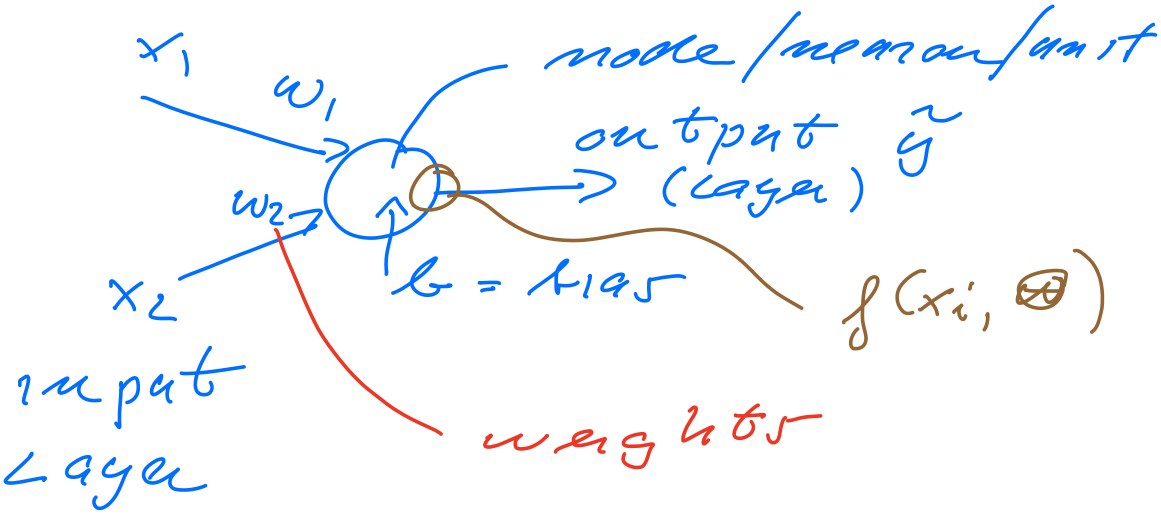
Lecture November 29

Mathematics of feed forward Neural Networks (FFNN) / Multi-Layer Perceptron (MLP)

- Simple perceptron model.
- activation function.



$$\tilde{y}_i = f(x_i ; \Theta) = w^T x_i + b$$

activation

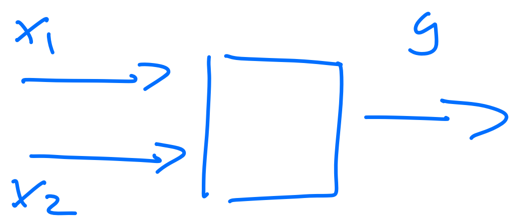
$$\Theta = \{ w, b \} \qquad w^T = [w_1 \; w_2]$$

Targets / outputs $\left\{ \; y_i \quad (= t_i) \right.$

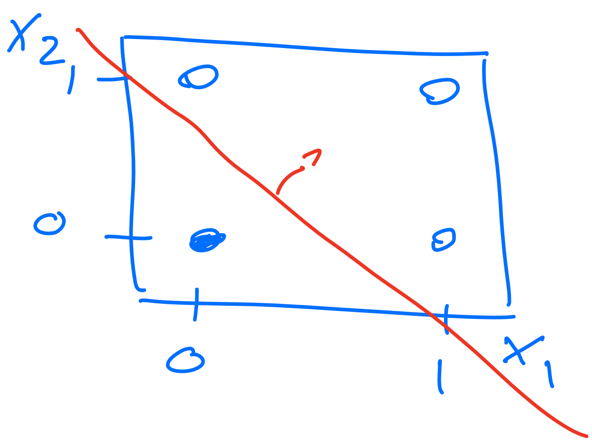
$$D = \{ (x_0, y_0), (x_1, y_1) \ldots (x_{n-1}, y_{n-1}) \}$$

Examples : XOR, OR, AND gates

OR-gate



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



$y = 0$ ●

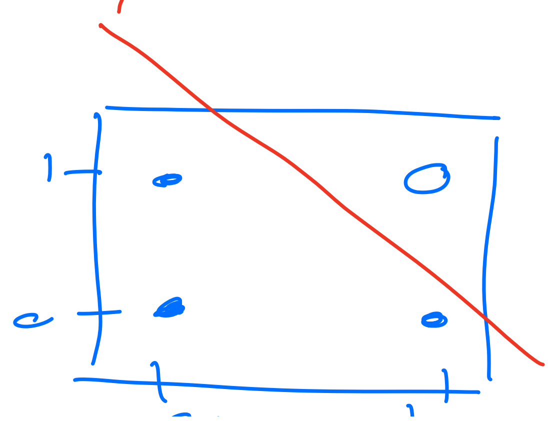
$y = 1$ ○

linear reg

logistic reg.

AND-gate

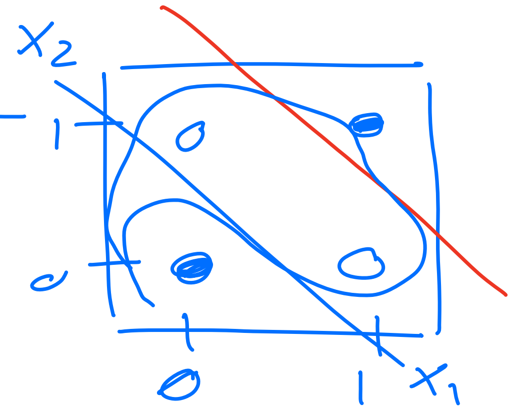
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



0 x_1

XOR - Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$\tilde{y}_i = x_i w_1 + x_i w_2 + b$$

intercept

Linear regression

$$X = \begin{array}{c} \\ \\ \end{array}\begin{array}{c} \begin{array}{ccc} & x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \end{array} \in \mathbb{R}^{4 \times 3}$$

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

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

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

OR-Gate

$$= [1/4,\ 1/2,\ 1/2]^T$$

$$\tilde{y} = X\hat{\theta} = [1/4,\ 3/4,\ 3/4,\ 5/4]$$

$\tilde{y} \leq 1/2$ then $\tilde{y} = 0$

$\tilde{y} > 1/2$ then $\tilde{y} = 1$

Linear/Logistic are ok

AND gate

$$y = [0, 0, 0, 1]^T$$

$$\tilde{y} = [-1/4, 1/4, 1/4, 3/4]^T$$

$\tilde{y} \leq 0.5$ then $\tilde{y} = 0$

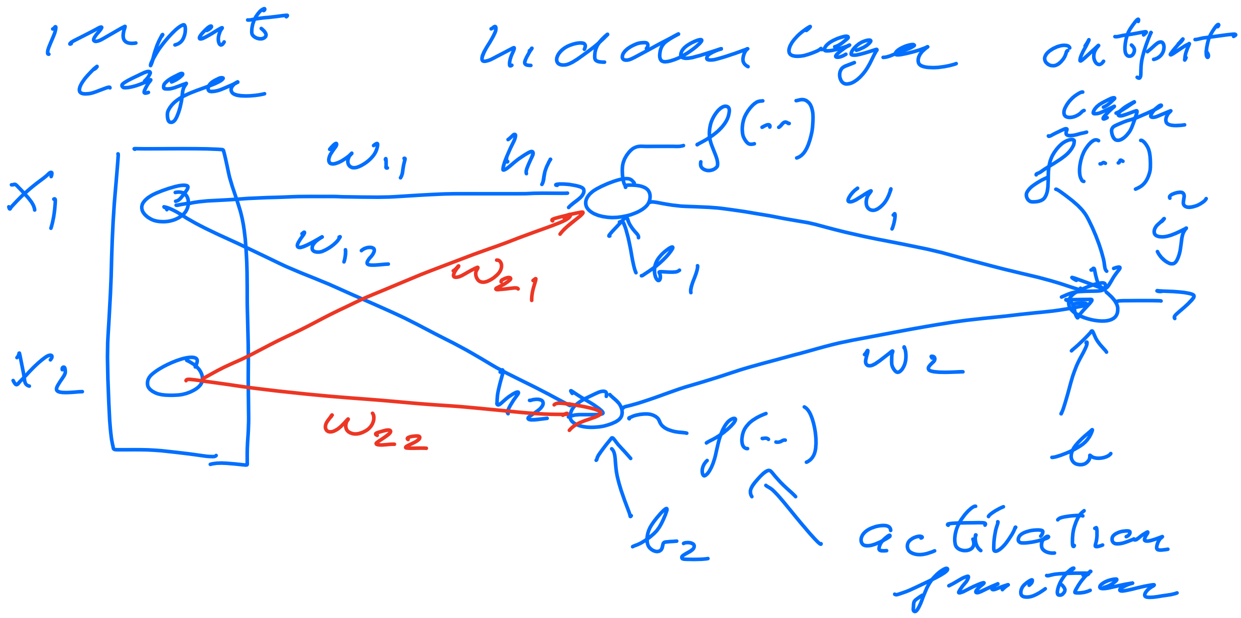
else $\tilde{y} = 1$

XOR-gate

$$y = [0, 1, 1, 0]^T$$

$$\tilde{y} = \left[\tfrac{1}{2}\ \tfrac{1}{2}\ \tfrac{1}{2}\ \tfrac{1}{2}\right]^T$$

With one hidden layer and two hidden neurons/nodes, we can reproduce the xor output



Hidden layer

- ~~# hidden layers~~
  - # neurons in a layer
- W : w_{12}, w_{11}, w_{21} w_{22}
  b : b_1 b_2
- activation functions
- output layer
  - # output nodes
  - activation functions

Universal approximation

theorem:

Given a function $F \in C[0,1]^d$ and a parameter $\varepsilon > 0$, then there is a one layer (= 1 hidden layer) neural network $f(x; W, b)$

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

with $W \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^m$ for which

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

for all $x \in [0,1]^d$ (unit cube)

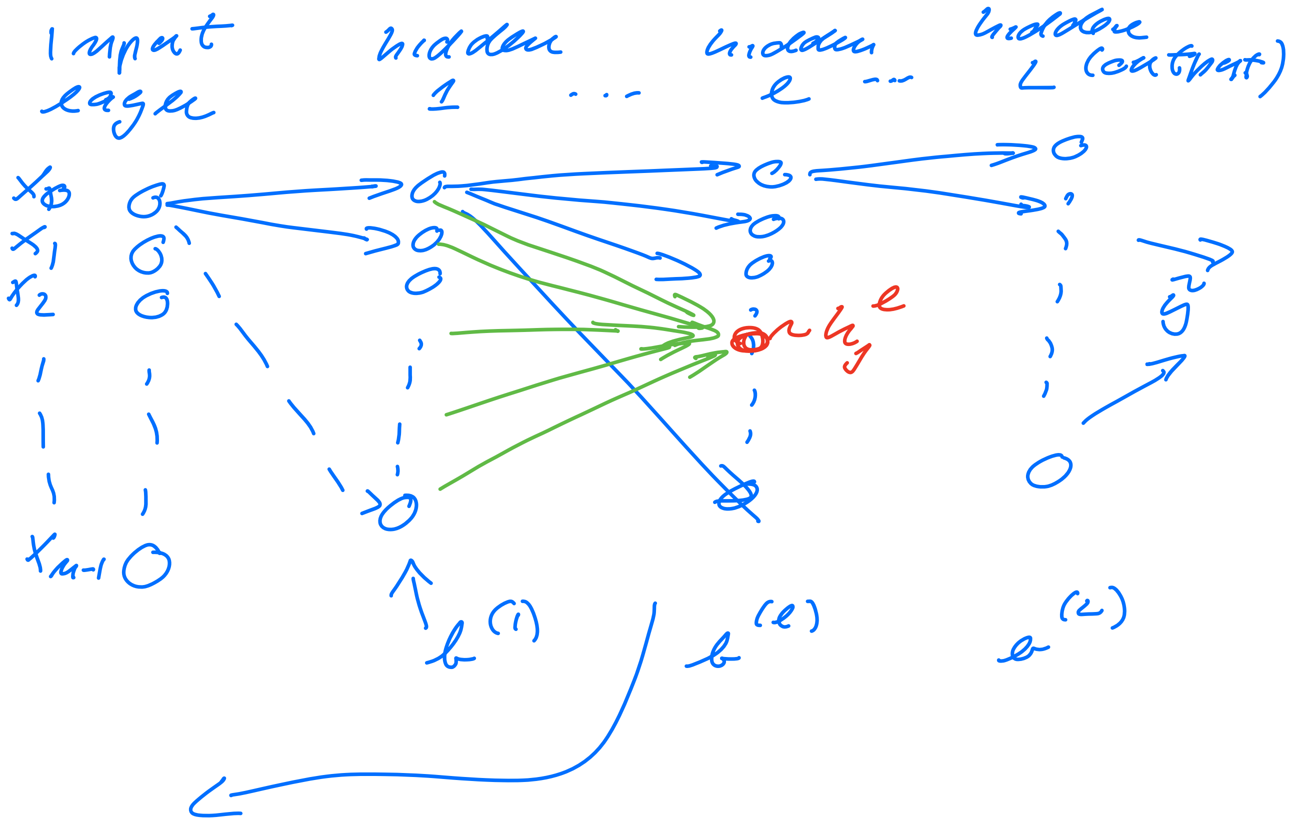
practicalities: How do we find θ?

- Back propagation algorithm (chain rule)
  $\Rightarrow$ Gradients to be used in optimization of W and b

Back propagation algorithm

- Regression

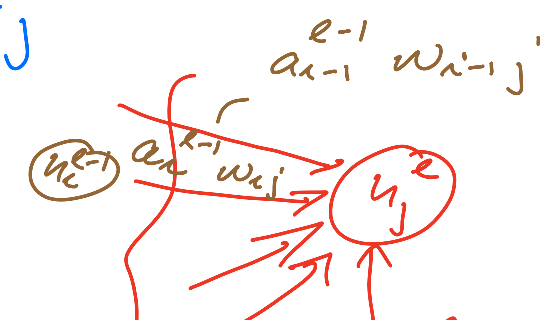
$$C(\Theta) = \frac{1}{2}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2$$

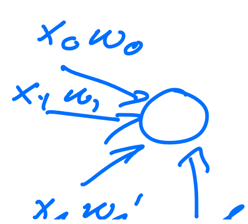


hidden layer - l -

input to node j :

z_j^l





$$z_j^\ell = \sum_{i=0}^{M_{\ell-1}-1} w_{ij}^\ell a_i^{\ell-1} + b_j^\ell \qquad z = \sum_i w_i x_i + b$$

outputs from layer $\ell-1$ and node i

inputs to layer $-\ell-$

$$z^\ell = (W^\ell)^T a^{\ell-1} + b$$

This is modulated by an activation $f(z_j^\ell)$ and produces an output

$$a_j^\ell = f(z_j^\ell)$$

$$a^\ell = f(z^\ell)$$

with two hidden layers

First hidden layer

$$z_j^1 = \sum_i w_{ij}^1 x_i + b_j^1$$

input layer

output $a_j^1 = f(z_j^1)$

$a^1 = f(z^1)$

2nd hidden = output layer
(last = output)

$$z_j^2 = \sum_i w_{ij}^2 a_i^1 + b_j^2$$

output = $\tilde{y} = a^2 = f(z^2)$

$$= f(f(z^1))$$

3-layers : $f(f(f(z^1)))$

we have defined

$$a_j^l = f(z_j^l)$$

typical function

$$f(z_j^l) = \frac{1}{1+\exp(-z_j^l)}$$

we need to define :

∂z_1^l $\quad l-1$

$$\frac{\quad}{\partial w_{ij}^{l}} = a_i$$

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

$$\frac{\partial a_j^{l}}{\partial z_j^{l}} = \frac{\partial f(z_j^{l})}{\partial z_j^{l}}$$

$$= f(z_j^{l})(1 - f(z_j^{l}))$$

Specialize to output layer $l = L$

my output is $\tilde{y}_i = a_i^L$

$$\frac{\partial C(\theta)}{\partial w_{jk}^{L}} = \frac{\partial}{\partial w_{jk}^{L}}\left[\frac{1}{2}\sum_i (\tilde{y}_i - y_i)^2\right]$$

$$= (a_j^L - y_j)\frac{\partial a_j^L}{\partial w_{jk}^{L}}$$

∂w_{jk}

chain-rule !

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

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

$$\frac{\partial C(\theta)}{\partial w_{jk}^L} = (a_j^L - y_j)(1 - a_j^L) \times a_k^{L-1} a_j^L$$

Define :

$$\delta_j^L = \underbrace{a_j^L (1 - a_j^L)}_{f'(z_j^L)} \underset{\frac{\partial C}{\partial a_j^L}}{(a_j^L - y_j)}$$

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

∂w_{jk}

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

$$\delta_j^L = \frac{\partial C}{\partial b_j^L}$$

$$\boxed{\begin{aligned} \frac{\partial C}{\partial w_{jk}^L} &= \delta_j^L a_k^{L-1} \\ \delta_j^L &= f'(z_j^L) \frac{\partial C}{\partial a_j^L} \\ \delta_j^L &= \frac{\partial C}{\partial b_j^L} \end{aligned}}$$

all for final layer.

Need to Back propagate

$L \to l$

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

$l+1$

$$\sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

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

$$z_j^{l+1} = \sum_i w_{ij}^{l+1} \underbrace{a_i^l}_{f(z_i^l)} + b_j^{l+1}$$

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

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

Now can set up gradients for each layer:

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

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

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

$$= b_j^{\ell} - \eta \delta_j^{\ell}$$