UNIT-III

Back propagation N/Ws

Architecture of a Backpropagation N/W

1) The Perceptron Model:

a) Rosenblatts' perceptron ⟶ limitations ⟶ Incapable to solve Non-linear separable problems.

b) Initial approach to solve linear inseparable problem

↓

to have more than one perceptron

↓

each perceptron set up Identifying small linearly separable sections of the I/ps

↓ then

combining their o/ps into another perceptron

↓ produce

a final Indication of the class to which the I/p belongs.

c) Ex - XOR problem

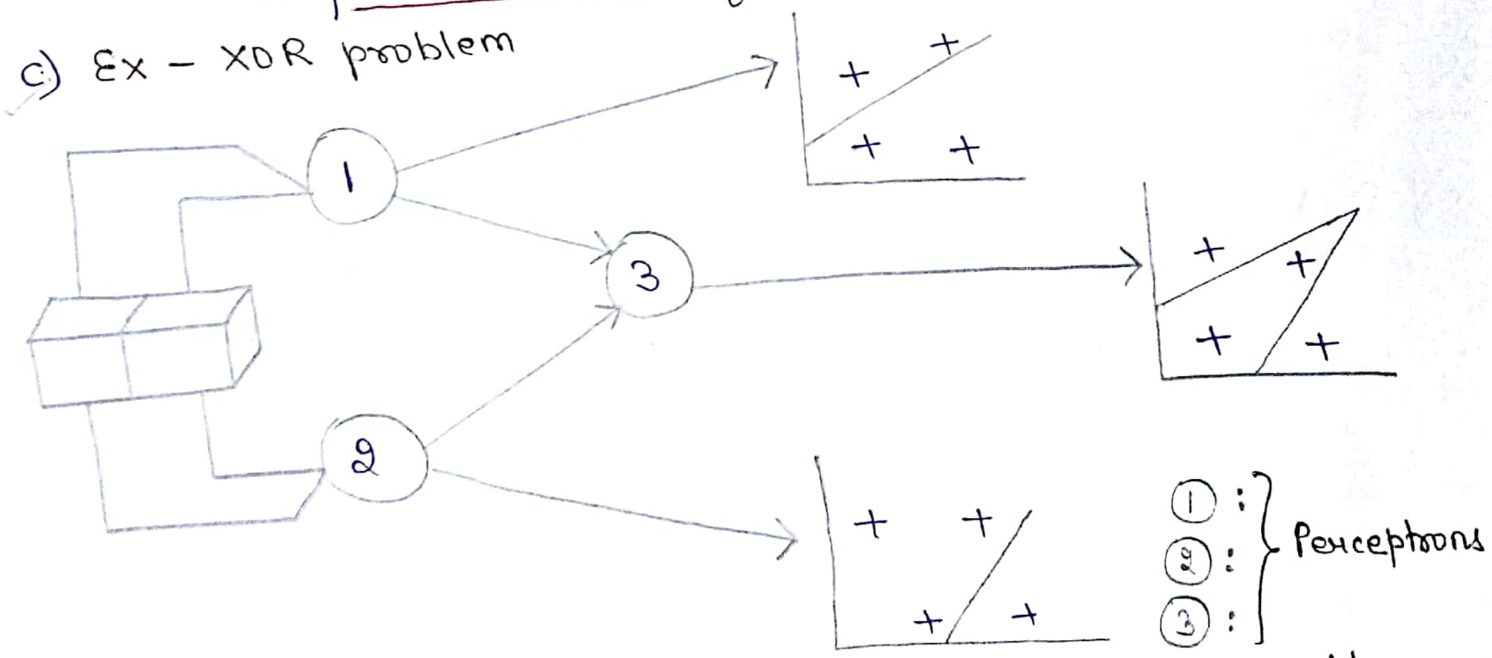


Fig: Combining perceptrons to solve XOR problem.

d) Even though, its looks that the arrangement can solve the problem $\longrightarrow$ but on examination

$\downarrow$ clears that

this arrangement of perceptron in layers

$\downarrow$

Unable to learn.

e) Initially, each neuron in the structure $\xrightarrow{\text{takes}}$ weighted sum of I/ps $\xrightarrow[\text{it}]{\text{threshold}}$ o/ps either 1 or 0

f) For perceptron in the first layer $\xrightarrow[\text{comes from}]{\text{I/p}}$ Actual I/p of the problem.

while, for perceptron in the second layer $\longrightarrow$ the I/ps are o/ps of the first layer.

g) The perceptron of the second layer $\xrightarrow[\text{know}]{\text{do not}}$ which of the real I/ps from the first layer were on or off

XOR $\Rightarrow$ $Y = x_1\bar{x}_2 + \bar{x}_1 x_2$

$Y = z_1 + z_2$

x_1	x_2	z_1
0	0	0
0	1	0
1	0	1
1	1	0

x_1	x_2	z_2
0	0	0
0	1	1
1	0	0
1	1	0

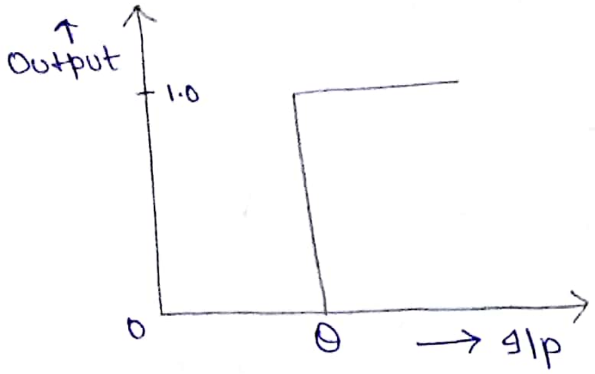
$\longrightarrow$

x_1	x_2	z_1	z_2	Y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

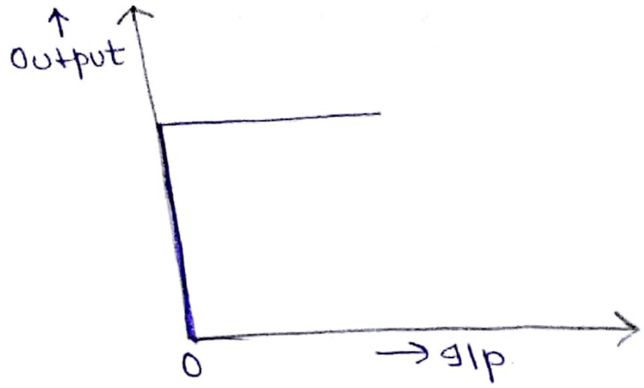
h) Actual I/ps $\longrightarrow$ effectively masked off $\longrightarrow$ from the o/p units by the Intermediate layer.

i) So, it is Impossible $\longrightarrow$ to strengthen the connections b/w active I/ps & strengthen the correct parts of the n/w

j) Two states of Neuron ON or OFF $\xrightarrow[\text{give}]{\text{do not}}$ Any Indication of the scale by which we have to adjust the weights



Threshold at θ



Threshold at 0

Step or Heaviside funⁿs
Fig : Hard-hitting threshold funⁿ.

k) The hard hitting threshold functions → remove the Inform ↓ i.e needed if the n/w is to successfully learn.

l) Hence, the n/w is unable ↓ to determine which of the I/ps weights should be Increased & which one should not. ↓ So, it is unable to work to produce a better solⁿ next time.

m) Difficulty in using step funⁿ as thresholding process ↓ is to adjust it slightly & to use a slightly different non-linearity.

2.) The Solution :

1) Smoothen the threshold funⁿ $\xrightarrow{\text{so that}}$ it more or less turns on or off as before ↓ but has a sloping region in the middle ↓ that will give us Some Inform on the I/ps ↓ So able to determine when we need to strength or weaken the weights.

↓

Now, the n/w will be able to learn as required.

2) Even now, value of o/ps will practically be 1

↓

If the I/ps exceeds the value of the threshold a lot

and o/p will be practically 0 → If the I/p is far less than the threshold

and when I/p and threshold are almost same

↓

o/p of neuron will have value b/w 0 & 1.

3) That means, the o/p of the neuron

↓ can be related

to the I/p in a more Informative way.

New Notations for ANN model

1) An AN is developed to mimic the characteristics & functions of a biological neuron.

2) Analogous to a biological neuron, an AN receives much I/p representing the o/p of the other neuron

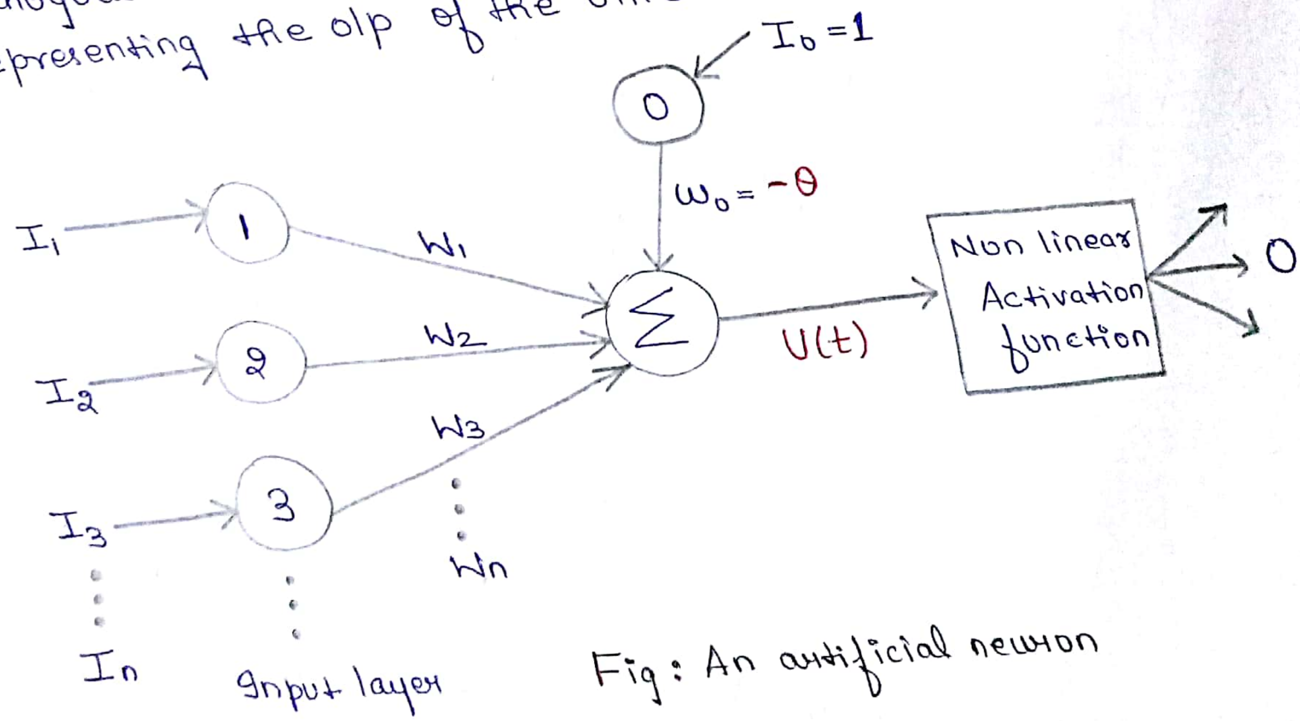


Fig: An artificial neuron

3) Each I/p is multiplied by the corresponding weights analogous to synaptic strengths.

↓

all these weights are summed up

↓

& passed through an Activation funⁿ to determine o/p of neuron.

4) $$u(t) = W_1 I_1 + W_2 I_2 + \cdots + W_n I_n$$

or $$u = \langle W \rangle \{ I \}$$

Considering threshold θ, the relative I/p to the neuron is given by:

$$u(t) = W_1 I_1 + W_2 I_2 + \ldots + W_n I_n - \theta$$

$$= \sum_{i=0}^{n} W_i I_i \quad \text{Where, } W_0 = -\theta, \; I_0 = 1$$

The o/p using the non-linear transfer funⁿ 'f' is given by

$$O = f(u)$$

5) The activation funⁿ $f(u)$ → chosen as a non linear funⁿ to emulate the non linear behavior of conduction current mechanism in a biological neuron.

6) Observation → for sigmoidal funⁿs

↓

o/p of a neuron varies continuously but not linearly with the I/p.

↓

Neurons with sigmoidal funⁿs bear a greater resemblance to biological neurons than with other activation funⁿs.

↓

Even, if sigmoidal funⁿ is differentiated, gives continous value of the o/p.

7) Most commonly used Activation funⁿs in multilayered static neural n/ws are:

Table 3.1 Typical nonlinear activation operators

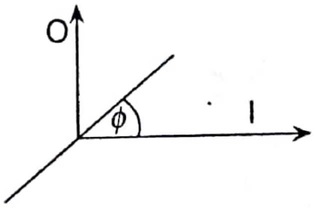
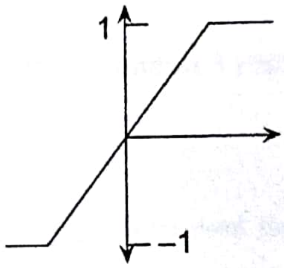
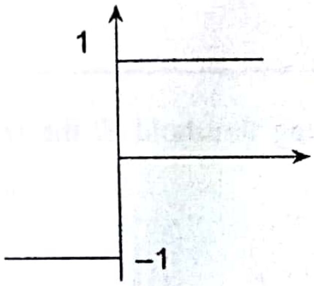
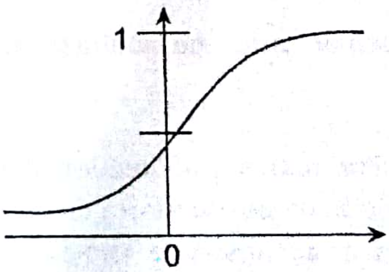
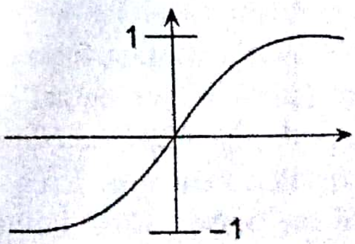
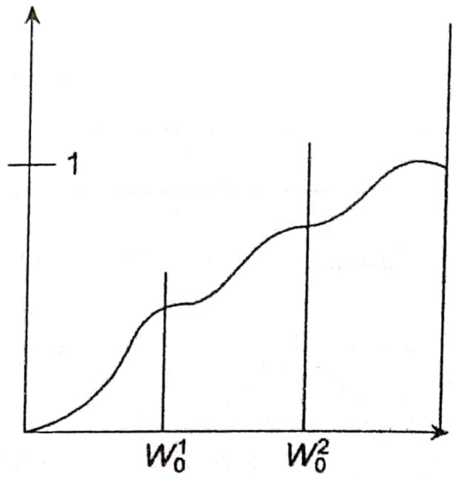
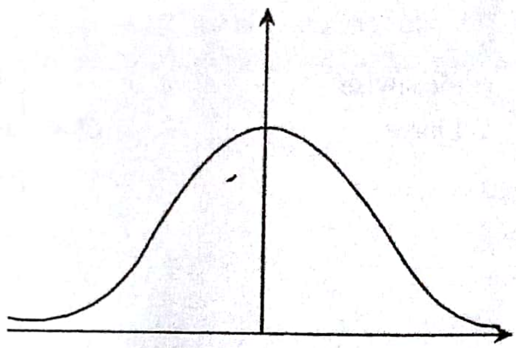
Type	Equation	Functional form
Linear	$O = gI$ $g = \tan \phi$	
Piecewise Linear	$O = \begin{cases} 1 & if \quad mI > 1 \\ gI & if \quad \lvert mI < 1 \rvert \\ -1 & if \quad mI > -1 \end{cases}$	
Hard Limiter	$O = \text{sgn}\ [I]$	
Unipolar Sigmoidal	$O = \dfrac{1}{(1 + \exp(-\lambda I)}$	
Bipolar Sigmoidal	$O = \tanh\ [\lambda I]$	

Table 3.1 Typical nonlinear activation operators (*cont.*)

Type	Equation	Functional form
Unipolar Multimodal	$O = \frac{1}{2}\left[1 + \frac{1}{M}\sum_{m=1}^{M} \tanh\left(g^{m}(I - W_O^{m})\right)\right]$	
Radial Basis Function (RBF)	$O = \exp(I)$ $I = \left[\dfrac{-\sum_{i=1}^{N}(W_i(t) - X_i(t))^2}{2\sigma^2}\right]$	

3) Single Layer Artificial Neural N/w :

1) Although a single neuron can perform certain simple pattern detection problems, we need larger n/ws to offer greater computational capabilities. ⟶ In order to mimic the layered structure of certain portions of the brain

2) It consists of an I/p layer to receive the I/ps and an o/p layer to o/p the vectors respectively.

3) I/p layer $\xrightarrow{\text{consists}}$ 'n' neurons &

o/p layer $\xrightarrow{\text{"}}$ 'm' neurons.

W_{ij} ⟶ Indicate weight of the synapse connecting i^{th} I/p neuron to the j^{th} o/p neuron.

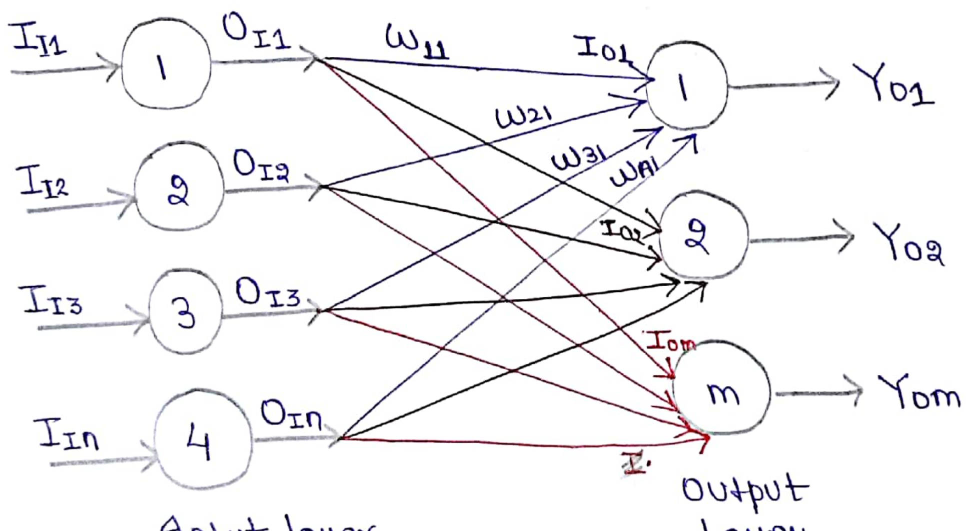


Fig : Single layer feed forward neural n/w.

4) The Inputs of the I/p layer & the corresponding o/ps of the o/p layer are given as :

$$I_I = \begin{Bmatrix} I_{I1} \\ I_{I2} \\ \vdots \\ I_{In} \end{Bmatrix}_{n \times 1} \qquad O_o = \begin{Bmatrix} O_{o1} \\ O_{o2} \\ \vdots \\ O_{om} \end{Bmatrix}_{m \times 1}$$

5) Neurons in the I/p layer use linear transfer funⁿ.

$$\{O_I\}_{n \times 1} = \{I_I\}_{m \times 1} \quad \text{(linear transfer fun}^n\text{)}$$

6) Neurons in the o/p layer use unipolar sigmoidal funⁿ.

Scanned by CamScanner

$$I_{oj} = W_{1j} I_{I1} + W_{2j} I_{I2} + \dots + W_{nj} I_{IN}$$

Hence, i/p to the o/p layer can be given as:

$$\{I_o\}_{m\times 1} = [W]^T \{O_I\} = [W]^T_{m\times n} \cdot \{I_I\}_{n\times 1}$$

7) Using unipolar sigmoidal or squashed - S funⁿ, for neurons in the o/p layer, the o/p is given by.

$$O_{OK} = \frac{1}{(1+e^{-\lambda I_{OK}})}$$

where,
λ = sigmoidal gain
$[W]$ = weight matrix or Connection matrix

or $$\{O_{OK}\} = f\,[WI]$$

8) Each Activation value is a scalar product of i/p w.r.t weight vector. The sigmoidal funⁿ (non-linear Activation funⁿ) is given as:

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

and $$f'(I) = \lambda f(I)\,(1-f(I))$$ (when differentiated)

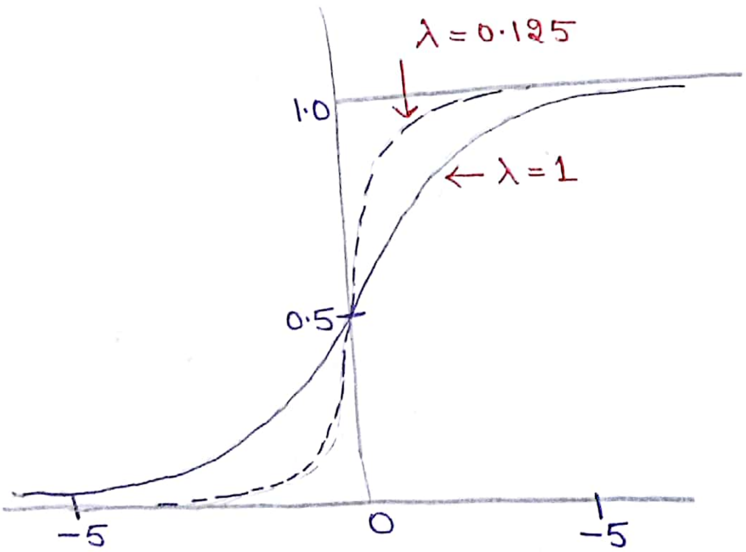


Fig: Squashed-S funⁿ for various values of λ

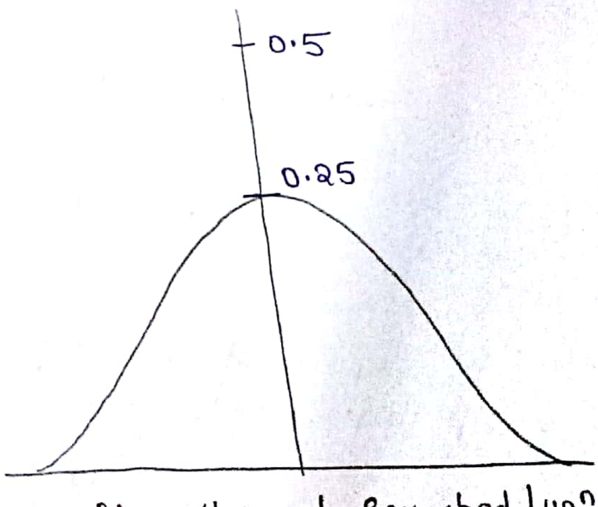


Fig: Slope of Squashed funⁿ

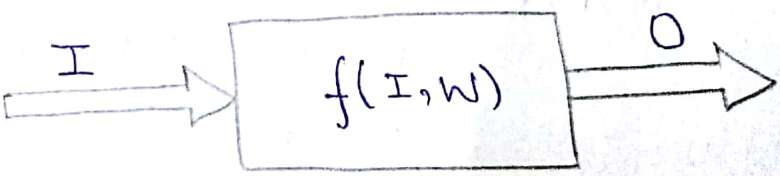


Fig: Block diagram of a single layer feedforward neural n/w.

4) Model of Multilayer Perceptron :

1) The adapted perceptrons $\xrightarrow[\text{in layers}]{\text{arranged}}$ so model is termed as multilayer Perceptron

2) Model has three layers :
   a) an I/p layer
   b) O/p layer
   c) a layer in b/w not connected directly to the I/p or O/p, hence called hidden layer.

3) Use linear transfer funⁿ for perceptrons in I/p layer

4) Use sigmoidal or squashed -S funⁿs for perceptrons in hidden & O/p layer.

5) I/p layer serves to distribute the values they receive to the next layer so, does not perform a weighted sum or threshold.

6) The Input-output of multilayer perceptron is represented by :

$$O = N_3[N_2[N_1[I]]]$$

where, N_1, N_2 & N_3 represent nonlinear mapping provided by I/p, hidden & O/p layer respectively.

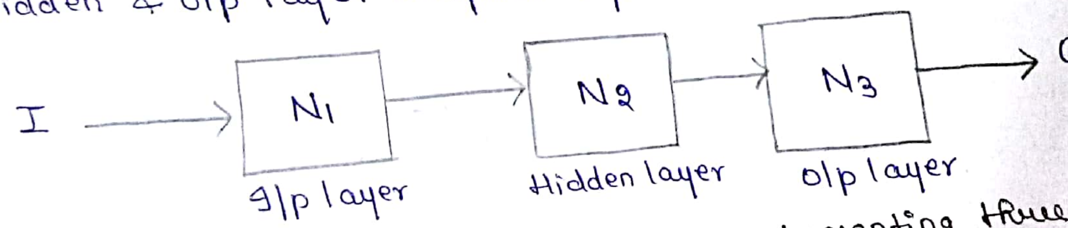


Fig : Block diagram representing three-layer ANN

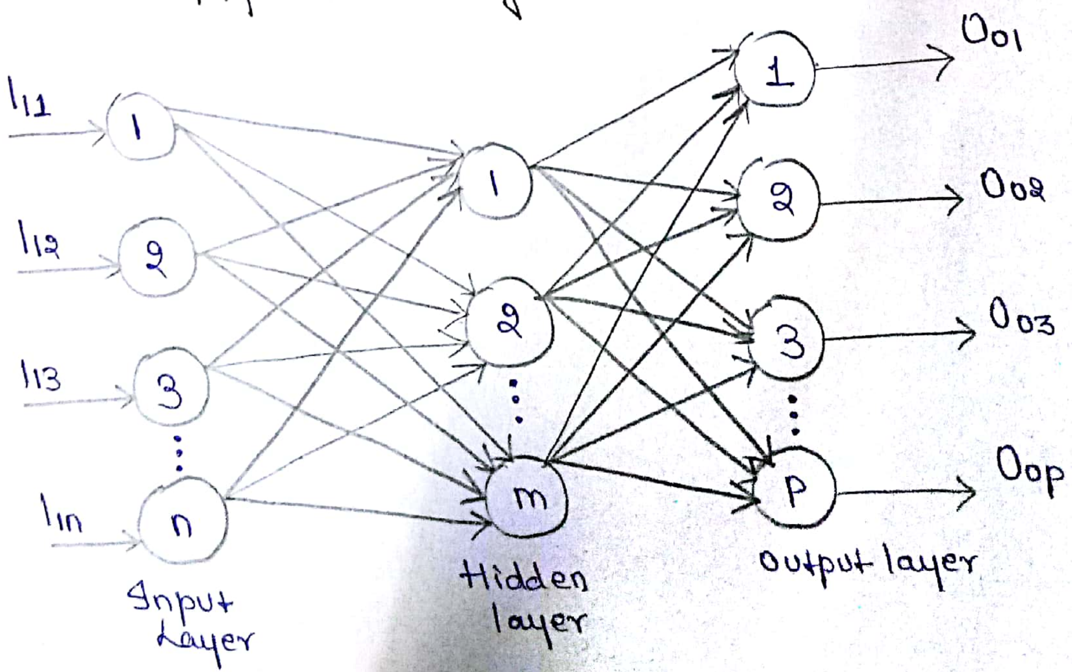


Fig : Multilayer Perceptron

BackPropagation Learning

1) Back-propagation is a systematic method of training multi-layer artificial neural n/ws.
2) This learning algorithm is applied to multilayer feed forward n/ws consisting of processing elements with continous differentiable activation functions.
3) The n/ws associated with back-propagation learning algorithm are also called Back-propagation networks (BPNs).
4) For a given set of training input-output pair, this algorithm provides a procedure for changing the weights in a BPN to classify the given Input patterns correctly.
5) The basic concept of this weight update algorithm is simply gradient-descent method.
6) This is a method where the error is propagated back to the hidden unit.
7) The aim of a neural n/w is to train the n/w to achieve a balance b/w the n/w's ability to respond (memorization) to the I/p and its ability to give reasonable responses to the Input that is similar but not Identical to the one i.e used in training (generalization).
8) The back-propagation algorithm is different from other networks in respect to the process by which the weights are calculated during the learning period of the n/w.
9) The general difficulty with the multilayer perceptron is calculating the weights of the hidden layers in an efficient way that would result in a very small or zero o/p error.
10) When the hidden layers are Increased the n/w training become more complex. To update weights, the error must be calculated. The error, which is the difference b/w the actual (calculated) & the desired (target) o/p, is easily measured at the o/p layer.
11) It should be noted that at the hidden layers, there is no direct Inform of the error. Therefore, other techniques should

be used to calculate an error at the hidden layer, which will cause minimization of the o/p error, and this is the ultimate goal.

12) The training of the BPN is done in three stages:

a) The feed forward of the i/p training pattern, the calculation

b) Back-propagation of the error

c) Updation of weights.

13) The testing of the BPN involves the computation of feed-forward phase only.

14) There can be more than one hidden layer (more beneficial) but one hidden layer is sufficient.

15) Even though the training is very slow, once the n/w is trained it can produce its o/p very rapidly.

Architecture :

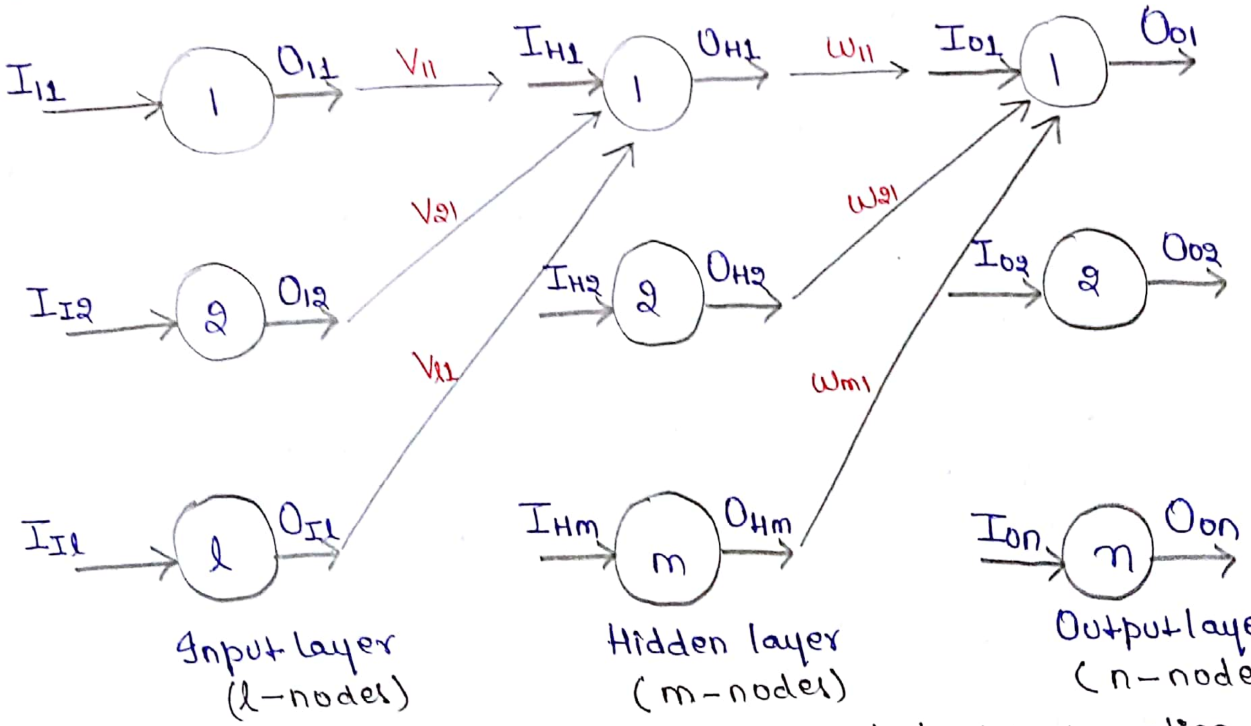


Fig: Multilayer feedforward backpropagation n/w.

where the subscripts I, H, O denote i/p, hidden and o/p neurons.

1.) Input Layer Computation :

a.) The o/p of the i/p Layer is input of i/p layer since linear activation function is used (considering $g = \tan\phi = 1$).

Taking one set of data

$$\underset{l\times 1}{\{O\}_I} = \underset{l\times 1}{\{I\}_I}$$

b) The hidden neurons are connected by synapses to i/p neurons and V_{ij} = weight of the arc b/w ith i/p neuron to jth hidden neuron.

c.) The i/p to the hidden neuron is the weighted sum of the o/ps of the i/p neurons to get I_{HP} (i.e i/p to the pth hidden neuron) :

$$I_{Hp} = V_{1p}O_{I1} + V_{2p}O_{I2} + \dots + V_{lp}O_{Il}$$ where, $p = 1, 2, 3, \dots, m$.

d.) Denoting weight matrix or Connectivity matrix b/w i/p neurons and hidden neurons as $[V]_{l\times m}$, we get an i/p to the hidden neuron as :

$$\underset{m\times 1}{\{I\}_H} = \underset{m\times l}{[V]^T}\ \underset{l\times 1}{\{O\}_I}$$

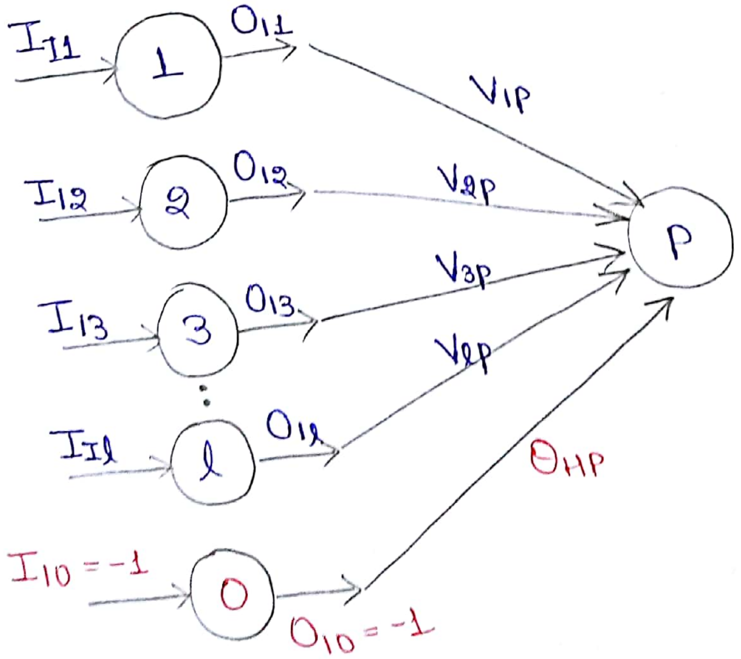


fig : Treating threshold in hidden layer

2) Hidden Layer Computation :

1) Applying sigmoidal funⁿ or squashed-S funⁿ, the o/p of the pth hidden neuron is :

$$O_{Hp} = \frac{1}{\left(1 + e^{-\lambda(I_{Hp} - \theta_{Hp})}\right)}$$

where, O_{Hp} = o/p of the pth hidden neuron

I_{Hp} = i/p of the pth hidden neuron

θ_{Hp} = threshold of the pth neuron.

2) A non-zero threshold neuron is computationally equivalent to an i/p i.e always held at -1 and the non-zero threshold becomes the connecting weight values.

3) o/p to the hidden neuron is given as :

$$\{O\}_H = \begin{bmatrix} - \\ - \\ \dfrac{1}{\left(1 + e^{-\lambda(I_{Hp} - \theta_{Hp})}\right)} \\ - \\ - \end{bmatrix}$$

Here each component of the i/p of the hidden neuron is treated separately and the o/p of the hidden neuron is obtained as above.

3) The i/p to the o/p neurons is the weighted sum of the o/ps of the hidden neurons.

$$I_{oq} = W_{1q} O_{H1} + W_{2q} O_{H2} + \dots + W_{mq} O_{Hm}$$

$q = 1, 2, 3, \dots n$

I_{oq} - i/p to the q^{th} o/p neuron.

4) Denoting weight matrix or connectivity matrix b/w hidden neurons & o/p neurons as $[W]$, we can get i/p to the o/p neuron as :

$$\underset{n \times 1}{\{I\}_o} = \underset{n \times m}{[W]^T} \underset{m \times 1}{\{O\}_H}$$

3) Output Layer Computation :

1) Applying sigmoidal funⁿ, the o/p of the q^{th} o/p neuron is given by :

$$O_{oq} = \frac{1}{\left(1+e^{-\lambda(I_{oq}-\theta_{oq})}\right)}$$

where, O_{oq} = o/p of the q^{th} o/p neuron.
I_{oq} = i/p to the q^{th} o/p neuron
θ_{oq} = threshold of the q^{th} neuron.

2) An extra 0^{th} neuron in the hidden layer with o/p of -1 & the threshold value θ_{oq} becomes the connecting weight value.

3) The o/p of o/p neuron are given by :

$$\{O\}_o = \begin{bmatrix} - \\ - \\ \dfrac{1}{\left(1+e^{-\lambda(I_{oq}-\theta_{oq})}\right)} \\ - \\ - \end{bmatrix}$$

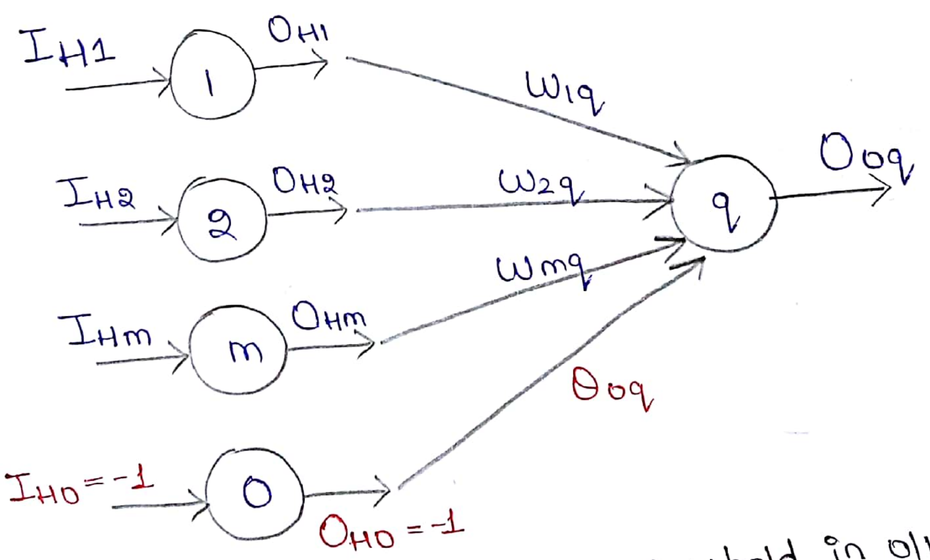


Fig : Treating threshold in o/p layer.

Scanned by CamScanner