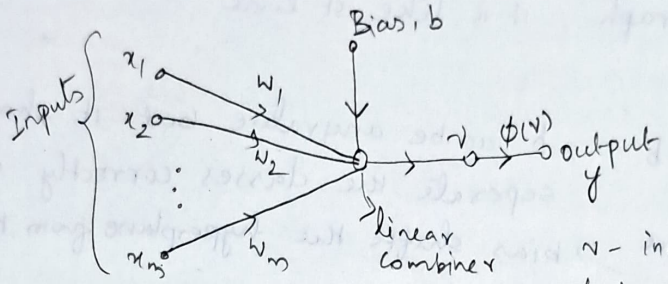
UNIT-2

Single Layer Perceptron:

→ Rosenblatt's perceptron | perceptron
→ built around a non-linear neuron, called as McCulloch-Pitts model of a neuron.
→ such a neuron model consists of a linear combiner followed by a hard limiter (performing the signum function).
→ Signal flow graph of the perceptron



v - induced local field
$\phi(v)$ - hard limit function
$w_1, w_2 \cdots w_m$ - synaptic weights
$x_1, x_2 \cdots x_m$ - inputs

$$\phi(v) = \begin{cases} 1, & v \geq 0 \\ -1, & v < 0 \end{cases}, \quad \phi(v) = \begin{cases} 1, & v > 0 \\ 0, & v = 0 \\ -1, & v < 0 \end{cases}$$ signum

$$v = \sum_{i=1}^{m} w_i x_i + b$$

Goal of Perceptron: To correctly classify the set of externally applied stimuli $x_1, x_2, \cdots, x_m$ into one of two classes $\mathcal{C}_1$ or $\mathcal{C}_2$.
Decision rule: for the classification is

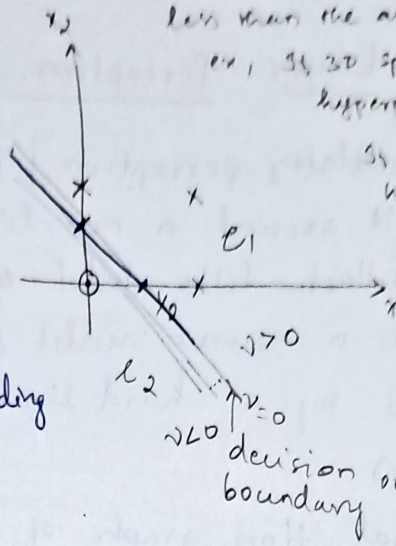
If output of perceptron $y = +1$, $x_1, x_2 \cdots x_m \in \mathcal{C}_1$
" " $y = -1$, " $\in \mathcal{C}_2$

To understand this better, we need to plot the input variables $x_1, x_2, \cdots, x_m$ on m-dimensional space, then there are two decision regions seperated by a hyperplane defined by $\sum_{i=1}^{m} w_i x_i + b = 0$

For example, consider OR Gate

hyperplane: is a subspace whose dimensions are one less than the ambient space
ex, 3D space i.e room
hyperplane - 2D
2D space, then hyperplane is a line

x_1	x_2	v	y	
0	0	$-1/2$	0	$\rightarrow c_2$
0	1	$1/2$	1	$\rightarrow c_1$
1	0	$1/2$	1	
1	1	$3/2$	1	



what is the equation corresponding to the hyperplane.

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

but from the graph, it is like st. line

$$y = mx + c$$

$$x_2 = -x_1 + b$$, b can be any value but it should seperate the classes correctly i-e

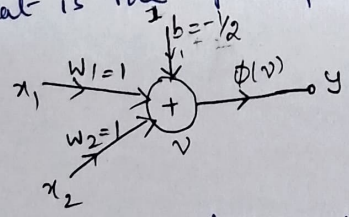
→ what should be the value of b?

→ bias shifts the hyperplane from the origin

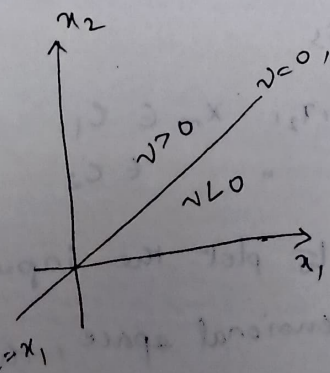
but I assume $b = 1/2$ (it is a guess from the graph)

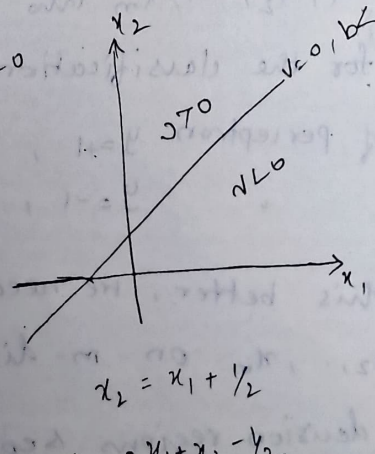
$\therefore x_2 = -x_1 + 1/2 \Rightarrow v = x_1 + x_2 - 1/2$ (using this fill the table)

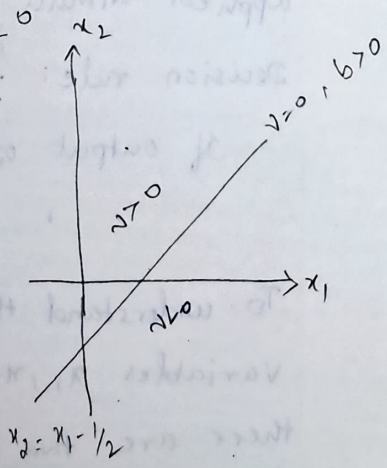
So, what is the perceptron for OR Gate?



what are the different bias values & corresponding decision boundary?

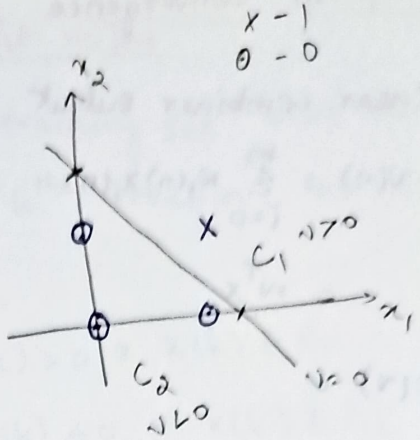






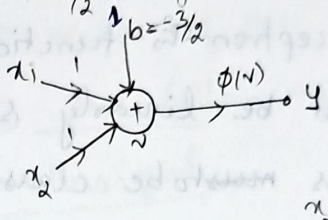
AND Gate:

x_1	x_2	v	y	
0	0	$-3/2$	0	$\in C_2$
0	1	$-1/2$	0	
1	0	$-1/2$	0	
1	1	$1/2$	1	$\rightarrow \in C_1$



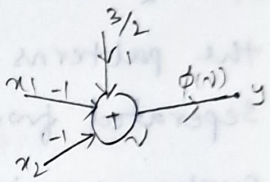
eq. of line $x_2 = -x_1 + 3/2$, 3/2 is a guess

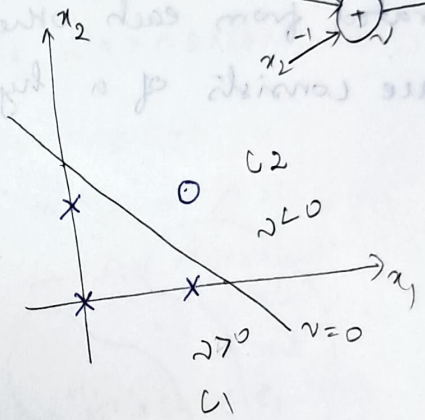
$v = +x_1 + x_2 - 3/2$



NAND Gate:

x_1	x_2	v	y	
0	0	$3/2$	1	$\in C_1$
0	1	$1/2$	1	
1	0	$1/2$	1	
1	1	$-1/2$	0	$\rightarrow C_2$





eq. of line $x_2 = -x_1 + 3/2$

$v = -x_2 + x_1 + 3/2$

This can be applied to any boolean function.

The synaptic weights $w_1, w_2, \ldots, w_m$ of the perceptron can be adapted on an iteration-by-iteration basis.

For the adaptation we may use an error-correction rule known as the perceptron convergence algorithm.

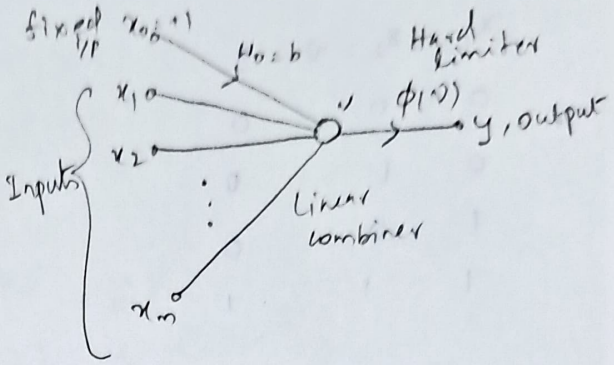
Perceptron Convergence Theorem:

Linear combiner output

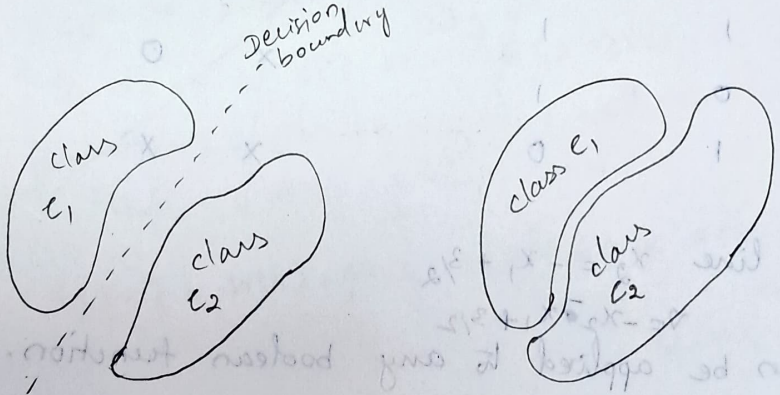
$$v(n) = \sum_{i=0}^{m} w_i(n) x_i(n)$$

$$= w^T x$$

$\phi(v) =$



For the perceptron to function properly, the two classes C_1 & C_2 must be linearly seperable. It means that the patterns ~~must~~ to be classified must be sufficiently seperated from each other to ensure that the decision surface consists of a hyperplane.



a) a pair of linearly seperable patterns

b) a pair of non-linearly seperable patterns.

Let C_1 & C_2 be 2 classes and let $H_1 \subset C_1$ & $H_2 \subset C_2$ be available. Given these training set, the training process involves the adjustment of weight vector w in such a way that 2 classes are linearly seperable.

i.e $\exists$ w s.t we may state

$w^T x > 0$ for every $x \in C_1$

$w^T x \geq 0$ for every $x \in C_2$ (our choice)

} Given the H_1 & H_2, training problem is to find w s.t these inequalities are satisfied

Algorithm for adapting the weight vector:

① If the k^{th} member of training set $x(k)$ is correctly classified at the k^{th} iteration, no correction is required for w.

$$w(k+1) = w(k) \quad \text{if } w^T x(k) > 0 \text{ \& } x(k) \in C_1$$

$$w(k+1) = w(k) \quad \text{if } w^T x(k) \leq 0 \text{ \& } x(k) \in C_2$$

② Otherwise, the weight vector of the perceptron is updated according to the following rule.
i.e If $x(k)$ is incorrectly classified.

$$w(k+1) = w(k) - \eta(k) x(k) \quad \text{if } w^T x > 0 \text{ \& } x(k) \in C_2$$

y can be either 1 or 0 & $e(k) = y_d(k) - y(k)$

If $w^T x > 0 \Rightarrow y = 1$ & $x(k) \in C_2 \Rightarrow y_d = 0 \Rightarrow e(k) = 0 - 1 = -1$

WKT, $w(k+1) = w(k) + \eta\, e(k)\, x(k)$ → LMS algorithm

$= w(k) - \eta\, x(k)$

$$w(k+1) = w(k) + \eta x(k) \quad \text{if } \underbrace{w^T x \leq 0}_{y = -1/0} \text{ \& } \underbrace{x(k) \in e_1}_{y_d = 1}$$

$e(k) = y_d - y = 1 - 0 = 1$

where $\eta(n)$ - learning rate parameter (can be fixed or adaptive)

Proof: If there is a weight vector w^* s.t $\phi(w^* x(k)) = y(k)$ $\forall k$ then for any starting vector w, the perceptron learning rule will converge to a weight-vector that gives the correct response for all training patterns & it will so in finite no. of steps.

Assumptions: for ease of analysis

i) Inputs to the perceptron originates from a linearly seperable classes

ii) $w(1) = 0$ - initial weight

iii) $\eta = 1$

Now consider that perceptron incorrectly classified

$$W^T x < 0 \quad \& \quad x(k) \in C_1$$

$$\therefore \quad W(k+1) = W(k) + x(k)$$

$k=1$, $W(2) = W(1) + x(1)$ (1 chosen)

$$W(2) = x(1)$$

$k=2$, $W(3) = W(2) + x(2)$

$$= x(1) + x(2)$$

$k=3$, $W(4) = W(3) + x(3)$

$$= x(1) + x(2) + x(3)$$

In general,

$$W(k+1) = x(1) + x(2) + x(3) + \cdots + x(k) \quad \rightarrow ① \rightarrow W^T x < 0 \text{ , } x(k) \in C_1$$

$$W(k+1) = -x(1) - x(2) - x(3) - \cdots - x(k) \rightarrow W^T x > 0 \text{ , } x(k) \in C_2 \text{ \& correctly classified terms}$$

Since the classes C_1 & C_2 are assumed to be linearly seperable, $\exists$ a solution W_0 for which $W^T x > 0$ that can classify all inputs correctly.

for a fixed solution W_0, we may define a +ve number α as

$$\alpha = \min_{x(n) \in H_1} W_0^T x(n)$$

multiply eq. ① by W_0^T

$$\Rightarrow W_0^T W(k+1) = W_0^T x(1) + W_0^T x(2) + W_0^T x(3) + \cdots + W_0^T x(k)$$

↳ each term is $> \alpha$, bcs α = min. value

$> \alpha \quad > \alpha \quad > \alpha \ldots$

$$W_0^T W(k+1) \geq k\alpha$$

squared norm of $W_0^T W(k+1) \geq k\alpha$ by using cauchy-schwartz inequality

$$\|W_0\|^2 \|W(k+1)\|^2 \geq \left(W_0^T W(k+1)\right)^2 \quad \text{— states that}$$

$$\geq (k^2\alpha^2) \qquad k = 1, 2 \ldots n$$

$$\|W(k+1)\|^2 \geq \frac{k^2\alpha^2}{\|W_0\|^2} \rightarrow \text{lower bound on } w \rightarrow ③$$

To find the upper bound, consider

$$w(k+1) = w(k) + x(k) \quad \text{②} \quad \text{for } k = 1, \cdots, n \text{ \& } x(k) \in H_1$$

Obtain the squared Euclidean norm on both sides

NORM:

A map $\|\cdot\| : V \to \mathbb{R}$, where V is a vector space over $\mathbb{R}$ s.t

a) $\|x\| \geq 0 \quad \forall x \in V \; ; \; \|x\| = 0$

b) $\|\alpha x\| = |\alpha| \|x\| \quad \forall x \in V \text{ \& } \alpha \in \mathbb{R}$

c) $\|x+y\| \leq \|x\| + \|y\| \quad \forall x, y \in V$

is called a norm on V

---

② $\Rightarrow \|w(k+1)\|^2 = \|w(k)\|^2 + \|x(k)\|^2 + 2 w^T(k) x(k)$

or

$$\|w(k+1)\|^2 = (w(k) + x(k))^T (w(k) + x(k))$$

$$= w^T(k) w(k) + w^T(k) x(k) + x^T(k) w(k) + x^T(k) x(k)$$

$$= w^T(k) w(k) + 2 w^T(k) x(k) + x^T(k) x(k)$$

$2 w^T(k) x(k) < 0$ misclassification

$$= \|w(k)\|^2 + \|x(k)\|^2$$

$$\|w(k+1)\|^2 \leq \|w(k)\|^2 + \|x(k)\|^2$$

$$\|w(k+1)\|^2 - \|w(k)\|^2 \leq \|x(k)\|^2$$

$$\|w(k+1)\|^2 \leq \sum_{k=1}^{n} \|x(k)\|^2 \quad , \quad \because \|w(0)\|^2 = 0$$

$k = 0, 1, 2 \cdots$

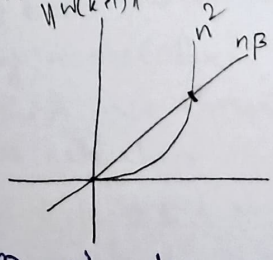
$$\leq n\beta$$

where $\beta = \max_{x(k) \in H_1} \|x(k)\|^2 \quad ---$

$$\|w(k+1)\|^2 \leq n\beta \quad \cdots \text{④} \quad \text{---> upper limit}$$

$$\frac{k^2 \alpha^2}{\|w_0\|^2} \leq \|w(k+1)\|^2 \leq n\beta$$

parabola, st line



It is impossible to satisfy ③ & ④ simultaneously

but at some point those 2 coincides beyond that w(k) will not change at all.

$\exists n = n_{max}$ s.t

$$\frac{n_{max}^2 \alpha^2}{\|w_0\|^2} = n_{max} \beta$$

$$\Rightarrow \quad n_{max} = \frac{\|w_0\|^2 \beta}{\alpha^2}$$

We have thus proved that for $\eta = 1$, $w(0) = 0$, & given that a solution vector w_0 exists, the rule for adapting the synaptic weights of the perceptron must terminate after at most n_{max} iterations.

Multilayer Perceptrons (chapter 4 in Simon Haykin)

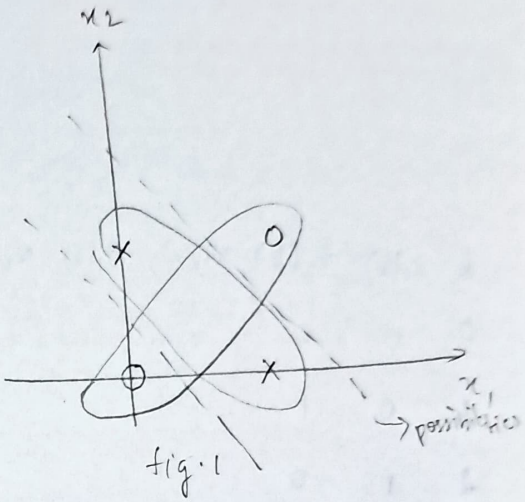
X-OR Problem

→ A single layer perceptron cannot classify input patterns that are not linearly seperable.

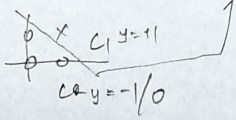
→ However nonlinearly seperable patterns are of common occurance.

→ This problem arises in X-OR problem.

k	0	1	2	3
$x_1(k)$	0	0	1	1
$x_2(k)$	0	1	0	1
$d(k)$	0	1	1	0
	C_2	C_1	C_1	C_2

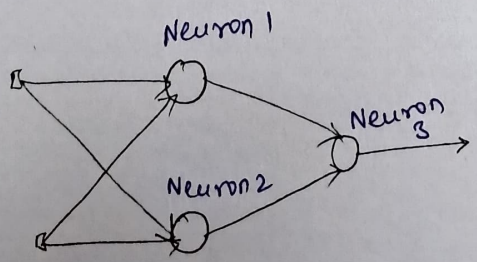


→ Generally, single neuron with 2 inputs results in a straight line for a decision boundary in the input space.



→ Fig·1 clearly shows that we cannot construct a stright line for a decision boundary, which implies with single perceptron we cannot solve this problem.

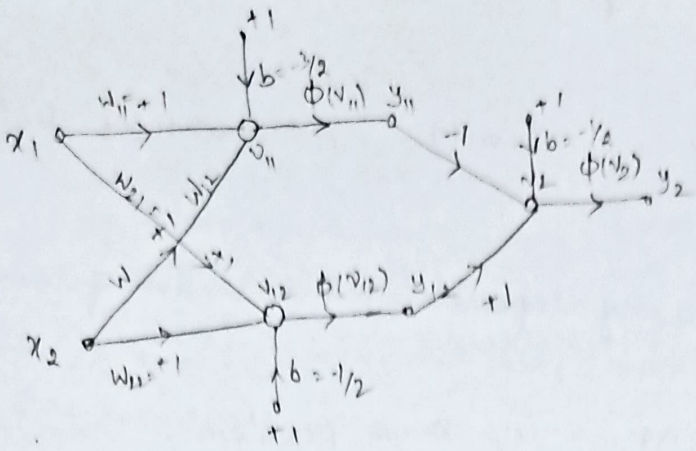
→ We may solve XOR problem by using a single hidden layer with 2 neurons. as shown in SFG.

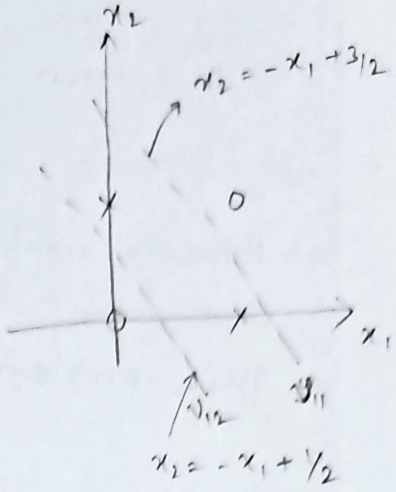


Assumptions:

i) Each neuron is represented by a McCulloch-Pitts model, which uses threshold function as activation

ii) Bits 0 & 1 are represented by the levels 0 & +1 respectively.



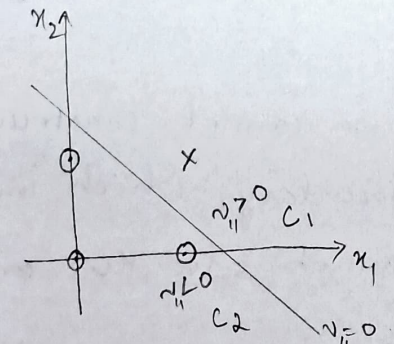


$v_{11} = x_1 + x_2 - 3/2$

$v_{12} = x_1 + x_2 - 1/2$

				AND		OR	
k	$x_1(k)$	$x_2(k)$	$v_{11}(k)$	$y_1(k)$	$v_{12}(k)$	$y_{12}(k)$	$v_2(k)$	$y_2(k)$
0	0	0	$-3/2$	0	$-1/2$	0	$-1/2$	0
1	0	1	$-1/2$	0	$1/2$	1	$1/2$	1
2	1	0	$-1/2$	0	$1/2$	1	$1/2$	1
3	1	1	$1/2$	1	$3/2$	1	$-1/2$	0

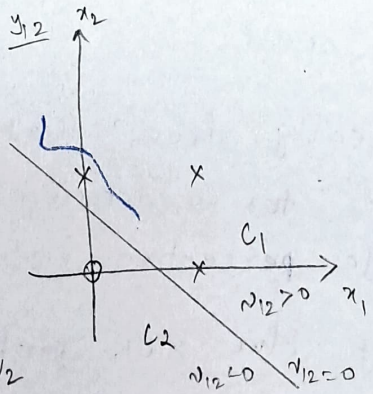
y_{11}.



$x_2 = -x_1 + 3/2$

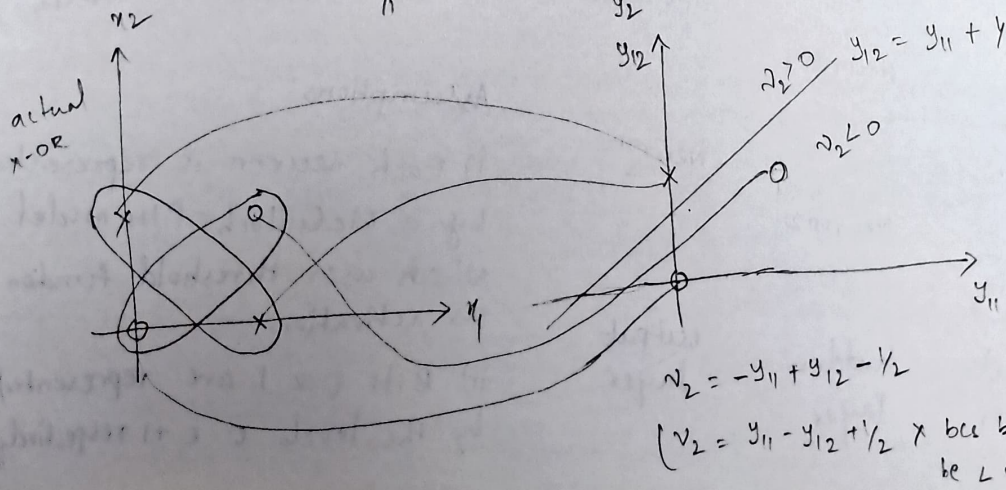
$v_{11} = x_2 + x_1 - 3/2$

y_{12}



$x_2 = -x_1 + 1/2$

$v_{12} = x_2 + x_1 - 1/2$



$v_2 = -y_{11} + y_{12} - 1/2$

($v_2 = y_{11} - y_{12} + 1/2$ ✗ bcz bias must be < 0)

① AND gate:

x_1	x_2	y_d	
0	0	0	ℓ_2
0	1	0	
1	0	0	
1	1	1	ℓ_1

$x(1) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $x(2) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $x(3) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $x(4) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

let

$w(1) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. $\eta = 1$

$x(1) \in \ell_1$

$w^T(1)x(1) = 0 \geq 0 \quad \therefore y(1) = 1 \Rightarrow$ which is incorrectly classified $\because y_d(1) = 0$.

update w as follows.

$$w(2) = w(1) - x(1) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

I $w^T(2)x(2) = 0 \geq 0 \quad y(2) = 1 \Rightarrow x(2) \in \ell_1$ which is incorrectly classified $\because y_d = 0$

$$w(3) = w(2) - x(2) = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

II $w^T(3)x(3) = \begin{pmatrix} 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0 \geq 0 \quad \therefore y(3) = 1$

$\Rightarrow x(3) \in \ell_1$

$w(4) = w(3) - x(3)$ which is incorrectly classified

$= \begin{pmatrix} 0 \\ -1 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$

V $w^T(4)x(4) = (-1 \quad -1) \begin{pmatrix} 1 \\ 1 \end{pmatrix} = -2 < 0 \quad \therefore y(4) = 0$

$\Rightarrow x(4) \in \ell_2$

which is incorrect

$$w(5) = w(4) + x(4)$$

$$= \begin{pmatrix} -1 \\ -1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

again same result will occur.
Hence start with different $w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

I $w^T(1)\, x(1) = (1 \;\; 1) \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0$ $y(1) = 1$ $x(1) \in l_2$
hence incorrectly classified.

$$w(2) = w(1) - x(1)$$

$$= \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 01 \\ 01 \end{pmatrix}$$

II $w^T(2)\, x(2) = (1 \;\; 1) \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1 \geq 0$ $y(2) = 1$ $x(2) \in l_1$
incorrectly classified.

$$w(3) = w(2) - x(2) \qquad = 0$$

$$= \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 01 \end{pmatrix} = \begin{pmatrix} 01 \\ 0 \end{pmatrix}$$

III $w^T(3)\, x(3) = (10 \;\; 0) \begin{pmatrix} 10 \\ 0 \end{pmatrix} = 1$ $x(3) \in l_1$
incorrect classification.

$$w(4) = w(3) - x(3)$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

IV $w^T(4)\, x(4) = (0 \;\; 0) \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0 \geq 0$ $x(4) \in l_1$
correct classif^n.

$$w(5) = w(4) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

but again for other pattern it fails.

$w(1) = \begin{pmatrix} +\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$

I $w^T(1)\, x(1) = 0 \Rightarrow x(1) \in \ell_1$ incorrect

$w(2) = w(1) - x(1)$

$= \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} +\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$

II $w^T(2)\, x(2) = \begin{pmatrix} +\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}^T \begin{pmatrix} 0 \\ 1 \end{pmatrix} = -\frac{1}{2} < 0 \Rightarrow x(2) \in \ell_2$ correct.

$w(3) = w(2)$.

III $w^T(3)\, x(3) = \begin{pmatrix} +\frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = -\frac{1}{2} < 0 \Rightarrow x(3) \in \ell_2$ good.

$w(4) = w(3)$

IV $w^T(4)\, x(4) = \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = -\frac{1}{2} - \frac{1}{2} = -1 \Rightarrow$ $x(4) \in \ell_2$ incorrect.

$w(5) = w(4) + x(4)$

$= \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$

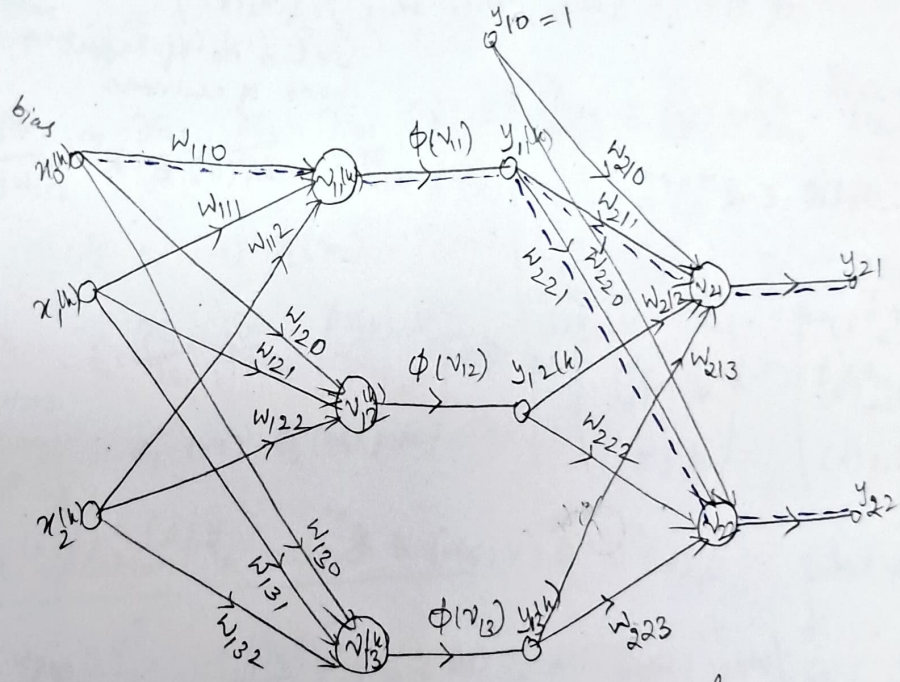
Back propagation Algorithm: is applied to multilayer ff network.

* uses continuous differentiable function as activation function
* based on error-correction learning rule

There are two passes

1. forward pass : during this information flows in forward direction & weights are fixed
2. Backward pass : error is back propagated & weights are updated.

$m_0 : m_1 : m_2$
$2 : 3 : 2$



layer : $l = 0$ | $l = 1$ | $l = 2$
index : i | i | k
i/p layer | hidden layer | o/p layer

w_{lij} =>
- l - layer number
- i - neuron number
- j - input connected to

Forward Pass

$$x(k) = \begin{pmatrix} x_0(k) \\ x_1(k) \\ x_2(k) \end{pmatrix} \qquad y_0(k) \triangleq \begin{pmatrix} 1 \\ x(k) \end{pmatrix} = \begin{pmatrix} 1 \\ x_1(k) \\ x_2(k) \end{pmatrix}$$

$$v_{11}(k) = w_{110} \times 1 + w_{111} x_1(k) + w_{112} x_2(k) = \sum_{j=0}^{2} w_{11j} x_j$$

$$v_{12}(k) = w_{120} + w_{121} x_1(k) + w_{122} x_2(k) = \sum_{j=0}^{2} w_{12j} x_j$$

$$v_{13}(k) = w_{130} + w_{131} x_1(k) + w_{132} x_2(k) = \sum_{j=0}^{2} w_{13j} x_j$$

$$v_1(k) = \begin{pmatrix} v_{11} \\ v_{12} \\ v_{13} \end{pmatrix} = \begin{pmatrix} w_{110} & w_{111} & w_{112} \\ w_{120} & w_{121} & w_{122} \\ w_{130} & w_{131} & w_{132} \end{pmatrix} \begin{pmatrix} 1 \\ x_1(k) \\ x_2(k) \end{pmatrix}, \quad x_0(k) = 1$$

3×3 → no. of input nodes + 1
↳ no. of neurons

In general, $v_1(k) = W_1(k)\, y_0(k)$

$y_0(k) \in \mathbb{R}^{m_0+1}$, $v_1(k) \in \mathbb{R}^{m_1}$, $W_1(k) \in \mathbb{R}^{m_1 \times m_0+1}$

$$\bar{y}_1 \triangleq \begin{pmatrix} y_{11}(k) \\ y_{12}(k) \\ y_{13}(k) \end{pmatrix} = \begin{pmatrix} \phi_1(v_{11}) \\ \phi_1(v_{12}) \\ \phi_1(v_{13}) \end{pmatrix} = \phi_1 \begin{pmatrix} v_{11}(k) \\ v_{12}(k) \\ v_{13}(k) \end{pmatrix} = \phi_1(v_1(k))$$

In general, $\bar{y}_1(k) = \phi_1(v_1(k)) \in \mathbb{R}^{m_1}$, $y_1(k) = \begin{pmatrix} 1 \\ \bar{y}_1(k) \end{pmatrix}$

$$\begin{pmatrix} v_{21}(k) \\ v_{22}(k) \end{pmatrix} = \begin{pmatrix} w_{210} & w_{211} & w_{212} & w_{213} \\ w_{220} & w_{221} & w_{222} & w_{223} \end{pmatrix} \begin{pmatrix} y_{10}(k) \\ y_{11}(k) \\ y_{12}(k) \\ y_{13}(k) \end{pmatrix} = W_2(k) \begin{pmatrix} 1 \\ \bar{y}_1(k) \end{pmatrix}$$

$$v_2(k) \underset{2\times1}{\triangleq} W_2(k) \underset{2\times4}{} y_1(k) \underset{4\times1}{}$$

In general,

$$y_2(k) = \begin{pmatrix} y_{21}(k) \\ y_{22}(k) \end{pmatrix} = \phi_2 \begin{pmatrix} v_{21}(k) \\ v_{22}(k) \end{pmatrix} = \phi_2(v_2(k))$$

Backward Pass

$$e_1(k) = y_{d_1}(k) - y_{21}(k)$$

$$e_2(k) = y_{d_2}(k) - y_{22}(k)$$

$$e_s(k) = y_{d_s}(k) - y_{2s}(k) \qquad 1 \le s \le 2$$

$$\xi(k) = \tfrac{1}{2} \sum_{s=1}^{2} e_s^2(k) = \tfrac{1}{2}\left(e_1^2(k) + e_2^2(k)\right)$$

$$\frac{\partial \xi(k)}{\partial w_{210}(k)} = \frac{\partial \xi}{\partial e_1(k)} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{210}}$$

$$= e_1(k) \cdot (-1) \cdot \phi_2'(v_{21}(k)) \cdot y_{10}(k)$$

$$\frac{\partial \xi}{\partial w_{211}} = \frac{\partial \xi}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{211}}$$

$$= e_1(k) (-1) \phi_2'(v_{21}) \, y_{11}(k)$$

$$\frac{\partial \xi}{\partial w_{212}} = \frac{\partial \xi}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{212}}$$

$$= e_1 (-1) \cdot \phi_2'(v_{21}) \cdot y_{12}(k)$$

$$\frac{\partial \xi}{\partial w_{213}} = \frac{\partial \xi}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{213}}$$

$$= e_1 (-1) \phi_2'(v_{21}) \cdot y_{13}(k)$$

$$\frac{\partial \xi}{\partial w_{220}} = \frac{\partial \xi}{\partial e_2} \cdot \frac{\partial e_2}{\partial y_{22}} \cdot \frac{\partial y_{22}}{\partial v_{22}} \cdot \frac{\partial v_{22}}{\partial w_{220}}$$

$$= e_2(k) \cdot (-1) \cdot \phi_2'(v_{22}) \cdot y_{10}(k)$$

$$\frac{\partial \xi}{\partial w_{221}} = \frac{\partial \xi}{\partial e_2} \cdot \frac{\partial e_2}{\partial y_{22}} \cdot \frac{\partial y_{22}}{\partial v_{22}} \cdot \frac{\partial v_{22}}{\partial w_{221}}$$

$$= e_2 (-1) \phi_2'(v_{22}) \cdot y_{11}$$

$$\frac{\partial \xi}{\partial w_{222}} = \frac{\partial \xi}{\partial e_2} \cdot \frac{\partial e_2}{\partial y_{22}} \cdot \frac{\partial y_{22}}{\partial v_{22}} \cdot \frac{\partial v_{22}}{\partial w_{222}}$$

$$= e_2 (-1) \phi_2'(v_{22}) \cdot y_{12}$$

$$\frac{\partial \xi}{\partial w_{223}} = \frac{\partial \xi}{\partial e_2} \cdot \frac{\partial e_2}{\partial y_{22}} \cdot \frac{\partial y_{22}}{\partial v_{22}} \cdot \frac{\partial v_{22}}{\partial w_{223}}$$

$$= e_2 (-1) \phi_2'(v_{22}) \cdot y_{13}$$

Weight updation: in output layer

$$\begin{pmatrix} \Delta w_{210}(k) \\ \Delta w_{211}(k) \\ \Delta w_{212}(k) \\ \Delta w_{213}(k) \end{pmatrix}^T = -\eta_2 \begin{pmatrix} \frac{\partial \xi}{\partial w_{210}} \\ \frac{\partial \xi}{\partial w_{211}} \\ \frac{\partial \xi}{\partial w_{212}} \\ \frac{\partial \xi}{\partial w_{213}} \end{pmatrix}^T = \eta_2 \underbrace{e_1(k) \phi_2'(v_{21}(k))}_{\delta_{21}} \begin{pmatrix} y_{10}(k) \\ y_{11}(k) \\ y_{12}(k) \\ y_{13}(k) \end{pmatrix}^T$$

local gradient

$$\delta_{21} = -\frac{\partial \xi}{\partial v_{21}} = -\frac{\partial \xi}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}}$$

$$= -e_1(-1) \cdot \phi_2'(v_{21})$$

$$= e_1(k) \phi_2'(v_{21}(k))$$

$$\begin{pmatrix} \Delta w_{220}(k) \\ \Delta w_{221}(k) \\ \Delta w_{222}(k) \\ \Delta w_{223}(k) \end{pmatrix}^T = -\eta_2 \begin{pmatrix} \frac{\partial \xi}{\partial w_{220}} \\ \frac{\partial \xi}{\partial w_{221}} \\ \frac{\partial \xi}{\partial w_{222}} \\ \frac{\partial \xi}{\partial w_{223}} \end{pmatrix}^T = \eta_2 \underbrace{e_2(k) \phi_2'(v_{22}(k))}_{\delta_{22}} \begin{pmatrix} y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \end{pmatrix}^T$$

$$\delta_{22} = -\frac{\partial \xi}{\partial v_{22}}$$

$$= -\frac{\partial \xi}{\partial e_2} \cdot \frac{\partial e_2}{\partial y_{22}} \cdot \frac{\partial y_{22}}{\partial v_{22}}$$

$$= -e_2(k)(-1) \phi_2'(v_{22}(k))$$

$$= e_2(k) \phi_2'(v_{22}(k))$$

$$\Delta W_2(k) = \begin{pmatrix} \Delta w_{210} & \Delta w_{211} & \Delta w_{212} & \Delta w_{213} \\ \Delta w_{220} & \Delta w_{221} & \Delta w_{222} & \Delta w_{223} \end{pmatrix}$$

$$= \begin{pmatrix} \eta_2 \delta_{21}(k) y_{10}(k) & \eta_2 \delta_{21}(k) y_{11}(k) & \eta_2 \delta_{21}(k) y_{12}(k) & \eta_2 \delta_{21}(k) y_{13}(k) \\ \eta_2 \delta_{22}(k) y_{10}(k) & \eta_2 \delta_{22}(k) y_{11}(k) & \eta_2 \delta_{22}(k) y_{12}(k) & \eta_2 \delta_{22}(k) y_{13}(k) \end{pmatrix}$$

$$= \eta_2 \begin{pmatrix} \delta_{21}(k) \\ \delta_{22}(k) \end{pmatrix}_{2\times 1} \begin{pmatrix} y_{10}(k) & y_{11}(k) & y_{12}(k) & y_{13}(k) \end{pmatrix}$$

but $\begin{pmatrix} \delta_{21} \\ \delta_{22} \end{pmatrix} = \begin{pmatrix} e_1(k)\, \phi_2'(v_{21}) \\ e_2(k)\, \phi_2'(v_{22}) \end{pmatrix}$

$$= \begin{pmatrix} e_1(k) \\ e_2(k) \end{pmatrix}_{2\times 1} \odot \begin{pmatrix} \phi_2'(v_{21}) \\ \phi_2'(v_{22}) \end{pmatrix}_{2\times 1}$$

↑ Hadamard product

$$\underset{2\times 4}{\Delta W_2(k)} = \eta_2 \underset{2\times 1}{\delta_2(k)} \underset{1\times 4}{y_1^T(k)}$$

$$W_2(k+1) = W_2(k) + \Delta W_2(k)$$

Weight updation in 1st layer

$$\frac{\partial \xi}{\partial w_{110}} = \frac{\partial \xi}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial y_{11}} \cdot \frac{\partial y_{11}}{\partial v_{11}} \cdot \frac{\partial v_{11}}{\partial w_{110}} + \frac{\partial \xi}{\partial e_2} \cdot \frac{\partial e_2}{y_{22}} \cdot \frac{\partial y_{22}}{\partial v_{22}} \cdot \frac{\partial v_{22}}{y_{11}} \cdot \frac{\partial y_{11}}{\partial v_{11}} \cdot \frac{\partial v_{11}}{\partial w_{110}}$$

$$= e_1 (-1)\, \phi_2'(v_{21})\, w_{211}\, \phi_1'(v_{11})\, x_0 + e_2 (-1) \cdot \phi_2'(v_{22}) \cdot w_{221}\, \phi_1'(v_{11}) \cdot x_0$$

$$= -\underbrace{e_1(k)\, \phi_2'(v_{21})}_{\delta_{21}}\, w_{211}\, \phi_1'(v_{11}) x_0 - \underbrace{e_2\, \phi_2'(v_{22})}_{\delta_{22}}\, w_{221}\, \phi_1'(v_{11})\, x_0$$

$$= -\delta_{21}\, w_{211}\, \phi_1'(v_{11}) x_0 - \delta_{22}\, w_{221}\, \phi_1'(v_{11})\, x_0$$

$$\left.\begin{aligned} \Delta w_{110} &= -\eta_1 \frac{\partial \xi}{\partial w_{110}} = \eta_1 \delta_{21} w_{211} \phi_1'(v_{11}) x_0 + \eta_1 \delta_{22} w_{221} \phi_1'(v_{11}) \cdot x_0 \\ \Delta w_{111} &= -\eta_1 \frac{\partial \xi}{\partial w_{111}} = \eta_1 \delta_{21} w_{211} \phi_1'(v_{11}) x_1 + \eta_1 \delta_{22} w_{221} \phi_1'(v_{11}) \cdot x_1 \\ \Delta w_{112} &= -\eta_1 \frac{\partial \xi}{\partial w_{112}} = \eta_1 \delta_{21} w_{211} \phi_1(v_{11}) x_2 + \eta_1 \delta_{22} w_{221} \phi_1'(v_{11}) x_2 \end{aligned}\right\} =$$

$$\Delta W_1 = \begin{pmatrix} \Delta w_{110} & \Delta w_{111} & \Delta w_{112} \\ \Delta w_{120} & \Delta w_{121} & \Delta w_{122} \\ \Delta w_{130} & \Delta w_{131} & \Delta w_{132} \end{pmatrix}_{3\times 3}$$

$$\Delta W_{120} = -\eta_1 \frac{\partial \xi}{\partial W_{120}} \qquad \eta_1$$

$$\frac{\partial \xi}{\partial W_{120}} = \frac{\partial \xi}{\partial e_1}\cdot\frac{\partial e_1}{\partial y_{21}}\cdot\frac{\partial y_{21}}{\partial v_{21}}\cdot\frac{\partial v_{21}}{\partial y_{12}}\cdot\frac{\partial y_{12}}{\partial v_{12}}\cdot\frac{\partial v_{12}}{\partial W_{120}} + \frac{\partial \xi}{\partial e_2}\cdot\frac{\partial e_2}{\partial y_{22}}\cdot\frac{\partial y_{22}}{\partial v_{22}}\cdot\frac{\partial v_{22}}{\partial y_{12}}\cdot\frac{\partial y_{12}}{\partial v_{12}}\cdot\frac{\partial v_{12}}{\partial W_{120}}$$

$$= e_1(-1)\,\phi_2'(v_{21})\,W_{212}\,\phi_1'(v_{12})\,x_0 + e_2(-1)\,\phi_2'(v_{22})\,W_{222}\,\phi_1'(v_{12})\,x_0$$

$$= -e_1\,\phi_2'(v_{21})\,W_{212}\,\phi_1'(v_{12})\,x_0 - e_2\,\phi_2'(v_{22})\,W_{222}\,\phi_1'(v_{12})\,x_0$$

$$\Delta W_{120} = \eta_1 e_1\,\phi_2'(v_{21})\,W_{212}\,\phi_1'(v_{12})\,x_0 + \eta_1 e_2\,\phi_2'(v_{22})\,W_{222}\,\phi_1'(v_{12})\,x_0$$

$$\Delta W_{121} = \eta_1 e_1\,\phi_2'(v_{21})\,W_{212}\,\phi_1'(v_{12})\,x_1 + \eta_1 e_2\,\phi_2'(v_{22})\,W_{222}\,\phi_1'(v_{12})\,x_1$$

$$\Delta W_{122} = \eta_1 \underbrace{e_1\,\phi_2'(v_{21})}_{\delta_{21}}\,W_{212}\,\phi_1'(v_{12})\,x_2 + \eta_1 \underbrace{e_2\,\phi_2'(v_{22})}_{\delta_{22}}\,W_{222}\,\phi_1'(v_{12})\,x_2$$

$$\frac{\partial \xi}{\partial W_{130}} = \frac{\partial \xi}{\partial e_1}\cdot\frac{\partial e_1}{\partial y_{21}}\cdot\frac{\partial y_{21}}{\partial v_{21}}\cdot\frac{\partial v_{21}}{\partial y_{13}}\cdot\frac{\partial y_{13}}{\partial v_{13}}\cdot\frac{\partial v_{13}}{\partial W_{130}} + \frac{\partial \xi}{\partial e_2}\cdot\frac{\partial e_2}{\partial y_{22}}\cdot\frac{\partial y_{22}}{\partial v_{22}}\cdot\frac{\partial v_{22}}{\partial y_{13}}\cdot\frac{\partial y_{13}}{\partial v_{13}}\cdot\frac{\partial v_{13}}{\partial W_{130}}$$

$$= e_1(-1)\,\phi_2'(v_{21})\,W_{213}\,\phi_1'(v_{13})\,x_0 + e_2(-1)\,\phi_2'(v_{22})\,W_{223}\,\phi_1'(v_{13})\,x_0$$

$$= -e_1\,\phi_2'(v_{21})\,W_{213}\,\phi_1'(v_{13})\,x_0 - e_2\,\phi_2'(v_{22})\,W_{223}\,\phi_1'(v_{13})\,x_0$$

$$\Delta W_{130} = -\eta_1 \frac{\partial \xi}{\partial W_{130}} = \eta_1 \underbrace{e_1\,\phi_2'(v_{21})}_{\delta_{21}}\,W_{213}\,\phi_1'(v_{13})\,x_0 + \eta_1 \underbrace{e_2\,\phi_2'(v_{22})}_{\delta_{22}}\,W_{223}\,\phi_1'(v_{13})\,x_0$$

$$= \eta_1\,\delta_{21}\,W_{213}\,\phi_1'(v_{13})\,x_0 + \eta_1\,\delta_{22}\,W_{223}\,\phi_1'(v_{13})\,x_0$$

$$\Delta W_1 = \begin{pmatrix} \eta_1\delta_{21}W_{211}\phi_1'(v_{11})x_0 + \eta_1\delta_{22}W_{221}\phi_1'(v_{11})x_0 & (\quad)x_1 & (\quad)x_2 \\ \left(\eta_1\delta_{21}W_{212}\phi_1'(v_{12}) + \eta_1\delta_{22}W_{222}\phi_1'(v_{12})\right)x_0 & (\quad)x_1 & (\quad)x_2 \\ \left(\eta_1\delta_{21}W_{213}\phi_1'(v_{13}) + \eta_1\delta_{22}W_{223}\phi_1'(v_{13})\right)x_0 & (\quad)x_1 & (\quad)x_2 \end{pmatrix}$$

$$= \eta_1 \left[\begin{pmatrix} \delta_{21}W_{211} + \delta_{22}W_{221} \\ \delta_{21}W_{212} + \delta_{22}W_{222} \\ \delta_{21}W_{213} + \delta_{22}W_{223} \end{pmatrix}_{3\times 1} \odot \begin{pmatrix} \phi_1'(v_{11}) \\ \phi_1'(v_{12}) \\ \phi_1'(v_{13}) \end{pmatrix}_{3\times 1} \right] \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}_{3\times 1}$$

Hadamard product ⊙ or $\dim(\phi_1'(\,)) \to$ matlab $3\times$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix}$$

$$= \eta_1 \left(\underbrace{\left(\begin{pmatrix} W_{211} & W_{221} \\ W_{212} & W_{222} \\ W_{213} & W_{223} \end{pmatrix}_{3\times 2} \begin{pmatrix} \delta_{21} \\ \delta_{22} \end{pmatrix}_{2\times 1} \right)}_{\delta_1(k)} \odot \begin{pmatrix} \phi_1'(v_{11}) \\ \phi_1'(v_{12}) \\ \phi_1'(v_{13}) \end{pmatrix} \right) \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

W_2^T ; $\dim(\phi_1'(\,))_{3\times 1}$; 3×1

$$\delta_1(k) = \bar{W}_1^T(k)\,\delta_2(k) \odot \phi_1'(v_1(k))$$

($\bar{W}_1^T$: 3×2, δ_2: 2×1; $\mathrm{diag}(\phi_1'(\cdot))$: 3×3)

$$\Delta W_1 = \eta_1\,\delta_1(k)\,y_0^T(k) \quad , \quad y_0 = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

(ΔW_1: 3×3; δ_1: 3×1, y_0^T: 1×3)

$$W_1(k+1) = W_1(k) + \Delta W_1(k)$$

Summary of Back propagation Algorithm:

consider $(L+1)$ layered FFNN with m_0 input nodes and $m_1, m_2, \dots, m_L$ neurons in the subsequent layers. Let $\phi_1, \phi_2, \dots, \phi_L$ be the activation functions of these layers where these functions are continuously differentiable. Let $\{(x(k), y_d(k))\}_{k=1}^{N}$ be a given set of training patterns. Then for each $k = 1, \dots, N$.

a) <u>forward pass:</u>

$$v_\ell(k) = W_\ell(k)\, y_{\ell-1}(k) \quad , \quad \bar{y}_\ell(k) = \phi_\ell(v_\ell(k)) \, , \quad 1 \le \ell \le L$$

where $y_0(k) = \begin{pmatrix} 1 \\ x(k) \end{pmatrix}$, $y_\ell(k) = \begin{pmatrix} 1 \\ \bar{y}_\ell(k) \end{pmatrix}$, $1 \le \ell \le L-1$

$x(k) \in \mathbb{R}^{m_0}$, $\bar{y}_\ell(k) \in \mathbb{R}^{m_\ell}$, $W_\ell(k) \in \mathbb{R}^{m_\ell \times (m_{\ell-1}+1)}$, $1 \le \ell \le L$

$$e(k) = d(k) - \bar{y}_L(k)$$

b) <u>Backward Pass</u>

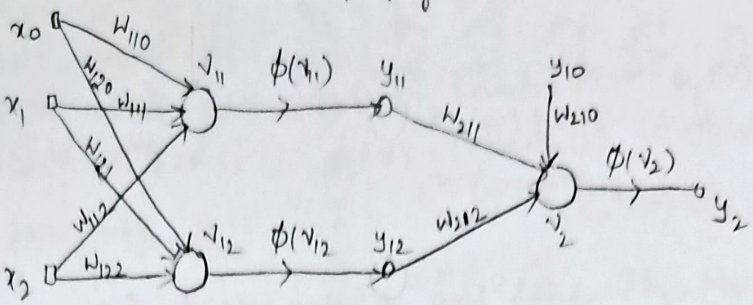
$$\delta_L(k) = e(k) \odot \phi_L'(v_L(k))$$

$$\delta_\ell(k) = \left(\bar{W}_{\ell+1}^T(k)\, \delta_{\ell+1}(k)\right) \odot \phi_\ell'(v_\ell(k)) \, , \quad \ell = L-1, L-2, \dots, 1$$

$$\Delta W_\ell(k) = \eta_\ell\, \delta_\ell(k)\, y_{\ell-1}^T(k) \quad , \quad 1 \le \ell \le L$$

2:2:1 → X-OR Gate using Φ - continuously differentiable

back propagation



forward pass:

$$v_1 = W_1 y_0 \quad (2\times1 = 2\times3 \; 3\times1)$$

$$= \begin{pmatrix} w_{110} & w_{111} & w_{112} \\ w_{120} & w_{121} & w_{122} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$v_{11} = w_{110}x_0 + w_{111}x_1 + w_{112}x_2$$

$$v_{12} = w_{120}x_0 + w_{121}x_1 + w_{122}x_2$$

$$\Downarrow$$

$$\begin{pmatrix} v_{11} \\ v_{12} \end{pmatrix} = \begin{pmatrix} w_{110} & w_{111} & w_{112} \\ w_{120} & w_{121} & w_{122} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$y_1 = \phi_1(v_1) = \begin{pmatrix} y_{11} \\ y_{12} \end{pmatrix} = \begin{pmatrix} \phi(v_{11}) \\ \phi(v_{12}) \end{pmatrix}$$

$$y_0 = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}, \quad \bar{y}_1 = \begin{pmatrix} 1 \\ y_1 \end{pmatrix}$$

$$v_2 = W_2 y_1 \quad (1\times3 \; 3\times1)$$

$$= \underbrace{\begin{pmatrix} w_{210} & w_{211} & w_{212} \end{pmatrix}}_{W_2} \begin{pmatrix} y_{10} \\ y_{11} \\ y_{12} \end{pmatrix} = W_2 \begin{pmatrix} 1 \\ y_1 \end{pmatrix}; \quad y_2 = \begin{pmatrix} 1 \\ y_2 \end{pmatrix}$$

$$y_2 = \phi_2(v_2), \quad e = y_d - y_2$$

Backward pass:

$$\delta_2 = e \odot \phi_2'(v_2(k))$$

$$\Delta W_2 = \eta_2 \delta_2 y_1^T$$

$$W_2(k+1) = W_2(k) + \Delta W_2(k)$$

$$\delta_1 = (\bar{W}_2^T \cdot \delta_2) \odot \phi_1'(v_1(k)) \quad (2\times1 \; 1\times1 \; 2\times1)$$

$$= \begin{pmatrix} w_{211} \\ w_{212} \end{pmatrix} \delta_2 \odot \phi_1' \begin{pmatrix} v_{11}(k) \\ v_{12}(k) \end{pmatrix}$$

$$\Delta W_1(k) = \eta_1 \delta_1(k) y_0^T(k) \quad (2\times3 = 2\times1 \; 1\times3)$$

$$= \begin{pmatrix} \Delta w_{110} & \Delta w_{111} & \Delta w_{112} \\ \Delta w_{120} & \Delta w_{121} & \Delta w_{122} \end{pmatrix}$$

$$W_1(k+1) = W_1(k) + \Delta W_1(k)$$

o/p layer

$$\xi_1(k) = \frac{1}{2} e^2(k)$$

$$\frac{\partial \xi}{\partial w_{210}} = \frac{\partial \xi}{\partial e} \cdot \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial v_2} \cdot \frac{\partial v_2}{\partial w_{210}}$$

$$= e(k)(-1)\phi_2'(v_2) \cdot y_{10}$$

$$\frac{\partial \xi}{\partial w_{211}} = \frac{\partial \xi}{\partial e} \cdot \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial v_2} \cdot \frac{\partial v_2}{\partial w_{211}}$$

$$= e(k) \cdot (-1) \phi_2'(v_2) \cdot y_{11}$$

$$\frac{\partial \xi}{\partial w_{212}} = \frac{\partial \xi}{\partial e} \cdot \frac{\partial e}{\partial y_2} \cdot \frac{\partial y_2}{\partial v_2} \cdot \frac{\partial v_2}{\partial w_{212}}$$

$$= e \cdot (-1) \cdot \phi_2'(v_2) \cdot y_{12}$$

$$\Delta W_2 = \begin{pmatrix} \Delta w_{210} \\ \Delta w_{211} \\ \Delta w_{212} \end{pmatrix} = -\eta_2 \begin{pmatrix} \frac{\partial \xi}{\partial w_{210}} \\ \frac{\partial \xi}{\partial w_{211}} \\ \frac{\partial \xi}{\partial w_{212}} \end{pmatrix}^T$$

$$= -\eta_2 e \cdot (-1) \cdot \phi_2'(v_2) \begin{pmatrix} y_{10} \\ y_{11} \\ y_{12} \end{pmatrix}^T$$

$$= \eta_2 \underbrace{e \phi_2'(v_2)}_{\delta_2} y_1^T$$

Weight updation 1st layer:

$$\frac{\partial \xi_i}{\partial w_{110}} = \frac{\partial \xi_i}{\partial e}\cdot\frac{\partial e}{\partial y_2}\cdot\frac{\partial y_2}{\partial v_2}\cdot\frac{\partial v_2}{\partial y_{11}}\cdot\frac{\partial y_{11}}{\partial v_{11}}\cdot\frac{\partial v_{11}}{\partial w_{110}}$$

$$= e(-1)\,\phi_2'(v_2)\,w_{211}\,\phi_1'(v_{11})\,x_0$$

$$\frac{\partial \xi_i}{\partial w_{111}} = e(-1)\,\phi_2'(v_2)\,w_{211}\,\phi_1'(v_{11})\,x_1$$

$$\frac{\partial \xi_i}{\partial w_{112}} = e(-1)\,\phi_2'(v_2)\,w_{211}\,\phi_1'(v_{11})\,x_2$$

$$\frac{\partial \xi_i}{\partial w_{120}} = \frac{\partial \xi_i}{\partial e}\cdot\frac{\partial e}{\partial y_2}\cdot\frac{\partial y_2}{\partial v_2}\cdot\frac{\partial v_2}{\partial y_{12}}\cdot\frac{\partial y_{12}}{\partial v_{12}}\cdot\frac{\partial v_{12}}{\partial w_{120}}$$

$$= e(-1)\,\phi_2'(v_2)\,w_{212}\,\phi_1'(v_{12})\,x_0$$

$$\frac{\partial \xi_i}{\partial w_{121}} = e(-1)\,\phi_2'(v_2)\,w_{212}\,\phi_1'(v_{12})\,x_1$$

$$\frac{\partial \xi_i}{\partial w_{122}} = e(-1)\,\phi_2'(v_2)\,w_{212}\,\phi_1'(v_{12})\,x_2$$

$$\Delta W_1 = \begin{pmatrix} \Delta w_{110} & \Delta w_{111} & \Delta w_{112} \\ \Delta w_{120} & \Delta w_{121} & \Delta w_{122} \end{pmatrix} = \begin{pmatrix} -\eta_1\frac{\partial \xi_i}{\partial w_{110}} & -\eta_1\frac{\partial \xi_i}{\partial w_{111}} & -\eta_1\frac{\partial \xi_i}{\partial w_{112}} \\ -\eta_1\frac{\partial \xi_i}{\partial w_{120}} & -\eta_1\frac{\partial \xi_i}{\partial w_{121}} & -\eta_1\frac{\partial \xi_i}{\partial w_{122}} \end{pmatrix}$$

$$= \begin{pmatrix} \eta_1 \underbrace{e\,\phi_2'(v_2)}_{\delta_2} w_{211}\,\phi_1'(v_{11})\,x_0 & \eta_1 \underbrace{e\,\phi_2'(v_2)}_{\delta_2} w_{211}\,\phi_1'(v_{11})\,x_1 & \eta_1 \underbrace{e\,\phi_2'(v_2)}_{\delta_2} w_{211}\,\phi_1'(v_{11})\,x_2 \\ \eta_1 \underbrace{e\,\phi_2'(v_2)}_{\delta_2} w_{212}\,\phi_1'(v_{12})\,x_0 & \eta_1 \underbrace{e\,\phi_2'(v_2)}_{\delta_2} w_{212}\,\phi_1'(v_{12})\,x_1 & \eta_1 \underbrace{e\,\phi_2'(v_2)}_{\delta_2} w_{212}\,\phi_1'(v_{12})\,x_2 \end{pmatrix}$$

$$= \eta_1 \begin{pmatrix} \delta_2 w_{211}\,\phi_1'(v_{11}) & \delta_2 w_{211}\,\phi_1'(v_{11}) & \delta_2 w_{211}\,\phi_1'(v_{11}) \\ \delta_2 w_{212}\,\phi_1'(v_{12}) & \delta_2 w_{212}\,\phi_1'(v_{12}) & \delta_2 w_{212}\,\phi_1'(v_{12}) \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$= \eta_1 \underbrace{\delta_2 \left[\begin{pmatrix} w_{211} \\ w_{212} \end{pmatrix}_{2\times1} \odot \begin{pmatrix} \phi_1'(v_{11}) \\ \phi_1'(v_{12}) \end{pmatrix} \right]_{2\times1}}_{\delta_1} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}_{3\times1}$$

$\odot \rightarrow$ Hadamard product

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \odot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix}$$

$$\delta_1 = \delta_2\,\bar{w}_2^T \odot \phi_1'(v_1)$$

$$\Delta W_1 = \eta_1 \underset{2\times1}{\delta_1}\; \underset{1\times3}{y_0^T}$$

(1). Consider 1:4:1 N/N.

$$W_1 = \begin{pmatrix} 0.2 & -0.4 \\ 0.25 & -0.4 \\ 0.4 & -0.1 \\ -0.1 & 0.4 \end{pmatrix} \qquad \eta_1 = \eta_2 = 0.01$$

$\phi_1(v) = \tanh(v)$ $\qquad \phi_2(v) = v$.

$\phi_1'(v) = 1 - \tanh^2(v)$ $\qquad \phi_2'(v) = 1$

or

$\text{sech}^2(v)$ or $y(1-y)$

$$W_2 = (-0.25 \quad 0.25 \quad 0.15 \quad -0.5 \quad 0.3)$$

x $\qquad$ y_d.

Data: $\{(0.5, 1)\}$ Use BPA and find the Next set of wts.

Sol^n:

$$v_1 = W_1 x \qquad \text{where} \quad x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$$

$$= \begin{pmatrix} 0.2 & -0.4 \\ 0.25 & -0.4 \\ 0.4 & -0.1 \\ -0.1 & 0.4 \end{pmatrix} \downarrow \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} = \begin{pmatrix} 0.2 - 0.2 \\ 0.25 - 0.2 \\ 0.4 - 0.05 \\ -0.1 + 0.2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.05 \\ 0.35 \\ 0.1 \end{pmatrix}$$

$4 \times 2 \quad 2 \times 1$

$$\bar{y}_1 = \phi_1(v_1) = \begin{pmatrix} \tanh(0) \\ \tanh(0.05) \\ \tanh(0.35) \\ \tanh(0.1) \end{pmatrix} = \begin{pmatrix} 0 \\ 0.0499 \\ 0.3363 \\ 0.0996 \end{pmatrix}$$

$$y_1 = \begin{pmatrix} 1 \\ 0.5 \\ 0.499 \\ 0.3363 \\ 0.0996 \end{pmatrix} \qquad 0.461$$

$$v_2 = W_2 y_1 = (-0.25 \quad 0.25 \quad 0.15 \quad -0.5 \quad 0.3) \begin{pmatrix} 1 \\ 0 \\ 0.499 \\ 0.3363 \\ 0.0996 \end{pmatrix}$$

$$= -0.380785$$

$y_2 = -0.380785$ → linear activation function

$$e = y_d - y_2 = 1.3808$$

$\delta_2 = e_1 \odot \phi_2'(U_2(k))$.

$= e \odot 1$

$= 1.3808$.

1:4:1

$\Delta W_2 = \eta_2 \delta_2 y_1^T = 0.01 \times 1.3808 \times (1 \quad 0 \quad 0.05 \quad 0.35 \quad 0.1)$.

$\Delta W_2 = [0.0138 \quad 0 \quad 0.00069 \quad 0.0048 \quad 0.00138]$.

UE17ECEP_PC[redacted]	Contribute and communicate effectively with the society confidently, be able to write effective reports and documents by adhering to the appropriate standards, make effective presentations, give and receive clear instructions
UE17ECE_PG_PO9	Engage in lifelong learning with persistent scientific temper for professional advancement and effective communication of the technical information.
UE17ECE_PG_PO10	Become a complete professional with high integrity and ethics, with excellent professional conduct and with empathy towards the environmental and contribute to the community for sustainable development of society
UE17ECE_PG_PO11	Critically evaluate the outcomes of one's actions and apply self corrective measures to improve the performance.

$\delta_1 = (W_2^T \delta_2) \odot \phi_1'(V_1(k))$.

$= \begin{pmatrix} 0.25 \\ 0.15 \\ -0.5 \\ 0.3 \end{pmatrix} \times 1.3808 \odot \begin{pmatrix} 1-0 \\ 1-0.05^2 \\ 1-0.35^2 \\ 1-0.1^2 \end{pmatrix}$ (1 − tanh²(V))

$= \begin{pmatrix} 0.3452 \\ 0.2071 \\ -0.6904 \\ 0.4142 \end{pmatrix} \odot \begin{pmatrix} 1 \\ 0.9975 \\ 0.8869 \\ 0.9901 \end{pmatrix}$

$\delta_1 = \begin{pmatrix} 0.3452 \\ 0.2066 \\ -0.6123 \\ 0.4101 \end{pmatrix}$.

$\Delta W_1 = \eta \delta_1 y_0^T$

$= 0.01 \times \begin{pmatrix} 0.3452 \\ 0.2066 \\ -0.6123 \\ 0.4101 \end{pmatrix} (1 \quad 0.5)$.

$= \begin{pmatrix} 0.0034 & 0.0017 \\ 0.0020 & 0.0010 \\ -0.0061 & -0.0031 \\ 0.0041 & 0.0021 \end{pmatrix}$.

$(\tanh x)' = \frac{1}{\cosh^2 x}$

$= \operatorname{sech}^2 x$.

4.b Heuristic For Making the BPA perform better.

1. Sequential v/s batch mode update:
   For large & redundant data, sequential processing is fast
   For smaller data & N/w, B.processing is faster.

(2). Maximizing information content:
   Aim: Every training example presented to the BPA should be chosen on the basis that its information content is the largest possible for the task at hand

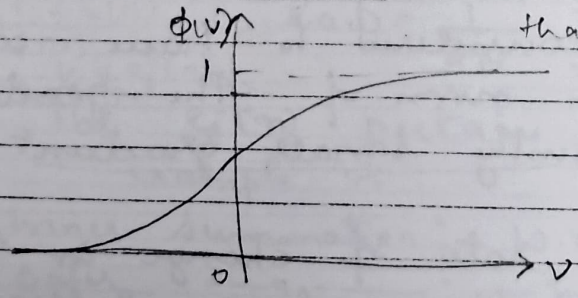
To achieve this aim ~~xxx~~ there are 2 ways.

(A). The use of an example that results in the largest training error.
   ∴ for updating w_{ji} value local gradient will be used which is a function of error
   When an example produces the larger error which leads to have larger local gradient, so faster will be the learning process

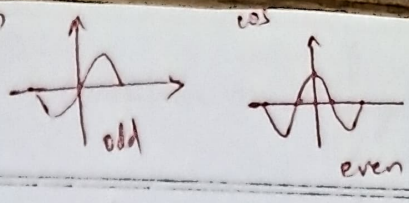
shuffling training pattern ex. classification

(B). The use of example that is radially different from all those previously used.

(3) Activation function: antisymmetric functions are better than non-symmetric functions



→ non-symmetric



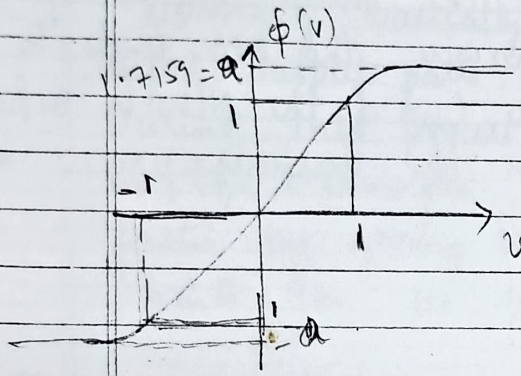
3. Activation function: choice of

As we discussed earlier, activation function play an impt. role in training NN using BPA. If we use tangent hyperbolic activation fun learning will be faster than logistic. Because T.Hyperbolic is antisymmetric where as logistic non symmetric about 0.

i.e $\phi(-v) = -\phi(v)$. → odd function (anti symmetric)

$\phi(v) = a\tanh(bv)$.

(LeCun chook suggested particular value for $a = 1.7159$ and $b = 2/3$. Because the following useful ppties.



① $\varphi(1) = 1$

$\varphi(-1) = -1$

② At the origin the slope (i.e effective gain) of the activation fun is close to 1

$\varphi(0) = ab$. - ?

$= 1.1424$

③ $\varphi''(v)$ attains its maximum value at $v = 1$.

$|v| < 1$: Region is linear

$|v| > 1$: Region is saturation

$\frac{d}{dx}\left(\frac{u}{v}\right) = \frac{u'v - v'u}{v^2}$ $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\frac{d\tanh(x)}{dx} = 1 - \tanh^2(x)$

$$\frac{d\tanh(x)}{dx} = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2}$$

$$= \frac{(e^x + e^{-x})^2}{(e^x + e^{-x})^2} - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - \tanh^2(x)$$

④ Target value:

It is important to choose the target value within the activation function range ie ± 1

for example

let us assume the desired o/p of neuron j is y_{dj}. If its it should be offset by some amount ϵ.

ie $y_{dj} = a - \epsilon$ (For ex. $\epsilon = 0.7159$ so that y_j is moved to linear region)

(or) $y_{dj} = -a + \epsilon$, $\epsilon \rightarrow$ +ve constant

(Here y_{dj} cannot exceed $a = 1.7159$ for tanh since y_{dj} are derived from input examples)

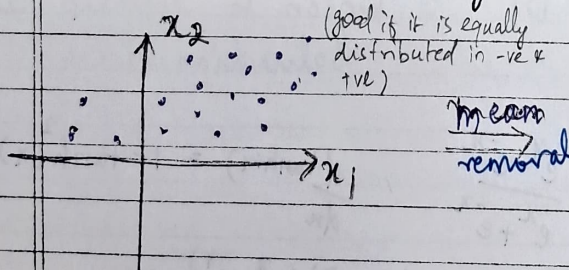
So that it will be in the linear region of activation function. Otherwise the BPA tends to drive the free parameters (w) of the NN to infinity and thereby slow down the learning process.

⑤ Normalization of inputs:

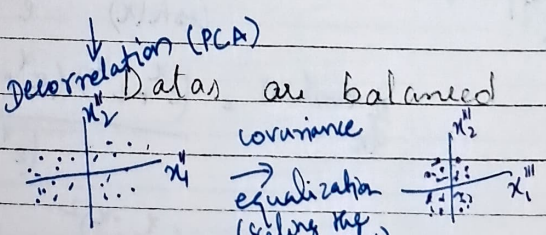
The training samples must be preprocessed before presenting to the NN

The preprocessing steps are

(a) Subtracting each input from mean



here data was not distributed properly as a result of the training process. from the sample coming from - ve side NN not able to perform bet



covariances in $x_1 \times x_2$ direction are equalized,

non zero mean → large E values → condition no is large → cost fun surface will be steep

(5) decorrelate the Training samples using PCA otherwise NN will take many epochs

The decorrelated variables must be scaled so that their covariances are approximately equal so that

(6) Covariances :^ it ensures the different synaptic weights in the n/w learn at approximately the same speed.

(6) Learning from hints: finding $f(\cdot)$ which is
learning basically means a mapping of unknown input-output pairs.
During this process it exploits the info present in the t-example. This process can be made faster by giving hints along with input examples. hints can be symmetric, invariance properties etc.

(7) Learning rate: assign η in this order

$$\eta_1 > \eta_2 > \eta_3 \dots > \eta_L$$

Initialization:

A good choice for the initial values of the synaptic weights and thresholds of the N/W can be more useful in a successful N/W design. Then what is a good choice?

If I choose large initial values for w, it is highly likely that the neurons in N/W will be driven into saturation. If this happens the local gradient in BPA assume small values, which in turn will cause the learning process to slow down.

If w's are assigned with small initial values, BPA may operate on a very flat area around the origin of the error surface; where origin is a saddle point

→ use of both large & small values of initial w's, but it lies in between these. (should be avoided)

To be specific, consider a tanh function as a.f

Let the bias applied to each neuron be set to zero & $v_j = \sum_{i=1}^{m} w_{ji} y_i$ (let i/ps applied to each neuron have $\mu_y = E[y_i] = 0 \quad \forall i$

and $\sigma_y^2 = E[(y_i - \mu_i)^2] = E[y_i^2] = 1$ → unit variance

assumption:
i) zero mean i/p
ii) unit variance

assumed that $E[y_i y_k] = \begin{cases} 1 & k = i \\ 0 & k \neq i \end{cases}$ iii) uncorrelated

and w's are drawn from a uniformly distributed set of no. with '0' mean

$\mu_w = E[w_{ji}] = 0 \quad \& \quad \sigma_w^2 = E[(w_{ji} - \mu_w)^2] = E[w_{ji}^2] - \mu_w^2 = E[w_{ji}^2]$

mean & variance of v_j induced local field.

$\mu_v = E[v_j] = E\left[\sum_{i=1}^{m} w_{ji} y_i\right] = \sum_{i=1}^{m} E[w_{ji}] E[y_i] = 0$

$E[(v_j - \mu_v)^2] \Leftarrow \sigma_v^2 = E[v_j^2] = E\left[\sum_{i=1}^{m} \sum_{k=1}^{m} w_{ji} w_{jk} y_i y_k\right] = \sum_{i=1}^{m} \sum_{k=1}^{m} E[w_{ji} w_{jk}] E[y_i y_k]$

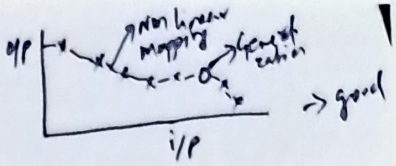
$= \sum_{i=1}^{m} E[w_{ji}^2] \quad i = k.$

$\sigma_w = \frac{1}{m^{1/2}}$ if $\sigma_v = 1$ $\qquad \sigma_v^2 = m \sigma_w^2$

where m is the no. of synaptic connections (fan-in)

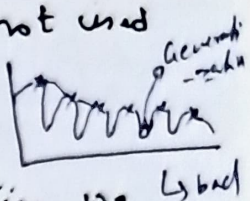
m - no. of inputs

Generalization & Cross validation.



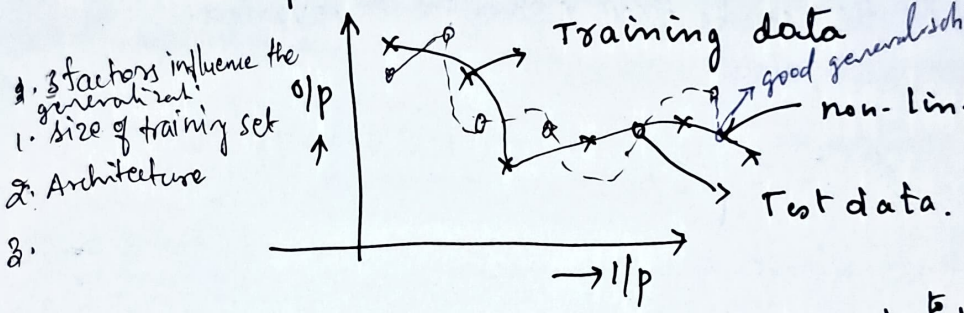
1) Generalization:

A N/W is said to generalize, when the I/p - o/p mapping computed by N/W is correct for test data not used in the training.



To understand the concept of generalization, we consider the N/W as o/m. which performs non linear input - output mapping.

The learning process is viewed as a curve-fitting problem. Then problem of mapping simply boils down to that of interpolation



The training data is used to help the N/W to learn the non linear input-output mapping represented by the curve.

When a test data is presented, the N/W computes the o/p as a result of interpolation performed by the N/W.

When a NN learns too many I/p - o/p samples, the N/W may end up memorizing the training set. w/o trying to find the underlying function to be modelled.

- overfitting / over training.

When over trained the N/w loses the ability to generalize

This normally happens when more no. of H. Neuron than necessary are used resulting in undesired contributions in the input space due to noise stored in wt

Generalization is influenced by.

- Size of training set and how close it is to the envt. of intrest.
- Arch. of NN.
- Physical complexity of the pblm. - no control over

→ either one of the them can be fixed & other one can be varied. this

Good generalization

Size of the training set

$$N = O\left(\frac{W}{\epsilon}\right) \quad O(\cdot) : \text{order of quantity.}$$

W: Total no. of free parameters (i.e synaptic weights & biases)

ϵ : error permitted on the test data.

ex. 0.01 = 1% of classification error.

~~Cross validation~~:

→ generalisation is equivalent to the use of this multi dimensional surface to interpolate test data. This is the motivation behind the method of RBF.

→ In the context of NN, the hidden units provide a set of functions that constitute an arbitrary basis for the input pattern when they are expanded into the hidden space. These functions are called as radial-basis functions.

Cross validation: available data set → training → estimation - used to select the model
→ test → validation - used to test or validate the model.

↓

To find multilayer perceptron with the best no. of hidden neurons & to figure out when it is best to stop training

4.15 Network Pruning Techniques:

→ Choosing a suitable topology for a NN, given the application is a difficult problem.

→ Usually, after a tedious trial & error process, an oversized topology is chosen which is prone to various drawbacks like:

1. high demand on computational resources
2. Increase in training time
3. non convergence of parameters
4. Decrease in the generalisation capability of a N/W
5. price for H/W implementation rises
6. less efficient.

If NN topology is too small, then it might not be able to learn the noise in the training data

To overcome this problem, we need to find minimal topologies with best performance

There are 2 ways to achieve

1. N/W growing approach: start with small multilayer perceptron neurons & then add a new neuron or a new layer of hidden neurons only when unable to reach desired specification
2. N/W Pruning Methods:

Generally latter method starts the training with a NN which is expected to be big enough to ensure a successful training

→ we start with a large multilayer perceptron with an adequate performance for the problem at hand, then prune it by weakening or eliminating certain synaptic weights in a selective and orderly fashion.

Pruning Methods: < deletion / Regularization

1. Deletion (Hessian based N/W pruning)

→ basic idea of an analytic approach to network pruning is to use information on 2nd order derivative of error surface.

→ a local model of the error surface is constructed for analytically predicting the effect of perturbations in synaptic weights.

→ starting point in the construction of such a model is the local approximation of the cost function $\mathcal{E}_{av}$ by using a Taylor series about the operating point

$$\mathcal{E}_{av}(w + \Delta w) = \mathcal{E}_{av}(w) + g^T(w)\,\Delta w + \tfrac{1}{2}\,\Delta w^T H\,\Delta w + H.O.T \quad \hookrightarrow (1)$$

where, Δw - perturbation applied to operating point w

$g(w)$ - gradient vector evaluated at w.

→ To identify the set of parameters to be deleted which causes the least increase in the value of the cost function $\mathcal{E}_{av}$.

→ To do this, we assume following approximations:

1. Extremal approximation: we assume that parameters are deleted only after the training process has converged. In such cases $g^T \Delta w = 0$

2. we assume that the error surface around a local minima is highly quadratic, hence H.O.T is neglected.

$$\therefore (1) \Rightarrow \mathcal{E}_{av}(w + \Delta w) - \mathcal{E}_{av}(w) = \frac{1}{2}\Delta w^T H \Delta w \quad \text{—(2)}$$

eq. (2) is the basis for the pruning procedure called optimal brain surgeon (OBS)

The goal of OBS is to set one of the synaptic weights to zero in order to minimize the incremental rise in ξ_{av}.

Let $w_i(n)$ denote this particular synaptic weight. The elimination of this weight is equivalent to

$$1_i^T \Delta w + w_i = 0 \quad , \quad 1_i^T \rightarrow \text{unit vector - all zeros except } i^{th} \text{ element}$$

Now the goal is

to minimize the quadratic form $\frac{1}{2}\Delta w^T H \Delta w$ w.r.t. to incremental change in the weight vector Δw, subject to the constraint that $1_i^T \Delta w + w_i = 0$ then minimize the result w.r.t. the index i.

To solve this constrained-optimization problem, we first construct Lagrangian

$$S = \frac{1}{2}\Delta w^T H \Delta w - \lambda(1_i^T \Delta w + w_i) \quad \text{— (3)}$$

λ - lagrange multiplier

$\frac{\partial}{\partial a}(a^T b) = b$

$$\frac{\partial S}{\partial \Delta w} = \frac{1}{2} \cdot 2 H \Delta w - \lambda 1_i^T = 0 \Rightarrow \Delta w = H^{-1} \lambda 1_i$$

$$\boxed{\Delta w = \lambda H^{-1} 1_i}$$

$$\frac{\partial S}{\partial \lambda} = -(1_i^T \Delta w + w_i)$$

$$= -(1_i^T H^{-1} 1_i \lambda + w_i) = 0$$

$$1_i^T H^{-1} 1_i \lambda = -w_i$$

$$\lambda = -\underbrace{(1_i^T H^{-1} 1_i)^{-1}}_{\text{scalar}} w_i$$

$$= \frac{-w_i}{1_i^T H^{-1} 1_i} \quad \text{or} \quad \frac{-w_i}{(H^{-1})_{ii}}$$

$\rightarrow ii$-th element of this inverse matrix.

$$\Rightarrow \Delta w = \frac{-w_i}{1_i^T H^{-1} 1_i} \cdot H^{-1} 1_i = \frac{-w_i}{(H^{-1})_{ii}} \cdot H^{-1} 1_i$$

$\lambda_i \to 1$ can be removed

Sub λ_i & ΔW in Eq (3) to obtain S_{min}.

$\lambda_i = \lambda$ in the following derivations.

$$= \frac{1}{2}(\lambda_i H^{-1} 1_i)^T H (W)(\lambda_i H^{-1} 1_i) - \lambda(1_i^T \lambda H^{-1} 1_i + W_i)$$

$$= \tfrac{1}{2}\lambda_i^2 1_i^T H^{-1T} H H^{-1} 1_i - \lambda_i(1_i^T \lambda_i H^{-1} 1_i + W_i) \quad (\because H^{-1T} = H^{-1})$$

$$= \tfrac{1}{2}\lambda_i^2 1_i^T H^{-1} 1_i - \lambda_i(\lambda_i (H^{-1})_{ii} + W_i)$$

$$= \tfrac{1}{2}\lambda_i^2 (H_i^{-1})_{ii} - \lambda_i^2 (H^{-1})_{ii} - \lambda_i W_i$$

$$= -\tfrac{1}{2}\lambda_i^2 (H^{-1})_{ii} - \lambda_i W_i$$

$$= -\tfrac{1}{2}\lambda_i^2 (H^{-1})_{ii} - \lambda_i(-\lambda_i (H^{-1})_{ii})$$

$$= -\tfrac{1}{2}\lambda_i^2 (H^{-1})_{ii} + \lambda_i^2 (H^{-1})_{ii}$$

$$S = \tfrac{1}{2}\lambda_i^2 (H^{-1})_{ii}$$

$$= \tfrac{1}{2}\left(\frac{-W_i}{(H^{-1})_{ii}}\right)^2 (H^{-1})_{ii}$$

$$S_{min} = \tfrac{1}{2}\frac{W_i^2}{(H^{-1})_{ii}}$$

Saliency $S_i = \frac{1}{2}\frac{W_i^2}{(H^{-1})_{ii}}$

small W_i → small effect on MSE but S_i inversely ∝ to H_{ii}^{-1} ∴ small H^* may ⇒ have ↑se in MSE, ∴ this is a trade off.

where H^* - Hessian matrix.

Lagrangian S, optimized w.r.t to ΔW subject to the constraint that the i^{th} synaptic weight W_i be eliminated is called saliency of W_i.

→ S_i represents the ↑se in the MSE or performance measure.

In OBS, W_i corresponds to smallest S_i is selected for deletion.

Supervised learning viewed as an optimization pblm

① optimization pblm means finding the best solution

② BPA is highly nonlinear function of w.

→ using Taylor series we expand $\xi_{av}(w)$ abt the current point on the error surface $w(n)$ as

$$\xi_{av}(w(n)+\Delta w(n)) = \xi_{av}(w(n)) + \frac{\partial \xi_{av}}{\partial w(n)} \frac{\Delta w(n)}{1!} + \Delta w^2 \frac{\partial^2 \xi_{av}}{\partial w(n)^2} \cdot \frac{\Delta w(n)}{2!} + \text{higher order terms}$$

(local gradient)

... continued.

→ sub, $\lambda \propto \Delta w$ in

$$S = \frac{1}{2}(-\lambda u_i H^{-1})^T H (-\lambda u_i H^{-1}) + \frac{w_i}{u_i^T H^{-1} u_i}(u_i^T \cdot -\lambda u_i H^{-1} + w_i)$$

$$S_i = \frac{w_i^2}{2[H^{-1}]_{ii}}$$

where H^{-1} – Hessian matrix

small w_i → small effect on MSE but S_i inversely ∝ to H^{-1}, ∴ small H^{-1} – may have ↑se in MSE, ∴ this is a trade off.

belong to previous page

Lagrangian S, optimized w. r. to Δw subject to the constraint that the ith synaptic weight w_i be eliminated, is called the saliency of w_i.

→ S_i represents the ↑se in the MSE or performance measure. In OBS, w_i corresponds to smallest S_i is selected for deletion.

- 2nd order optimization Method: rate of

→ Drawback in SDM is slower convergence, by adding α-momentum might help, but adding one more term will make the training process little complicated. (∵ Δw depends on $\frac{\partial \xi}{\partial w}$)

→ This is bcs SDM takes many steps to reach minimum. The method repeatedly moves in the steepest direction, consequently the repeated steps are ⊥r to each other, i.e in zig-zag manner. ∴ it is slower

→ To avoid this, we go for another tech. CGM, in which we consider conjugate vector v which will not take any steps, w.r.t 'u' has taken, by which it neglects multiple steps.

= steepest descent
= conjugate gradient

4.18 Conjugate Gradient Method.

→ it is a second-order optimization methods

→ Let us assume minimization of quadratic function

$$f(x) = \frac{1}{2} x^T A x - b^T x + c.$$

Where x is a $N \times 1$ parameter vector

A : $N \times N$ symmetric positive definite matrix

b : $N \times 1$ vector

c : constant.

minimization of quadratic fun. $f(x)$ is achieved by

$$f'(x) = 0.$$

$$\Rightarrow \quad A x^* - b = 0$$

$$x^* = A^{-1} b.$$

Thus minimizing of $f(x)$ and solving Linear sys of Eq^n's $Ax^* = b$ are equivalent pblms.

Definition:

Given for a given matrix A, we say that a set of non zero vectors $[s(0)\ s(1) \ldots s(N-1)]$ is A-conjugate. If the following condition is satisfied.

$$s^T(n) A s(j) = 0 \qquad \forall\ n, j \text{ s.t } n \neq j.$$

If A = Identity matrix, then it implies notion of orthogonality

For a given set of A-conjugate vectors $s(0), s(1), \ldots, s(N-1)$, the corresponding conjugate direction method for unconstrained minimization of the quadratic error function $f(x)$ is defined by

$$x(n+1) = x(n) + \eta(n)\, s(n) \qquad n = 0, 1, \ldots, N-1.$$

where $x(0)$ is an arbitrary starting vector

$\eta(n)$: scalar. defined by

$$f(x(n) + \eta(n)\, s(n)) = \min_{\eta} f(x(n) + \eta\, s(n)).$$

The procedure of choosing η so as to minimize the fun $f(x(n) + \eta\, s(n))$ for some fixed n is referred to as line search which represents 1-Dimensional minimization problem

$$f(x(n) + \eta(n) s(n)) = \frac{1}{2}\left[x(n) + \eta(n) s(n)\right]^T A \left[x(n) + \eta(n) s(n)\right] - b \left[x(n) + \eta(n) s(n)\right]^T + c.$$

$$f(x(n) + \eta(n) s(n)) = \frac{1}{2}\{x^T(n) A x(n) + \eta(n) s^T(n) A x(n) + \eta x^T(n) A s(n) + \eta^2(n) s^T(n) A s(n)\} - b^T x(n) - \eta(n) b^T s(n) + c$$

$$\frac{\partial f(\cdot)}{\partial \eta} = 0 + \frac{1}{2} s^T(n) A x(n) + \frac{1}{2} x^T(n) A s(n) + \eta(n) s^T(n) A s(n) - b^T s(n)$$

$$0 = s^T(n) A x(n) - b^T s(n) + \eta(n) s^T(n) A s(n)$$

$$\eta(n)\, s^T(n) A s(n) = -s^T(n)\{Ax - b\} \qquad A = A^T = A^{-1}$$

$$\eta(n) = -\frac{s^T(n) A \{x(n) - A^{-1} b\}}{s^T(n) A s(n)}$$

$$x(n) - x^* = \varepsilon$$

where $x^* = A^{-1} b$

$$\boxed{\eta(n) = \frac{-s^T(n) A \varepsilon(n)}{s^T(n) A s(n)}}$$

starting from an arbitrary point $x(0)$ the conjugate direction method is guaranteed to find optimal sol^n x^* of $f(x)=0$ in at most N iterations.

At successive iterations, the CG method minimizes the quadratic function $f(x)$ over a progressively expanding linear vector space that eventually includes the global minimum of $f(x)$. ie for each iteration $x(n+1)$ minimizes $f(x)$ that passes through some arbitrary point $x(0)$

For the CG method to work we require the availability of set A-conjugate vectors $s(0)\ s(1) \ldots s(N-1)$

It is determined in a sequential manner at successive step.

→ Define residual $r(n) = b - Ax(n)$ as the steepest descent direction

linear combination
∴ L.C of $r(n)$ & $s(n-1)$ ⇒ $s(n) = r(n) + \beta(n)\, s(n-1)$

$n = 1 \ldots N-1$

Where $\beta(n)$ is scaling factor to be determined s.T, A-conjugacy is preserved.

$$\Rightarrow s^T(n-1)\, A\, s(n) = s^T(n-1)\, A\, r(n) + \beta(n)\, s^T(n-1)\, A\, s(n-1).$$

$$0 = s^T(n-1)\, A\, r(n) + \beta(n)\, s^T(n-1)\, A\, s(n-1)$$

$$\Rightarrow \boxed{\beta(n) = -\frac{s^T(n-1)\, A\, r(n)}{s^T(n-1)\, A\, s(n-1).}}$$

To avoid the dependency on A to generate s's vectors. we consider

$x^* = A^{-1}b$

$e(n) = x(n) - x^*$

Polak - Ribiere formula

$$\beta(n) = \frac{r^T(n)[r(n) - r(n-1)]}{r^T(n-1)\, r(n-1)}$$

Fletcher - Reeves formula

$$\beta(n) = \frac{r^T(n)\, r(n)}{r^T(n-1)\, r(n-1)}.$$

Given $x(0), s(0)$

find $r(0)$

① $r(0) = b - Ax(0)$

$\beta(0) = -\dfrac{s^T(0) A e(0)}{s^T(0) A s(0)}$

$x(1) = x(0) + \eta(0)\beta(0)$

② $r(1) = b - Ax(1)$

$\beta(1) = -\dfrac{s^T(0) A r(1)}{s^T(0) A s(0)}$

$s(1) = r(1) + \beta(1) s(0)$

$\eta(1) = -s^T(1) A e(1)$