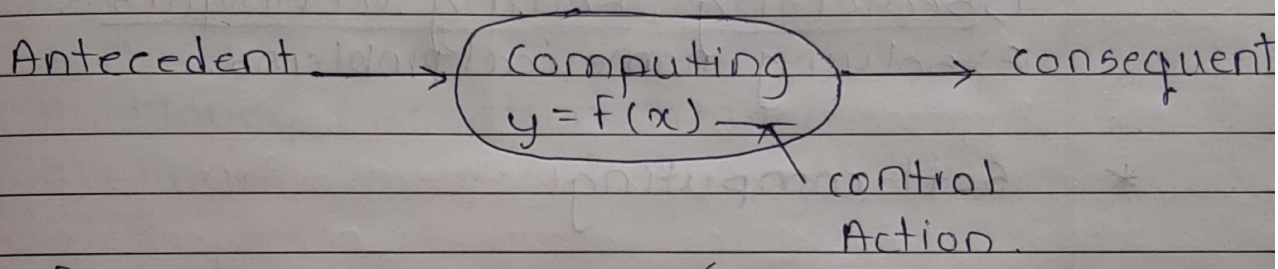
Chap 1. Fundamentals of soft computing

* Soft computing

- concept of computation
- It is the use of Approximate calculations to provide Approximate but usable solutions to complex computational problems



f = mapping function (maps input to O/P)
or also called Algorithm.

* characteristics of computing

1) Should provide precise solution.
2) control action should unambiguous & accurate
3) Suitable for model problem, which is easy to model mathematically.
4) Should be Algorithm/ function available.

* Hard computing

- Introduced by Lotfi Zadeh.
- computing is hard if:-
  a) Precise result is guranteed.
  b) control action is unambiguous.
  c) Control actions are defined by either in the form of some mathe. model or algo.
- Ex:-
  1) Searching, sorting techniques.
  2) Solving numerical problems.

* Soft computing

- Introduced by Lotfi Zadeh.
- Soft computing is defined as:-
  A collection of methodologies that aim to exploit the tolerance for imprecision & uncertainty to achieve tractability, robustness & low solution cost.

Soft computing →
- Fuzzy Logic
- Neural computing
- Genetic Algorithm

* Characteristics of SC.

1) Does not require mathematical model.
2) May not yeild the precise solution.
3) Algorithms are adaptive means it can adjust to the change of any dynamical situation.
4) Use some biological inspired methodology

Ex:-
a) Handwritten Character Recognition (ANN)
b) Money Allocation Problem
c) Robot Movement (Fuzzy logic)

* Difference between HC & SC

	HC	SC.
1)	Requires precisely stated analytical model & it is computationally expensive	It is tolerance to imprecision, partial truth & approximation.
2)	Based on Binary logic, crisp system, numerical analysis.	It is based on fuzzy logic, the neural N/W, evolutionary computation
3)	Characteristics of precision & categoricity.	Chara. of approximation & dispositionality
4)	It is deterministic	It is probabilistic.
5)	Requires exact input data	Can deal with ambiguous & noisy data.
6)	Strictly Sequential	can be carried out by parallel computing

* (check Human Nervous system) slide = 18-21

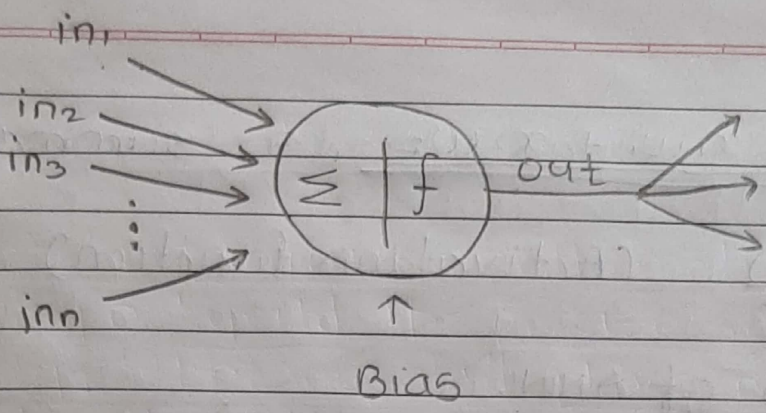
* Neural Network.

- NN are computational models that mimic the complex function of the human brain.
- It consist of interconnected nodes or neurons that process & learn from data, enabling tasks such as pattern recognition & decision making in machine learning.

- Applications:-
  a) As powerful problem solver :-
     character recognition, object detection

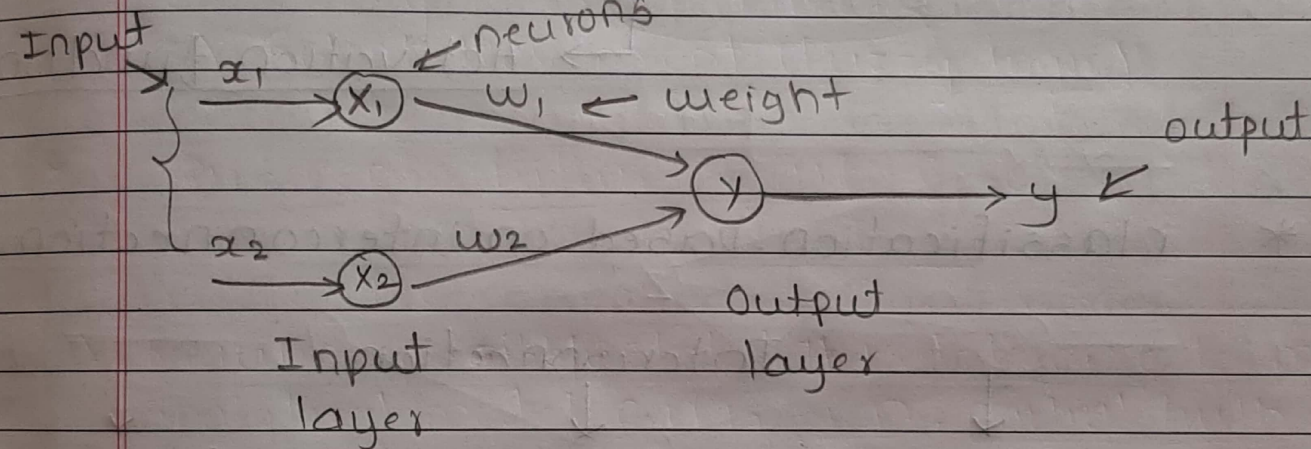
* Artificial NN

1) ANN is a machine learning approach inspired by the way in which the brain performs a particular learning task
2) ANN posses a large no. of processing elements called nodes/neurons which operate in parallel.
3) Neurons are connected with each others by connection link.
4) Each link is associated with weights which contain information about input signal.
5) Each neuron has its own function of the inputs that neuron receives (Activation function) or has its own state.



~~6) Ex: human being~~

6) Application:-
spam classification, Face Recognition, pattern recognition. through a learning process.
(ANN - only ~~one~~ one problem at a time)



(Basic Neuron Network)

- Here links have weights which are random weights initially.
- Each neuron perform some summation & apply Activation function on summation.
(summation = (product of input & weights) sum
i.e sum of product of I/P & weights

* Slide (28-29) charac & Appln of NN.

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

$y = f(y_{in})$ (Activation function)

* Basics of ANN.

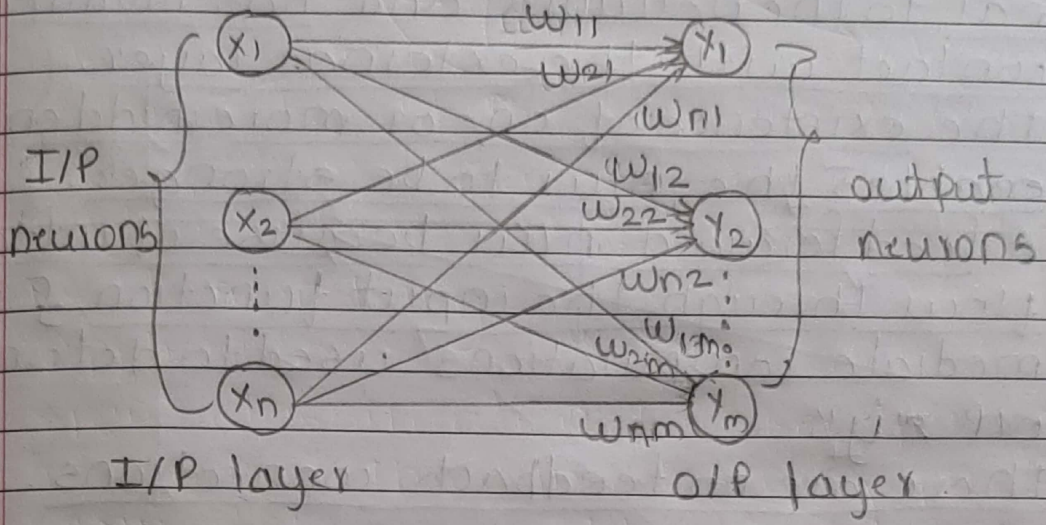
Basic Models of ANN
- Synaptic Interconnection
- Training or learning Rules for updation or adjusting weights
- Activation functions

* classification based on interconnection.

Interconnection.
- Feed Forward
  - single layer
  - Multi layer
- Feed back
- Recurrent
  - Single layer
  - Multi layer

A) Connections

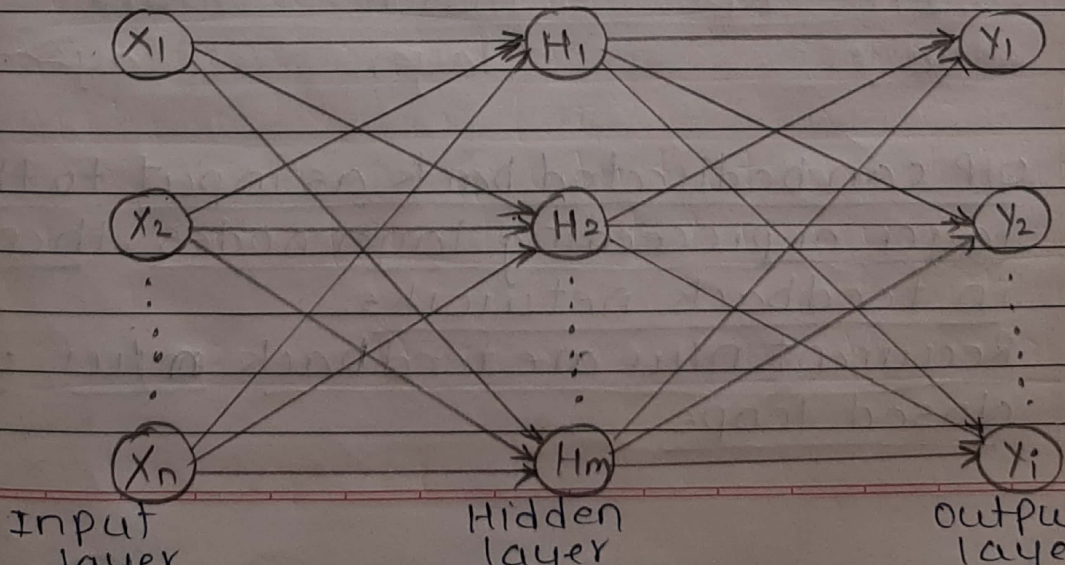
1) Single Layer Feed Forward.



- Only two layers, input layer and output layer but the input layer does not count. because computation is performed in this layer.
- The output layer is formed when different weights are applied to input nodes.
- Neurons collectively gives the output layer to compute the O/P signals.

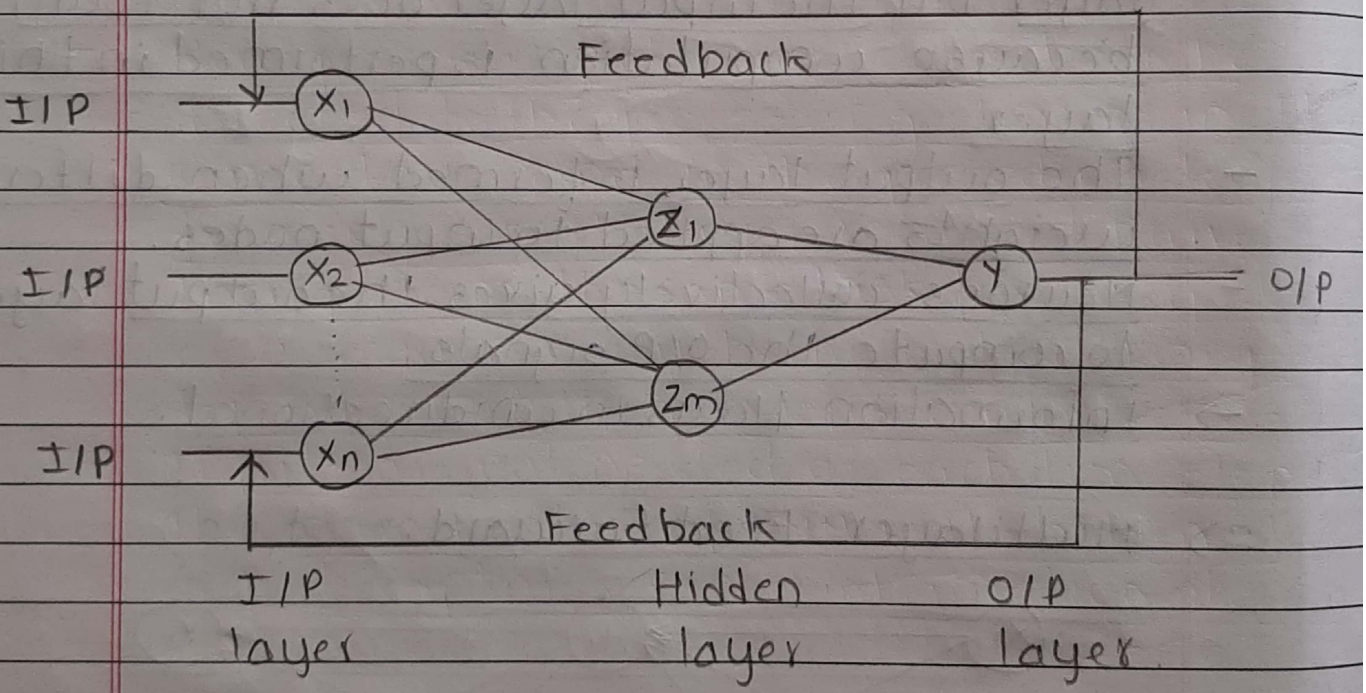
→ Information flows in unidirectional.

2) Multilayer Feed Forward.



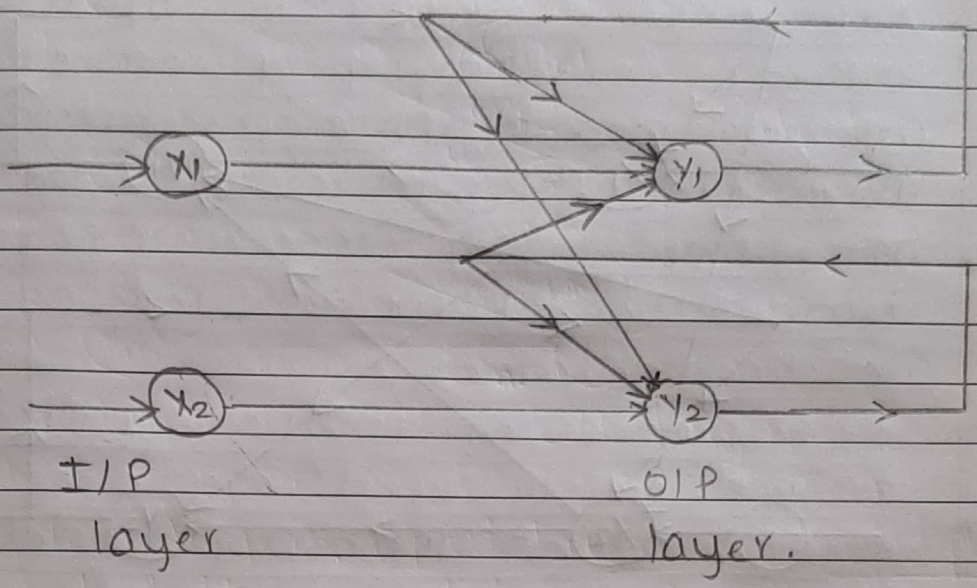
- This layer has a hidden layer that is internal to the network and has no direct contact with the external layer.
- The existence of one or more hidden layer enables the NW to be stronger.
- A feed forward n/w because of information flow through the input function, & intermediate computations used to determine O/P "y"
- There are no feedback connections.

3) Feedback Network



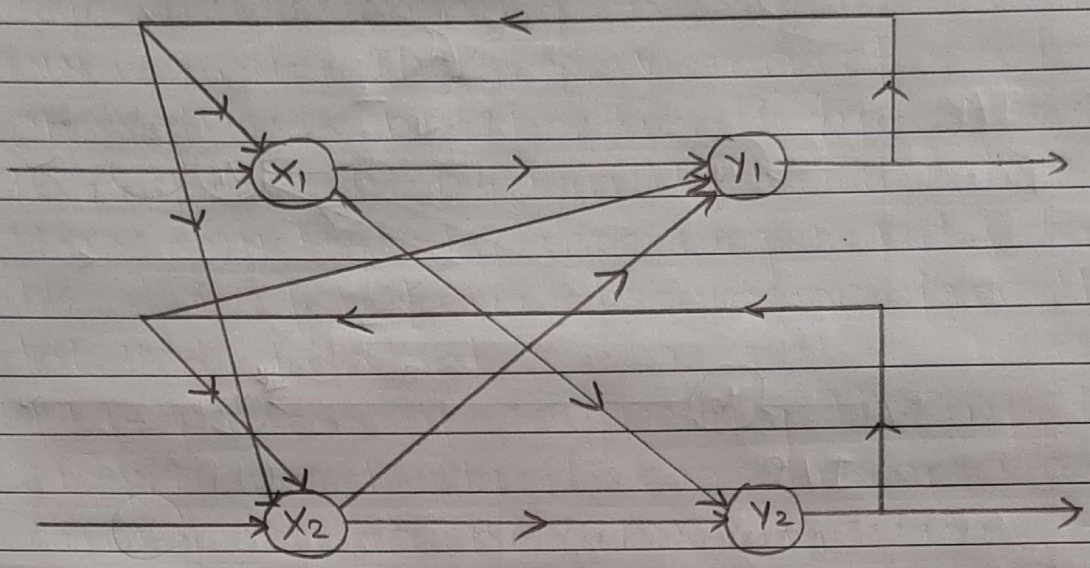
- O/P can be directed back as input to the same layer or preceeding layer nodes, then it results in feedback networks.
- Recurrent n/w are feedback n/w with closed loops.

4) Recurrent Network (single layer)



- Feedback netlw with closed loop are called Recurrent N/W
- Here the processing element's o/p can be directed back to itself or to another processing element or both.
- A recurrent neural n/w is a class of ANN where connection betⁿ nodes form a directed graph along a sequence.
- Allows it to exhibit dynamic temporal behaviour for a time sequence.

5) Multilayer Recurrent Network.



- Here processing element output can be directed to the processing element in the same layer and in the preceding layer forming a multilayer recurrent nlw.
- They perform the same task for every elem of a sequence, with the output being dependent on the previous computations.
- Inputs are not needed at each time step
- The main feature of RNN is its hidden state, which captures some information about a sequence.

*3) Learning Rules.

- The main property of an ANN is its capability to learn
- Learning or training is a process by means of which a neural network adapts itself to a stimulus by making proper parameter adjustments, resulting in desired response.

* Types:-

1) Parameter Learning:-
- It updates the connecting weights in a neural net.
Ex Gradient decent, Back Propogation algorithm.

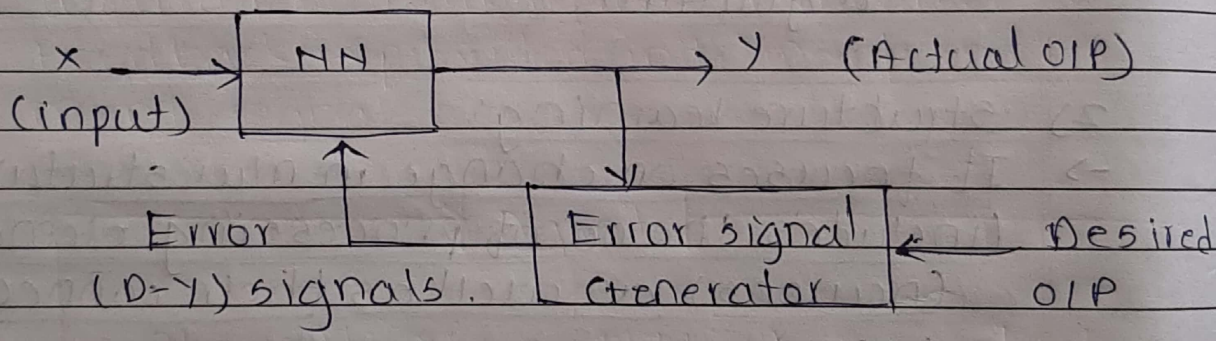
2) Structure learning:-
-> It focuses on change in n/w structure (includes the no. of processing elements (neurons) as well as their connection types)

* Training (Parameter Learning)

The process of modifying the weights in the connections bet^n n/w layers with objective of achieving the expected output is called training a network.

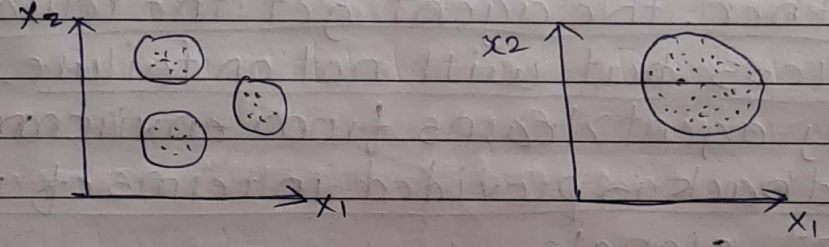
* Types (Supervised, Unsupervised, Reinforcemen

1) Supervised learning
- ANN is under the supervision of an educator who utilizes his/her knowledge of system to prepare the n/w with labeled data sets.
- Thus ANN learn by receiving input & target the set of a few observations from the labelled data set.
- It is process of comparing the input & output with objective and computing the error between the o/p & objective.
- It is used to solve classification and regression problems
- Training pair = [input vector, target vector]

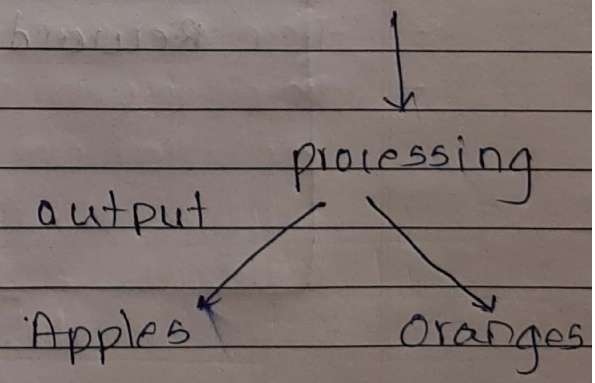


2) Unsupervised learning
- This algorithm is completely opposite to supervised learning. In short there is no complete and clean labelled dataset in unsupervised learning.
- Unsupervised learning is self-organized learning. N/w must discover patterns, regularities, features for input data over the output.

- Its main aim to explore the underlying patterns and predicts the output.
- We basically provide the machine with the data and ask to look for hidden features and cluster the data in a way that make sense.
- Example:-
a) k means clustering
b) Neural Network
- c) All similar input patterns are grouped together as clusters (clustering). If a matching input pattern is not found a new cluster is formed.

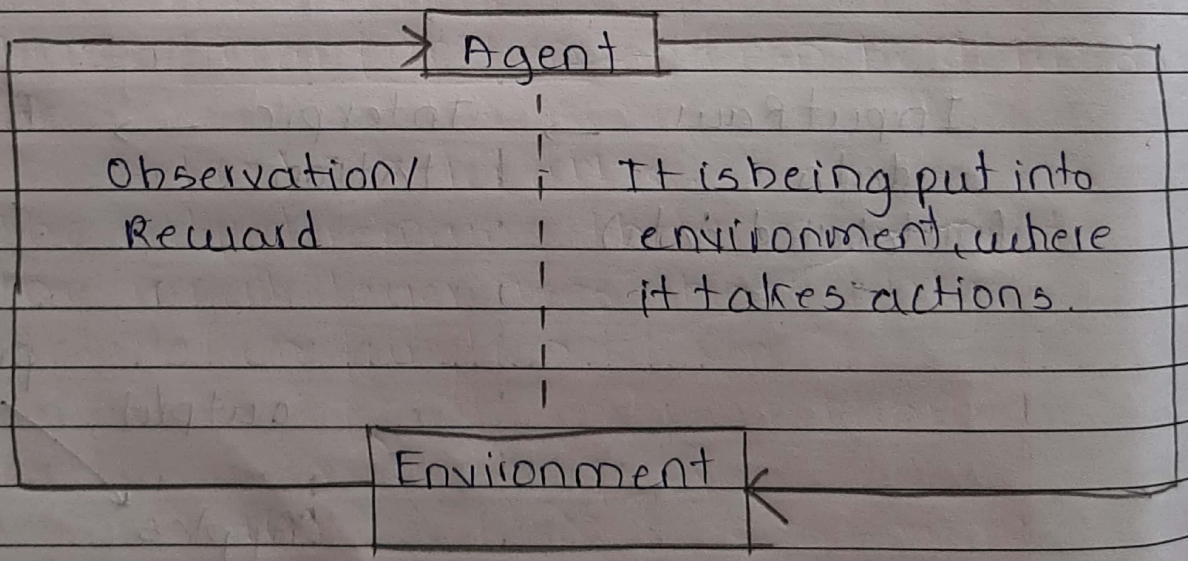


Input Raw data (Unlabelled data) → Interpretation → Algorithm



3) Reinforcement Learning.

- It is neither based on supervised or unsupervised learning.
- Moreover, here the algorithms learn to react to an environment on their own.
- It is rapidly growing and moreover producing a variety of learning algorithm
- These algorithms are useful in the field of Robotics, Gaming, etc.
- In RL learning problem an agent tries to manipulate the environment.
- The agent travels from one state to another. The agent get the reward on sucess but will not on failure. In this way agent learns from environment.
- Feedback is provided in terms of reinforcement signals.



* Comparison table

Criteria	Supervised	Unsupervised	Reinforcement
1) Definition	Learns by using labelled data.	Trained using unlabeled data w/o any guidance.	works on interacting with the environment
2) Type of data	Labelled data	Unlabelled data	No predefined data
3) Type of problem	Regression and classification.	Association & clustering	Exploitation or Exploration.
4) Algorithm	1) Linear Regression 2) Non Linear " 3) SVM, KNN	1) K-Means 2) C-Means 3) Neural N/W	1) Q-Learning SARSA.
5) Aim	Calculate outcomes	Discover patterns.	Learn a series of action.
6) Application	Risk evaluation, forecast sales.	Recommendation system, Anamoly Detection.	Self driving cars, Gaming, Healthcare.
7) Supervision	Extra supervision	No supervision	No supervision.

Date
Page

* Activation function

- The Activation function determines whether or not to stimulate a neuron by generating a weighted sum and then adding bias to it.
- In order to add non-linearity to a neuron's output, the activation function was created.
- It is used to determine the NN's O/P such as yes or no.
- The obtained values are mapped between ~~0, 1, -1 &~~ 0 and 1 or -1 and 1, etc.

(In short:- A.F. decides neuron should be activated or not)

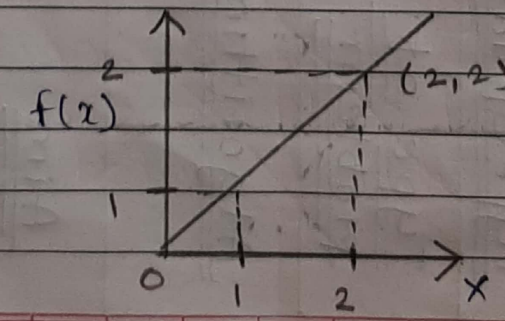
if value > 0 (threshold) = Active
value < 0 (threshold) = Inactive

1) Identity function:-

- It is a linear function and can be defined as:-

$$f(x) = x \text{ for all } x.$$

- The output remains the same as input.
- The input layer uses the identity activation function.

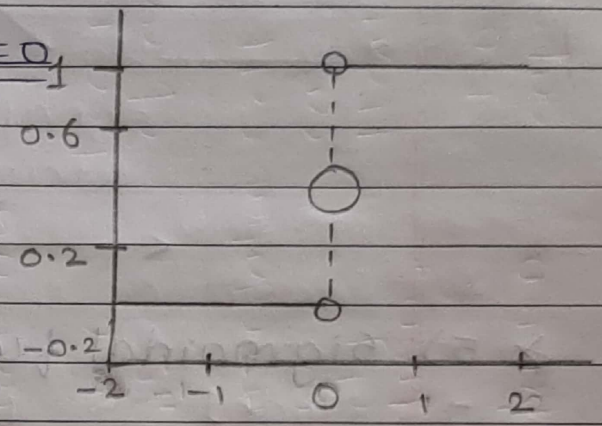


2) Binary step function.

- Binary step function depends on a threshold value that decides whether a neuron should be activated or not.
- If neuron is deactivated means its output is not passed onto the next hidden layer.

Binary step:- here $\theta = 0$,

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

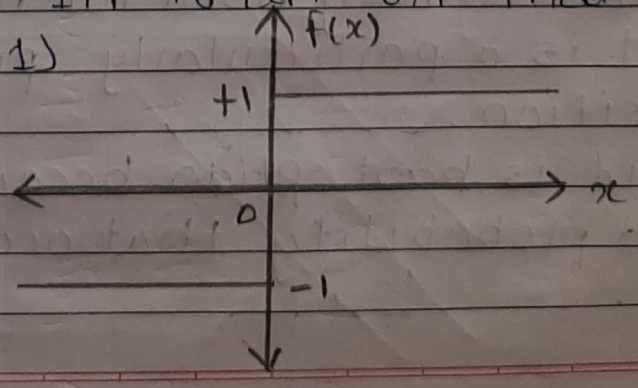


- But it cannot provide multivalue outputs

3) Bipolar step function:-

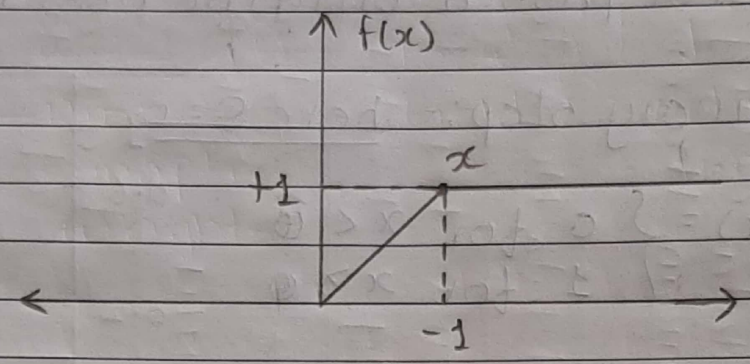
$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ -1 & \text{if } x < \theta \end{cases}$$

- θ (threshold value)
- This function is also used in single layer nets to convert
- the net, I/P to an O/P that is bipolar (-1 or 1)



4) The Ramp function:-

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



★ 5) Sigmoidal function:-

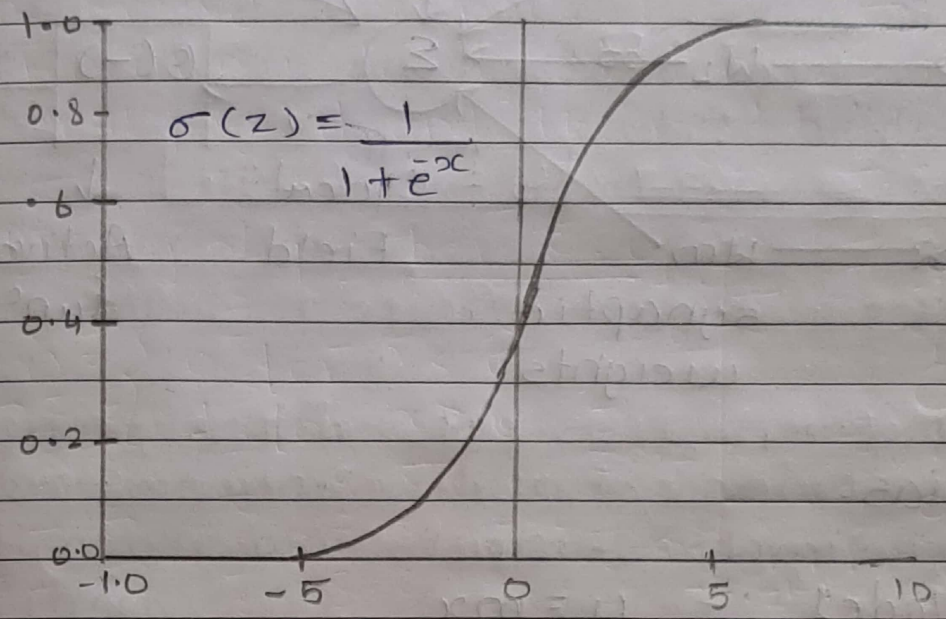
- Widely used in back-propogation nets because of therelationtionship between the value of the functions at a point & the value of the derivative at that point which redues the computational buiden during training.

⟶

- The sigmoid function curve has a S-shaped appearance. We employ the sigmoid function primarily because it occurs between (0 & 1)
- It is particularly used for models whose output is a particularly probabilistic predicition.
- Sigmoid is best option because anything has a probability that occurs bet^n 0 to 1

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

x = summation term



* Types
  1) Binary sigmoid.
  2) Bipolar sigmoid.

* The neuron:-
  - It is basic information processing unit of a NN
  - Acti fun^n (θ) for limiting the amplitude of o/p (of the neuron).

$$y = \theta(v+b)$$

- Adder fun^n :- computes weighted sum of I/P

$$v = \sum_{j=1}^{m} w_j x_j$$

By changing algo to $y = mx + c$.. model can find a line which fits the given data

Here constant "c" is a bias.

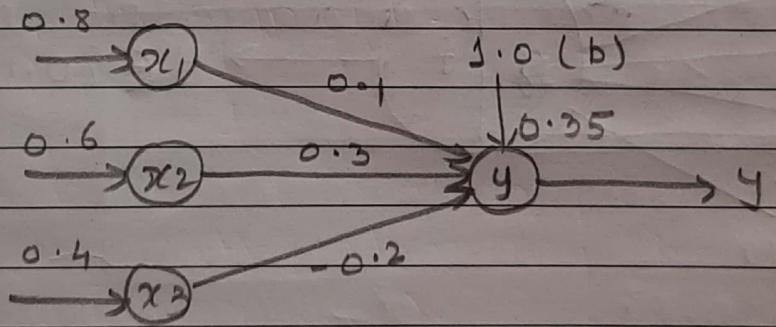
- Bias is a constant which helps the model in a way that it can fit best for the given data.
- In short, it gives freedom to perform best.

* Types
1) Positive bias :- increase the net I/P
2) Negative bias :- decrease the net I/P

(check slide :- 63-65)

* Imp (Diff :- Brain & ANN)

* Example of sigmoid function.



$$\sigma = y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} \quad \text{or} \quad y = f(x) = \frac{1}{1+e^{-x}}$$

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

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$= 0.08 + 0.18 - 0.08 + 0.35$$

$$y_{in} = 0.53$$

$y_1 = \frac{1}{1+e^{-x}}$ $\quad x = 0.53$

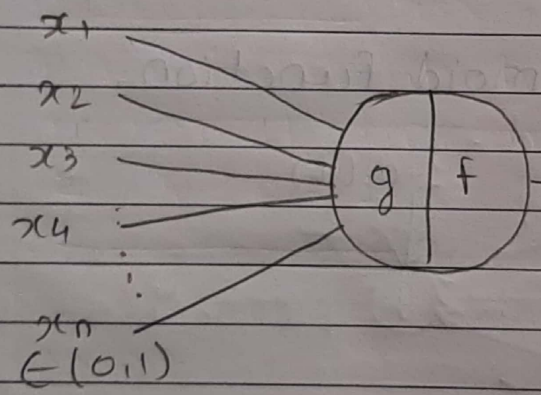
$= \frac{1}{1+e^{-(0.53)}}$

$y = 0.6294$

* McCulloch-Pitts Neuron Model.

- McCulloh-Pitts proposed highly simplified computational Model of neuron (1943)
- g aggreates: inputs

f: takes a decision based on aggregation



- The inputs can be excitatory or inhibitory
- y=0 if any xi is inhibitory.

$g(x_1, x_2, \ldots x_n) = g(x) = \sum_{i=1}^{n} x_i$

$y = f(g(x)) = 1$ if $g(x) \geq 0$ (Fires)

$z = f(g(x)) = 0$ if $g(x) < 0$ (Does not Fire)

1) AND Gate.

0	1	O/P	
0	0	0	0*0 = 0
0	1	0	0*1 = 1
1	0	0	1+0 = 1
1	1	(1)	1+1 = 2 =>

→ Here Θ = 2 (take value 1st i.e (minimum) from O/P here it is 1, then the sum is 2)
Hence Θ = 2

2) OR Gate.

I/P		O/P	
0	0	0	0+0 = 0
0	1	(1)	0+1 = 1 => mini O/P (1st)
1	0	1	1+0 = 1
1	1	1	1+1 = 2

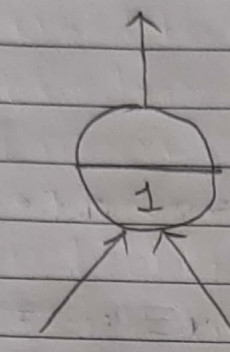
Θ = 1

3) NOR function:- Θ = 0

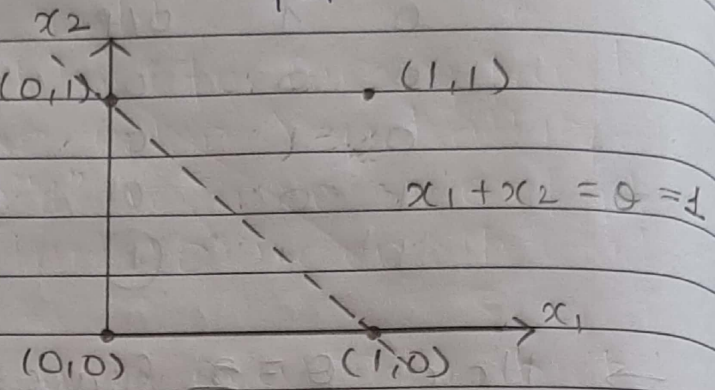
4) NOT function:- Θ = 0

1) OR

⇒ $y \in \{0, 1\}$



$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$

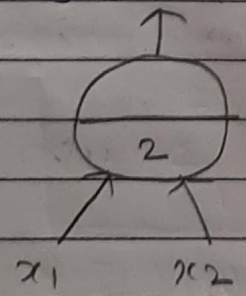


OR function

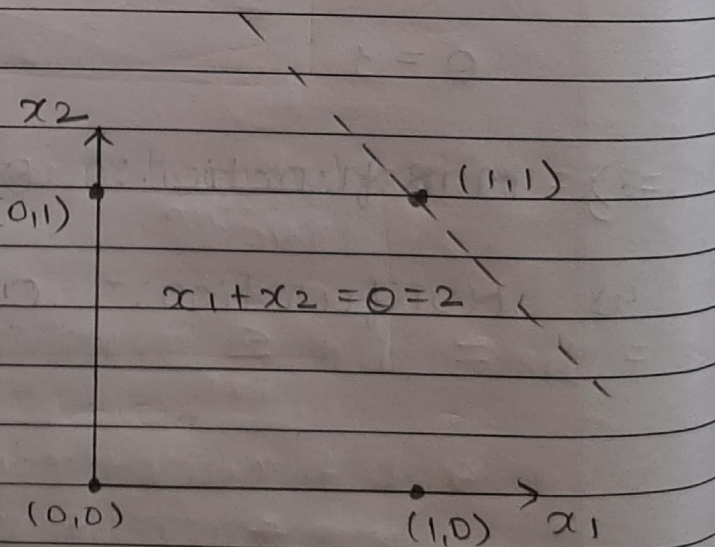
- Single MP neuron splits the I/P points (4 points for 2 binary inputs) into two halves.
- All inputs which produce an output 0 will be on one side (below the line) i.e $\sum_{i=1}^{n} x_i < \theta = 0$
- All inputs which produces an O/P 1 will lie on the line or above the line i.e $\sum_{i=1}^{n} x_i \geq \theta = 1$

2) AND

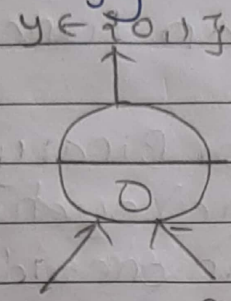
$y \in \{0, 1\}$



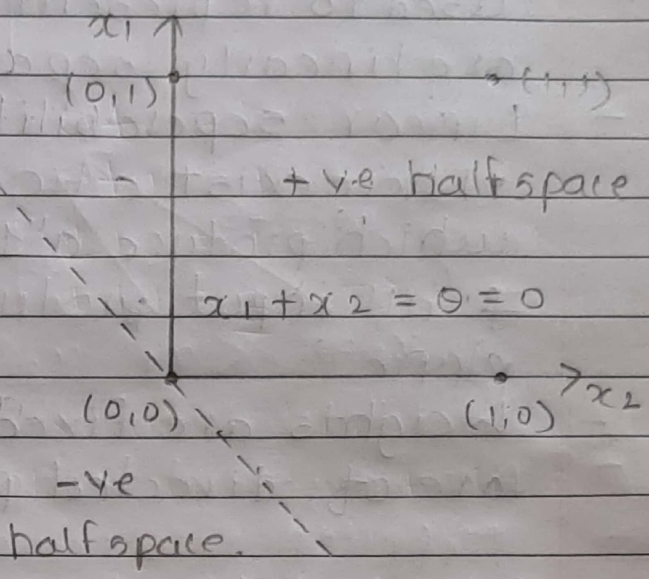
$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$



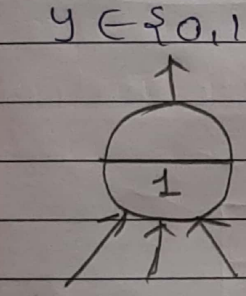
3) Tautology

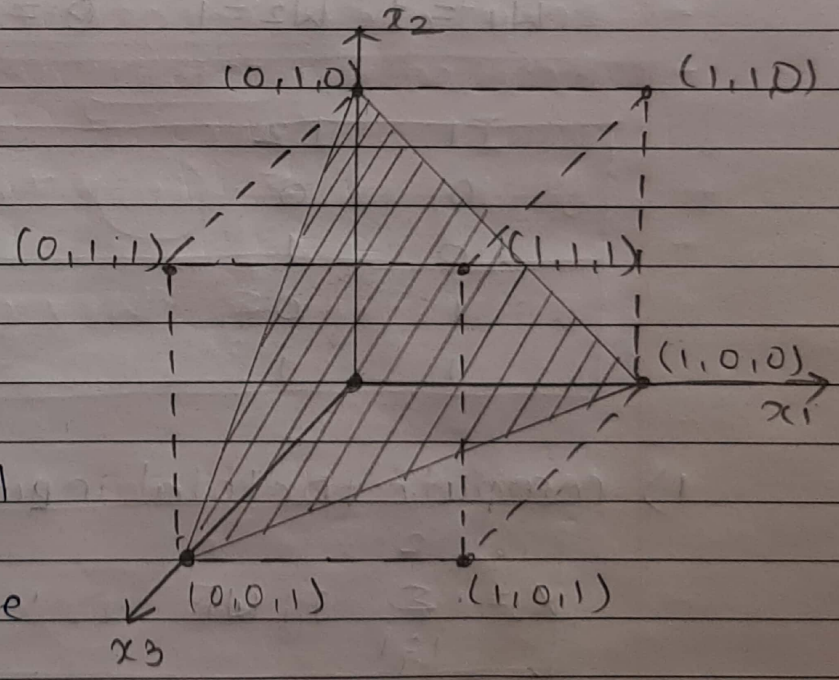


$x_1 + x_2 = \sum_{i=1}^{2} x \geq \theta$



* For 2 points (Neurons)





OR function

only (0,0,0) will not produce 1 hence it will be hidden

$x_1 + x_2 + x_3 = \theta = 1$

- A single McCulloh Pitts Neuron can be used to represent boolean functions which are linearly seperable.
- Linear seperability (for Binary functions)
  - There exist a line(plane) such all inputs which produce a "1" lie on one side of plane & all input produces "0" lie on other side of plane.
- Weights are fixed of neurons which are of directed path.

(check slide : 70 to 74) Bird's example.

* Example

$W_1 = 1 \quad W_2 = 1 \quad \theta = 0$

I_1	I_2
0	0
0	1
1	0
1	1

1) Compute the total input weighted

$$X = \sum_{i=1}^{2} I_i W_i$$

$$X = I_1 W_1 + I_2 W_2$$
$$= I_1 + I_2$$
$$= 1 + 1$$
$$X = 2$$

2) Calculate the O/P using logistic sigmoid activation function.

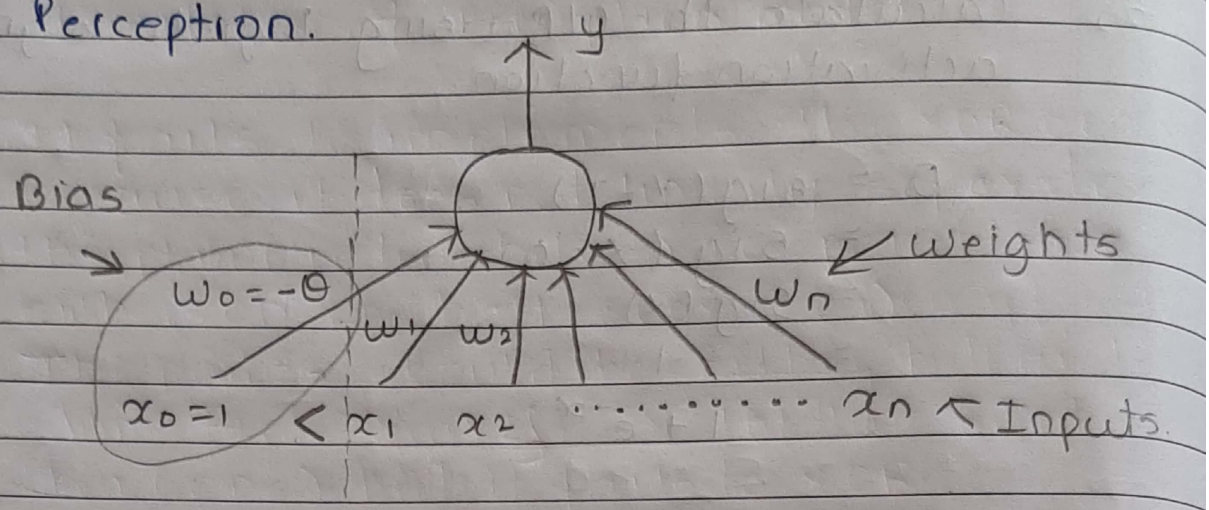
$$O = sig(X - T)$$
$$= sig(X - 0)$$
$$= sig(X)$$
$$= \frac{1}{1 + e^{-x}}$$

let $I_1 = 0$, $I_2 = 0$, $X = 0$

$$O = sig(X - T)$$
$$= \frac{1}{1 + e^{-(0)}}$$
$$= \frac{1}{1 + 1}$$
$$= \frac{1}{2}$$

$$O = 0.5$$

* Perception.



- A more general computational model than McCulloh Pits neuron.
- (Imp) Introduction of numerical weights for inputs and mechanism for learning weigh
- Includes learning algorithm includes theta and weights.
- Input are no longer limited to boolean values.
- Refined by Misky

$$y = 1 \quad \text{if} \quad \sum_{i=1}^{n} (w_i) \times x_i \geq \theta$$

$$y = 0 \quad \text{if} \quad \sum_{i=1}^{n} w_i \times x_i < \theta$$

* More Accepted convention.

$$y = 1 \quad \text{if} \quad \sum_{i=0}^{n} (w_i) \times x_i \geq 0$$

(fires if cond^n true

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

Imp Note : w_0 is also called as Bias as it represents the prior.

* Example.

$(w_0 = \text{bias}) = -\theta$

x_1	x_2	OR	$w_0 + \sum_{i=1}^{2} w_i x_i$
0	0	0	$w_0 < 0$
0	1	1	≥ 0
1	0	1	≥ 0
1	1	1	≥ 0

$w_0 + w_1 \cdot 0 + w_2 \cdot 0 = w_0 < 0$

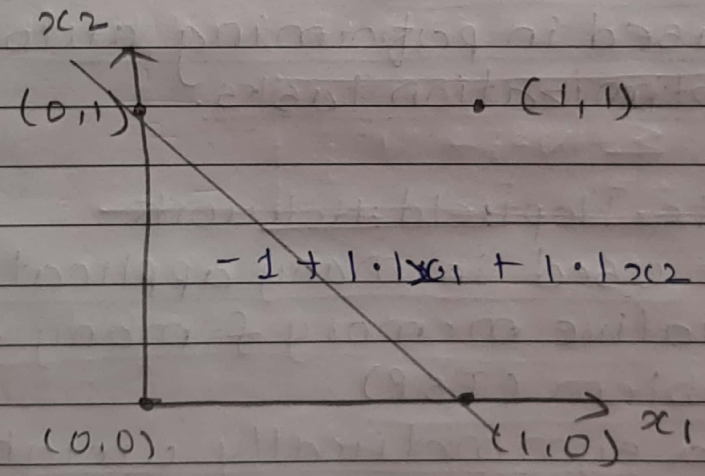
$w_0 + w_1 \cdot 0 + w_2 \cdot 1 = w_2 > -w_0$

$w_0 + w_1 \cdot 1 + w_2 \cdot 0 = w_1 > -w_0$

$w_0 + w_1 \cdot 1 + w_2 \cdot 1 = w_1 + w_2 > -w_0$

→ One of possible solution.

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



Imp

* Limitation of Perception model

- It fails to implements the XOR or XNOR because it does not linearly separate the plain (in two halves) i.e points are either on one half (ex 0 & 1).
- The points should be either below the line (for zero) & on the line or above for (one) but it fails to XOR.

* Hopfield Neural Network.

- Invented by Dr. John J. Hopfield.
- Consist of one layer of 'n' fully connected recurrent neurons
- It is used in performing auto-association and optimization tasks.

* ~~Discrete Hopfield Network~~
- Can be used in many application of associative memory & many optimization of problem (TSP)
- It is a symmetrically weighted N/W i.e the value of (suppose) w_{12} is 2 then value of w_{21} is also 2
- Stores pattern in the form of weight, forms the O/P in which external pattern is provided
- Restores incomplete pattern.

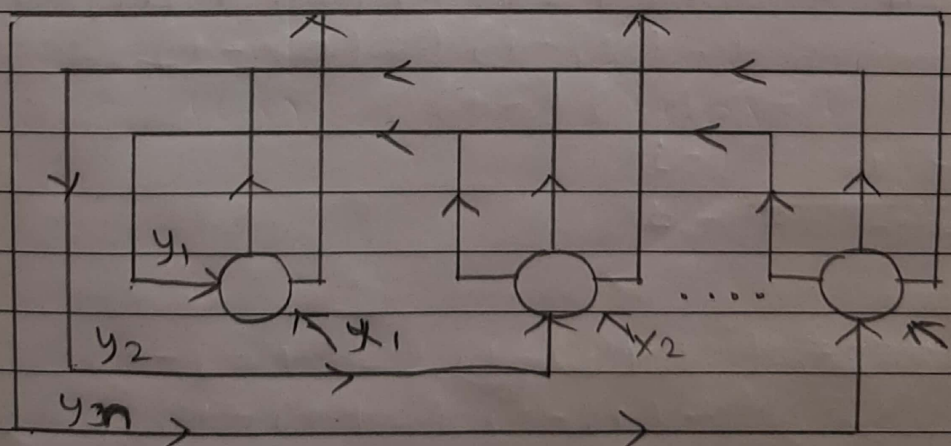
1) Discrete Hopfield N/W

- It is fully interconnected NN where each unit is connected to every other unit.
- It behaves in a discrete manner (fixed values) i.e it gives finite distinct O/P. generally of 2 types:-
  1) Binary (0/1)
  2) Bipolar (-1/1)
- The N/W has symmetrical weights with no-self-connections.

$$W_{ij} = W_{ji} \qquad W_{ii} = 0$$

* Structure & Architecture

- Each neuron has an inverting & non-inverting O/P
- Being fully connected, the O/P of each neuron is an I/P to all other neurons but not the self.



$x_1, x_2, \ldots, x_n$ → I/P to the n given neurons

$y_1, y_2, \ldots, y_n$ → O/P obtained from n given neurons

W_{ij} → weight associated with conn^n bet^n i & j neuron.

* Training Algorithm (Finding the weight matrix)

1) Binary pattern.

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p) - 1][2s_j(p) - 1] \quad (w_{ij} \text{ for all } i \neq j)$$

$$\boxed{w_{ij} = \sum_{s=1}^{n} [2v^s i - 1][2v^s j - 1]} = w_{ji}$$

2) Bipolar pattern.

$$w_{ij} = \sum_{p=1}^{P} [s_i(p) s_j(p)] \quad (\text{where } w_{ij} = 0 \text{ for all } i = j)$$

Imp $\Rightarrow$ $v_{i_{in}}$ $\begin{cases} 1 & \text{if } v_i > \theta \\ v_i & \text{if } v_i = \theta \\ 0 & \text{if } v_i < \theta \end{cases}$

$$\boxed{w_{ij} = \sum_{s=1}^{n} v^s i * v^s j}$$