

Two basic signals flows in MLP

### Back-propagation Algo:

jth neuron  $\rightarrow$  output

$$e_j(m) = d_j(m) - y_j(m)$$

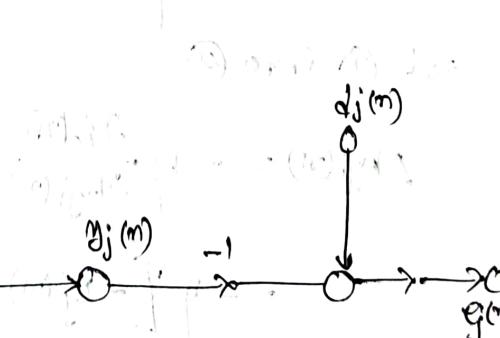
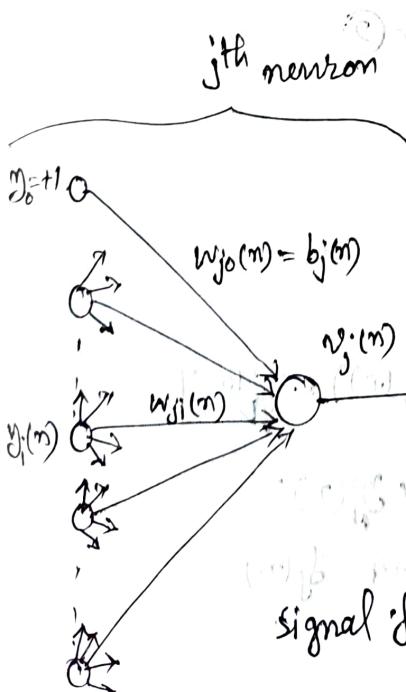
$$\bar{e}(m) = \frac{1}{2} \sum_{j \in C} e_j^2(m)$$

# class

$y_j(m) \rightarrow$  o/p of jth neuron

$d_j(m) \rightarrow$  desired value of jth neuron

$$\bar{e} = \frac{1}{N} \sum_{n=1}^N \bar{e}(m)$$



signal flow graph when j is off top neuron

$$v_j(m) = \sum_{i=1}^m w_{ji}(m) \tilde{y}_i(m)$$

$$\tilde{y}_j(m) = \phi(v_j(m))$$

$$\frac{\partial E(m)}{\partial w_{ji}(m)} = \frac{\partial E(m)}{\partial v_j(m)} * \frac{\partial v_j(m)}{\partial \tilde{y}_j(m)} * \frac{\partial \tilde{y}_j(m)}{\partial v_j(m)}$$

$$\frac{\partial E(m)}{\partial v_j(m)} = e_j(m)$$

$$\frac{\partial v_j(m)}{\partial \tilde{y}_j(m)} = -1$$

$$\frac{\partial \tilde{y}_j(m)}{\partial v_j(m)} = \phi'(v_j(m))$$

$$\frac{\partial v_j(m)}{\partial w_{ji}(m)} = y_i(m)$$

$$\therefore \frac{\partial E(m)}{\partial w_{ji}(m)} = e_j(m) * (-1) * \phi'(v_j(m)) * y_i(m) \rightarrow ①$$

According to delta learning rule (i.e. steepest descent algo.)

$$\Delta \vec{w}(m) = -\eta \vec{g}(m)$$

$$= -\eta \nabla E(m)$$

$$\Delta w_{ji}(m) = -\eta \frac{\partial E(m)}{\partial w_{ji}(m)} \rightarrow ②$$

put ① into ②

$$\Delta w_{ji}(m) = -\eta \frac{\partial E(m)}{\partial w_{ji}(m)}$$

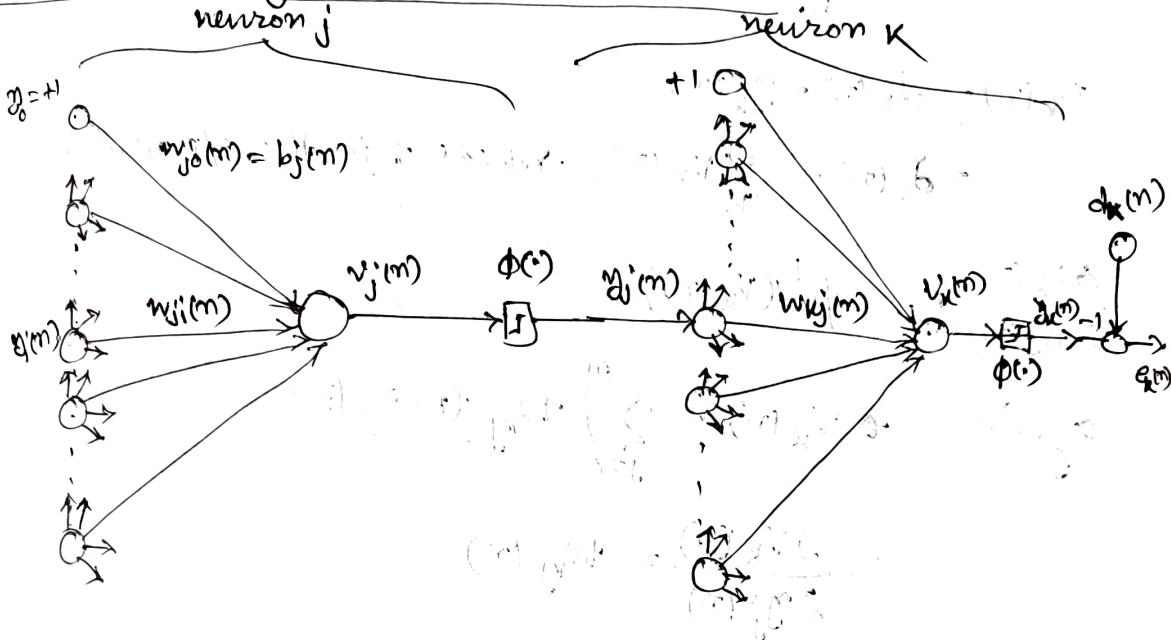
$$= -\eta \left[ -e_j(m) * \phi'(v_j(m)) * y_i(m) \right]$$

$$= \underbrace{\eta e_j(m) * \phi'(v_j(m)) * y_i(m)}_{\text{total gradient } \delta_j(m)}$$

$$\begin{aligned}
 d_j(m) &= -e_j(m) \phi'(v_j(m)) = -e_j(m) * (-) * \phi'(v_j(m)) \\
 &= -\frac{\partial \delta(m)}{\partial e_j(m)} \frac{\partial e_j(m)}{\partial y_j(m)} \frac{\partial y_j(m)}{\partial v_j(m)} \\
 &= -\frac{\partial \delta(m)}{\partial v_j(m)}
 \end{aligned}$$

$\Delta w_{ji}(m) = \eta \cdot d_j(m) \cdot w_{ji}(m)$  when  $j$  is o/p node.

when neuron  $j$  is a hidden neuron:



$$\begin{aligned}
 d_j(m) &= -\frac{\partial \delta(m)}{\partial y_j(m)} + \frac{\partial \delta(m)}{\partial v_j(m)} \\
 &= -\frac{\partial \delta(m)}{\partial y_j(m)} \cdot \phi(v_j(m))
 \end{aligned}$$

$$(w_{ji}(m) \cdot y_i(m) + b_j(m)) \cdot \phi'(v_j(m))$$

$$= (w_{ji}(m) \cdot y_i(m) + b_j(m)) \cdot \phi'(v_j(m)) \cdot \phi(v_j(m))$$

$$(w_{ji}(m) \cdot y_i(m) + b_j(m)) \cdot \phi'(v_j(m)) \cdot \phi(v_j(m))$$

09/10/25  
soft computing

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

where,

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} * \phi'(v_j(n))$$

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \cdot \frac{\partial e_k(n)}{\partial y_j(n)}$$

$$= \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} * \frac{\partial v_k(n)}{\partial y_j(n)}$$

$$e_k(n) = d_k(n) - y_k(n)$$

$$= d_k(n) - \phi(v_k(n)) \text{ where } k \text{ is the o/p neuron}$$

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\phi'_k(v_k(n))$$

$$\text{again, } v_k(n) = \sum_{j=0}^m w_{kj}(n) y_j(n)$$

$$\therefore \frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n)$$

$$\begin{aligned} \text{then, } \frac{\partial E(n)}{\partial y_j(n)} &= -\sum_k e_k(n) \cdot \phi'(v_k(n)) w_{kj}(n) \\ &= -\sum_k \delta_k(n) w_{kj}(n) \end{aligned}$$

$$\delta_j(n) = \phi'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

$$\boxed{\delta_j(n) = \phi'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)}$$

where j is hidden neuron

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

- ① Forward pass  
② Backward pass

① forward pass:

$$v_j(m) = \sum_{i=0}^m w_{ji}(m) y_i(m) \text{ where, } y_i(m) = x_i(m)$$

output,

$$y_j(m) = \phi(v_j(m))$$

jth neuron output,  $y_j(m) = \phi(v_j(m))$

② Backward pass:

$$\Delta w_{ji}(m) = -\eta \frac{\partial E_{\text{ans}}}{\partial w_{ji}}$$

$$w_{ji}(m+1) = w_{ji}(m) + \Delta w_{ji}(m)$$

Activation fn.

$$\Delta w_{ji}(m) = \eta \cdot \delta_j(m) y_i(m)$$

local gradient

(i) Logistic function

(ii) Hyperbolic tangent function

(i) Logistic fn.

$$\phi_j(v_j(m)) = \frac{1}{1 + e^{-\alpha v_j(m)}} \text{ where } \alpha > 0$$

$$\phi'_j(v_j(m)) = \frac{\alpha e^{-\alpha v_j(m)}}{(1 + e^{-\alpha v_j(m)})^2}$$

$$\phi'(v_j(m)) = \alpha y_j(m) [1 - y_j(m)]$$

Neuron j is o/p node:

$$\begin{aligned}
 d_j(m) &= e_j(m) \cdot \phi'(v_j(m)) \\
 &= [d_j(m) - o_j(m)] \cdot \alpha \cdot o_j(m) [1 - o_j(m)] \\
 &= \alpha [d_j(m) - o_j(m)] o_j(m) [1 - o_j(m)]
 \end{aligned}$$

when  $y_j(m) = o_j(m)$

Neuron j is hidden node:

$$\begin{aligned}
 d_j(m) &= \phi'(v_j(m)) \sum_k d_k(m) w_{kj}(m) \\
 &= \alpha y_j(m) [1 - y_j(m)] \sum_k d_k(m) w_{kj}(m)
 \end{aligned}$$

(ii) Hyperbolic Tangent fn.:

$$\phi_j(v_j(m)) = \alpha \tanh(\beta v_j(m)), \quad (\alpha, \beta) > 0$$

$$\begin{aligned}
 \phi'_j(v_j(m)) &= \alpha \beta \cdot \operatorname{sech}^2(\beta v_j(m)) \quad [\operatorname{sech}^2(x) : \text{hyperbolic secant squared}] \\
 &= \alpha \beta \cdot (1 - \tanh^2(\beta v_j(m))) \\
 &= \frac{\beta}{\alpha} [\alpha - y_j(m)] [\alpha + y_j(m)]
 \end{aligned}$$

→ If j is output node,  $y_j(m) = o_j(m)$

$$\begin{aligned}
 d_j(m) &= e_j(m) \cdot \phi'(v_j(m)) \\
 &= \frac{\beta}{\alpha} [d_j(m) - o_j(m)] [\alpha - o_j(m)] [\alpha + o_j(m)]
 \end{aligned}$$

If j is hidden node:

$$\begin{aligned}
 d_j(m) &= \phi'(v_j(m)) \sum_k d_k(m) w_{kj}(m) \\
 &= \frac{\alpha \beta}{\alpha} [\alpha - y_j(m)] [\alpha + y_j(m)] \sum_k d_k(m) w_{kj}(m)
 \end{aligned}$$

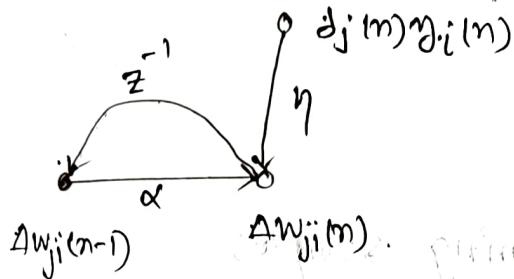
## Rate of learning:

Include a momentum term  $\alpha$

after including momentum term  $\alpha$

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta d_j(n) y_i(n)$$

where, momentum term  $\alpha$  is positive number,  
is also called momentum content



## Signal flow graph

## Modes of training:

i) Sequential Mode: Data are feed to the neuron sequentially

$$\{(x(1), d(1)), (x(2), d(2)), \dots, (x(N), d(N))\}$$

ii) Batch Mode:

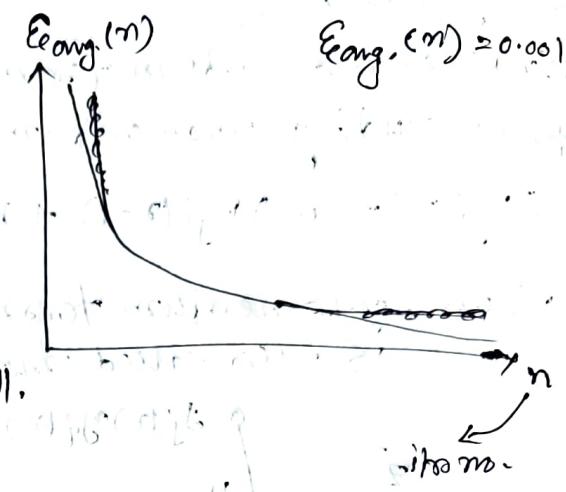
$$E_{avg} = \frac{1}{2N} \cdot \sum_{n=1}^N \sum_{j \in C} e_j^2(n)$$

$$\Delta w_{ji}(n) = -\eta \frac{\partial E_{avg}}{\partial w_{ji}(n)}$$

$$= -\frac{\eta}{N} \cdot \sum_{n=1}^N e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}(n)}$$

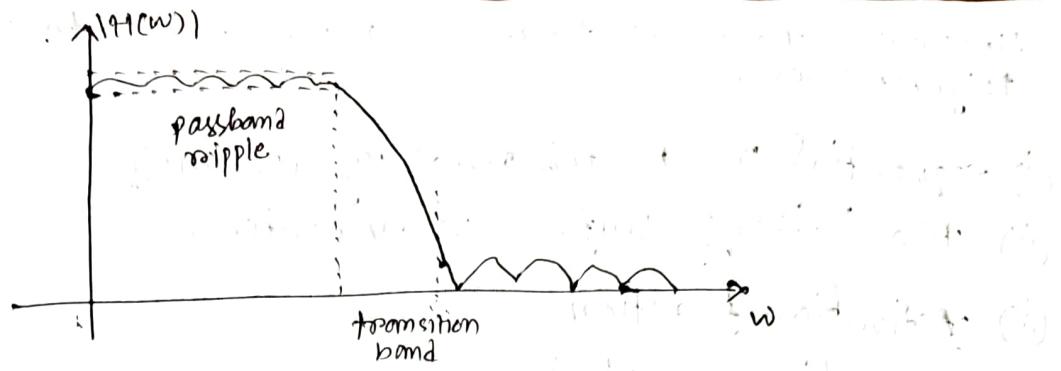
## Stopping Criteria:

- (i) Gradient vector is sufficiently small
- (ii) when absolute rate of change of the avg. square error per epoch is sufficiently small.



## Summary:

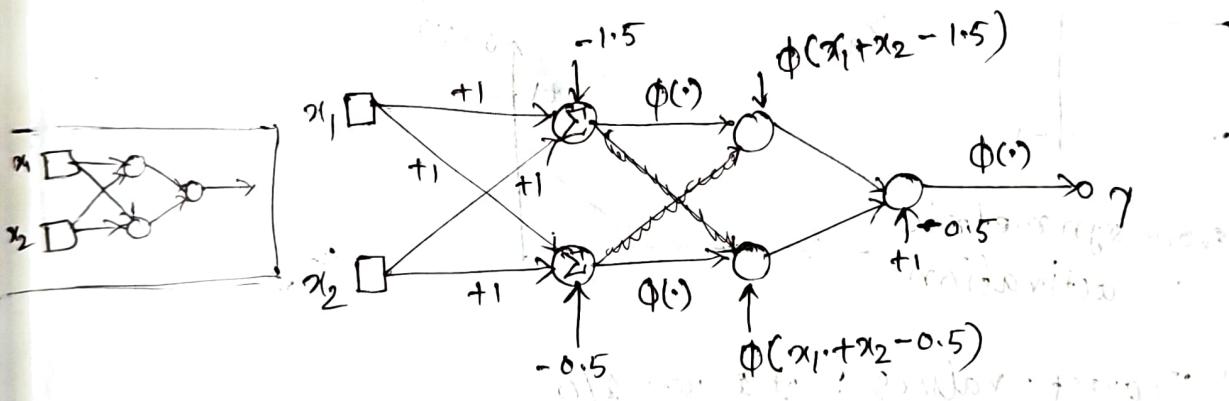
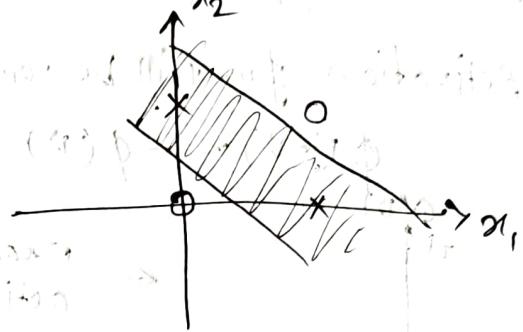
- i) Initialization
- ii) representation of training samples
- iii) forward computation
- iv) backward
- v) Iteration interaction



10/10/25  
soft computing

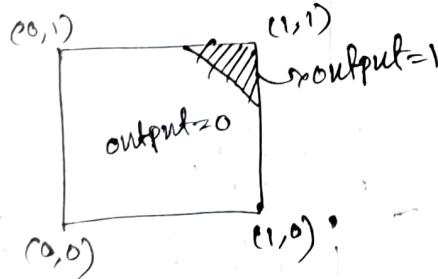
XOR problem:

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

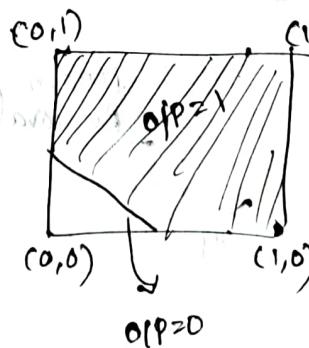


Assume  $\phi(\cdot)$  hard-limit

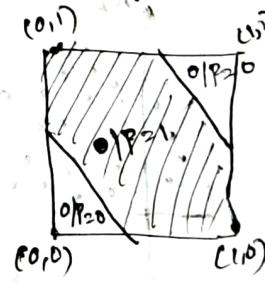
Decision Boundary:



At neuron-1



At neuron-2



At neuron-3

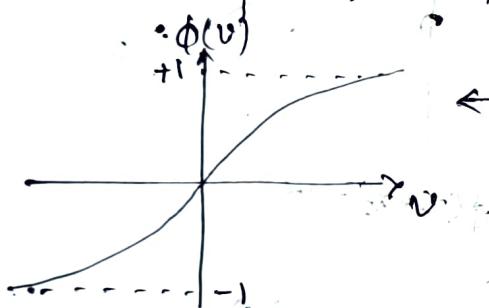
Heuristics to move to BP-MLP algo. with better performance.

- (i) Sequential vs Batch mode of learning
- (ii) Maximizing Information content
- (iii) Activation function

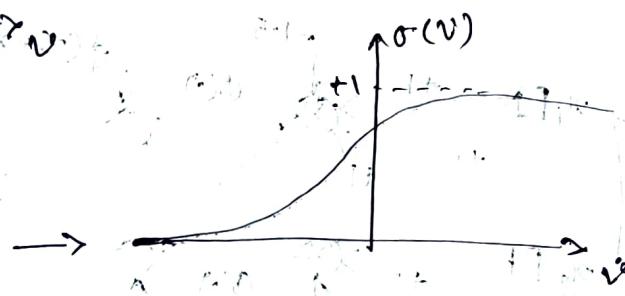
- (A) Anti-symmetric
- (B) Non-symmetric

Activation fn. will be anti-symmetric if

$$\phi(-v) = -\phi(v)$$



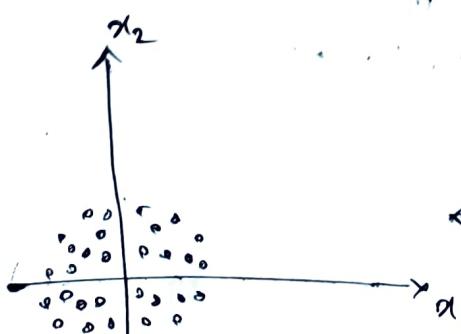
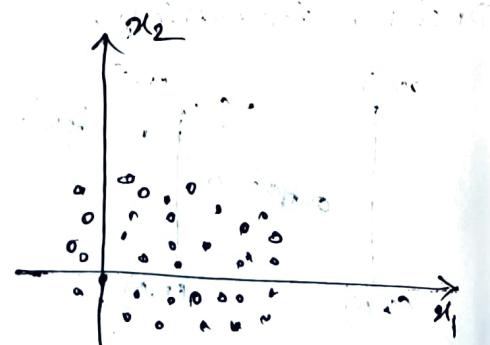
example of anti-symmetric activation fn.  $\boxed{\phi(v) = \tanh(bv)}$



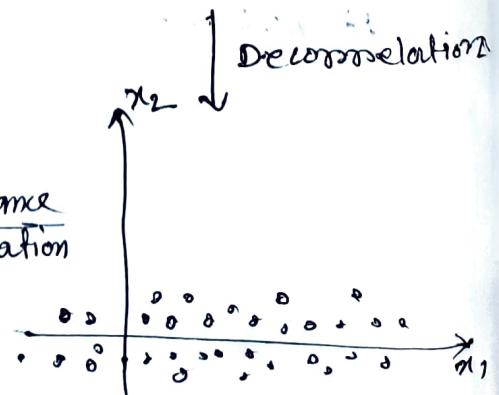
Non-symmetric activation

(iv) Target values:  $\pm 1$  or  $1/0$

(v) Normalizing the inputs



covariance equalization



De-correlation

## ⑥ Initialization of weights

Assume,  $b_0 = 0$

$$y_j = \sum_{i=1}^m w_{ji} y_i \quad \text{if}$$

→ Assume, i/p applied with zero mean & unit variance  
→ Assume, i/p's are uncorrelated

$$\mu_y = E[y_i] := 0 \quad \forall i$$

$$\sigma_y^2 = E[(y_i - \mu_y)^2] \\ = E[y_i^2] \approx 1 \quad \forall i \quad [\because \mu_y = 0]$$

$$E[y_i y_k] = \begin{cases} 1 & \text{for } k=i \\ 0 & \text{for } k \neq i \end{cases}$$

$$\mu_w = E[w_{ji}] = 0 \quad \forall (j,i) \text{ pairs}$$

and variance

$$\sigma_w^2 = E[(w_{ji} - \mu_w)^2] \\ = E[w_{ji}^2] \quad \forall (j,i) \text{ pairs}$$

Synaptic weights are drawn from uniformly distributed set of numbers with zero mean.  
 $\mu_w = E[w_{ji}] = 0$   
 $\forall (j,i) \text{ pairs}$

### Mean & variance of $y_j$

$$\mu_y = E[y_j] = E\left[\sum_{i=1}^m w_{ji} y_i\right] = \sum_{i=1}^m E[w_{ji}] \cdot E[y_i] = 0$$

$$\sigma_y^2 = E[(y_j - \mu_y)^2] = E[y_j^2]$$

$$= E\left[\sum_{i=1}^m \sum_{k=1}^m w_{ji} y_i \cdot w_{ik} y_k\right]$$

$$= \sum_{i=1}^m \sum_{k=1}^m E[w_{ji} w_{ik}] E[y_i y_k]$$

$$= \sum_{i=1}^m E[w_{ji}^2] \cdot \text{for } i=k$$

$$= m \sigma_w^2 \quad \text{where } m : \# \text{ synaptic connection of neuron}$$

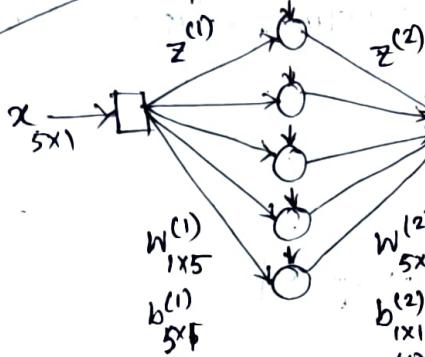
$$\sigma_w^2 = \frac{1}{m} \sigma_y^2 \Rightarrow \sigma_w = \frac{1}{\sqrt{m}} \cdot \sigma_y \quad \text{for } \sigma_y = 1, \sigma_w = m^{-1/2}$$

(vii) Learning from slants

(viii) Learning rate.

13/10/25  
soft computing  
lab

### Experiment - 06



$$y-hat \cdot L = \frac{1}{2} (y - y-hat)^2$$

$$\begin{aligned} \frac{\partial E}{\partial w^{(2)}} &= \frac{\partial E}{\partial y} * \frac{\partial y}{\partial z_2} * \frac{\partial z_2}{\partial w^{(2)}} \\ &= -(y - y-hat) * \sigma'(z_2) * \frac{\partial z_2}{\partial w^{(2)}} \end{aligned}$$

$$\frac{\partial E}{\partial w^{(1)}} = \frac{\partial E}{\partial y} * \frac{\partial y}{\partial z_2} * \frac{\partial z_2}{\partial o_1} * \frac{\partial o_1}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial y}{\partial z_2} = \sigma'(z_2)$$

$$\frac{\partial z_2}{\partial o_1} = 0_1$$

$$\frac{\partial o_1}{\partial z_1} = w_2$$

$$\frac{\partial z_1}{\partial w_1} = \sigma'(z_1)$$

$$\frac{\partial z_1}{\partial w_1} = x$$

$$z_1 = (x, w_1)$$

$$x_{5 \times 1} \Rightarrow x_{5 \times 2} \quad \left. \begin{array}{l} x_{5 \times 2} \\ w_{2 \times 5} \end{array} \right\} \cdot \left. \begin{array}{l} w_{2 \times 5} \\ x_{2 \times 5} \end{array} \right\}^T = -w_{2 \times 5}^T \cdot x_{2 \times 5}^T \leq z_{5 \times 5}^{(1)}$$

$$o_{5 \times 5}^{(1)} \Rightarrow o_{5 \times 6}^{(1)} \cdot w_{6 \times 1}^{(2)} \Rightarrow z_{5 \times 1}^{(2)}$$

$$(y - o_2)_{5 \times 1} \cdot o_{5 \times 1}^{(2)} = \text{O/P gradient}$$

$$\Delta out_{5 \times 1} \cdot w_{5 \times 1}^{(2)} = \boxed{5 \times 1} \cdot \boxed{5 \times 5} = \boxed{5 \times 5} \cdot \boxed{5 \times 1}$$

$$\boxed{1 \times 5} \quad \boxed{1 \times 5} \cdot \boxed{5 \times 5} = \boxed{1 \times 5} \cdot \boxed{5 \times 1}$$

$$\Delta out_{1 \times 5} \cdot w_{5 \times 1}^{(2)} = \boxed{1 \times 1} \cdot \boxed{5 \times 5} = \boxed{1 \times 5} \cdot \boxed{5 \times 1}$$

$$\boxed{5 \times 5} \quad \boxed{5 \times 5}$$

15/10/25

Soft Computing

Radial Basis Function Network

② Covers theorem: A complex pattern classification problem in a high dimensional space ~~dimensionality~~, non-linearity is more likely to be linearly separable than in a low dimensional space.

$$\vec{x} = [x_1, x_2, \dots, x_N]$$

$\phi(\vec{x}) \rightarrow$  hidden fn.

$$\begin{aligned}\Phi(\vec{x}) &= [\phi_1(\vec{x}), \phi_2(\vec{x}), \dots, \phi_m(\vec{x})]^T \\ &= \{\phi_i(\vec{x})\}_{i=1}^m \rightarrow \text{feature space.}\end{aligned}$$

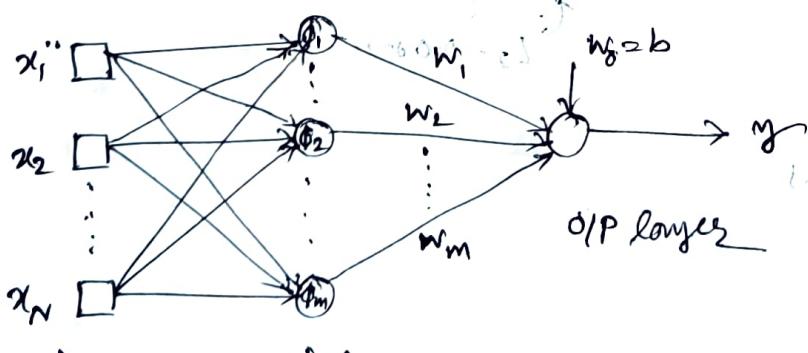
$$w^T \phi(\vec{x}) \geq 0 ; \vec{x} \in C_1$$

$$w^T \phi(\vec{x}) \leq 0 ; \vec{x} \in C_2$$

Separation hyperplane,

$$w^T \phi(\vec{x}) = 0$$

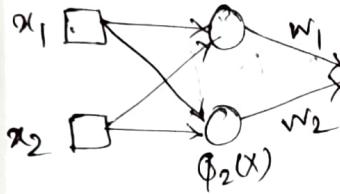
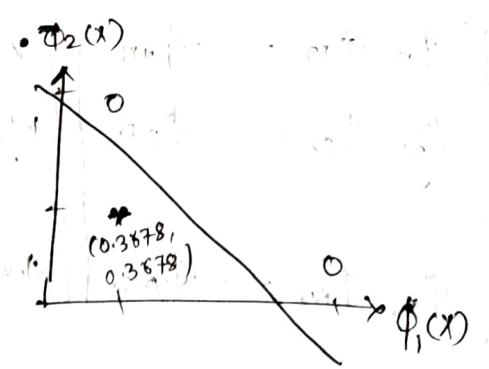
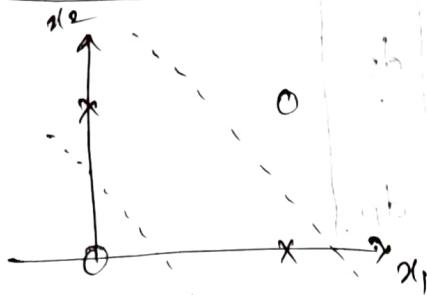
↳ separating surface



i/p layer  
hidden layer of  
 $m$  radial basis fn.

$$f(\vec{x}) = \sum_{i=1}^N w_i \phi_i(\|\vec{x} - \vec{x}_i\|)$$

### XOR problem:



$x_1, x_2$	$y$	$\phi_1(x)$	$\phi_2(x)$
(1, 1)	0	0.1353	0.1353
(0, 1)	1	0.3678	0.3678
(0, 0)	0	0.1353	0.1353
(1, 0)	1	0.3678	0.3678

### Gaussian fn.

$$\phi_1(x) = e^{-\|x - t_1\|^2}, \quad t_1 = [1, 1]^T \quad \text{center}$$

$$\phi_2(x) = e^{-\|x - t_2\|^2}, \quad t_2 = [0, 0]^T \quad \text{center}$$

### Interpolation problem:

$S: \mathbb{R}^{m_0} \rightarrow \mathbb{R}^1$   
↑  
mapping

Multivariable nonlinear mapping

Multivariable interpolation

a set of  $N$  different points

$$\{x_i \in \mathbb{R}^{m_0}\}_{i=1,2,\dots,N}$$

$$\{d_i \in \mathbb{R}^1\}_{i=1,2,\dots,N}$$

$$F(x_i) = d_i, \quad i=1,2,\dots,N$$

$$F(x) = \sum_{i=1}^N w_i \phi(\|x - x_i\|)$$

$$\{\phi(\|x - x_i\|)\}_{i=1,2,\dots,N}$$

radial basis function

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} \\ \vdots & & & \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

Where  $\phi_{ji} = \phi((\|\mathbf{x}_j - \mathbf{x}_i\|))$ ,  $(j, i) = 1, 2, \dots, n$

$$\vec{\Phi} = \{ \phi_{(j, i), d_j} \mid (j, i) = 1, 2, \dots, n \}$$

$$\vec{\Phi} \vec{w} = \vec{d}$$

$$\boxed{\vec{w} = \vec{\Phi}^{-1} \vec{d}}$$

### Micchelli's theorem:

i) Multi-quadratics:

$$\phi(r) = \sqrt{(r^2 + c^2)} \text{ for some } c > 0 \text{ and } r \in \mathbb{R}$$

ii) Inverse multi-quadratics:

$$\phi(r) = \frac{1}{\sqrt{r^2 + c^2}} \text{ for some } c > 0 \text{ and } r \in \mathbb{R}$$

iii) Gaussian form:

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \text{ for some } \sigma > 0 \text{ and } r \in \mathbb{R}$$