K. J. Somaiya College of Engineering, Mumbai-77
Department of Computer Engineering

Batch: C1 **Roll No.:** 16010122270

Experiment No. 05

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of OR function with bipolar inputs and targets using Adaline network. Assume the required parameters for training of the network.

Objective: To learn Adaline network.

Expected Outcome of Experiment:

CO2: To understand the features of neural networks and different learning methods.

Books/ Journals/ Websites referred:

Pre Lab/ Prior Concepts:

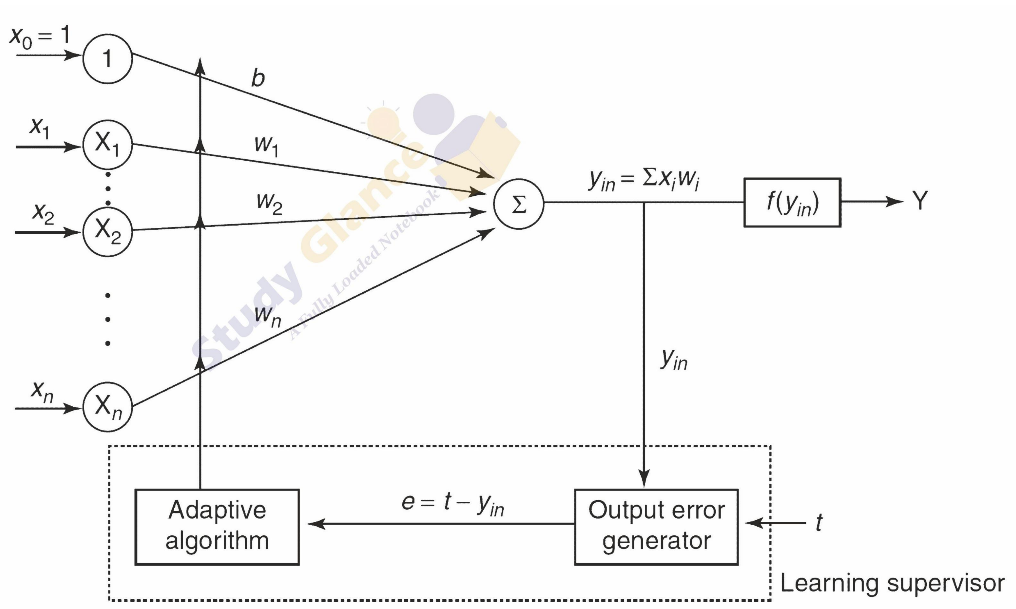
Adaptive Linear Neuron (Adaline):

Adaline which stands for Adaptive Linear Neuron, is a network having a single linear unit.
It was developed by Widrow and Hoff in 1960.
Some important points about Adaline are as follows −

- It uses bipolar activation function.
- It tries to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.

The weights and the bias are adjustable

Architecture:



Algorithm:

Step 1: Initialize the following to start the training −Weights, Bias, Learning rate α

Step 2: While the stopping condition is False do steps 3 to 7.

Step 3: for each training set perform steps 4 to 6.

Step 4: Set activation of input unit xi = si for (i=1 to n).

Step 5: compute net input to output unit $y_{in} = \sum w_i x_i + b$

Here, b is the bias and n is the total number of neurons.

Step 6: Update the weights and bias for i=1 to n

$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$$
$$b(new) = b(old) + (t - y_{in})$$

and calculate $error : (t - y_{in})^2$

Step 7: Test the stopping condition. The stopping condition may be when the weight changes at a low rate or no change.

Implementation Details:

Implementation of OR function with bipolar inputs and targets using Adaline network.

x1	x2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

Code:

```
import numpy as np

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

w = np.array([0.1, 0.1])
b = 0.1
learning_rate = 0.1
threshold_error = 1
epochs = 10

def linear_activation(x):
    return x

for epoch in range(epochs):
    print(f'Epoch {epoch + 1}/{epochs}')
    total_error = 0

    for xi, target in zip(X, y):
        net_input = np.dot(xi, w) + b
        output = linear_activation(net_input)

        error = target - output

        w += learning_rate * error * xi
```

```
        b += learning_rate * error

        total_error += error ** 2

        print(f'Input: {xi}, Target: {target}, Output: {output:.2f}, Error: {error:.2f}')

    print(f'Total Error: {total_error:.2f}')

    if total_error <= threshold_error:
        print("Training complete!")
        break
    print(f'Updated weights: {w}, Bias: {b}\n')
```

```
print('Testing the trained ADALINE:')
for xi in X:
    net_input = np.dot(xi, w) + b
    output = linear_activation(net_input)
    prediction = 1 if output >= 0.5 else 0
    print(f'Input: {xi}, Prediction: {prediction}')
```

Output screenshot:

```
PS C:\Users\sanji\Desktop\sem5 code> & C:/Python312/python.exe "c:/Users/sanji/Desktop/sem5 code/SCexp5.py"
Epoch 1/10
Input: [0 0], Target: 0, Output: 0.10, Error: -0.10
Input: [0 1], Target: 1, Output: 0.19, Error: 0.81
Input: [1 0], Target: 1, Output: 0.27, Error: 0.73
Input: [1 1], Target: 1, Output: 0.60, Error: 0.40
Total Error: 1.36
Updated weights: [0.21312 0.22122], Bias: 0.28412000000000004

Epoch 2/10
Input: [0 0], Target: 0, Output: 0.28, Error: -0.28
Input: [0 1], Target: 1, Output: 0.48, Error: 0.52
Input: [1 0], Target: 1, Output: 0.52, Error: 0.48
Input: [1 1], Target: 1, Output: 0.89, Error: 0.11
Total Error: 0.60
Training complete!
Testing the trained ADALINE:
Input: [0 0], Prediction: 0
Input: [0 1], Prediction: 1
Input: [1 0], Prediction: 1
Input: [1 1], Prediction: 1
PS C:\Users\sanji\Desktop\sem5 code>
```

Conclusion: Thus we have successfully implemented the ADALINE network for Implementation of OR function with bipolar inputs and targets.

Post Lab Descriptive Questions:

Use Adaline network to train AND NOT function with bipolar inputs and targets. Perform 1 epoch of training

Soln:

x_1	x_2	t
1	1	-1
1	-1	1
-1	1	-1
-1	-1	-1

$w_1 = 0.2$ $w_2 = 0.2$ $b = 0.2$ $\alpha = 0.2$

least squared error

I for first input

$x_1 = 1$ $x_2 = 1$ $t = -1$

$$Y_{in} = w_1 x_1 + w_2 x_2 + b$$
$$= (0.2)\,1 + (0.2)(1) + 0.2$$
$$= 0.6$$

$$(t - Y_{in}) = -1 - 0.6 = -1.6$$

∴ Updating weights.

$$w_1(new) = w_1(old) + \alpha\,(t - y_{in})\,x_1$$
$$= 0.2 + 0.2(-1.6)(1) = -0.12$$

$$w_2(new) = w_2(old) + \alpha\,(t - y_{in})\,x_2$$
$$= 0.2 + 0.2(-1.6)(1) = -0.12$$

$$b(new) = b(old) + \alpha(t - y_{in}) = 0.2 + 0.2(-1.6) = -0.12$$

$\Delta w_1 = -0.32$, $\Delta w_2 = -0.32$, $\Delta b_2 = -0.32$

$\therefore$ Error (E),

$E = (t - Y_{in})^2 = (-1.6)^2$

$E = 2.56$

II Second input

$x_1 = 1 \quad x_2 = -1 \quad t = 1$

$Y_{in} = w_1 x_1 + w_2 x_2 + b$
$= -0.12$

$\therefore (t - Y_{in}) = 1 - (-0.12) = 1.12$

Updating weights,

$w_1(new) = w_1(old) + \alpha(t - Y_{in})x_1$
$= 0.10$

$w_2(new) = w_2(old + \alpha(t - Y_{in})x_2$
$= -0.34$

$b(new) = b(old) + (t - Y_{in})\alpha = 0.10$

$E = (t - Y_{in})^2 = (1.12)^2 = 1.25$

$\Delta w_1 = 0.22 \qquad \Delta w_2 = -0.22 \qquad \Delta b_2 = 0.22$

III Third Input

$x_1 = -1 \quad x_2 = 1$
$t = -1$

$Y_{in} = w_1 x_1 + w_2 x_2 + b$
$= -0.34$

$(t - Y_{in}) = (-1 - (-0.34)) = -0.66$

Updating weights,

$w_1(new) = w_1(old) + \alpha(t - Y_{in})x_1$
$= 0.24$

$w_2(new) = w_2(old) + \alpha(t - Y_{in})x_2$

$b(new) = b(old) + \alpha(t - Y_{in}) = -0.03$

$E = (t - Y_{in})^2 = 0.43$

$\Delta w_1 = 0.13 \qquad \Delta w_2 = 0.13 \qquad \Delta b = -0.13$

(iv) Fourth Input

$x_1 = -1 \quad x_2 = -1$

$t = -1$

$Y_{in} = w_1 x_1 + w_2 x_2 + b$

$= 0.21$

$(t - Y_{in}) = -1 - 0.21 = -1.2$[illegible]

Updating weights,

$w_1(new) = w_1(old) + \alpha(t - Y_{in}) x_1$

$= 0.48$

$w_2(new) = w_2(old) + \alpha(t - Y_{in}) x_2$

$= -0.24$

$b(new) = b(old) + \alpha(t - Y_{in}) = -0.27$

$E = (t - Y_{in})^2 = 1.47$

$\therefore$ After 1 Epoch,

$E = 1.47$

$w_1 = 0.48$

$w_2 = -0.24$

$b = -0.27$

Date:

Signature of faculty in-charge