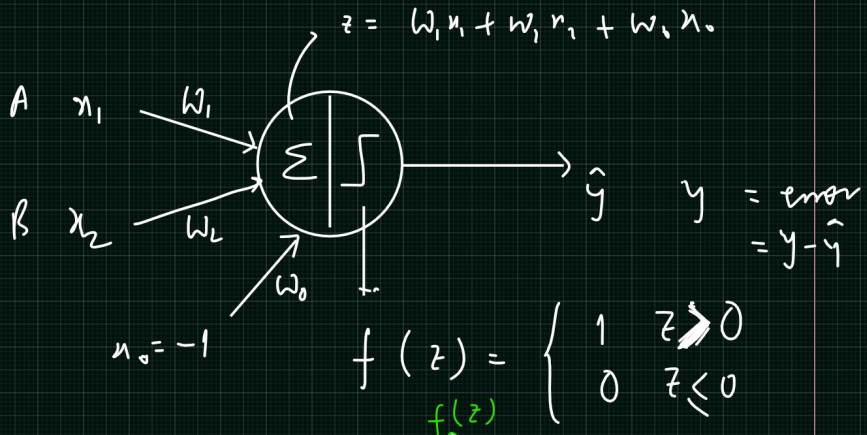
Perceptron implimentation :-

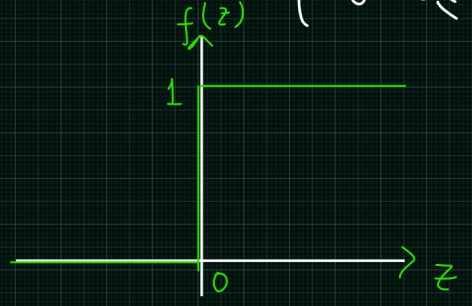
Dataset

OR Gate

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



$$f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$



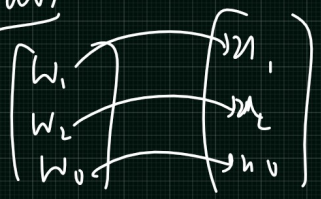
Forward pass :-

$$z = w_1 x_1 + w_2 x_2 + w_0 x_0$$

$$= \begin{bmatrix} w_1 & w_2 & w_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_0 \end{bmatrix}$$

$$z = W \cdot x$$

np.dot



When you are training :-

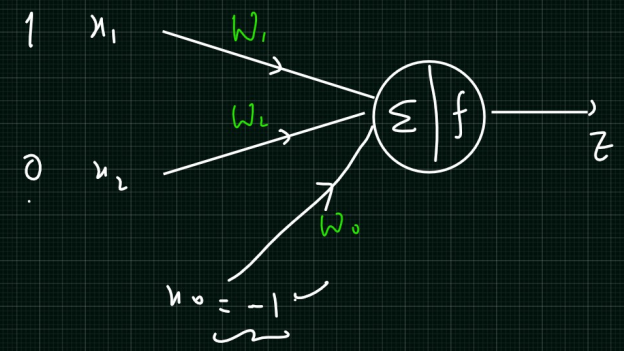
forward + backward pass

prediction

forward pass.

prediction

A	B	Y
1	0	

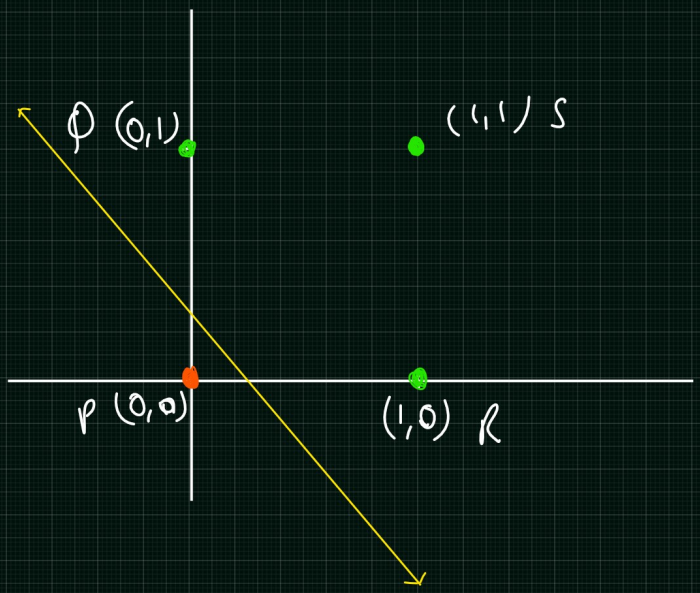


XOR

0	0	0
0	1	1
1	0	1
1	1	0

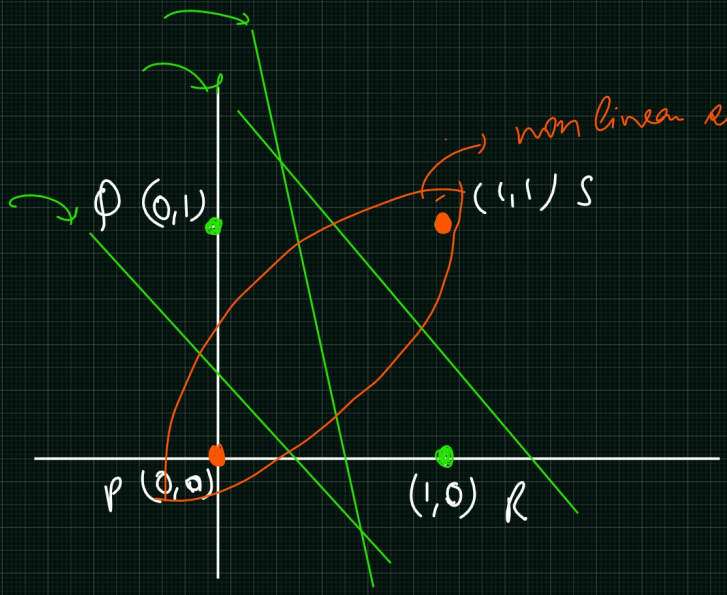
OR Gate

P →	0	0	0
Q →	0	1	1
R →	1	0	1
S →	1	1	1



XOR Gate

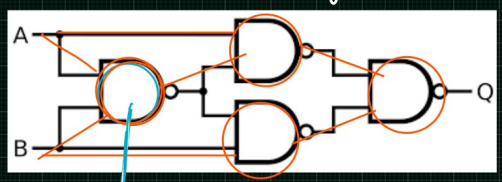
P →	0	0
Q →	0	1
R →	1	0
S →	1	1



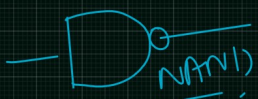
NAND (linear data)

0	0	1
0	1	0
1	0	0
1	1	0

XOR Gate using NAND Gate



NAND



$A \cdot B$ = AND operation

$\overline{A \cdot B}$ = NAND operation

Antn → neuron

Drawback of Perceptron :-

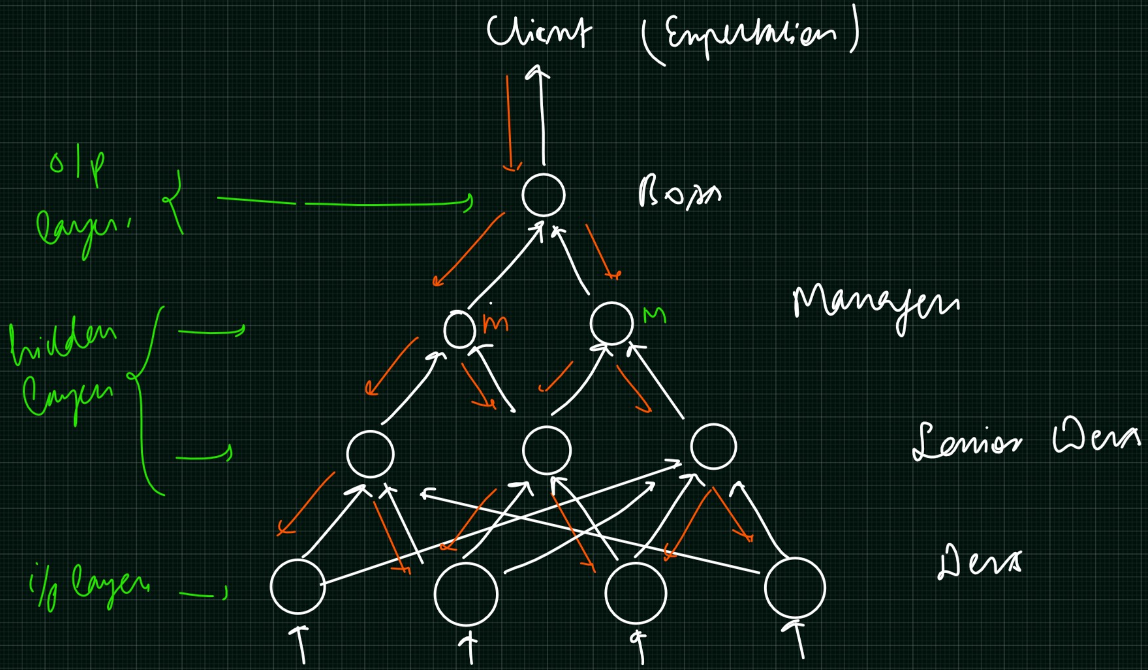
→ It cannot solve non linear data.

Solution :-

→ You can stack layers of perceptron

⇓ aka

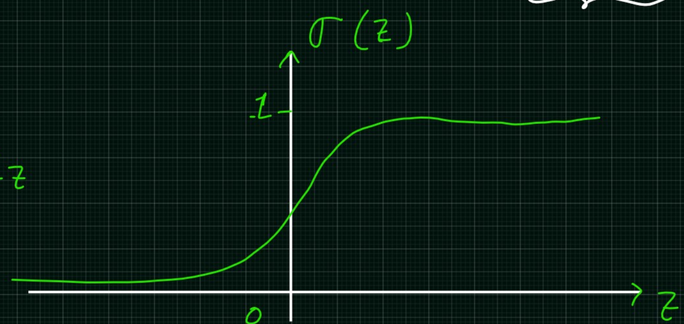
Multilayer Perceptron (MLP)



Activation function.

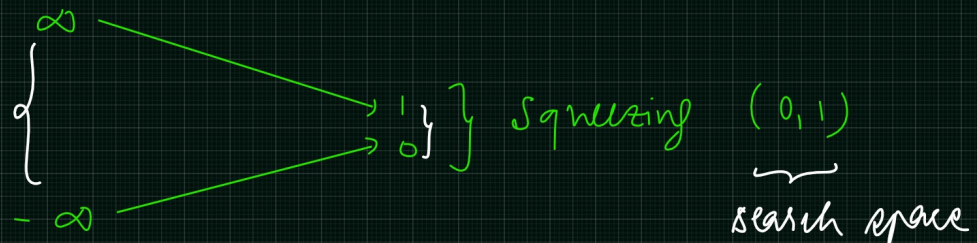
initially people have used sigmoid

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



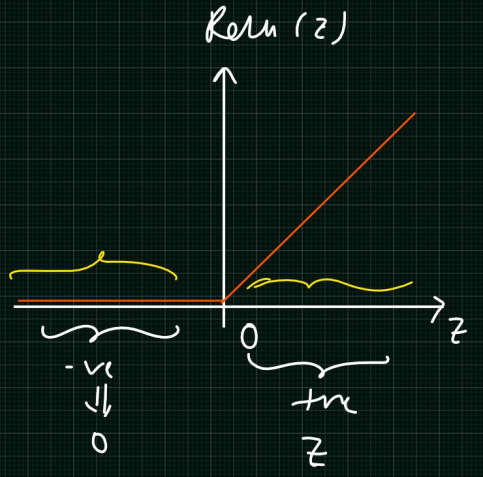
domain of σ (input range of values) $\rightarrow (-\infty, \infty) \Rightarrow z \in (-\infty, \infty)$

Range (o/p range) $(0, 1) \Rightarrow \sigma(z) \in (0, 1)$ $[0, 1]$



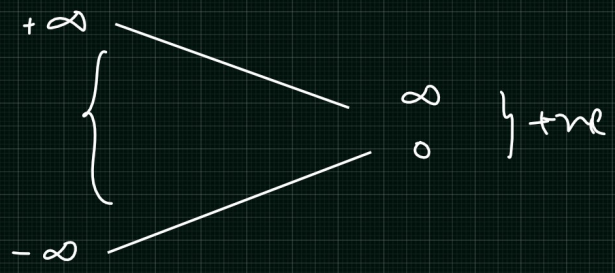
ReLU => Rectified linear unit

$$\text{ReLU}(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$



domain: $z \in (-\infty, \infty)$

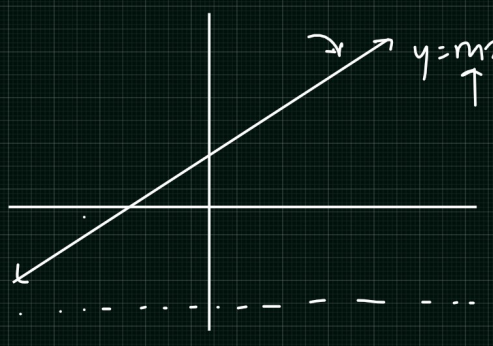
Range: $\text{ReLU}(z) \in [0, \infty)$

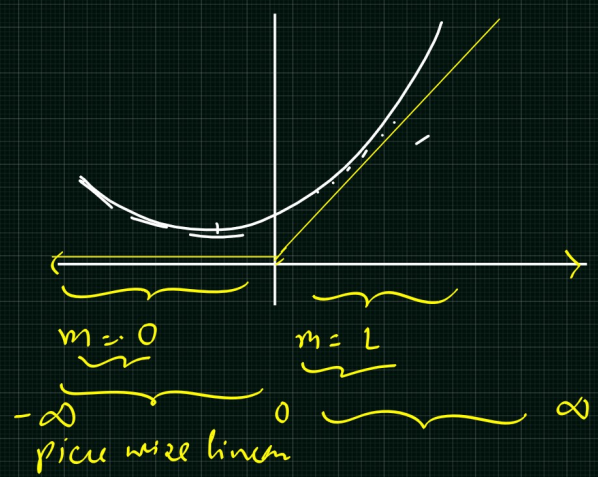


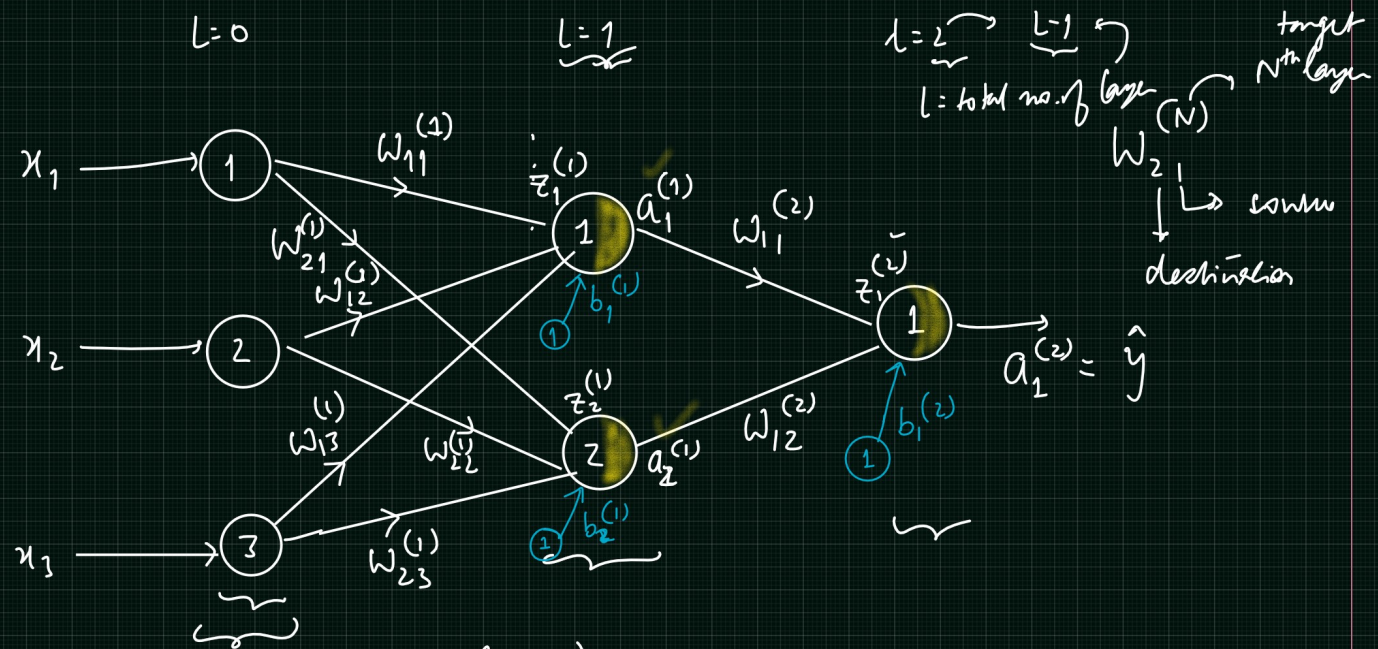
Rectification → filtering out -ve values

Linear → graph whose slope is constant

non Linear → whose slope changes







features			
x_1	x_2	x_3	y
0	0	2	
3	4	6	
⋮	⋮	⋮	

at layer 1 :-

$$\textcircled{1}\begin{cases} z_1^{(1)} = W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)} \\ a_1^{(1)} = \sigma(z_1^{(1)}) \end{cases}$$

$$\textcircled{2}\begin{cases} z_2^{(1)} = W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)} \\ a_2^{(1)} = \sigma(z_2^{(1)}) \end{cases}$$

final layer (o/p layer)

$$z_1^{(2)} = W_{11}^{(2)} a_1^{(1)} + W_{12}^{(2)} a_2^{(1)} + b_1^{(2)}$$

$$a_1^{(2)} = \sigma(z_1^{(2)}) \Rightarrow \hat{y} \text{ predicted}$$

Calculating loss ⇒ error fn / loss fn mse = $f(y, \hat{y})$ =

Weight update rule (General)

$$\begin{cases} w = w + \Delta w \\ \Delta w = -\eta \dfrac{\partial e}{\partial w} \end{cases} \qquad \begin{cases} b = b + \Delta b \\ \Delta b = -\eta \dfrac{\partial e}{\partial b} \end{cases}$$

$\frac{dy}{dx}$ change in y wrt small change in x

$$\begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix}_{2\times 3} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}_{3\times 1} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}_{2\times 1} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \end{bmatrix}_{2\times 1}$$

at layer 1

$$\begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \end{bmatrix}_{2\times 1} = \begin{bmatrix} \sigma(z_1^{(1)}) \\ \sigma(z_2^{(1)}) \end{bmatrix}_{2\times 1} \leftarrow \text{act}$$

at layer 2

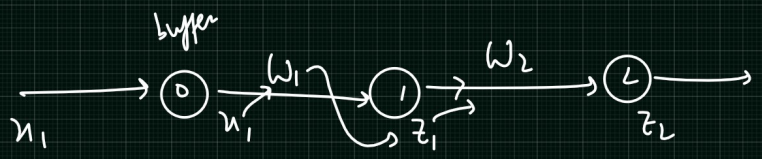
$$\overrightarrow{\begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \end{bmatrix}}_{1\times 2} \begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \end{bmatrix}_{2\times 1} + \begin{bmatrix} b_1^{(2)} \end{bmatrix}_{1\times 1} = \begin{bmatrix} z_1^{(2)} \end{bmatrix}_{1\times 1}$$

$$\underbrace{\begin{bmatrix} w_{11}^{(2)} \\ w_{12}^{(2)} \end{bmatrix}^T}_{1\times 2} \begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \end{bmatrix}_{2\times 1}$$

$W^T \cdot X$

$$\underbrace{\hat{y}}_{} = \begin{bmatrix} a_1^{(2)} \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{(2)}) \end{bmatrix} \leftarrow \text{act}$$

$$W \cdot X + B = Z \xrightarrow{\sigma} \hat{Y}$$



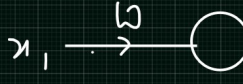
$$z_1 = \underbrace{w_1 x_1}_{} \quad -(1)$$

$$z_2 = w_2 z_1$$

$$z_2 = w_2 \cdot w_1 \cdot x_1 \quad \text{from eqn}^n (1)$$

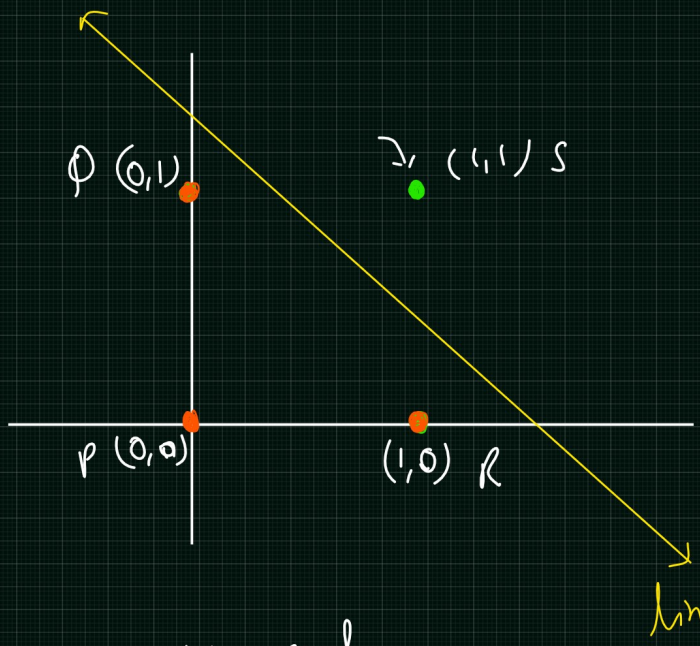
$$z_2 = \underbrace{w \cdot x_1}_{} \quad \} \text{ linear}$$

$y = mx$



AND Gate

P →	0	0	0
Q →	0	1	0
R →	1	0	0
S →	1	1	1

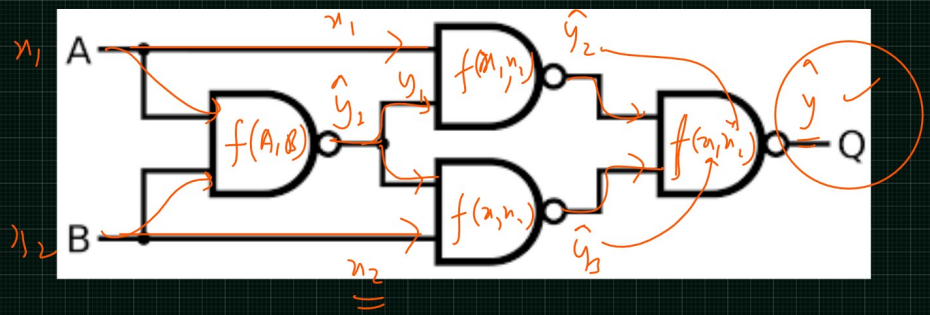


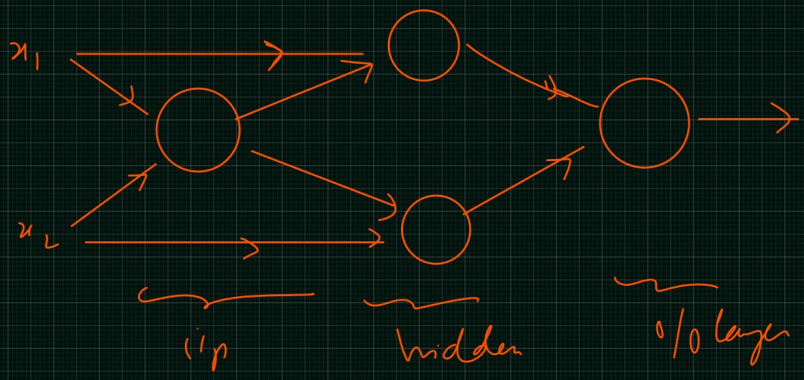
NAND Gate → model_nand

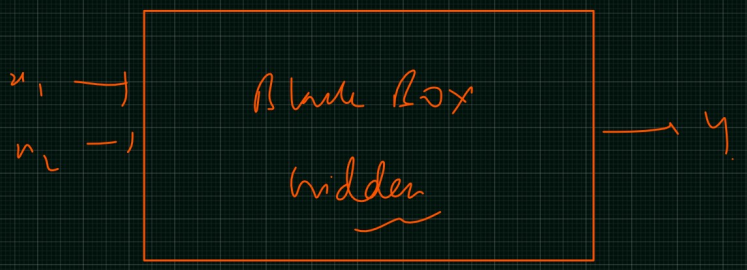
	x_1	x_2	
P →	0	0	1
Q →	0	1	1
R →	1	0	1
S →	1	1	0

$\hat{y} = model\_nand.predict(X)$ → (x_1, x_2)

$\hat{y} = f(x_1, x_2)$






x_1
x_2
Black Box
hidden
y