Artificial Intelligence

Supervised Learning

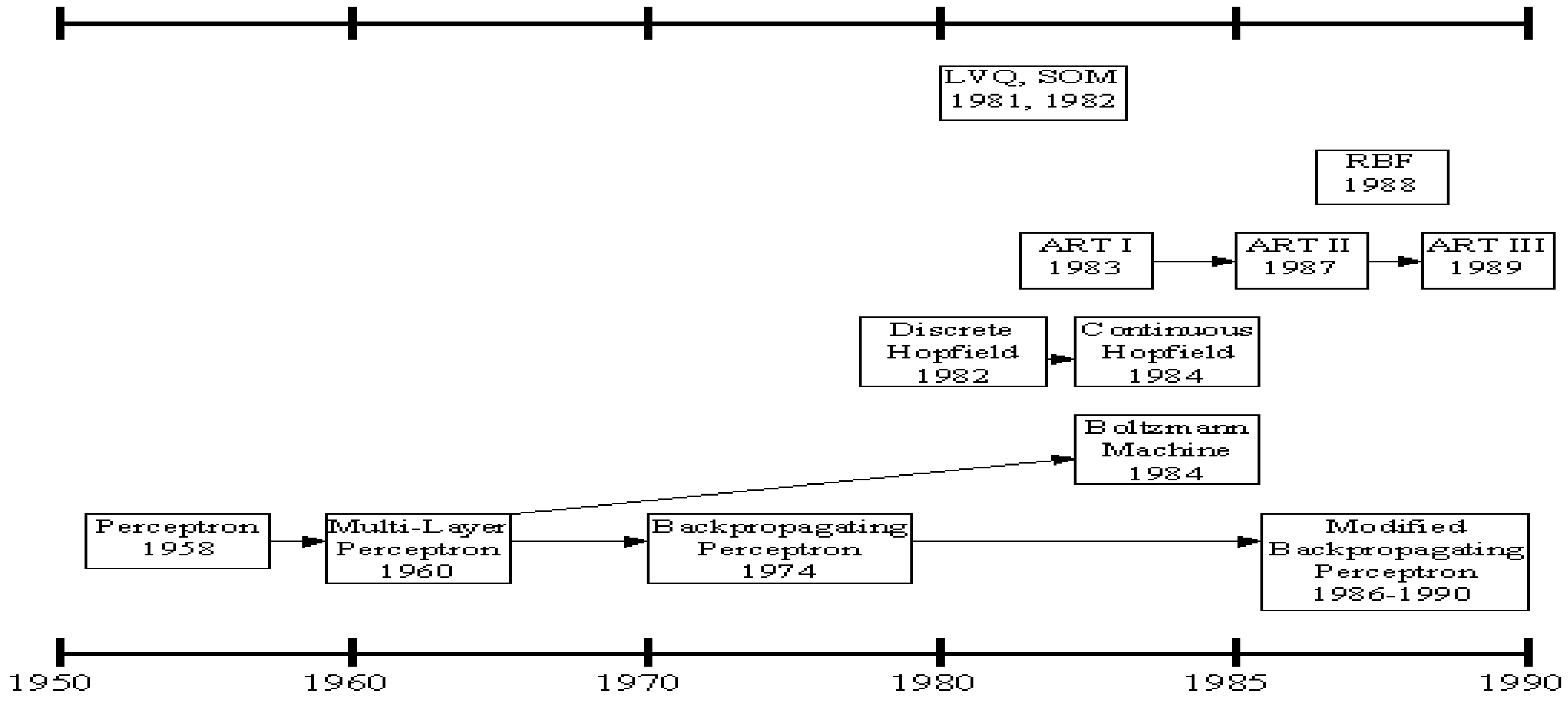
Following Slides contain material from Bing Liu, Mackassy, Stuart Russell, Berrin Yanikoglu

Artificial Neural Network

- Computational models inspired by the human brain:
  - Massively parallel, distributed system, made up of simple processing units (neurons)
  - Synaptic connection strengths among neurons are used to store the acquired knowledge.
  - Knowledge is acquired by the network from its environment through a learning process

History of ANN

- History of the ANNs stems from the 1940s, the decade of the first electronic computer.
- In 1957 Rosenblatt introduced the first concrete neural model, the perceptron.
- Rosenblatt also took part in constructing the first successful neurocomputer, the Mark I Perceptron.
- After this, the development of ANNs has proceeded as described in *Figure*.

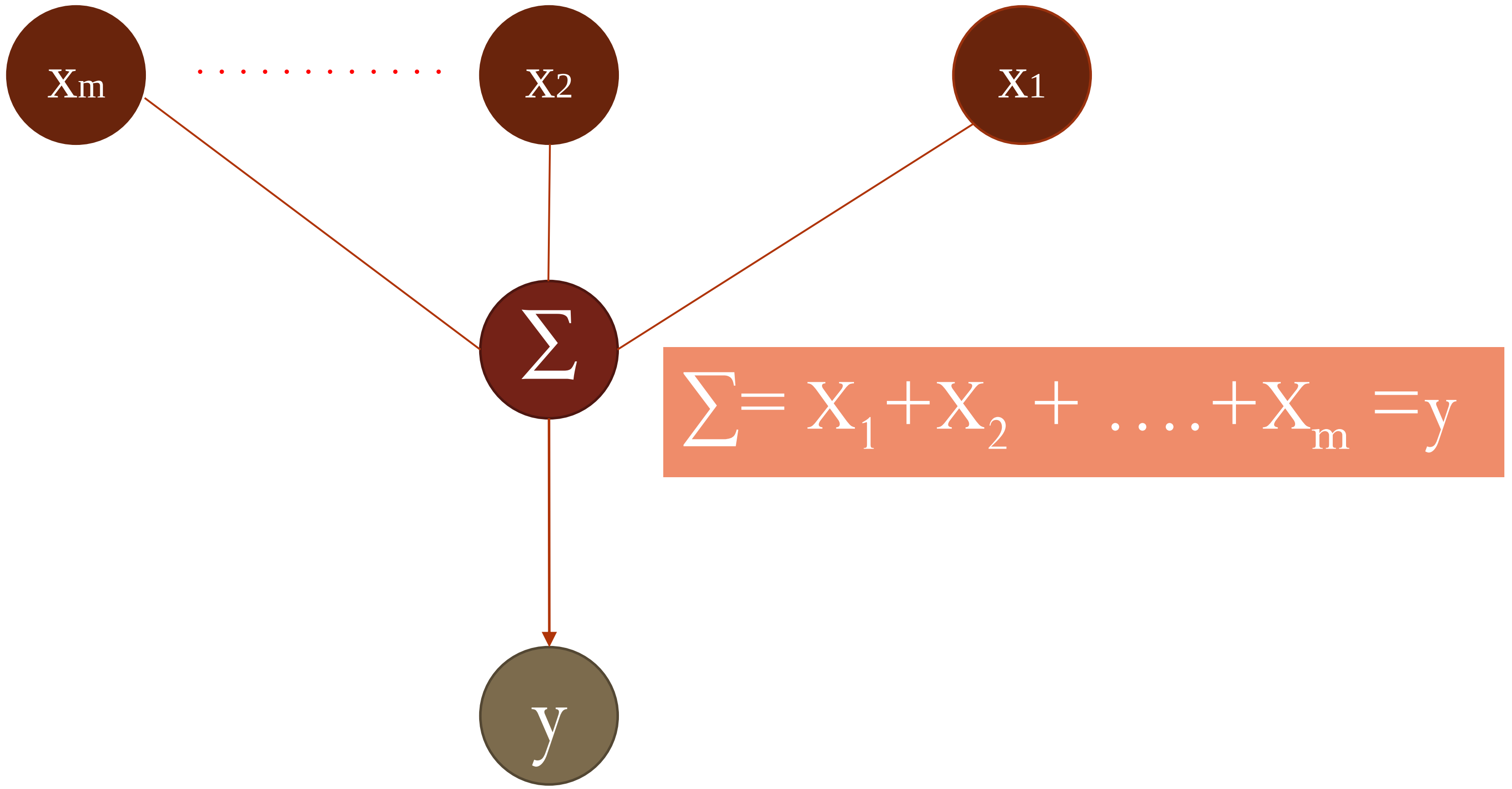


ANN

Input

Processing

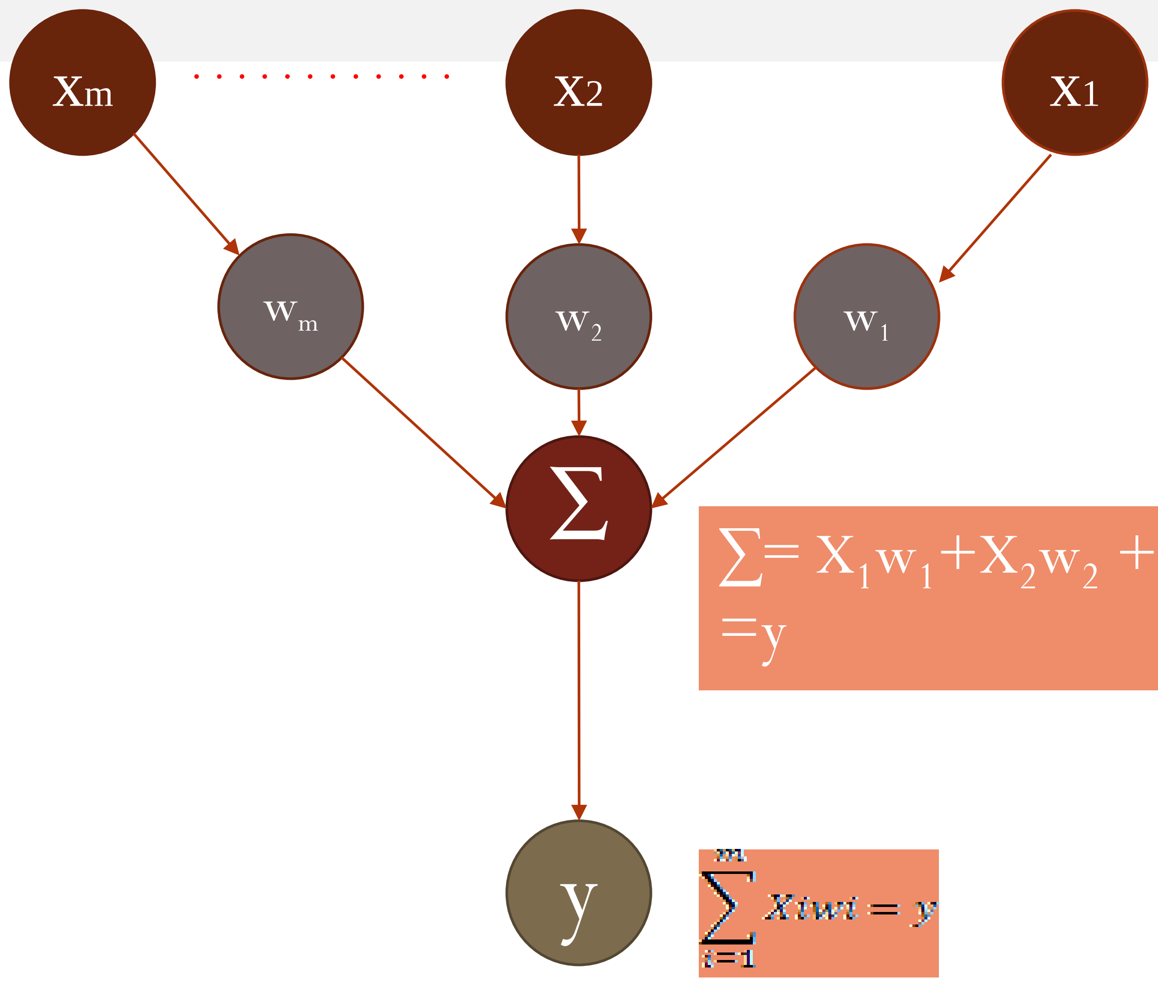
Output



$$\sum = X_1 + X_2 + \ldots + X_m = y$$

ANN

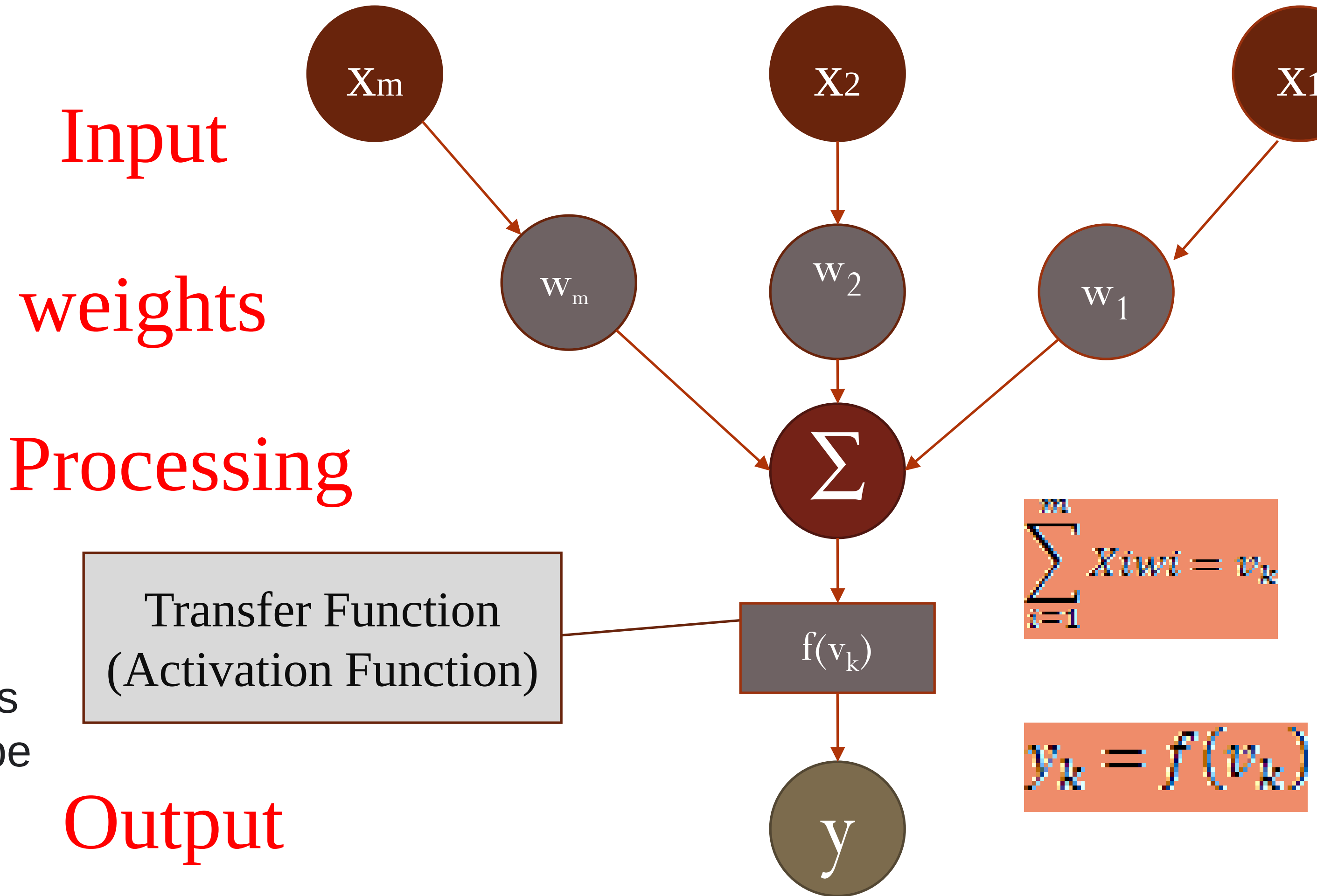
Not all inputs are equal



$$\sum = X_1w_1 + X_2w_2 + \ldots + X_mw_m = y$$

$$\sum_{i=1}^{m} Xiwi = y$$

ANN



$$\sum_{i=1}^{m} Xiwi = v_k$$

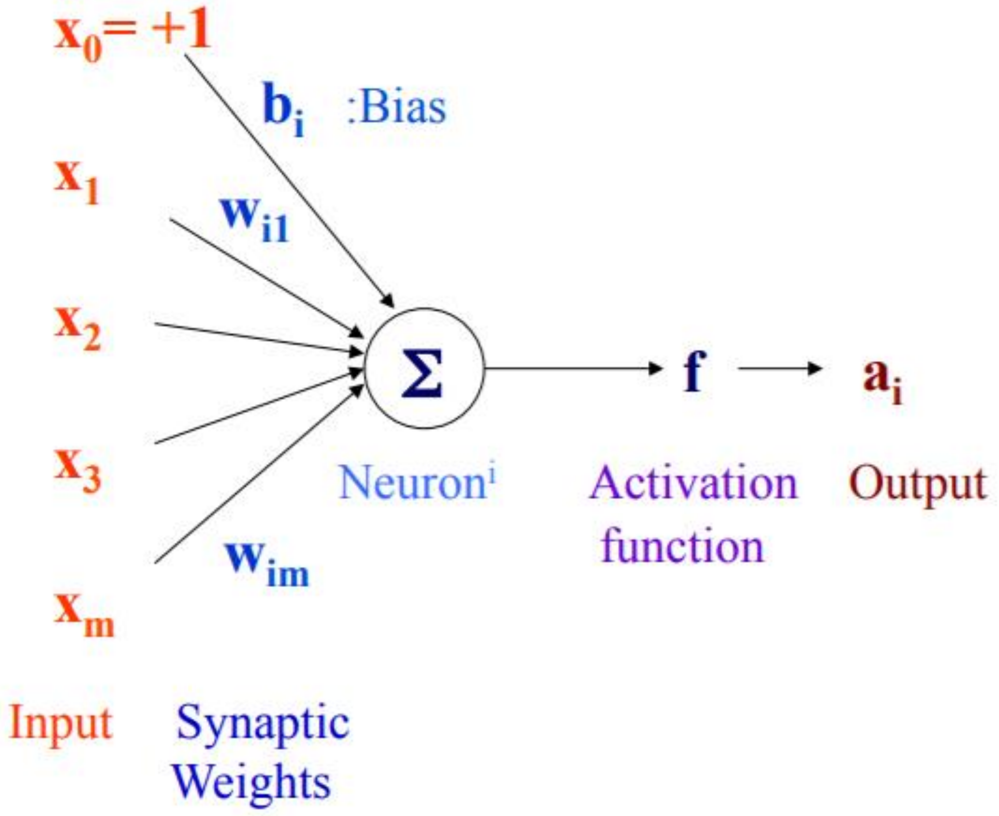
$$y_k = f(v_k)$$

The purpose of the **activation function is to add non-linearity to the neural** networks. It decides whether a neuron should be activated or not

Artificial Neural Network

Weight determines the connection between the two neurons.

Each connection of neurons has its own weight, and those are the only values that will be modified during the learning process.



Artificial Neural Network

$$a_i = f(n_i) = f\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right)$$

An artificial neuron:

- computes the weighted sum of its input (called its **net input**)
- adds its bias
- passes this value through an activation function

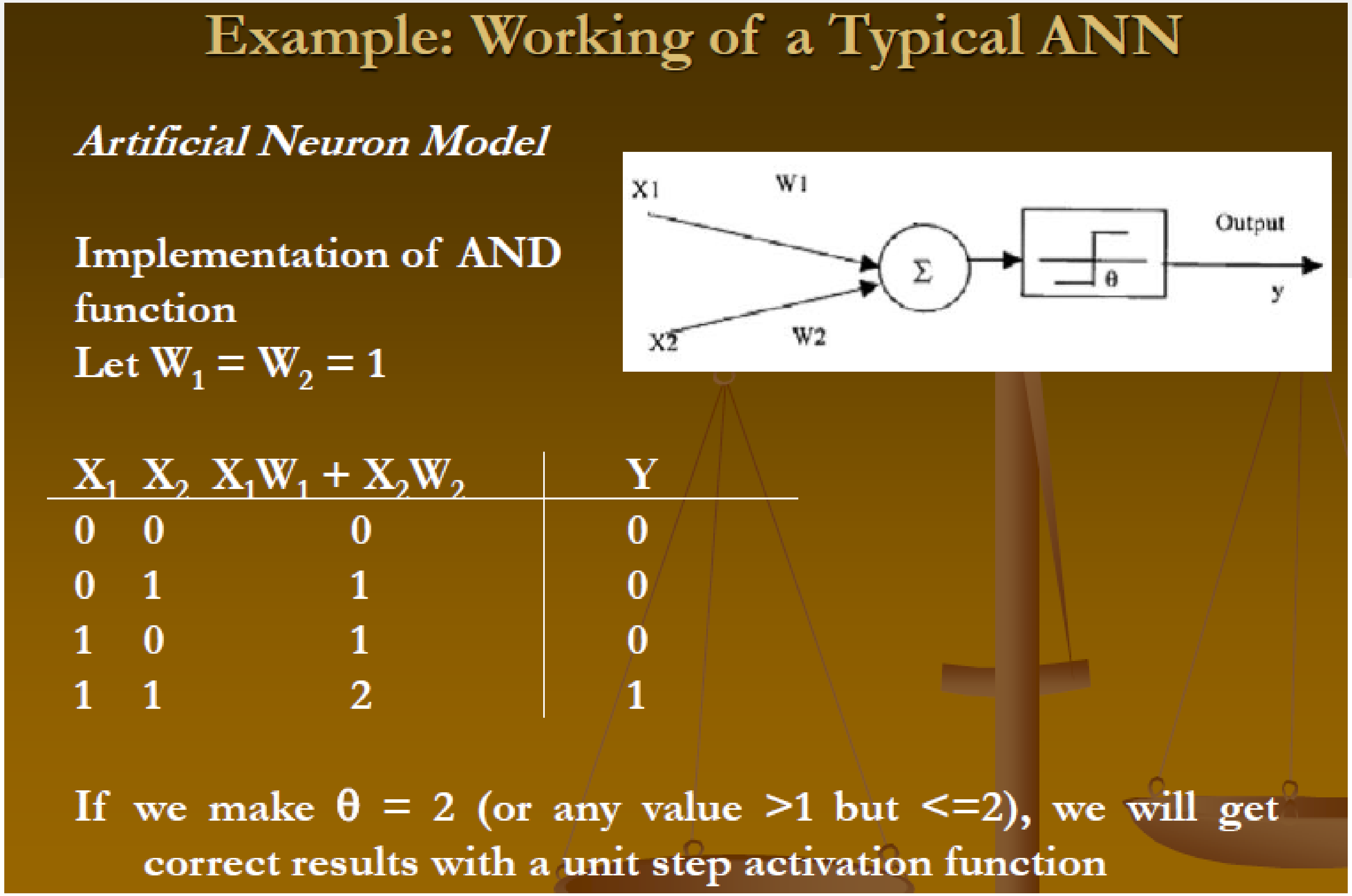
We say that the neuron “fires” (i.e. becomes active) if its output is above zero.

Artificial Neural Network

Bias can be incorporated as another weight clamped to a fixed input of +1.0

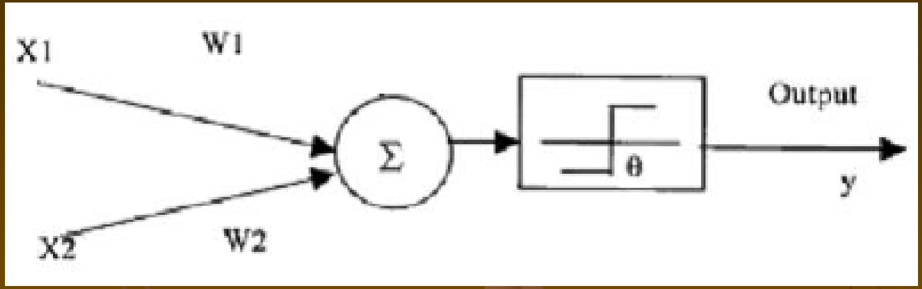
This extra free variable (bias) makes the neuron more powerful.

$$a_i = f\,(n_i) = f\,\left(\sum_{j=0}^{n} w_{ij}x_j\right) = f(\mathbf{w}_i.\mathbf{x}_j)$$



Example: Working of a Typical ANN

Artificial Neuron Model



Implementation of AND function

Let $W_1 = W_2 = 1$

X_1	X_2	$X_1W_1 + X_2W_2$	Y
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

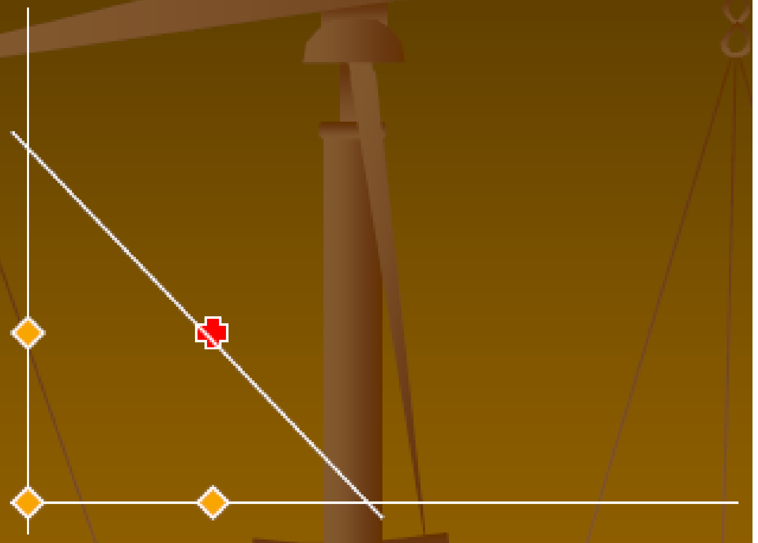
If we make $\theta = 2$ (or any value >1 but <=2), we will get correct results with a unit step activation function

Unit step activation

It takes an input value and returns a binary output based on a threshold.

Artificial Neuron Model

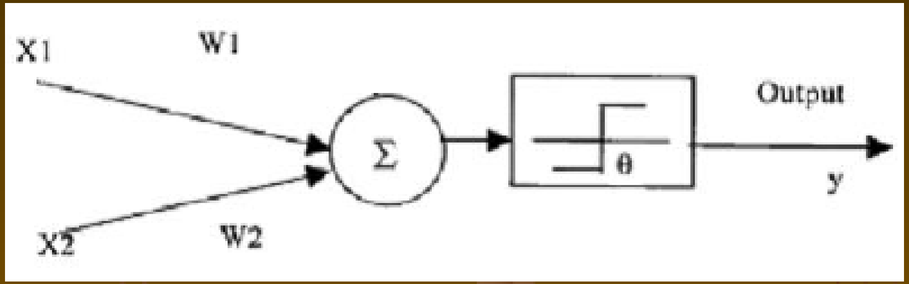
If we place the 4 points in a two coordinate system (X1 and X2), we have drawn a line from (2, 0) to (0, 2) in the resulting plane



Any new data falling on the left side of the line will give an output of zero and the data on the right side of the line will be classified as one

Example: Working of a Typical ANN

Artificial Neuron Model



Implementation of OR function

Let $W_1 = W_2 = 1$

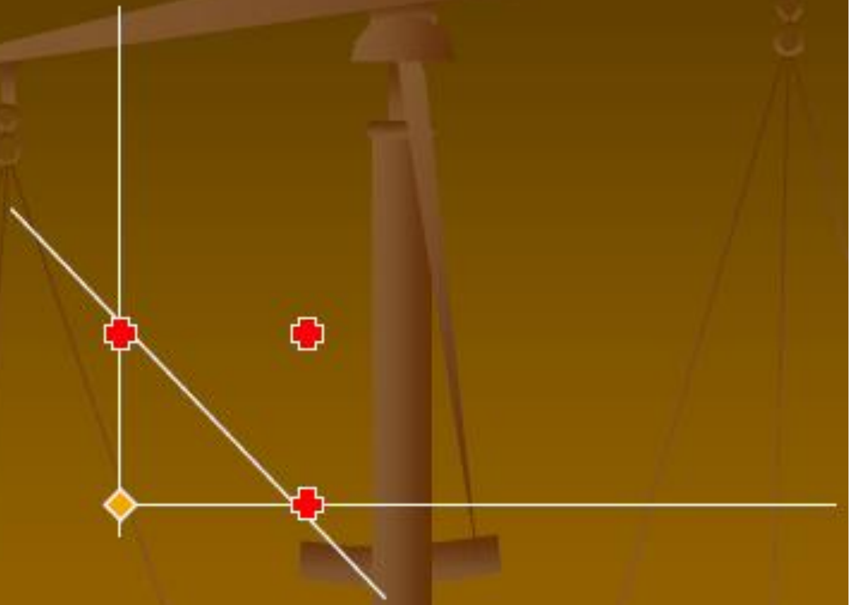
X_1	X_2	$X_1W_1 + X_2W_2$	Y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

If we make $\theta = 1$ (or any value >0 but <=1), we will get correct results with a unit step activation function

Artificial Neuron Model

If we place the 4 points in a two coordinate system (X1 and X2), we have drawn a line from (1, 0) to (0, 1) in the resulting plane

Any new data falling on the left side of the line will give an output of zero and the data on the right side of the line will be classified as one



Learning:

Continuous process of:

Trial:

Processing an input to produce an output (In terms of ANN: Compute the output function of a given input)

Evaluate:

Evaluating this output by comparing the actual output with the expected output.

Adjust:

Adjust the weights.

Representing Boolean Function

Simple example: AND

$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$

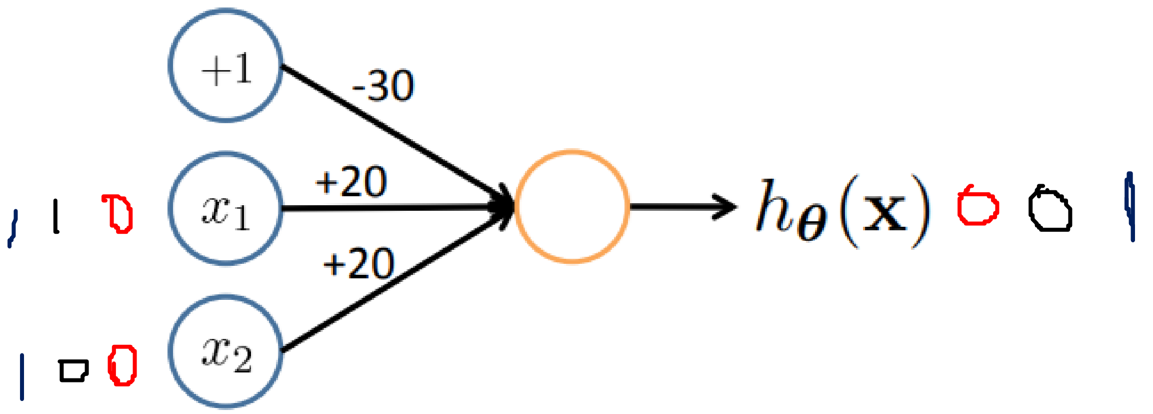
x_1	x_2	$h_\Theta(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

Representing Boolean Function

Simple example: AND

$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$



$$h_\Theta(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

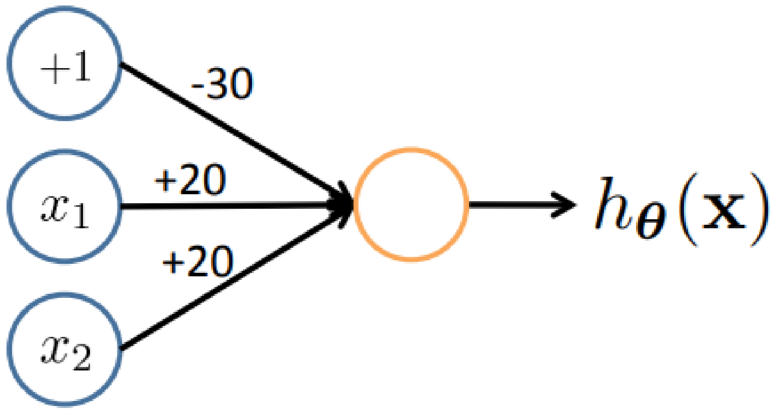
x_1	x_2	$h_\Theta(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

Representing Boolean Function

Simple example: AND

$x_1, x_2 \in \{0, 1\}$

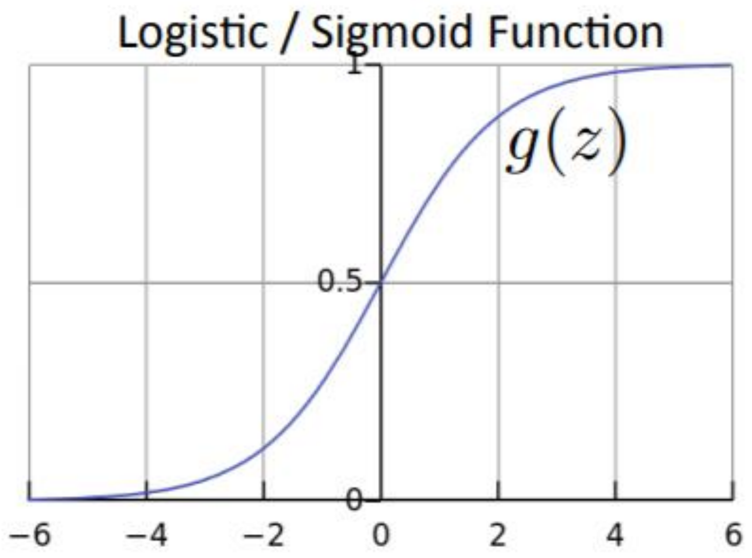
$y = x_1 \text{ AND } x_2$



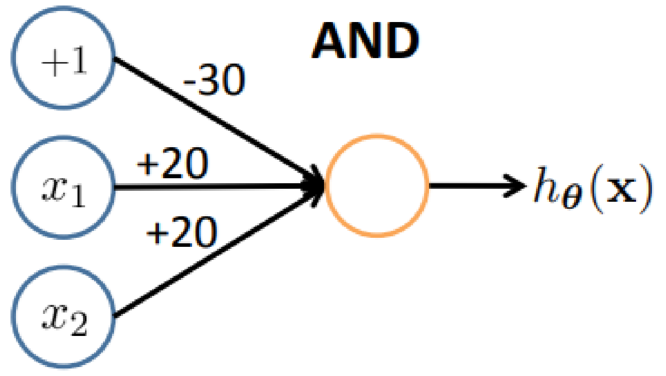
$h_\Theta(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$

x_1	x_2	$h_\Theta(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

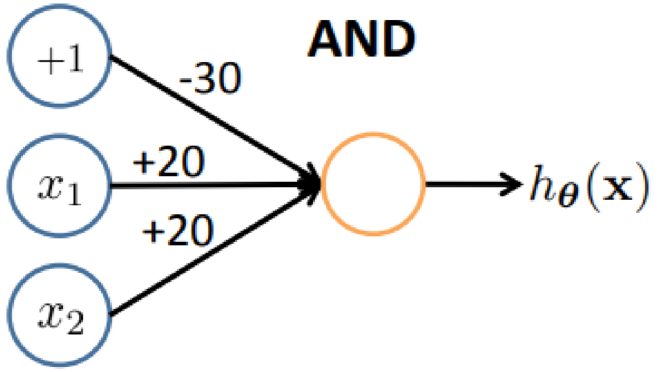
$$\sigma(z) = \frac{1}{1+\exp(-z)}$$



Representing Boolean Function

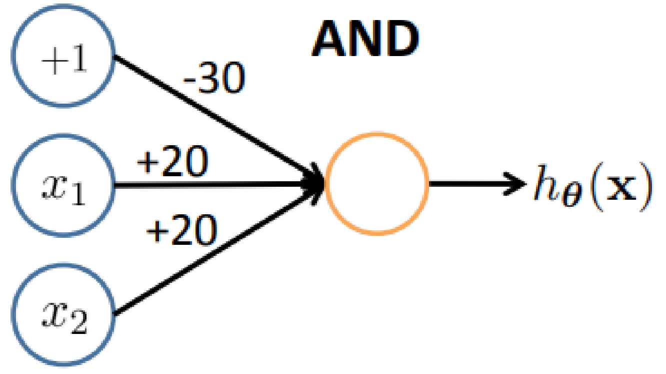


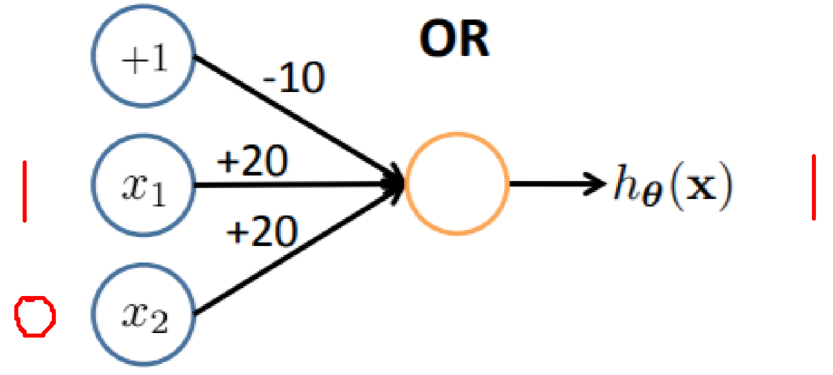
Representing Boolean Function



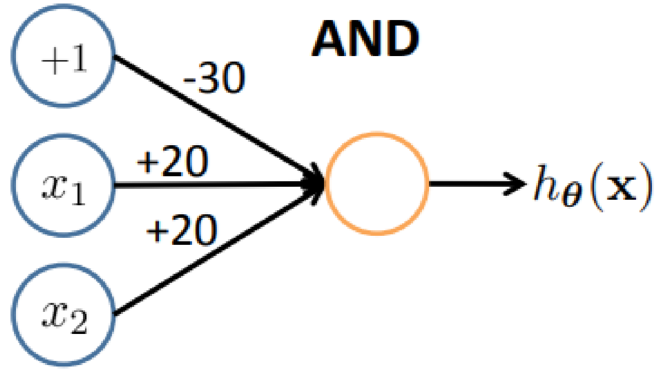
What changes will you make to convert this network into an OR function?

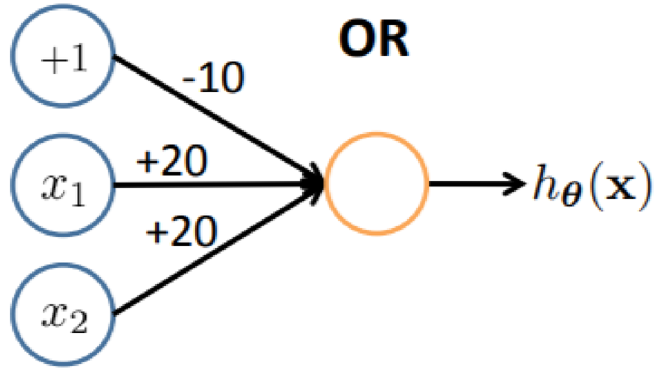
Representing Boolean Function

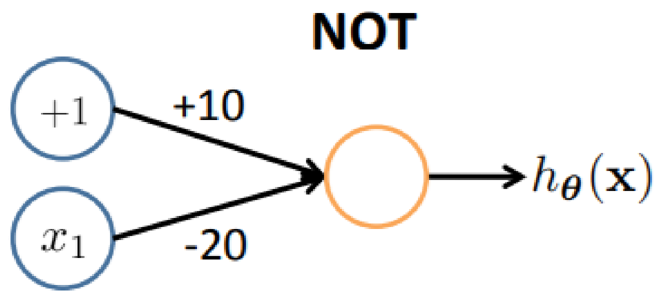




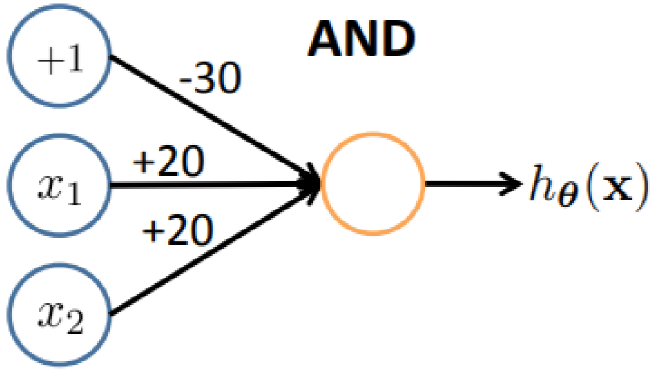
Representing Boolean Function

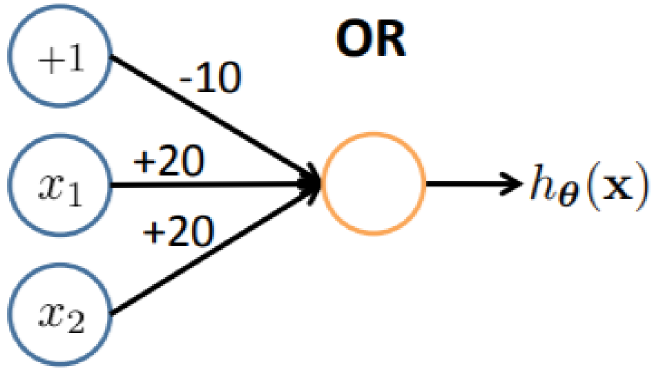


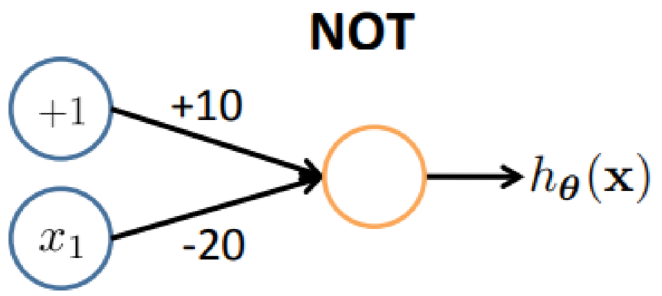


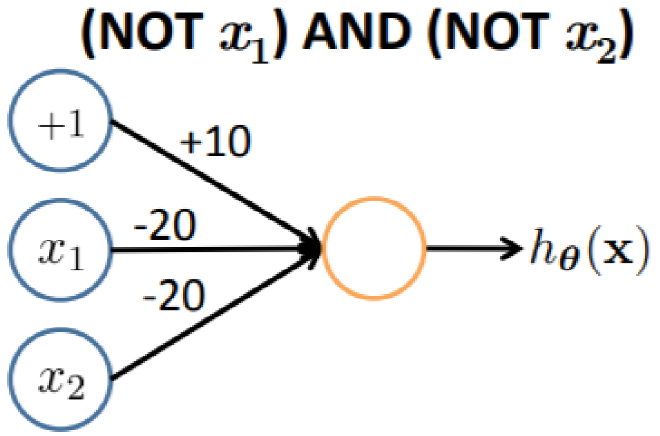


Representing Boolean Function









Show the mathematical working of Artificial Neural Network by taking the case in figure below. First two columns are the input values for X1 and X2 and the third column is the desired output.

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

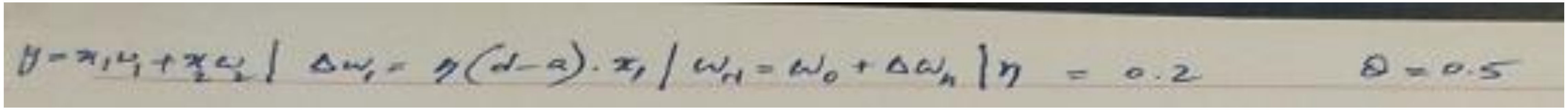
Learning rate = 0.2
Threshold = 0.5
Actual output = W1X1+W2X2

Next weight adjustment = Wn+ΔWn

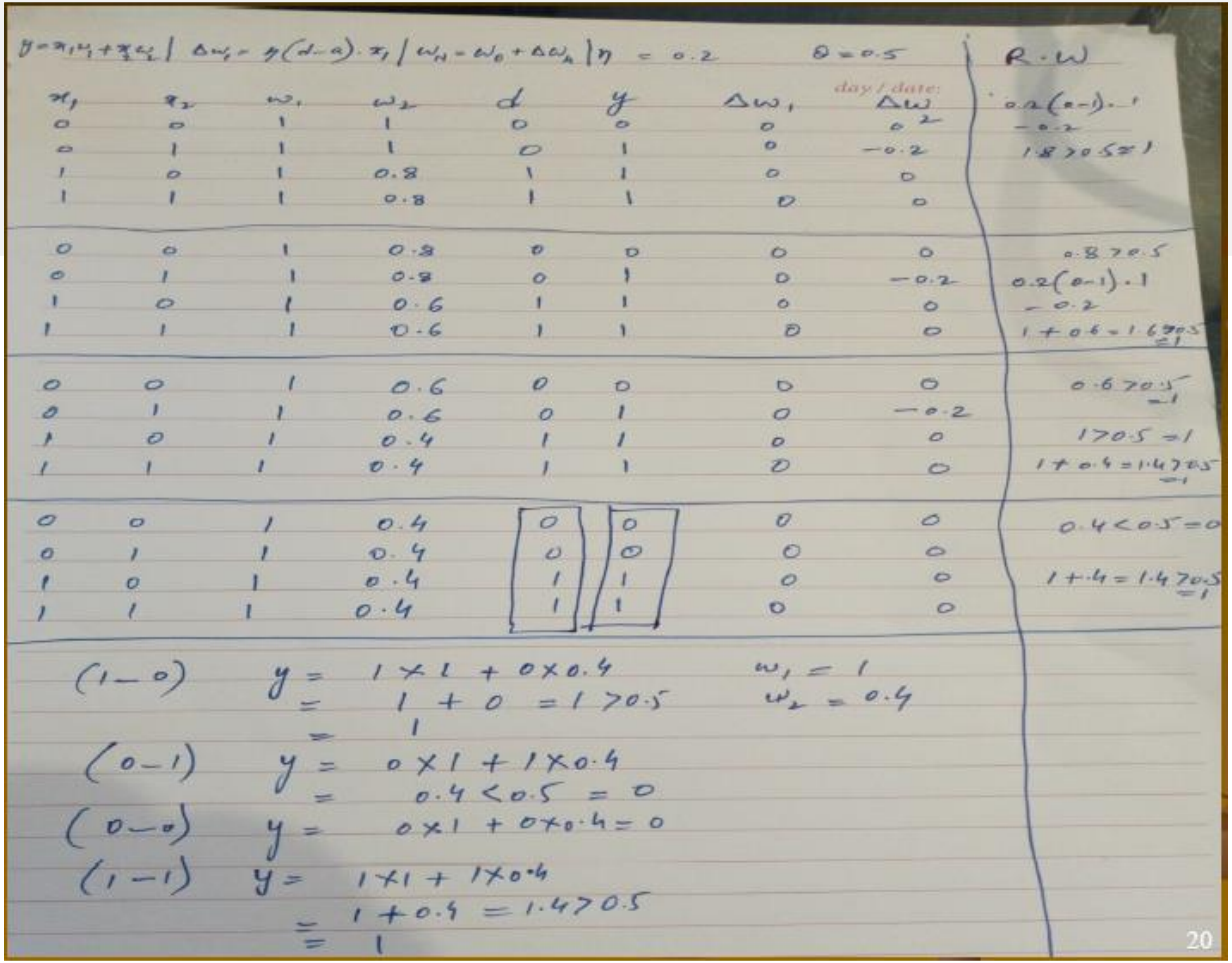
Change in weight = ΔWn = learning rate * (desired output- output) * Xn

Show complete iterations for acquiring the desired output?

x1	x2	w1	w2	d	y	Δw1	Δw2



$y = x_1 w_1 + x_2 w_2$ | $\Delta w_1 = \eta (d - a) . x_1$ | $w_{n1} = w_0 + \Delta w_n$ } $\eta = 0.2$ $\theta = 0.5$



$y = x_1w_1 + x_2w_2$ | $\Delta w_i = \eta(d-a)\cdot x_i$ | $w_n = w_o + \Delta w_n$ | $\eta = 0.2$ $\theta = 0.5$

x_1	x_2	w_1	w_2	d	y	Δw_1	Δw_2	R·W
0	0	1	1	0	0	0	0	$0.2(0-1)\cdot 1$
0	1	1	1	0	1	0	−0.2	−0.2 ; $1.8 > 0.5 = 1$
1	0	1	0.8	1	1	0	0	
1	1	1	0.8	1	1	0	0	
0	0	1	0.8	0	0	0	0	$0.8 > 0.5$
0	1	1	0.8	0	1	0	−0.2	$0.2(0-1)\cdot 1$
1	0	1	0.6	1	1	0	0	−0.2
1	1	1	0.6	1	1	0	0	$1 + 0.6 = 1.6 > 0.5 = 1$
0	0	1	0.6	0	0	0	0	$0.6 > 0.5 = 1$
0	1	1	0.6	0	1	0	−0.2	
1	0	1	0.4	1	1	0	0	$1 > 0.5 = 1$
1	1	1	0.4	1	1	0	0	$1 + 0.4 = 1.4 > 0.5 = 1$
0	0	1	0.4	0	0	0	0	$0.4 < 0.5 = 0$
0	1	1	0.4	0	0	0	0	
1	0	1	0.4	1	1	0	0	$1 + .4 = 1.4 > 0.5 = 1$
1	1	1	0.4	1	1	0	0	

(1 − 0) $y = 1 \times 1 + 0 \times 0.4$ $w_1 = 1$
$= 1 + 0 = 1 > 0.5$ $w_2 = 0.4$
$= 1$

(0 − 1) $y = 0 \times 1 + 1 \times 0.4$
$= 0.4 < 0.5 = 0$

(0 − 0) $y = 0 \times 1 + 0 \times 0.4 = 0$

(1 − 1) $y = 1 \times 1 + 1 \times 0.4$
$= 1 + 0.4 = 1.4 > 0.5$
$= 1$