
بسم الله الرحمن الرحیم

Artificial Neural Networks (ANN) and Deep Learning

References:

[1] S. Samarasinghe, Neural Networks for Applied Sciences and Engineering, Taylor & Francis, 2006.

[2] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[3] S. Haykin, Neural Networks and Learning Machines, Prentice-Hall, 2009.

[4] J. M. Zurada, Introduction to Artificial Neural Systems, Info Access and Distribution, 1992.

[5] L. Fausett, Fundamentals of Neural Networks, Prentice-Hall, 1994.

[6] Selected Papers

Artificial Neural Networks

General functions of neural networks

Classification

separation of input data in specified classes

e.g. Handwritten character recognition

Clustering

grouping together objects that are similar to each other

e.g. Data conceptualization

e.g. Solving the TSP

Prediction

given a sequence of data, predict the upcoming ones

e.g. Stock market forecasting

Associative memory

restores deformed input samples to its original content

e.g. Filtering of noisy samples

e.g. Image compression

Artificial Neural Networks

- McCulloch & Pitts (1943) are generally recognised as the designers of the first neural network
- Many of their ideas still used today (e.g. many simple units combine to give increased computational power and the idea of a threshold)

Artificial Neural Networks

- **Hebb (1949) developed the first learning rule (on the premise that if two neurons were active at the same time the strength between them should be increased)**

Historical Development of ANNs

- 1940s: The beginning of ANN
- McCulloch & Pitts neurons. Simple logic function represented in a temporal framework
- 1950s & 60s: The first golden age of ANNs
- Perceptrons (Rosenblatt, Minsky, Papert, Block). Typical configuration consists of input nodes connected by paths with adjustable weights to associated neurons. Learning rule enabled configuration consists of input nodes connected by paths with weights to converge to associate training inputs with outputs. Adjustment occurs when response is incorrect.
- ADALINE (Widrow & Hoff). Developed delta rule for single layer networks. Adjusts weights to reduce the difference between the net input to the output unit and the desired output.

Historical Development of ANNs

- 1970s: The quiet years
- Kohonen – development of self-organizing feature maps that use a topological structure for cluster units (SOM)
- Grossberg and Carpenter – Adaptive resonance theory (ART)
- 1980s: Renewed enthusiasm
- Backpropagation (Parker & LeCun). Publicized by PDP Group (Rumelhart, McClelland et al.)
- Hopfield networks – associative networks to solve constraint satisfaction problems based on fixed weights and adaptive activations
- Boltzmann machine – nondeterministic neural nets in which weights or activations are changed on the basis of a probability density function. Incorporates classical ideas such as simulated annealing and Bayesian decision theory

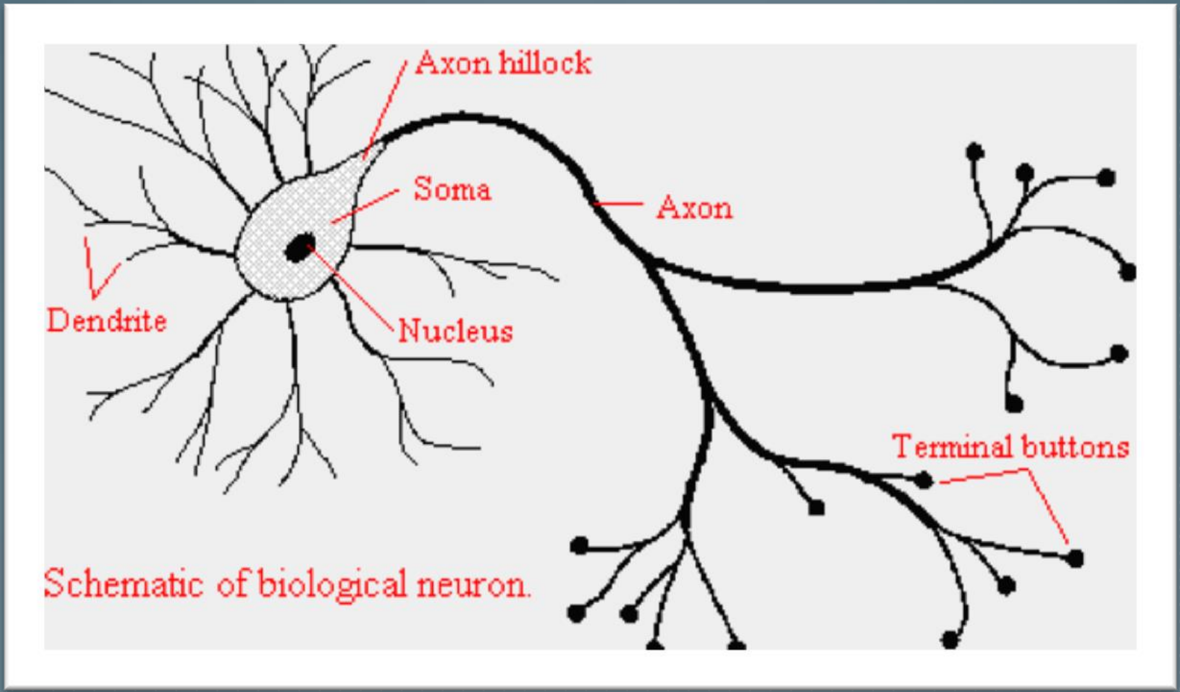
Historical Development of ANNs

- 1990s: Many New Structures of ANN
- Wavelet Neural Networks
- Quantum-based Neural Networks
- Cellular Networks
- Others
- 2000 – onwards:
- Spiking Neural Network (3th Generation of ANN)
- After 2000 – Development of Deep Learning

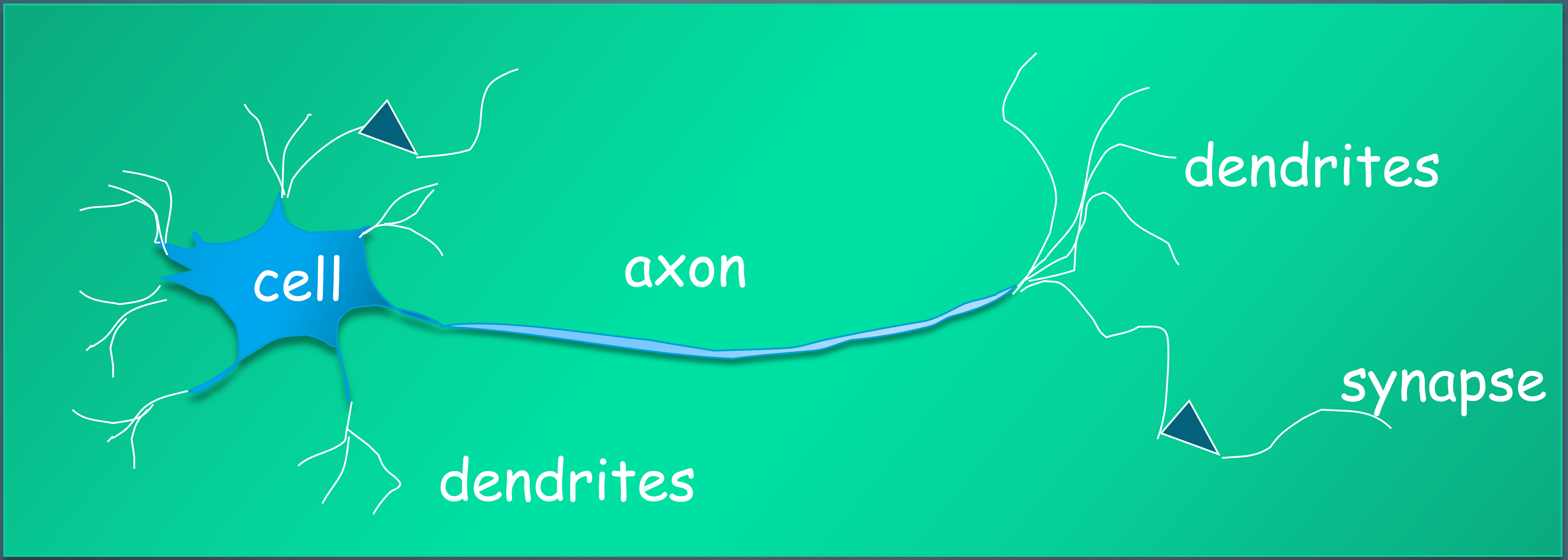
How Does the Brain Work ?

NEURON

- The cell that perform information processing in the brain
- Each consists of : SOMA, DENDRITES, AXON, and SYNAPSE



Biological neurons

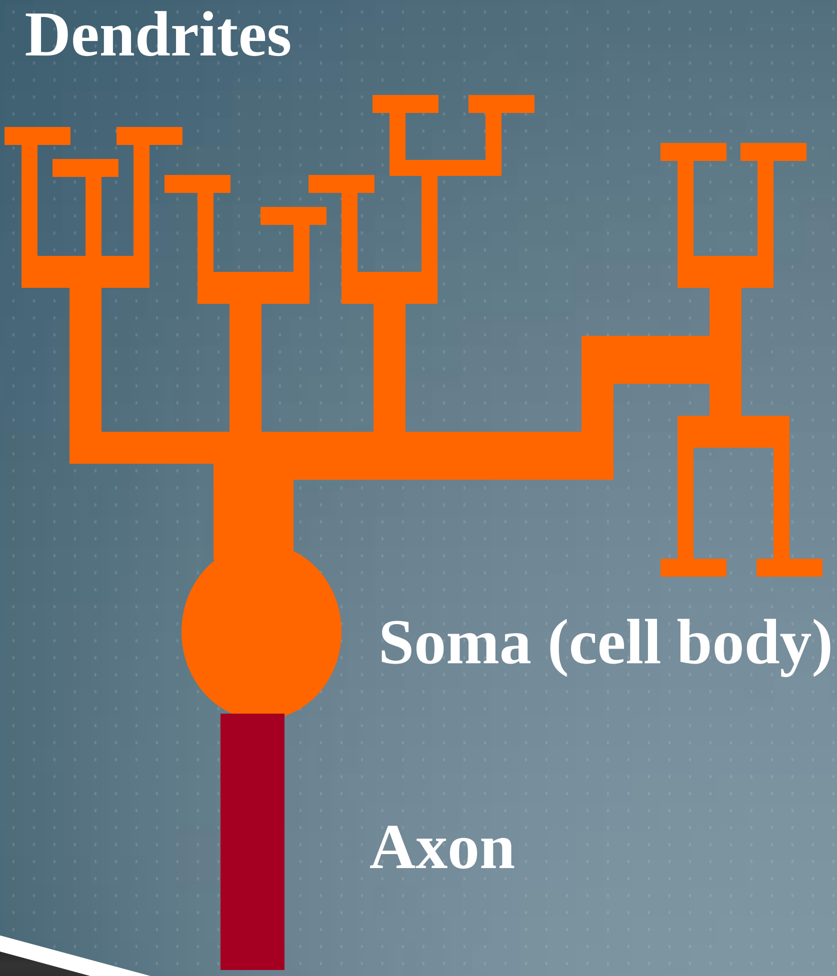


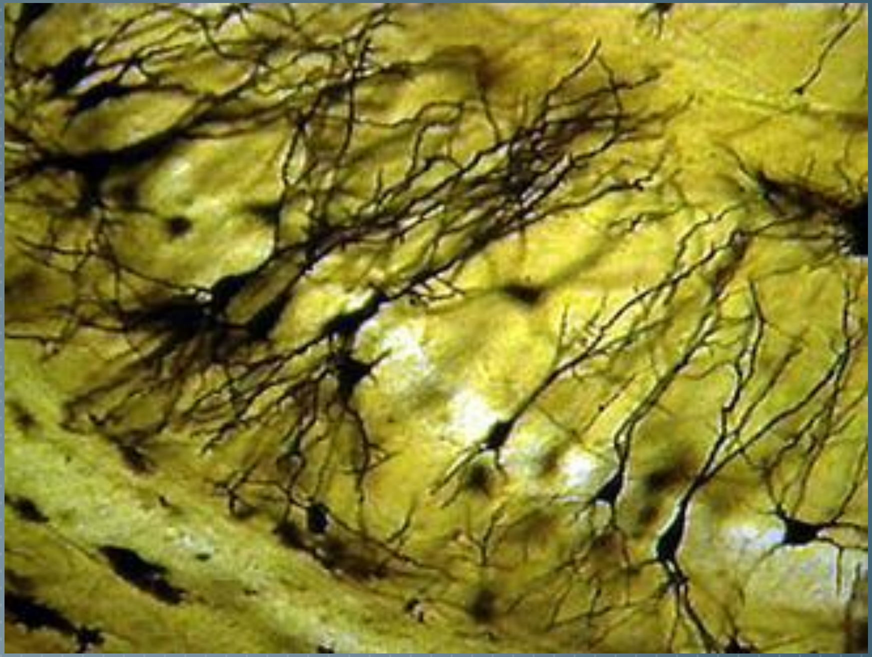
Neural Networks

- We are born with about 100 billion neurons

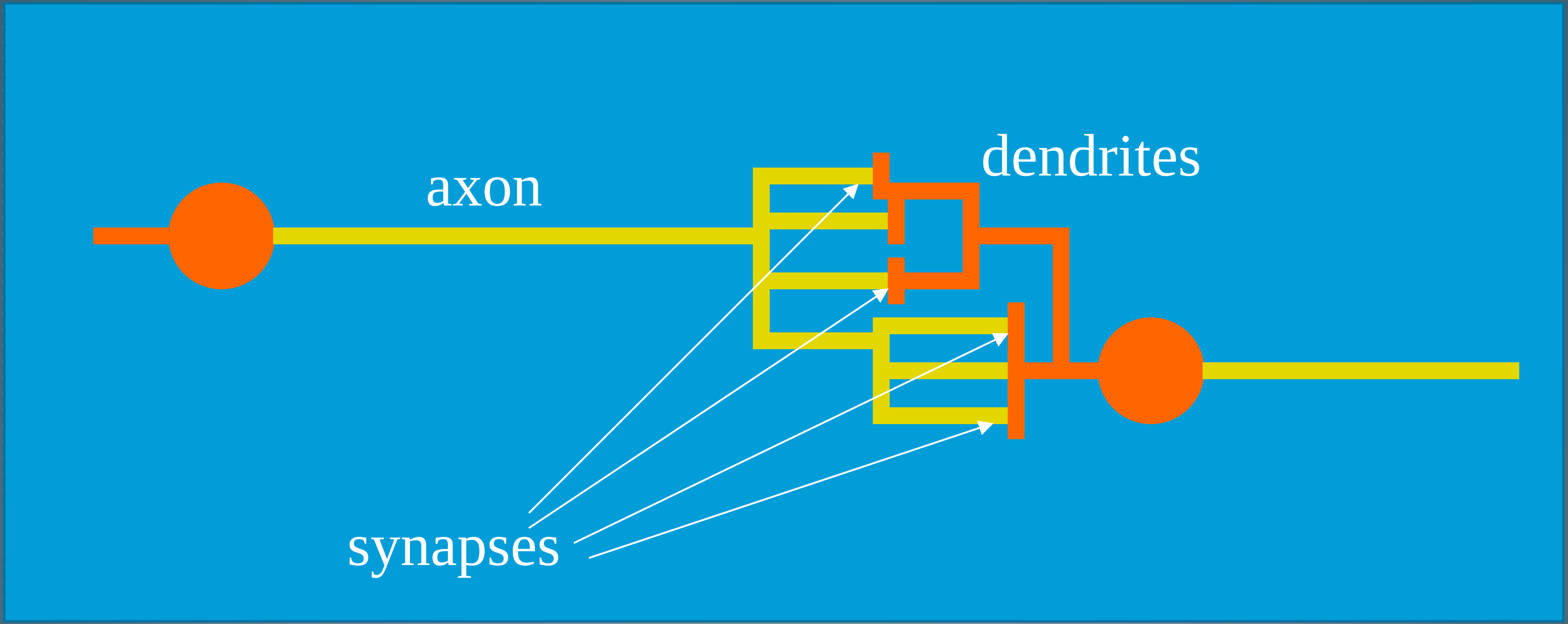
- A neuron may connect to as many as 100,000 other neurons

Biological inspiration





Biological inspiration



The information transmission happens at the synapses.

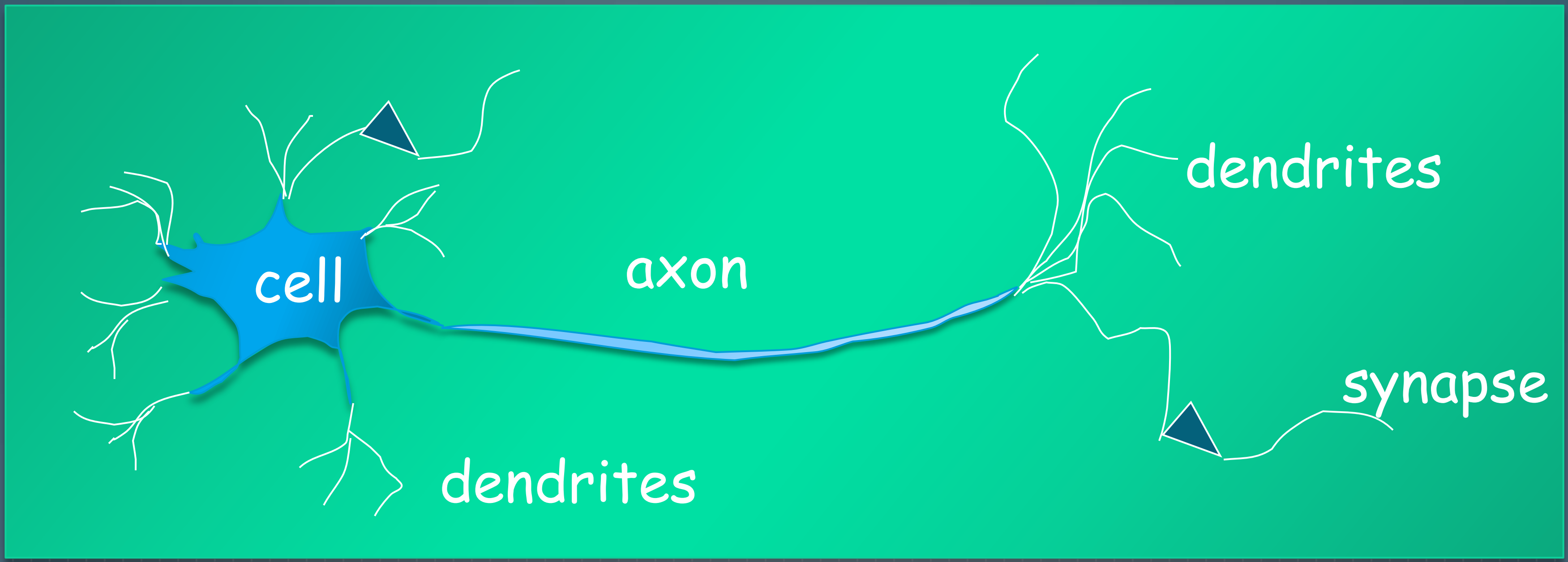
Biological inspiration

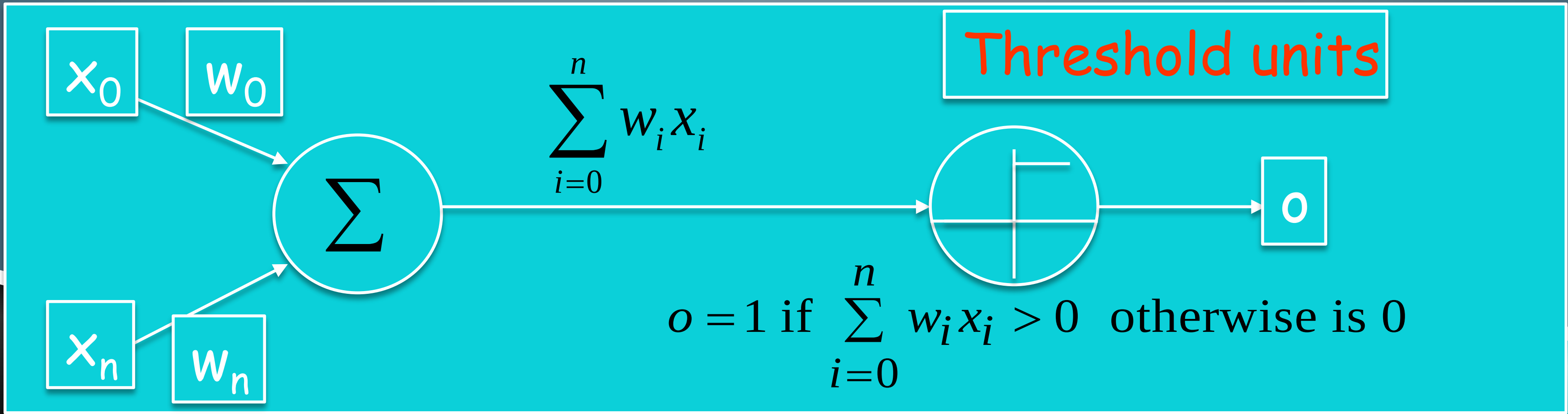
- The spikes travelling along the axon of the pre-synaptic neuron trigger the release of neurotransmitter substances at the synapse.
- The neurotransmitters cause excitation or inhibition in the dendrite of the post-synaptic neuron.
- The integration of the excitatory and inhibitory signals may produce spikes in the post-synaptic neuron.
- The contribution of the signals depends on the strength of the synaptic connection.

Biological Neurons

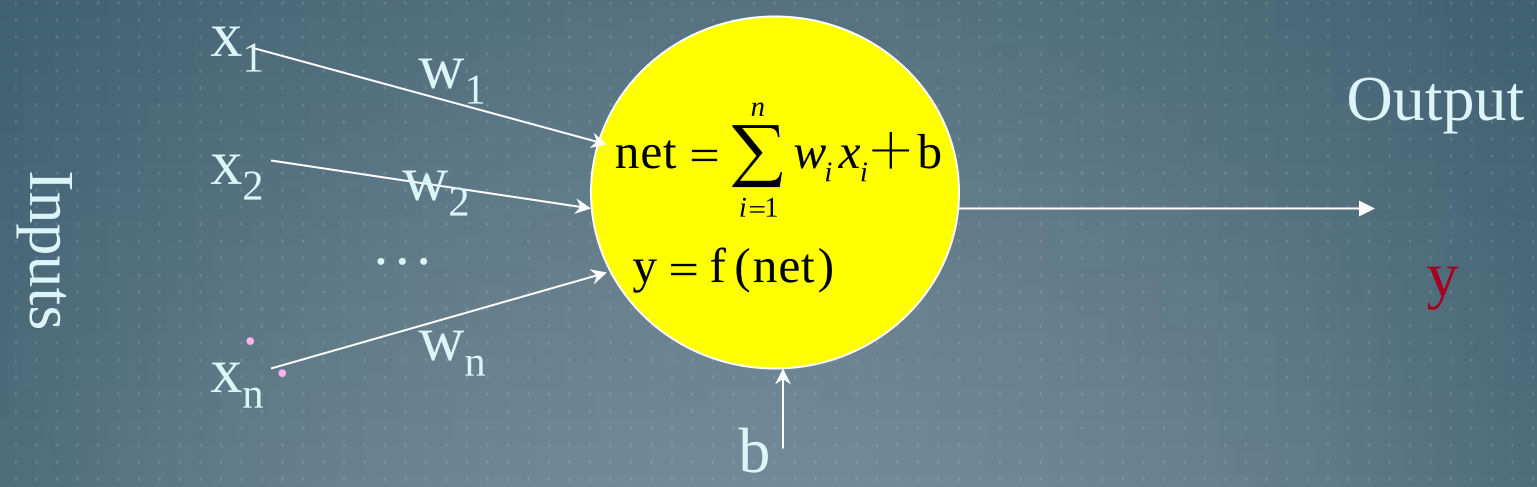
- human information processing system consists of brain neuron: basic building block
  - cell that communicates information to and from various parts of body
- Simplest model of a neuron: considered as a threshold unit –a processing element (PE)
- Collects inputs & produces output if the sum of the input exceeds an internal threshold value

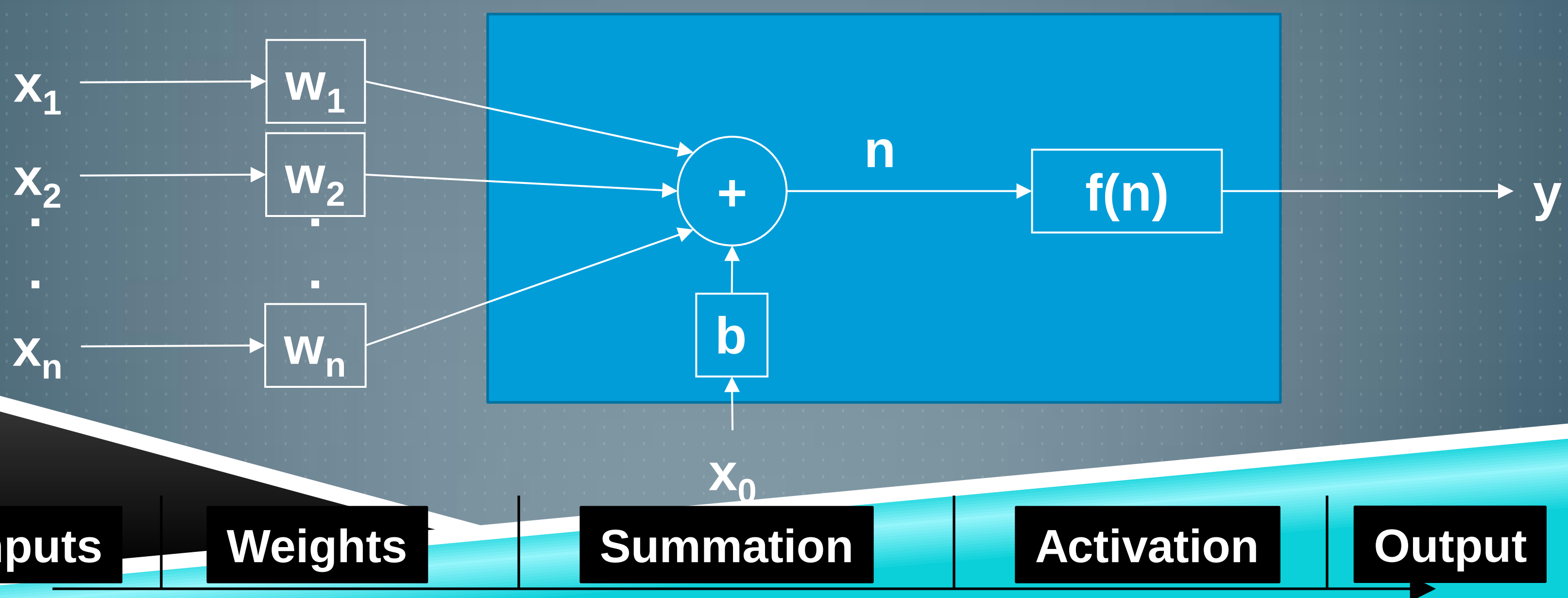
Real vs Artificial neurons



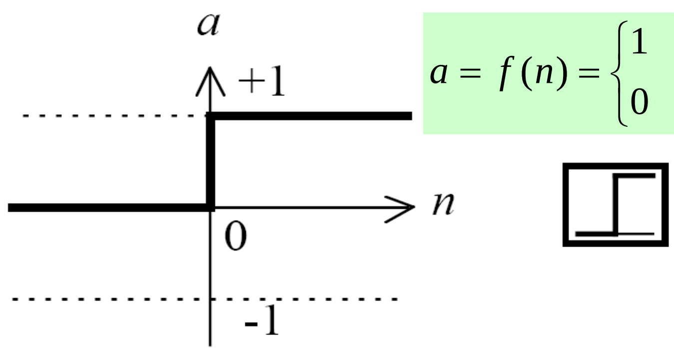


Mathematical Representation





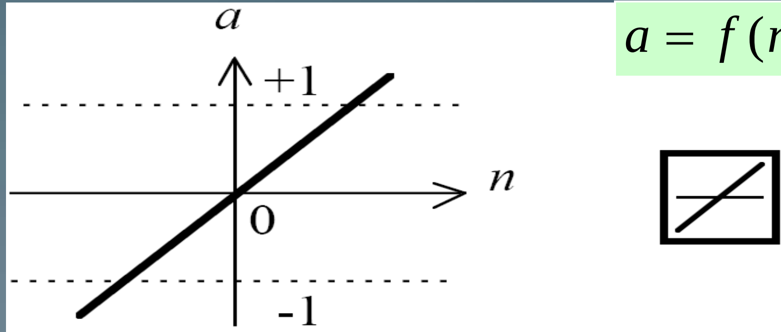
Mathematical Representation of some Activation Functions



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

$a = hardlim(n)$

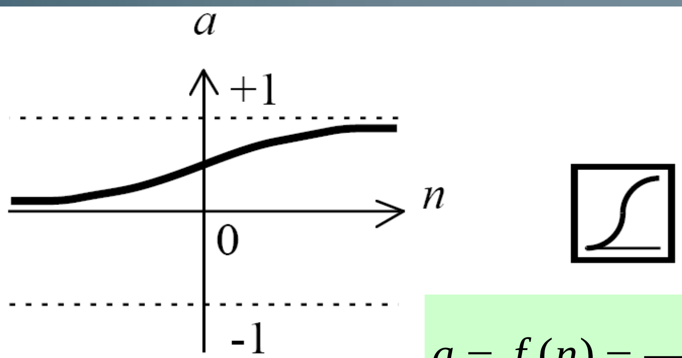
Hard-Limit Transfer Function



$$a = f(n) = n$$

$a = purelin(n)$

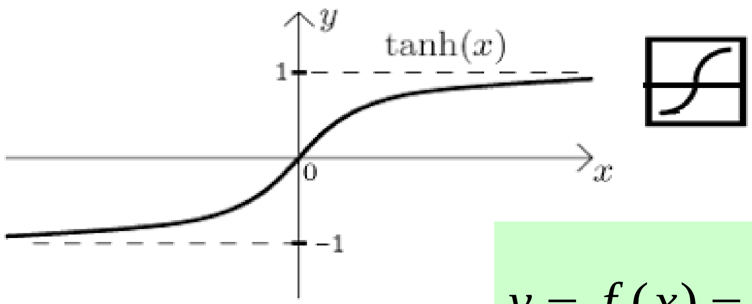
Linear Transfer Function



$$a = f(n) = \frac{1}{1 + e^{-n}}$$

$a = logsig(n)$

Log-Sigmoid Transfer Function



$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

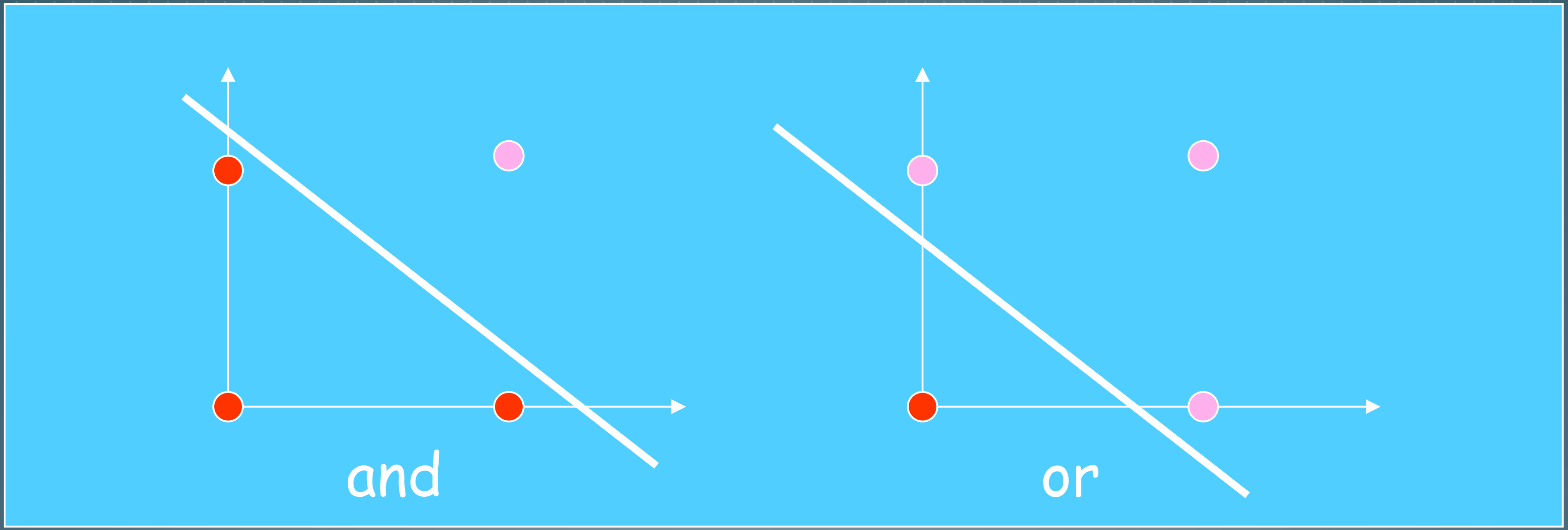
$y = tanh(x)$

Hyperbolic Tangent Transfer Function

Linear Separable

some functions not representable - e.g., (b) not linearly separable

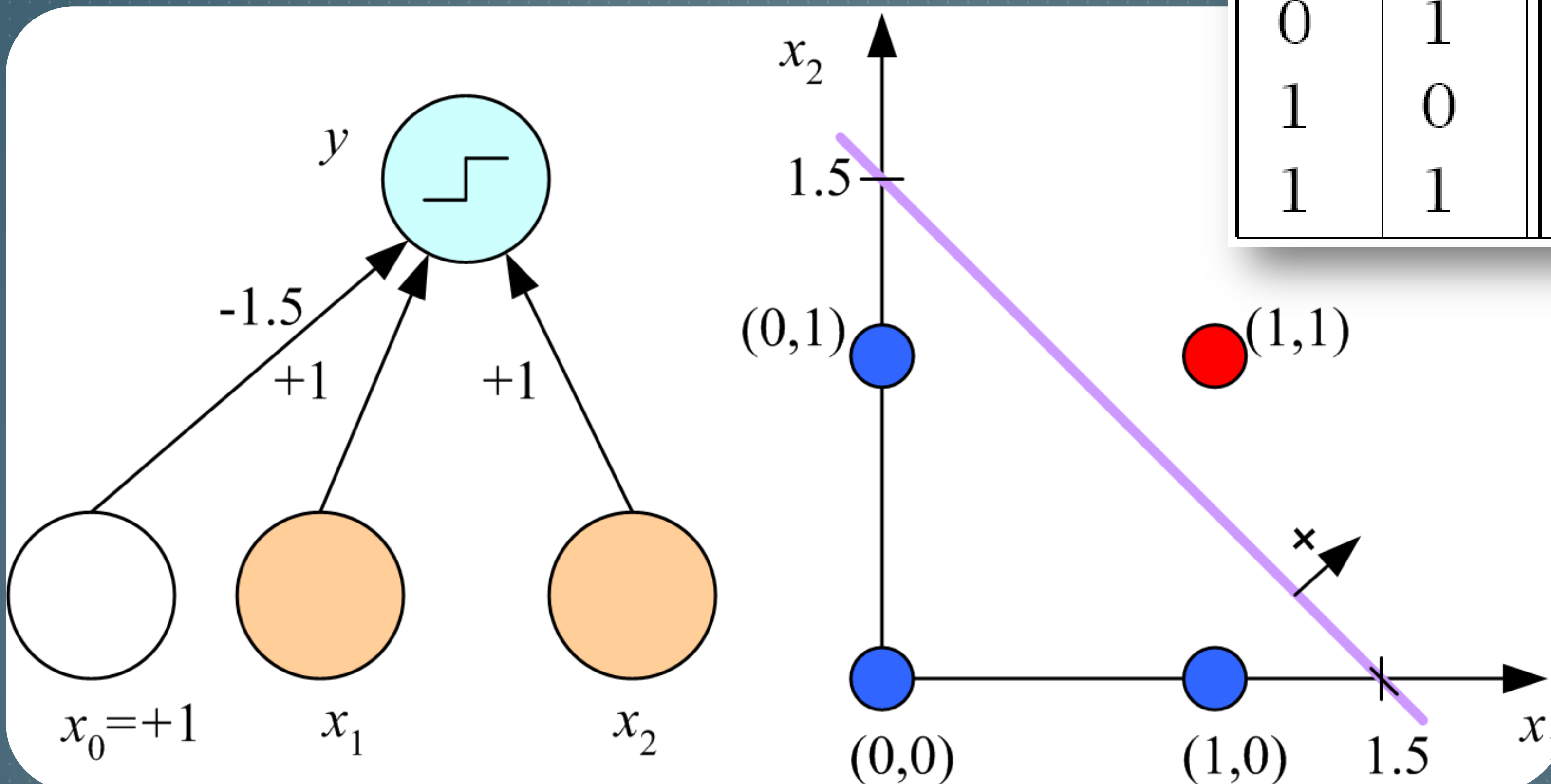
So what can be represented using perceptron?



Representation theorem: 1 layer feedforward networks can only represent linearly separable functions. That is, the decision surface separating positive from negative examples has to be a plane.

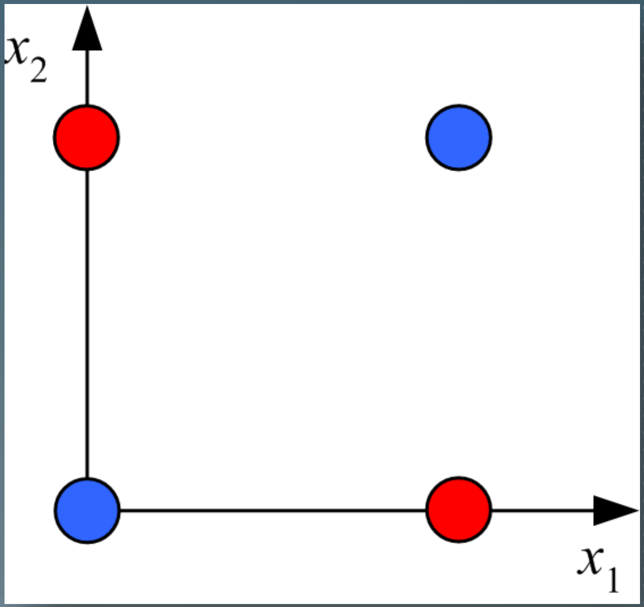
Learning Boolean AND

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

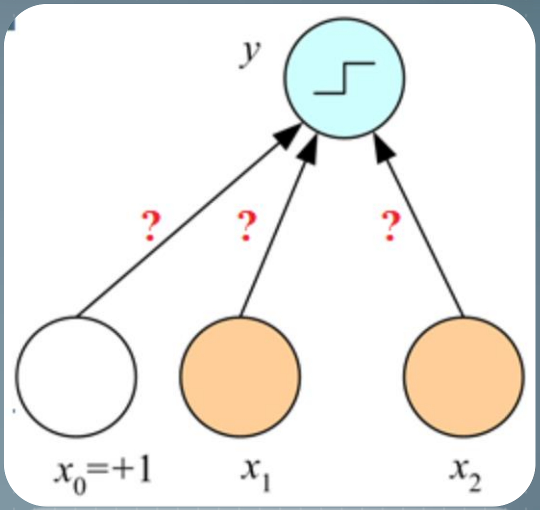


XOR

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

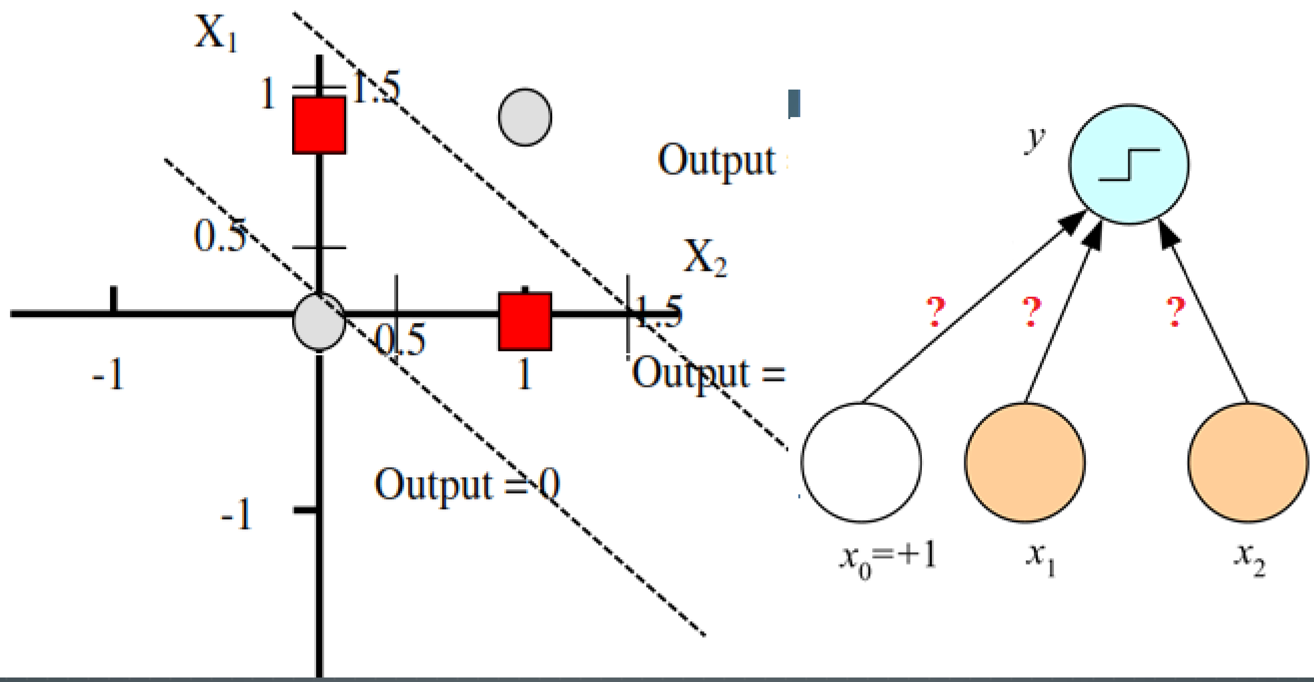


- **No w_0, w_1, w_2 satisfy**



XOR

A Possible Solution to the XOR Problem By Using Two Lines to Separate the Plane into Three Regions



Expressive limits of perceptrons

- Can the XOR function be represented by a perceptron (a network without a hidden layer)?

XOR **cannot** be represented.

Hebb Learning Rule Algorithm for a perceptron:

1. Set all weights to zero, $w_i = 0$ for i=1 to n, and bias to zero.
2. For each input vector, S(input vector) : t(target output pair), repeat steps 3-5.
3. Set activations for input units with the input vector $X_i = S_i$ for i = 1 to n.
4. Set the corresponding output value to the output neuron, i.e. y = t.
5. Update weight and bias by applying Hebb rule for all i = 1 to n:

$$w_i\,(new) = w_i\,(old) + x_i\,y_i$$

$$b\,(new) = b\,(old) + y_i$$

Hebb Learning Rule Algorithm for AND Gate

INPUT				TARGET	
	x_1	x_2	b		y
X_1	-1	-1	1	Y_1	-1
X_2	-1	1	1	Y_2	-1
X_3	1	-1	1	Y_3	-1
X_4	1	1	1	Y_4	1

Hebb Learning Rule Algorithm for AND Gate

There are 4 training samples, so there will be 4 iterations. Also, the activation function used here is Bipolar Sigmoidal Function so the range is [-1,1].

Step 1 :

Set weight and bias to zero, $w = [\ 0\ 0\ 0\ ]^T$ and $b = 0$.

Step 2 :

Set input vector $X_i = S_i$ for $i = 1$ to 4.

$X_1 = [\ -1\ -1\ 1\ ]^T$

$X_2 = [\ -1\ 1\ 1\ ]^T$

$X_3 = [\ 1\ -1\ 1\ ]^T$

$X_4 = [\ 1\ 1\ 1\ ]^T$

Hebb Learning Rule Algorithm for AND Gate

Step 3 :
Output value is set to y = t.
Step 4 :
Modifying weights using Hebb Rule:
First Sample:
$w(new) = w(old) + x_1 y_1 = [\ 0\ 0\ 0\ ]^T + [\ -1\ -1\ 1\ ]^T . [\ -1\ ] = [\ 1\ 1\ -1\ ]^T$
For the second iteration, the final weight of the first one will be used and so on.
Second Sample:
$w(new) = [\ 1\ 1\ -1\ ]^T + [\ -1\ 1\ 1\ ]^T . [\ -1\ ] = [\ 2\ 0\ -2\ ]^T$

Hebb Learning Rule Algorithm for AND Gate

Third Sample:

w(new) = $[\ 2\ 0\ -2]^T + [\ 1\ -1\ 1\ ]^T . [\ -1\ ] = [\ 1\ 1\ -3\ ]^T$

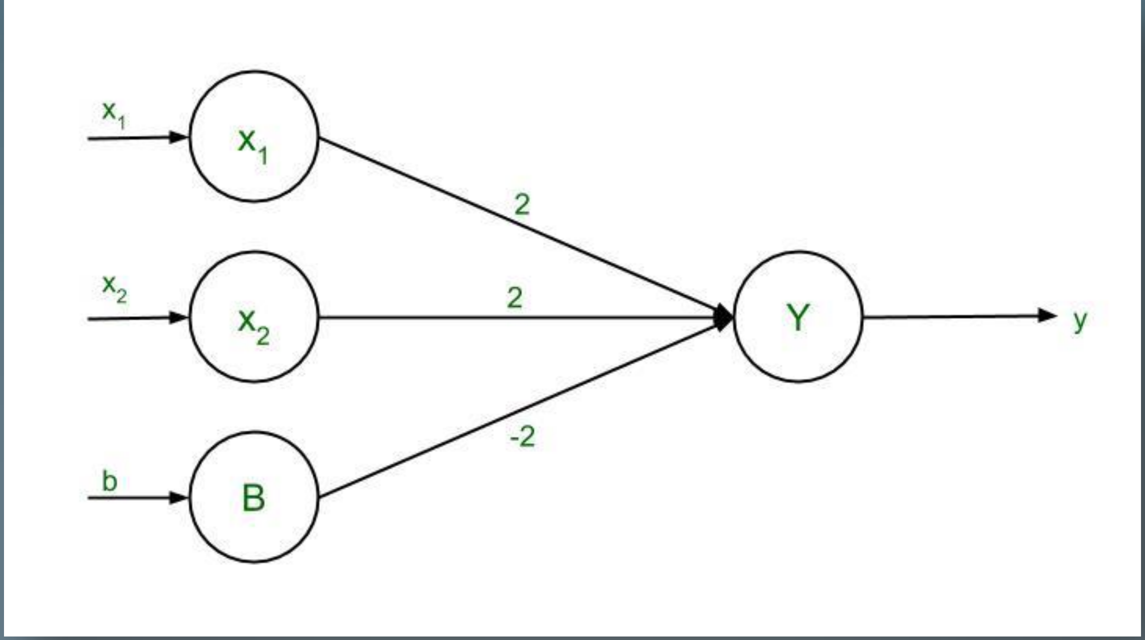
Fourth Sample:

w(new) = $[\ 1\ 1\ -3]^T + [\ 1\ 1\ 1\ ]^T . [\ 1\ ] = [\ 2\ 2\ -2\ ]^T$

So, the final weight matrix is $[\ 2\ 2\ -2\ ]^T$

Hebb Learning Rule Algorithm for AND Gate

Testing the network :



The network with the final weights

Hebb Learning Rule Algorithm for AND Gate

For $x_1 = -1, x_2 = -1, b = 1, Y = (-1)(2) + (-1)(2) + (1)(-2) = -6$

For $x_1 = -1, x_2 = 1, b = 1, Y = (-1)(2) + (1)(2) + (1)(-2) = -2$

For $x_1 = 1, x_2 = -1, b = 1, Y = (1)(2) + (-1)(2) + (1)(-2) = -2$

For $x_1 = 1, x_2 = 1, b = 1, Y = (1)(2) + (1)(2) + (1)(-2) = 2$

The results are all compatible with the original table.

Hebb Learning Rule Algorithm for AND Gate

Decision Boundary :

$2x_1 + 2x_2 – 2b = y$

Replacing y with 0, $2x_1 + 2x_2 – 2b = 0$

Since bias, b = 1, so $2x_1 + 2x_2 – 2(1) = 0$

$2(x_1 + x_2) = 2$

The final equation, $x_2 = -x_1 + 1$