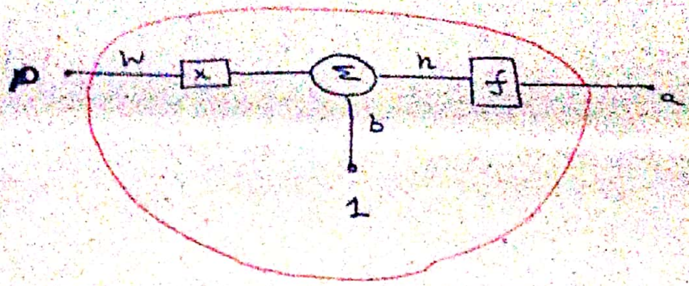
NN-FL

Class - 1

Neural Networks

Single input neuron



$x = p \cdot w$

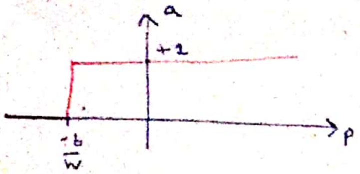
$n = x + b \cdot 1 = pw + b$

$a = f(n) = f(p \cdot w + b)$

Transfer / Activation functions

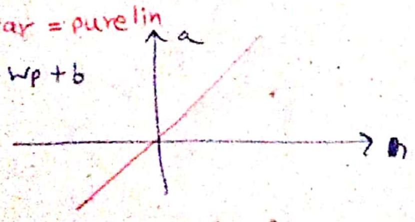
* Hardlimit = step function

  * $a = \text{hardlim}(wp+b) \Rightarrow p \geq -\frac{b}{w} \rightarrow 1 \;,\; p < -\frac{b}{w} \rightarrow 0$



* Linear = purelin

  * $a = wp + b$

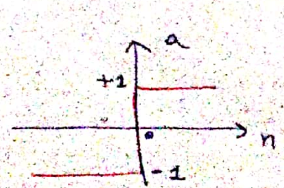


* Log-sigmoid

  * $a = \frac{1}{1+e^{-n}}$; $n = wp + b$

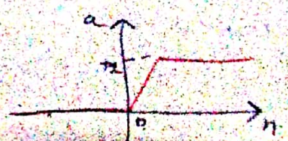
* Symmetric - Hard limit

  * $a = \begin{cases} +1, & n \geq 0 \\ -1, & n < 0 \end{cases}$



* saturating linear

  * $a = \begin{cases} 0 ; & n < 0 \\ n ; & 0 \leq n \leq 1 \\ 1 ; & n > 1 \end{cases}$



* Symmetric Saturating linear

* $a = \begin{cases} -1 \; ; \; n < -1 \\ n \; ; \; -1 \le n \le 1 \\ 1 \; ; \; n > 1 \end{cases}$

* Hyperbolic tangent sigmoid

* $a = \left\{ \dfrac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \right\}$

* Positive linear

* $a = \begin{cases} 0 \; ; \; n \le 0 \\ n \; ; \; n > 0 \end{cases}$

* Competitive

* $a = \begin{cases} 1 \; ; \text{ neuron with maximum of value } n \\ 0 \; ; \text{ all other neurons} \end{cases}$

$\longrightarrow$ $n = \underline{W} \cdot \underline{P} + b$

$W = [\, w_{11} \;\; w_{12} \;\; w_{13} \; \cdots \; w_{1r} \,]_{1 \times R}$

$P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_r \end{bmatrix}_{R \times 1}$ $\Rightarrow a = f(W \cdot P + b)$

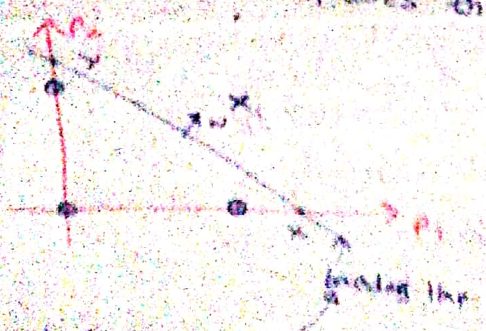
Class - 4

1) Graphical method

$W^T p + b = 0$ ← Boundary line ⇒ $W^T p$ same $= (-b)$ in boundary line.

$\longleftrightarrow$ boundary $\perp$ W

$\longrightarrow$ if boundary line passes through origin then $b = 0$.



$W = (w_1 \;\; w_2)$

slope of boundary line $= \dfrac{x - y}{u}$ ← (0, v)
$= (u - v)$

$\therefore W \cdot (u - v)^T = 0$

$\therefore W = [\, \frac{v}{u} \;\; 1 \,]$

$\therefore b = -W p_1$

$= [\, \frac{v}{u} \;\; 1 \,] \begin{bmatrix} u \\ 0 \end{bmatrix}$

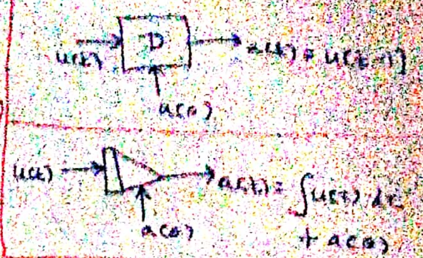
$\therefore b = v$ //

Hardlim //.

2) General Method → Perceptron Rule

* $w^{new} = w^{old} + e \cdot \underline{p}$

* $b^{new} = b^{old} + e$; $e = t - a$ (target, output)



Purelin //.

3) Hebbian Learning Rule (Hebb Rule)

* Training set = $\{\underline{p_1}, \underline{t_1}\}, \{\underline{p_2}, \underline{t_2}\}, \{\underline{p_3}, \underline{t_3}\}, \ldots, \{\underline{p_n}, \underline{t_n}\}$

$$W^{new} = W^{old} + \vec{t_q} \cdot \vec{p_q}$$

$$b^{new} = b^{old} + \vec{t_q}$$

* Initial weight matrix, bias = 0 :- $W = T \cdot P^T$ where

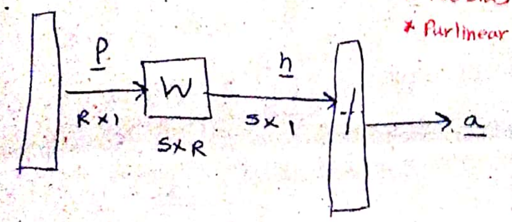
$$b = \sum_{i=1}^{n} t_i$$

$T = [\underline{t_1} \;\; \underline{t_2} \cdots \underline{t_n}]$,

$P = [\underline{p_1} \;\; \underline{p_2} \cdots \underline{p_n}]$.

Linear Associator

* No bias
* Purlinear function



; $\underline{a} = purelin(\underline{n}) = \underline{n} = \underline{W} \cdot \underline{p} + 0$

$\underline{a} = \underline{W} \cdot \underline{p}$ — (1)

$a_i = \sum_{j=1}^{R} w_{ij} \cdot p_j$

For an input p_k ;

But, $W = \underline{t_1} \cdot \underline{p_1}^T + \underline{t_2} \cdot \underline{p_2}^T + \cdots + \underline{t_n} \cdot \underline{p_n}^T$

$$\underline{W} = \sum_{i=1}^{n} \underline{t_i} \cdot \underline{p_i}^T \quad \text{— (2)}$$

$$a_k = \left(\sum_{i=1}^{n} \underline{t_i} \cdot \underline{p_i}^T\right) \cdot \underline{p_k}$$

$$a_k = \sum_{i=1}^{n} \underline{t_i} \cdot (\underline{p_i}^T \cdot \underline{p_k})$$

If p_i's are orthonormal Then, $\underline{p_i}^T \cdot \underline{p_k} = \begin{cases} 1 ; & i = k \\ 0 ; & i \neq k \end{cases}$

$a_k = t_k$

∴ Input vectors are orthonormal then Heb rule produces correct output.

Page 20 in Hasindhu's notes //

purelin/1, **Pseudo inverse Rule**

* If the input vectors are not orthonormal then hebb rule gives an error.

* $W \cdot \underline{p_q} = \underline{t_q} \quad \forall q = 1, 2, 3, \ldots, Q$

↳ performance index / error ; $F(W) = \sum_{q=1}^{Q} \| \underline{t_q} - W \cdot \underline{p_q} \|^2$

$$F(W) = \|E\|^2 \Rightarrow E = T - W \cdot P$$

* $F(W)$ can be zero if, $\boxed{W = TP^{-1}}$ ← P is not always a square matrix ⇒ Not always possible to take P^{-1}

$$W = T P^{+}$$

↑ Pseudoinverse matrix

$$\boxed{\begin{aligned} &P \cdot P^{+} \cdot P = P \\ &P^{+} \cdot P \cdot P^{+} = P^{+} \\ &P^{+} P = (P^{+} P)^T \\ &P P^{+} = (P P^{+})^T \end{aligned}}$$

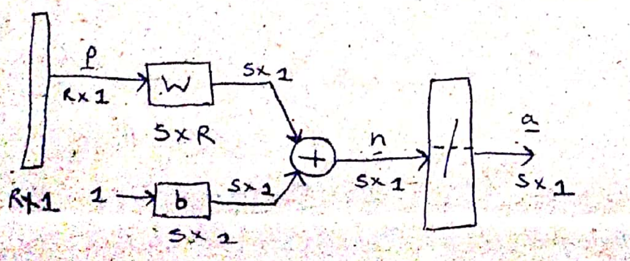
Note :- Number of rows in P > Number of columns in P, Columns of P are independant

$\longrightarrow P^{+} = (P^T P)^{-1} \cdot P^T$

purelin/1, **Widrow - Hoff Learning Rule**

ADALINE Network → A perceptron network with a linear transfer function

↑ ADAptive LInear NEuron



; $\underline{a} = purelin(\underline{n})$

$\underline{a} = \underline{n}$

LMS [Least Mean Squared] Algorithm

* $\underline{x} = \begin{bmatrix} \underline{w} \\ b \end{bmatrix}$, $\underline{z} = \begin{bmatrix} \underline{p} \\ 1 \end{bmatrix}$:- $a = purelin(n) = n$

$$\boxed{a = \underline{x}^T \cdot \underline{z}}$$

But, MSE $= F(\underline{x}) \Rightarrow$

$F(x) = E(e^2) = E[(t-a)^2] = E[(t - \underline{x}^T \underline{z})^2]$

$\therefore F(\underline{x}) = E[t^2 - 2t \cdot \underline{x}^T \underline{z} + (\underline{x}^T \cdot \underline{z}) \cdot (\underline{x}^T \underline{z})^T]$

$= E[t^2] - 2\underline{x}^T \cdot E[t \cdot \underline{z}] + \underline{x}^T \cdot E[\underline{z} \cdot \underline{z}^T] \cdot \underline{x}$

(min(0,0), min(1,0), min(0,0) = 0

$$F(\underline{x}) = c - 2\underline{x}^T \cdot \underline{h} + \underline{x}^T \cdot R \cdot \underline{x}$$

c ↑ $E[t^2]$, $\underline{h}$ ↑ $E[t \cdot \underline{z}]$, R ↑ $E[\underline{z} \cdot \underline{z}^T]$

→ General form of quadratic function :-

$$F(\underline{x}) = c + \underline{d}^T \cdot \underline{x} + \frac{1}{2} \cdot \underline{x}^T \cdot A \cdot \underline{x}$$

$$\therefore \underline{d} = -2\underline{h} \quad , \quad A = 2R$$

Locating the minimum ⇒

$$\nabla F(\underline{x}) = \nabla(c + \underline{d}^T \cdot \underline{x} + \frac{1}{2} \cdot \underline{x}^T \cdot A \cdot \underline{x})$$

$$= \underline{d}^T \cdot \frac{1}{2} \cdot 2A \cdot \underline{x}$$

$$\therefore \nabla F(\underline{x}) = -2\underline{h} + 2R \cdot \underline{x}$$

Positive definite :-

* Square matrix = A ;

$\forall \underline{x} \neq 0$:- $\underline{x}^T \cdot A \cdot \underline{x} > 0$

Semi-Positive definite :-

* square matrix = A ;

$\forall \underline{x} \neq 0$:- $\underline{x}^T \cdot A \cdot \underline{x} \geq 0$

* @ stationary points,

$$\nabla F(\underline{x}) = 0$$

$$R \cdot \underline{x} = \underline{h}$$

$$\underline{x}^* = R^{-1} \cdot \underline{h}$$

→ $W(k+1) = W(k) + 2\alpha \cdot e(k) \cdot \underline{p}^T(k)$

→ $b(k+1) = b(k) + 2\alpha \cdot e(k)$

where,

$$0 < \alpha < \frac{1}{\lambda_{max}} ;$$

λ_{max} = Maximum eigen value of R ;

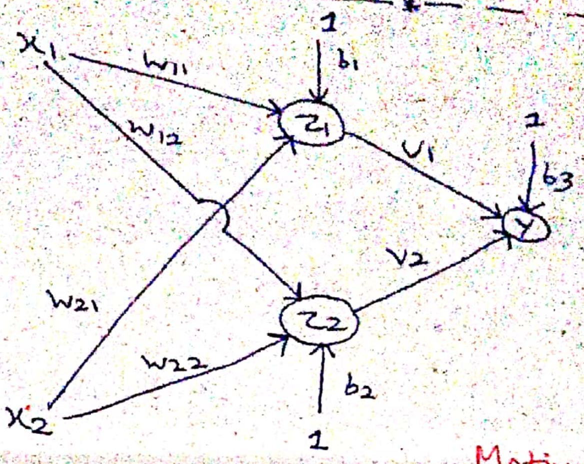
$R = E[\underline{z} \cdot \underline{z}^T]$:-

$$\underline{z} = \begin{bmatrix} \underline{p} \\ 1 \end{bmatrix}$$

$$R = \frac{1}{2}[p_1 \cdot p_1^T + p_2 \cdot p_2^T]$$

hardlim/1.

MADALINE [Many ADALINE]



Training
→ MR1 : changes $w_{11}, w_{12}, w_{21}, w_{22}, b_1, b_2$
→ MR2 : changes all parameters

Motivation for updates ;

1) we perform the updates only if there is an error

2) If $t = 1$ & $y = -1$
* $z_1 = z_2 = -1$
* Need to make atleast one of them to 1

3) If $t = -1$ & $y = 1$
* Atleast one z has 1
* Need to make all to -1

* → $t = 1, y = -1$:→

$w_{11}(new) = w_{11}(old) + \alpha \cdot (1 + z_{in1})$,

$w_{21}(new) = w_{21}(old) + \alpha \cdot (1 + z_{in1})$,

$w_{12}(new) = w_{12}(old) + \alpha \cdot (1 + z_{in2})$,

$w_{22}(new) = w_{22}(old) + \alpha \cdot (1 + z_{in2})$.

* → $t = -1, y = 1$:→

$w_{11}(new) = w_{11}(old) + \alpha \cdot (-1 - z_{in1})$,

$w_{21}(new) = w_{21}(old) + \alpha \cdot (-1 - z_{in1})$,

$w_{12}(new) = w_{12}(old) + \alpha \cdot (1 + z_{in2})$,

$w_{22}(new) = w_{22}(old) + \alpha \cdot (1 + z_{in2})$.