**Assignment 2**

**Experiment 4 and 5**

**Title: ANN Learning Rules**

Name of Student: Sangeet Agrawal PRN No. 21070122140

DoP4: 5 Aug

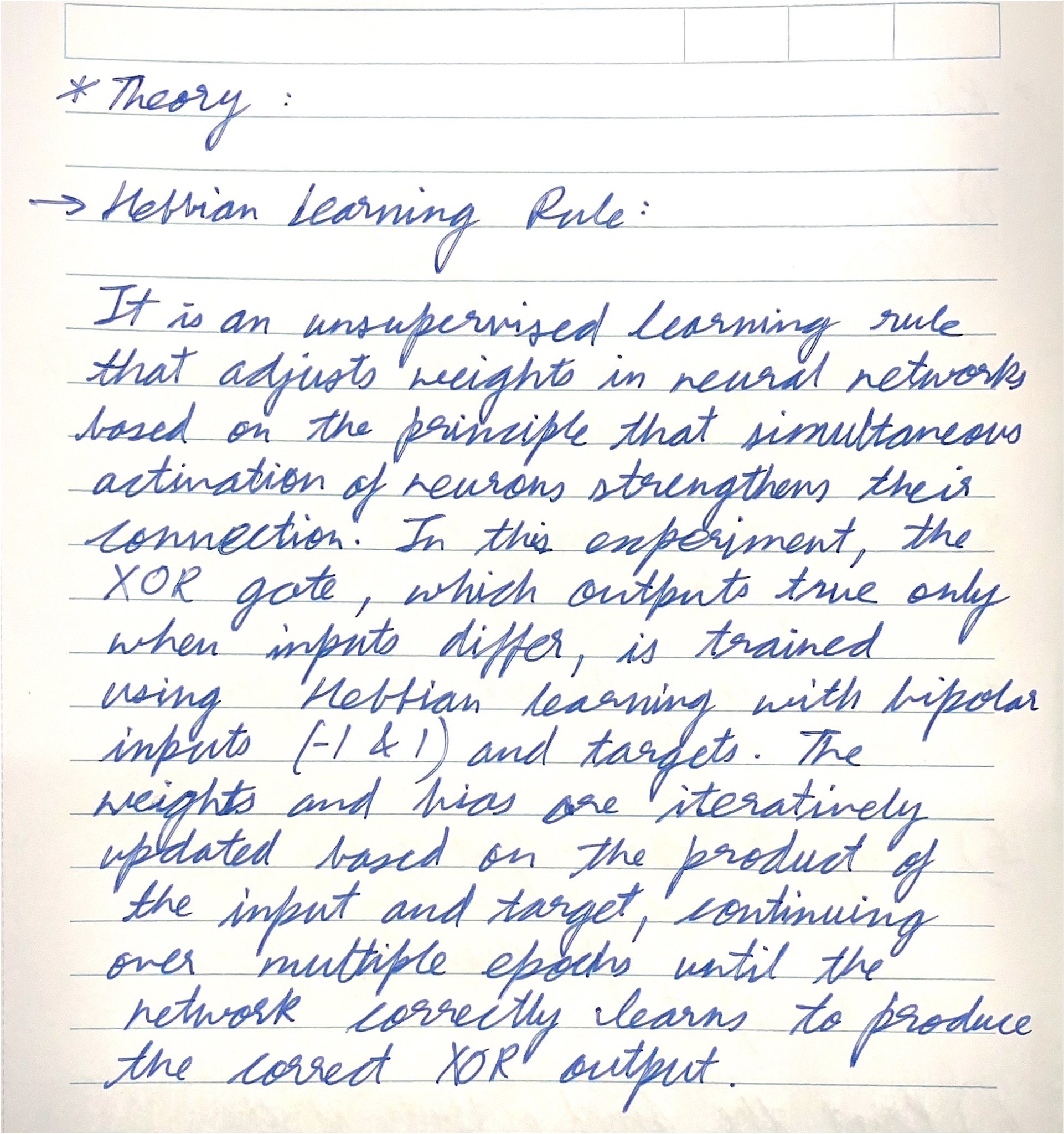
DoP5: 12 Aug DoS: 19 Aug

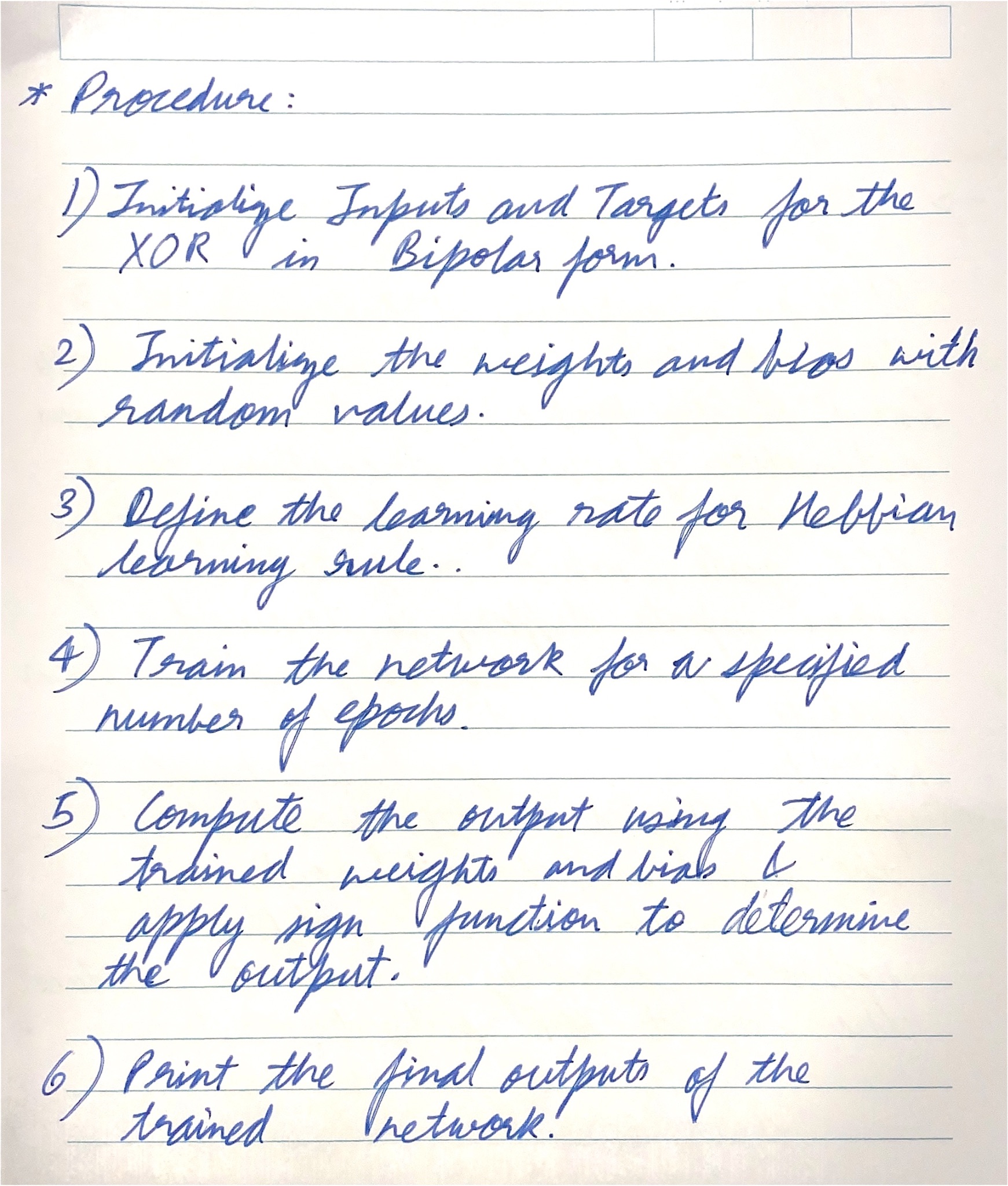
**Title:** ANN Learning Rules

**Aim:** Implement a Program to Train XOR Gate Using Hebbian Learning with

Bipolar Input and Targets

**Objective:** Students will learn and implement

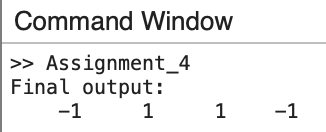
* Unsupervised Learning model
* Hebbian learning 

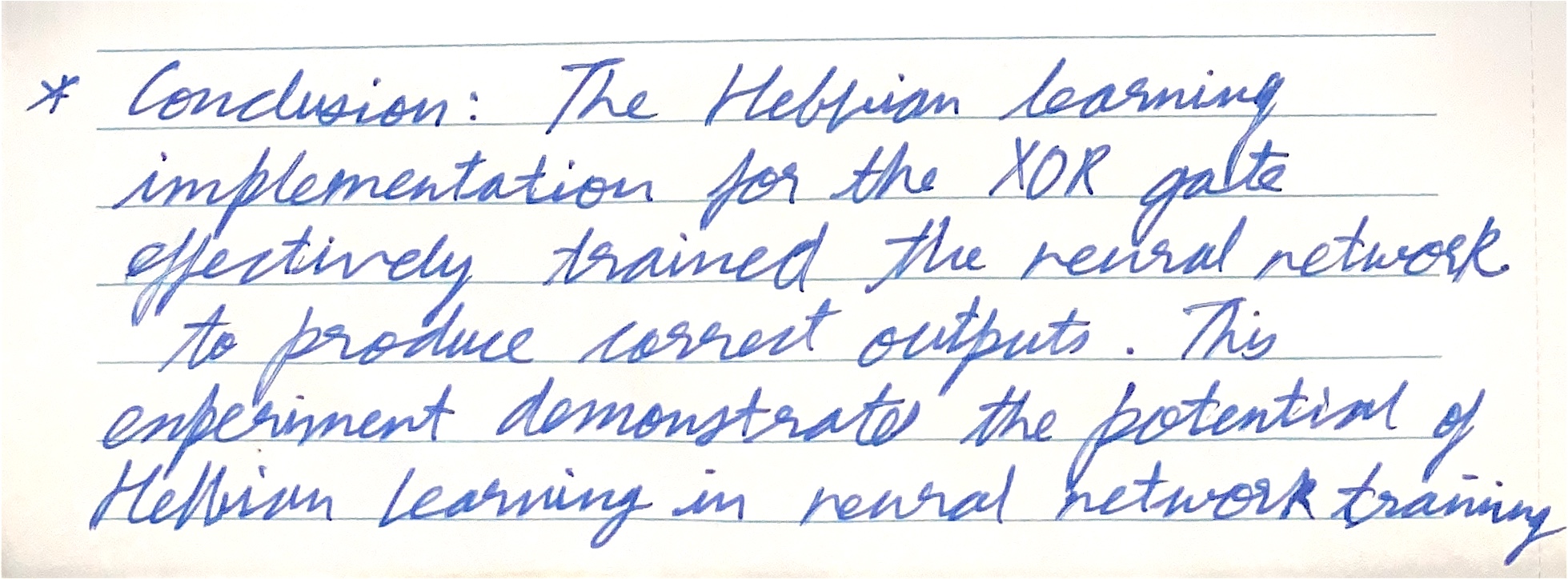


**Code:**

| inputs = [-1 -1; -1 1; 1 -1; 1 1]; targets = [-1; 1; 1; -1];  weights = randn(2, 1); bias = randn();  learning\_rate = 0.1;  % Hebbian learning rule implementation epochs = 100; for epoch = 1:epochs  for i = 1:size(inputs, 1)  x = inputs(i, :)';  t = targets(i);  y = dot(weights, x) + bias;  weights = weights + learning\_rate \* x \* t;  bias = bias + learning\_rate \* t;  end end % Compute the output outputs = zeros(size(targets)); for i = 1:size(inputs, 1)  y = dot(weights, inputs(i, :)') + bias;  outputs(i) = sign(y); end  disp('Final output:'); disp(targets'); |
| --- |

**Output:**





**Experiment 5**

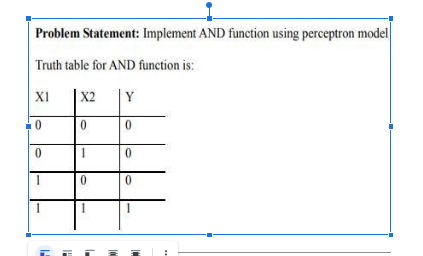
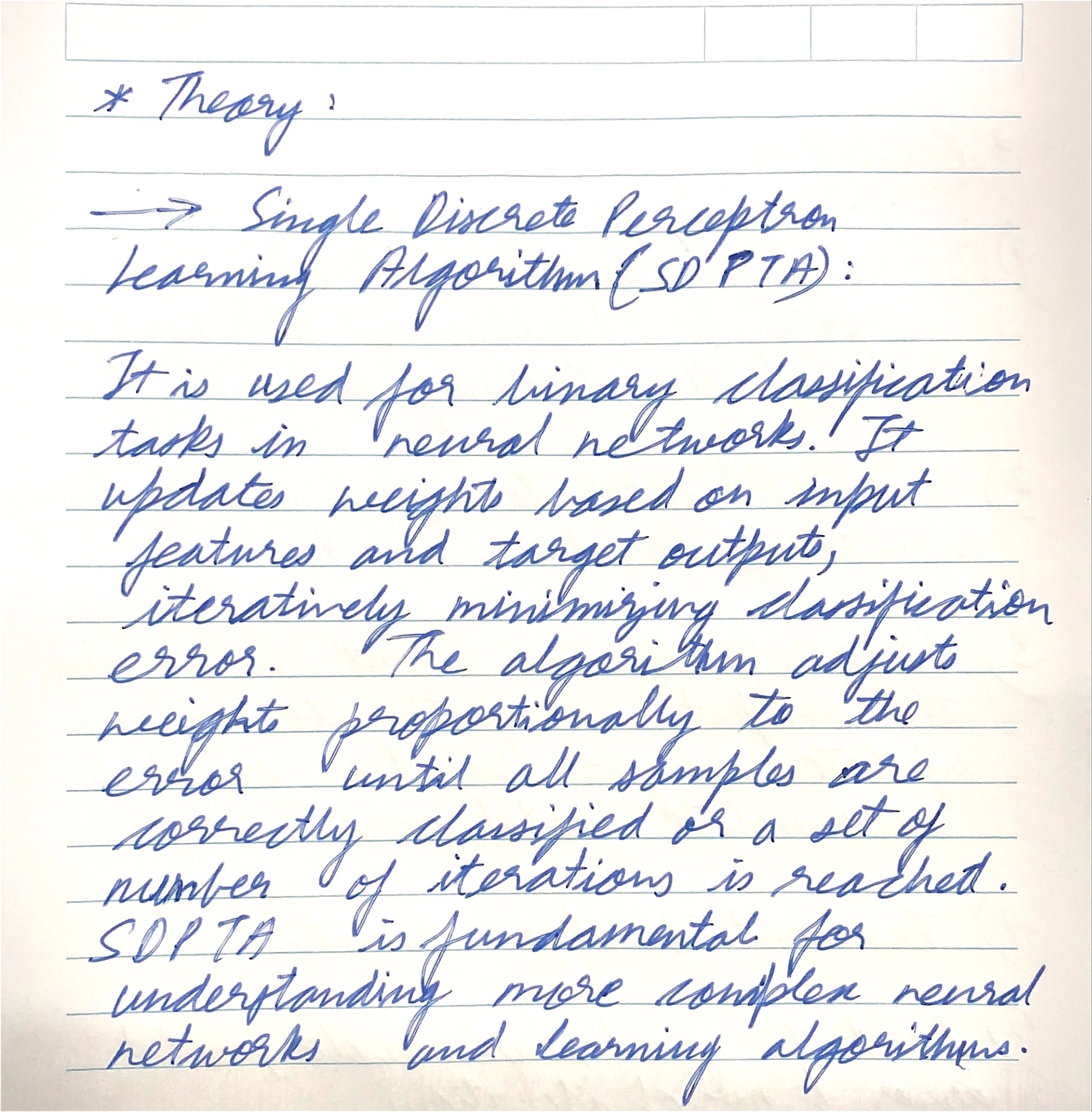
**Title:** ANN Learning Rules

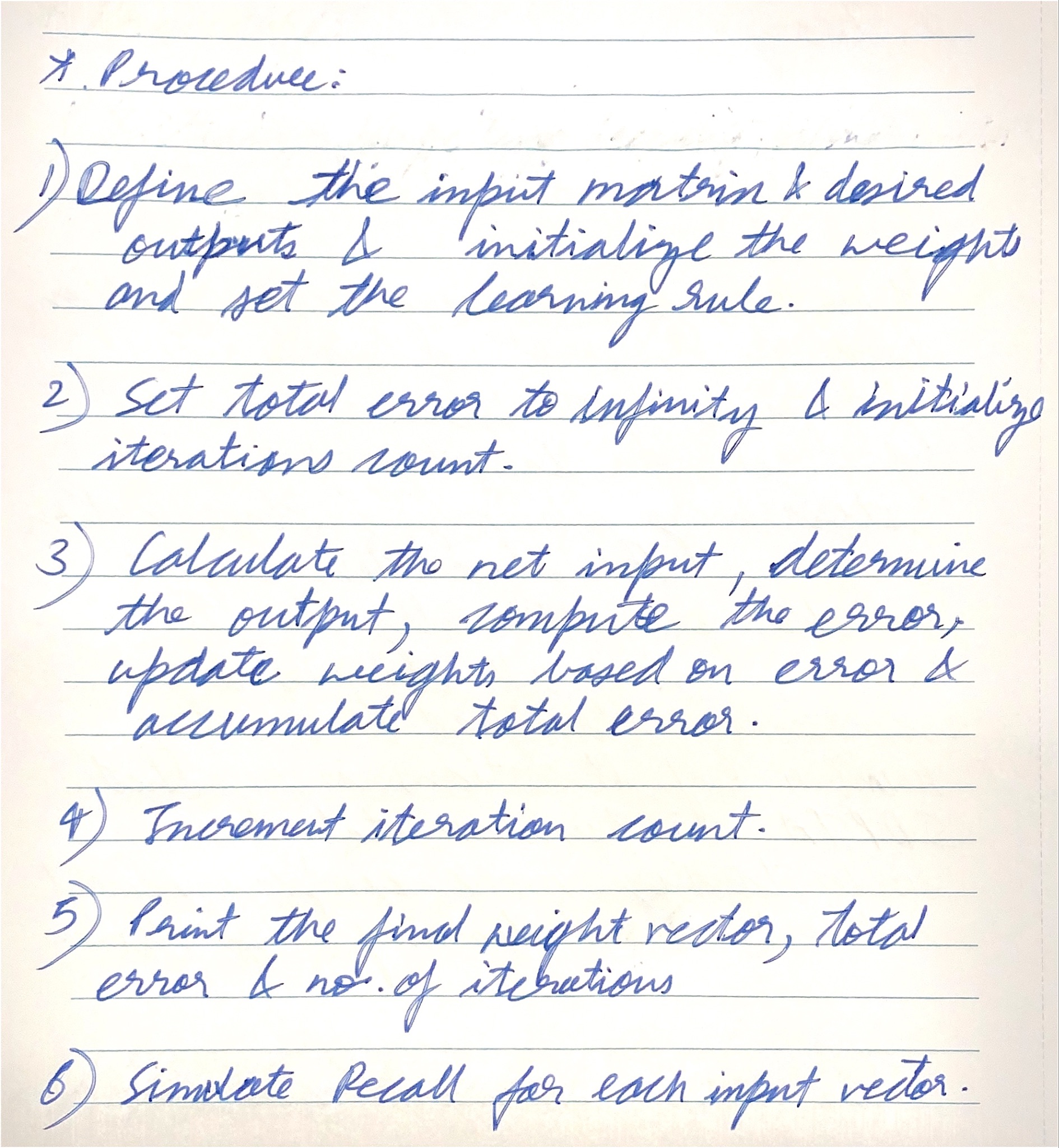
**Aim:** Design and simulate program for implementation of Single discrete single layer Perceptron Learning Algorithm (SDPTA). To Simulate Recall for SDPTA.

**Objective:** Students will learn and implement

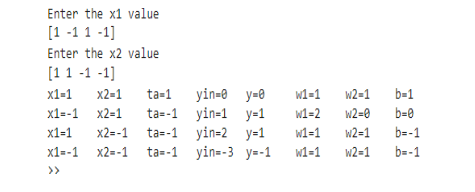
* Perceptron Model
* Neural Networks Fundamentals

**Explanation/Stepwise Procedure/ Algorithm:**





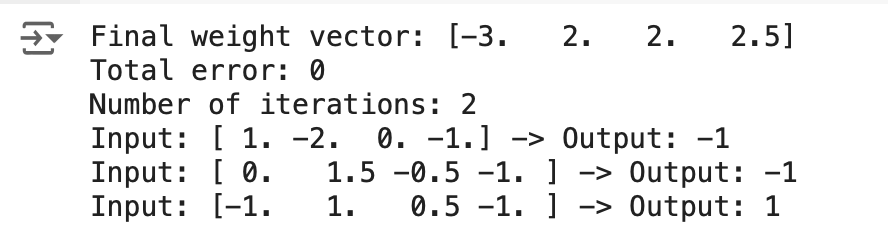
**Input & Output:**



**Code:**

| import numpy as np  inputs = np.array([[1, -2, 0, -1],  [0, 1.5, -0.5, -1],  [-1, 1, 0.5, -1]]) desired\_outputs = np.array([-1, -1, 1])  weights = np.array([1, -1, 0, 0.5]) learning\_rate = 1 iterations = 0  total\_error = float('inf') while total\_error != 0:  total\_error = 0  for i in range(len(inputs)):  net\_input = np.dot(weights, inputs[i, :])  output = 1 if net\_input >= 0 else -1  error = desired\_outputs[i] - output  total\_error += abs(error)  weights = weights + learning\_rate \* error \* inputs[i, :]  iterations+=1  print("Final weight vector:", weights) print("Total error:", total\_error) print("Number of iterations:", iterations)  for i in range(len(inputs)):  net\_input = np.dot(weights, inputs[i, :])  output = 1 if net\_input >= 0 else -1  print(f"Input: {inputs[i]} -> Output: {output}") |
| --- |

**Output:**



**Conclusion:**