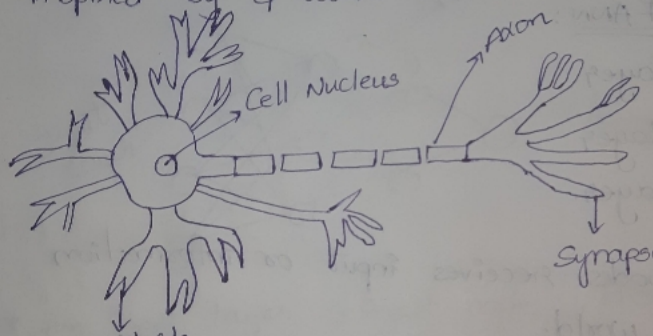
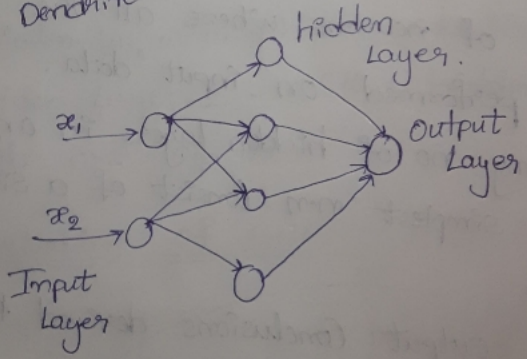
$2/2 \log_2 2/2))$

25/03/23. Unit-4

Artificial Neural Network

* ANN is inspired by a working of human brain.
* The human brain has neurons interconnected to one another, ANN also have neurons that are interconnected to one another in various layers of the N/w. This neurons are known as nodes.
* Neural Networks are set of Algorithm that tries to recognise the patterns, relationships & information from the data through the process which is inspired by & works likes the human brain.





Biological	Artifical
* Dentrits	* Inputs
* Cell Nucleus	* Nodes.
* Synapse	* weights
* Axon	* output.

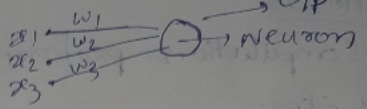
* Components of ANN:
  1. Input layer
  2. Hidden Layer
  3. Output Layer.

1. I.L: Input nodes receives input or information from outside world.
2. H.L: H.L is set of neurons where all the computations are performed on input data.
* There can be any no. of hidden layers in an neural n/w. The simplest n.n consist of a single hidden layer.
3. O.L: O.L is the output Conclusions derived from all the computations performed

* There can be single (or) multiple nodes in the output layer. if we have a binary classification problem the output known is 1. But in case of multi class classification, the output nodes can be more than 1.

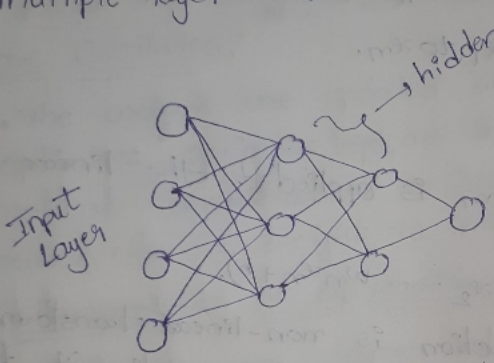
* ~~perception~~:- perception:

* perception is a simple form of N.N/w & Consists of single layer where all the mathematical Computations are performed.



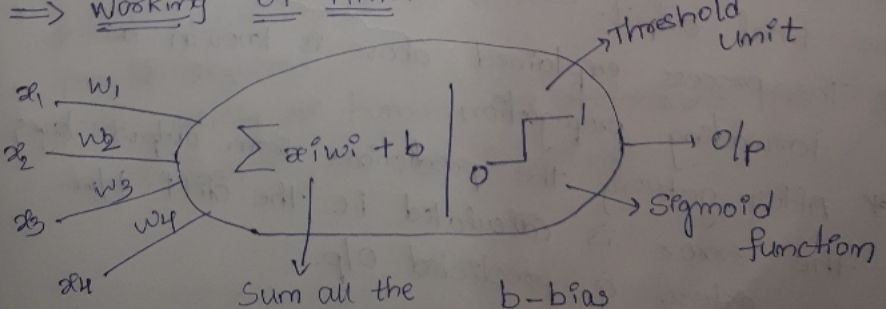
* Multilayer perception:

* M.p is also known as A.N.N Consists of more than 1 perceptron which is grouped together to form multiple layer N.N/w.



* An input layer, 6 input nodes, Hidden Layer-1, ~~Hidden Lay~~ with 4 hidden nodes (or) 4 perceptrons, hidden Layer-2 - with 4 hidden nodes, o/p Layer with one o/p node.

⇒ Working of ANN:



b - bias

x_1, x_2, x_3 - inputs

w_1, w_2, w_3 - weights of Connection.

* bias is the Constant that helps the model to fit in the best way possible.

27/03/23

* Computation perform in hidden Layers done in 2 steps:

1. All the inputs are multiplied by their weights

Step-1:

$$Z_1 = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

where,

$w_1, w_2, w_3 \cdots w_n$ are the weights assigned to inputs x_1 to x_n.

$b \rightarrow$ bias.

Step-2:

Activation function is applied to the linear equation.

$$z_1 : (z_1 = w_1 x_1 + w_2 x_2 \cdots w_n x_n + b).$$

* The activation function is non-linear transformation that is applied to the input before sending it to the next Layer of neurons.

Step-3:

The Computation results from hidden layer passed to the last layer i.e output layer which gives as the final output.

* The process explained above is known as forwading propagation.

* After getting the predictions from output layer the error is calculated i.e the diff b/w the actual & predicted o/p.

* If the Error is large, then the steps are taken to minimize the error & for the same purpose, Back Propagation is performed.

⇒ Back propagation:

* It is process of updating & finding the Optimal values of weights (or) Co-efficients. which helps to minimise the error. i.e diff b/w the actual & predicted value.

* How the weights are updated & new weights are Calculated?

→ The weights are updated with the help of optimises, optimises (opt), are the methods (or) mathe matical formulation to change attributes of neural N/w i.e weight to minimises the error.

* Characteristics of B.P Algorithm:

* instances are represented by many attributes. may be
* The target function output (make) discrete value, real value, (or) a vector of several real (or) discrete value attributes.
* The training examples may contain error.
* large training times are acceptable.
* fast evalution of the learned target function may be required.
* The ability of humans to understand the learned target function is not important.

29/08/23

*Perceptron learning Algorithm:

1. Initialise weights & threshold.
   * Define $w_i(t)$, $(0 \le i \le n)$, to be the weight from input i at the time t, & θ to be the threshold value in the output node.
   * Set w_0 to be $-\theta$, the bias, & x_0 to be always 1.
2. present input & desired output.
   * present input $x_0, x_1, x_2 \dots x_n$ & desired o/p $d(t)$.
3. Calculate actual o/p.

$$y(t) = f_n\left[\sum_{i=0}^{n} w_i(t)\, x_i(t)\right]$$

4. Adapt weights.

if Correct $w_i(t+1) = w_i(t)$

* if o/p 0, should be 1 (class A) $w_i(t+1) = w_i(t) + x_i(t)$.
* if o/p 1, should be 0 (class B) $w_i(t+1) = w_i(t) - x_i(t)$.

→ Adapt weights - modified Version.

* if Correct, $w_i(t+1) = w_i(t)$.
* if output 0, should be 1 (class A), $w_i(t+1) = w_i(t) + \eta x_i(t)$.
* if output 1, should be 0 (class B), $w_i(t+1) = w_i(t) - \eta x_i(t)$.

where $0 \le \eta \le 1$, a positive gain term that Console the adaption rate.

* The error term Δ can be written

$$\Delta = d(t) - y(t).$$

→ Adapt weights — widrow-Hoff delta rule.

$$\Delta = d(t) - y(t).$$

$$w_i(t+1) = w_i(t) + \eta \cdot \Delta x_i(t)$$

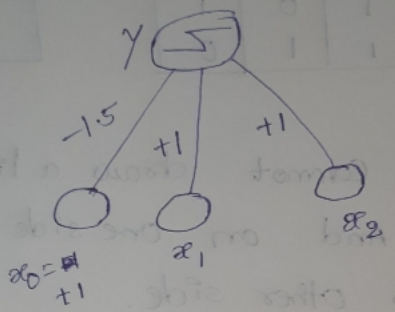
$$d(t) = \begin{cases} +1, & \text{if input from class A} \\ 0, & \text{if input from class B.} \end{cases}$$

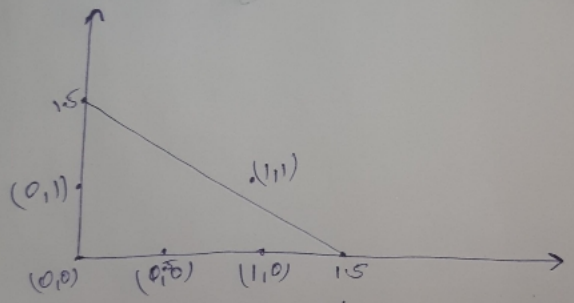
* Input of o/p for the and function:

Learning Boolean function:

AND

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





* In a Boolean function,

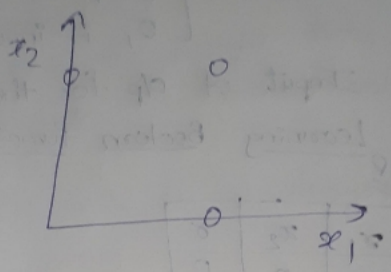
The perceptron that implements And and its geomatic interpretation.

* in a Boolean function, that inputs are binary & output is 1 if the corresponding function value is true ~~(& the inputs)~~ (or) zero otherwise therefore, it can be seen as a 2 class classification problem.

* Input of o/p for XOR function

XOR.

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



* we cannot draw a line where empty circle and on one side & filled circles are on other side.

*XOR process is not linearly Separable.

Multilayer Perceptron:-

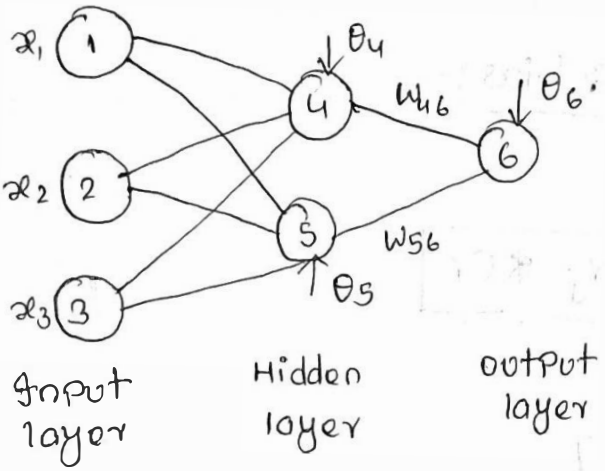
→ Multilayer Perceptron have been applied to solve difficult

→ It consists three layers.

=> Input layer

=> one or more hidden layer

=> Output layer



Algorithm:-

Step-1:- Initialize weights & thresholds set all weights and thresholds to small random value.

Step-2:- Present the input & the desired output

Input $xp = x_0, x_1, x_2, \ldots x_{n-1}$ 'n' is the no. of input nodes.

Target output = $t_0, t_1, t_2, \ldots t_{m-1}$ 'm' is the no. of o/p nodes.

Set w_0 is $-\theta$ called bias and x_0 is always +1 active neuron.

Step-3:-

Calculate the actual output

$$O_j = \frac{1}{1 + e^{-ij}}$$

and passes that as input to the next layer the final layer o/p values O_{pj}

Error Calculation: Hidden layer

$$Err = O_j(1 - O_j) \sum_K err_K \cdot w_{ik}$$

Input $I_j = \sum_i w_{ij} x_i + \theta_i$

Error of Output layer

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

Updation of weight & bias :-

weight updation

$$w_{ij} = w_{ij} + l * err_j * O_i$$

Bias updation

$$\theta_j = \theta_j + l * err_j$$

Multilayer Perception

→ Consider the following multilayer feed forward neural network. Let the Learning rate is 0.9.

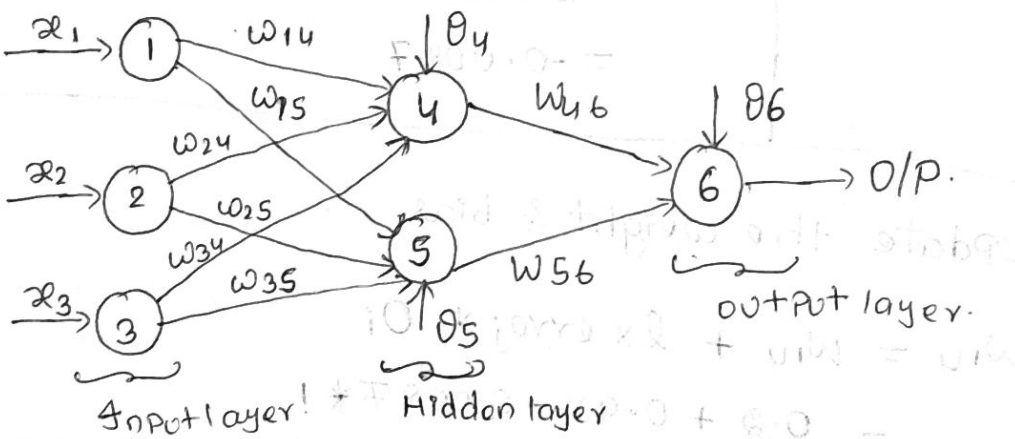
Initial input, weights, bias value are given in the table.

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4

θ_5	θ_6
0.2	0.1

Training tuple is (1, 0, 1) - Target value is '1'

Sol



(a) Calculate the input & output.

Unit	Input	Output
4	$w_{14} \times x_1 + w_{24} \times x_2 + w_{34} \times x_3 + \theta_4 \times \text{Target}$. $\Rightarrow 1 \times 0.2 + 0 \times 0.4 + 1 \times -0.5 - 0.4 \times 1 = -0.7$	$\Rightarrow \frac{1}{1+e^{-ij}}$ $\Rightarrow \frac{1}{1+e^{0.7}} = 0.332$
5	$\Rightarrow w_{15} \times x_1 + w_{25} \times x_2 + w_{35} \times x_3 + \theta_5 \times \text{Target}$. $\Rightarrow 1 \times -0.3 + 0 \times 0.1 + 1 \times 0.2 + 0.2$ ~~$\theta_6 \times$ Target~~ $\Rightarrow 0.1$	$\Rightarrow \frac{1}{1+e^{-0.1}} = 0.525$
6	$\Rightarrow w_{46} \times Out_4 + w_{56} \times Out_5 + Tar \times \theta_6$ $\Rightarrow 0.332 \times -0.3 + 0.525 \times -0.2 + 0.1$ $\Rightarrow -0.105$	$\Rightarrow \frac{1}{1+e^{0.105}} = 0.474$

* Calculate the error rate

unit	Error rate calculation
6	$Err_j = O_j(1-O_j)(Tar_j - O_j)$ $= 0.474(1-0.474)(1-0.474)$ $= 0.1311$
5	$Err = O_j(1-O_j) \sum err_k \cdot w_{ik}$ $= 0.525(1-0.525) * 0.1311 \times -0.2$ $= -0.0065$
4	$Err = O_j(1-O_j) \sum err_k \cdot w_{ik}$ $= 0.332(1-0.332) * 0.1311 \times -0.3$ $= -0.0087$

* update the weight & bias

$$w_{14} = w_{14} + l \times error_j * O_i$$
$$= 0.2 + 0.9 \times -0.0087 * 1$$
$$= 0.192$$

$$w_{15} = w_{15} + l \times error_j * O_i$$
$$= -0.3 + 0.9 * -0.0065 * 1$$
$$= -0.306$$

$$w_{24} = w_{24} + l \times error_j * O_i$$
$$= 0.4 + 0.9 \times -0.0087 \times 0$$
$$= 0.4$$

$$w_{25} = 0.1 + 0.9 \times -0.0065 \times 0$$
$$= 0.1$$

$$w_{34} = -0.5 + 0.9 \times -0.0087 \times 1$$
$$= -0.508$$

$w_{35} = 0.2 + 0.9 \times -0.0065 \times 1$

$= 0.194$

$w_{46} = -0.3 + 0.9 * 0.1311 \times 0.332$

$= -0.138$

$w_{56} = -0.2 + 0.9 \times 0.1311 \times 0.525$

$= 0.192$

Bias

$\theta_6 = \theta_6 + l * errj$

$= 0.1 + 0.9 * 0.1311$

$= 0.218$

$\theta_5 = \theta_5 + l \times errj$

$= 0.2 + 0.9 \times -0.0065$

$= 0.194$

$\theta_4 = \theta_5 + l \times errj$

$= -0.4 + 0.9 \times -0.0087$

$= -0.408$

Thus the weight & θ values are updated.