Fractal Assignment

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

x1	x1=(x1-0)	x1^2	x2	x2=(x 2-0.5)	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 makesa 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 theinput vector x has an additional 1 at the beginning.

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

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

We can apply this algorithm to the given training samples and record the number ofupdates 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 1update.

Therefore, the PLA converges in 1 step for the given training samples with the initialweight 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 vectorwhenever 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 decisionboundary.

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. Updateweight vector: w = [1, 1] New activation: a = wT x = -2 Prediction incorrect. Updateweight 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