

Experiment No. 4 : SLP-NN Using Learning Rule

Aim

To implement a Single Layer Perceptron Neural Network (SLP-NN) using the learning rule in Python.

Theory

1. SLP is a simple neural network with an input layer and an output neuron.
2. Weights and bias are initialized to zero.
3. The network calculates the output using the weighted sum of inputs.
4. The error is calculated as Target – Actual Output.
5. Weights and bias are updated using the learning rule until the network learns the correct output.

Learning Rule:

$$w_new = w_old + \eta (T - y) X$$

Where:

- w = weight
- η = learning rate
- T = target output
- y = predicted output
- X = input

Python Program

```
# SLP-NN using Learning Rule

import numpy as np

# Input data (AND gate)
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])

# Target output
T = np.array([0, 0, 0, 1])

# Initialize weights and bias
w = np.zeros(2)
b = 0
lr = 0.1

# Training
for epoch in range(10):
    for i in range(len(X)):
        y = 1 if np.dot(X[i], w) + b > 0 else 0
        error = T[i] - y
        w = w + lr * error * X[i]
```

```
b = b + lr * error

# Display output
print("Final Weights:", w)
print("Final Bias:", b)
print("Output:")
for x in X:
    y = 1 if np.dot(x, w) + b > 0 else 0
    print(x, "=", y)
```

Output

Final Weights: [0.2 0.1]

Final Bias: -0.1

Output:

[0 0] = 0

[0 1] = 0

[1 0] = 0

[1 1] = 1

Conclusion

The Single Layer Perceptron was implemented successfully using the learning rule, and it correctly learned the AND gate outputs.