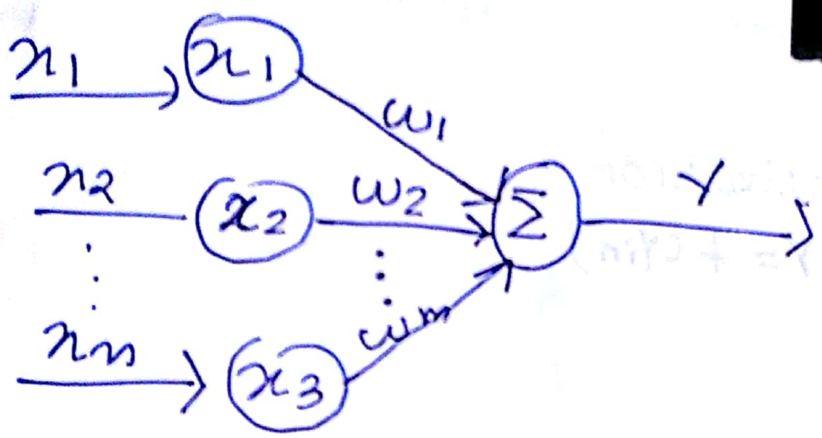
2

Supervised Learning

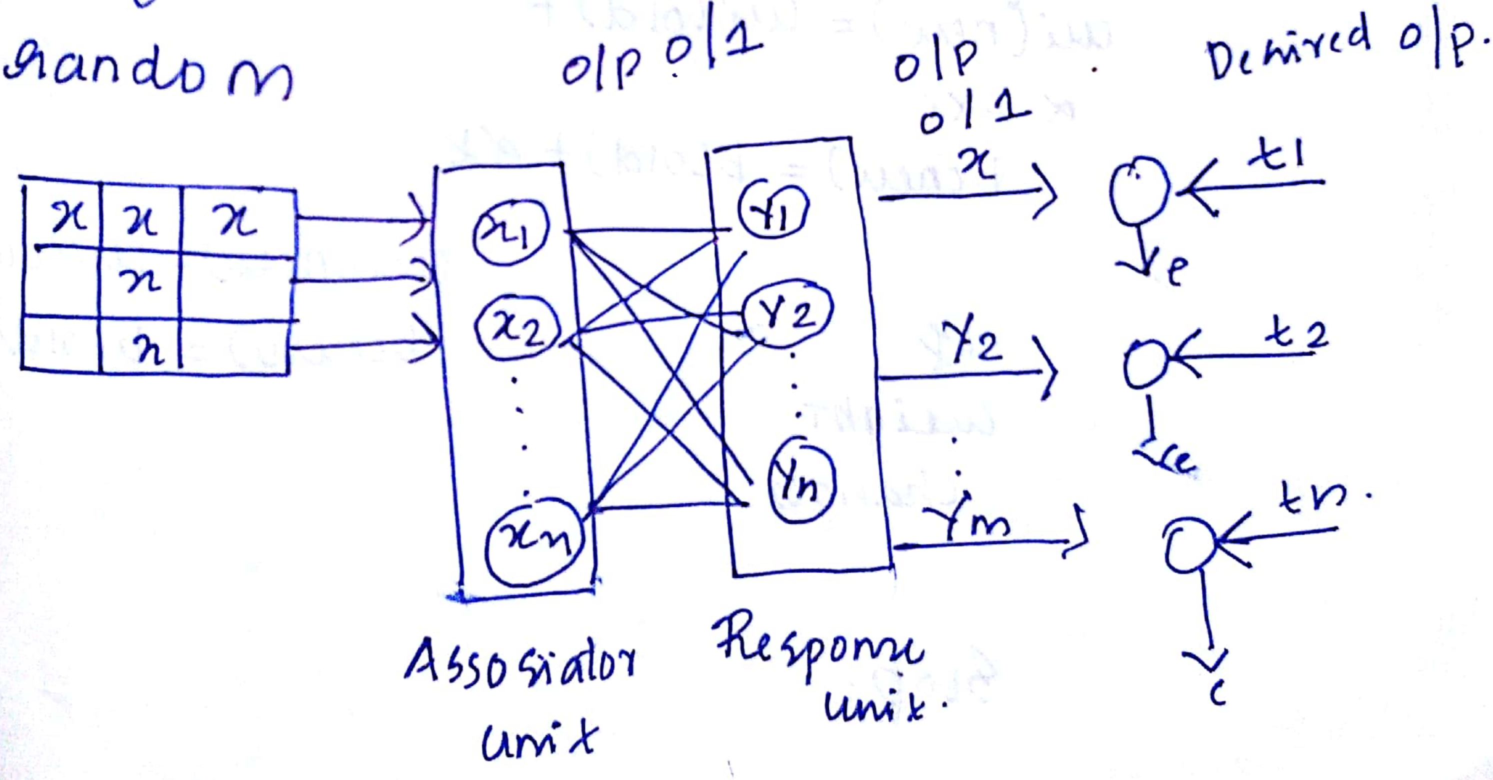
Perceptron network.

Architecture :-



weight fixed

random



Start

Initialize weight and bias.

Set α (0 to 1)

for each s:t

Activate i/p units
$x = s_i$

calculate net i/p
Y_{in}

Apply activation
Obtain $Y = f(Y_{in})$

if
$Y != 1$

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

$b(new) = b(old) + \alpha x$

if weight changes

$w_i(new) = w_i(old)$
$b(new) = b(old)$

Stop.

Learning rule:

$$Y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } \theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

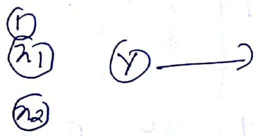
if $y \neq t$ then

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

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

Q Implement AND function using perceptron network for bipolar i/p's and targets.

x_1	x_2	b	y	t
1	1	1	1	1
1	-1	1	-1	-1
-1	1	1	-1	-1
-1	-1	1	-1	-1



$\Delta w_1 = \alpha t x_1$ $\quad w_1(new) = w_1(old) + \alpha t x_1$

$\Delta w_2 = \alpha t x_2$ $\quad w_2(new) = w_2(old) + \alpha t x_2$.

$\Delta b = \alpha t$ $\quad b(new) = b(old) + \alpha t$.

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

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

AND function using preceptron neuron

Input x_1	x_2	1	Target (t)	Net i/p (y_{in})	Calculated o/p (y)	Weights changes Δw_1	Δw_2	Δb	Weights w_1	w_2	b
1	1	1	1	0	0	1	1	1	1	1	1
1	-1	1	-1	1	1	-1	1	-1	0	2	0
-1	1	1	-1	2	1	+1	-1	-1	1	1	-1
-1	-1	1	-1	-3	-1	0	0	0	1	1	-1

EPOCHI-1

Target ≠ calculated ~~o/p~~ output.

EPOCH I 1

x_1	x_2	1	Target (t)	Net i/p y_{in}	Calculated o/p	Δw_1	Δw_2	Δb	w_1	w_2	b
-1	-1	1	-1	-3	-1	-1	1	-1	2	2	-2

EPOCH 2

x_1	x_2	1	(t)	y_{in}	o/p	Δw_1	Δw_2	Δb	w_1	w_2
1	1	1	1							
1	-1	1	-1							
-1	1	1	-1							
-1	-1	1	1							

$y_{in} = 2 + (-1 \times 2) + (-1 \times 2) = +2 + 2 - 2 = \cancel{6}\ 2$

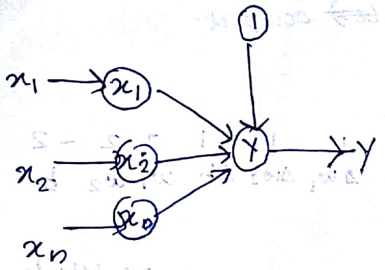
$y = 1$

$\Delta w_1 = 1 \times 1 \times 1 = 1$

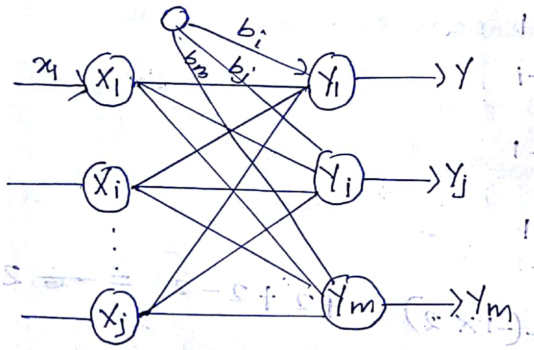
$\Delta w_2 = 1 \times 1 \times 1 = 1$

$\Delta b = 1 \times 1 = 1$

y_{in}

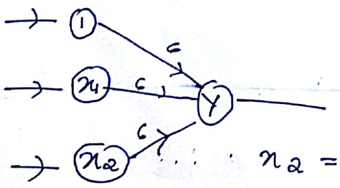


Single classification of perceptron Neuron.

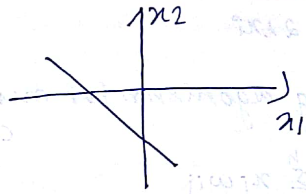


$b + x_1 w_1 + x_2 w_2 = 0.$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b_1}{b_2}$$



$x_2 = -x - 1$



Preceton

O/p layers has more than one nodes with weight for each.

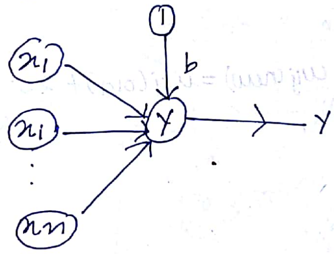


fig: single classification of perceptron neuron.

Perceptron training Algorithm for single class.

Step 4: $y_{in} = b + \sum_{i=1}^{n} x_i w_i$

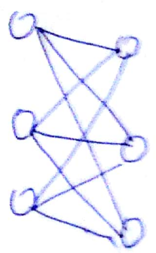
$$y = f(in) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

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

Perceptron training algorithm for multiple class:

Step 4 : $y_{in}j = b_j + \sum_{i=1}^{n} x_i w_{ij}$

$$y_j = f(y_{inj}) = \begin{cases} 1 & \text{if } y_{inj} > \theta \\ 0 & \text{if } -\theta \le y_{inj} \le \theta \\ -1 & \text{if } y_{inj} < -\theta \end{cases}$$



$w_{ij}(new) = w_{ij}(old) + \alpha t_j x_i$

Adaptive linear Neuron:

Flowchart

Start

↓

Set initial values weights and bias, learning state w, b, α

For each $s:t$

↓

Calculate the net input
$Y_{in} = b + \sum x_i w_i$

↓

weight updation
$w_i(new) = w_i(old) + \alpha(t - Y_{in}) x_i$
$b(new) = b(old) + \alpha(t - Y_{in})$

↓

Calculate error
$E_i = \sum (t - Y_{in})^2$

↓

If $E_i = E_s$

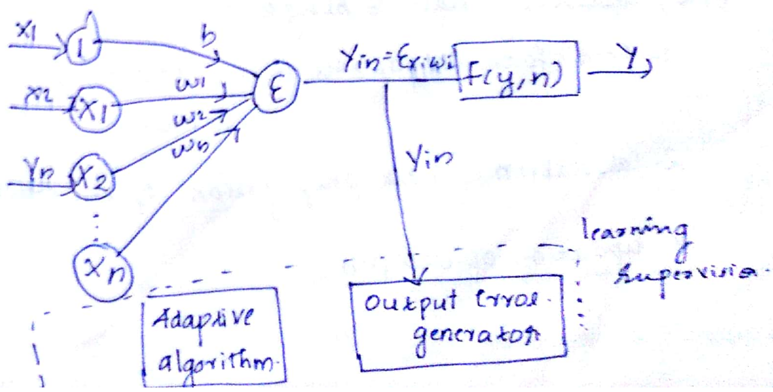
No → 1

No → 2

Yes ↓

1 →

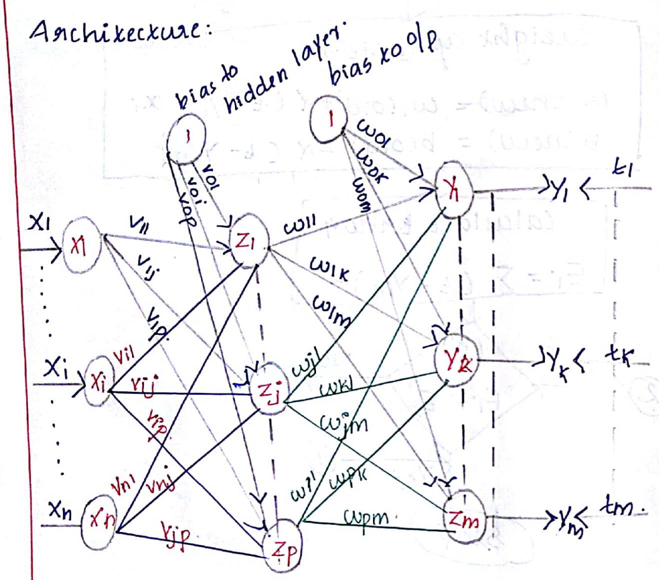
Stop



Back propogation network. (BPN)

BPN. Multilayer feed forward networks

Architecture:



BPN training have 3 stages:-

1- feed forward of the input training Pattern.

2- Calculation of back propogation of the error.

3- updation of weights.

Flowchart for training process.

X = Input training vector $(x_1 \cdots x_j \cdots x_n)$

t = target $(t_1 \cdots t_k \cdots t_m)$

α = learning rate

x_i = Input unit i

v_{oj} = bias on j^{th} hidden unit.

W_{oK} = bias on k^{th} output unit

Z_j = hidden unit j

The net input to Z_j is

$$Z_{inj} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

Output is,

$$Z_j = f(Z_{inj})$$

Y_K = output unit K

The net input to Y_K is

$$Y_{inK} = W_{oK} + \sum_{j=1}^{p} z_j w_{jK}$$

Output is,

$$y_K = f(y_{inK})$$

δ_K = error correction weight adjustment for w_{jK} due to an error at output unit y_K.

δ_i = error correction weight adjustment for v_{ij}

Start

Initialize the weights to some random values

C

For each training pair → B

Receive i/p signal x_i and transmit to hidden unit.

In hidden unit calculate

$z_{inj} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$

$z_j = f(z_{in})$, $j = 1$ to p, $i = 1$ to n.

Send z_j to the o/p layer units.

Calculate o/p signal from o/p layer

$y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$

$y_k = f(y_{ink})$, $k = 1$ to m.

Target pair t_k enters

Compute error correction factor

$\delta_k = (t_k - y_k) f'(y_{ink})$

(b/w o/p & hidden

$f'(x) = \lambda f(x)[1 - f(x)]$

Find weight & bias correction term

error factor

$\Delta w_{jk} = \alpha \delta_k z_j$, $\Delta w_{ok} = \alpha \delta_k$

Calculate error term δ_j

(b/w hidden & i/p)

$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$

$\delta_j = \delta_{inj} f'(z_{inj})$

Compute change in weights & bias based on

error factor

δ_j, $\Delta v_{ij} = \alpha \delta_j x_i$, $\Delta v_{oj} = \alpha \delta_j$

Update weight and bias on o/p unit

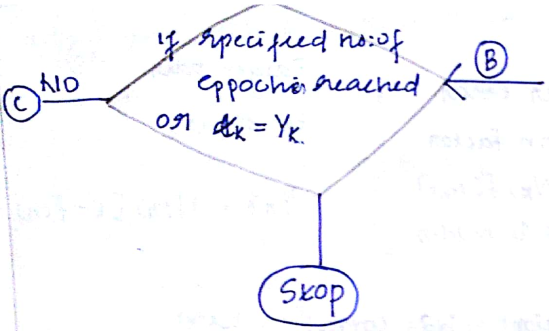
$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$

$w_{ok}(new) = w_{ok}(old) + w_{ok}$

Update weight and bias on hidden unit

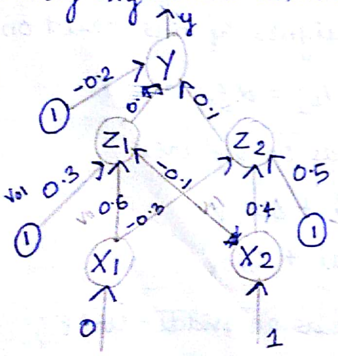
$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$

$v_{oj}(new) = v_{oj}(old) + \Delta v_{oj}$



Algorithm- Page no: 69

Q Using back propogation network the new weight for the net shown in figure it is presented with the i/p patter and the target o/p address is 1 using a learning gate $\alpha = 0.25$ and binary sigmoidal activation function.



Section 3.5.

Initial weights are $[v_{11}\ v_{21}\ v_{01}] = [0.6, -0.1, 0.3]$

$[v_{12}\ v_{22}\ v_{02}] = [-0.3\ \ 0.4\ \ 0.5]$

$[w_1\ w_2\ w_0] = [0.4, 0.1, -0.2]$

$\alpha = 0.25$

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

$[x_1\ x_2] = [0, 1]$, $t = 1$
input target

PDF Neuro fuzzy hybrid
Genetic Neuron
Genetic fuzzy

1. Neuro fuzzy

I Comparison:

Neural Processing	Fuzzy
• mathematical model not necessary	Not
• Learning	No
• algorithms	×
Black box Behaviour	Simple

II Characteristics

III Classification

Hidden layer.

$Z_{in}1 = V_{01} + x_1 V_{11} + x_2 V_{21}$.

$= 0.3 + 0 * 0.6 + 1 * -0.1$

$= 0.3 - 0.1$

$= 0.2$

$Z_{in}2 = 0.5 + 0 \times -0.3 + 1 \times 0.4$

$= 0.5 + 0.4$

$= 0.9$

~~z1 = f(zin1)~~ Applying activation to calculate the o/p, we obtain.

$z_1 = f(z_{in}1) =$

$=$

$z_2 =$

$=$

Calculate the net i/p entering the o/p layer.

for y layer.

$Y_{in} = w_0 + z_1 w_1 + z_2 w_2$

$= -0.2 + 0.5498 \times 0.4 + 0.7109 \times 0.1$

$= 0.09101$.

Applying activations to calculate the output, we obtain

$Y = f(Y_{in}) = \frac{1}{1+e^{-Y_{in}}} =$

KTUStudents
Dedication to Success

Scanned by CamScanner