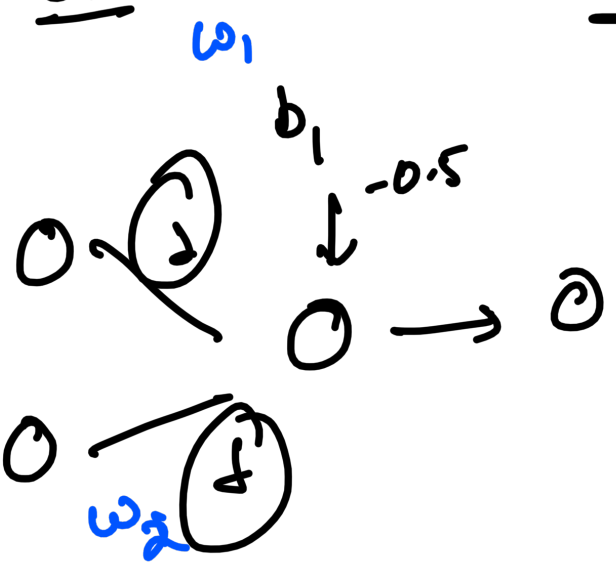
Weights fixed ✗

Learning → Single neuron

OR



AND

$b_1 = 1.5$

Learning

weights = learned

$$w_i = w_i + \Delta w$$

$$\Delta w_i = \eta (y - \hat{y}) x_i$$

w_i = weights

Δw = amount of change required

η = learning rate

y = true class

$\hat{y}$ = predicted class

→ x_i = input sample

η → very small number

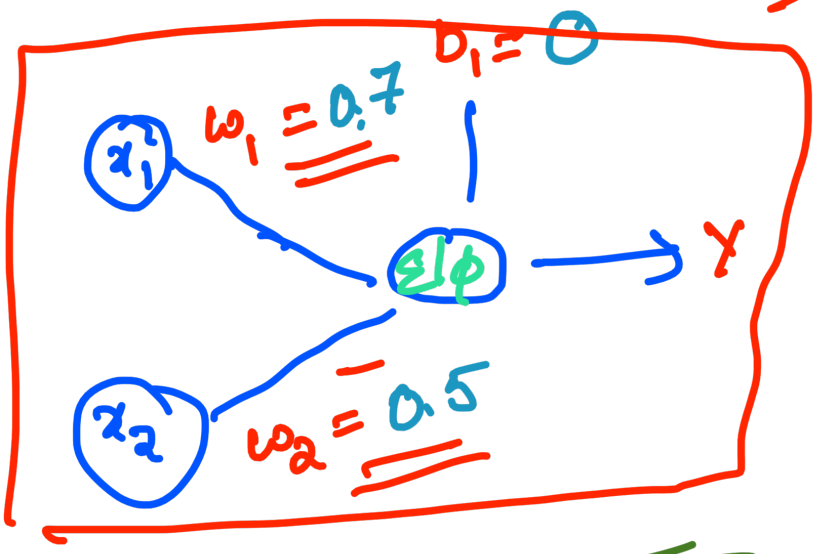
1×10^{-3} 1×10^{-4}

dom

[0-1]

OR Neural

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	1



$$\phi(z) = \begin{cases} 0 & w \cdot x + b < 0 \\ 1 & w \cdot x + b \geq 0 \end{cases}$$

Threesshold

$x_1 = 0 \quad x_2 = 0$

$w_1 = 0.7 \quad b_1 = 0$

$w_2 = 0.5$

$\eta = 0.7$

$$\phi(\Sigma) = \phi(x_1 \cdot w_1 + x_2 \cdot w_2 + b_1)$$

$$= \phi(0 \cdot 0.7 + 0 \cdot 0.5 + 0)$$

$$= \phi(0)$$

$$= 0$$

$$w_i = w_i + \Delta w \leftarrow 0$$

$$\Delta w_i = \eta (0 - 0)\, x_i$$

$$= 0$$

$$w_i = w_i$$

$x_1 = 0 \quad x_2 = 1 \quad y = 1 \quad \hat{y} = 0$

$\phi(\Sigma) = \phi(x_1 \cdot \omega_1 + x_2 \cdot \omega_2 + b_1)$

$= \phi(0 \times 0.7 + 1 \times 0.5 + 0)$

$= \phi(-0.1)$

$= 0$

$\omega_i = \omega_i + \Delta\omega \qquad \omega_1 = \omega_1$

$\Delta\omega_1 = 0.7(1-0)\,0$

$= 0$

$\Delta\omega_2 = 0.7(1-0)\,1$

$= 0.7$

$w_2 = w_2 + \Delta w_2$

$= -0.5 + 0.7$

$= 0.2$

Adjustment of 0.0.1 data

$w_1 = 0.7$

$w_2 = 0.2$

} New weight

we need to ensure that it is also working for previous samples

$w_1 = 0.7$

$w_2 = 0.2$

$\eta = 0.7$

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

$\Rightarrow$

$b_1 = 0$

$x_1 = 0 \quad x_2 = 0$

$\phi(\Sigma) = (0.7 \times 0 + 0.2 \times 0 + 0)$

$= \phi(0)$

$= 0$

$x_1 = 0 \quad x_2 = 1$

$\phi(\Sigma) = \phi(0.0.7 + 0.2 \times 1 + 0)$

$= \phi(0.2) = 1$

$x_1 = 1 \quad x_2 = 0$

$$\phi(z) = \phi(1 \times 0.7 + 0 \times 0.2 + 0)$$

$$= \phi(0.7)$$

$$= 1 \checkmark$$

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \eta (y - \hat{y}) x_i$$

weight update Rule

logic

C, C++, Python

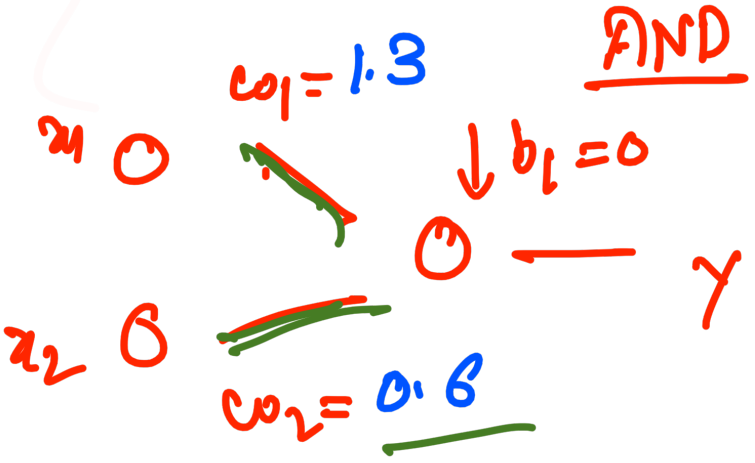
10 lines

lib

import

1 line

.sum( )



x_1	x_2	Y
0	0	0 ✓
0	1	0
1	0	0
1	1	1

y.

$$\phi = \begin{cases} 0 & \omega \cdot x + b \le 0 \\ 1 & \omega \cdot x + b > 0 \end{cases}$$

$\eta = 0.7$

$x_1 = 0 \quad x_2 = 0$

$\phi(\varepsilon) = \phi(0) = 0$ ✓

$x_1 = 0 \quad x_2 = 1$

$\phi(\varepsilon) = \phi(0 \times 1.3 + 1 \times 0.6 + 0) =$

$= \phi(0.6) = 1 \quad \hat{y}$ ✓

$$\Delta\omega_i = \eta (y - \hat{y}) x_i$$

$\Delta\omega_1 = 0.7 (0 - 1) 0 = 0$

$\Delta\omega_2 = 0.7 (0 - 1) 1 = -0.7$

$\omega_1 = \omega_1 + \Delta\omega_1$

$= 1.3 + 0$

$= 1.3$

$\omega_2 = \omega_2 + \Delta\omega_2$

$= 0.6 - 0.7$

$= -0.1$

x_1	x_2	Y

$w_1 = 1.3$

$w_2 = -0.1$

0	0	0
0	1	0
→ 1	0	0
1	1	1

$x_1 = 0 \quad x_2 = 1$

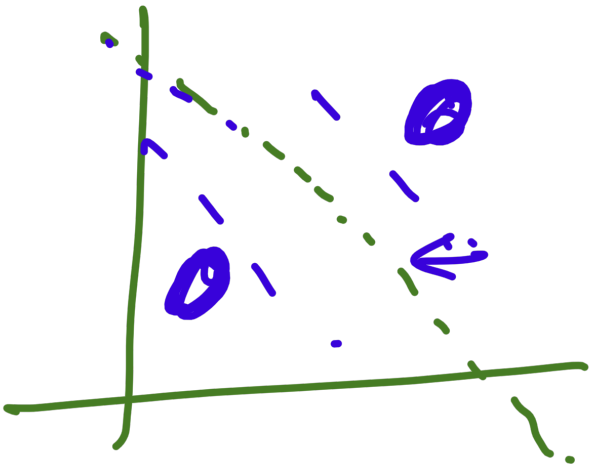
$\phi(\varepsilon) = \phi(0 \times 1.3 + 1 \times -0.1 + 0)$

$= \phi(-0.1)$

$= 0$

Δw

ML vs Deep Neural Networks



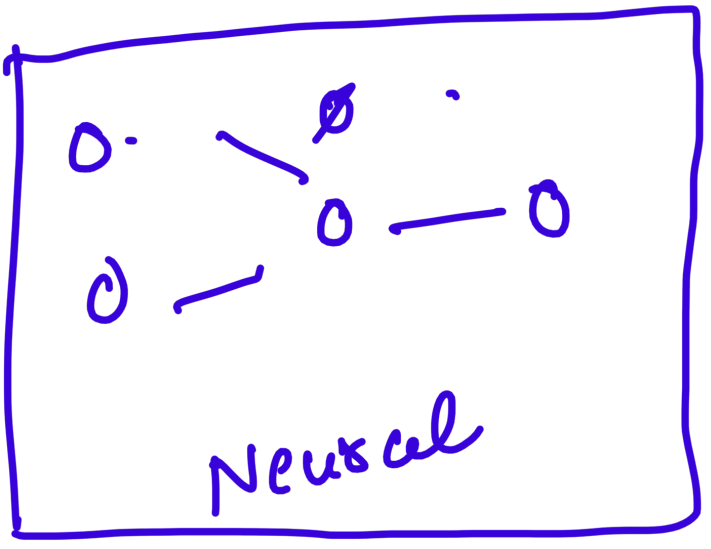
Kmeans

centroid

x

y

x<y

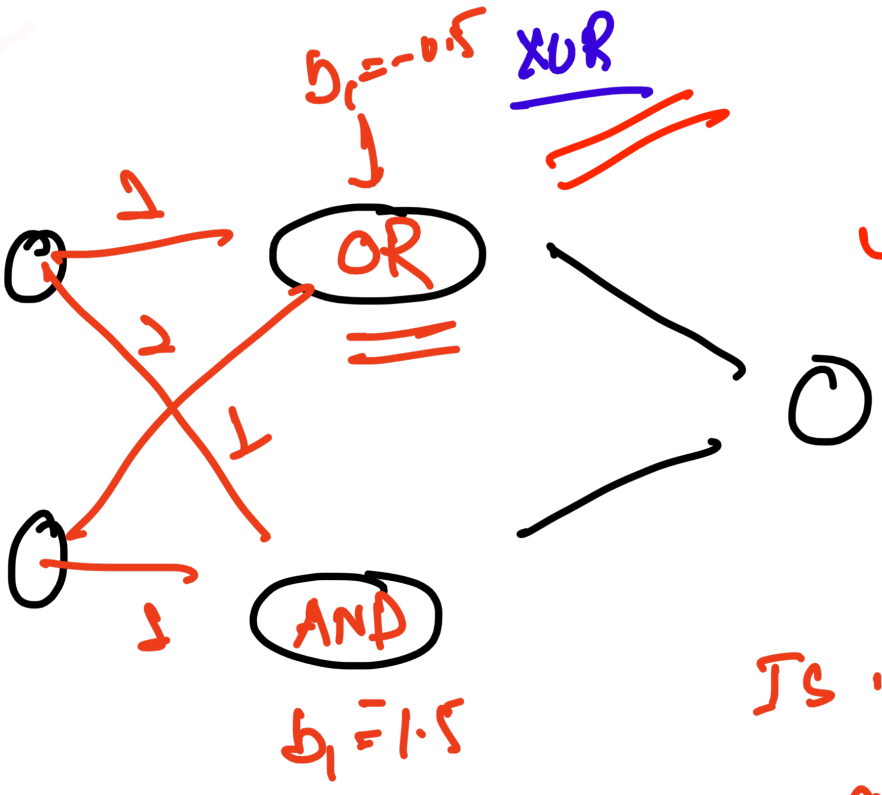


Neural

Brain

XOR

Non-linear



Is it the only way to solve it XOR?

Architecture.

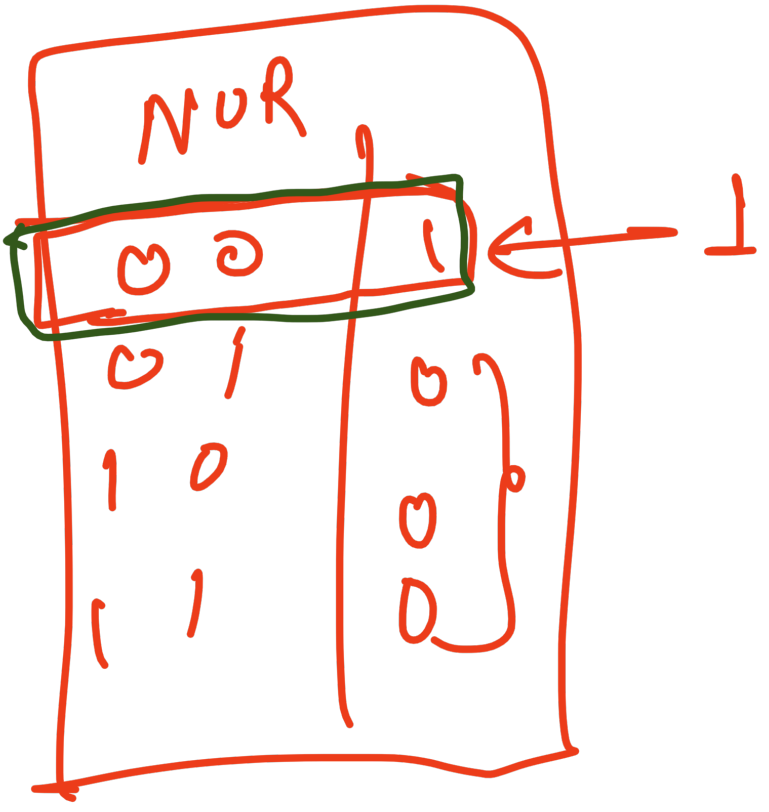
NOR

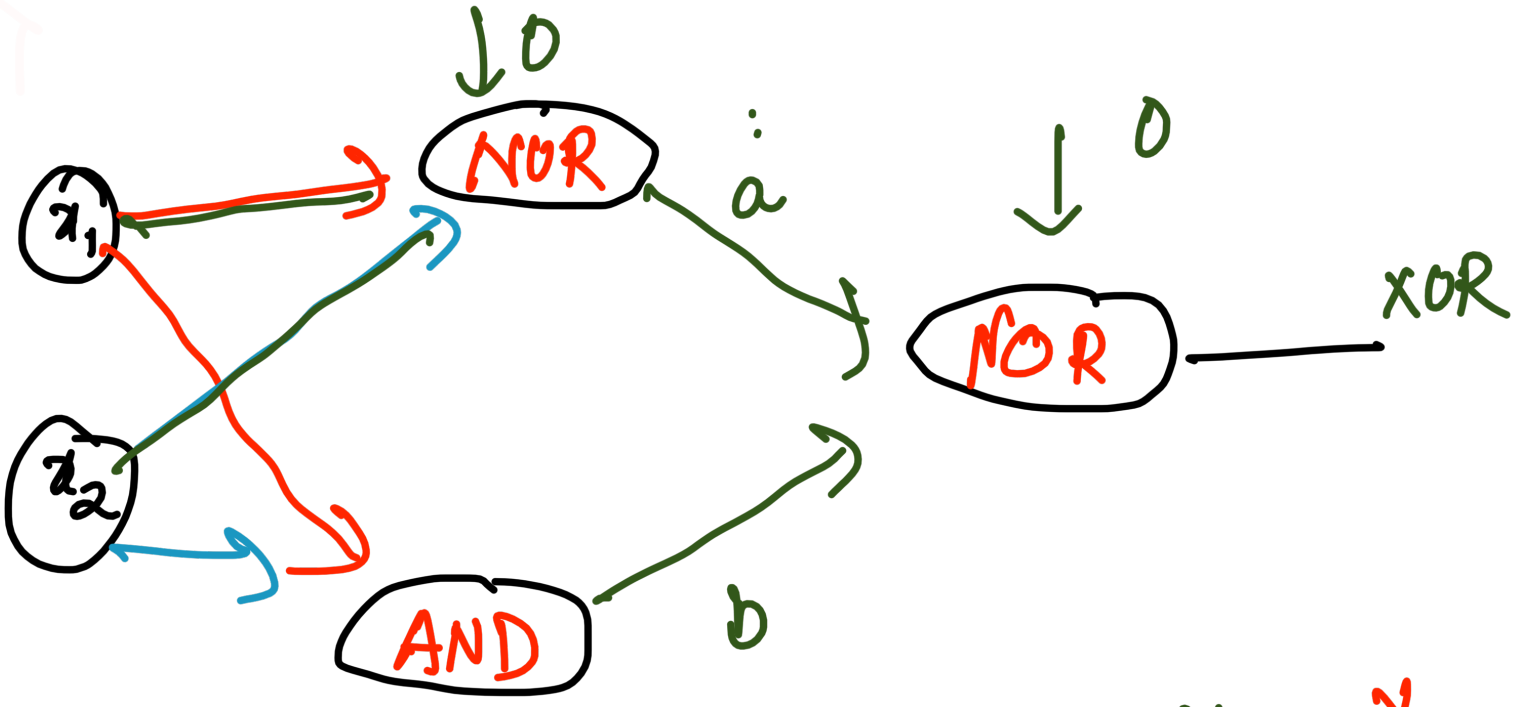
AND

OR + NOT

0	0	0	$\bar{y}$
0	1	1	
1	0	1	
1	1	1	

OR





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

		a	b	
X_1	X_2	NOR	AND	NOR(a,b)
0	0	1	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0

XOR