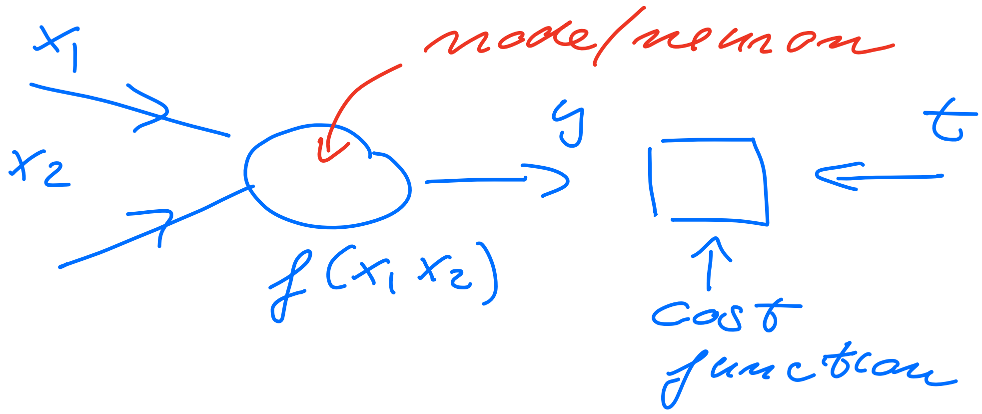
ML-ERASMUS, Dec 15, 2022

input x_1, x_2,

output $y \to t$ (target)



$$f(x_1 x_2) \longrightarrow \sigma(x_1, x_2, w_1, w_2, b)$$

activation function

$$= \sigma(x; \theta)$$

$$\theta = \{w, b\}$$

weights

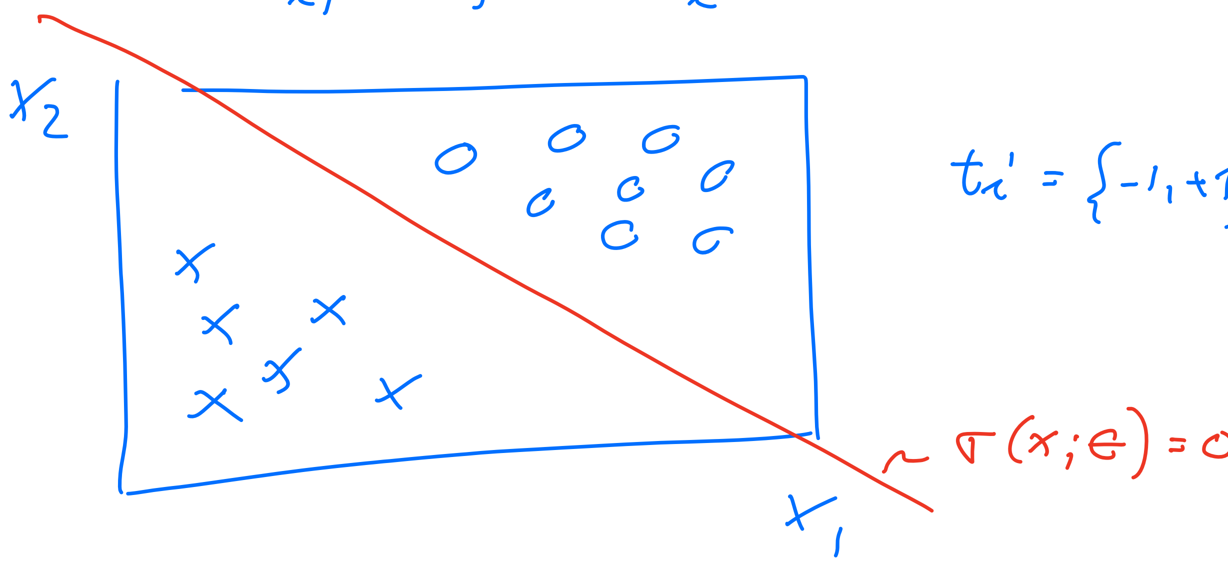
bias

$$\sigma(x_1, x_2, w_1, w_2, b) = x_1 w_1 + x_2 w_2 + b$$

$= y$

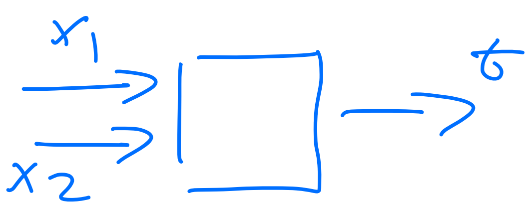
$X^T = [x_1, x_2]$ $\quad$ $w^T = [w_1 \; w_2]$

$\sigma(x_i; \theta) = x_i^T w + b = y_i$



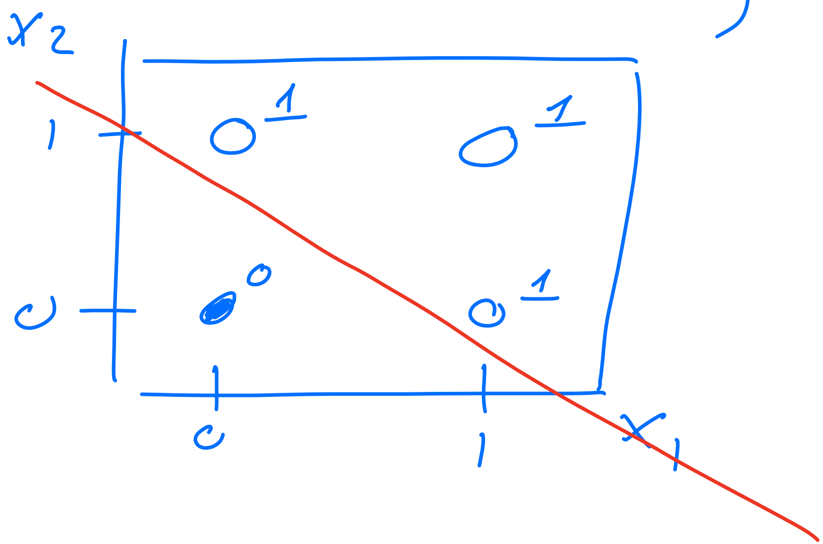
$t_i = \{-1, +1\}$

- Example: OR Gate



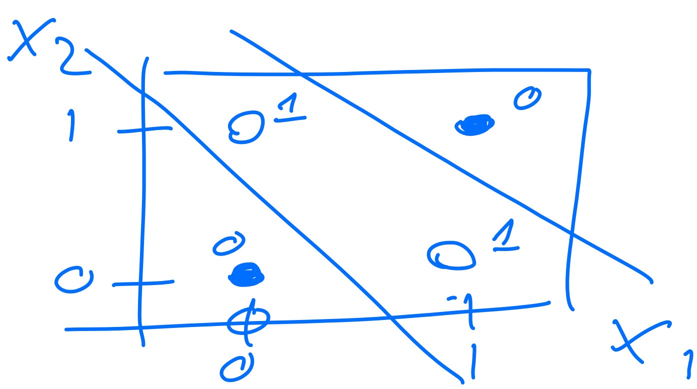
x_1	x_2	t
0	0	0
0	1	1
1	0	1
1	1	1

$$X = \{ [0,0]^T, [0,1]^T, [1,0]^T, [1,1]^T \}$$



XOR

x_1	x_2	$t(y)$
0	0	0
0	1	1
1	0	1
1	1	0



$$\Theta = \begin{bmatrix} b \\ w_1 \\ w_2 \end{bmatrix}$$

OR - gate

$$\begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = t = 0$$

$$\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = t = \underline{1}$$

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = \underline{1}$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = \underline{1}$$

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 4 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

$$\hat{e} = \hat{\beta} = (X^T X)^{-1} X^T y$$

$$= \begin{bmatrix} 1/4 & 1/2 & 1/2 \end{bmatrix}^T$$

$$y = X\hat{e} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/2 \\ 0 \\ 0 \end{bmatrix}$$

$$y^T = \begin{bmatrix} 1/4 & 3/4 & 3/4 & 5/4 \end{bmatrix}$$

$$t^T = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}$$

if prediction < 0.5 then
prediction $= 0$

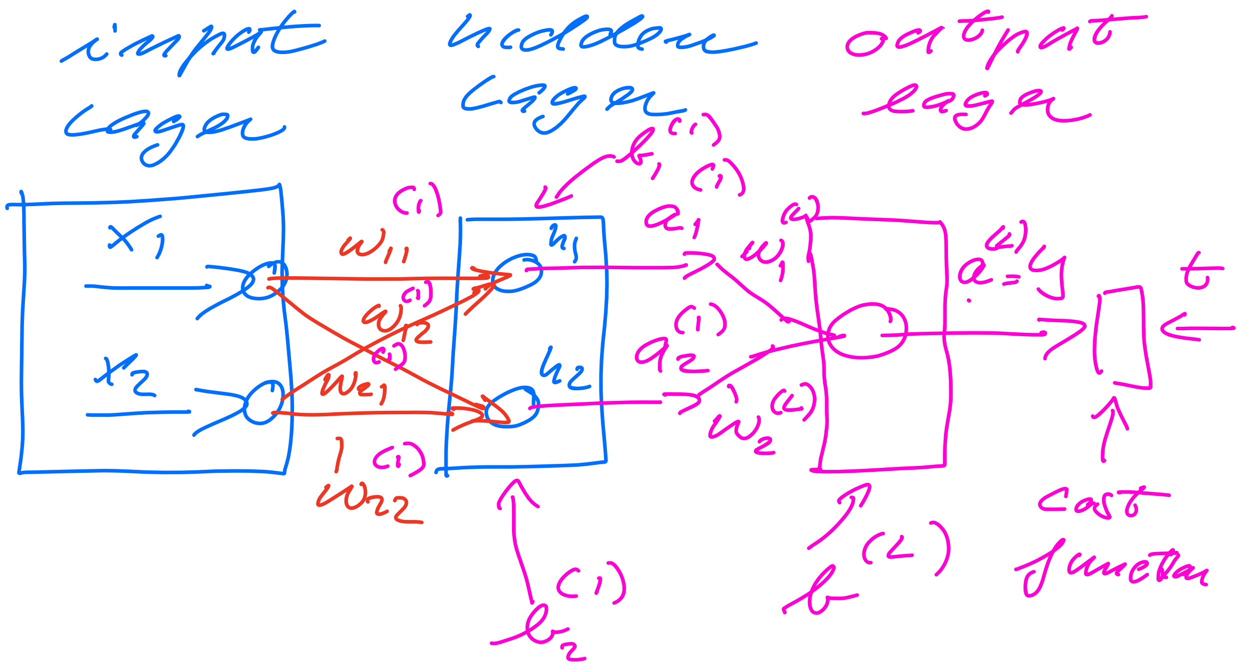
<u>XOR</u>

$$e^T = \begin{bmatrix} \frac{1}{2} & 0 & 0 \end{bmatrix}$$

$$y^T = \left[\tfrac{1}{2} \; \tfrac{1}{2} \; \tfrac{1}{2} \; \tfrac{1}{2} \right] \quad \text{FAIL}$$

$$t^T = [\,0 \;\; 1 \;\; 1 \;\; 0\,]$$

XOR we need to have a more complex with non-linearity. We can achieve this with a simple neural network with one hidden layer.



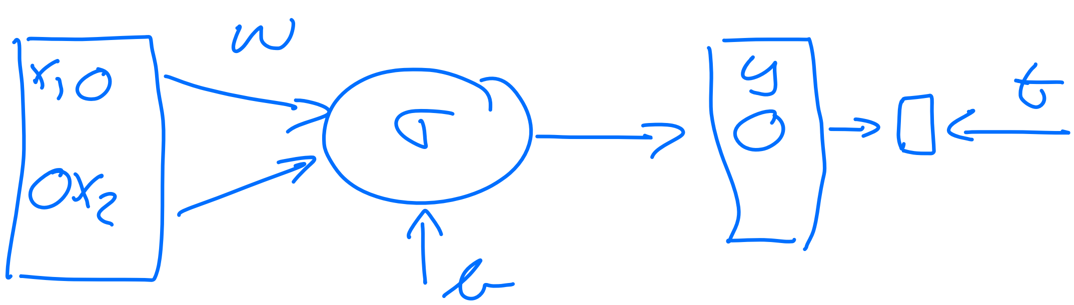
(i) Feed Forward stage with initial values for weights and biases

(ii) Backward propagation to update the parameters
$$\Theta = \{ w, b \}$$

(iii) Repeat (i) and (ii) till we reach convergence

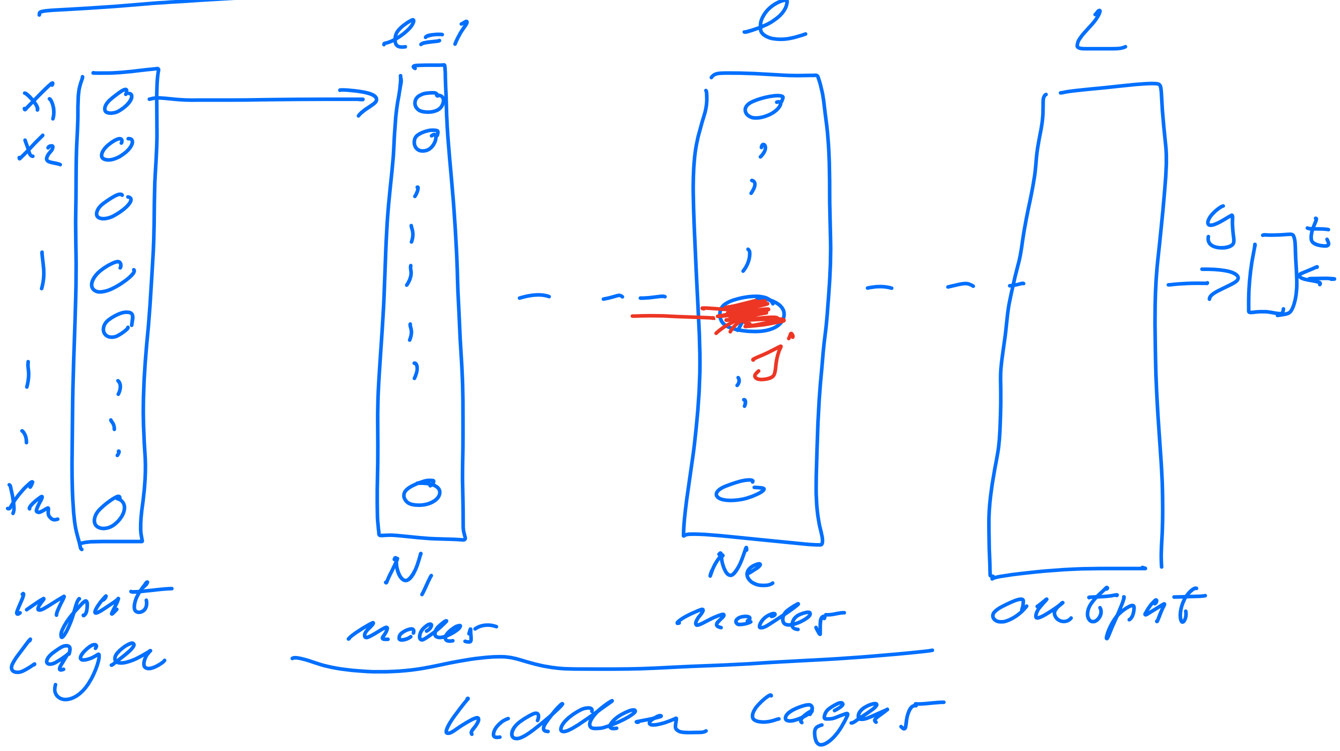
Backward propagation is a usage of the chain rule in order to compute gradients and find optimal parameters Θ

($\equiv$ reverse mode in automatic differentiation)



$$\sigma(x;\theta)$$

FFNN (MLP)



$$z_j^l = \sum_{i=1}^{N_{l-1}} w_{ij}^l \, a_i^{l-1} + b_j^l$$

$$a_i^{l-1} = \sigma(z_i^{l-1})$$

input

output from node i in layer $l-1$

output $a_j^l = \sigma(z_j^l)$

w_{ij}^l = weight

b_j^l = bias

Regression case

$$C(\theta) = C(W, b)$$

$$= \frac{1}{2} \sum_{i=1}^{n} (y_i - t_i)^2$$

$$= \frac{1}{2} \sum_{i=1}^{n} (a_i^L - t_i)^2$$

$$y_i = a_i^L$$

$$a_j^l = \sigma(z_j^l) = \frac{1}{1+e^{-z_j^l}}$$

(sigmoid function)

Back propagation algo gives the expressions for the gradients of a neural network

$$\frac{\partial C}{\partial W} = 0 = \frac{\partial C}{\partial b}$$

Aiding quantities

$$\frac{\partial z_j^l}{\partial w_{ij}^l} = a_i^{l-1}$$

$$\frac{\partial z_j^l}{\partial a_i^{l-1}} = w_{ij}^l$$

$$a_j^l = \frac{1}{1+e^{-z_j^l}} = \sigma(z_j^l)$$

$$\frac{\partial a_j^l}{\partial z_j^l} = \sigma(z_j^l)(1-\sigma(z_j^l))$$

$$\frac{\partial C(t)}{\partial w_{jk}^L} = (a_j^L - t_j)\frac{\partial a_j^L}{\partial w_{jk}^L}$$

output

$$\frac{\partial a_j^L}{\partial w_{jk}^L} = \frac{\partial a_j^L}{\partial z_j^L}\frac{\partial z_j^L}{\partial w_{jk}^L}$$

$$= a_j^L(1-a_j^L)a_k^{L-1}$$

$$\frac{\partial C}{\partial w_{jk}^L} = \underbrace{a_j^L(1-a_j^L)a_k^{L-1}}_{\delta_j^L}(a_j^L - t_j)$$

$$= \delta_j^L (a_j^L - t_j)$$

$$\delta_j^L = \sigma'(z_j^L) \frac{\partial C}{\partial a_j^L}$$

$$\frac{\partial C}{\partial w_{jk}^L} = \delta_j^L a_k^{L-1}$$

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L}$$

$$= \frac{\partial C}{\partial z_j^L}$$

$$\boxed{\begin{aligned} \frac{\partial C}{\partial w^L_{jk}} &= \delta^L_j \, a^{L-1}_k \\ \delta^L_j &= \sigma'(z^L_j) \frac{\partial C}{\partial a^L_j} \\ \delta^L_j &= \frac{\partial C}{\partial b^L_j} \end{aligned}}$$

$$L \to l$$

$$\delta^l_j = \frac{\partial C}{\partial z^l_j} = \sum_k \frac{\partial C}{\partial z^{l+1}_k} \frac{\partial z^{l+1}_k}{\partial z^l_j}$$

$$= \sum_k \delta^{l+1}_k \frac{\partial z^{l+1}_k}{\partial z^l_j}$$

$$z^{l+1}_{\cancel{k}j} = \sum_{i=1}^{N_l} w^{l+1}_{ij} a^l_i + b^{l+1}_j$$

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l)$$

$$\boxed{\begin{aligned} \delta_j^l &= \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l) \\ w_{jk}^l &\leftarrow w_{jk}^l - \eta\, \delta_j^l a_k^{l-1} \\ b_j^l &\leftarrow b_j^l - \eta \frac{\partial C}{\partial b_j^l} \\ &= b_j^l - \eta\, \delta_j^l \end{aligned}}$$