# Neural Network

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

# Neural Network

# Neural Network

- Logistic regression model, with sigmoid:

$$\hat{y}_i = \sigma(w_1 x_i^{(1)} + w_2 x_i^{(2)} + w_0) \tag{1}$$

- Two steps:

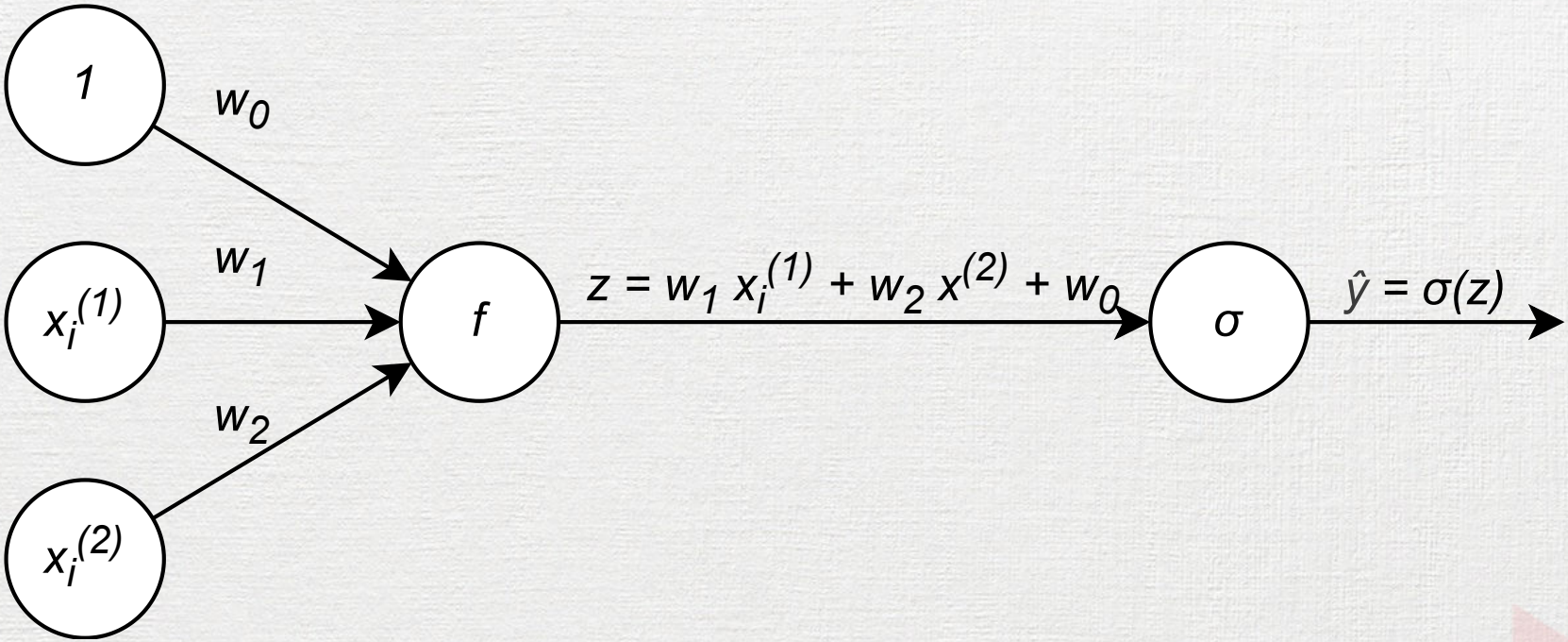
(1) Linear sum:

$$z = w_1 x_i^{(1)} + w_2 x_i^{(2)} + 1 * w_0 \tag{2}$$

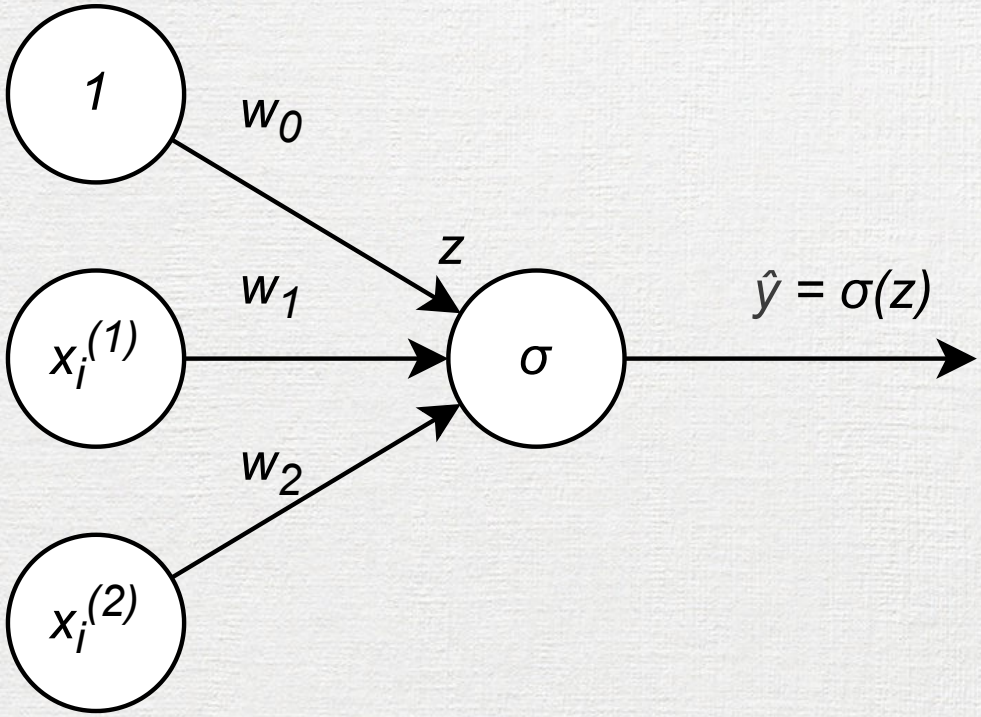
(2) Apply sigmoid function:

$$\hat{y} = \sigma(z) \tag{3}$$

# Neural Network



# Neural Network

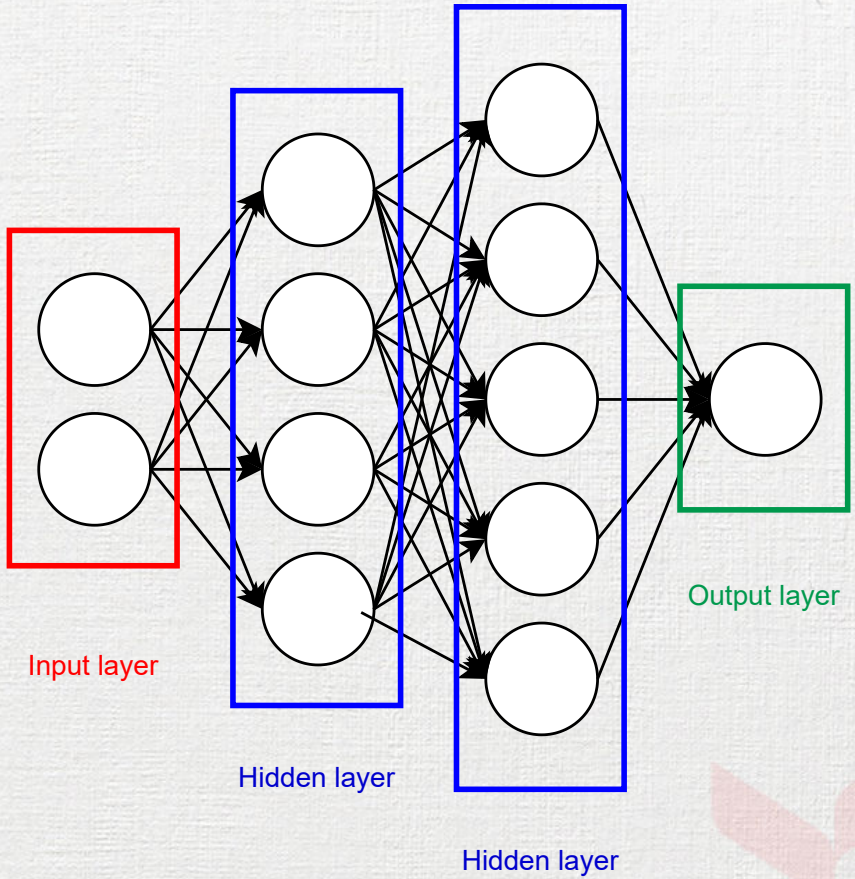



Flowchart of logistic regression model

- $w_0$ called bias coefficient, or free coefficient
- Sigmoid function is called activation function

# General Neural Network Model

- Input layer and output layer are required
- Hidden layers are optional
- Total layers = # layers - 1
- Each circle is called one node

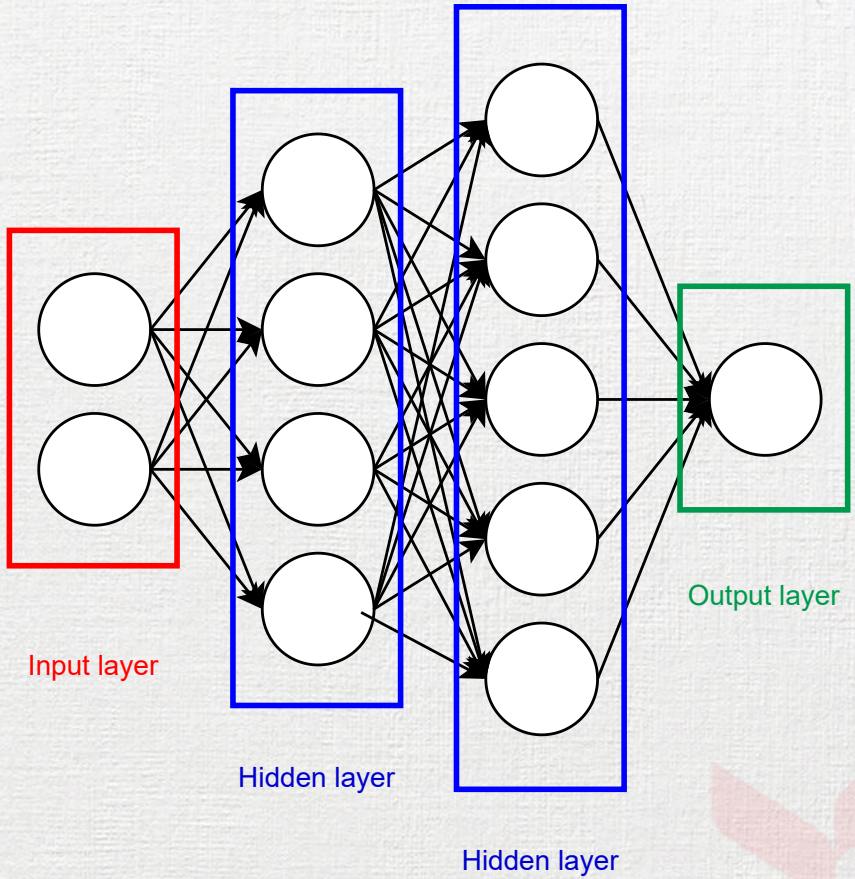



General architecture

# General Neural Network Model

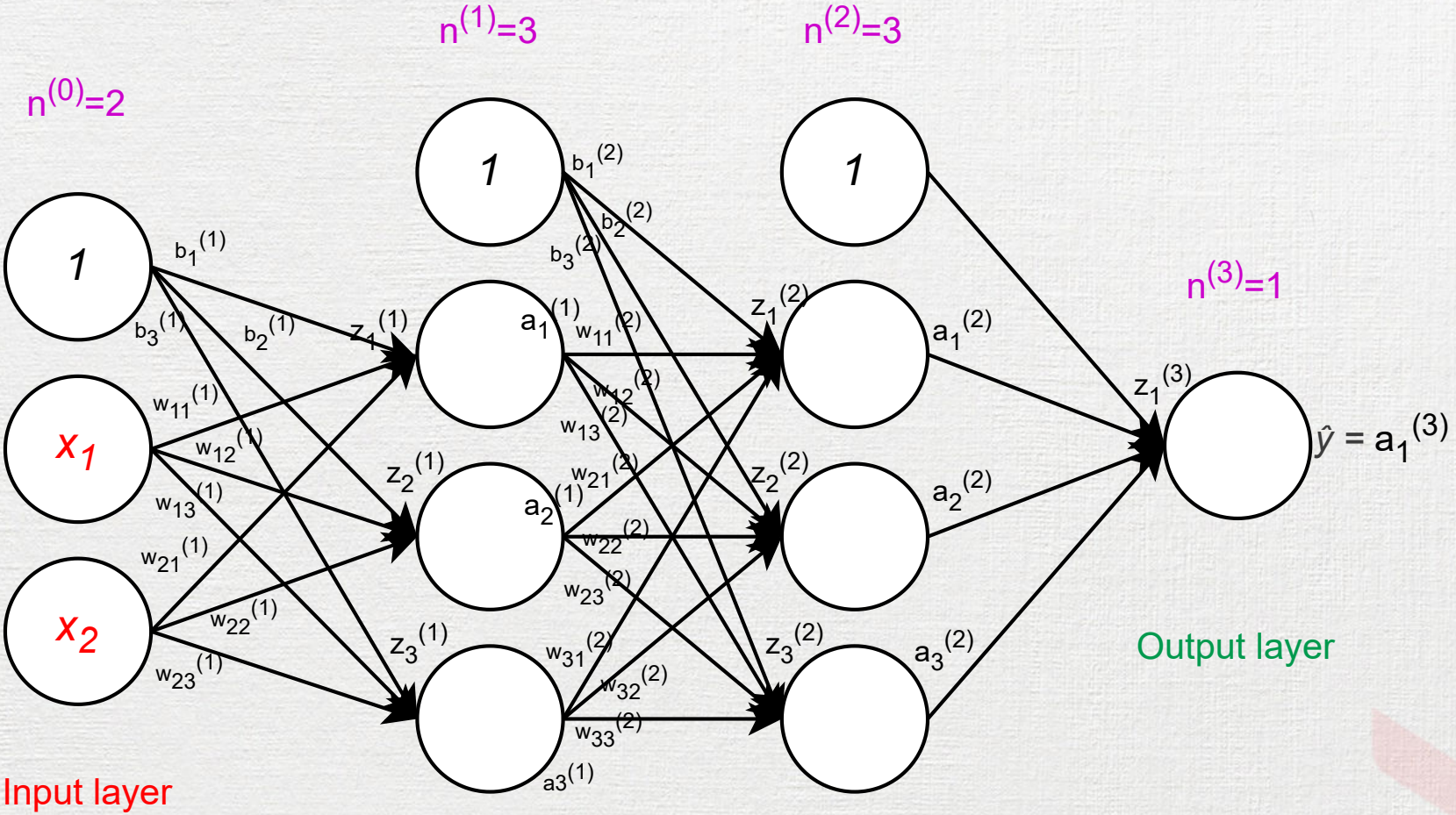
Each node in hidden layer and output layer:

- Is connected with all nodes with previous layer with the coefficents $w$
- Has a bias coefficient $w_0$
- Follows two steps of linear sum and appliance of activation function (sigmoid)




General architecture

# General Neural Network Model



# General Neural Network Model

- Node $i$ in layer $l$ with bias $b_i^{(l)}$ has 2 steps:

(1) Linear sum:

$$z_i^{(l)} = \sum_{j=1}^{l} a_j^{n^{(l-1)}} w_{ji}^{(l)} + b_i^{(l)} \tag{4}$$

Where $n^{(l)}$ is the number of neurons in layer $l$

(2) Apply activation function:

$$a_i^{(l)} = \sigma(z_i^{(l)}) \tag{5}$$

# General Neural Network Model

- At node 2 of layer 1, we have:

$$\begin{aligned} z_2^{(1)} &= x_1 w_{12}^{(1)} + x_2 w_{22}^{(1)} + b_2^{(1)} \\ a_2^{(1)} &= \sigma(z_2^{(1)}) \end{aligned} \tag{6}$$

- At node 3 of layer 2, we have:

$$\begin{aligned} z_3^{(2)} &= a_1^{(1)} w_{13}^{(2)} + a_2^{(1)} w_{23}^{(2)} + a_3^{(1)} w_{33}^{(2)} + b_3^{(2)} \\ a_3^{(2)} &= \sigma(z_3^{(2)}) \end{aligned} \tag{7}$$

# Feedforward

# Feedforward

- Let $x = a^{(0)}$ be the input layer, with size $3 \times 1$, we have:

$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{bmatrix} = \begin{bmatrix} a_1^{(0)} w_{11}^{(1)} + a_2^{(0)} w_{21}^{(1)} + a_3^{(0)} w_{31}^{(1)} + b_1^{(1)} \\ a_1^{(0)} w_{12}^{(1)} + a_2^{(0)} w_{22}^{(1)} + a_3^{(0)} w_{32}^{(1)} + b_2^{(1)} \\ a_1^{(0)} w_{13}^{(1)} + a_2^{(0)} w_{23}^{(1)} + a_3^{(0)} w_{33}^{(1)} + b_3^{(1)} \end{bmatrix} \quad (8)$$
$$= (W^{(1)})^T a^{(0)} + b^{(1)}$$

- Let $a^{(1)}$ be the output of the first layer. We have

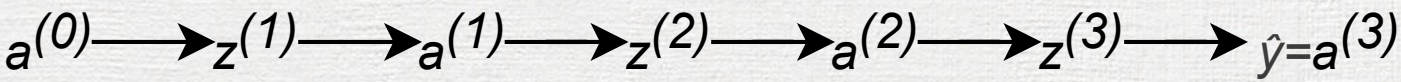
$$a^{(1)} = \sigma(z^{(1)}) \quad (9)$$

# Feedforward

- The output of the network will be

$$
\begin{aligned}
z^{(2)} &= (W^{(2)})^T a^{(1)} + b^{(2)} \\
a^{(2)} &= \sigma(z^{(2)}) \\
z^{(3)} &= (W^{(3)})^T a^{(2)} + b^{(3)} \\
\hat{y} &= a^{(3)} = \sigma(z^{(3)})
\end{aligned}
\tag{10}
$$

# Feedforward



# Loss function

- Gradient descent algorithm
- Step of derivative calculation of coefficients of loss function is done with the backpropagation algorithm

# Loss function

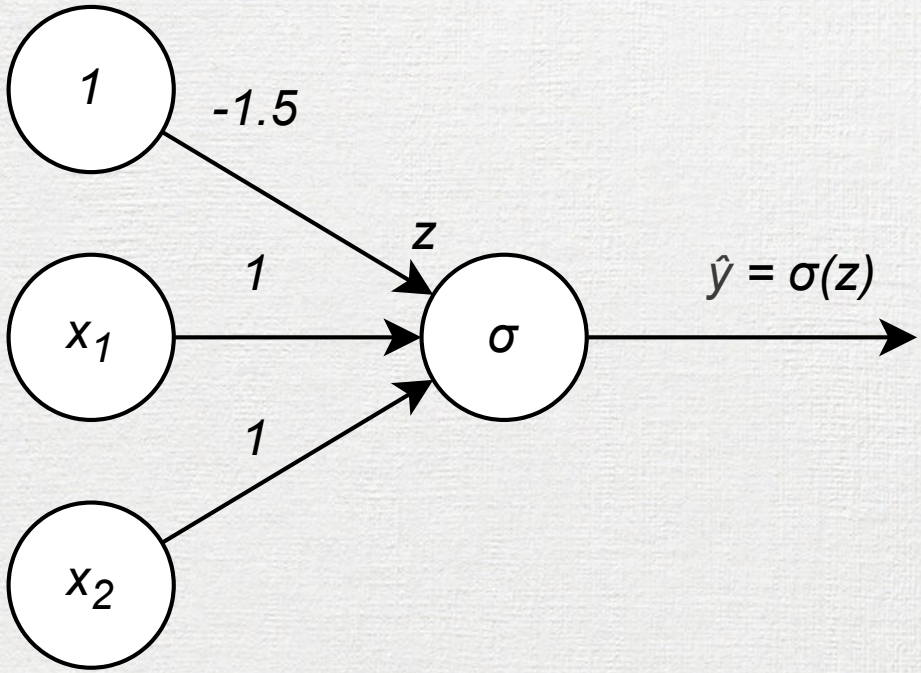
- Gradient descent algorithm
- Step of derivative calculation of coefficients of loss function is done with the backpropagation algorithm
- .... next lecture

# Logistic Regression vs Neural Network

# Problem: AND

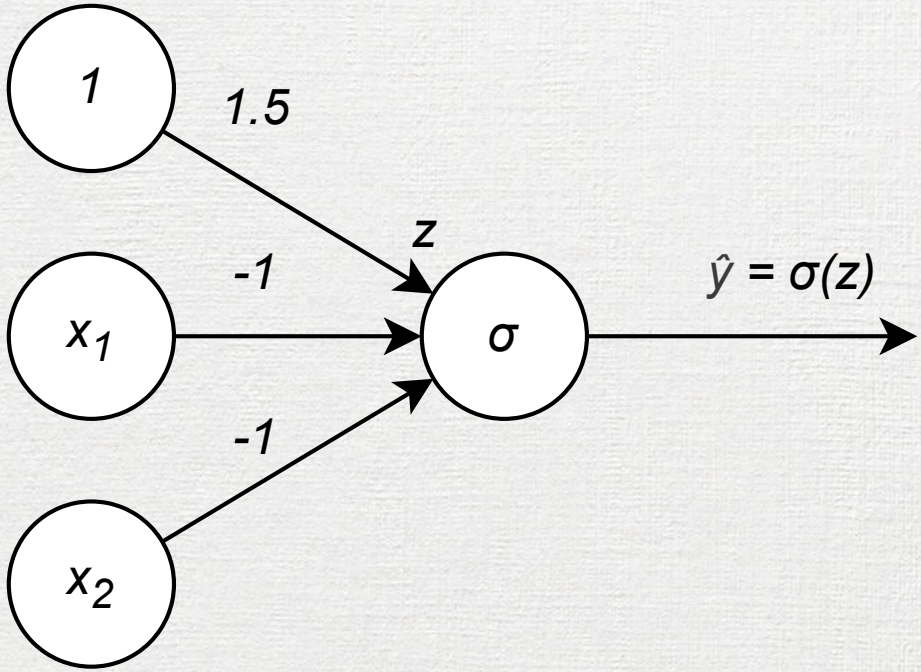
| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Problem: AND



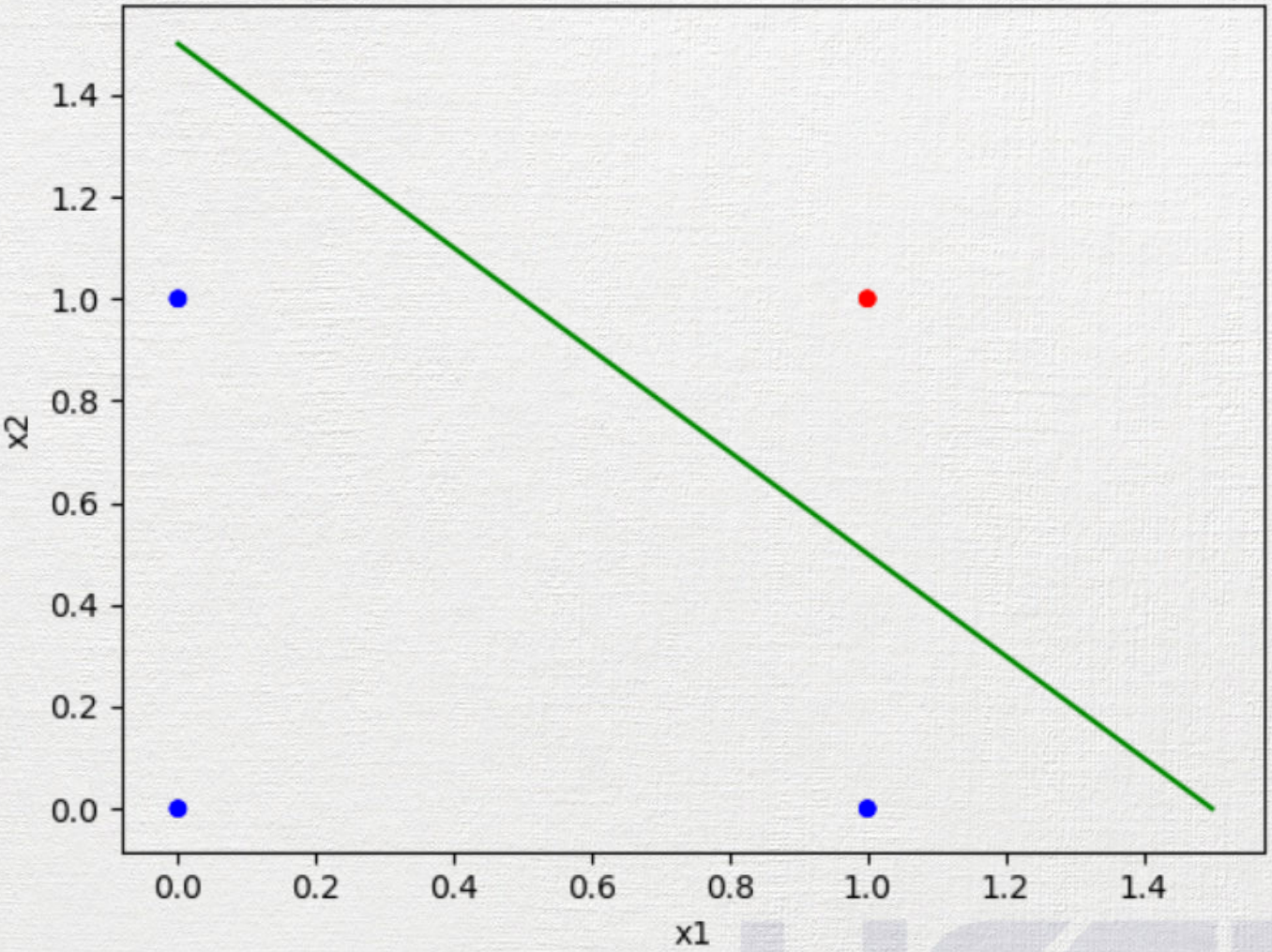

Flow chart for the problem $x_1$ & $x_2$

# Problem: AND




Flow chart for the problem $!(x_1 \ \& \ x_2)$

# Problem: AND

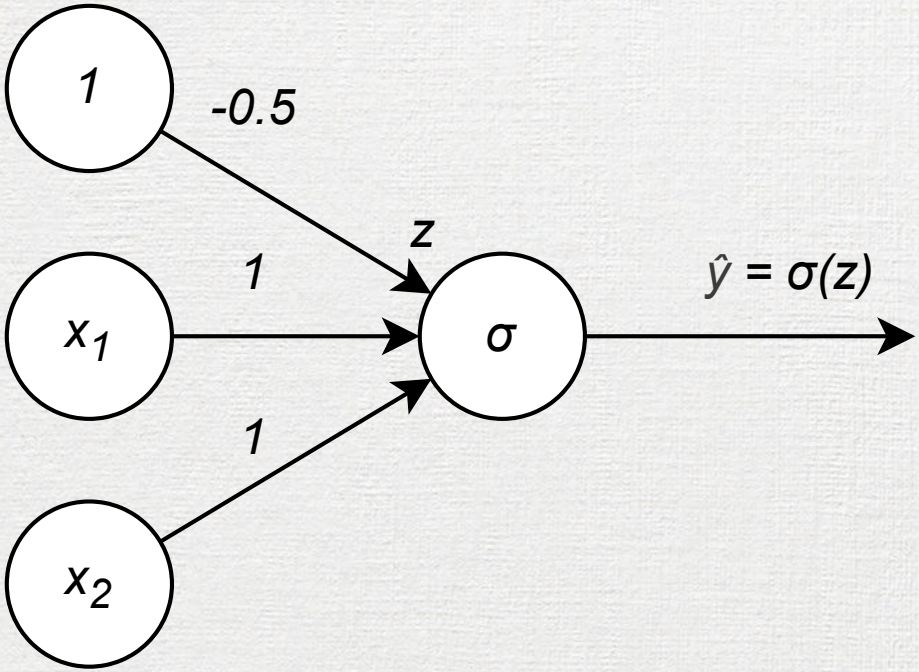



Separation line $y = 1.5 - 1 * x_1 - 1 * x_2$

# Problem: OR

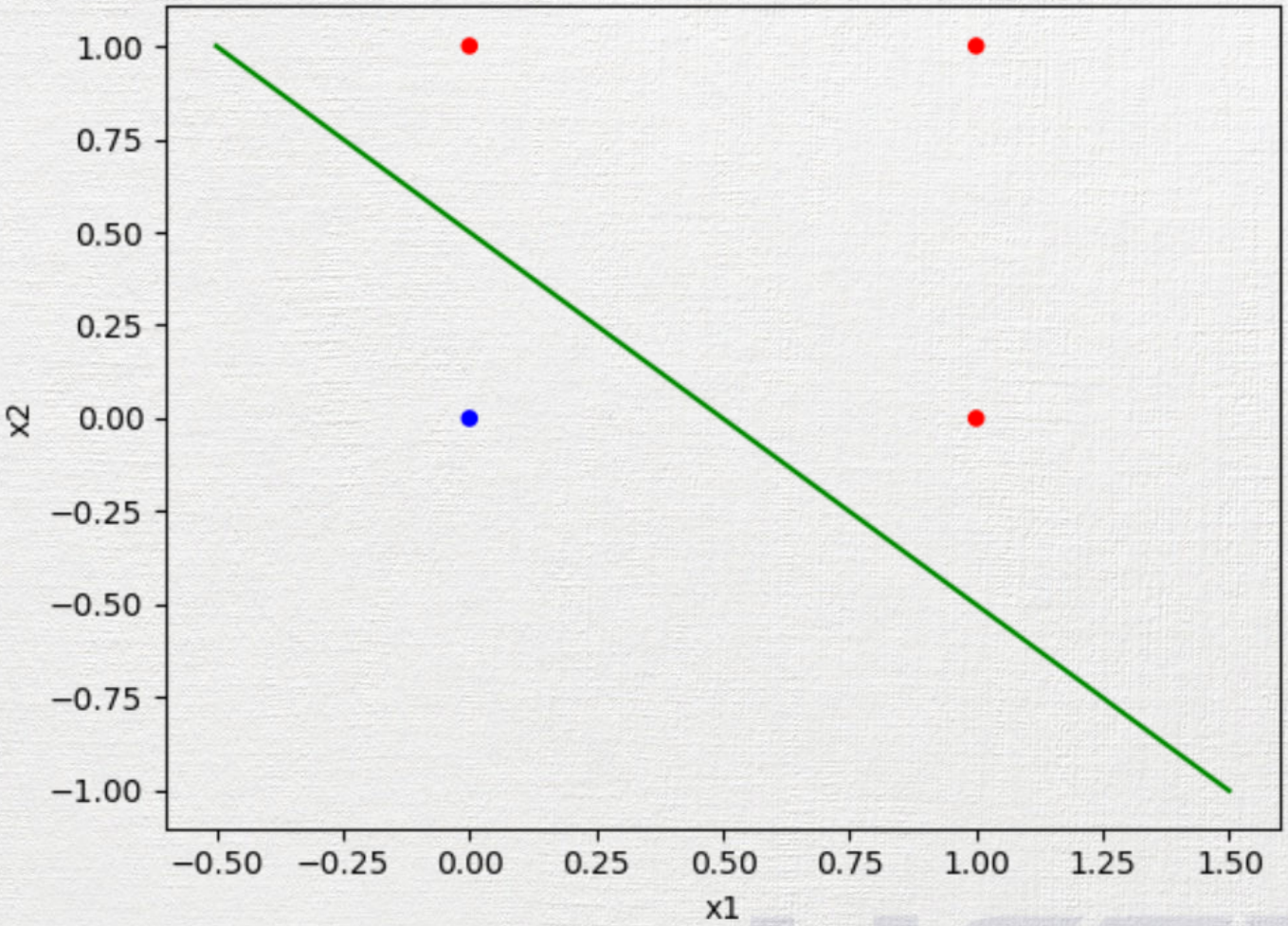
| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Problem: OR




Flow chart for the problem $x_1 \mid x_2$

# Problem: OR

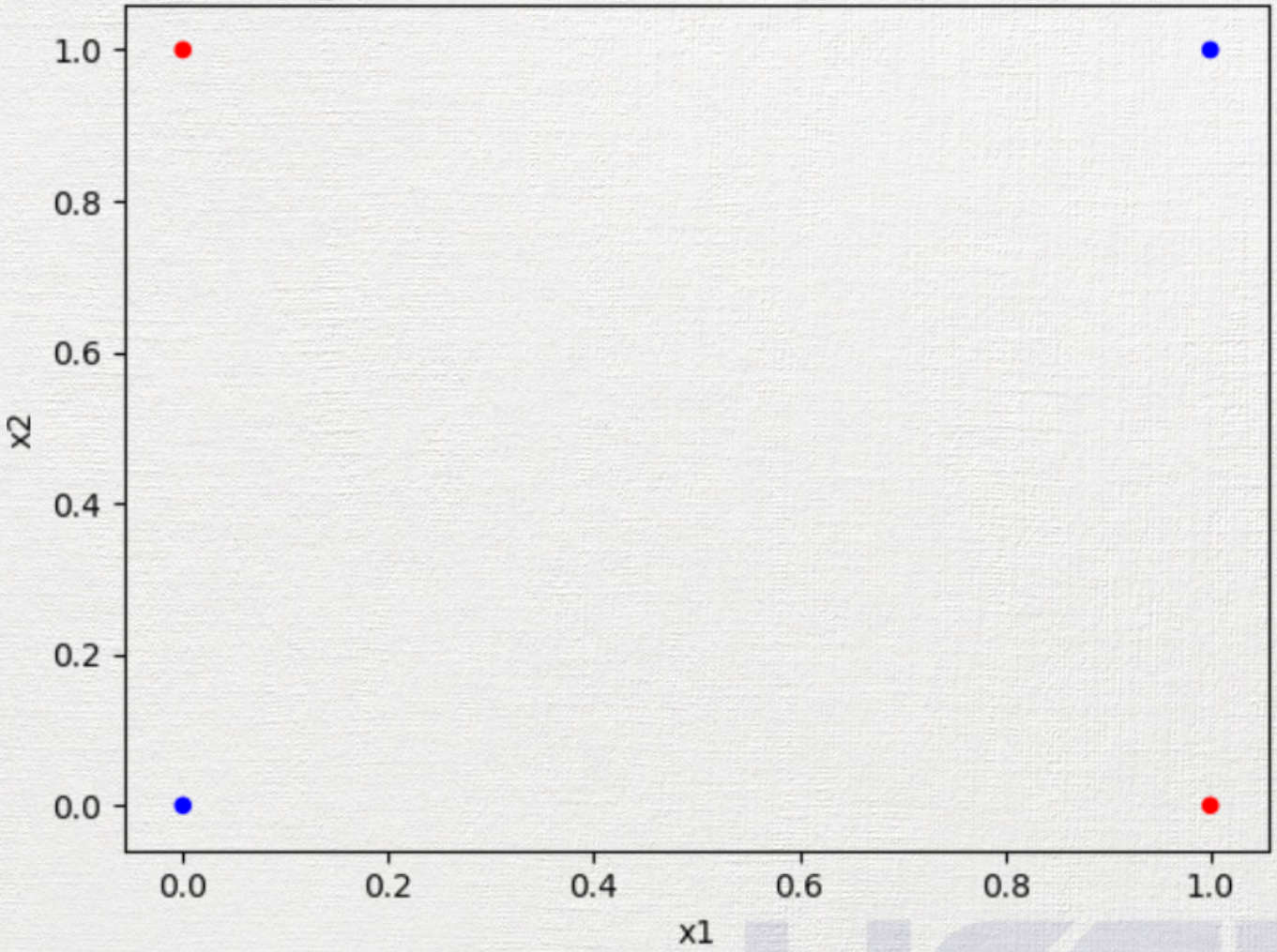



Separation line $y = -0.5 + 1 * x_1 + 1 * x_2$

# Problem: Exclusive OR (XOR)

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Problem: XOR



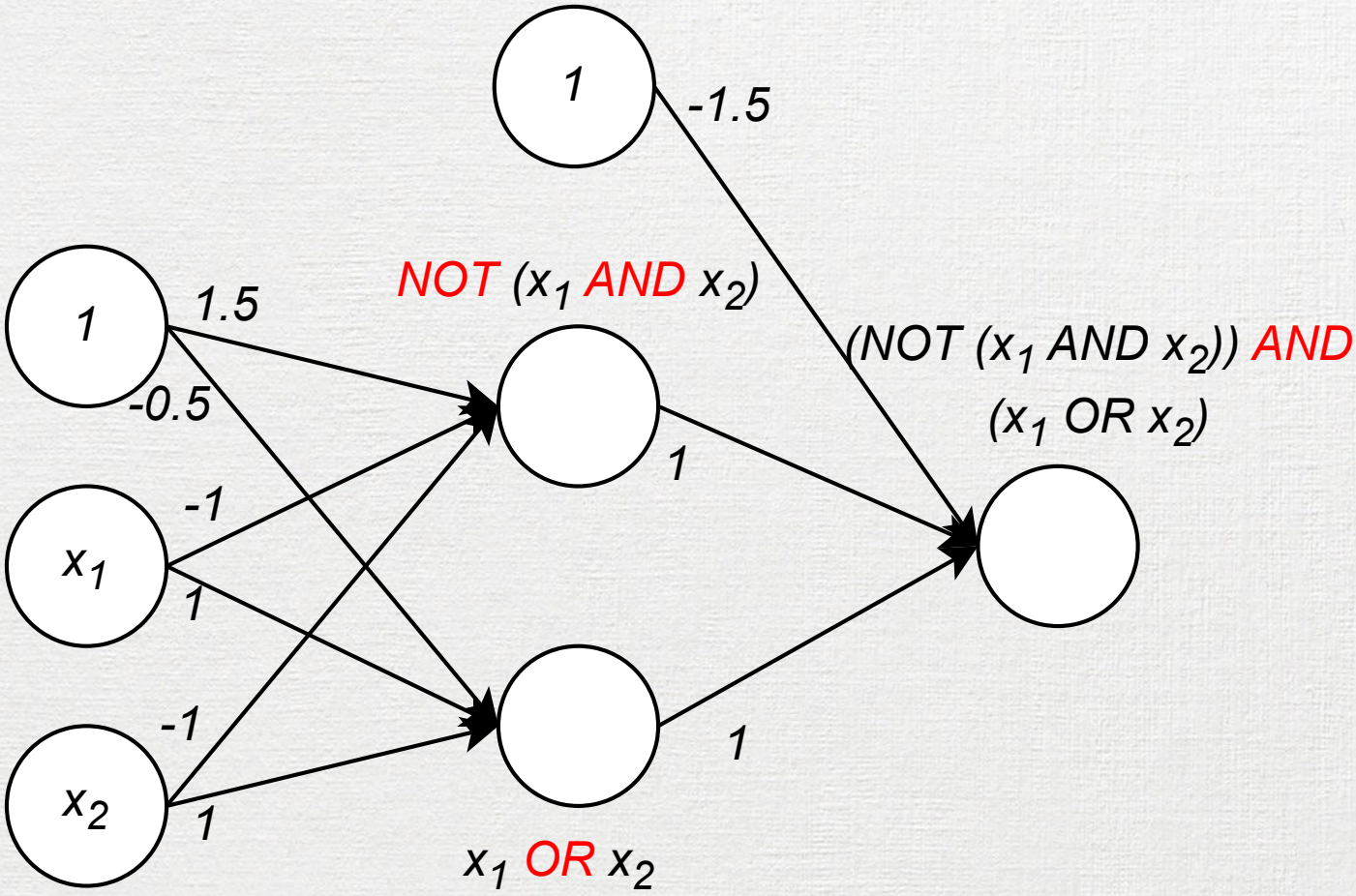

Cannot draw a separation line for XOR problem

# Problem: XOR

Rewrite XOR problem:

| $x_1$ | $x_2$ | $x_1 \oplus x_2$ | $x_1$ & $x_2$ | !($x_1$ & $x_2$) | $x_1 \mid x_2$ | !($x_1$ & $x_2$) & ($x_1 \mid x_2$) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |

# Solution: Neural Network XOR




$x_1 \oplus x_2$ with neural network

# Practice!

# Labwork 4: Neural Network

- Implement **(from scratch!)** a neural network, using proper object oriented programming concepts
  - dict, tuple, list ... are difficult to maintain
  - Input: a text file
    - First line: number of layers N
    - Next N lines: number of neurons for the $i^{th}$ layer
    - First layer is the input layer, last layer is the output layer
  - Output: a properly initialized neural network, with 2 options for weights and bias initialization:
    - Randomly [0..1]
    - From text file
  - Supports feedforward

# Labwork 4: Neural Network

- Write a report (in LaTeX):
  - Name it « Report.4.Neural.Network.tex »
  - How you design and implement the network architecture
  - How you implement feedforward
  - Experiment with the XOR network
    - Initialize network weight as in slides
    - Run with different inputs: (0, 0), (0, 1), (1, 0), (1, 1)
- Push your code and report to your forked repository