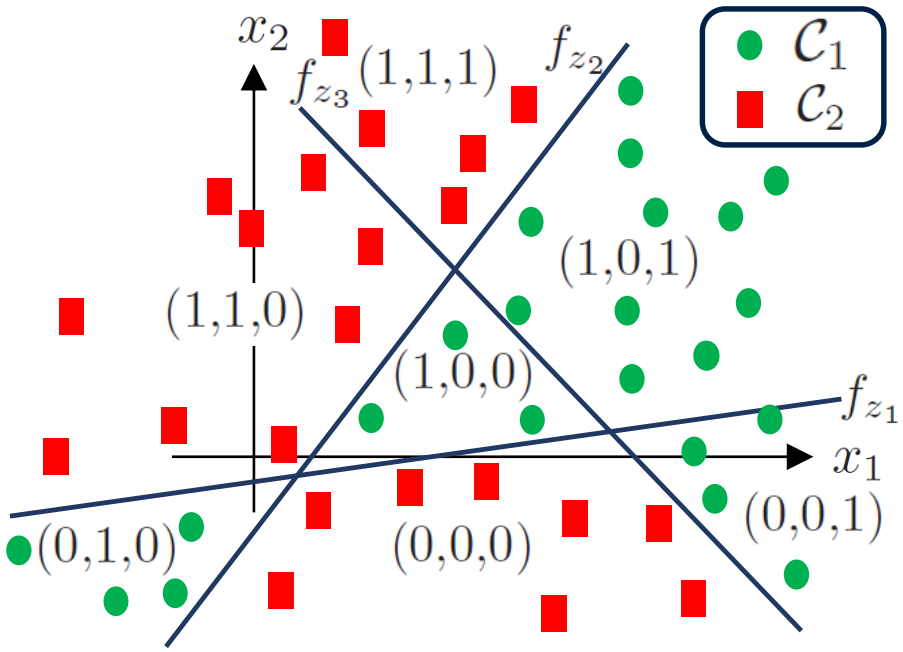


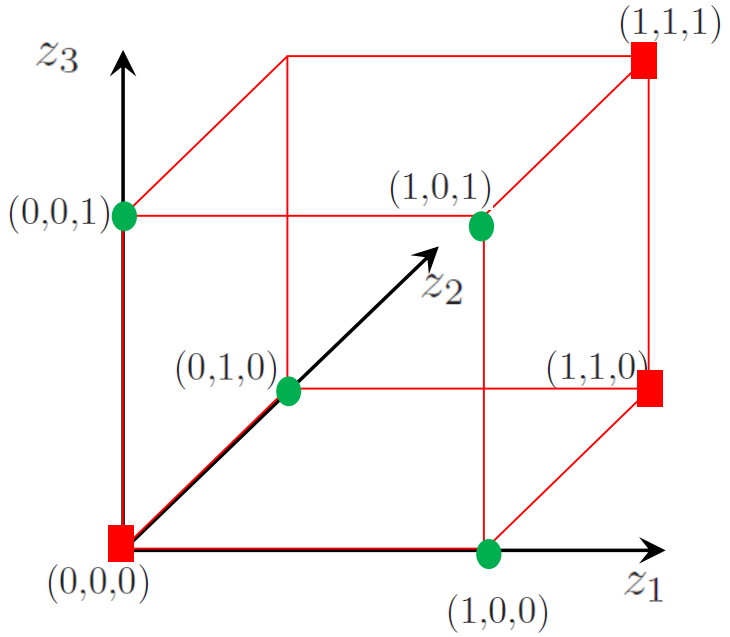


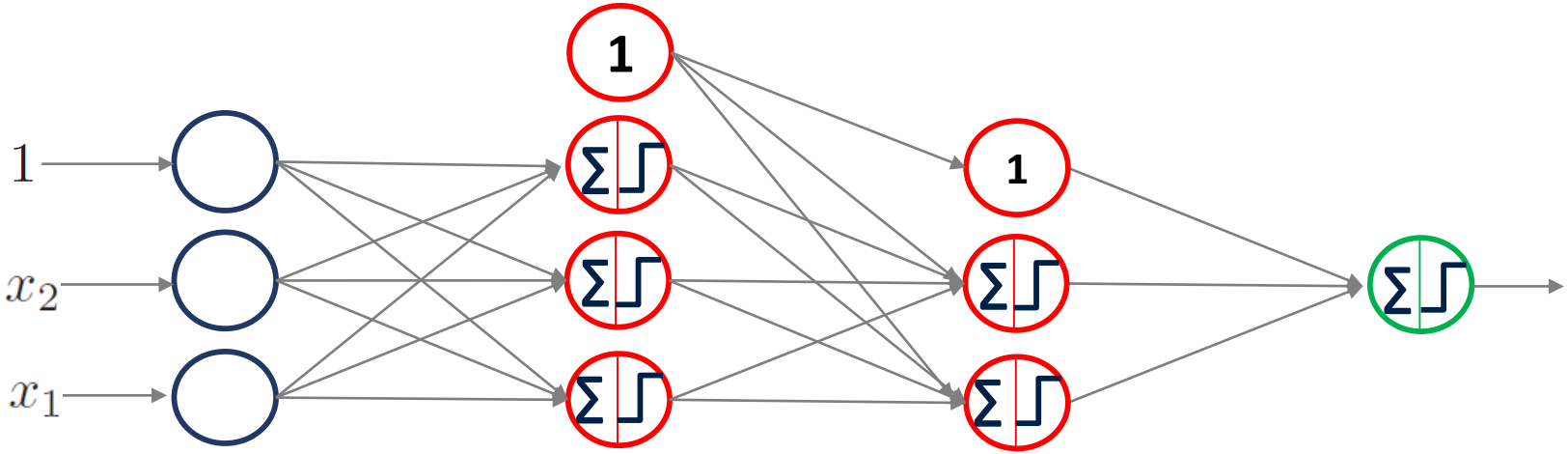
Limitation of single hidden layer

- A network with a single hidden layer can classify data points into classes comprising union of regions.
  - In the previous example, the class $\mathcal{C}_1$ comprised regions $(1, 1, 1)$, $(1, 1, 0)$, $(0, 1, 0)$ and $(0, 0, 0)$.
- How to adapt to more complex decision boundaries with the same hidden layer structure?
- But the hidden layer (in the last example) cannot generate classes with any arbitrary union of regions.
- Solution: Increase the number of hidden layers.

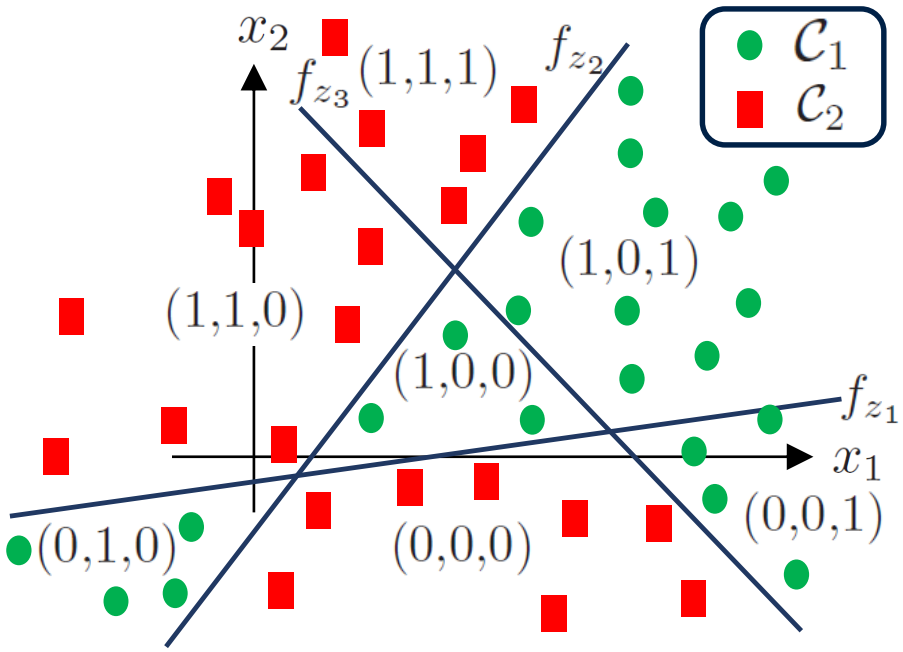
Two hidden layers

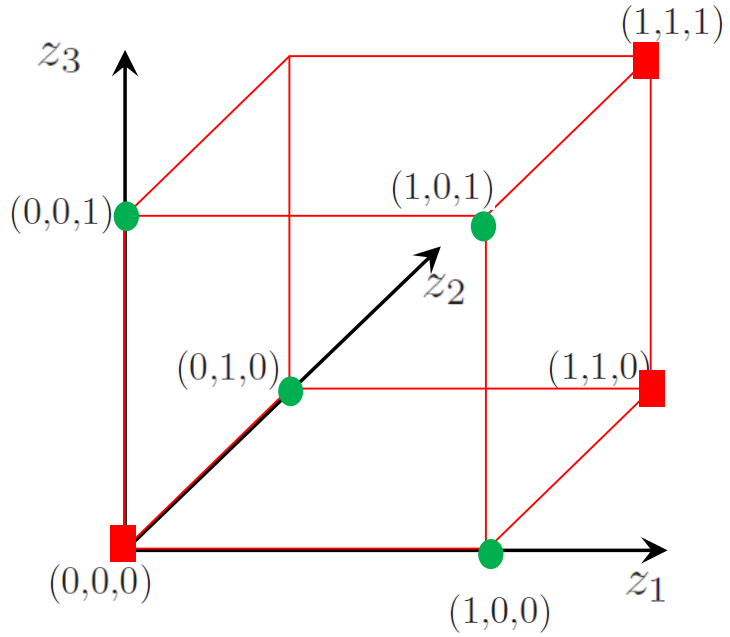


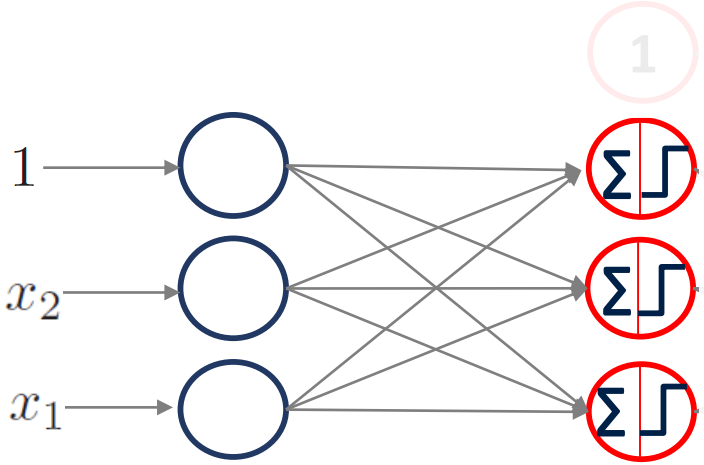




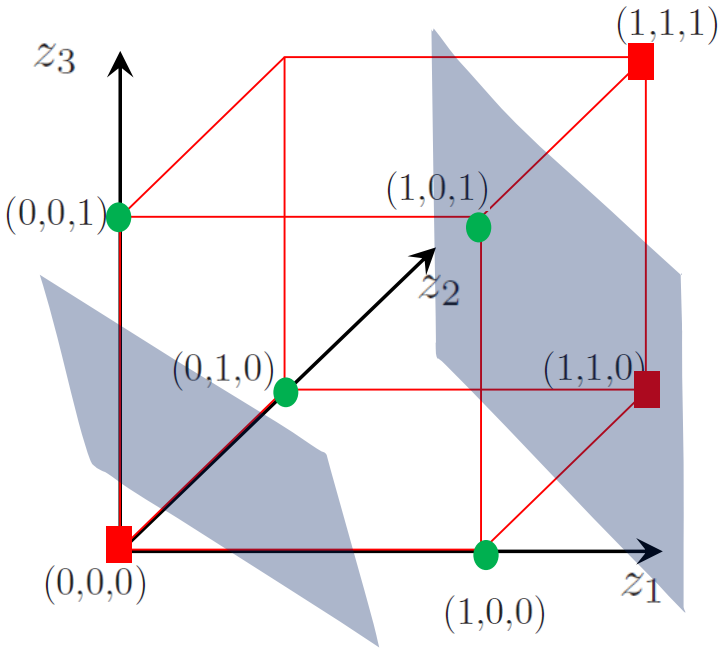
Two hidden layers

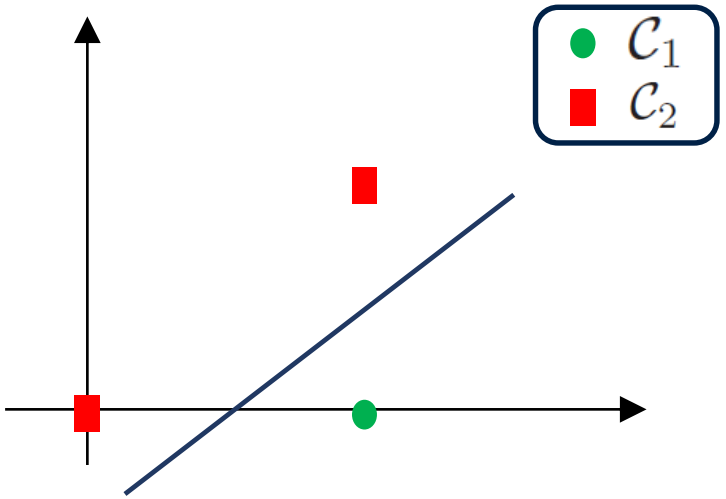


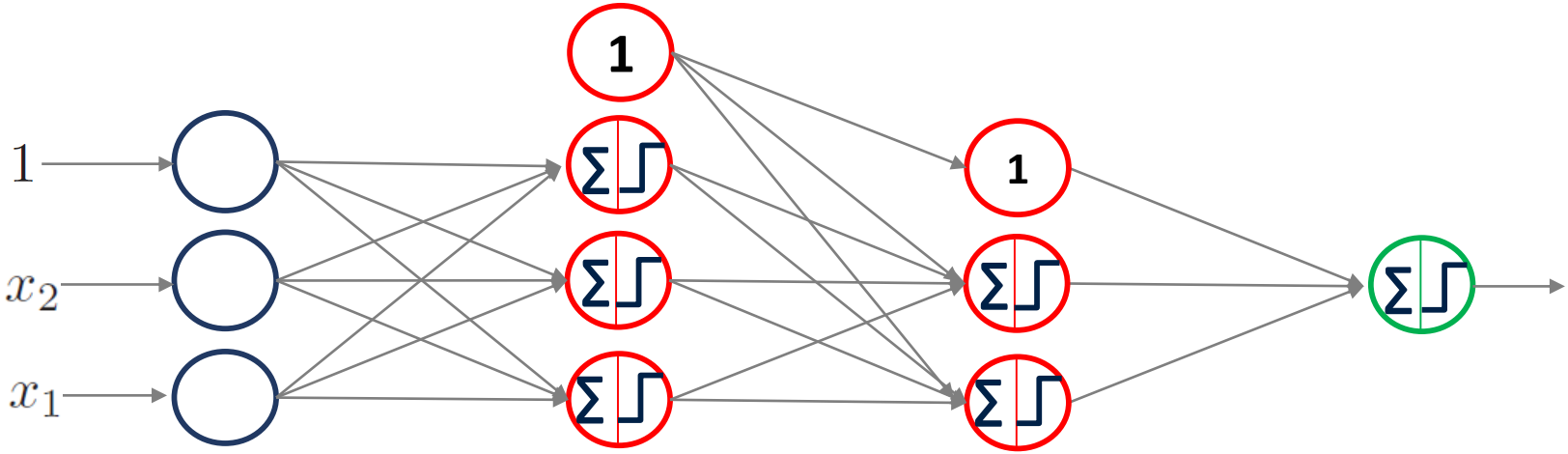




Two hidden layers



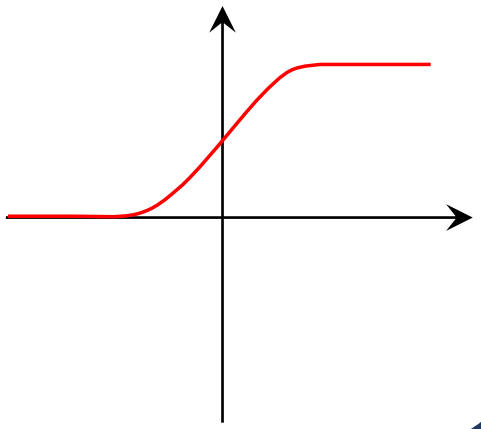




Sigmoid

- Bounded output: [0,1]
- Saturates for large input values, positive or negative.
  - Gradient becomes 0 (almost).
  - Leads to the problem of vanishing gradient in deep networks.
- Outputs not centered at 0.
- Not used much.

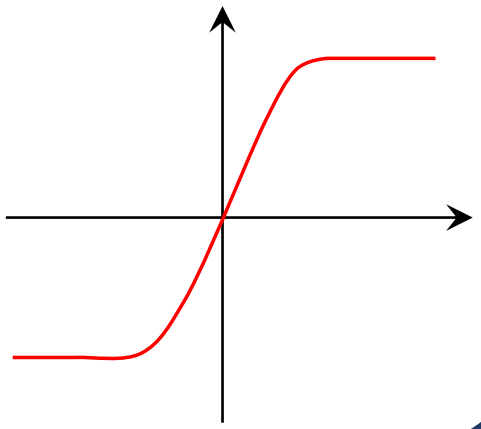
$$\mathcal{A}(z) = \frac{1}{1 + \exp(-z)}$$



tanh

- Bounded output: [-1,1]
- Saturates for large input values, positive or negative.
  - Gradient becomes 0 (almost).
  - Leads to the problem of vanishing gradient in deep networks.
- Outputs centered at 0.
- Better than sigmoid activation function.

$$\mathcal{A}(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$



ReLU

- Output not bounded on the positive side.
- Very efficient in derivative computation:

$$\mathcal{A}'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

- Known to have much faster convergence than tanh in some cases.
- If in the negative region, then unit is dead as there is no gradient.

$\mathcal{A}(z) = \max(z, 0)$