Perceptron

① First ~~NH~~ NN Model
Mc Culloch-Pitts

① Learning
supervised & unsupervised

→ (i) classification
(ii) Regression

→ clustering

(ii) N/W
(a) feed forward N/W
(b) Recurrent N/W

Weight Matrix

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots \\ w_{21} & w_{22} & \cdots & \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

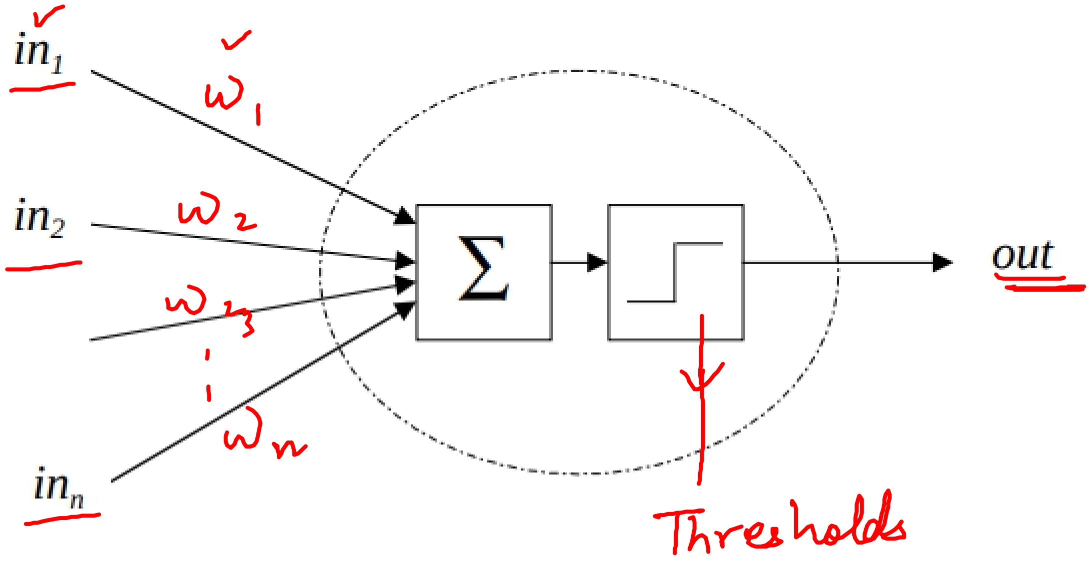
McCulloch-Pitts Neuron

- In 1943, Warren McCulloch and Walter Pitts introduced one of the first artificial neurons.
- The main feature of their neuron model is that a weighted sum of input signals is compared to a threshold to determine the neuron output.
- When the sum is greater than or equal to the threshold, the output is 1. When the sum is less than the threshold, the output is 0.

$$\text{output} = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i \, in_i \geq \theta \\ 0, & \text{if } \sum w_i \, in_i < \theta \end{cases}$$

Threshold value

Threshold Activation Function

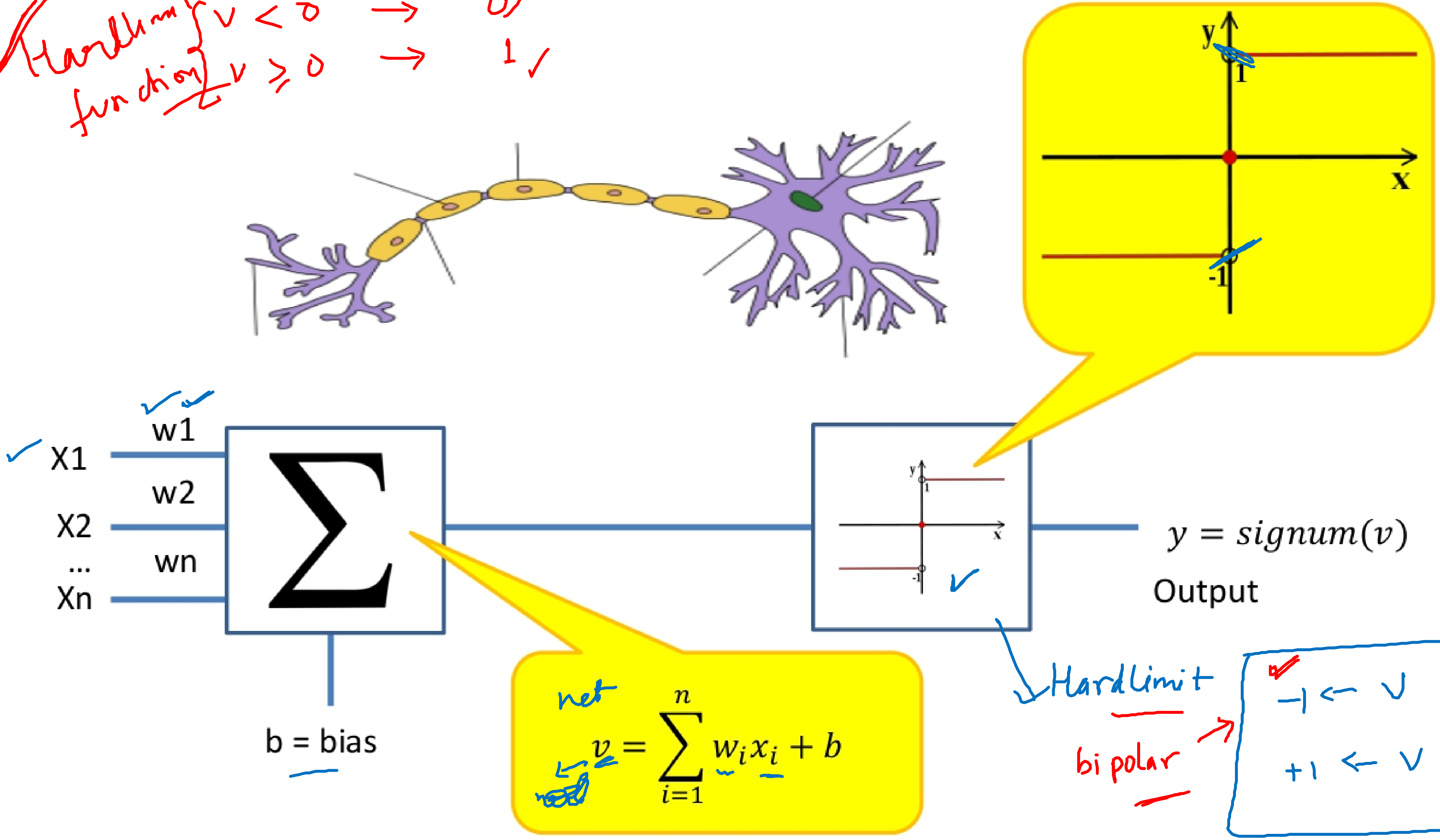


Not considered bias

Perceptron Classifier

- The perceptron is a simplified representation of the biological neuron in the brain.
- It is also known as the Single Layer Perceptron (SLP).
- The perceptron model was proposed by McCulloch & Pitts in 1943.
- The perceptron is the simplest form of a neural network for patterns that are linearly separable.
- The structure of a perceptron consists of a single neuron with adjustable synaptic weights and bias.
- The weights are adjusted during the training phase, as training data is presented to it.
- The model consists of a linear combiner followed by a hard limiter (performing the signum function).
- Also incorporates an externally applied bias.
- Output is +1 (if hard limiter output is positive) and −1 (if hard limiter output is negative).

Hardlimit function $\begin{cases} v < 0 \rightarrow 0 \\ v \geq 0 \rightarrow 1 \end{cases}$

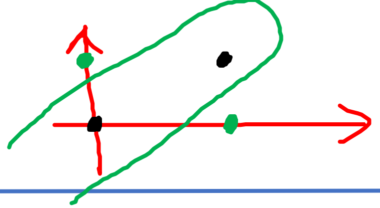


Hardlimit

bipolar

$-1 \leftarrow v < 0$

$+1 \leftarrow v \geq 0$



Lineraly Separable

XOR

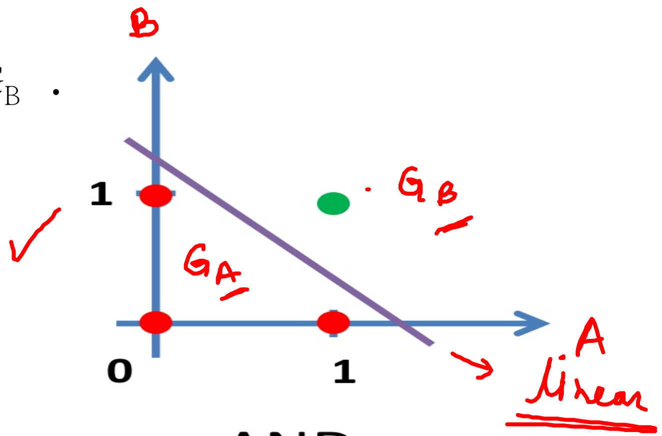
- Perceptrons can only classify linearly separable cases.
- Lets say we want to classify a set of data into either Group A (G_A) or Group B (G_B).
- If G_A and G_B are linearly separable, there exists a separating hyperplane between the two groups which is linear in nature.
- In simple terms, there is a straight line dividing between G_A and G_B .
- Consider the cases of AND and OR:

AND

INPUT (A)	INPUT (B)	OUTPUT (AND)
0	0	0
0	1	0
1	0	0
1	1	1

G_A

G_B



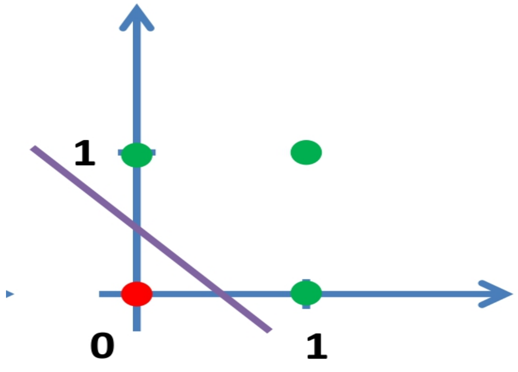
AND

OR

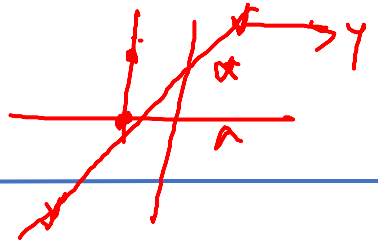
INPUT (A)	INPUT (B)	OUTPUT (OR)
0	0	0
0	1	1
1	0	1
1	1	1

G_A

G_B

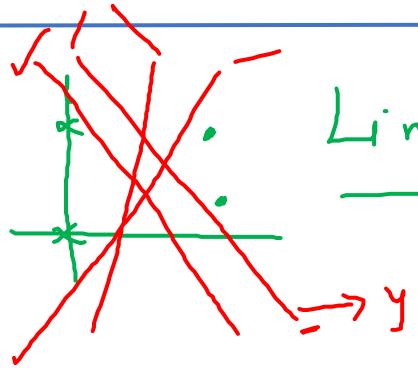


OR

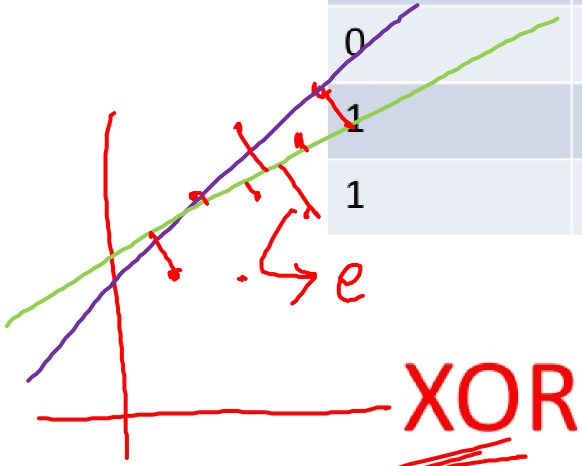


$e^2 = (t-a)^2 \to$ LMS

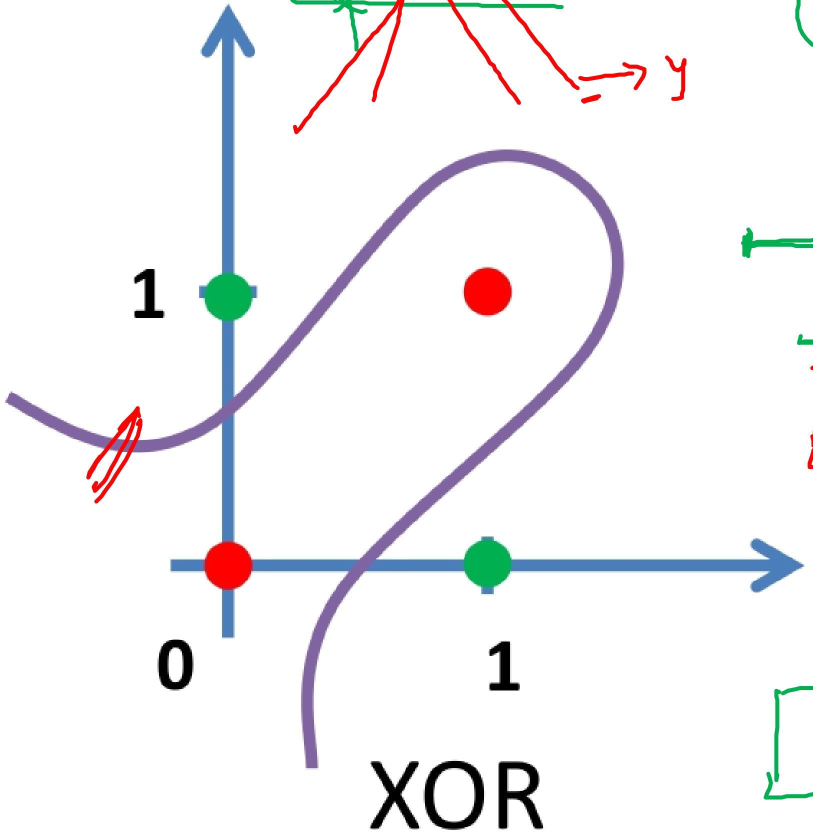
INPUT (A)	INPUT (B)	OUTPUT (XOR)
0	0	0
0	1	1
1	0	1
1	1	0



Linear ~~Separatta~~ Separable



XOR case is nonlinearly separable!



$y = mx + c$

oup i/p

$Y \Rightarrow wx + b$

Perceptron Architecture

- First, consider the network weight matrix:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}.$$

- define a vector composed of the elements of the ith row

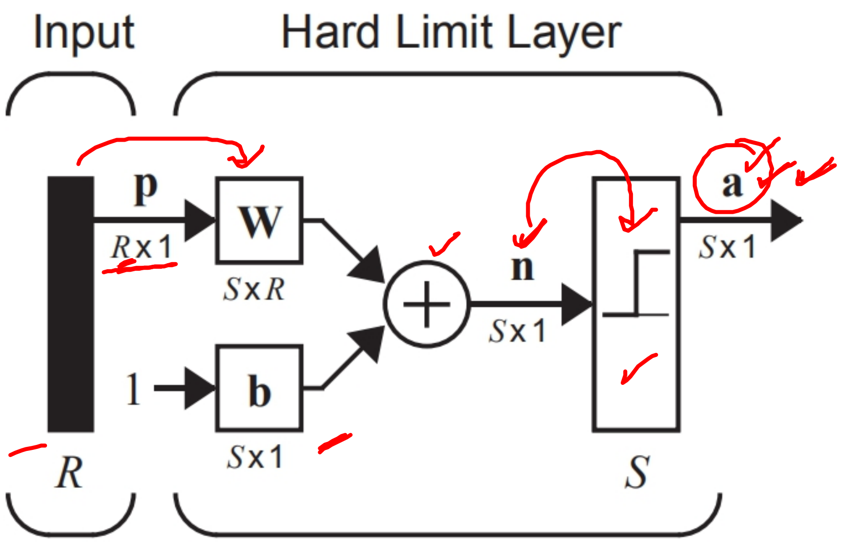
$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

- Now we can partition the weight matrix:

$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

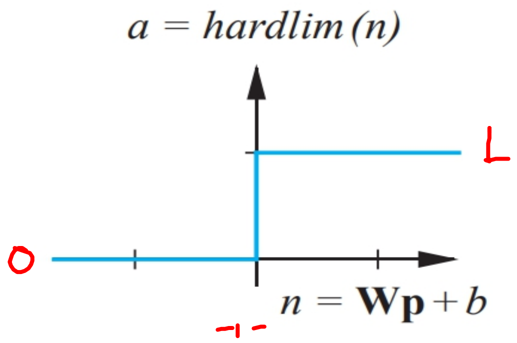
- the ith element of the network output vector as

$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

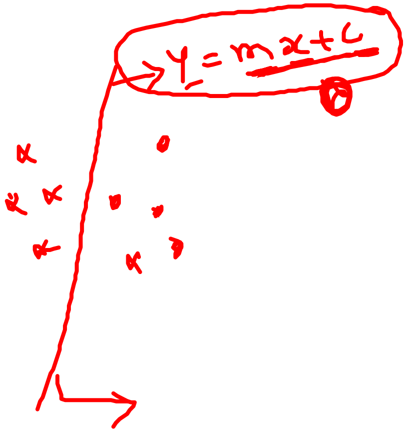
$$a = hardlim(n) = \begin{cases} 1 & if \quad n \geq 0 \\ 0 & otherwise. \end{cases}$$

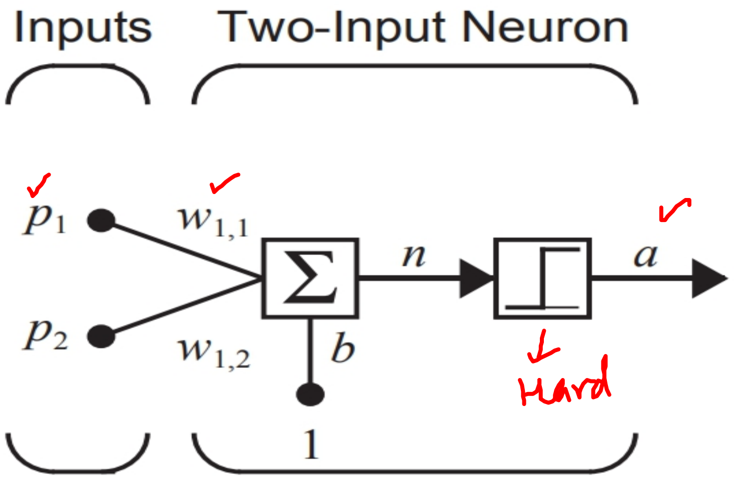


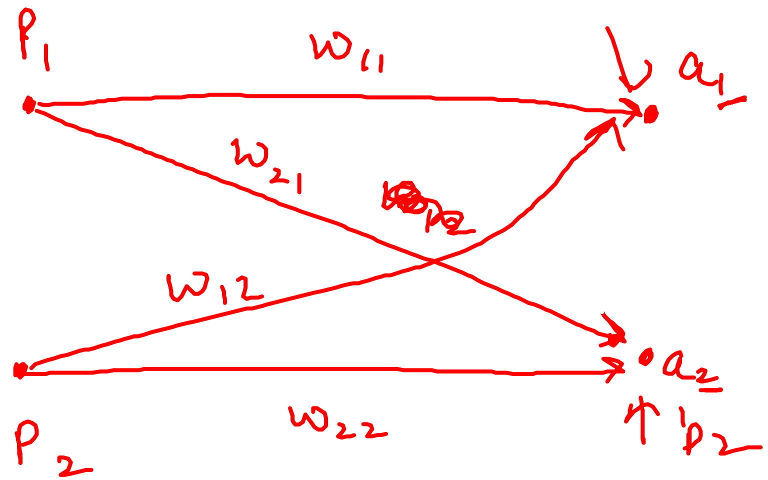
Single-Neuron Perceptron

$w_{ij} \to$ ip, $\to$ o/p

Let's consider a two-input perceptron with one neuron







$$a_1 = f(p_1 w_{11} + p_2 w_{12} + b_1)$$

$$a_2 = f(p_1 w_{21} + p_2 w_{22} + b_2)$$

The output of this network is determined by

$$a = hardlim(n) = hardlim(\mathbf{Wp} + b)$$

$$= hardlim({}_1\mathbf{w}^T\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

Single-Neuron Cont..

Decision Boundary: The decision boundary is determined by the input vectors for which the net input is zero:

$$n = \underbrace{{}_1\mathbf{w}^T\mathbf{p}} + b = w_{1,1}p_1 + w_{1,2}p_2 + b = 0$$

let's assign the following values for the weights and bias:

$$w_{1,1} = 1 \,,\; w_{1,2} = 1 \,,\; b = -1$$

The decision boundary is then

$$n = {}_1\mathbf{w}^T\mathbf{p} + b = w_{1,1}p_1 + \underbrace{w_{1,2}p_2}_{0} + b = p_1 + p_2 - 1 = 0$$

To find the p_2 intercept set p_1=0:

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{-1}{1} = 1 \qquad \text{if } p_1 = 0$$

To find the p_1 intercept set p_2=0:

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{-1}{1} = 1 \qquad \text{if } p_2 = 0$$

$n < 0$

$wp + b = 0$

$w_{11} p_1 + w_{12} p_2 + b = 0 \Rightarrow$

$\boxed{\text{let us} \quad w_{11} = 1, \quad w_{12} = 1, \quad b = -1}$

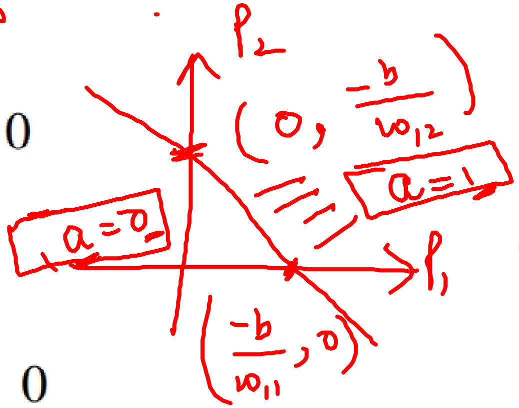
for $p_1 = ?\,,\ p_2 = 0$

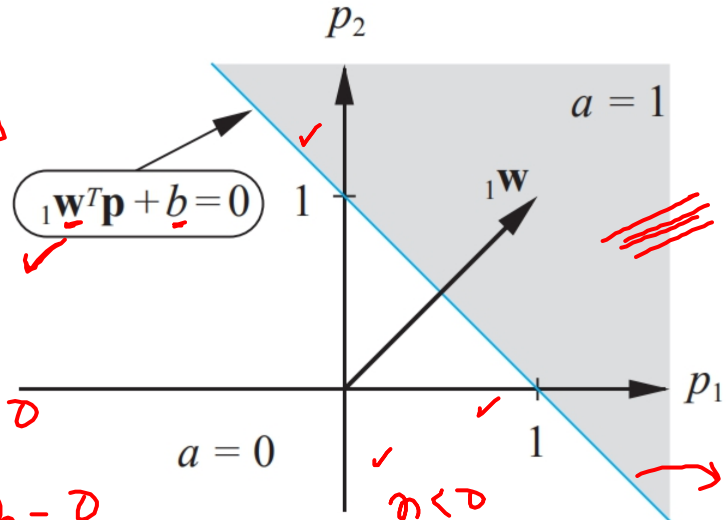
$w_{11} p_1 + b = 0$

$\boxed{p_1 = -b / w_{11}}$

$wp + b \geq 0$

$n \geq 0$





$p_2 = ?\,,\ p_1 = 0$

$w_{12} p_2 + b = 0$

$p_2 = -b / w_{12}$

$n < 0$

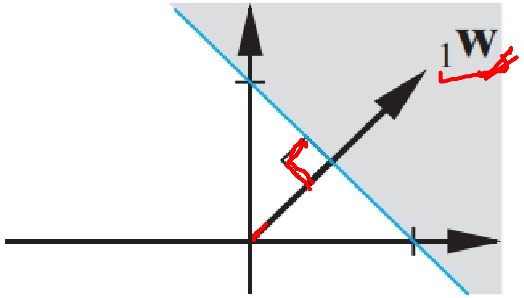
$wp + b < 0$

$n = 0$

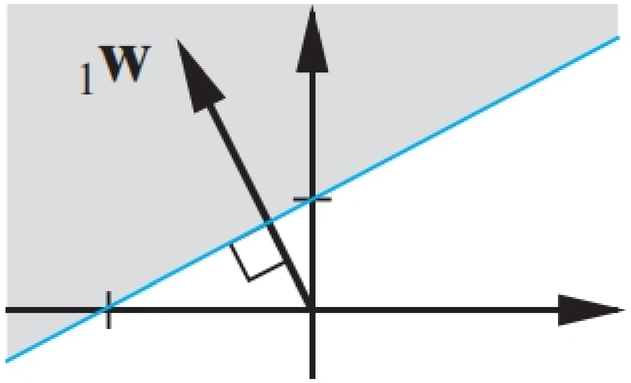
Single-Neuron Cont..

$n > 0$

$$_1\mathbf{w}^T\mathbf{p} + b = 0$$



- The boundary is always orthogonal to $_1W$
- any vector in the shaded region, will have an inner product greater than $-b$, and vectors in the unshaded region will have inner products less than $-b$.
- the weight vector will always point toward the region where the neuron output is 1



Supervised Learning Algorithm

1. Generate a training pair or pattern:
   - an input $\mathbf{x} = [\, x_1 \; x_2 \ldots x_n]$
   - a target output y_{target} (known/given)
2. Then, present the network with $\mathbf{x}$ and allow it to generate an output $\mathbf{y}$
3. Compare y with y_{target} to compute the error
4. Adjust weights, $\mathbf{w}$, to reduce error
5. Repeat 2-4 multiple times

10000

i/p	target
	0
	1

Training process

compute error

$\boxed{e = t - a}$ → −1

1 − 0

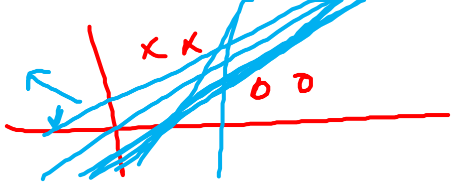
$\boxed{1 > 0}$

$e = 0$ → Can not weight & bias

$e > 0$

$e < 0$

→ we can update weight & bias



Perceptron Learning

Supervised binary classification 0 1

- For simple Perceptrons performing classification, we have seen that the decision boundaries are hyperplanes, and we can think of learning as the process of shifting around the hyperplanes until each training pattern is classified correctly.
- Somehow, we need to formalise that process of "shifting around" into a systematic algorithm that can easily be implemented on a computer.
- The "shifting around" can conveniently be split up into a number of small steps.
- If the network weights at time t are $w_{ij}(t)$, then the shifting process corresponds to moving them by an amount $\Delta w_{ij}(t)$ so that at time t+1 we have weights

new old

$$\boxed{w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)}$$

ηe w, b →

- It is convenient to treat the thresholds as weights, as discussed previously, so we don't need separate equations for them.

Formulating the Weight Changes

- Suppose the target output of unit j is $targ_j$ and the actual output is $out_j = sgn(\sum in_i\ w_{ij})$, where in_i are the activations of the previous layer of neurons (e.g. the network inputs). Then we can just go through all the possibilities to work out an appropriate set of small weight changes, and put them into a common form:

If $out_j = targ_j$ do nothing — Note $targ_j - out_j = 0$

so $w_{ij} \to w_{ij}$

If $out_j = 1$ and $targ_j = 0$ — Note $targ_j - out_j = -1$

then $\sum in_i\ w_{ij}$ is too large

first when $in_i = 1$ decrease w_{ij}

so $w_{ij} \to w_{ij} - \eta = w_{ij} - \eta\ in_i$

and when $in_i = 0$ w_{ij} doesn't matter

so $w_{ij} \to w_{ij} - 0 = w_{ij} - \eta\ in_i$

so $w_{ij} \to w_{ij} - \eta\ in_i$

If $out_j = 0$ and $targ_j = 1$ — Note $targ_j - out_j = 1$

then $\sum in_i w_{ij}$ is too small

first when $in_i = 1$ increase w_{ij}

so $w_{ij} \to w_{ij} + \eta = w_{ij} + \eta\ in_i$

and when $in_i = 0$ w_{ij} doesn't matter

so $w_{ij} \to w_{ij} - 0 = w_{ij} + \eta\ in_i$

so $w_{ij} \to w_{ij} + \eta\ in_i$

It has become clear that each case can be written in the form:

$$w_{ij} \to w_{ij} + \eta\ (targ_j - out_j)\ in_i$$

$$\Delta w_{ij} = \eta\ (targ_j - out_j)\ in_i$$

- This weight update equation is called the Perceptron Learning Rule. The positive parameter η is called the *learning rate* or *step size* — it determines how smoothly we shift the decision boundaries.

Convergence of Perceptron Learning

- The weight changes Δw_{ij} need to be applied repeatedly — for each weight w_{ij} in the network, and for each training pattern in the training set. One pass through all the weights for the whole training set is called one *epoch* of training.
- Eventually, usually after many epochs, when all the network outputs match the targets for all the training patterns, all the Δw_{ij} will be zero and the process of training will cease. We then say that the training process has *converged* to a solution.
- It can be shown that if there does exist a possible set of weights for a Perceptron which solves the given problem correctly, then the Perceptron Learning Rule will find them in a finite number of iterations
- Moreover, it can be shown that if a problem is linearly separable, then the Perceptron Learning Rule will find a set of weights in a finite number of iterations that solves the problem correctly

PERCEPTRON LEARNING ALGO $\rightarrow$ input X, t

[1] Initialize weight & bias ; set Randomly $W(0)$ & $b(0)$ $\Rightarrow$ $W = [0]$ & $b = [0]$

x			t
1	2	1	1
1	1	1	1
2	2	2	0

[2] For each training pairs (X, t) $\rightarrow$ y_{target}

$\rightarrow$ compute $\underset{y_{actual}}{a} = \text{hardlim}(W X + b)$

$\rightarrow$ compute error = target − actual

$e = t - a = y_{target} - y_{actual}$

$\boxed{\text{if } e = 0 \text{ No update}}$

$\rightarrow$ for updation of weight & bias

$$W^{new} = W^{old} + \eta * e * X$$

$$b^{new} = b^{old} + \eta * e$$

update for new weight

current weight

$W(1) = W(0) + \eta * e * X$ $\rightarrow$ input

$b(1) = b(0) + \eta * e$ $\rightarrow$ error

new bias, current bias, learning rate

[3] Repeat untill convergence

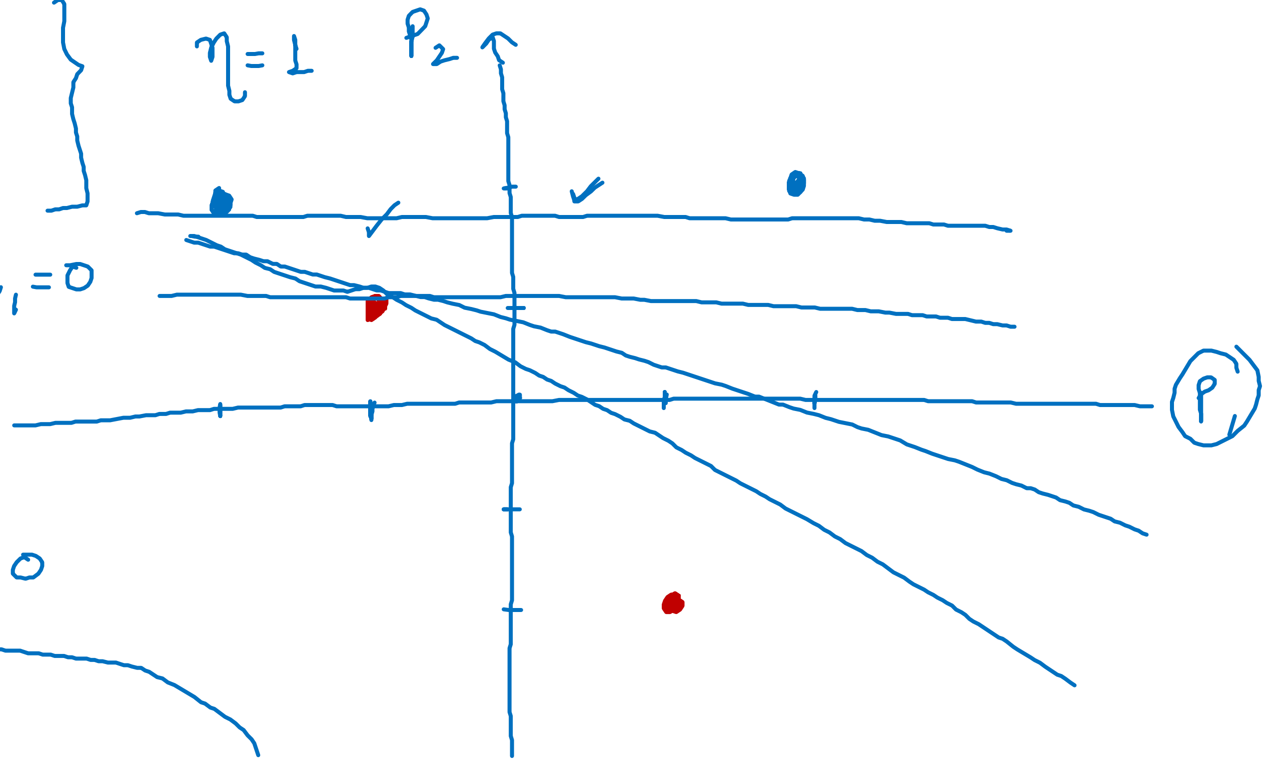
W & b $\rightarrow$ Last & current values are same

Ex →

	p		t
$\to p_1$	2	2	0
p_2	1	−2	1
p_3	−2	2	0
p_4	−1	1	1

$$\left\{ W(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\quad b(0) = 0 \right\}$$

$\eta = 1$



$$p_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad t_1 = 0$$

for p_1

$$a = \text{hardlim}\left(W(0)^T \overset{p_1}{x} + b\right)$$

$$= \text{hardlim}\left(\begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0\right)$$

$$= \text{hardlim}(0)$$

$$= 1$$

$$e = t_1 - a = 0 - 1 = -1$$

update the weight & bias

$$W(1) = W(0) + \eta * e * p_1$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \times (-1) \times \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

$$b(1) = b(0) + \eta * e$$

$$b(1) = -1$$

$$W \boxed{x} + b = 0$$

Epoch

$$\overset{W,\,b}{p_1} \to \overset{W,\,b}{p_2} \to p_3 \to p_4$$

$$a = \text{hardlim}(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

Epoch → 2

1

Perceptron Learning Algo →

① Initialize weights & bias at ~~ron~~ random $w(0)$ $b(0)$

② for each training pair (X, y_{target}) → t

- compute y_a
- compute error (target − actual) or $(y_{target} - y_a)$ or $(t - a) = e$

  if $e = 0$ — No updation ✓
- for updation of weights & bias

$$\left\{ \begin{aligned} W^{new} &= W^{old} + \eta * e * X \\ b^{new} &= b^{old} + \eta * e \end{aligned} \right\} \Rightarrow \begin{aligned} W(new) &= W(old) + \eta * e * X \\ b(new) &= b(old) + \eta * e \end{aligned}$$

η → learning

③ Repeat untill convergence $(e = 0)$

— or last two values is same w & b

Ex → Perceptron Learning Algo

Assume $\eta = 1$

① $\begin{cases} W(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ b(0) = 0 \end{cases}$

$n = 0$

$W^T x + b = 0$

Step 2: Start with $p_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $t_1 = 0$ (target)

$$a = \text{hardlim}\left[W(0)^T p_1 + b(0)\right] = \text{hardlim}\left[\begin{bmatrix} 0 & 0 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0\right]$$

actual $a = \text{hardlim}[0] = 1$

Calculate error $e = t_1 - a = 0 - 1 = -1$

update weight & bias.

$$W(1) = W(0) + \eta * e * p_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

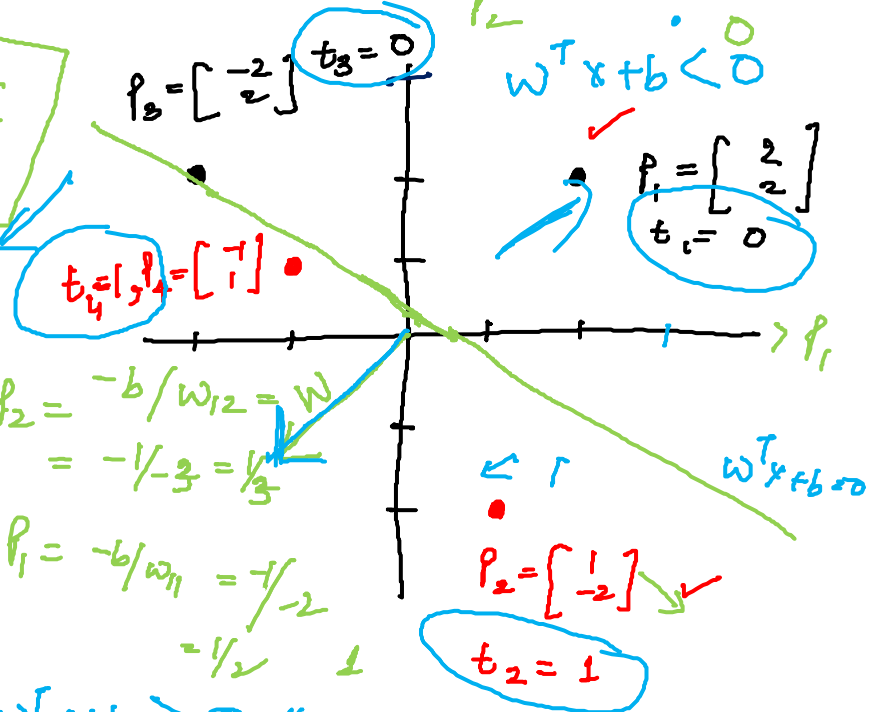
$$b(1) = b(0) + \eta * e = 0 - 1 = -1$$

$W(1) = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ ✓

$b(1) = -1$ ✓

$W = \begin{bmatrix} -2 \\ -3 \end{bmatrix}$

$b = 1$



$p_1 = 0$, $p_2 = -b/w_{12} = -1/-3 = 1/3$

$p_2 = 0$, $p_1 = -b/w_{11} = -1/-2 = 1/2$

$W^T x + b > 0$ ✓

$$a = \text{hardlim}(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

$n = W^T x + b$

$p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow p_4$

$e = 0$

	x_1	x_2	t	a
p_1	2	2	0	1 →
p_2 ✓	1	-2	1	
p_3	-2	2	0	
p_4	-1	1	1	

for $p_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, $t_2 = 1$

$$a = \text{hardlim}[W(1)^T p_2 + b(1)]$$

$$a = \text{hardlim}\left[[-2\ -2]\begin{bmatrix} 1 \\ -2 \end{bmatrix} - 1\right]$$

$$a = \text{hardlim}[1] = 1$$

$e = 0 \Rightarrow$ No update

$$W(2) = W(1) + \eta * e * p_2$$

$$W(2) = W(1) = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

$$b(2) = b(1) + \eta * e$$

$$b(2) = b(1) = -1$$

for $p_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$, $t_3 = 0$

$$a = \text{hardlim}(W(2)^T p_3 + b(3))$$

$$a = \text{hardlim}\left([-2\ -2]\begin{bmatrix} -2 \\ 2 \end{bmatrix} - 1\right)$$

$$a = \text{hardlim}(-1) = 0$$

$$e = t_3 - a = 0 - 0 = 0$$

No update

$$W(3) = W(2) = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

$$b(3) = b(2) = -1$$

for $p_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, $t_4 = 1$

$$a = \text{hardlim}[W(3)^T p_4 + b(3)]$$

$$a = \text{hardlim}(-1) = 0$$

$$e = t_4 - a = 1 - 0 = 1$$

So new weight & ~~bais~~ bias

$$W(4) = W(3) + p_4$$

$$= \begin{bmatrix} -2 \\ -2 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

$$b(4) = b(3) + 1$$

$$= -1 + 1 = 0$$

Step 3 - Repeat

again check the i/p vector p_1

$$a = hl\,(W(4)^T p_1 + b(4)) =$$

$a = 0$, $t_1 = 0$

$$e = t_1 - a = 0 - 0 = 0$$

So No update

$$W(5) = W(4) = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$

$$b(5) = b(4) = 0$$

for p_2, $t_2 = 1$

$a = 0 \Rightarrow$

$e = t_2 - a = 1 - 0 = 1$

$W(6) = W(5) + \eta * e * p_2$

$W(6) = \begin{bmatrix} -2 \\ -3 \end{bmatrix}$

$b(6) = b(5) + \eta * e$

$b(6) = 1$

---

for p_3, $t_3 = 0$

$a = 0$, $e = 0$

No update

$$\boxed{\begin{aligned} W(7) &= W(6) \\ b(7) &= b(6) \end{aligned}}$$

for p_4, $t_4 = 1$

$a = 1$, $e = 0$

No update

$$\boxed{\begin{aligned} W(8) &= W(7) \\ b(8) &= b(7) \end{aligned}}$$

always positive $\rightarrow$ 1

W

$p_1 \longrightarrow p_4$ =

$p_1 \longrightarrow p_4$ = Iteration 1

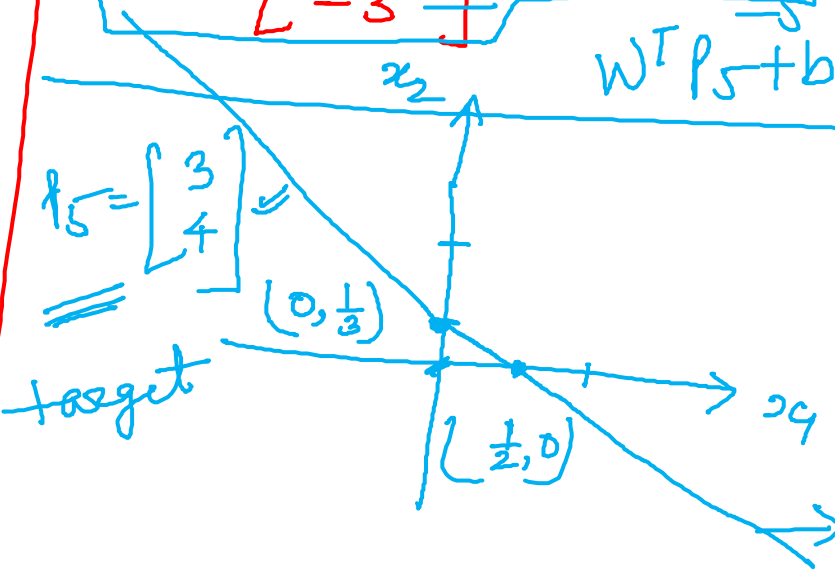
$p_1 \longrightarrow p_4 \rightarrow$ Iter 2

Therefor the algo has converged the final weights & bias are

$$\boxed{W = \begin{bmatrix} -2 \\ -3 \end{bmatrix}, \quad b = 1}$$

$W^T p_5 + b \leq 0$

> 0



$\Rightarrow$ DB $\Rightarrow W^T x + b = 0$

Linear Case

~~$x_1 = 0 \Rightarrow w_{11} x_1 + w_{12} x_2 + b = 0$~~

$w_{11} x_1 + w_{12} x_2 + b = 0$

$x_1 = 0 \Rightarrow w_{12} x_2 + b = 0$

$x_2 = -b / w_{12}$

$(0, -b/w_{12}) \Rightarrow (0, \frac{1}{3})$

$x_2 = 0 \Rightarrow w_{11} x_1 + b = 0$

$x_1 = -b / w_{11}$

$(\frac{-b}{w_{11}}, 0) \Rightarrow (\frac{1}{2}, 0)$

Example:

$$\left\{\mathbf{p}_1=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{t}_1=\begin{bmatrix}0\\0\end{bmatrix}\right\}\left\{\mathbf{p}_2=\begin{bmatrix}1\\2\end{bmatrix},\mathbf{t}_2=\begin{bmatrix}0\\0\end{bmatrix}\right\}\left\{\mathbf{p}_3=\begin{bmatrix}2\\-1\end{bmatrix},\mathbf{t}_3=\begin{bmatrix}0\\1\end{bmatrix}\right\}$$

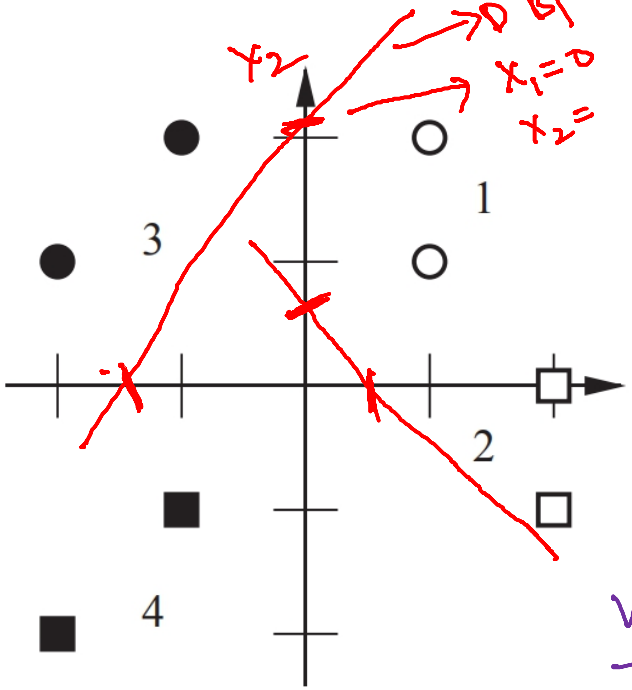
$$\left\{\mathbf{p}_4=\begin{bmatrix}2\\0\end{bmatrix},\mathbf{t}_4=\begin{bmatrix}0\\1\end{bmatrix}\right\}\left\{\mathbf{p}_5=\begin{bmatrix}-1\\2\end{bmatrix},\mathbf{t}_5=\begin{bmatrix}1\\0\end{bmatrix}\right\}\left\{\mathbf{p}_6=\begin{bmatrix}-2\\1\end{bmatrix},\mathbf{t}_6=\begin{bmatrix}1\\0\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_7=\begin{bmatrix}-1\\-1\end{bmatrix},\mathbf{t}_7=\begin{bmatrix}1\\1\end{bmatrix}\right\}\left\{\mathbf{p}_8=\begin{bmatrix}-2\\-2\end{bmatrix},\mathbf{t}_8=\begin{bmatrix}1\\1\end{bmatrix}\right\}.$$

			target		
$p_1 \to$	1	1	0	0	class 1
$p_2 \to$	1	2	0	0	
$p_3 \to$	2	−1	0	1	class 2
$p_4 \to$	2	0	0	1	
$p_5 \to$	−1	2	1	0	class 3
$p_6 \to$	−2	1	1	0	
$p_7 \to$	−1	−1	1	1	class 4
$p_8 \to$	−2	−2	1	1	

$_1W^T x + b = 0$

$x_2 = \frac{b}{_1W^T}$



$_1W = \begin{bmatrix}1\\0\end{bmatrix}$

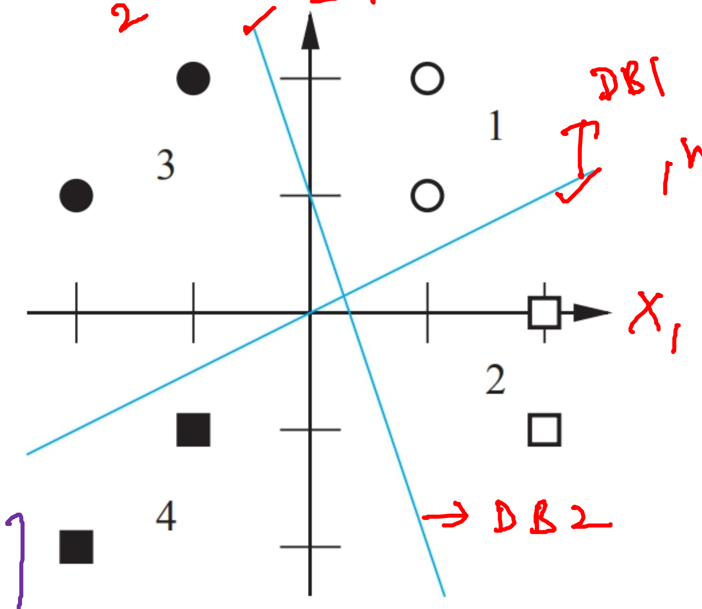
$_2W = \begin{bmatrix}0\\1\end{bmatrix}$

$W = \begin{bmatrix}_1W^T\\ _1W^T\end{bmatrix}$

$$\underline{W} = \begin{array}{cc} _1W & _2W \\ \end{array}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

$b = \begin{bmatrix}1\\-1\end{bmatrix}$ ($_1b$, $_2b$)

$_2W^T x + b = 0$



$\begin{bmatrix}_1W^T\\ _2W^T\end{bmatrix}$

$$\begin{cases} W(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \eta = 1 \\ b(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{cases}$$

first i/p vector, $p_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $t_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$a = \text{hardlim}\,(W^T(0)\, p_1 + b(0)]$$

$$= \text{hardlim}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$e = t_1 - a = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = +\begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$W(1) = W(0) + \eta\, e\, p_1$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + 1 \times \begin{bmatrix} -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$$b(1) = b(0) + \eta e$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

for second i/p vector $p_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $t_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ✓

$$a = \text{hardlim}\,(W(1)^T p_2 + b(1)]$$

$$= \text{hardlim}\left(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} ✓$$

$$e = t_2 - a = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

No update

$$W(2) = W(1) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$$b(2) = b(1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

for 3point $p_3=\begin{bmatrix}2\\-1\end{bmatrix}$, $t_3=\begin{bmatrix}0\\1\end{bmatrix}$

$a=\begin{bmatrix}1\\0\end{bmatrix}$, $e=\begin{bmatrix}-1\\1\end{bmatrix}$

$$W(3)=W(2)+\eta e p_3=\begin{bmatrix}-2&0\\1&-1\end{bmatrix}$$

$$b(3)=b(2)+\eta e=\begin{bmatrix}-1\\1\end{bmatrix}$$

for $p_4=\begin{bmatrix}2\\0\end{bmatrix}$, $t_4=\begin{bmatrix}0\\1\end{bmatrix}$

$$\underset{p_8}{W(8)}=\underset{p_7}{W(7)}=\underset{p_6}{W(6)}=\underset{p_5}{W(5)}=\underset{p_4}{W(4)}=W(3)$$

$$=\begin{bmatrix}-2&0\\1&-1\end{bmatrix}$$

$$b(8)=b(7)=b(6)=b(5)=b(4)=b(3)$$

$$=\begin{bmatrix}-1\\1\end{bmatrix}$$

$p_1 \longrightarrow p_8$

Stoping criteria

① No. of iteration

② After step ① we check current weight & bias values

$$\begin{cases}W(n+1)=W(n)\\ b(n+1)=b(n)\end{cases}$$

$$W(9)=\begin{bmatrix}-2&0\\0&-2\end{bmatrix}$$

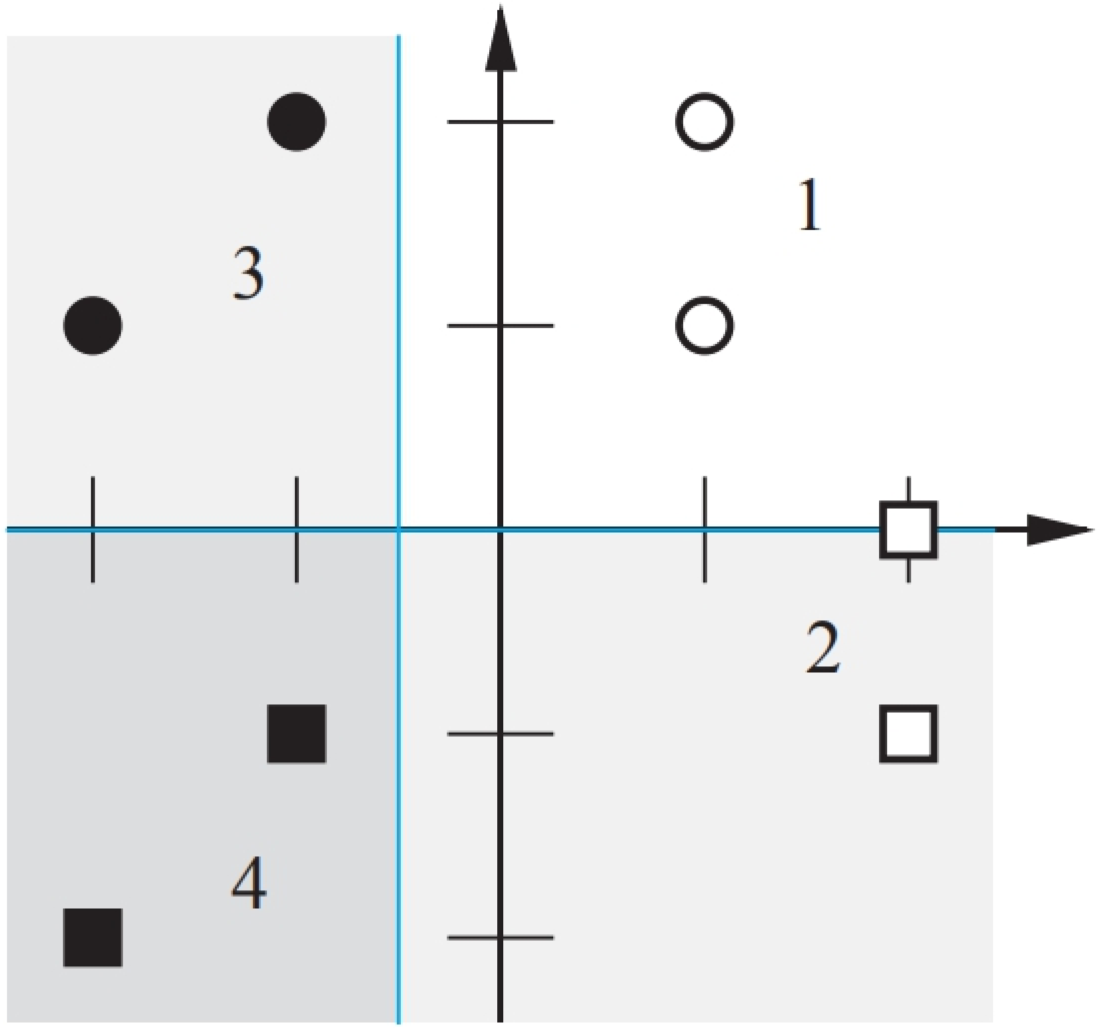
$$b(9)=\begin{bmatrix}-1\\0\end{bmatrix}$$

p_1

for p_2

$$\boxed{\begin{aligned}W(10)&=W(9)\\ b(10)&=b(9)+e\eta=b(9)\end{aligned}}$$

At this point the algo has converged,


1
2
3
4