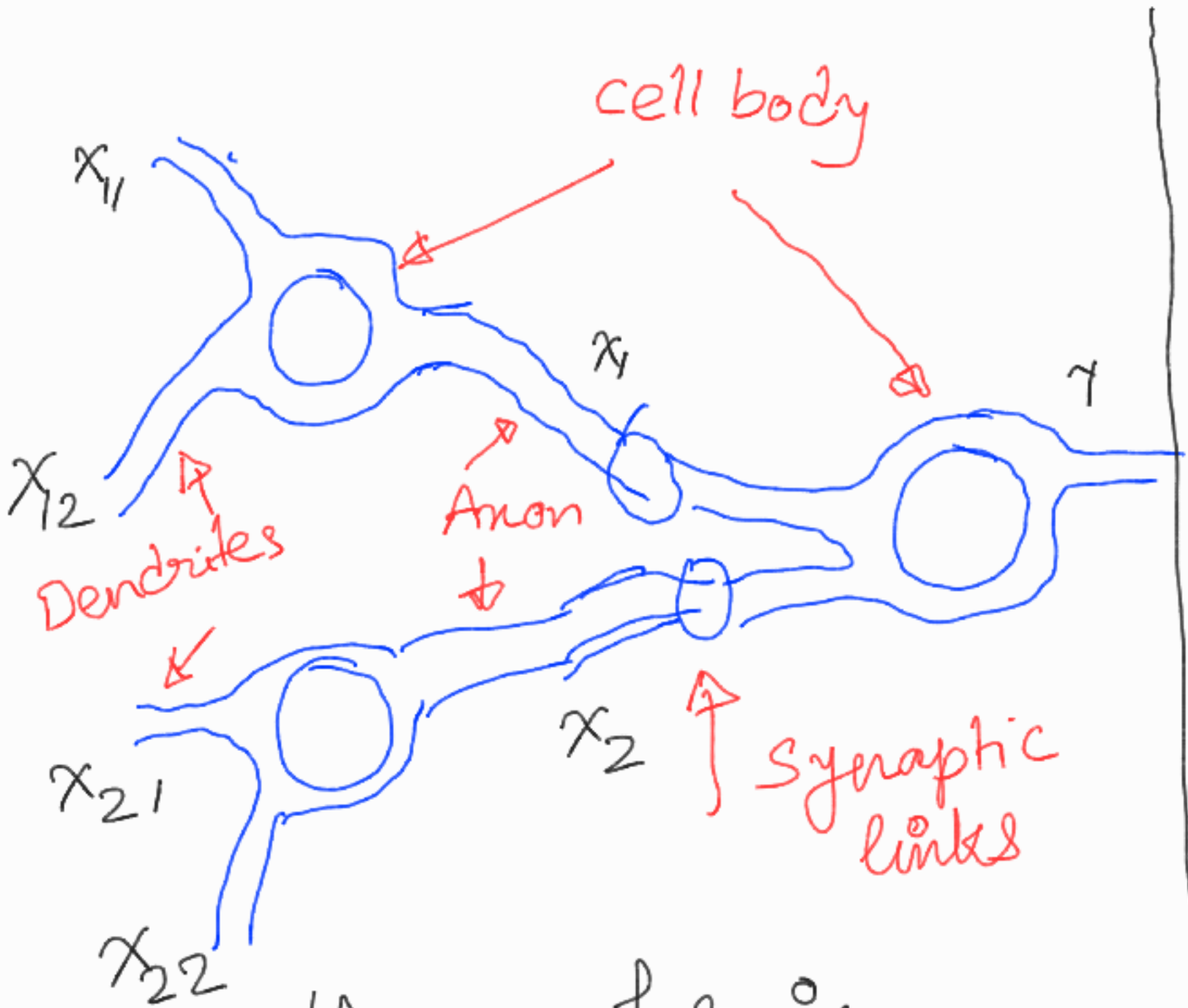
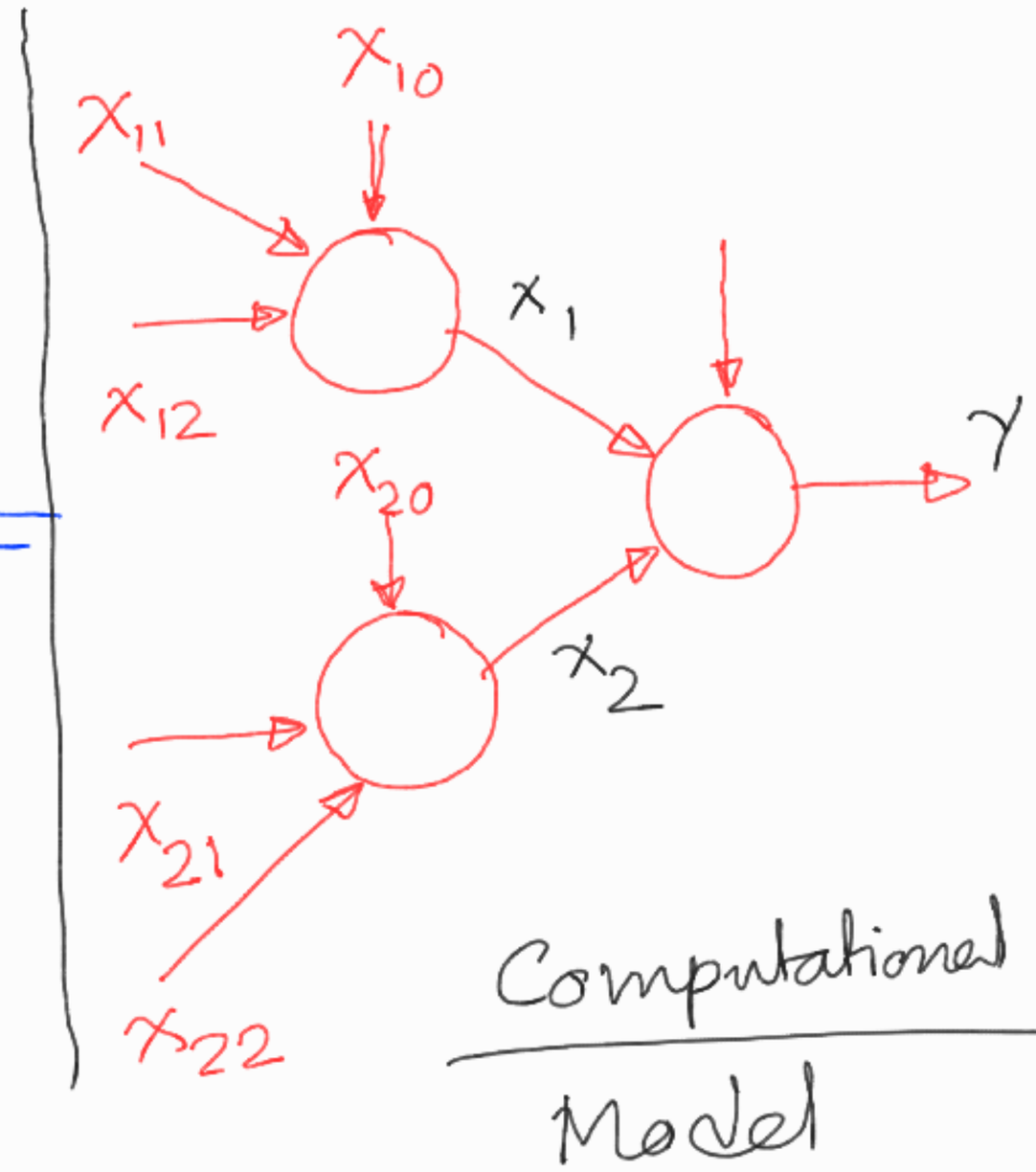
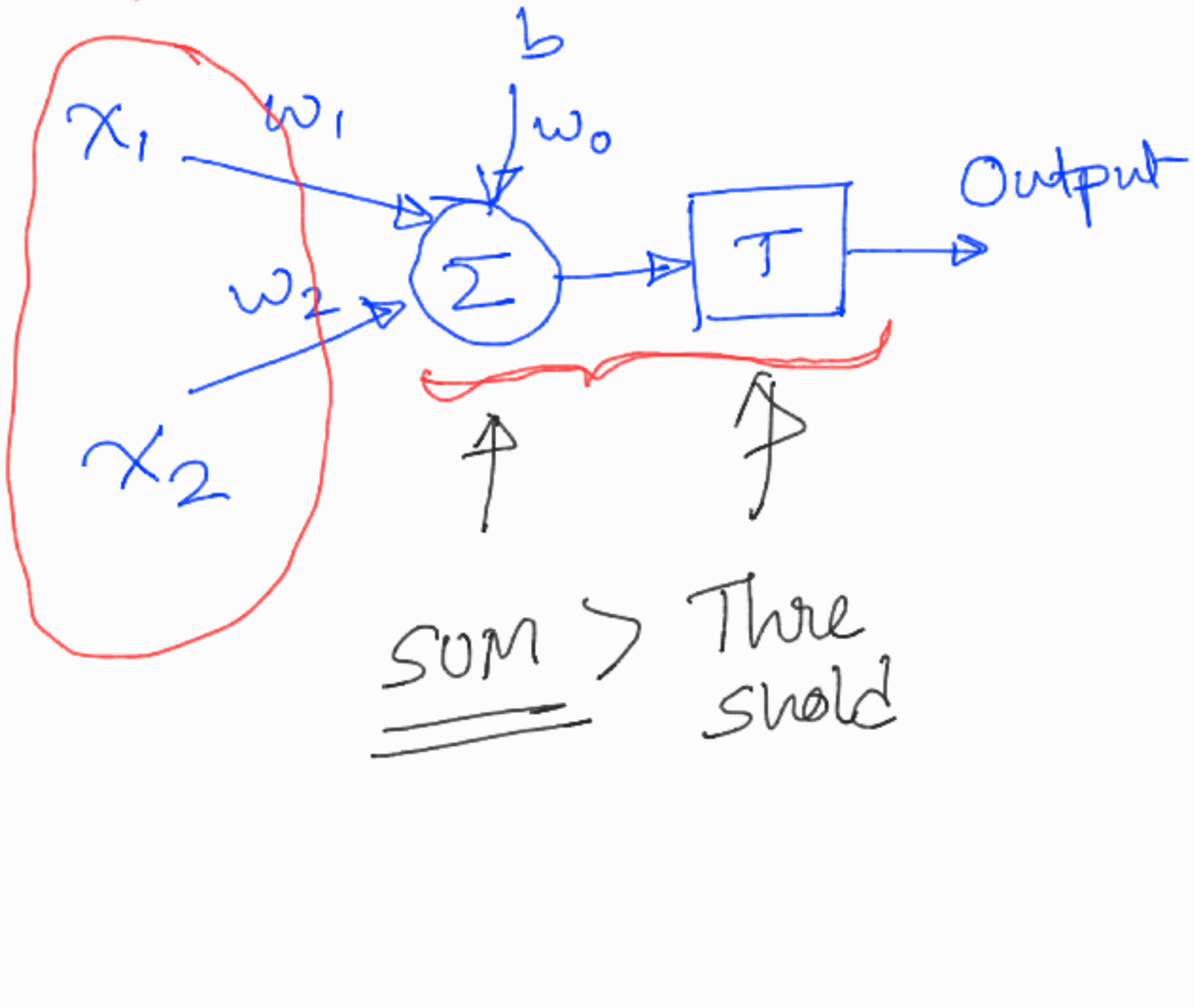
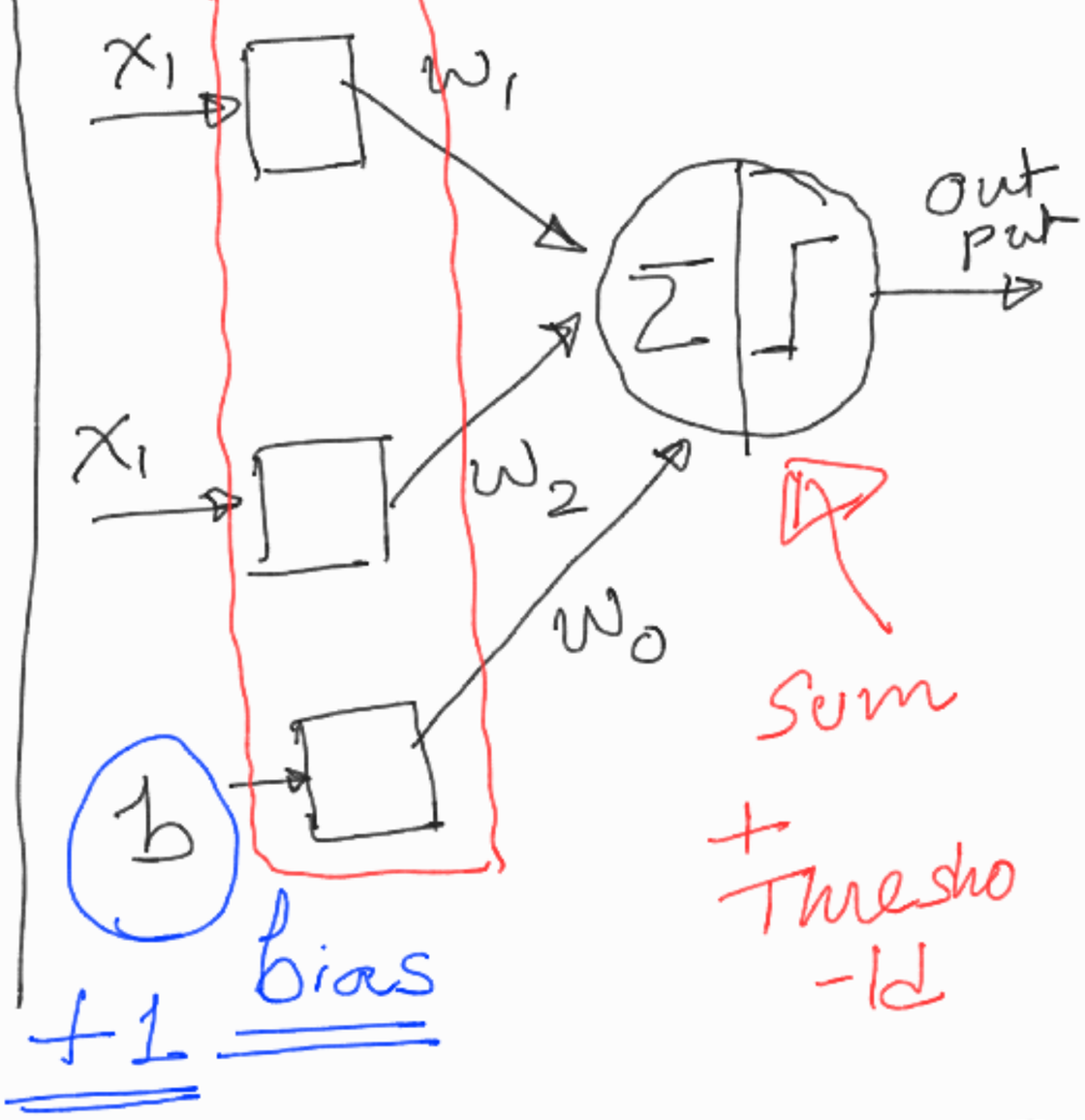


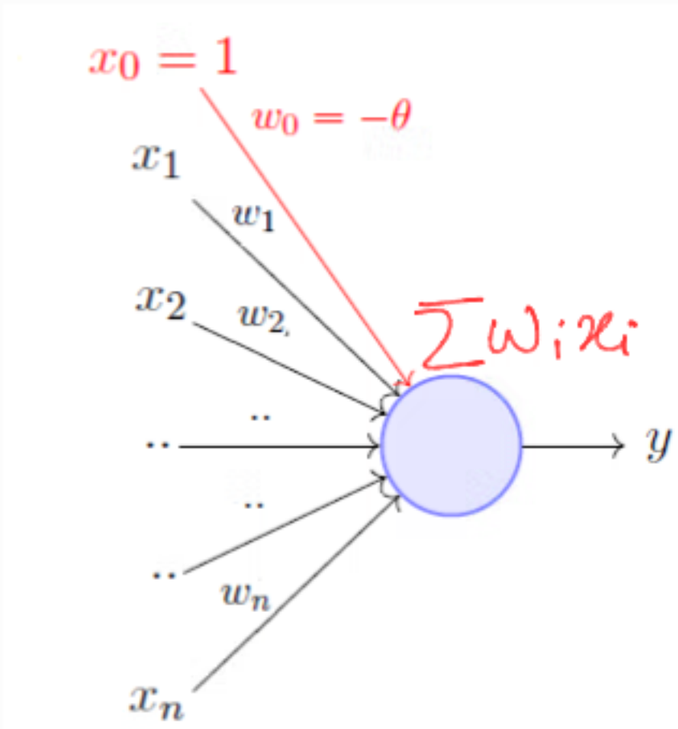


Perceptron (Equivalent representations)





Perceptron Learning :



A more accepted convention,

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$

Bias

Linearly Separable : we can draw a line to separate two classes.

OR Gate using a single perceptron

x_1	x_2	OR	Computed output (Actual)
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$

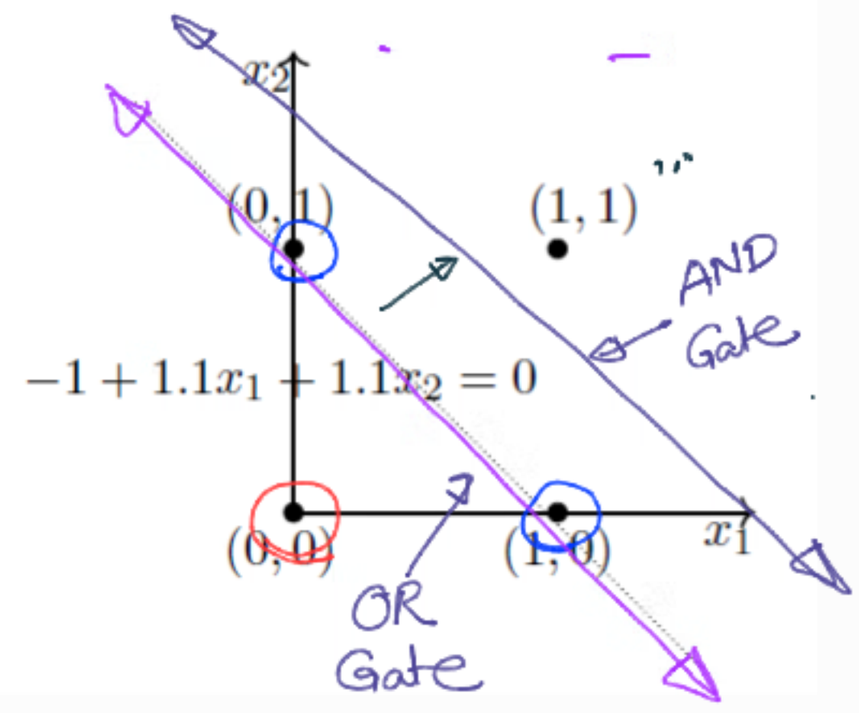
Class Label - Desired

≥ 0

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 > -w_0$$

One possible solution is

$w_0 = -1, \; w_1 = 1.1, \; w_2 = 1.1$



(Image source: Towards data science)

Algorithm: Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize **w** randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P^{1}$ *and* $\mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N^{0}$ *and* $\mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

Start with initial values for parameter

Calculate errors

Computed W are parameters.

Random(0,1)

- Learning happens by changing W

for $x \in P$: $wx \geq 0$ — Class Label we know/want

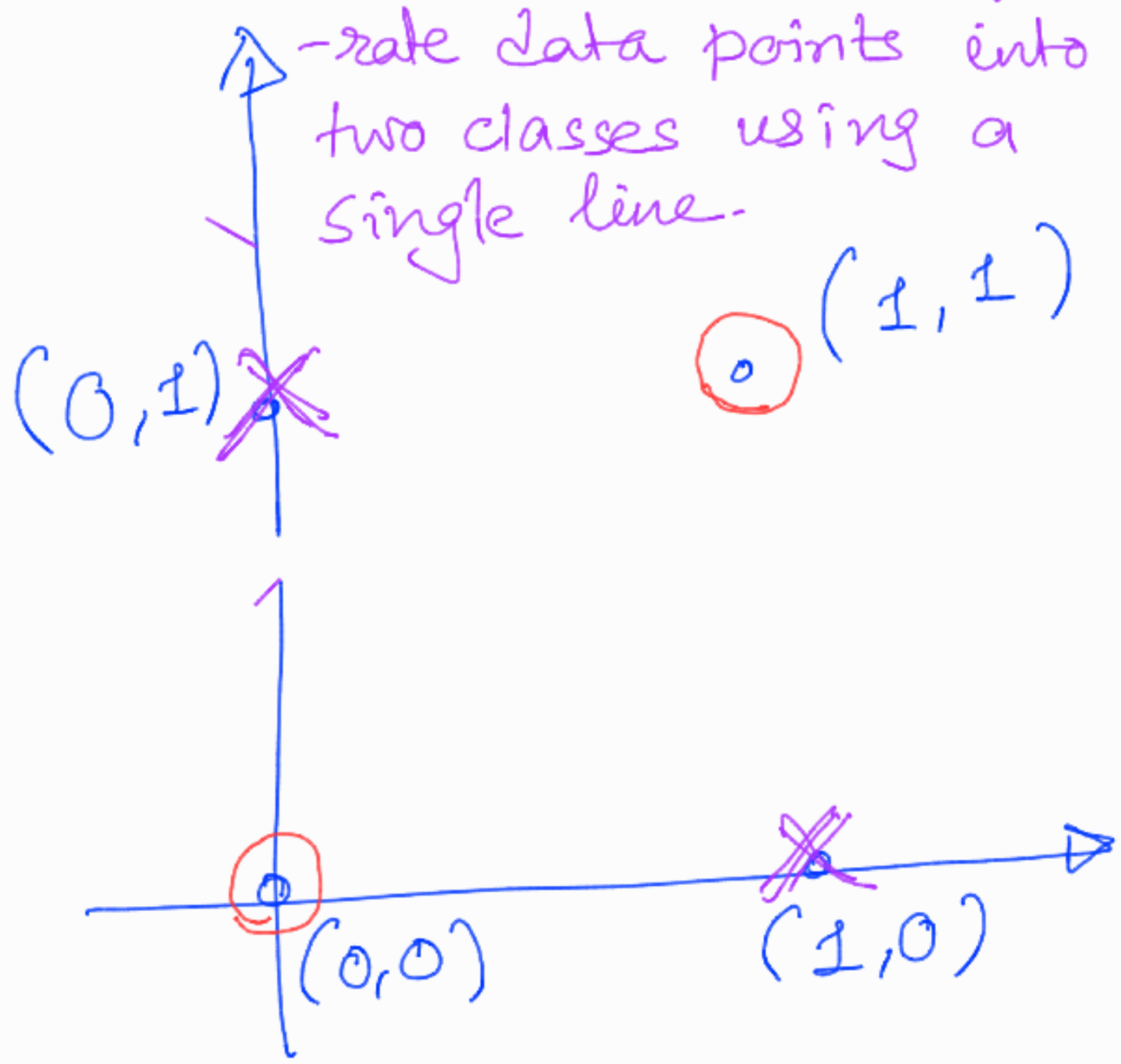
for $x \in N$: $wx < 0$

(Image source: Towards data science)

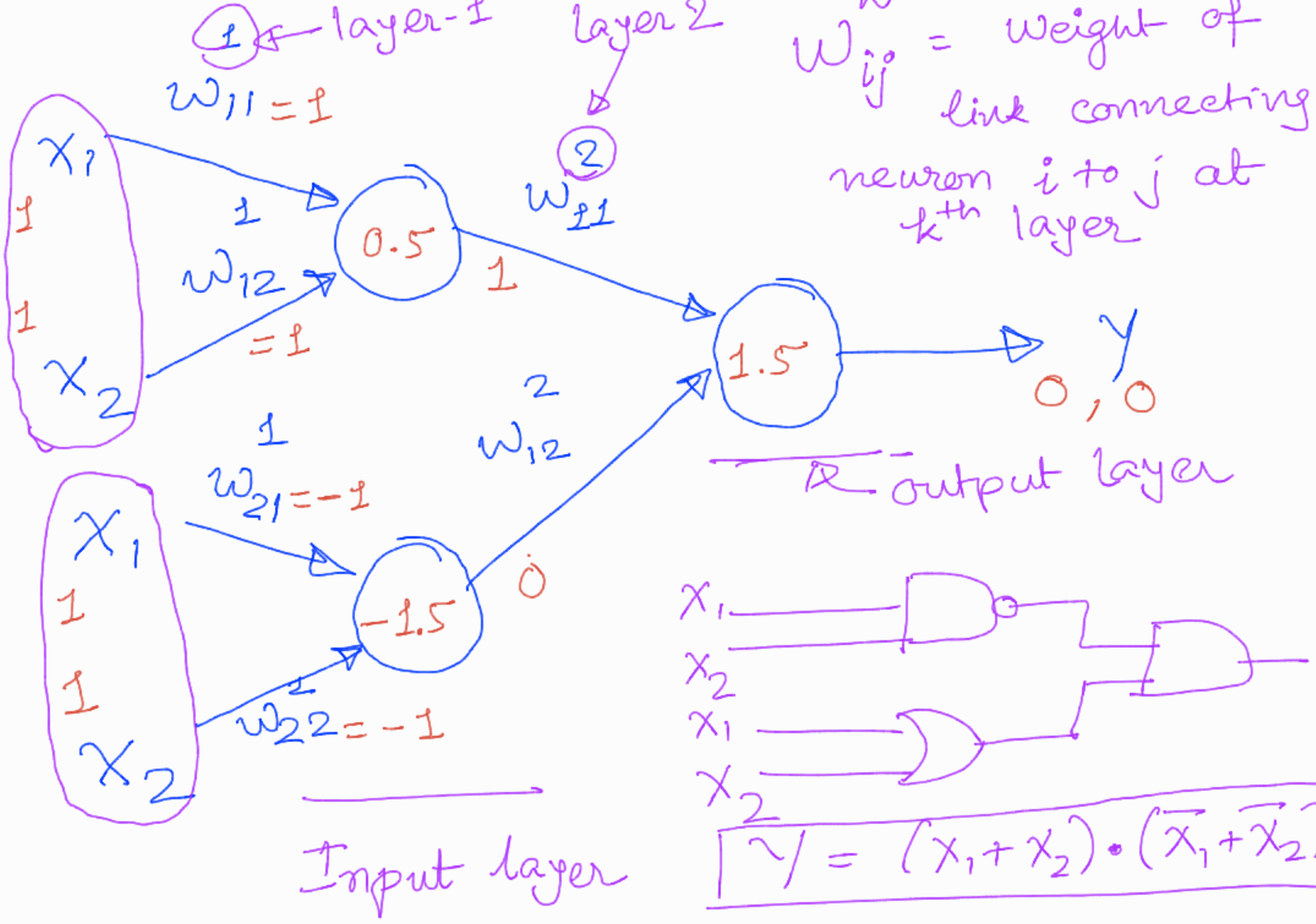
XOR : Linearly 'NOT' seperable

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

we can not separate data points into two classes using a single line.

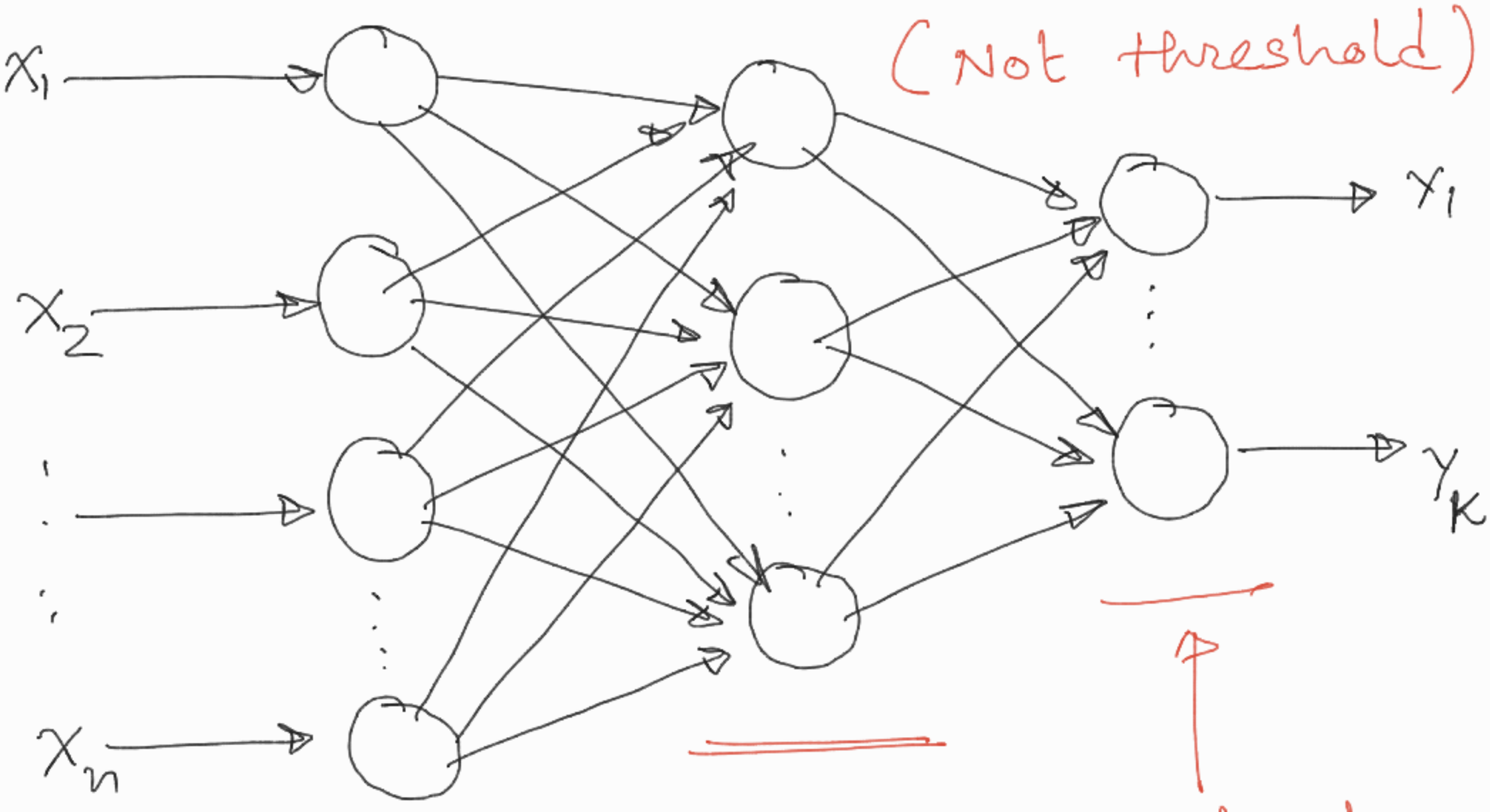


Can be solved with more than one neuron !!



Multilayered Feed forward neuran network

activation function: Non Linear (Not threshold)



Hidden Layer + Non linear activation

↳ can model any non linear function

$y = f(x)$