2/6/17

MODULE - 1

Hard problems and soft problems

eg: I am 55 ft tall eg: I like icecream very much.

→ Hard Computing and Soft computing

① Soft computing is just automating process of computing - "Hard computing means just doing computing according to your needs.

② SC is when program leads its own HC is when you tell the prgm to compute particular thing.

③ SC not directly connected to hardware.

④ SC is not a homogeneous body of concepts

⑤ SC is combination of different logics, fuzzy logic, Neural network, probabilistic reasoning genetic algorithm.

⑥ Hard computing requires prgm to be written, SC involves its own program.

⑦ HC is strictly sequential, SC allows parallel computation

⑧ HC produces precise answer, SC can yield

approximate answers.

Soft computing main constituents.

- Fuzzy systems - imprecision
- Neural Network - learning
- Evolutionary computing - uncertainty
- Probabilistic reasoning - optimization

Advantages:

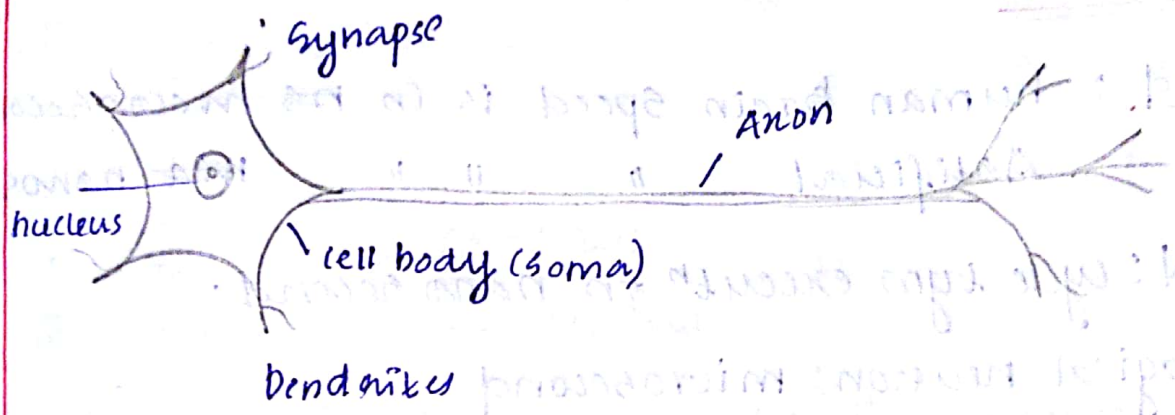
* Models based on human reasoning.
* closer to human thinking.
* Models can be,
  - Linguistic
  - Simple
  - Faster when computing
  - Effective in practice.

The main goal of SC is to develop intelligent machines to provide solutions to real world problems which are not model or too difficult to model mathematically. Its aim is to exploit the tolerance for approximation uncertainity imprecision and partial truth. Inorder to achieve close resemblence with human like decision making.

9/11/17

1. Artificial Neural Network.

AN ANN is may be defined as an information processing model that is inspired by the way biological nervous systems, such as brain, process information.



Impulse (electric potential) carried by AXON.

Nearer connected to cell body is Dendrites.

~~Schematic~~

Artificial

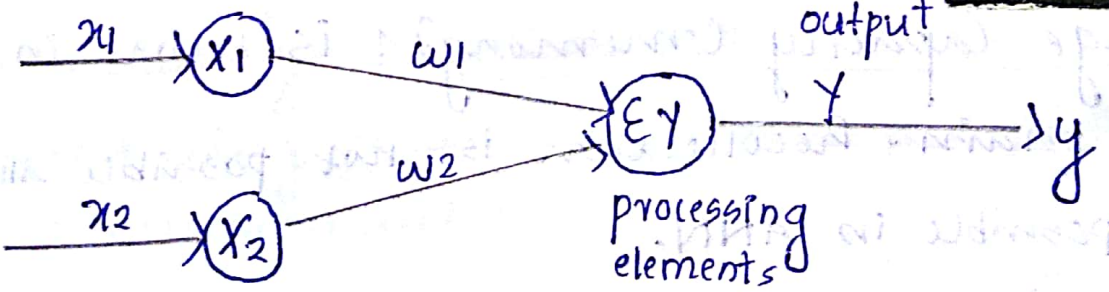


fig: Architecture of simple ANN

Biological Neuron	Artificial Neuron.
Cell	Neuron.
Dentrits	Weight
Soma	Net input.
Axon	Output

DIFFERENCES

1. Speed : human brain speed is in ~~ms~~ microsecon
   Artificial " " " ~~ms~~ nanosecond.

   ANN : cycle tym execution in nanosecond.
   Biological neuron : microsecond

2. Processing.
   Multitasking is possible

3. Size and complexity : is more in human brain compared to ANN

4. Storage capacity (memory) : is more in Biological brain. Recollection is not possible always. But possible in ANN.

5. Tolerance:
   Small damages to neurons is not effected
   ∴ tolerance is greater.

6. Control mechanism:
   CPU in ANN.

Advantages of ANN

1. Adaptive learning
2. Self organizing
3. Real time ~~org~~ operations
4. Fault tolerance.

Application

1. Air traffic control
2. Criminal sentencing
3. Data mining / cleaning / validation
4. Mail advertising
5. Medical diagonising

~~Differen~~ Basic models of ANN

ANN Specified by three basic entities

1. model synaptic interconnection
2. Training or learning rules for updating the connection weights.
3. Activation function.

10/11/17

Basic methods of ANN:

ANN specified by 3 basic entities

1. connection

In ANN there Processing unit but in brain there is no " "
How to connect neurons.

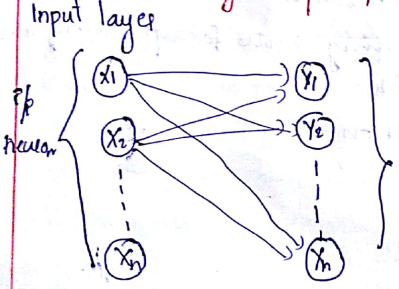
2. learning:
At the begining no data

3. Activation function:

Connection

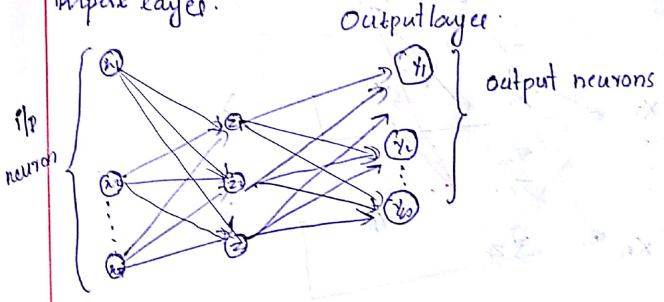
1. Single layer feed forward n/w
2. Multilayer feed forward n/w.
3. Single node with its own feedback.
4. Single layer recurrent n/w
5. Multilayer recurrent n/w.

Single layer feed forward n/w.

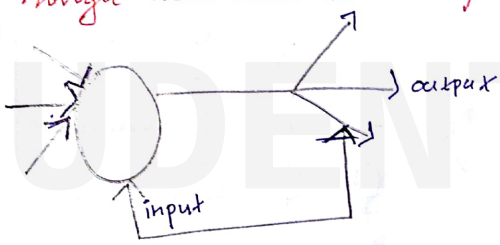


Multilayer feed forward n/w

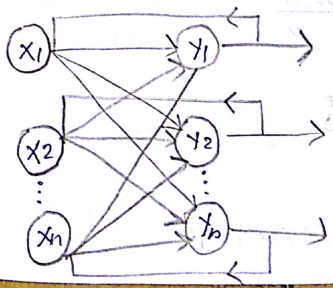
There is hidden layer b/w input layer and o/p layer



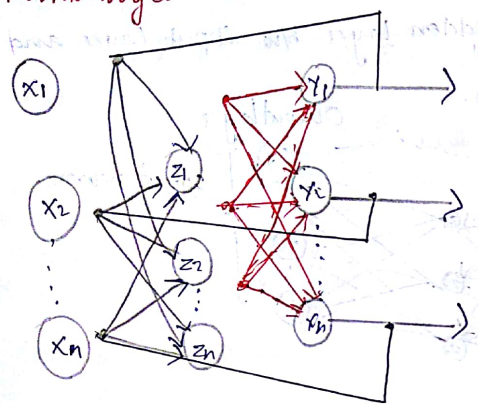
Single node with its own feedback:



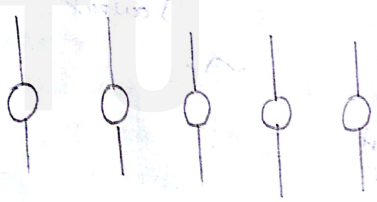
Single layer recurrent



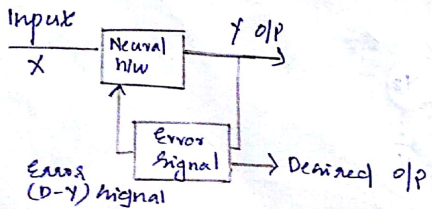
Multilayer recurrent.



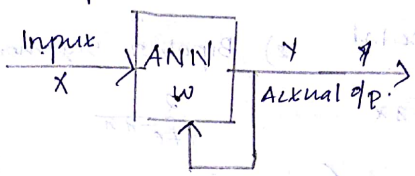
On center



Supervised learning



Unsupervised



Learning

1. parameter learning
2. Structure learning

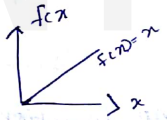
Learning types

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning.

11/1/22

Activation function

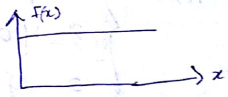
1. Identity function.
   $f(x) = x$ for all x



2. Binary step function

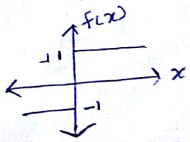
θ = threshold value.

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



2. Bipolar step function.

$$f(x) = \begin{cases} 1 & x \geq \theta \\ -1 & x < \theta \end{cases}$$



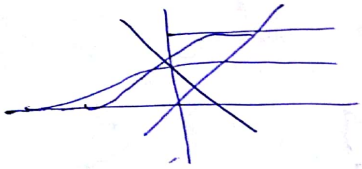
4. Sigmoidal function

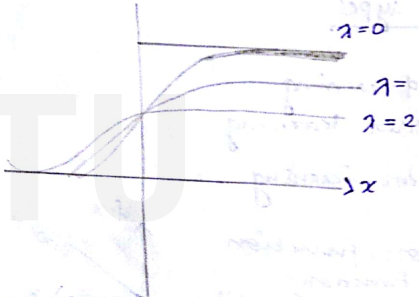
1. Binary sigmoidal

$$f(x) = \frac{1}{1-e^{-\lambda x}}$$

2) Bipolar Sigmoidal

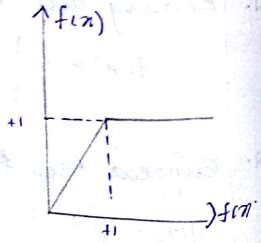
$$\frac{2}{1+e^{-\lambda x}}$$



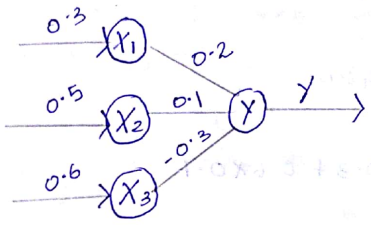


5. RAMP function.

$$f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \le x \le 1 \\ 0 & \text{if } x < 0 \end{cases}$$

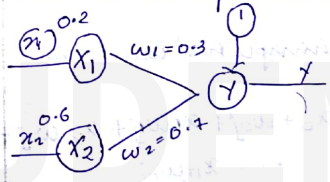


Q Calculate the net i/p to the o/p neuron.



$$Y_{in} = 0.3 \times 0.2 + 0.5 \times 0.1 + 0.6 \times -.3$$
$$= 0.01$$

Q Calculat net input with bias .45



Study it

Terminocogies in ANN weights

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_n^T \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1m} \\ w_{21} & w_{22} & \dots & w_{2m} \\ \vdots & & & \\ w_{m1} & \dots & & w_{nm} \end{bmatrix}$$

Study it

Bias

$x_0 = 1$

$X = (1, x, \dots x, \dots x_n)$

$$y_{in} = w_0 + \sum_{i=1}^{n} x_i w_i$$

$$= b_j + \sum_{i=1}^{n} x_i w_{ij}$$

$y = 0.45 + 0.2 + 0.3 + 0.6 \times 0.7$

$= 0.93$

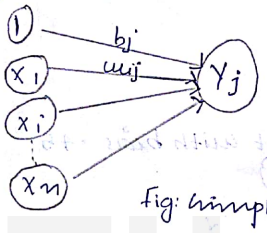


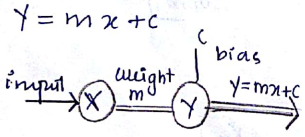
fig: simple net with bias

$$y_{inj} = \sum_{i=0}^{n} x_i w_{ij} + x_0 w_{0j} + x_1 w_{1j} + x_2 w_{2j} + \cdots x_n w_{nj}$$

$$= w_{oj} + \sum_{i=1}^{n} x_i w_{ij}$$

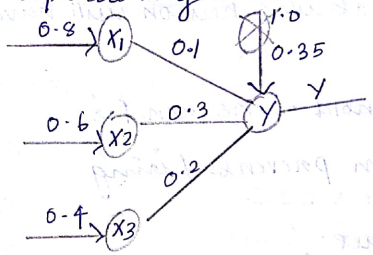
$$\boxed{y_{inj} = b_j + \sum_{i=1}^{n} x_i w_{ij}}$$

$y = mx + c$



Q | Obtain the o/p of the neuron for the nlw- given using activation functions as i) binary sigmoi

2) Bipolar sigmoidal



$y = 0.8 \times 0.1 + 0.6 + 0.3 + 0.4 + 0.2 + 1.0 \times 0.35$

$= 0.53$

1) Binary sigmoidal

$$f(x) = \frac{1}{1+e^{-\lambda x}}$$

$$= \frac{1}{1+e^{-0.53}}$$

$= .625$

2) Bipolar sigmoidal

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1$$

$$= \frac{2}{1+e^{-0.53}} - 1$$

$= .259$

McCulloch-Pitts Neuron (M-P)

Theory:-

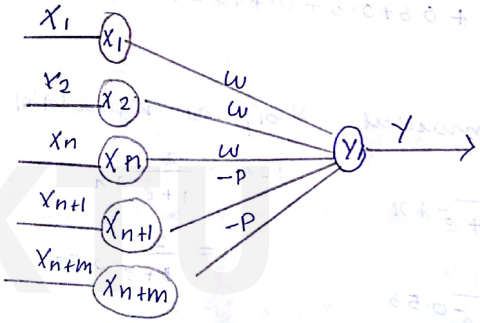
Connected by directed weighted path

Activation of a M-P neuron is binary

At any time step the neuron may fire or not

- weight may be excitatory / inhibitory (-ve)
- All the excitatory neuron connected weight entering into a particular neuron will have same weight
- if net input > threshold → neuron fire
- Inhibitory neuron prevents firing

Architecture



-p- Inhibitory

Excitatory ($w > 0$)

Inhibitory ($p < 0$)

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$ firing neuron

$$\boxed{\theta > nw - p}$$

o/p will fire if it receives "k" or more excitatory inputs $kw \geq \theta > (k-1)w$

Implement AND function using MP (binary data)

Truth table

x_1	x_2	y	
0	0	0	(0,0) $\Rightarrow y_{in} = 0x1 + 0x1 = 0$
0	1	0	(0,1) $\Rightarrow y_{in} = 0x1 + 1x1 = 1$
1	0	0	(1,0) $\Rightarrow y_{in} = 1x1 + 0x1 = 1$
1	1	1	(1,1) $\Rightarrow y_{in} = 1x1 + 1x1 = 2$

$y_{in} = x_1 w_1 + x_2 w_2$

assume $w_1 = 1$, $w_2 = 1$

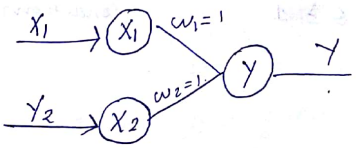
$\theta \geq nw - p$

$\theta \geq 2x1 - 0 = \theta \geq 2$

Output neuron can be written as

$$y = f(y_{in}) = \begin{cases} 1 & y_{in} \geq 2 \\ 0 & y_{in} < 2 \end{cases}$$

Architecture



In the case of AND firing only if greater than 0

Q Implement ANDNOT function using MP neuron.

First i/p true 2nd i/p false

x_1	x_2	y	
0	0	0	(0,0) $\Rightarrow y_{in} = 0\times1 + 0\times1 = 0$
0	1	0	(0,1) $\Rightarrow y_{in} = 0\times1 + 1\times1 = 1$
1	0	1	(1,0) $\Rightarrow y_{in} = 1\times1 + 0\times1 = 1$
1	1	0	(1,1) $\Rightarrow y_{in} = 1\times1 + 1\times1 = 2$

$w_1 = 1$
$w_2 = i$

we need to fire only in (1,0) but we get $\theta = 2$ in (1,1)

Case-2 $w_1 = 1$ and $w_2 = -1$

$(0,0) \Rightarrow y_{in} = 0\times1 + 0\times-1 = 0$.

$(0,1) \Rightarrow y_{in} = 0\times1 + 1\times-1 = -1$

$(1,0) \Rightarrow y_{in} = 1\times1 + 0\times-1 = 1$

$(1,1) \Rightarrow y_{in} = 1\times1 + 1\times-1 = 0$

(1,1)
(-1,1)
(1,-1)

Threshold value = 1 $\quad \theta \geq nw - p$

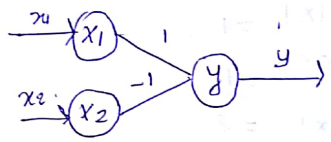
$\theta \geq 2\times1 - 1$

$\theta \geq 1$

p = no. of negative conditions.

$$y_{in} = \begin{cases} 1 & y_{in} \geq 1 \\ 0 & y_{in} < 1 \end{cases}$$

architecture



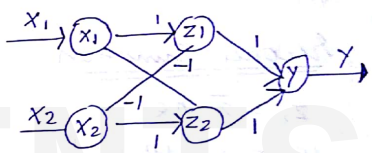
Q Implement XOR function using MP neuron:

Truth table

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$y = x_1\bar{x_2} + \bar{x_1}x_2$

$y = z_1 + z_2$

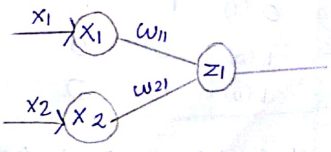


$z_1 = x_1\bar{x_2}$ & $z_2 = \bar{x_1}x_2$

First function

$z_1 = x_1\bar{x_2}$

x_1	x_2	$\bar{x_2}$	$z_1 = x_1\bar{x_2}$
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0



Assume $w_{11} = w_{21} = 1$

(0,0) => Zin = 0x1 + 0x1 = 0

(0,1) => Zin = 0x1 + 1x1 = 1

(1,0) => Zin = 1x1 + 0x1 = 1

(1,1) => Zin = 1x1 + 1x1 = 2

$w_{11} = 1$, $w_{12} = -1$

$\theta \Rightarrow 1$

(0,0) => 0

(0,1) => -1

(1,0) => 1

(1,1) => 0

Second function

$Z_2 = x_1 x_2$

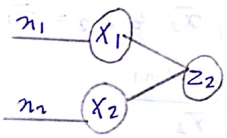
$w_{11} = -1$, $w_{22} = 1$

Second function

$Z_2 = \bar{x}_1 x_2$

$w_{11} = -1$, $w_{22} = 1$

x_1	x_2	Z_2
0 (1)	0	0
0 (1)	1	1
1 (0)	0	0
1 (0)	1	0



case 1 : $w_{11} = -1$ and $w_{22} = 1$.

(0, 0) => Zin = 0x-1 + 0x1 = 0

(0,1) => Zin = 0x-1 + 1x1 = 1

(1,0) => Zin = 1x-1 + 0x1 = -1

(1,1) => Zin = 1x-1 + 1x1 = 0

Third function

$Y = z_1$ OR z_1

x_1	x_2	y	z_1	z_2
0	0	0	0	0
0	1	1	0	1
1	0	1	1	0
1	1	0	0	0

$y_{in} = z_1 v_1 + z_2 v_2$

$v_1 = 1$ and $v_2 = 1$

x_1	x_2	
0	0	=> 0x1 + 0x1 = 0
0	1	=> 0x1 + 1x1 => 1
1	0	=> 1x1 + 0x1 => 1
1	1	=> 0x1 + 0x1 => 0

$w_{11} = w_{22} = 1$ Excitatory

$w_{12} = w_{21} = -1$ Inhibitory

$v_1 = v_2 = 1$ Excitatory

Hebb network

Theory $w_i(new) = w_i(old) + xy$

Training algorithm:

Step 1: Initialize the weight.

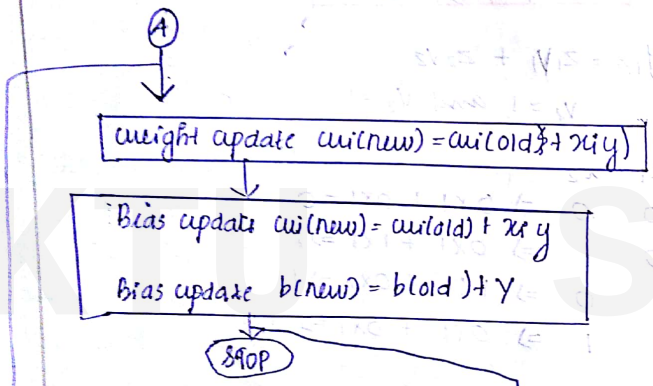
2: For each training input s:t

Set activations

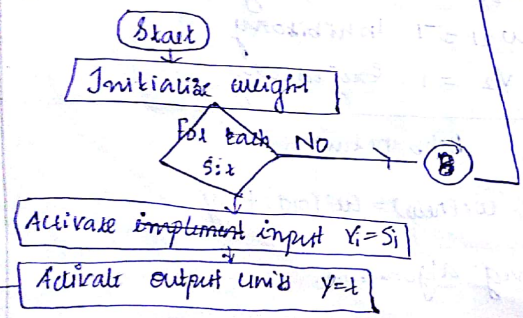
weight adjustment

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

$b(new) = b(old)$



Flow chart



Hebb network

Donald Hebb stated that "In the brain the learning is performed by the change in the synaptic gap when an axon of cells A is near enough to x_i cell B, and repeatedly/permanently takes place in firing it, some growth process / metabolic change take place one or both the cells such that A is efficiency as one of the cells firing B is increased"

* According to hebb rule, the weight vector is found to increase proportionality to the product of the learning signal and input.
* Here the learning signal = neurons output
* In hebb learning interconnected neurons are "on" simultaneously, then the weights associated with this neurons can be increased by the modification ~~in~~ made in their synaptic gap of strength
* The weight update in hebb rule is given by

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

Training algorithm of H

1: Initialize the weight.
basically in this network they may be zero
$w_i = 0$ for $i = 1, \ldots n$ where n may be total no: of input neurons.

2: Step: 3 to 4 have to perform for each i/p training vectors and target output (s:t)

3: Input units activations are set (identity function) $x_i = s_i$ for $i = 1$ to n.

4: Output unit activations are set $(y = t)$

5: weight adjustments & bias adjustments are performed

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

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

weight updation formula.

$w(new) = w(old) + xy$

change in weight

$\Delta w = xy$

$w(new) = w(old) + \Delta w$.

15/11/17 Linear Reparability :-

Reperating +ve response and -ve response using a boundary.

$$y_{in} = b + \sum_{i=1}^{n} x_i w_1$$

Decision making line: draw to reperate +ve & -ve responses.

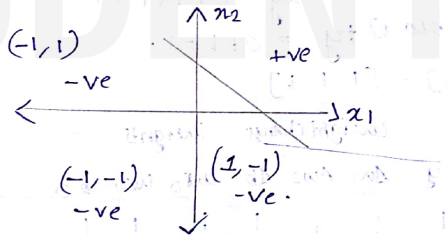
Decision making line | Decision support line | Linear Reparable line.

$y_{in} > 0$ (+ve region); $y_{in} < 0$ (negative region)

ie $b + \sum_{i=1}^{n} x_i w_i = 0$.

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

$x_2 = -x_1 \frac{w_1}{w_2} - \frac{b}{w_2}$ where $w_2 \neq 0$.



Q Design a hebb net to implement logical AND gate using bipolar i/p and target.

x_1	x_2	bias	target y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

$-1=0$.

Initialize $w_1=w_2=b=0$ $[w_1\ w_2\ b]=[0\ 0\ 0]$

$w_i(new) = w_i(old) + x_i y = w_i(old) + \Delta w$

First input $[x_1\ x_2\ b]=[1\ 1\ 1]$, target $=1$ $(y=1)$

$w_1(new) = w_1(old) + x_1 y = 0 + 1\times 1 = 1$

$w_2(new) = w_2(old) + x_2 y = 0 + 1\times 1 = 1$.

$b(new) = b(old) + y = 0 + 1 = 1$.

$[w_1\ w_2\ b] = [1\ 1\ 1]$

Inputs				weight change			weights		
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
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

$\Delta w_i = x_i y$

$\Delta w_1 = x_1 y$

$\Delta w_2 = x_2 y$

$\Delta b = y$

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

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

$\Delta b = y$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} \quad \text{whose} \quad w_2 \neq 0$$

1. Input $[x_1\ x_2\ b]$ $[1\ 1\ 1]$

   $[w_1\ w_2\ b]$ $[1\ 1\ 1]$

$$x_2 = \frac{-1}{1} x_1 - \frac{1}{1}$$

$$x_2 = -x_1 - \frac{1}{1}$$

$$x_2 = -x_1 - 1 \quad \text{—(1)}$$

2. Input $[x_1\ x_2\ b]$ $[1\ 1\ 1]$

   $[w_1\ w_2\ b]$ $[0\ 2\ 0]$

$$x_2 = -\frac{0}{2} x_1 - \frac{0}{2} = 0 \Rightarrow x_2 = 0 \quad \text{—(2)}$$

3. Input $[-1\ 1]$ $w_1\ w_2\ b$ $[1\ 1\ -1]$

$$x_2 = \frac{-1}{1} x_1 + \frac{1}{1}$$

$$x_2 = -x_1 + 1 \quad \text{—(3)}$$

4. Input [-1,-1,1] = w_1 w_2 b [2 2 -2]

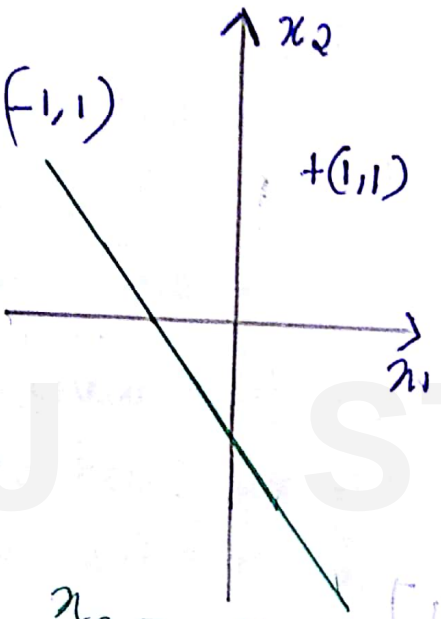
$$x_2 = \frac{-2}{2} x_1 + \frac{2}{2}$$

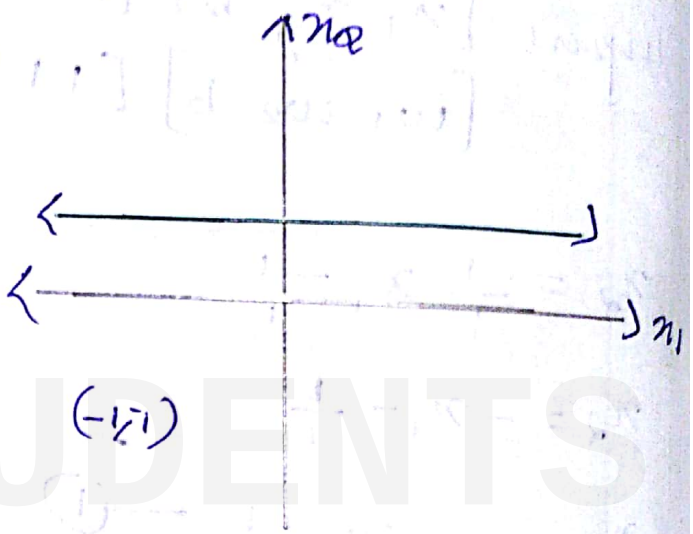
$$x_2 = -x_1 + 1 \longrightarrow (4)$$

Initialize ~~$w_1 = w_2$~~, $b = 0$

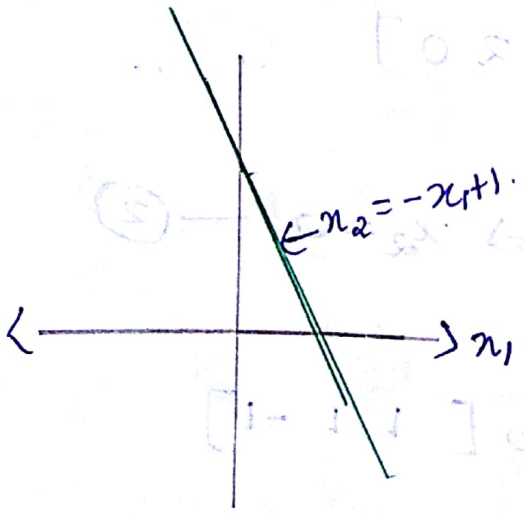
$x_2 = -x_1 - 1$

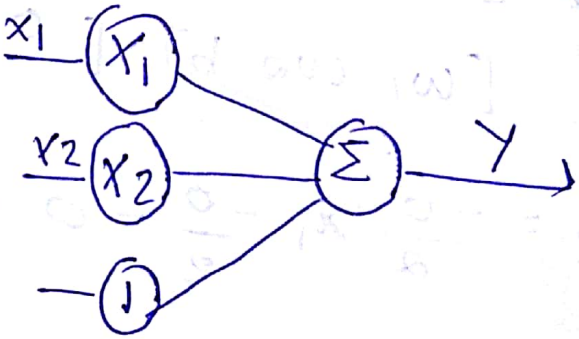
$x_2 = 0$





$x_2 = -x_1 + 1$





This graph is selected.