Perceptron Learning Algorithm

**Objective:**

Write a Program to implement the Perceptron Learning Algorithm using numpy in Python. Evaluate the performance of a single perceptron for NAND and XOR truth tables as input datasets.

**Description:**

A perceptron is a single-layer neural network and the simplest artificial neural network. It consists of input nodes, weights, a bias term, and an activation function.

* Weighted sum is calculated by multiplying the inputs with the weights and adding them.
* Activation function classifies the output using these weighted sums.

If weighted sum >= 0, then output = 1

If weighted sum < 0, then output = 0

**Python Implementation:**

import numpy as np

class Perceptron:

    def \_\_init\_\_(self, input\_size, learning\_rate=0.1, epochs=1000):

        # Initialize weights and bias

        self.weights = np.zeros(input\_size)

        self.bias = 0.0

        self.learning\_rate = learning\_rate

        self.epochs = epochs

    def activation\_function(self, x):

        # Step activation function

        return 1 if x >= 0 else 0

    def predict(self, inputs):

        linear\_output = np.dot(inputs, self.weights) + self.bias

        return self.activation\_function(linear\_output)

    def train(self, X, y):

        for epoch in range(self.epochs):

            for i in range(len(X)):

                prediction = self.predict(X[i])

                error = y[i] - prediction

                # Update weights and bias

                self.weights += self.learning\_rate \* error \* X[i]

                self.bias += self.learning\_rate \* error

    def evaluate(self, X, y):

        correct\_predictions = sum(self.predict(x) == y\_true for x, y\_true in zip(X, y))

        accuracy = correct\_predictions / len(y)

        return accuracy

# NAND Truth Table (inputs and labels)

nand\_X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

nand\_y = np.array([1, 1, 1, 0])  # NAND output

# XOR Truth Table (inputs and labels)

xor\_X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

xor\_y = np.array([0, 1, 1, 0])  # XOR output

# Training and evaluation for NAND Gate

print("Training Perceptron for NAND Gate")

nand\_perceptron = Perceptron(input\_size=2)

nand\_perceptron.train(nand\_X, nand\_y)

nand\_accuracy = nand\_perceptron.evaluate(nand\_X, nand\_y)

print("NAND Perceptron Accuracy:", nand\_accuracy)

print("Predictions for NAND Truth Table:")

for x in nand\_X:

    print(f"Input: {x}, Prediction: {nand\_perceptron.predict(x)}")

# Training and evaluation for XOR Gate

print("\nTraining Perceptron for XOR Gate")

xor\_perceptron = Perceptron(input\_size=2)

xor\_perceptron.train(xor\_X, xor\_y)

xor\_accuracy = xor\_perceptron.evaluate(xor\_X, xor\_y)

print("XOR Perceptron Accuracy:", xor\_accuracy)

print("Predictions for XOR Truth Table:")

for x in xor\_X:

    print(f"Input: {x}, Prediction: {xor\_perceptron.predict(x)}")

**Description of code:**

1. **Class perceptron –**

* Initializes the weights and bias to 0.
* It contains an activation function that classifies the output.
* A predict function that predicts the output using the activation function.
* A train function that trains the model to find the required weights.
* An evaluate function that calculates the accuracy.

1. **Training and Evaluation for NAND Gate -**

* The NAND truth table (nand\_X) and labels (nand\_y) are defined.
* The perceptron is trained using the train() method.
* The model accuracy is calculated using the evaluate() function.
* The predictions for all inputs in the NAND table are displayed.

1. **Training and Evaluation for XOR Gate**

* The XOR truth table (xor\_X) and labels (xor\_y) are defined.
* The perceptron is trained and evaluated on the XOR data.
* Accuracy is computed, and predictions for all XOR inputs are printed.

**Output:**

Training Perceptron for NAND Gate

NAND Perceptron Accuracy: 1.0

Predictions for NAND Truth Table:

Input: [0 0], Prediction: 1

Input: [0 1], Prediction: 1

Input: [1 0], Prediction: 1

Input: [1 1], Prediction: 0

Training Perceptron for XOR Gate

XOR Perceptron Accuracy: 0.5

Predictions for XOR Truth Table:

Input: [0 0], Prediction: 1

Input: [0 1], Prediction: 1

Input: [1 0], Prediction: 0

Input: [1 1], Prediction: 0

**Performance:**

NAND Gate

- Accuracy: 1.0 (100%)

- Predictions: Correct for all inputs.

XOR Gate

- Accuracy: Approximately 0.5 (50%)

- Predictions: Incorrect for half inputs.

**My comments:**

* The perceptron cannot work properly on non-linearly separable data. Because of this, the perceptron is unable to learn the XOR gate.
* To improve this, we can use multi-layer perceptron.