آرین شکر - تکلیف ۱ درس شبکه‌ها عصبی - ۴۰۰۲۳۴۹۴

Magan E4.8

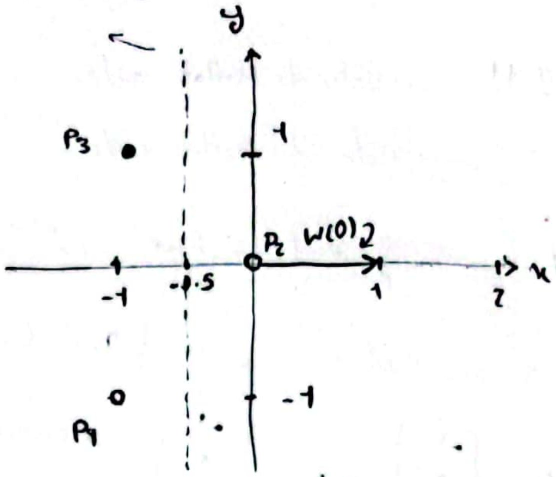
$\{p_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_1 = 0\}, \{p_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_2 = 0\}, \{p_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_3 = 1\}$

$w(0) = [1 \quad 0]$, $b(0) = 0.5$ initial decision boundary

i. decision boundary:

$[1 \quad 0] \begin{bmatrix} x \\ y \end{bmatrix} + 0.5 = 0$

$\Rightarrow x = -0.5$



only (P_1) is correctly classified.

ii. perceptron Learning Rule:

$w^{new} = w^{old} + ep^T$

$b^{new} = b^{old} + e$

where $e = t - a$

P_1 : $a = \text{hardlim}([1 \quad 0]\begin{bmatrix} -1 \\ -1 \end{bmatrix} + 0.5)$

$= 0$, $t_1 = 0 \Rightarrow e = 0$

$\Rightarrow w^{new} = w^{old}$, $b^{new} = b^{old}$

P_2 : $a = \text{hardlim}([1 \quad 0]\begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.5)$

$= 1$, $t_2 = 0 \Rightarrow e = -1$

$\Rightarrow w^{new} = [1 \quad 0] + (-1)[0 \quad 0] = [1 \quad 0]$

$b^{new} = 0.5 + (-1) = -0.5$

P_3 : $a = \text{hardlim}([1 \quad 0]\begin{bmatrix} -1 \\ 1 \end{bmatrix} - 0.5) = 0$

$t_3 = 1 \Rightarrow e = 1$

$\Rightarrow w^{new} = [1 \quad 0] + [-1 \quad 1] = [0 \quad 1]$

$b^{new} = -0.5 + 1 = 0.5$

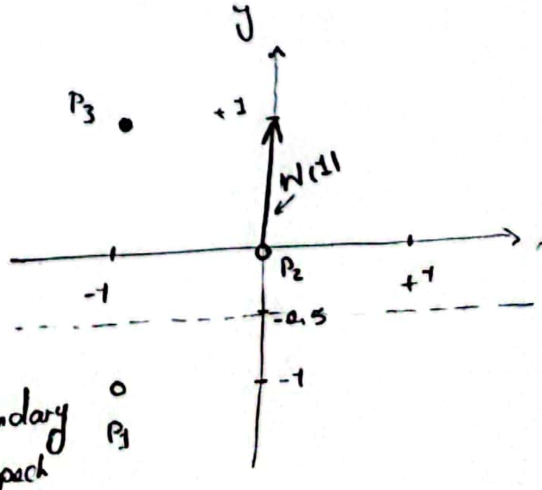
iii. $w = [0 \quad 1]$, $b = 0.5$

decision boundary :

$[0 \quad 1]\begin{bmatrix} x \\ y \end{bmatrix} + 0.5 = 0$

$\Rightarrow y = -0.5$

Only (P_3) is correctly classified



decision boundary after one epoch

iv. Yes according to Novikoff's proof of convergence and taking into account the fact that this classification problem has linear solution, any non-zero weight vector should converge to a valid solution after enough iterations.

E 4.11 $\longrightarrow$ Refer to Matlab code.

E 4.12 $\longrightarrow$ Refer to Matlab code.

E 7.3 using $\alpha = 1$ we have; $w^{new} = w^{old} + t_q p_q^T$, assuming zero initial weight vector

$\longrightarrow w = TP^t$, $w = [0, 0, 0, 0, 0, 0]$

$$P_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} / \sqrt{6}, \quad P_2 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} / \sqrt{6}$$

assume $t_1 = -1$, $t_2 = +1$ (because network returns a 1×1 output)

$$\rightarrow w = -1 \times \left[\frac{-1}{\sqrt{6}} \frac{1}{\sqrt{6}} \frac{-1}{\sqrt{6}} \frac{1}{\sqrt{6}} \frac{-1}{\sqrt{6}} \frac{1}{\sqrt{6}}\right] + 1 \times \left[\frac{-1}{\sqrt{6}} \frac{1}{\sqrt{6}} \frac{1}{\sqrt{6}} \frac{-1}{\sqrt{6}} \frac{-1}{\sqrt{6}} \frac{-1}{\sqrt{6}}\right]$$

$$= \left[0 \; 0 \; \frac{2}{\sqrt{6}} \; \frac{-2}{\sqrt{6}} \; 0 \; \frac{-2}{\sqrt{6}}\right]$$

E 7.6. i. Assuming Symmetric Hard Limit activation we have to have: (no bias)

P_1: $hardlims(wp_1^T) = hardlims(w_1 \times 1 + w_2 \times 0) = 1 \Rightarrow w_1 \geq 0$

P_2: $hardlims(wp_2^T) = hardlims(w_1 + w_2) = -1 \Rightarrow w_1 + w_2 < 0$

P_3: $hardlims(wp_3^T) = hardlims(w_2) = 1 \Rightarrow w_2 \geq 0$

+

$w_1 + w_2 \geq 0$ contradiction

network has to be biased

ii. we now augment the input vectors with a bias term and form the p matrix.

$$P = [P_1^+ \quad P_2^+ \quad P_3^+] = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P^+ = (P^T P)^{-1} P^T = \left(\begin{bmatrix} 2 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 2 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix}$$

now we can calculate the weights and biases :

$$W = TP^+ = [1 \quad -1 \quad 1] \begin{bmatrix} 0 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} = [-2 \quad -2 \quad 3] \rightarrow W = [-2 \quad -2], \; b = 3$$

we will now verify that the network is transforming the input patterns as intended:

P_1: $\text{hardlims}\left([-2 \quad -2]\begin{bmatrix} -1 \\ 0 \end{bmatrix} + 3\right) = 1$

P_2: $\text{hardlims}\left([-2 \quad -2]\begin{bmatrix} 1 \\ 1 \end{bmatrix} + 3\right) = -1$ → all patterns correctly classified

P_3: $\text{hardlims}\left([-2 \quad -2]\begin{bmatrix} 0 \\ 1 \end{bmatrix} + 3\right) = 1$

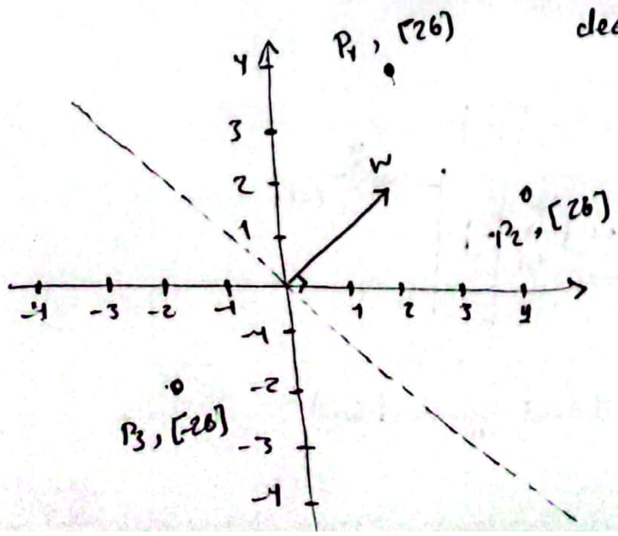
E 7.7. i. if we knew for certain that the network could only produce $-26, 26$ as output we could $26 \times \text{hardlims}(n)$ as activation but we will just go with purelin for now:

normalized points: $P_1 = \begin{bmatrix} 0.4472 \\ 0.8944 \end{bmatrix}$, $P_2 = \begin{bmatrix} 0.8944 \\ 0.4472 \end{bmatrix}$, $P_3 = \begin{bmatrix} -0.7071 \\ -0.7071 \end{bmatrix}$

$$W^{hebb} = TP^T = [26 \quad 26 \quad -26] \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & 0.4472 \\ -0.7071 & -0.7071 \end{bmatrix} = [53.2674 \quad 53.2674]$$

decision boundary: $Wp = 0 \rightarrow y = -x$

ii



Scanned with CamScanner

iii. $p = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 2 & -2 \end{bmatrix} \rightarrow p^+ = \begin{bmatrix} 2 & 4 \\ 4 & 2 \\ -2 & -2 \end{bmatrix} \left(\begin{bmatrix} 24 & 20 \\ 20 & 24 \end{bmatrix} \right)^{-1} = \begin{bmatrix} -0.1828 & 0.3182 \\ 0.3182 & -0.1818 \\ -0.0455 & -0.0455 \end{bmatrix}$

$w^P = Tp^+ = [26 \;\; 26 \;\; -26]\, p^+ = [4.7273 \;\; 4.7273]$

iv. w^P produces the exact same decision boundary except with a smaller weight vector.

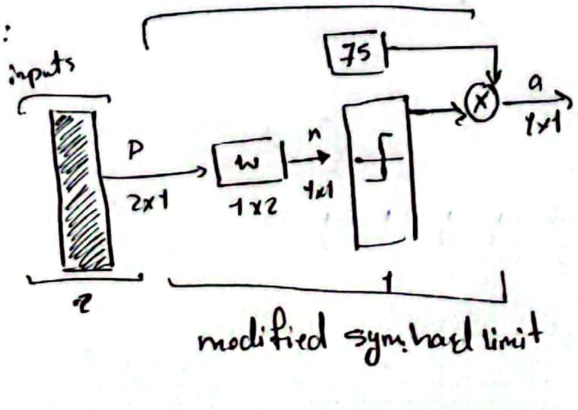
v. If we consider 26 and -26 as 1 and -1, both classifiers perform well and classify all of the inputs correctly. However that's assumming a hardlims transfer function. Since we are using purelin we can see which method performs better using MSE.

$$MSE_{Hebb}: \begin{cases} P_1: w^h p_{1n}^T = 71.4658 \rightarrow \text{squared error} \simeq 206.71 \\ P_2: w^h p_{2n}^T = 71.4858 \rightarrow \text{squared error} \simeq 206.71 \\ P_3: w^h p_{3n}^T = -75.3315 \rightarrow \text{squared error} \simeq 243.36 \end{cases} \rightarrow MSE \simeq 218.92$$

$$MSE_{pinv}: \begin{cases} P_1: w^P p_1^T = 28.3638 \rightarrow \text{squared error} \simeq 5.5876 \\ P_2: w^P p_2^T = 28.3638 \rightarrow \text{squared error} \simeq 5.5876 \\ P_3: w^P p_3^T = -18.9092 \rightarrow \text{squared error} \simeq 50.2794 \end{cases} \rightarrow MSE \simeq 20.4849$$

→ we can see that the pinv rule produces a much better MSE as expected.

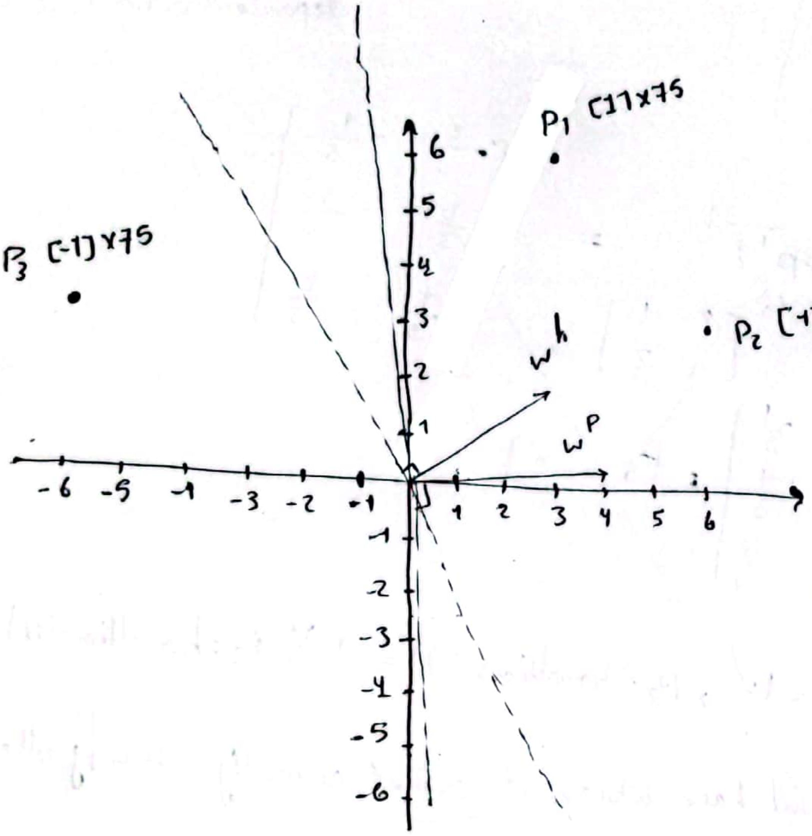
E7.9. Since this problem is similar to the one before we will try a different approach this time. If we assume [75, -75] are the only possible outputs, we can replace purelin(n) with 75 × hardlims(n) and train the network on it. The diagram for this network would look like below:



ii. normalized points: $p_1 = \begin{bmatrix} 0.4472 \\ 0.8944 \end{bmatrix}$, $p_2 = \begin{bmatrix} 0.8944 \\ 0.4472 \end{bmatrix}$, $p_3 = \begin{bmatrix} -0.8944 \\ 0.4472 \end{bmatrix}$

$$w^{hebb} = TP^T = [1 \quad 1 \quad -1] \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & 0.4472 \\ -0.8944 & 0.4472 \end{bmatrix} = [2.2361 \quad 0.8944]$$

iii.



decision boundary:

$$2.2361x + 0.8944y = 0$$

$$\Rightarrow y = -\frac{2.2361}{0.8944}x \approx 2.5x$$

→ All patterns classified correctly

iv. $P = \begin{bmatrix} 3 & 6 & -6 \\ 6 & 3 & 3 \end{bmatrix} \rightarrow P^+ = \begin{bmatrix} 3 & 6 \\ 6 & 3 \\ -6 & 3 \end{bmatrix} \left(\begin{bmatrix} 81 & 18 \\ 18 & 54 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 0.0133 & 0.1067 \\ 0.0667 & 0.0333 \\ -0.0933 & 0.0867 \end{bmatrix}$

$$w^P = TP^+ = [1 \quad 1 \quad -1]P^+ = [0.1733 \quad 0.0533]$$

If we treat labels [1, -1] as continuous values, w^P performs better than w^h in MSE.

From the diagram it also seems that w^P seeks to maximize the distance between the two classes and divides the space more fairly w.r.t. to the data points.

E.7.11 ⟶ Refer to the MATLAB code.