Name: Jaydeep Solanki

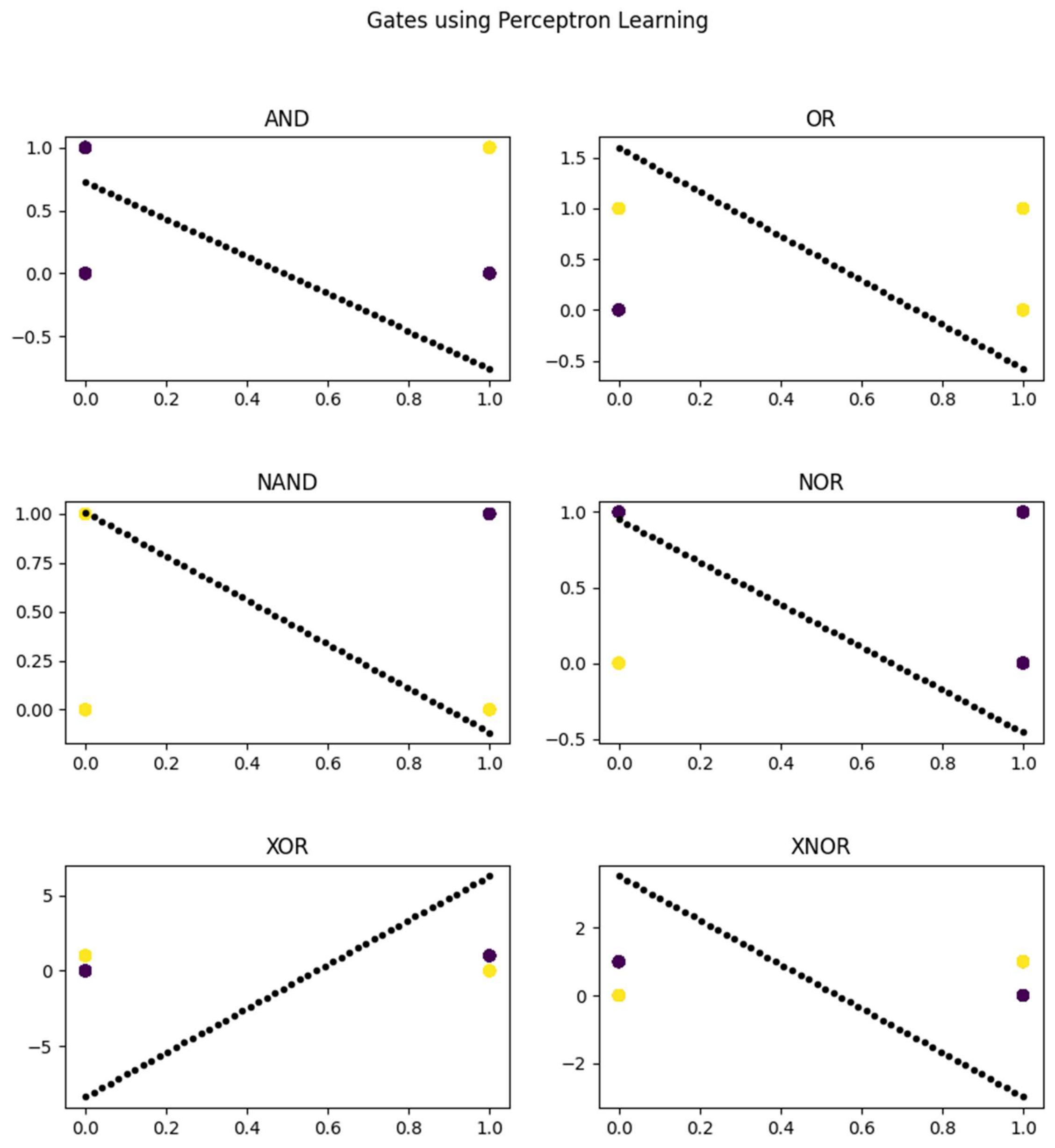
Roll: 22BEC059

Practical 9: Write a program that implements AND gate using perceptron learning algorithm

⇨ **Without using inbuilt module**

```
import numpy as np
from matplotlib import pyplot as plt

current = ""


def plotGates(X, Y):
    global current
    Y = np.array([0, 1, 1, 0])
    plt.scatter(x=X[:, 0], y=[X[:, 1]], c=Y)
    # plt.show()
    n_samples = X.shape[0]
    n_features = X.shape[1]
    w = np.random.uniform(0, 1, size=n_features)
    b = np.random.uniform(0, 1, 1)
    n_epoch = int(input(f"Enter the number of epochs for {current} Gate: "))
    lr = 0.01
    for e in range(n_epoch):
        for s in range(n_samples):
            net = np.dot(X[s, :], w) + b
            if net >= 0:
                a = 1
            else:
                a = 0
            error = Y[s] - a
            w = w + lr * error * X[s, :]
            b = b + lr * error
    m = -w[0] / w[1]
    c = -b / w[1]
    return m, c


def plot_decision_boundary(X):
    global current
    plotDict = {"AND": [0, 0, 0, 1], "OR": [0, 1, 1, 1], "NAND": [1, 1, 1, 0],
"NOR": [1, 0, 0, 0], "XOR": [0, 1, 1, 0],
                "XNOR": [1, 0, 0, 1]}
    fig, axes = plt.subplots(3, 2, figsize=(10, 10))
    fig.subplots_adjust(hspace=0.5)
    hmm = 0
    for i in plotDict:
        current = i
        m, c = plotGates(X, plotDict[i])
        for x in np.linspace(np.min(X[:, 0]), np.max(X[:, 0])):
            y = m * x + c
            ax = axes[hmm // 2, hmm % 2]
            ax.plot(x, y, linestyle="-", color="k", marker=".")
            ax.scatter(X[:, 0], X[:, 1], c=plotDict[i])
            ax.set_title(i)
        hmm += 1
    plt.suptitle("Gates using Perceptron Learning")
    plt.savefig("Gates using Perceptron Learning")
    plt.show()


X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

plot_decision_boundary(X)
```

⇨ Output:


Gates using Perceptron Learning
AND
OR
NAND
NOR
XOR
XNOR
0.0
0.2
0.4
0.6
0.8
1.0