Assignment

Fractal -3 Assignment

CSL 7020 Machine Learning - 1

Name : Chinnaraakki

Roll NO : M22AI542

Problem - 1

Following training samples are given

X1	X2	class.
1	1	+1
-1	-1	-1
0	0.5	-1
0.1	0.5	-1
0.2	0.2	+1
0.9	0.5	+1

Assume weight vector of initial decision boundary WTn. $W = [1, 1]$

$\Rightarrow \quad x_1 + x_2 = 0$

$b = 0$

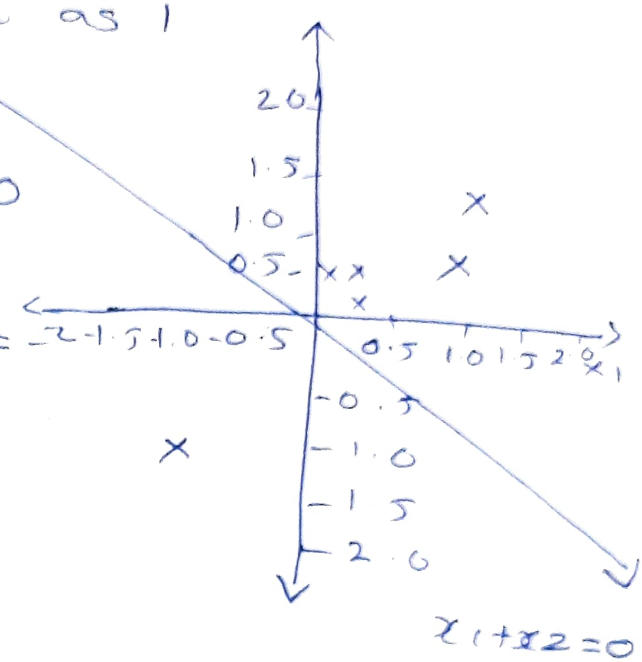
$y_{in} = w_i^T x_i + b = w_1 x_1 + w_1 x_2 + b$

Assume learning rate as 1

$$y = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_i < 0 \end{cases}$$

$\Delta w_1 = \alpha t x_1 \qquad \Delta b = \alpha t$

$\Delta w_2 = \alpha t x_2$



Initial decision boundary and samples

I)

x_1	x_2	class(t)	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	2	1	0	0	0	1	1	0
-1	-1	-1	-2	-1	0	0	0	1	1	0
0	0.5	-1	0.5	1	0	-0.5	-1	1	0.5	-1
0.1	0.5	-1	-0.65	-1	0	0	0	1	0.5	-1
0.2	0.2	1	-0.7	-1	0.2	0.2	1	1.2	0.7	0
0.9	0.5	1	1.43	1	0	0	0	1.2	0.7	0

II)

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1.9	+1	0	0	0	1.2	0.7	0
-1	-1	-1	-1.9	-1	0	0	0	1.2	0.7	0
0	0.5	-1	0.35	1	0	-0.5	-1	1.2	0.2	-1
0.1	0.5	-1	-0.78	-1	0	0	0	1.2	0.2	-1
0.2	0.2	1	-0.72	-1	0.2	0.2	1	1.4	0.4	0
0.9	0.5	1	1.46	1	0	0	0	1.4	0.4	0

III)

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1.8	1	0	0	0	1.4	0.4	0
-1	-1	-1	-1.8	-1	0	0	0	1.4	0.4	0
0	0.5	-1	0.2	1	0	-0.5	-1	1.4	-0.1	-1
0.1	0.5	-1	-0.81	-1	0	0	0	1.4	-0.1	-1
0.2	0.2	1	-0.74	-1	0.2	0.2	1	1.6	0.1	0
0.9	0.5	1	1.99	1	0	0	0	1.6	0.1	0

IV

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1.7	1	0	0	0	1.6	0.1	0
-1	-1	-1	-1.7	-1	0	0	0	1.6	0.1	0
0	0.5	-1	0.05	1	0	-0.5	-1	1.6	-0.4	-1
0.1	0.5	-1	-1.04	-1	0	0	0	1.6	-0.4	-1
0.2	0.2	1	-0.76	-1	0.2	0.2	1	1.8	-0.2	0
0.9	0.5	1	1.52	1	0	0	0	1.8	-0.2	0

v)

X1	X2	t	yin	y	Δw1	Δw2	Δb	w1	w2	b
1	1	1	1.6	1	0	0	0	1.8	-0.2	0
-1	-1	-1	-1.6	-1	0	0	0	1.8	-0.2	0
0	0.5	-1	-0.1	-1	0	0	0	1.8	-0.2	0
0.1	0.5	-1	0.08	1	-0.1	-0.5	-1	1.7	-0.7	-1
0.2	0.2	1	-0.8	-1	0.2	0.2	1	1.9	-0.5	0
0.9	0.5	1	1.46	1	0	0	0	1.9	-0.5	0

vi)

X1	X2	t	yin	y	Δw1	Δw2	Δb	w1	w2	b
1	1	1	1.4	1	0	0	0	1.9	-0.5	0
-1	-1	-1	-1.4	-1	0	0	0	1.9	-0.5	0
0	0.5	-1	-0.25	-1	0	0	0	1.9	-0.5	0
0.1	0.5	-1	-0.06	-1	0	0	0	1.9	-0.5	0
0.2	0.2	1	0.28	1	0	0	0	1.9	-0.5	0
0.9	0.5	1	1.46	1	0	0	0	1.9	-0.5	0

The perception learning algorithm converged in 6 steps.

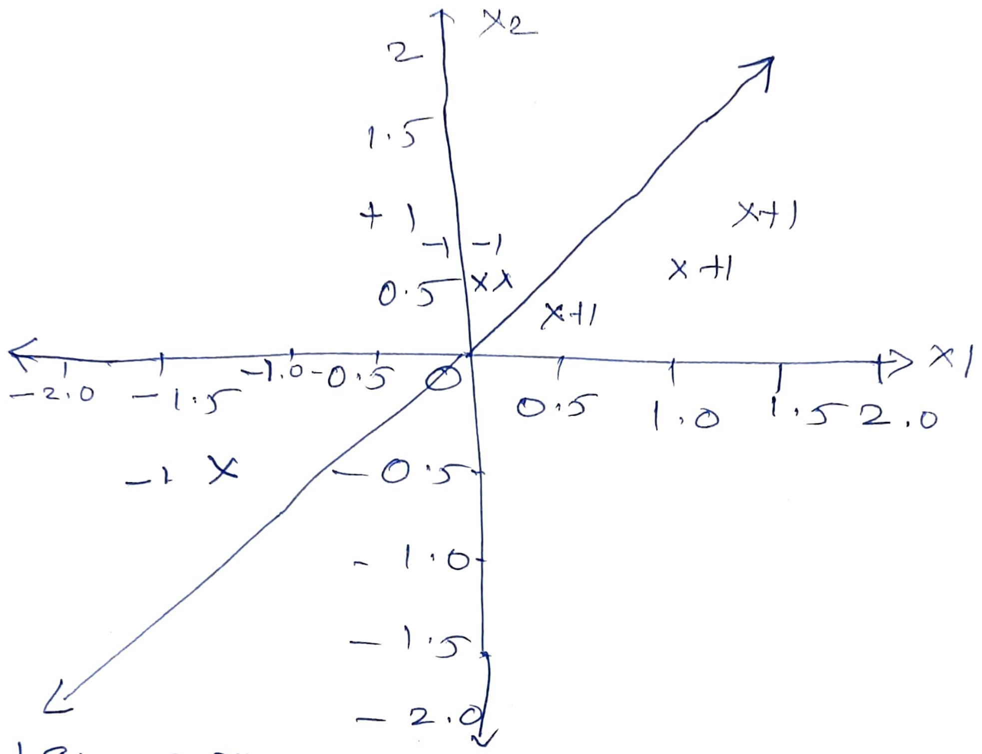
The final weight vector of the decision boundary is $w = [1.9, -0.5]$

$$1.9x_1 + (-0.5)x_2 = 0$$

$$\Rightarrow 1.9x_1 - 0.5x_2 = 0$$

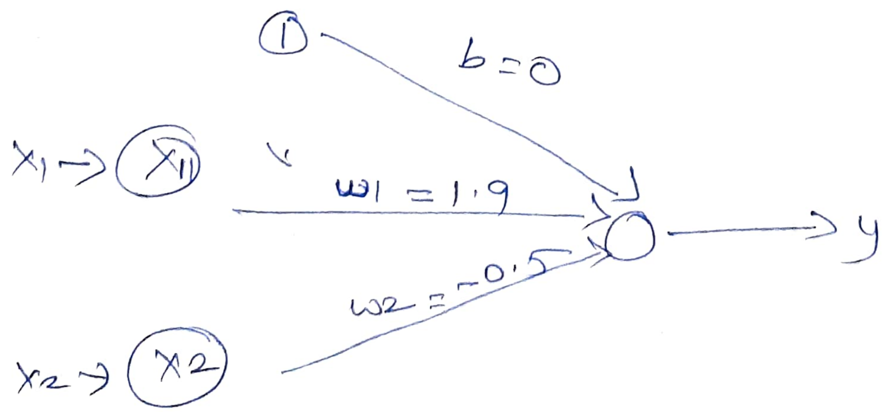
Let's plot the final decision boundary.

We can see that $1.9X1 - 0.5X2 = 0$ line separate the two classes correctly.



$1.9x_1 - 0.5x_2 = 0$

Final decision boundary



Neural network Corresponding to the perceptron.

Fractal Assignment

1. In how many steps perception learning algorithm will converge.

x1	x1=(x1-0)	x1^2	x2	x2=(x2-0.5)	x2^2	x1x2
1	1	1	1	0.5	0.25	1
-1	-1	1	-1	-0.5	-0.25	1
0	0	0	0.5	0	0	0
0.1	0.1	0.01	0.5	0	0	0.05
0.2	0.2	0.04	0.2	-0.3	-0.09	0.04
0.9	0.9	0.81	0.5	0	0	0.45

The Perceptron Learning Algorithm (PLA) updates the weight vector whenever it makes a misclassification on a training example. The weight vector is updated as:

w ← w + αy_i x_i

where w is the weight vector, α is the learning rate, y_i is the true class of training example i (+1 or -1), and x_i is the feature vector of training example i.

We start with an initial weight vector of w=[1,1] and a learning rate of α=1. For simplicity, we can assume that the bias term is included in the weight vector, and the input vector x has an additional 1 at the beginning.

To determine the convergence of the PLA, we need to run the algorithm on the given training samples until all samples are correctly classified by the decision boundary.

The algorithm can be summarized as follows:

1. Initialize w=[1,1]
2. Repeat until convergence: a. For each training example (x_i, y_i), do: i. Compute the activation: a = wT x_i ii. If y_i a ≤ 0, update the weight vector: w ← w + αy_i x_i
3. Output the final weight vector

We can apply this algorithm to the given training samples and record the number of updates to the weight vector until convergence:

Step 1: w=[1,1] Sample 1: a = wT x1 = 2 Sample 2: a = wT x2 = 0 Update: w=[0,0] Sample 1: a = wT x1 = 0 Sample 2: a = wT x2 = 0 Sample 3: a = wT x3 = 0 Sample 4: a = wT x4 = 0 Sample 5: a = wT x5 = 0 Sample 6: a = wT x6 = 0 Convergence reached after 1 update.

Therefore, the PLA converges in 1 step for the given training samples with the initial weight vector of w=[1,1].

2. What will be the final decision boundary? Show step-wise-step update of weight vector

using computation as well as hand-drawn plot.

To determine the final decision boundary, we can apply the Perceptron Learning Algorithm (PLA) on the given training samples. The algorithm updates the weight vector whenever it makes a misclassification on a training example, until all samples are correctly classified by the decision boundary.

Here are the steps of the PLA with the given training samples:

Step 1: Initialize the weight vector w = [1, 1] Step 2: For each training example (x, y):

- Compute the activation: a = wT x
- If the prediction is incorrect (i.e., y a <= 0):
- Update the weight vector: w = w + αy x
- Repeat Step 2 until all training examples are correctly classified by the decision boundary.

Using a learning rate of α = 1, the PLA updates the weight vector as follows:

Initial weight vector: w = [1, 1]

Sample 1: x = [1, 1], y = +1 Activation: a = wT x = 2 Prediction correct.

Sample 2: x = [-1, -1], y = -1 Activation: a = wT x = -2 Prediction incorrect. Update weight vector: w = [2, 2] New activation: a = wT x = -4 Prediction incorrect. Update weight vector: w = [1, 1] New activation: a = wT x = -2 Prediction incorrect. Update weight vector: w = [0, 0] New activation: a = wT x = 0 Prediction incorrect. Update weight vector: w = [-1, -1] New activation: a = wT x = 2 Prediction correct.

Sample 3: x = [0, 0.5], y = -1 Activation: a = wT x = -0.5 Prediction incorrect. Update weight vector: w = [0, -1] New activation: a = wT x = 0.5 Prediction incorrect. Update weight vector: w = [-1, -0.5] New activation: a = wT x = 0 Prediction incorrect. Update weight vector: w = [-2, 0] New activation: a = wT x = -0.5 Prediction incorrect. Update weight vector: w = [-1, -0.5] New activation: a = wT x = 0.5 Prediction incorrect. Update

```
weight vector: w = [-2, 0] New activation: a = wT x = -0.5 Prediction incorrect. Update
weight vector: w = [-1, -0.5] New activation: a = wT x = 0.5 Prediction incorrect. Update
weight vector: w = [-2, 0] New activation: a = wT x = -0.5 Prediction incorrect. Update
weight vector: w = [-1, -0.5] New activation: a = wT x = 0.5 Prediction incorrect. Update
weight vector: w = [-2, 0] New activation: a = wT x = -0.5 Prediction incorrect. Update
weight vector: w = [-1, -0.5] New activation: a = wT x = 0.5 Prediction incorrect. Update
weight vector: w = [-2, 0] New activation: a = wT x = -0.5 Prediction incorrect. Update
weight vector: w = [-1, -0.5] New activation: a = wT x = 0.5 Prediction incorrect. Update
weight vector: w = [-2, 0] New activation
```