Lecture Fys-Stk3155/4155, October 12, 2023

NNs (Neural Networks) part 1

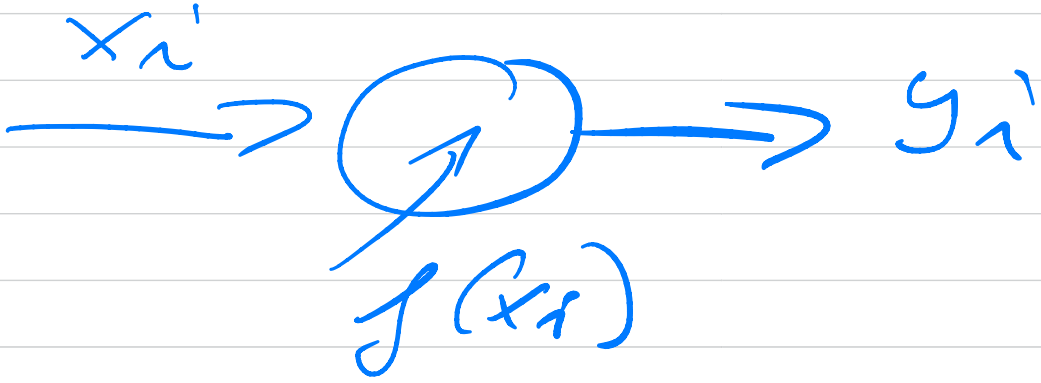
- Universal approximation theorem
- components of NNs
  - (i) cost / objective function
  - (ii) Architecture of NN = model
    - a) number of hidden layers
    - b) number of nodes in hidden layer

c) type of activation function $\sigma(z)$

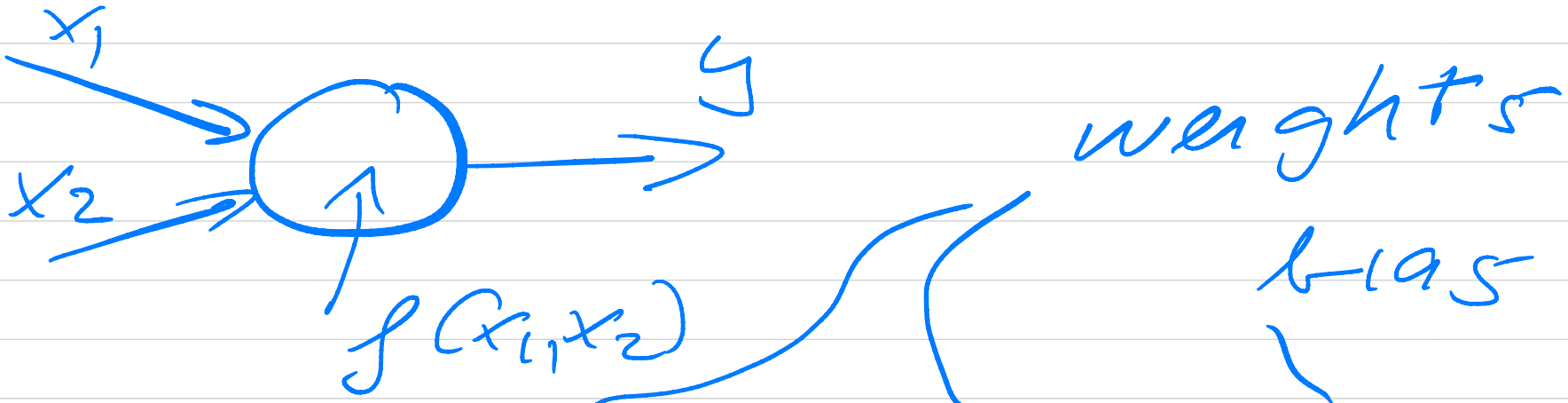
- Back propagation for setting up gradients for training.
- Gradient methods for training (GD, SGD...)
- Hyperparameters, λ in $\lambda \sum_j \beta_j^2$ etc, learning rates etc etc

Graphical representation

simple network with one node

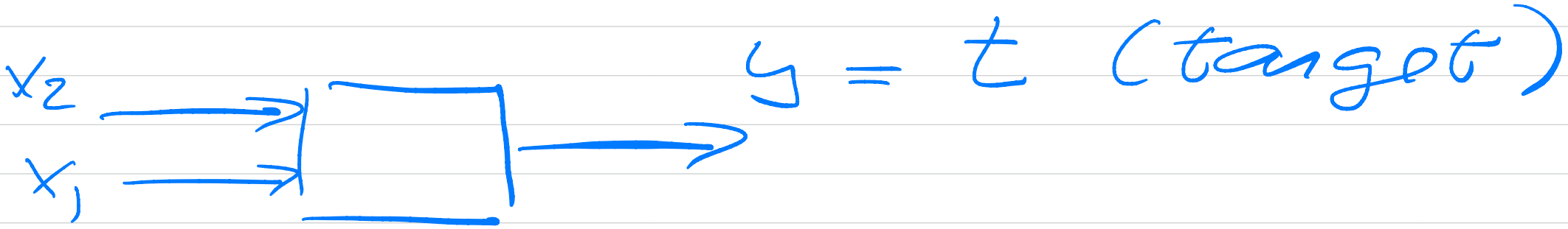


Example

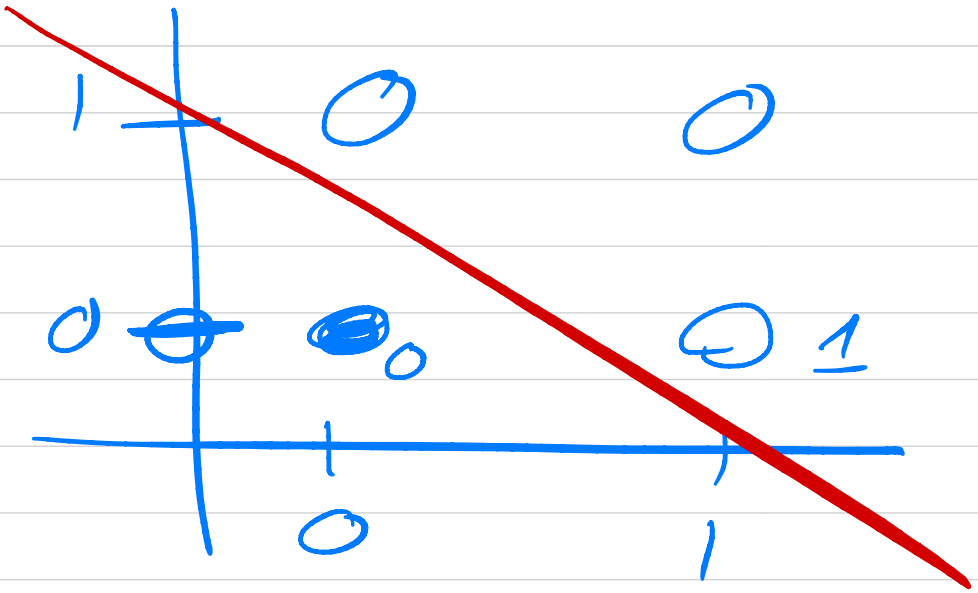


$$y = x_1 w_1 + x_2 w_2 + b$$

OR-gate



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



$$x_1 w_1 + x_2 w_2 + b = y$$

$$w^T = [w_1, w_2]$$

$$x^T = [x_1, x_2]$$

$$y = x^T w + b$$

$$x^T = [0\ 0], [0, 1], [1, 0]$$
$$[1, 1]$$

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

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

if $y < \frac{1}{2}$
$y = 0$
if $y > 1/2$
$y = 1$

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

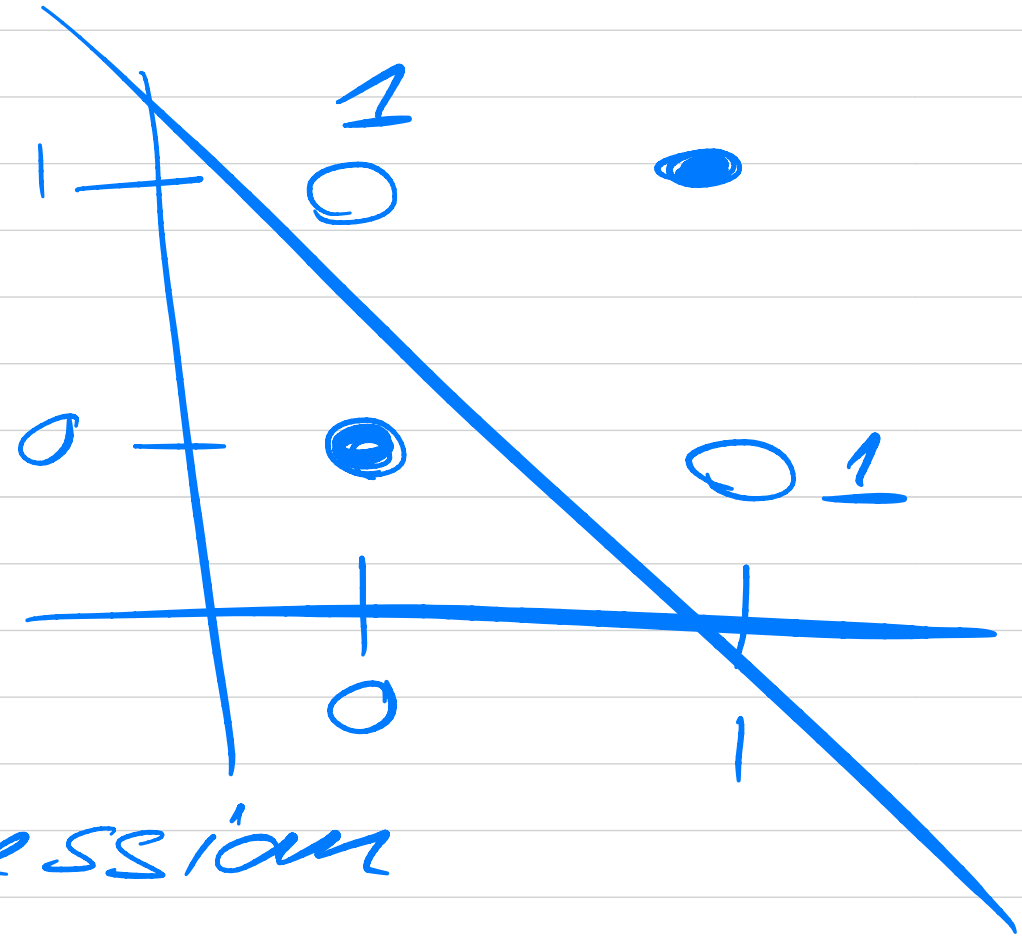
$$t = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

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

$$\hat{\theta} = [1/4, 1/2, 1/2] \quad \hat{y}^T = y^T = [1/4, 3/4, 3/4, 3/4]$$

XOR

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



Linear regression

$$y = \left[\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\right]$$

$$t = [0, 1, 1, 0]$$

Universal approximation theorem

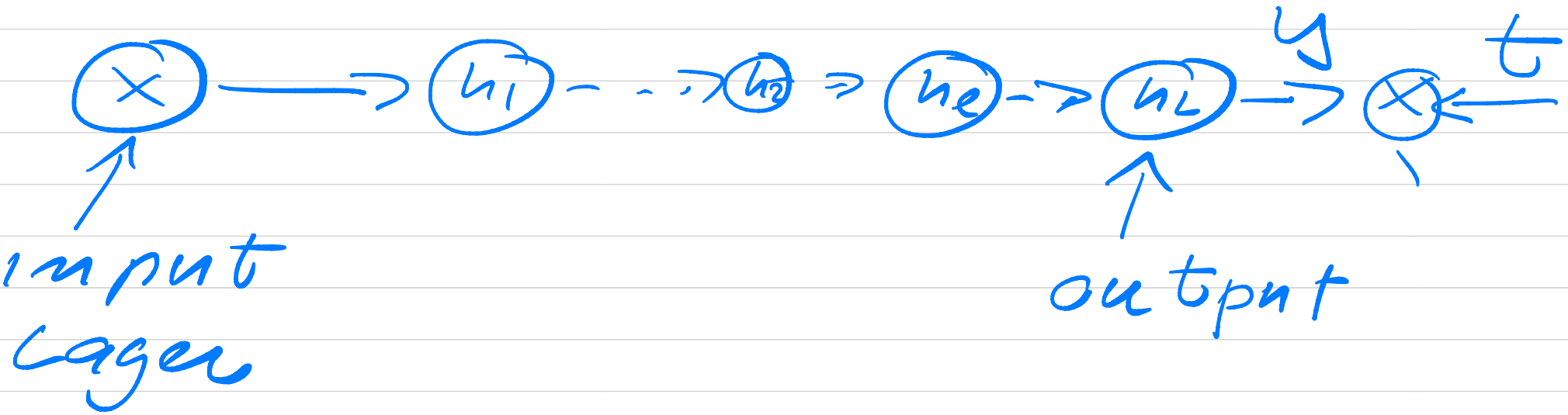
(Cybenko 1989, Hornik 1991)

Given function $F \in [0,1]^d$ and $\varepsilon > 0$, there exists/is a one hidden-layer NN $f(x;\Theta)$ of the form with $\Theta = \{W, b\}$ $W \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$ for which

$$|f(x;\Theta) - F(x)| < \varepsilon$$

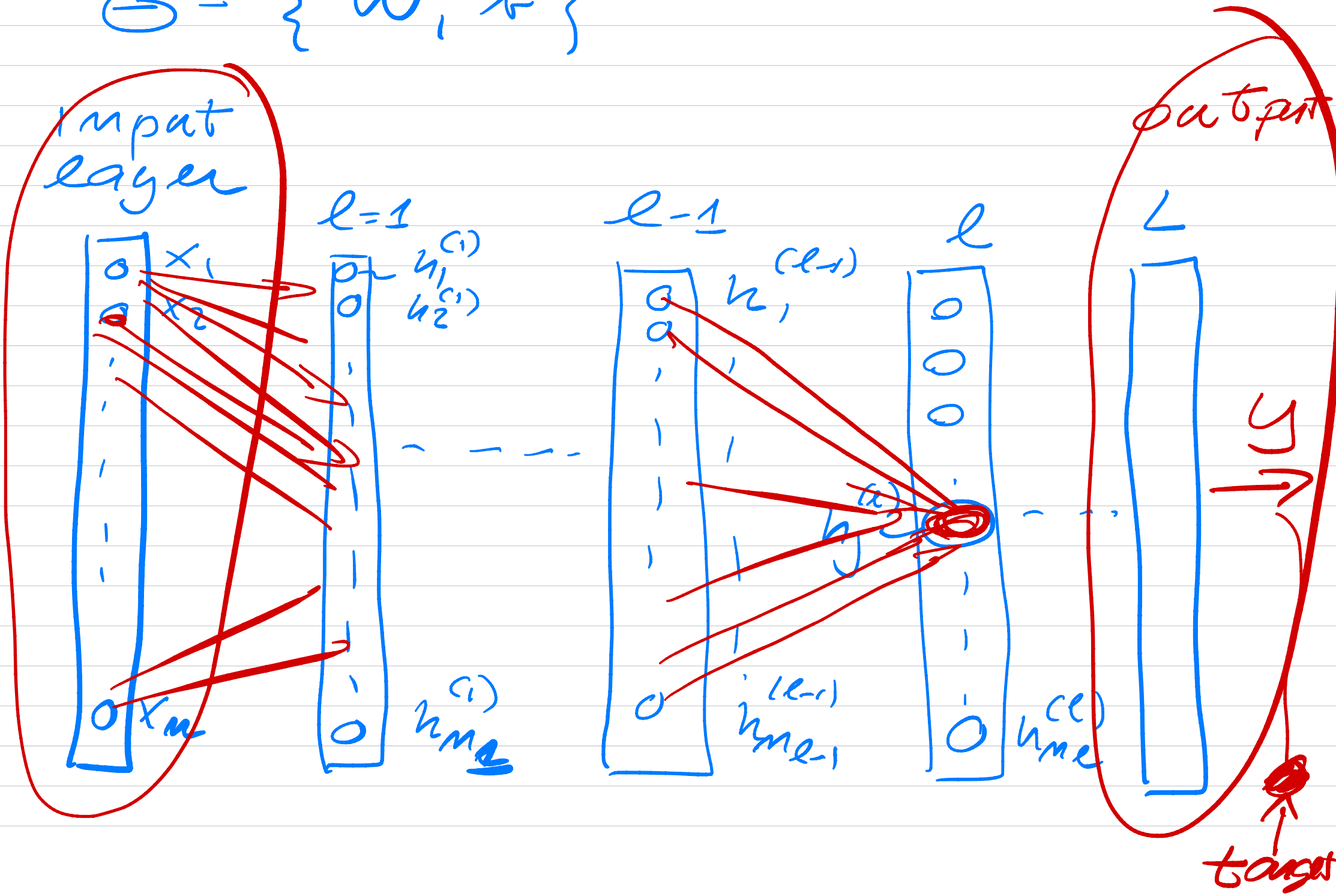
for all $x \in [c,1]^d$

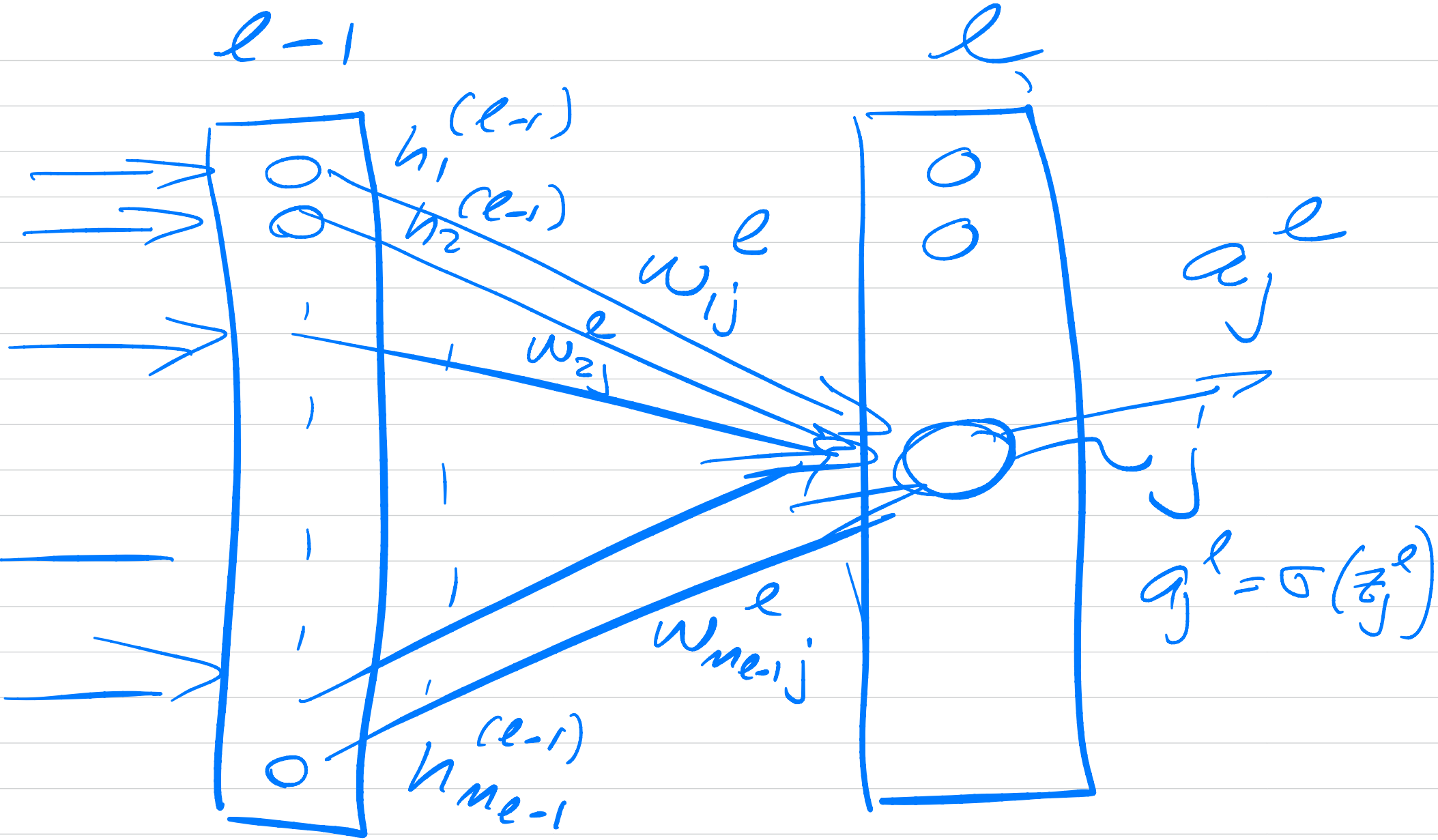
Back propagation Algo



- Feed Forward stage
- Back propagation stage

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


input layer
l=1
l-1
l
L
output
x_1
x_2
x_m
$h_1^{(1)}$
$h_2^{(1)}$
$h_{m_1}^{(1)}$
$h_1^{(l-1)}$
$h_{m_{l-1}}^{(l-1)}$
$h_i^{(l)}$
$h_{m_l}^{(l)}$
y
target



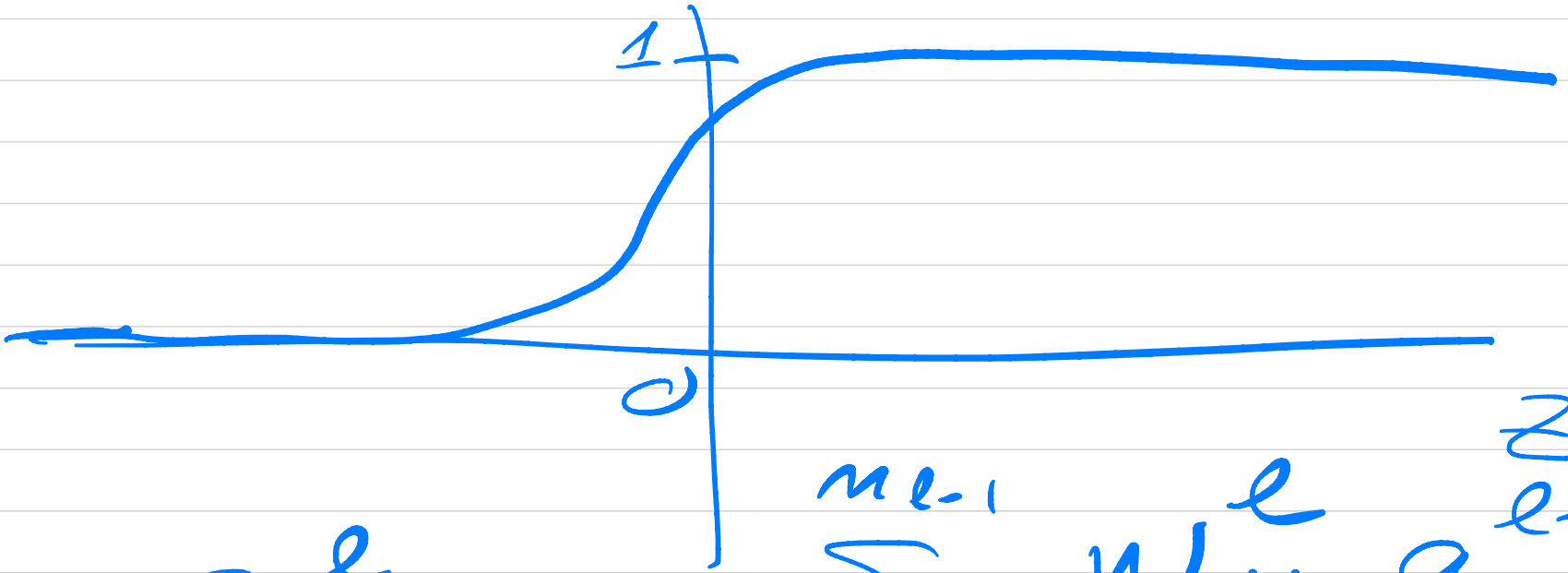
input to j in layer ℓ

z_j^{ℓ}

$$a_j^{\ell} = \sigma(z_j^{\ell})$$

sigmoid is much used

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



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

$$z^{l} = \left[W^{(l)}\right]^{T} a^{(l-1)} + b^{(l)}$$

W is a matrix containing the unknown parameters to train.

b is so-called bias

Need a cost function.

Here $C(\theta) = \frac{1}{2} \sum_{i=1}^{m_L} \left(y_i - t_i\right)^2$

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

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

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

in example here

some intermediate steps

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

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

$$a_j^l \left(1 - a_j^l\right)$$

we start at $l = L$ (output)

$$C(\theta) = \frac{1}{2} \sum_i \left(a_i^L - t_i \right)^2$$

y_i

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

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

$$a_j^L \left(1 - a_j^L \right) a_k^{L-1}$$

$$\frac{\partial C}{\partial w^l_{jk}} = (a^l_j - t_j) \underbrace{a^l_j (1 - a^l_j)}_{\text{from specific activation function } \frac{d\sigma(z)}{dz}} a^{l-1}_k$$

$$\delta^l_j = \underbrace{a^l_j (1 - a^l_j)}_{\sigma'(z)} \underbrace{(a^l_j - t_j)}_{\frac{\partial C}{\partial a^l_j}}$$

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

$$\delta_j^{l} = \frac{\partial C}{\partial b_j^{l}} \frac{\partial b_j^{l}}{\partial z_j^{l}} = \frac{\partial C}{\partial b_j^{l}}$$

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

expressions for layer L

what about layer $-l-$?

$l \to e$

$$\delta_j^e = \frac{\partial C}{\partial z_j^e} = \sum_k \frac{\partial C}{\partial z_k^{e+1}} \frac{\partial z_k^{e+1}}{\partial z_j^e}$$

$$= \sum_k \delta_k^{e+1} \frac{\partial z_k^{e+1}}{\partial z_j^e}$$

$$z_j^{e+1} = \sum_{i=1}^{N_e} w_{ij}^{e+1} a_i^e + b_j^{e+1}$$

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

$$\delta_j^\ell = \sum_k \delta_k^{\ell+1} w_{kj}^{\ell+1} \sigma'(z_j^\ell)$$

updates -

$$w_{jk}^\ell \leftarrow w_{jk}^\ell - \eta \, \delta_j^\ell a_k^{\ell-1}$$

$$b_j^\ell \leftarrow b_j^\ell - \eta \frac{\partial C}{\partial b_j^\ell}$$

$$= b_j^\ell - \eta \, \delta_j^\ell$$

Algorithm

- Define architecture
- initialize $\{W, b\} = \theta$
- Define nodes, hidden layers + activation functions
- hyperparameters
- learning rate + gradient approx
- Define cost functions

- set up x
- Perform first feed forward
- continue to layer L

$$a^L = \sigma^L(\sigma^{L-1}(\dots \sigma^1(z^1))\dots)$$

- compute δ_j^L
- Then Backpropagate

$$l = L-1, L-2, \dots 1$$

$$\delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l)$$

Training (Gradient part)

$$w_{jk}^{\ell} \leftarrow w_{jk}^{\ell} - \eta \, \delta_j^{\ell} a_k^{\ell-1}$$

$$b_j^{\ell} \leftarrow b_j^{\ell} - \eta \, \delta_j^{\ell}$$

continue till convergence