Neuro Fuzzy Techniques

Dr. Poonam Sharma

SYLLABUS

Neural Networks: History, overview of biological neuro-system, mathematical models of neurons, ANN architecture, Learning rules, Learning Paradigms-Supervised, Unsupervised and reinforcement Learning, Learning Tasks, ANN training Algorithms-Single layer perceptron, multi-layer perceptron, Self-organizing Map, Applications of Artificial Neural Networks.

Introduction to fuzzy set, Operations on fuzzy sets, Fuzzy relation, Fuzzy implication, approximate reasoning, Fuzzy rule-based systems, Fuzzy reasoning schemes, Fuzzy logic controller.
Implementing fuzzy IF-THEN rules by trainable neural nets. Fuzzy neurons, Hybrid neural networks, Neuro-fuzzy classifiers.

LIST OF BOOKS

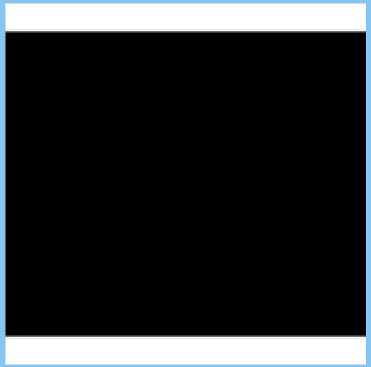
Neuro-Fuzzy and Soft Computing: A computational Approach to Learning & Machine Intelligence; Roger Jang, Tsai Sun, Eiji Mizutani, PHI

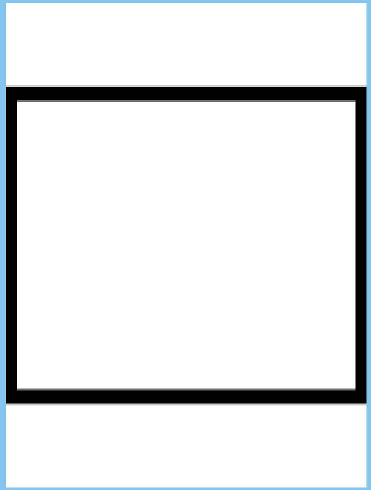
Soft Computing and Its Applications : R.A. Aliev, R.R. Aliev

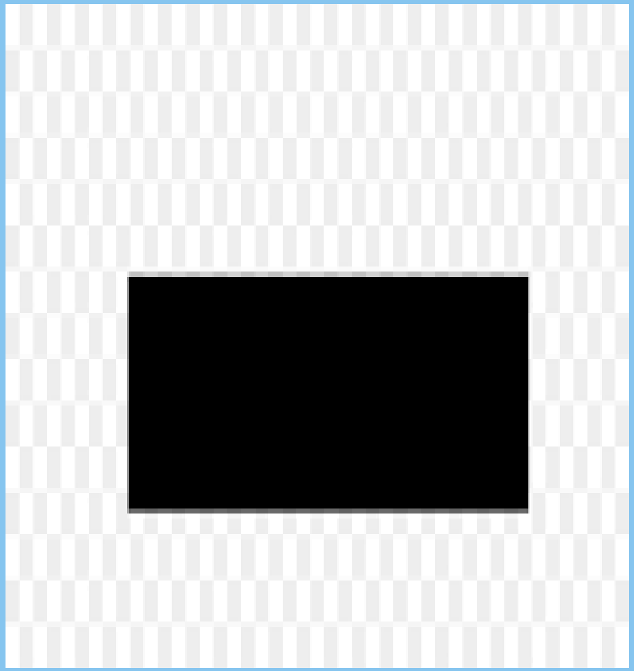
Neural Network: A Comprehensive Foundation; Simon Haykin, PHI.

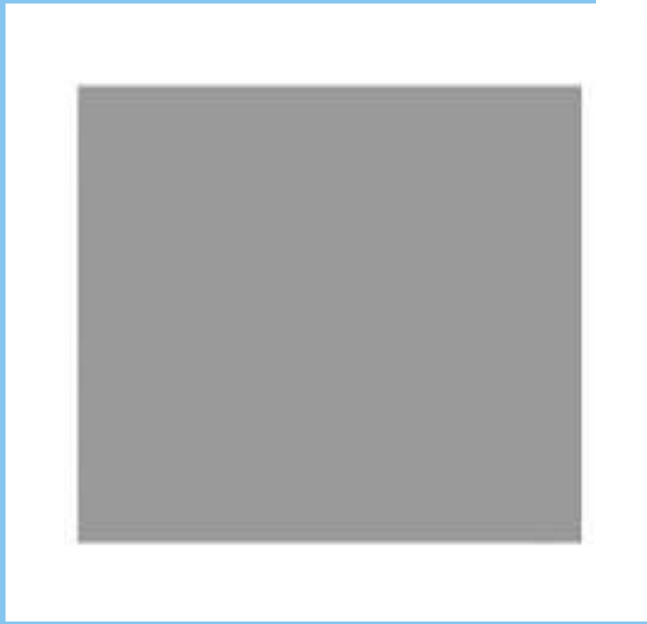
Elements of artificial Neural Networks; Kishan Mehtrotra, S. Ranka, Penram International Publishing (India).

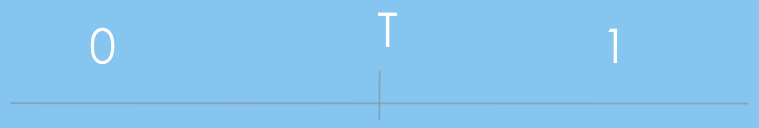
Fuzzy Logic with Engineering Applications; Timothy Ross, McGraw-Hill. Neural Networks and Fuzzy Systems: Bar Kosko , PHI.

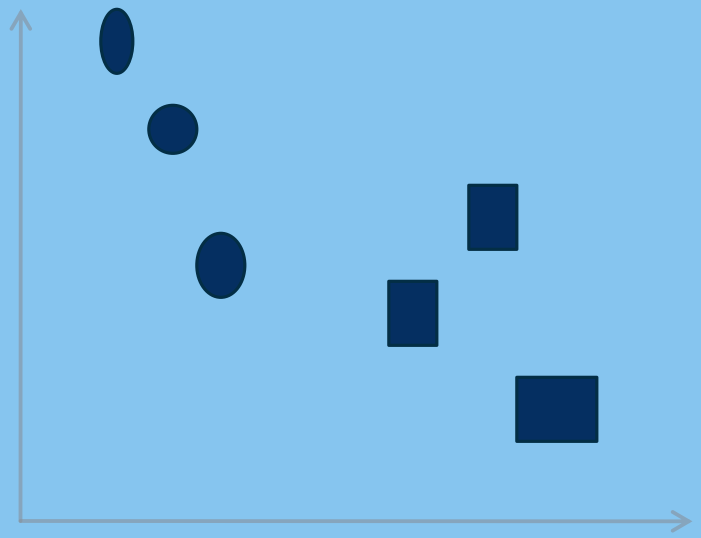


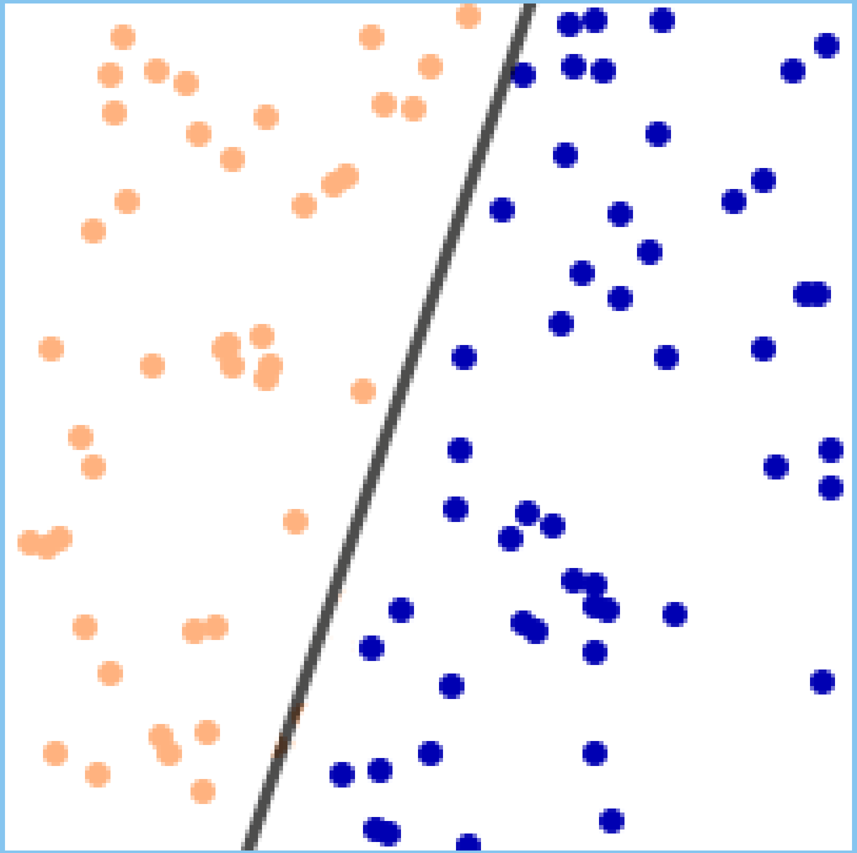


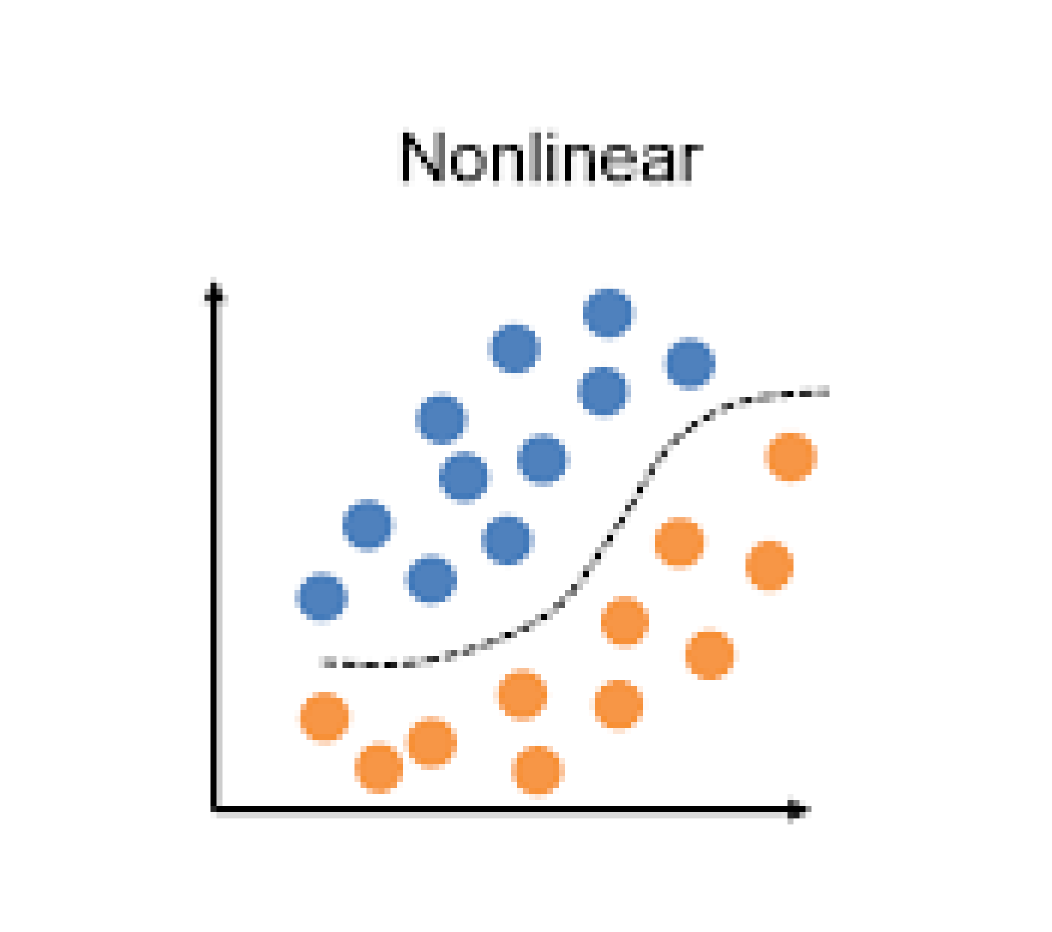


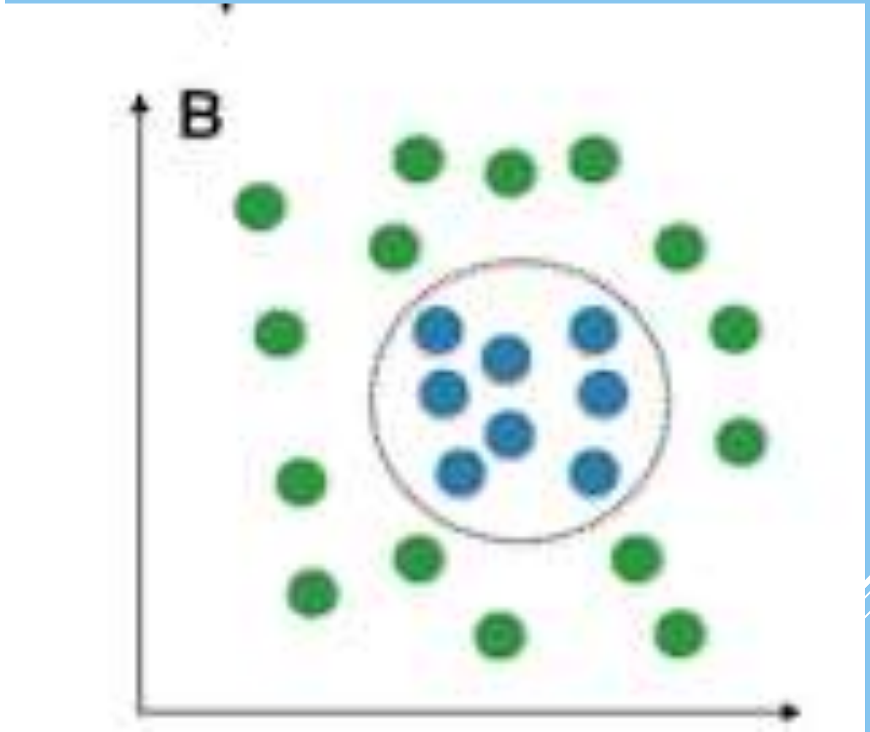



0
T
1





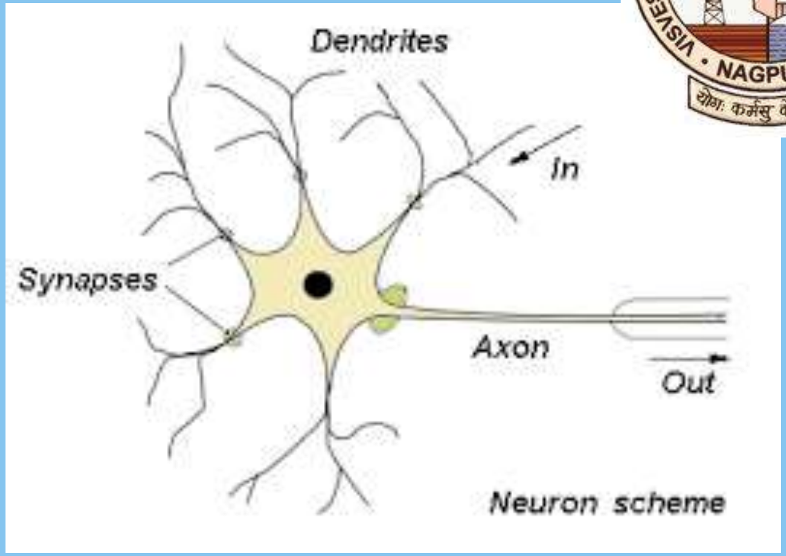

Nonlinear


B

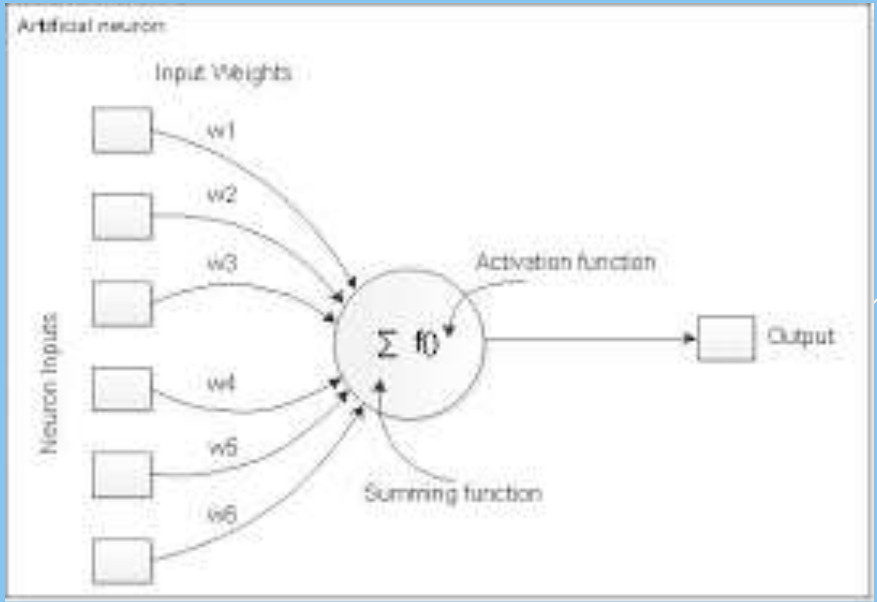
SOME REAL ANN USAGES

- Character recognition
- Image Compression
- Classification of Neurodegenerative Diseases
- sentiment analysis
- Forecasting

ARTIFICIAL NEURAL NETWORK

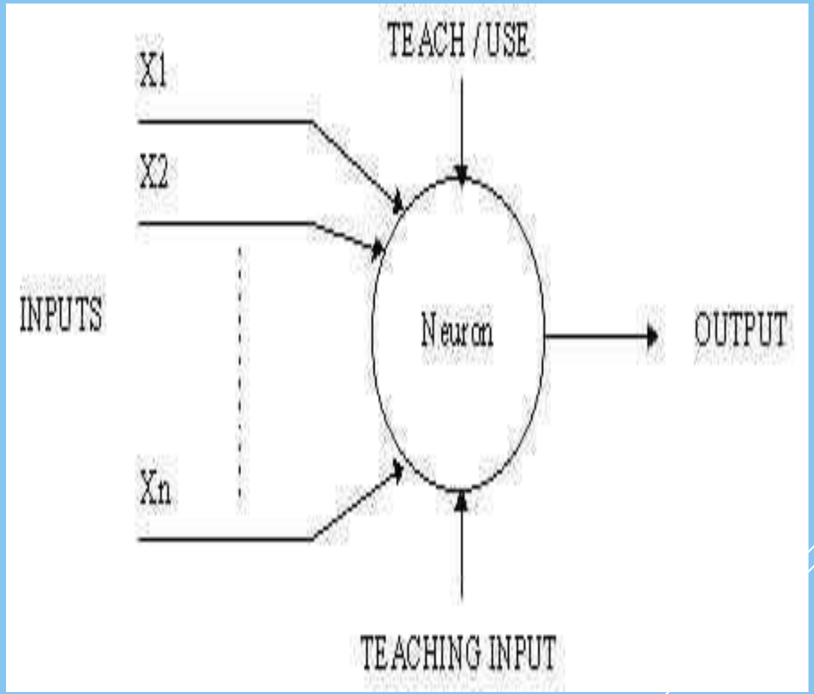


- Biologically inspired
- A network of simple processing elements

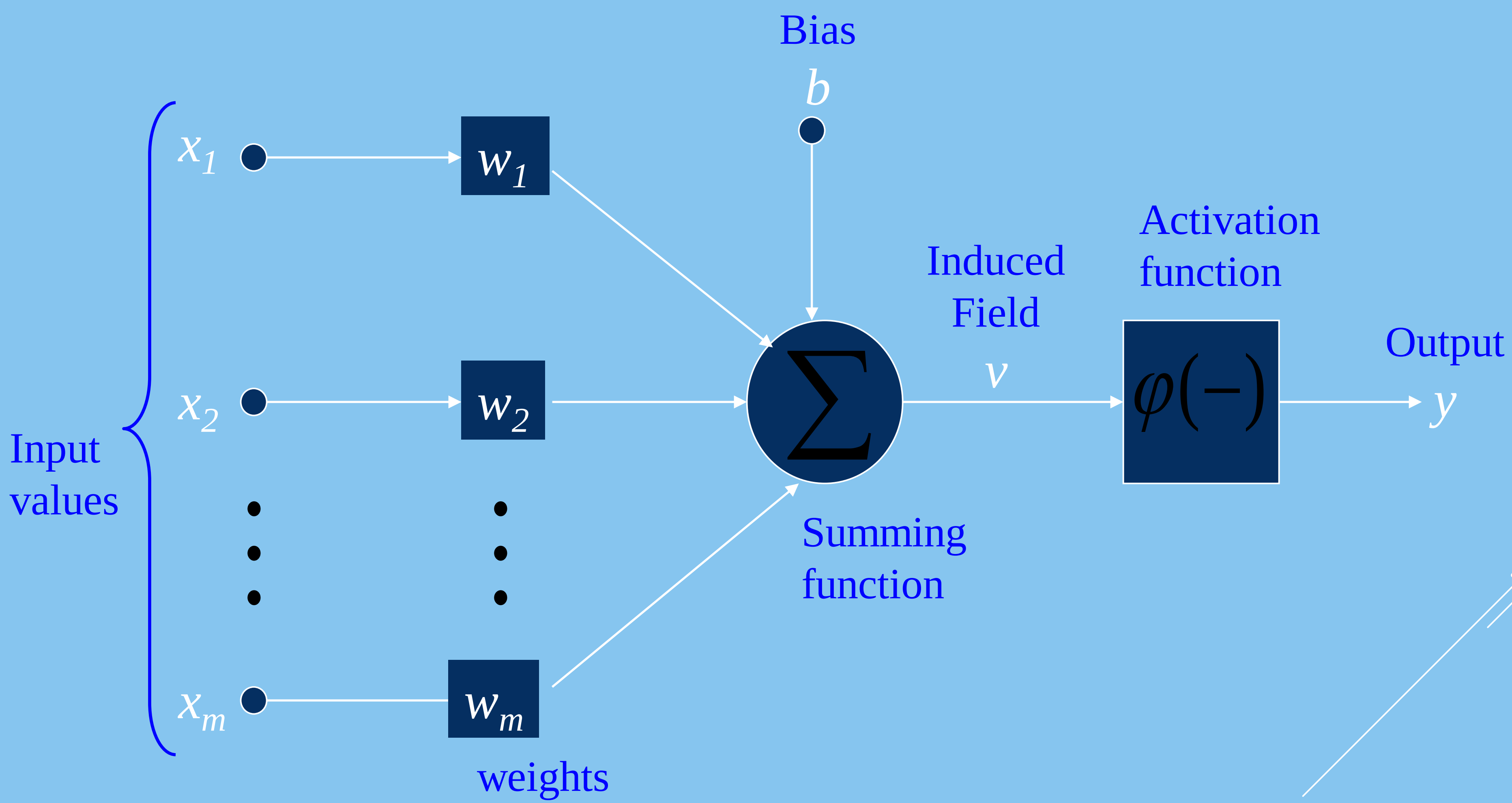


A SIMPLE NEURON

- Takes the Inputs .
- Calculate the summation of the Inputs .
- Compare it with the threshold being set during the learning stage.



THE NEURON DIAGRAM



NEURON

- The neuron is the basic information processing unit of a NN. It consists of:
  1. A set of links, describing the neuron inputs, with weights $W_1, W_2, \ldots, W_m$
  2. An adder function (linear com … he weighted sum of the inputs: $$\mathbf{u} = \sum_{j=1}^{m} \mathbf{w}_j \mathbf{x}_j$$
     (real numbers) $\boxed{\varphi}$
  3. Activation function for limiti … the neuron output. Here 'b' denot… $$y = \varphi(u + b)$$

BIAS OF A NEURON

- The bias ***b*** has the effect of applying a transformation to the weighted sum ***u***

$$v = u + b$$

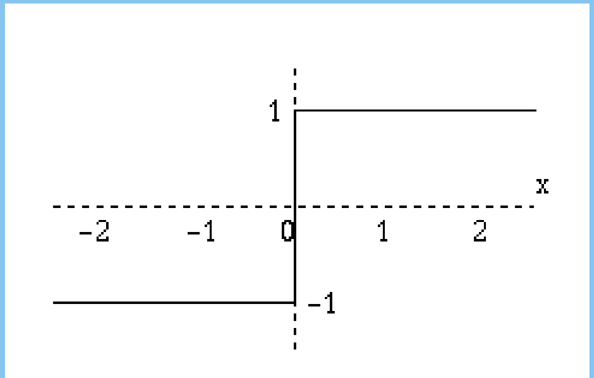
- The bias is an external parameter of the neuron. It can be modeled by adding an extra input.
- ***v*** is called **induced field** of the neuron

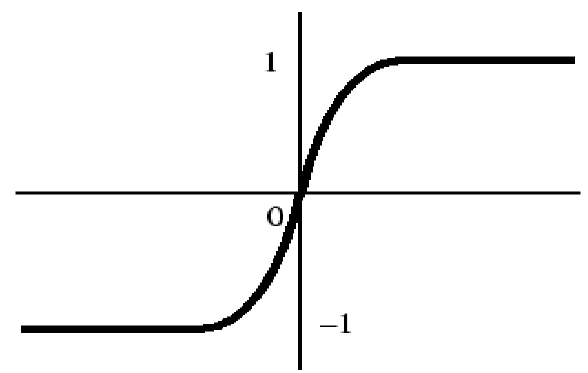
$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$

ACTIVATION FUNCTIONS

- Controls when unit is "active" or "inactive"
- Threshold function outputs 1 when input is positive and 0 otherwise
- Sigmoid function = 1 / (1 + e-x)





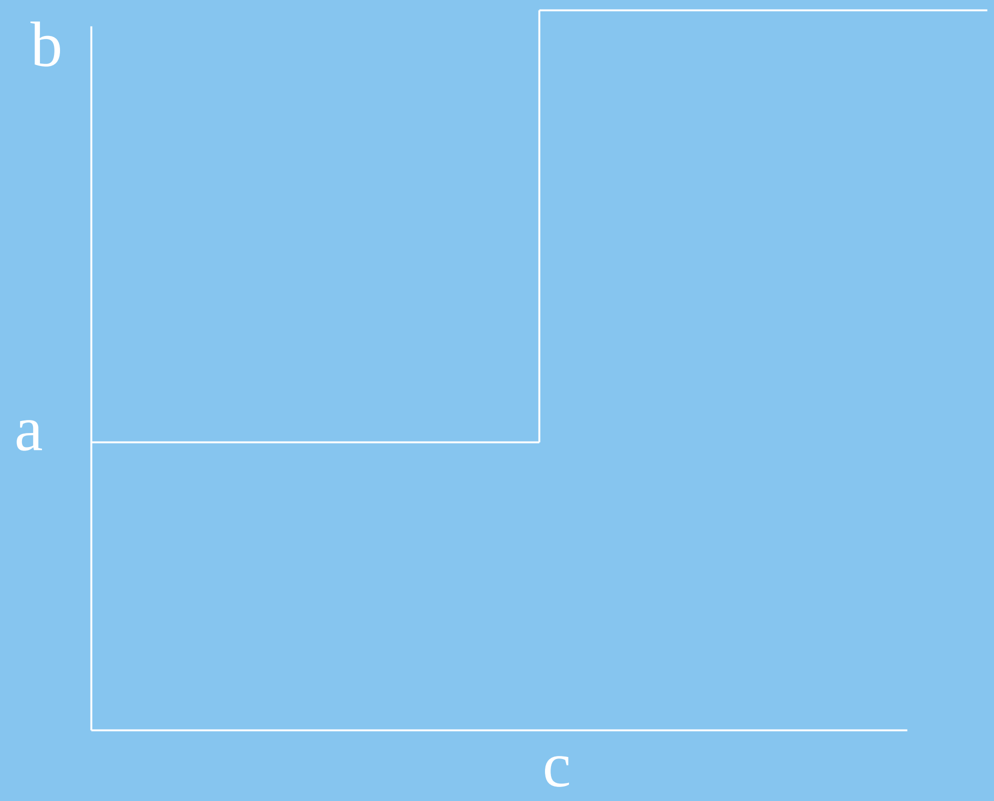
NEURON MODELS

- The choice of activation function φ determines the neuron model.

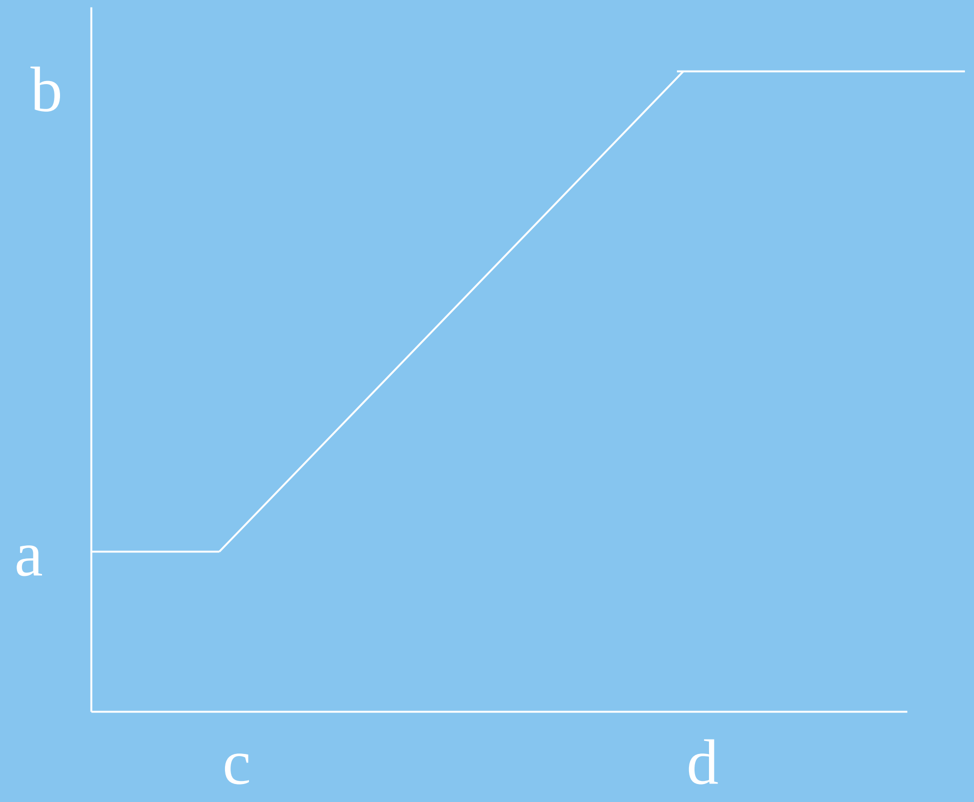
Examples:

- step function: $$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > c \end{cases}$$
- ramp function: $$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > d \\ a + ((v - c)(b - a)/(d - c)) & \text{otherwise} \end{cases}$$
- sigmoid function with z,x,y parameters $$\varphi(v) = z + \frac{1}{1 + \exp(-xv + y)}$$

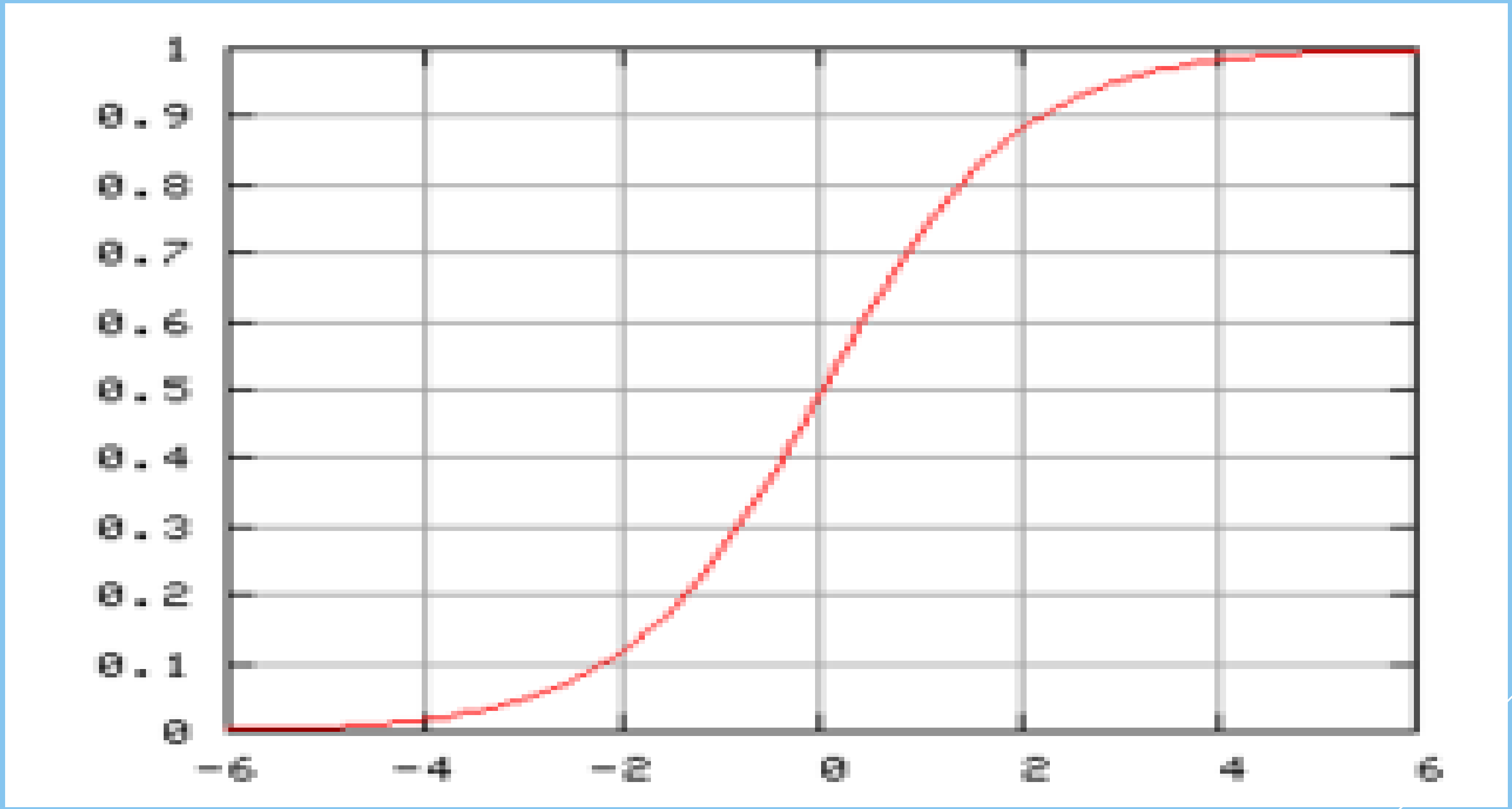
Step Function



Ramp Function

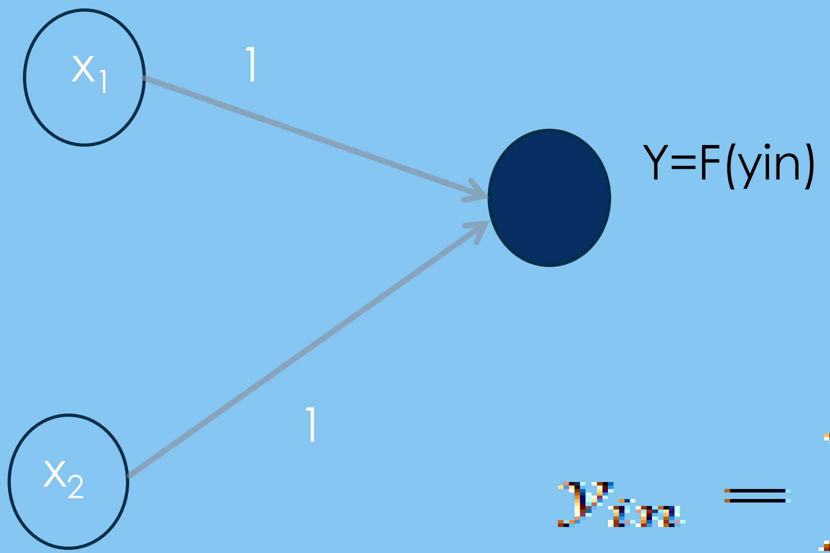


Sigmoid function



Mc-Culloch Pitts Model

AND Model

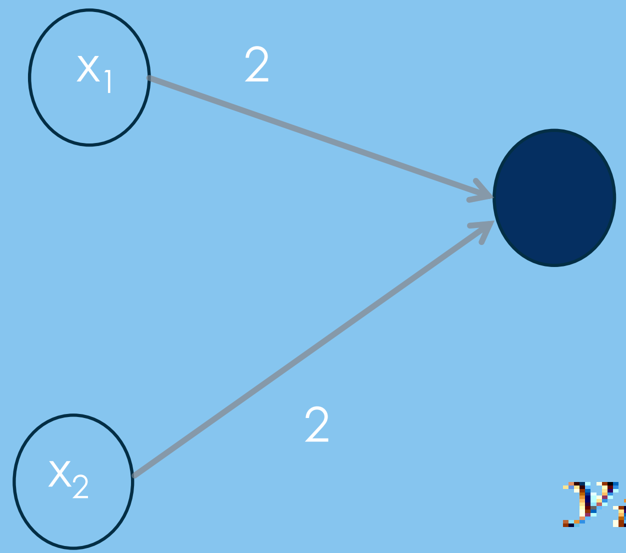


x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

$$y_{in} = \sum_i x_i * w_i$$

$$Y = f(y_{in}) = \begin{cases} 1 & if \quad y_{-in} \geq 2 \\ 0 & if \quad y_{-in} < 2 \end{cases}$$

OR Model



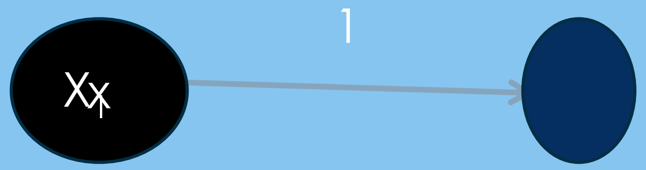
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

$$y_{in} = \sum_i x_i * w_i$$

$$Y = f(y_{in}) = \begin{cases} 1 & if \quad y_{-in} \geq 2 \\ 0 & if \quad y_{-in} < 2 \end{cases}$$

x_1

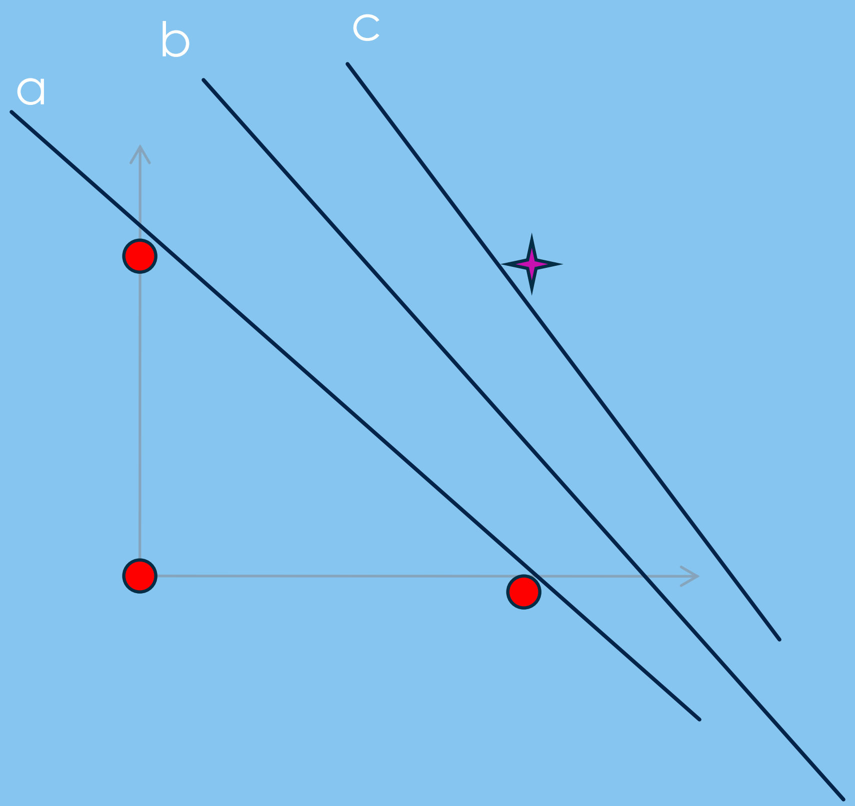
NOT Model



x	y
0	1
1	0

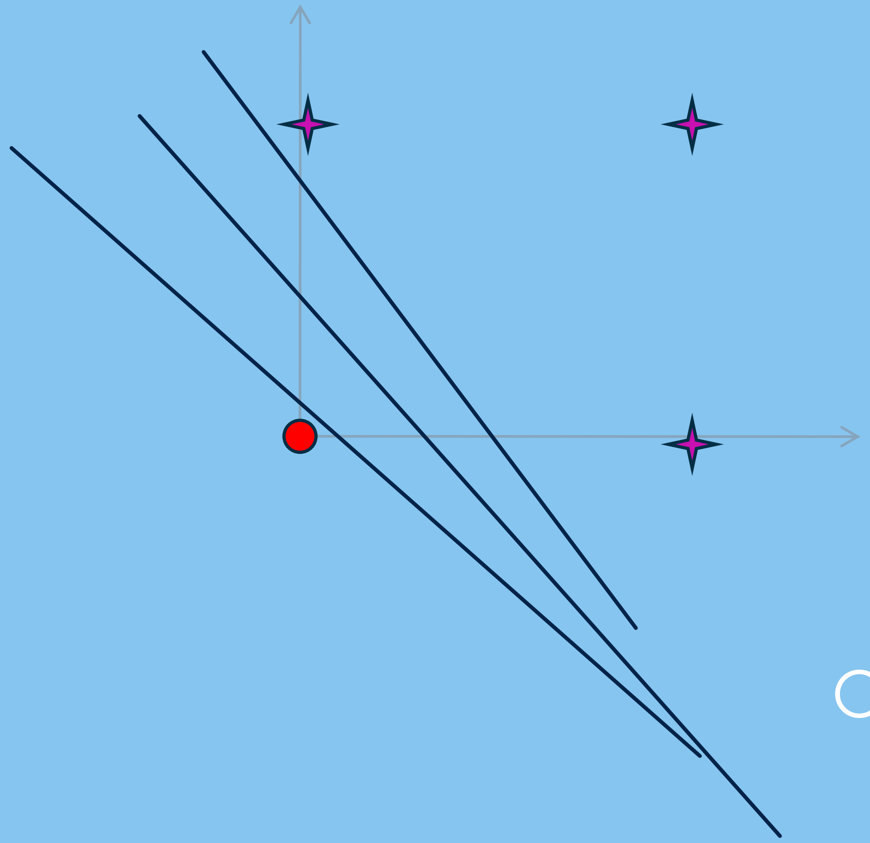
$$y_{in} = \sum_i x_i * w_i$$

$$Y = f(y_{in}) = \begin{cases} 1 & if \quad y_{-in} < 1 \\ 0 & if \quad y_{-in} \geq 1 \end{cases}$$



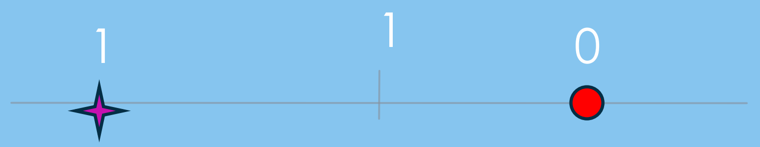
x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

AND Model



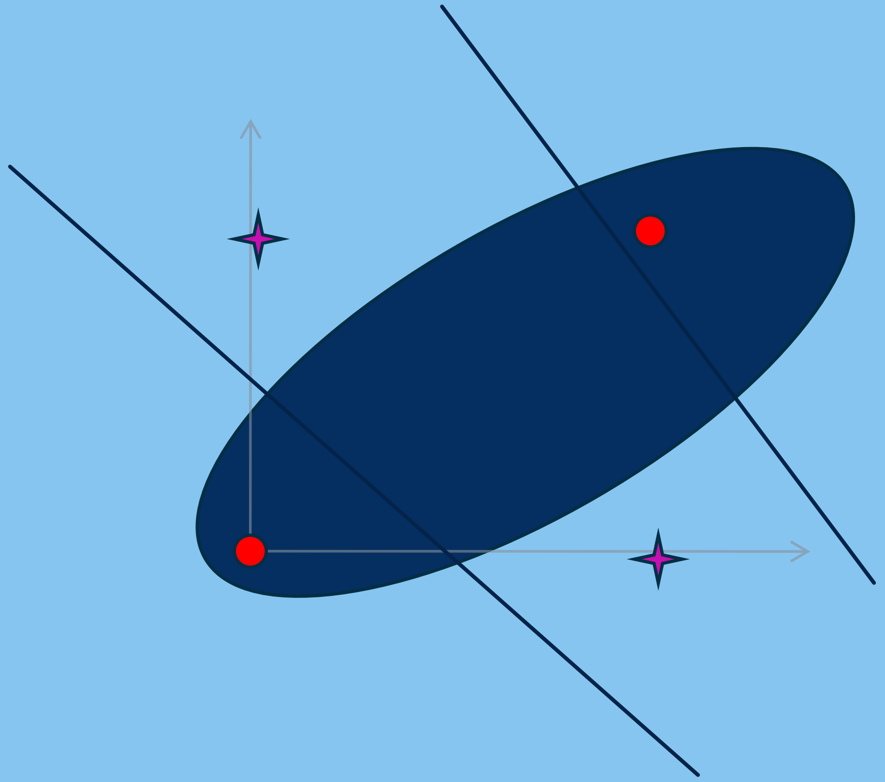
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

OR Model



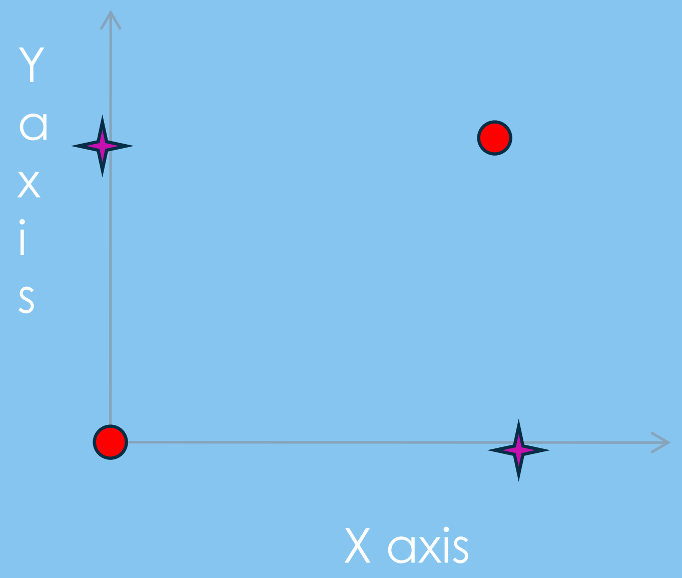
x	y
0	1
1	0

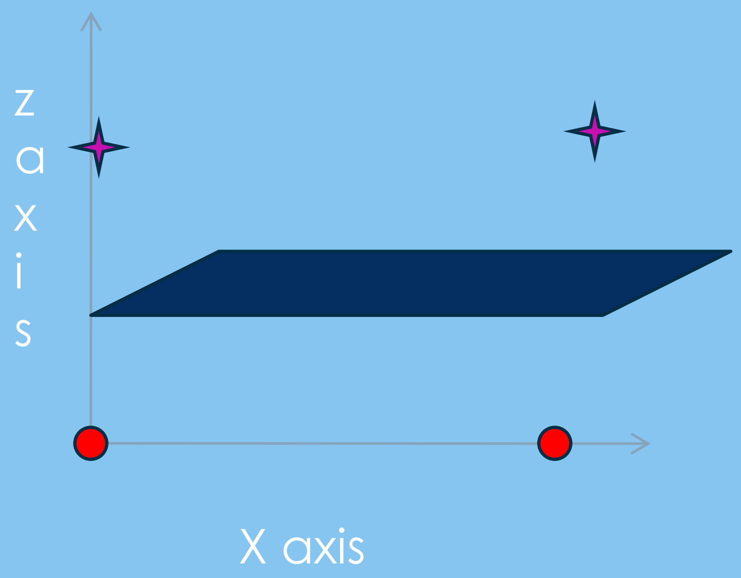
NOT Model

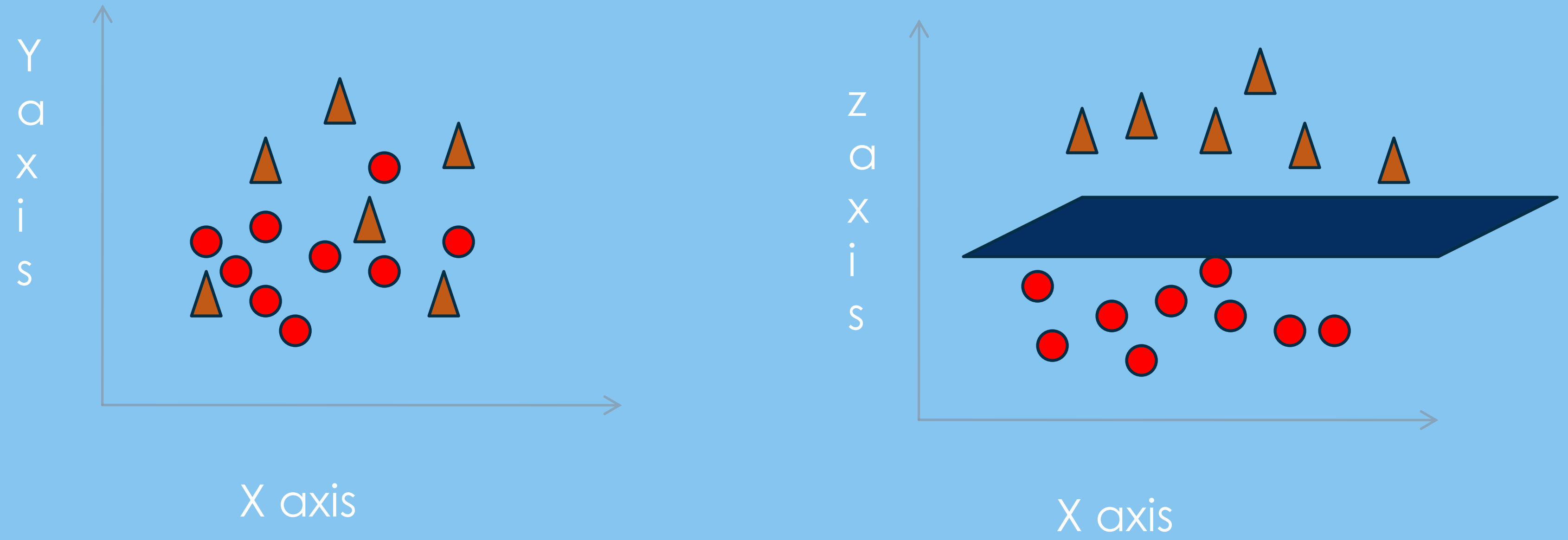


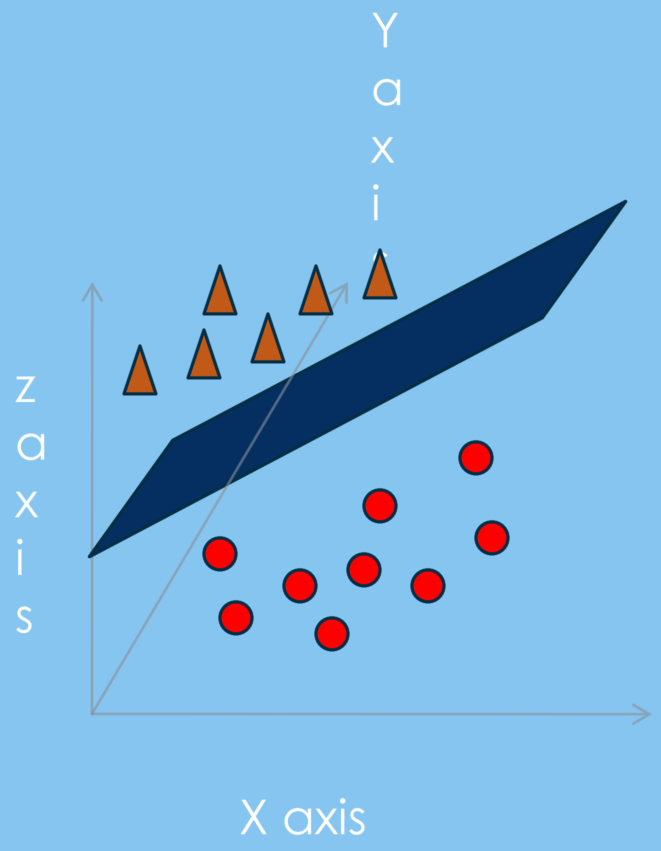
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

XOR Model

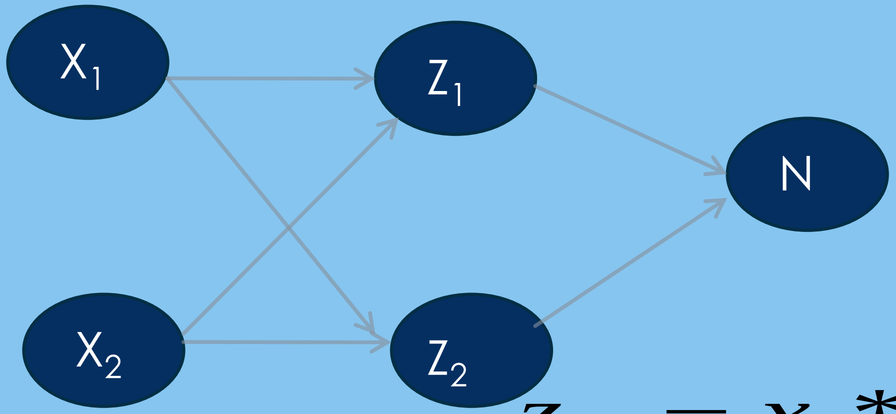

Y axis
X axis


z axis
X axis


Y axis
X axis
z axis
X axis


Yaxi
zaxis
X axis

XOR Model



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

$$z_{in1} = x_1 * w_{11} + x_2 * w_{21}$$

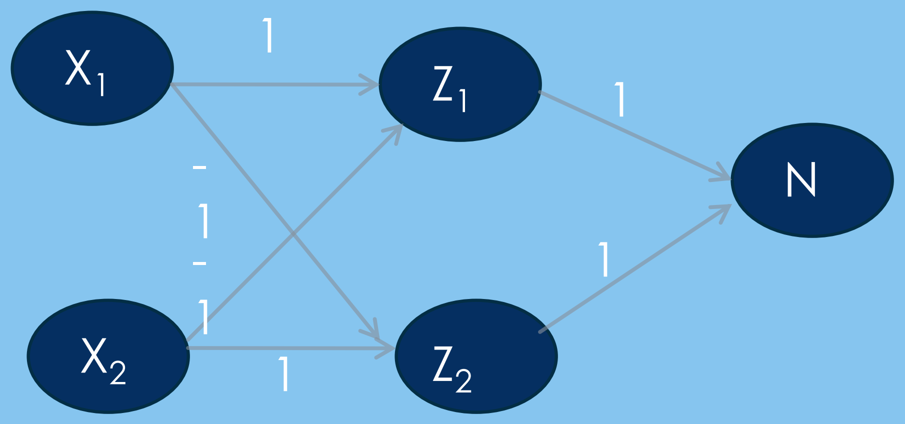
$$z_{in1} = x_1 * w_{12} + x_2 * w_{22}$$

$$z_1 = f(z_{in1}) = \begin{cases} 1\_if\_z_{in1} \geq 1 \\ 0\_if\_z_{in1} < 1 \end{cases}$$

$$z_2 = f(z_{in2}) = \begin{cases} 1\_if\_z_{in2} \geq 1 \\ 0\_if\_z_{in2} < 1 \end{cases}$$

$$Y_{in} = z_1 * v_1 + z_2 * v_2$$

$$Y = f(Y_{in}) = \begin{cases} 1\_if\_Y_{in} \geq 1 \\ 0\_if\_Y_{in} < 1 \end{cases}$$


X_1
X_2
Z_1
Z_2
N
1
-1
-1
1
1
1

HEBB ALGORITHM

Step 1: Initialize all weights and bias are set to zero
Step 2: For each training vector and target output pair (s, t) perform steps 3-6
Step 3: Set activations for input units with input vector
Step 4: Set activation for output unit with the output neuron.
Step 5: Adjust the weights by applying Hebb rule

$$w_i(new) = w_i(old) + x_i * y$$

Step 6: Adjust the bias

$$b(new) = b(old) + y$$

If net input is +ve the output is +ve
If the input is –ve the output is –ve

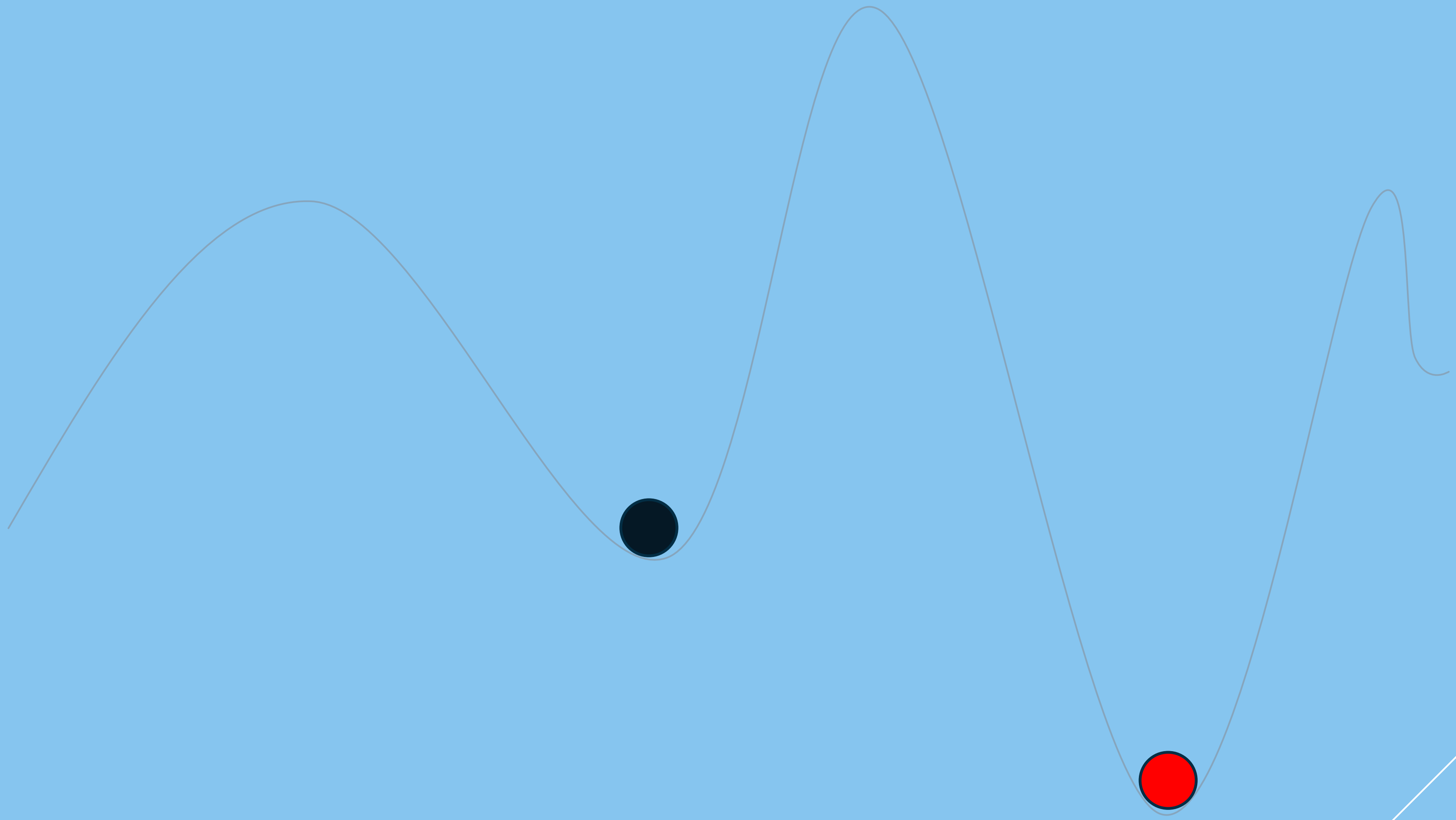
X_1	X_2	B	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

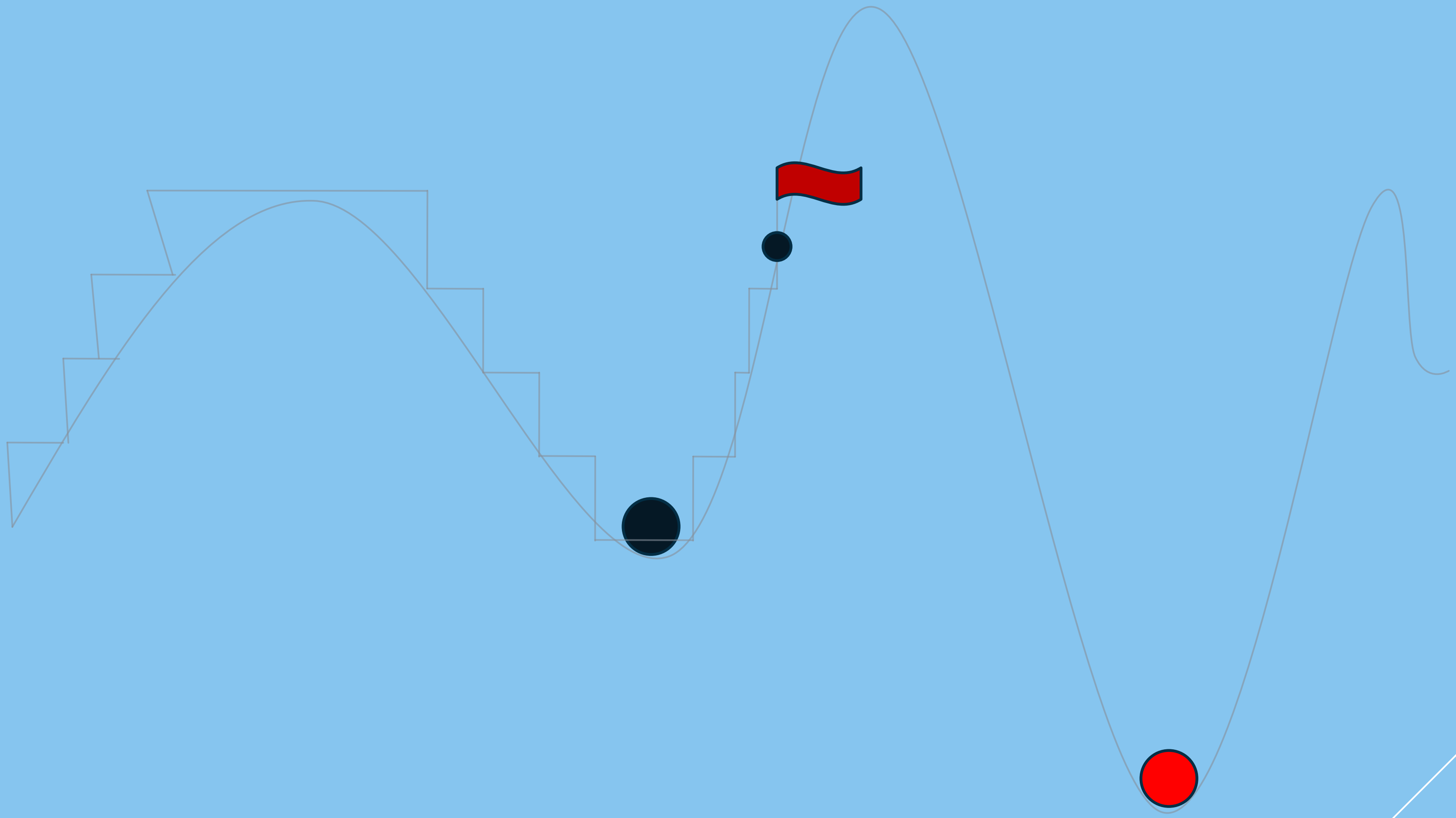
W_1=0, W_2 =0 and b=0

$$\Delta w_i = x_i * y$$

$$\Delta b = y$$

AND Model





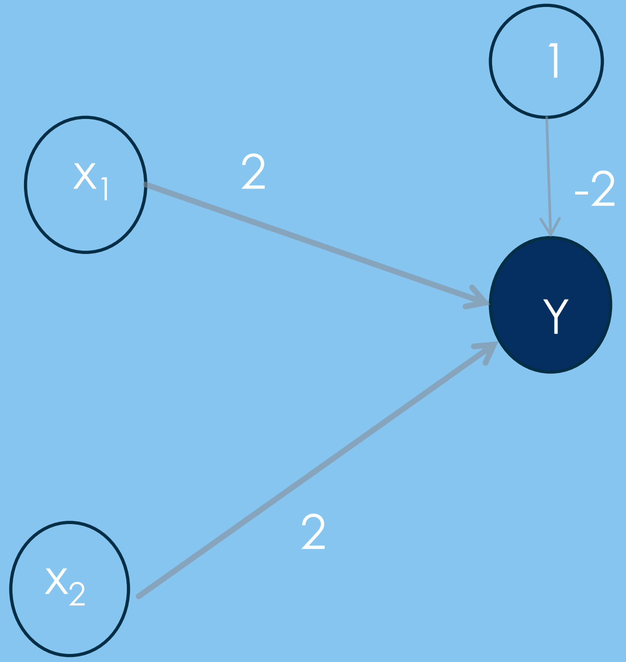
X_1	X_2	b	y	ΔW_1	ΔW_2	Δb	W_1	W_2	B
							0	0	0
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

$$\Delta w_1 = x_1 * y \qquad \Delta b = y$$

$$\Delta w_2 = x_2 * y$$

$$w_1(new) = w_1(old) + \Delta w_1$$

$$w_2(new) = w_2(old) + \Delta w_2$$



Example 2:

X_1	X_2	B	y
1	1	1	-1
1	-1	1	1
-1	1	1	1
-1	-1	1	-1

X_1	X_2	B	y	ΔW_1	ΔW_2	Δb	W_1	W_2	b
							0	0	0
1	1	1	-1	-1	-1	-1	-1	-1	-1
1	-1	1	1	1	-1	1	0	-2	0
-1	1	1	1	-1	1	1	-1	-1	1
-1	-1	1	-1	1	1	-1	0	0	0

Example 3:

X_1	X_2	X_3	X_4	B	y
1	1	-1	1	1	1
-1	1	-1	1	1	-1
1	-1	1	-1	1	1
-1	-1	1	1	1	-1

X_1	X_2	X_3	X_4	B	y	ΔW_1	ΔW_2	ΔW_3	ΔW_4	Δb	W_1	W_2	W_3	W_4	b
											0	0	0	0	0
1	1	-1	1	1	1	1	1	-1	1	1	1	1	-1	1	1
-1	1	-1	1	1	-1	1	-1	1	-1	-1	2	0	0	0	0
1	-1	1	-1	1	1	1	-1	1	-1	1	3	-1	1	-1	1
-1	-1	1	1	1	-1	1	1	-1	-1	-1	4	0	0	-2	0

Example 4:

*	*	*
	*	
*	*	*

*		
*		
*	*	*

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y
1	1	1	-1	1	-1	1	1	1	1
1	-1	-1	1	-1	-1	1	1	1	-1

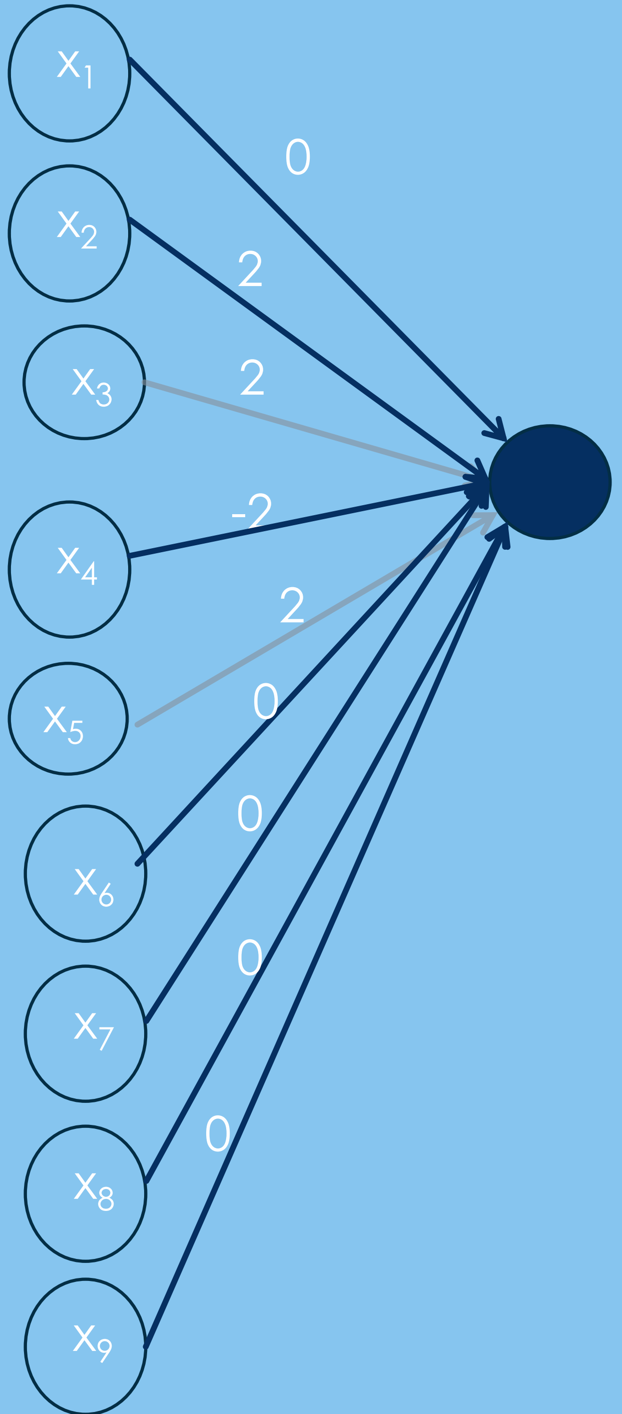
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y
1	1	1	-1	1	-1	1	1	1	1
1	-1	-1	1	-1	-1	1	1	1	-1

$$\Delta w_i = x_i * y$$

$$\Delta b = y$$

ΔW_1	ΔW_2	ΔW_3	ΔW_4	ΔW_5	ΔW_6	ΔW_7	ΔW_8	ΔW_9	Δb
1	1	1	-1	1	-1	1	1	1	1
-1	1	1	-1	1	1	-1	-1	-1	-1

W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Δb
0	0	0	0	0	0	0	0	0	0
1	1	1	-1	1	-1	1	1	1	1
0	2	2	-2	2	0	0	0	0	0



x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
0
2
2
-2
2
0
0
0
0

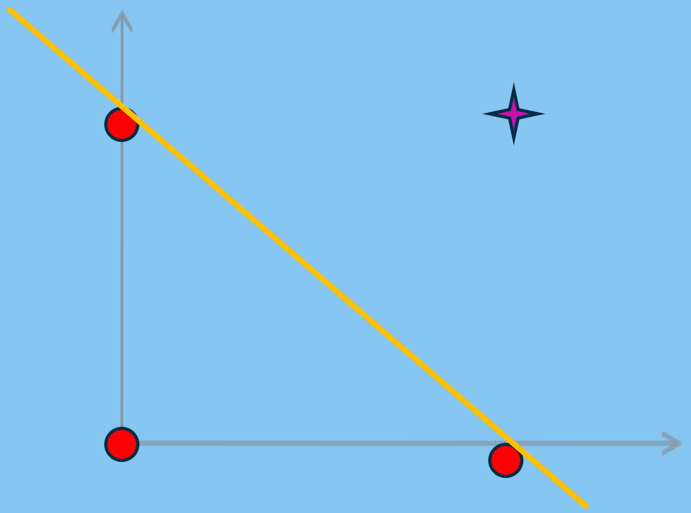
AND GATE

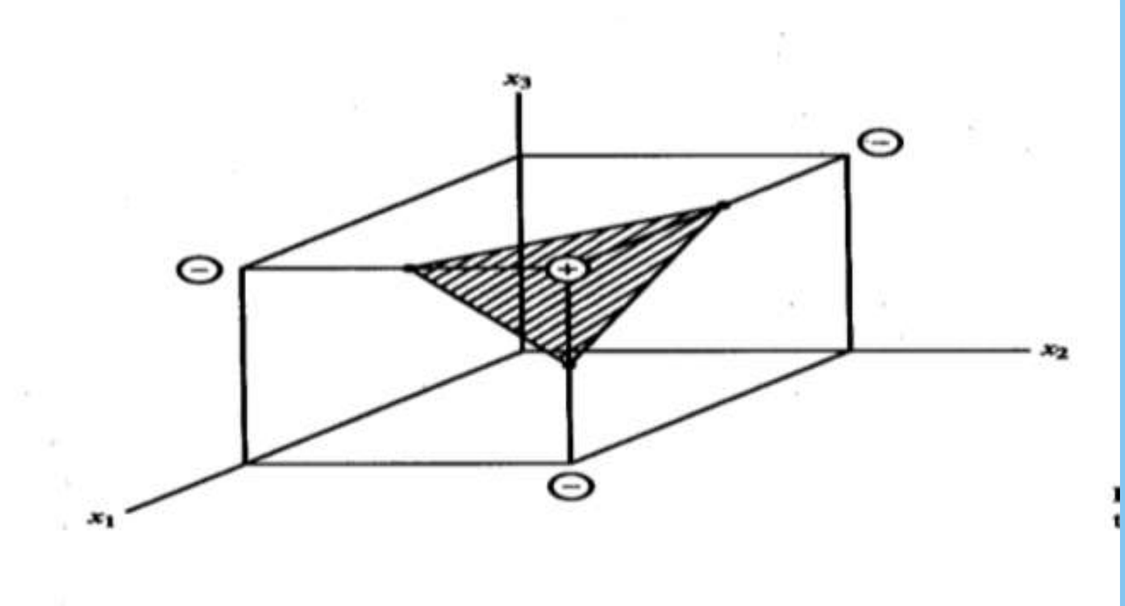
W_1	W_2	B
2	2	-2

$$x_1 * w_1 + x_2 * w_2 + b = 0$$

$$2x_1 + 2x_2 - 2 = 0$$

$$x_2 = 1 - x_1$$




x_3
x_2
x_1

PERCEPTRON RULE

Step 1: Initialize all weights and bias are set to zero. Set learning rate =0to 1.
Step 2: while stopping condition is false do step 3-7
Step 3: For each training vector and target output pair (s, t) perform steps 4-6
Step 4: Set activations for input units with input vector
Step 5: Compute the output unit response

$$Y_{in} = b + \sum_i x_i * w_i$$

$$Y = f(Y_{in}) = \begin{cases} 1\_if\_Y_{in} > \theta \\ 0\_if\_-\theta < Y_{in} < \theta \\ -1\_if\_Y_{in} < -\theta \end{cases}$$

Step 6: The weights and bias are updated if the target is not equal to the output response. If t≠y and value of x_i is not zero

$$w_i(new) = w_i(old) + \alpha * t * x_i$$

Adjust the bias

$$b(new) = b(old) + \alpha * t$$

Step 7: Test for stopping condition.

AND MODEL

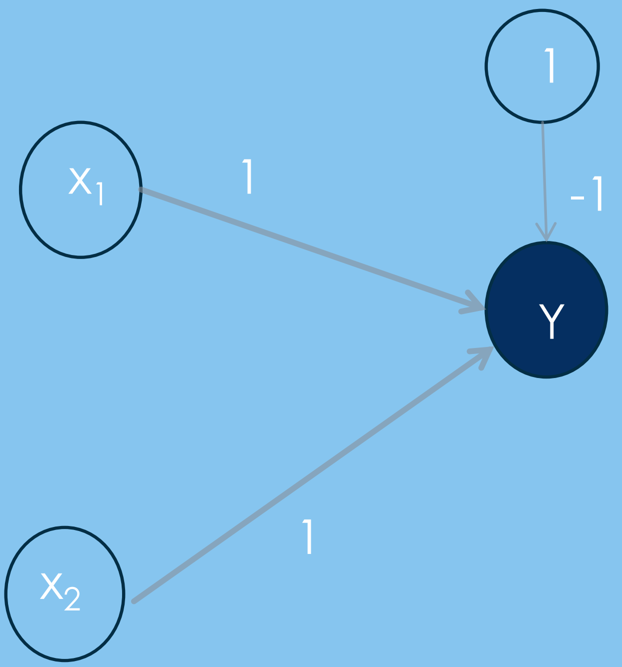
X_1	X_2	b	Y_{in}	y	t	ΔW_1	ΔW_2	Δb	W_1	W_2	B
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
1	-1	1	1	1	-1	1	-1	-1	2	0	0
-1	1	1	2	1	-1	-1	1	-1	1	1	-1
-1	-1	1	-3	-1	-1	0	0	0	1	1	-1

$$\Delta w_1 = \alpha * x_1 * t \qquad \Delta b = \alpha * y$$

$$\Delta w_2 = \alpha * x_2 * t$$

$$w_1(new) = w_1(old) + \Delta w_1$$

$$w_2(new) = w_2(old) + \Delta w_2$$



AND MODEL

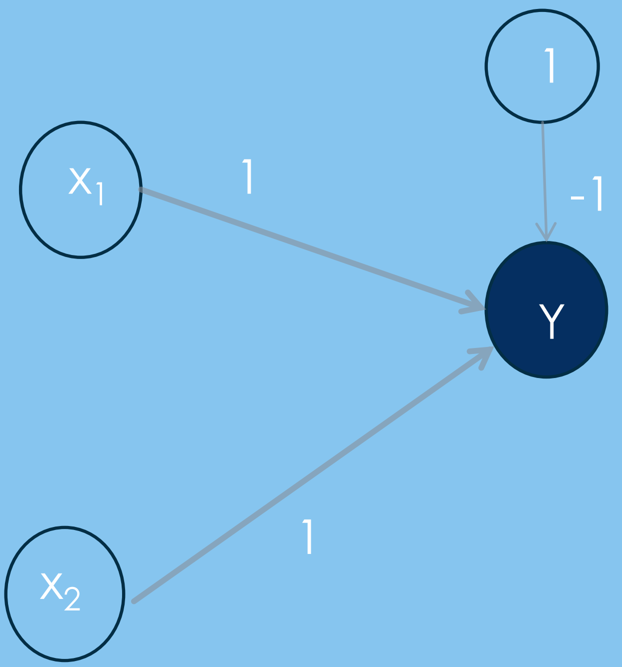
X_1	X_2	b	Y_{in}	y	t	ΔW_1	ΔW_2	Δb	W_1	W_2	B
									1	1	-1
1	1	1	0	0	1	1	1	1	1	1	-1
1	-1	1	1	-1	-1	1	-1	-1	1	1	-1
-1	1	1	2	-1	-1	-1	1	-1	1	1	-1
-1	-1	1	-3	-1	-1	0	0	0	1	1	-1

$$\Delta w_1 = \alpha * x_1 * t \qquad \Delta b = \alpha * y$$

$$\Delta w_2 = \alpha * x_2 * t$$

$$w_1(new) = w_1(old) + \Delta w_1$$

$$w_2(new) = w_2(old) + \Delta w_2$$



OR MODEL

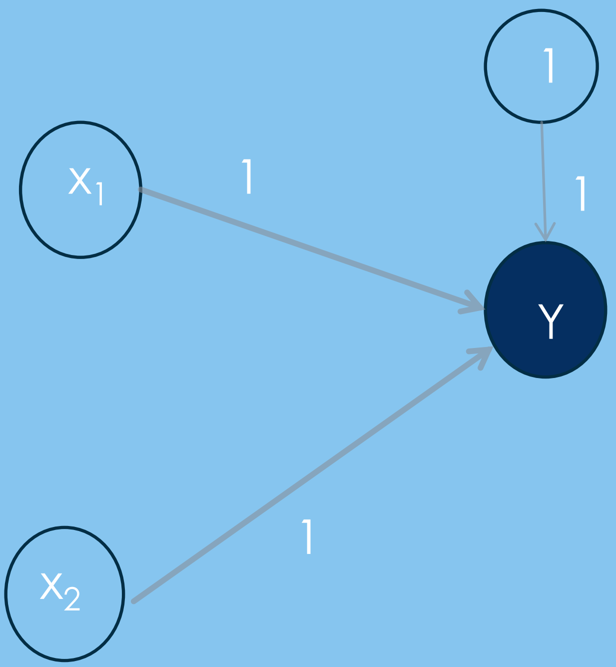
X_1	X_2	b	Y_{in}	y	t	ΔW_1	ΔW_2	Δb	W_1	W_2	B
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
1	-1	1	1	1	1	0	0	0	1	1	1
-1	1	1	1	1	1	0	0	0	1	1	1
-1	-1	1	-1	-1	-1	0	0	0	1	1	1

$$\Delta w_1 = \alpha * x_1 * t \qquad \Delta b = \alpha * y$$

$$\Delta w_2 = \alpha * x_2 * t$$

$$w_1(new) = w_1(old) + \Delta w_1$$

$$w_2(new) = w_2(old) + \Delta w_2$$



AND MODEL

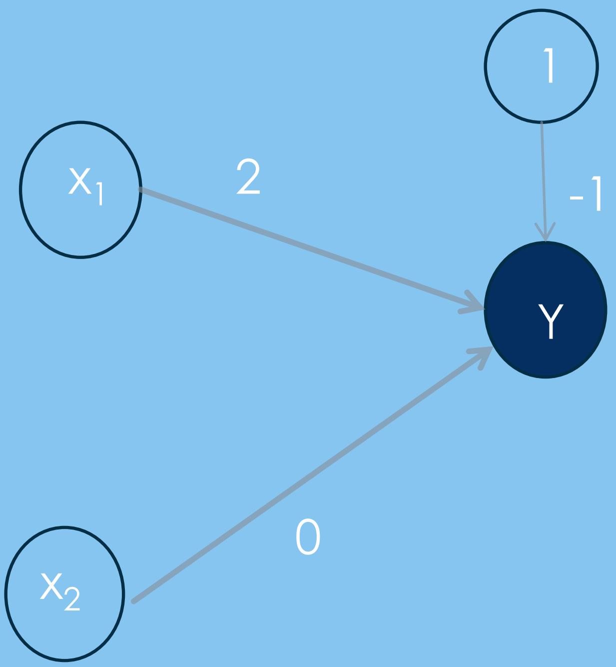
X_1	X_2	b	Y_{in}	y	t	ΔW_1	ΔW_2	Δb	W_1	W_2	B
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	1	0	-1	2	1	0
0	1	1	1	1	-1	0	-1	-1	2	0	-1
0	0	1	-1	-1	-1	0	0	0	2	0	-1

$$\Delta w_1 = \alpha * x_1 * t \qquad \Delta b = \alpha * y$$

$$\Delta w_2 = \alpha * x_2 * t$$

$$w_1(new) = w_1(old) + \Delta w_1$$

$$w_2(new) = w_2(old) + \Delta w_2$$



Example 3:

X_1	X_2	X_3	X_4	B	t
1	1	1	1	1	1
1	1	1	-1	1	-1
-1	1	-1	-1	1	1
1	-1	-1	1	1	-1

X_1	X_2	X_3	X_4	B	Y_{in}	y	t	ΔW_1	ΔW_2	ΔW_3	ΔW_4	Δb	W_1	W_2	W_3	W_4	b
													0	0	0	0	0
1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	-1	1	3	1	-1	-1	-1	-1	1	-1	0	0	0	2	0
-1	1	-1	-1	1	-2	-1	1	-1	1	-1	-1	1	-1	1	-1	1	1
1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-2	2	0	0	0

Adaline

Step 1: Initialize all weights and bias any small value other than zero. Set learning rate =0to 1.
Step 2: while stopping condition is false do step 3-7
Step 3: For each bipolar training vector and target output pair (s, t) perform steps 4-6
Step 4: Set activations for input units with input vector
Step 5: Compute the output unit response

$$Y_{in} = b + \sum_i x_i * w_i$$

$$Y = f(Y_{in}) = \begin{cases} 1\_if\ \_Y_{in} >= 0 \\ -1\_if\ \_Y_{in} < -0 \end{cases}$$

Step 6: The weights and bias are updated

$$w_i(new) = w_i(old) + \alpha * (t - y_{in}) * x_i$$

Adjust the bias

$$b(new) = b(old) + \alpha * (t - y_{in})$$

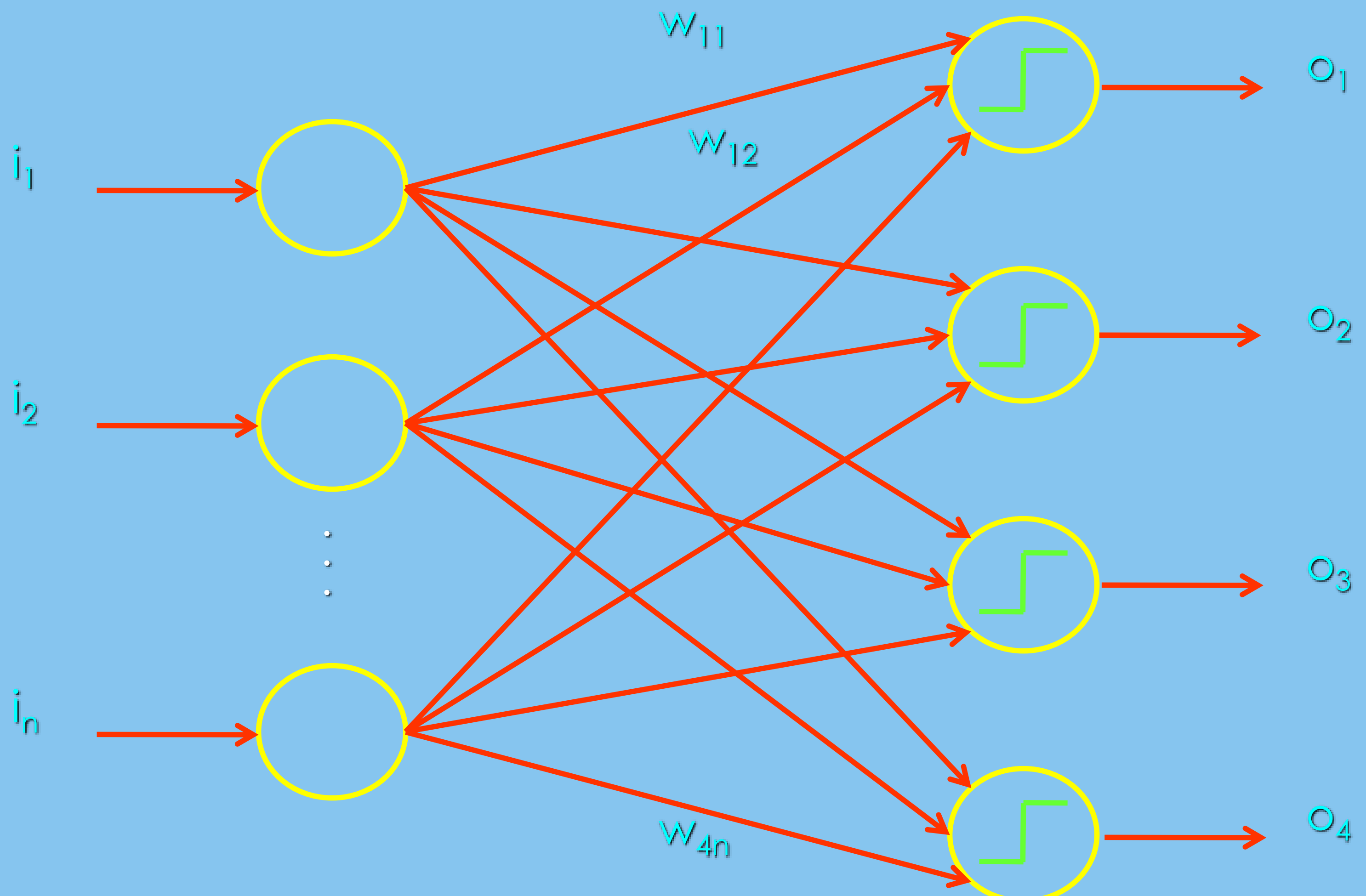
Step 7: Test for stopping condition.

X_1	X_2	b	Y_{in}	t	$t-Y_{in}$	ΔW_1	ΔW_2	Δb	W_1	W_2	B	$(t-Y_{in})^2$
									0.2	0.2	0.2	
1	1	1	0.6	1	0.4	0.08	0.08	0.08	0.28	0.28	0.28	0.16
1	-1	1	0.28	1	0.72	0.144	-0.144	0.144	0.424	0.136	0.424	0.51
-1	1	1	0.136	1	0.864	-0.173	0.173	0.173	0.251	0.309	0.597	0.74
-1	-1	1	0.037	1	0.963	-0.193	-0.193	0.963	0.0584	0.116	1.56	0.92

MULTICLASS DISCRIMINATION

- Often, our classification problems involve more than two classes.
- For example, character recognition requires at least 26 different classes.
- We can perform such tasks using layers of perceptrons or Adalines.

MULTICLASS DISCRIMINATION



- A four-node perceptron for a four-class problem in n-dimensional input space

MULTICLASS DISCRIMINATION

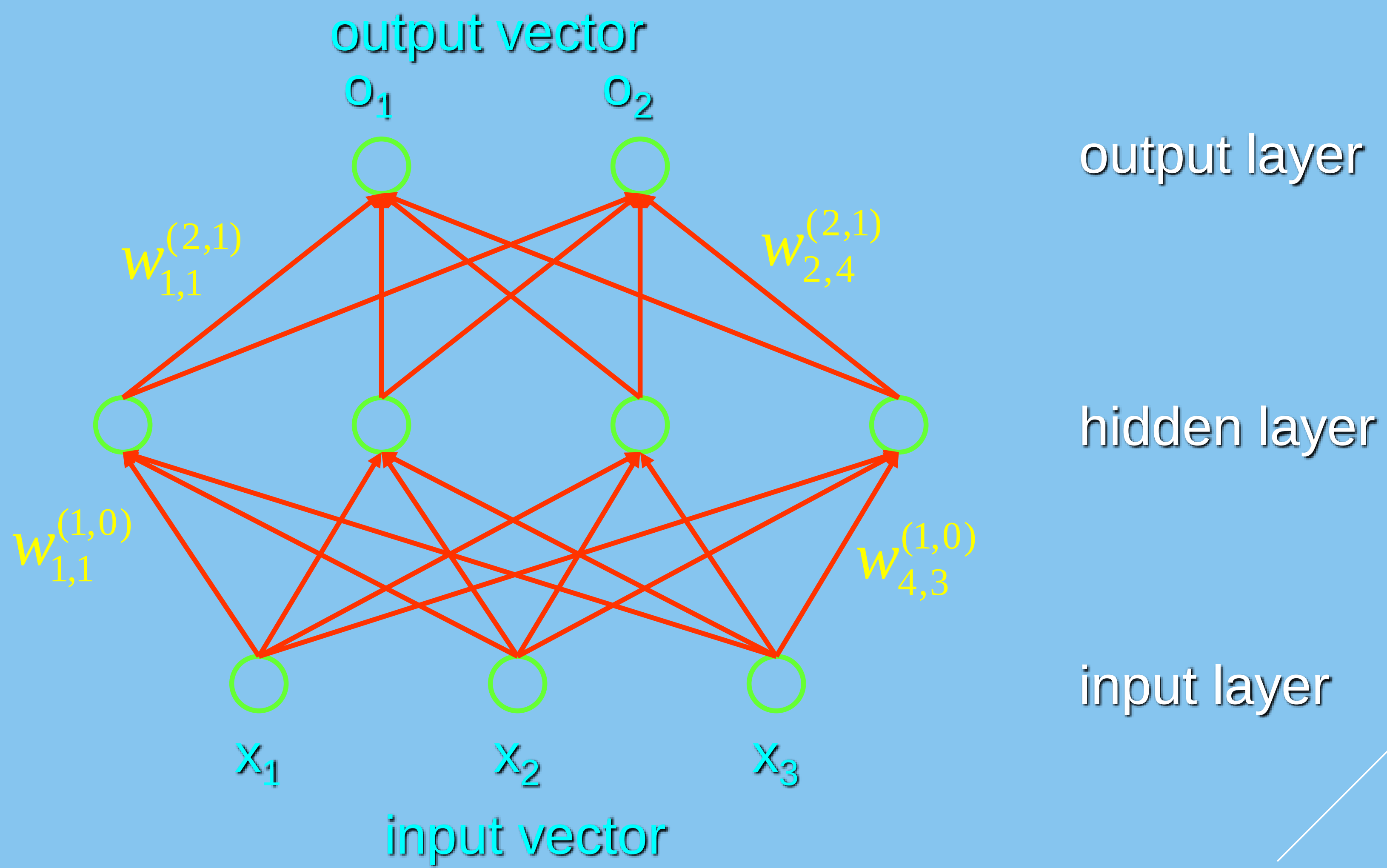
- Each perceptron learns to recognize one particular class, i.e., output 1 if the input is in that class, and 0 otherwise.
- The units can be trained separately and in parallel.
- In production mode, the network decides that its current input is in the k-th class if and only if $o_k = 1$, and for all $j \neq k$, $o_j = 0$, otherwise it is misclassified.
- For units with real-valued output, the neuron with maximal output can be picked to indicate the class of the input.
- This maximum should be significantly greater than all other outputs, otherwise the input is misclassified.

MULTILAYER NETWORKS

- Although single-layer perceptron networks can distinguish between any number of classes, they still require linear separability of inputs.
- To overcome this serious limitation, we can use multiple layers of neurons.
- Rosenblatt first suggested this idea in 1961, but he used perceptrons.
- However, their non-differentiable output function led to an inefficient and weak learning algorithm.
- The idea that eventually led to a breakthrough was the use of continuous output functions and gradient descent.

TERMINOLOGY

- **Example:** Network function f: $\mathbf{R}^3 \rightarrow \mathbf{R}^2$



MULTI LAYER ARTIFICIAL NEURAL NET

INPUT: records without class attribute with normalized attributes values.

INPUT VECTOR: $X = \{ x_1, x_2, \ldots, x_n \}$ where n is the number of (non-class) attributes.

INPUT LAYER: there are as many nodes as non-class attributes, i.e. as the length of the input vector.

HIDDEN LAYER: the number of nodes in the hidden layer and the number of hidden layers depends on implementation.

WEIGHT AND BIAS UPDATION

Per Sample Updating

- updating weights and biases after the presentation of each sample.

Per Training Set Updating (Epoch or Iteration)

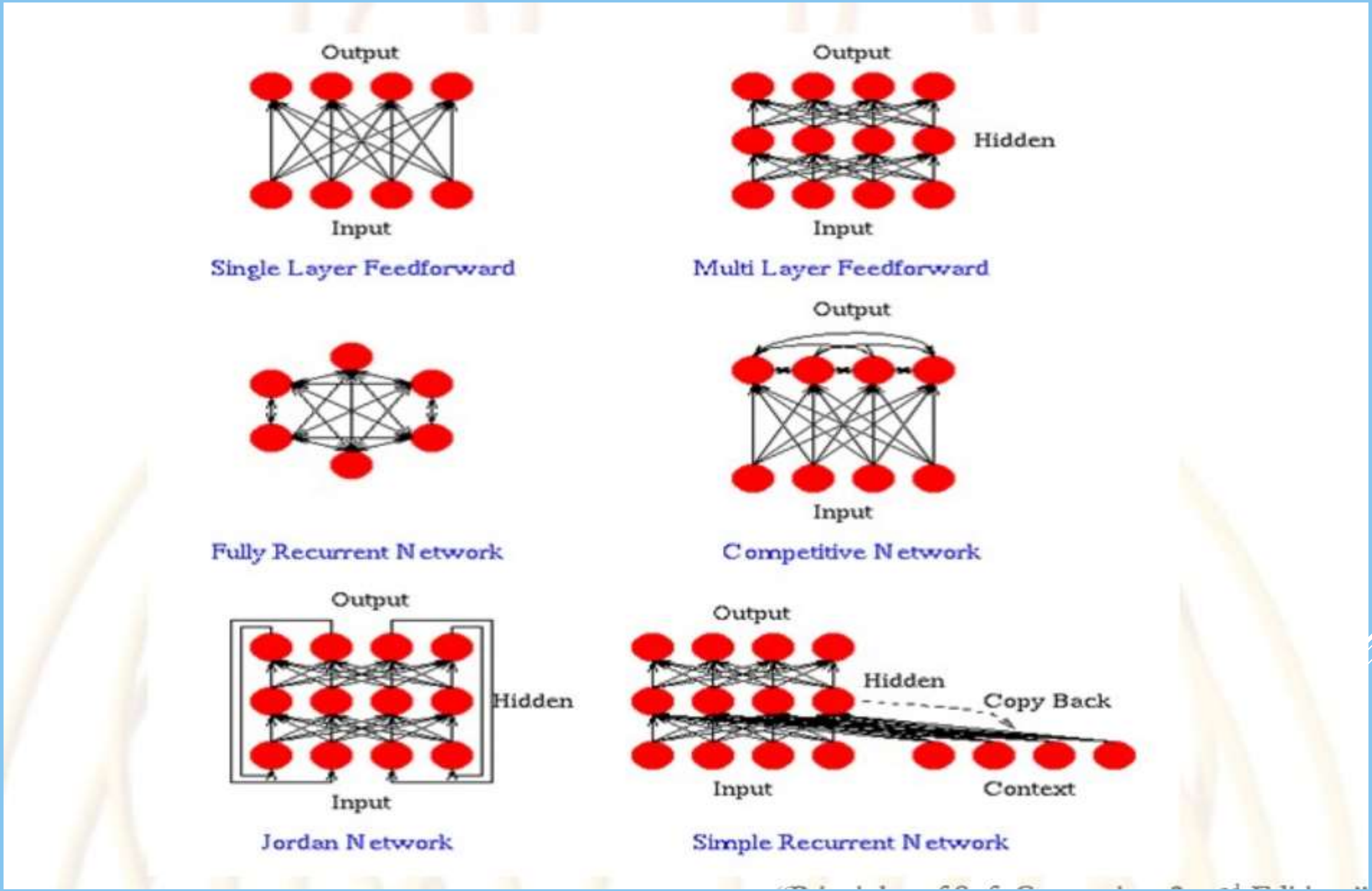
- weight and bias increments could be accumulated in variables and the weights and biases updated after all the samples of the training set have been presented.

STOPPING CONDITION

- All change in weights (Δwij) in the previous epoch are below some threshold, or
- The percentage of samples misclassified in the previous epoch is below some threshold, or
- A pre-specified number of epochs has expired.
- In practice, several hundreds of thousands of epochs may be required before the weights will converge.

BUILDING BLOCKS OF ARTIFICIAL NEURAL NET

- Network Architecture (Connection between Neurons)
- Setting the Weights (Training)
- Activation Function


Output
Input
Single Layer Feedforward
Output
Hidden
Input
Multi Layer Feedforward
Fully Recurrent Network
Output
Input
Competitive Network
Output
Hidden
Input
Jordan Network
Output
Hidden
Copy Back
Input
Context
Simple Recurrent Network

Madaline

Step 1: Initialize all weights and bias any small value other than zero. Set learning rate =0to 1.
Step 2: while stopping condition is false do step 3-7
Step 3: For each bipolar training vector and target output pair (s, t) perform steps 4-7
Step 4: Set activations for input units with input vector
Step 5: Compute the output unit response

$$Z_{inj} = b + \sum_i x_i * w_i$$

$$Z_j = f(Z_{inj}) = \begin{cases} 1\_if\_Z_{inj} >= 0 \\ -1\_if\_Z_{inj} < -0 \end{cases}$$

Step 6: Find the output of the net: $Y_{in} = b + \sum_i z_j * V_j$

$$Y = f(Y_{in}) = \begin{cases} 1\_if\_Y_{in} >= 0 \\ -1\_if\_Y_{in} < -0 \end{cases}$$

Step 7: Calculate the error and update the weights.

1. If t=y, no weight updation is required.

2. If t≠y and t=+1, update the weights on Z_j, where net input is closest to 0(zero).

$$w_i(new) = w_i(old) + \alpha * (1 - z_{inj}) * x_i$$

$$b(new) = b(old) + \alpha * (1 - z_{inj})$$

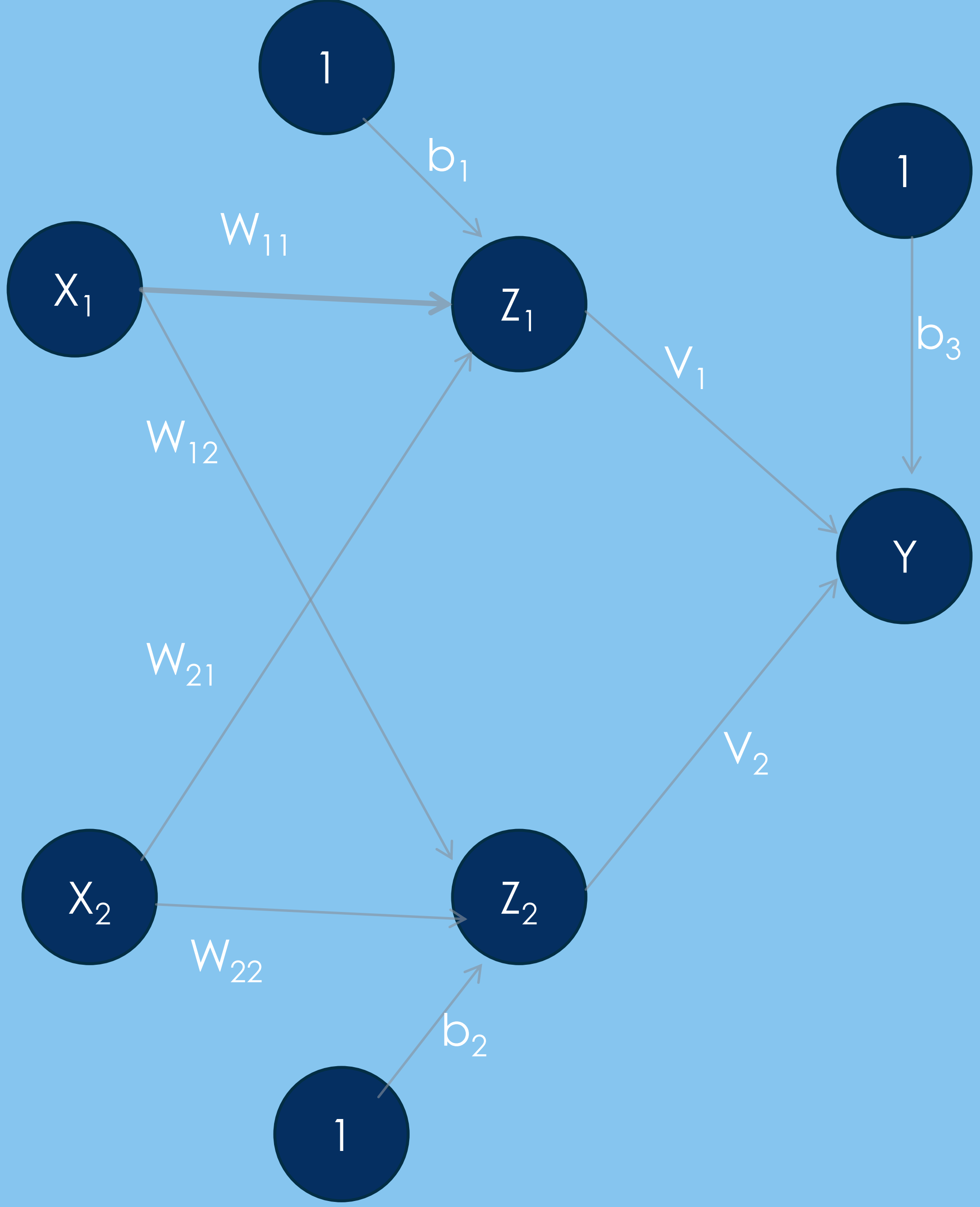
3. If t≠y and t=-1, update the weights on Z_j, where net input is closest to 0(zero).

$$w_i(new) = w_i(old) + \alpha * (-1 - z_{inj}) * x_i$$

$$b(new) = b(old) + \alpha * (-1 - z_{inj})$$

Step 8: Test for the stopping condition.

$W_{11} = 0.05$
$W_{12} = 0.1$
$W_{21} = 0.2$
$W_{22} = 0.2$
$V_1 = 0.5$
$V_2 = 0.5$
$b_1 = 0.3$
$b_2 = 0.15$
$b_3 = 0.5$



X_1	X_2	B	y
1	1	1	-1
1	-1	1	1
-1	1	1	1
-1	-1	1	-1

$$Z_{in1} = b_1 + x_1 w_{11} + x_1 w_{21}$$
$$Z_{in2} = b_2 + x_2 w_{12} + x_2 w_{22}$$

$$Z_1 = f(Z_{in1})$$
$$Z_2 = f(Z_{in2})$$
$$Y_{in} = b_2 + z_1 V_1 + z_2 V_2$$
$$Y = f(Y_{in})$$

$$w_i(new) = w_i(old) + \alpha * (1 - z_{inj}) * x_i$$

$$b(new) = b(old) + \alpha * (1 - z_{inj})$$

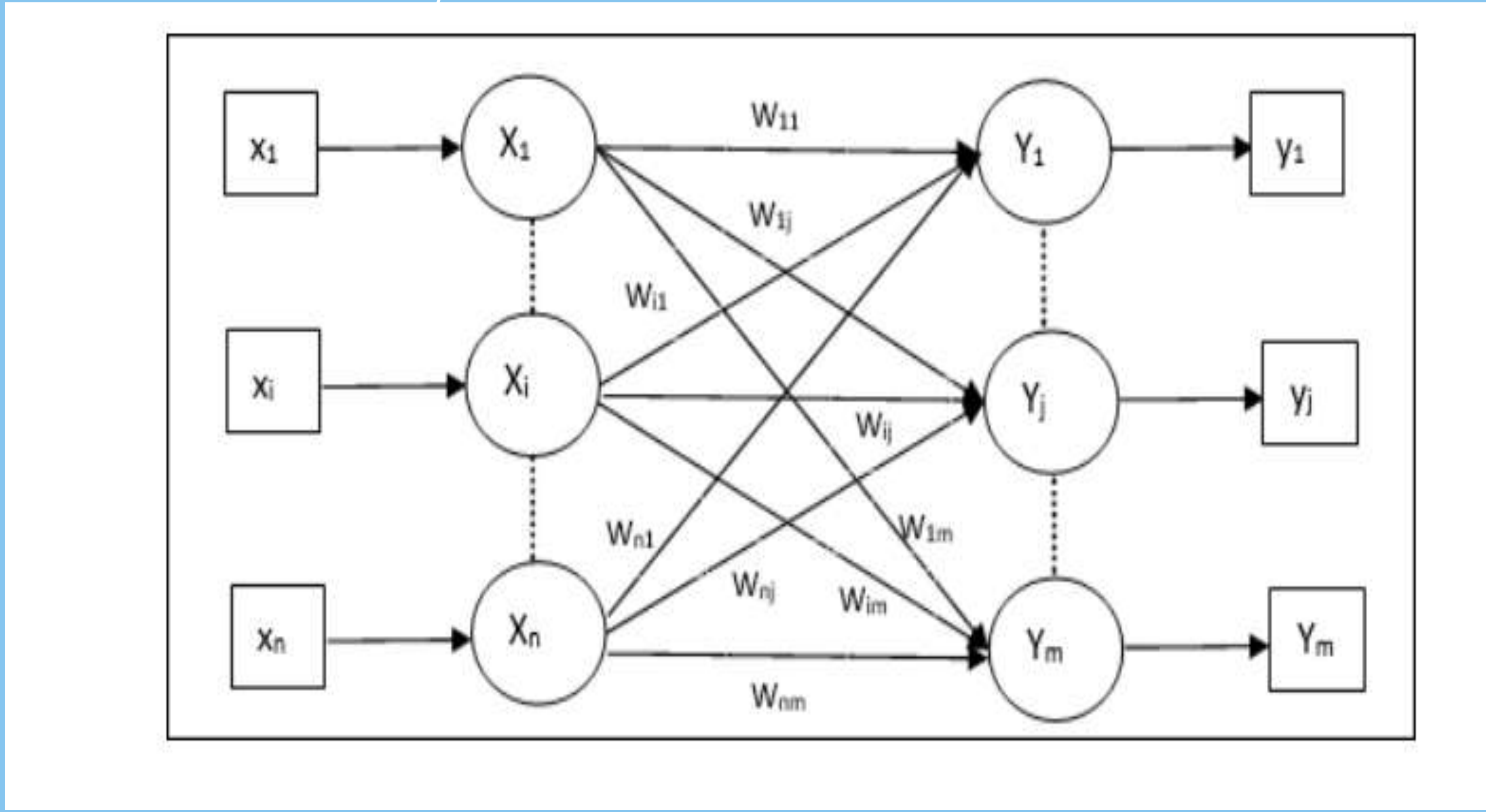
Hetero Associative Memory Neural Networks

Step 1: Initialize all weights using Hebb or Delta rule.
Step 2: whiFor each input vector do step 3-5
Step 3: For each bipolar training vector and target output pair (s, t) perform steps 4-7
Step 4: Set activations for input units with input vector
Step 5: Compute net input to the output units

$$Y_{inj} = \sum_i x_i * w_{ij}$$

$$Y_j = \begin{cases} 1\_if\_Y_{inj} > 0 \\ 0\_if\_Y_{inj} = 0 \\ -1\_if\_Z_{inj} < 0 \end{cases}$$

Hetero Associative Memory Neural Networks



Example 1: A hetero associative neural network is trained by Hebb outer product rule for input row vector S=(x1,x2,x3,x4) to the output row vectors t=(t1,t2). Find the weight matrix.

S1=(1 1 0 0) t1=(1 0)
S2=(1 1 1 0) t2=(0 1)
S3=(0 0 1 1) t3=(1 0)
S4=(0 1 0 0) t4=(1 0)

Step 1: Initialize all weights to zero
Step 2: Find the output for each input

S1=(1 1 0 0) t1=(1 0)

$$w_1 = S_1^t * t_1$$

$$= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Step 2: Find the output for each input

S2=(1 1 1 0)   t2=(0 1)

$$w_2 = S_2^t * t_2$$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Step 2: Find the output for each input

S3=(0 0 1 1)   t3=(1 0)

$$w_3 = S_3^t * t_3$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Step 2: Find the output for each input

S4=(0 1 0 0) t4=(1 0)

$$w_4 = S_4^t * t_4$$

$$= \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Step 3Weight matrix of all the four patterns is the sum of the weight matrix for each stored pattern

$$w = w_1 + w_2 + w_3 + w_4$$

$$= \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Example 2: A hetero associative neural network is trained by Hebb outer product rule for input row vector S=(x1,x2,x3,x4) to the output row vectors t=(t1,t2). Find the weight matrix.

S1=(1 1 0 0) t1=(1 0)
S2=(0 1 0 0) t2=(1 0)
S3=(0 0 1 1) t3=(0 1)
S4=(0 0 1 0) t4=(0 1)

Step 1: Initialize all weights to zero
Step 2: Find the output for each input

S1=(1 1 0 0) t1=(1 0)

$$w_1 = S_1^t * t_1$$

$$= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Step 2: Find the output for each input

S2=(0 1 0 0)   t2=(1 0)

$$w_2 = S_2^t * t_2$$

$$= \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Step 2: Find the output for each input

S3=(0 0 1 1)   t3=(0 1)

$$w_3 = S_3^t * t_3$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Step 2: Find the output for each input

S4=(0 0 1 0) t4=(0 1)

$$w_4 = S_4^t * t_4$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Step 3Weight matrix of all the four patterns is the sum of the weight matrix for each stored pattern

$$w = w_1 + w_2 + w_3 + w_4$$

$$= \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 0 & 2 \\ 0 & 1 \end{bmatrix}$$

Test the weight using different input set:

$$y_{inj} = \sum_{i=1}^{4} x_i * w_{ij}$$

$$y_{in1} = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41}$$

$$y_{in1} = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42}$$

S1=(1 1 0 0) t1=(1 0)

$$= \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 0 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

S2=(0 1 0 0) t2=(1 0)

$$= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 0 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

S3=(0 0 1 1) t3=(0 1)

$$=\begin{bmatrix}0 & 0 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ 2 & 0\\ 0 & 2\\ 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 3\end{bmatrix}=\begin{bmatrix}0 & 1\end{bmatrix}$$

S4=(0 0 1 0) t4=(0 1)

$$=\begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 2 & 0\\ 0 & 2\\ 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 2\end{bmatrix}=\begin{bmatrix}0 & 1\end{bmatrix}$$

Test the weight using different input set:

S3=(1 1 1 0) t3=?

$$=\begin{bmatrix}1 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 2 & 0\\ 0 & 2\\ 0 & 1\end{bmatrix}=\begin{bmatrix}3 & 2\end{bmatrix}=\begin{bmatrix}1 & 1\end{bmatrix}$$

Example 3: A hetero associative neural network is trained by Hebb outer product rule for input row vector S=(x1,x2,x3,x4) to the output row vectors t=(t1,t2). Find the weight matrix.

S1=(1 1 -1 -1) t1=(1 -1)
S2=(-1 1 -1 -1) t2=(1 -1)
S3=(-1 -1 1 1) t3=(-1 1)
S4=(-1 -1 1 -1) t4=(-1 1)

Step 1: Initialize all weights to zero
Step 2: Find the output for each input
S1=(1 1 -1 -1) t1=(1 -1)

$$w_1 = S_1^t * t_1$$

$$= \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Step 2: Find the output for each input

S2=(-1 1 -1 -1)   t2=(1 -1)

$$w_2 = S_2^t * t_2$$

$$= \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Step 2: Find the output for each input

S3=(-1 -1 1 1)   t3=(-1 1)

$$w_3 = S_3^t * t_3$$

$$= \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Step 2: Find the output for each input

S4=(-1 -1 1 -1) t4=(-1 1)

$$w_4 = S_4^t * t_4$$

$$= \begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}$$

Step 3Weight matrix of all the four patterns is the sum of the weight matrix for each stored pattern

$$w = w_1 + w_2 + w_3 + w_4$$

$$= \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix}$$

Test the weight using different input set:

$$y_{inj} = \sum_{i=1}^{4} x_i * w_{ij}$$

$$y_{in1} = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41}$$

$$y_{in1} = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42}$$

S1=(1 1 -1 -1) t1=(1 -1)

$$= [1 \quad 1 \quad -1 \quad -1] \begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix} = [12 \quad 0] = [1 \quad 0]$$

S2=(-1 1 -1 -1) t2=(1 -1)

$$= [-1 \quad 1 \quad -1 \quad -1] \begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix} = [8 \quad 0] = [1 \quad 0]$$

S3=(-1 -1 1 1) t3=(-1 1)

$$=\begin{bmatrix}-1 & -1 & 1 & 1\end{bmatrix}\begin{bmatrix}2 & -2\\ 4 & -4\\ -4 & 4\\ -2 & 2\end{bmatrix}=\begin{bmatrix}-12 & 12\end{bmatrix}=\begin{bmatrix}0 & 1\end{bmatrix}$$

S4=(-1 -1 1 -1) t4=(-1 1)

$$=\begin{bmatrix}-1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix}2 & -2\\ 4 & -4\\ -4 & 4\\ -2 & 2\end{bmatrix}=\begin{bmatrix}-8 & 8\end{bmatrix}=\begin{bmatrix}0 & 1\end{bmatrix}$$

Test the weight using different input set:

$$y_{inj} = \sum_{i=1}^{4} x_i * w_{ij}$$

S1=(1 1 0 0) t1=(1 0)

$$y_{in1} = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41}$$

$$y_{in1} = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42}$$

$$= [1 \quad 1 \quad 0 \quad 0] \begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix} = [6 \quad -6] = [1 \quad 0]$$

S2=(0 1 0 0) t2=(1 0)

$$= [0 \quad 1 \quad 0 \quad 0] \begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix} = [4 \quad -4] = [1 \quad 0]$$

S3=(0 0 1 1) t3=(0 1)

$$=[0 \quad 0 \quad 1 \quad 1]\begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix}=[-6 \quad 6]=[0 \quad 1]$$

S4=(0 0 1 0) t4=(0 1)

$$=[0 \quad 0 \quad 1 \quad 0]\begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix}=[-4 \quad 4]=[0 \quad 1]$$

S3=(1 0 0 0)   t3=?

$$=\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}2 & -2\\ 4 & -4\\ -4 & 4\\ -2 & 2\end{bmatrix}=\begin{bmatrix}2 & -2\end{bmatrix}=\begin{bmatrix}1 & 0\end{bmatrix}$$

S4=(1 1 1 0)   t4=?

$$=\begin{bmatrix}1 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}2 & -2\\ 4 & -4\\ -4 & 4\\ -2 & 2\end{bmatrix}=\begin{bmatrix}2 & -2\end{bmatrix}=\begin{bmatrix}1 & 0\end{bmatrix}$$

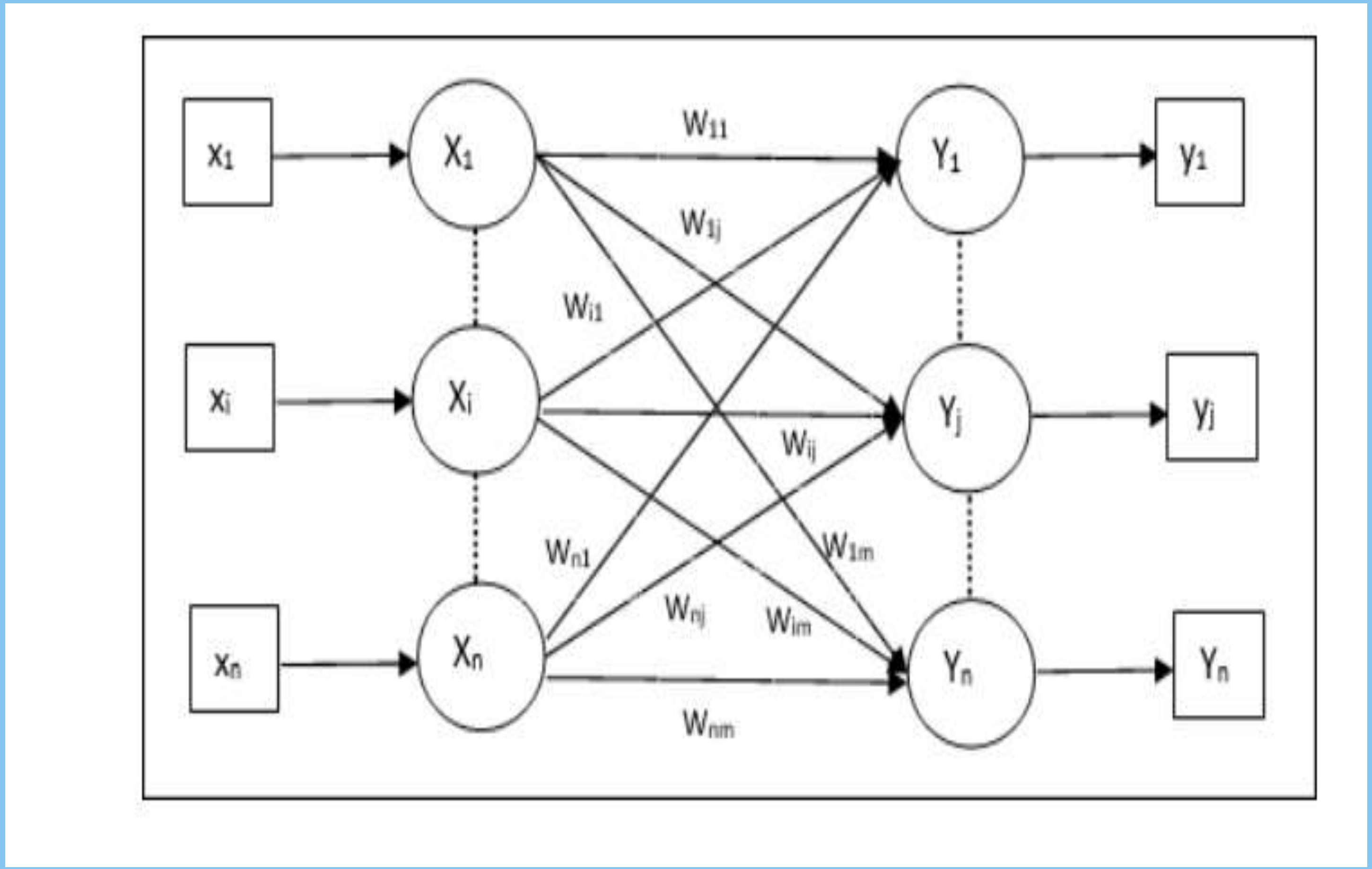
S5=(1 1 1 1)   t5=?

$$=\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}2 & -2\\ 4 & -4\\ -4 & 4\\ -2 & 2\end{bmatrix}=[0 \quad 0]=[0 \quad 0]$$

S6=(0 1 -1 0)   t6=?

$$=\begin{bmatrix}0 & 1 & -1 & 0\end{bmatrix}\begin{bmatrix}2 & -2\\ 4 & -4\\ -4 & 4\\ -2 & 2\end{bmatrix}=[8 \quad -8]=[1 \quad 0]$$

Auto Associative Memory Neural Networks



Mutually Orthogonal Pairs

Two vectors x and y are orthogonal if

$$\sum_i x_i * y_i = 0$$

$$= \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} * \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = [0]$$

Example 3: A auto associative neural network is trained by Hebb outer product rule for input row vector S=(1 1 -1 -1). Find the weight matrix.
S1=(1 1 -1 -1) t1=(1 1 -1 -1)

Step 1: Initialize all weights to zero
Step 2: Find the output for each input
S1=(1 1 -1 -1) t1=(1 1 -1 -1)

$$w_1 = S_1^t * t_1$$

$$= \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}$$

Test the weight using different input set:

$$y_{inj} = \sum_{i=1}^{4} x_i * w_{ij}$$

$$y_{in1} = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41}$$

$$y_{in2} = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42}$$

$$y_{in3} = x_1 w_{13} + x_2 w_{23} + x_3 w_{33} + x_4 w_{43}$$

$$y_{in4} = x_1 w_{14} + x_2 w_{24} + x_3 w_{34} + x_4 w_{44}$$

S1=(1 1 0 0) t1=(1 0)

$$= \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}$$

S2=(0 1 0 0) t2=(1 0)

$$= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & -2 \\ 4 & -4 \\ -4 & 4 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 4 & -4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$