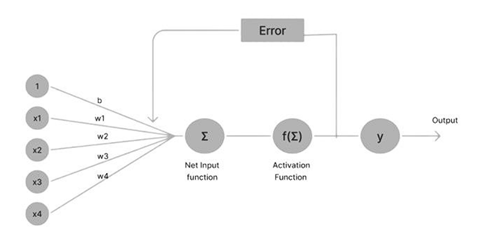
**Experiment No.: 06**

**Aim:** Write a program to implement logical OR using the Adaline network



**Steps**:

1.     **Initialize Parameters:**

·        Set initial weights **w1** and **w2** to 0.1. Set the learning rate (**learning\_rate**) to 0.1.Set the initial bias (**bias**) to 0.1.

2.     **Define Input Patterns:**

·        Define input patterns **xi** as [(1, 1), (1, 0), (0, 1), (0, 0)].

3.     **Iterative Training:**

·        For each input pattern **x** in **xi**:

·        Calculate the linear combination: **linear\_combination = w1 \* x[0] + w2 \* x[1] + bias**.

·        Calculate the target output: **target\_output = x[0] or x[1]**.

·        Calculate the difference between target and actual output:

**diff\_output = target\_output - linear\_combination**.

·        Calculate the error: **error = pow(diff\_output, 2)**. and Update total error: **terr = error + terr**.

·        Update weights and bias using the perceptron learning rule:

·        **w1 = w1 + learning\_rate \* diff\_output \* x[0]**

·        **w2 = w2 + learning\_rate \* diff\_output \* x[1]**

·        **bias = bias + learning\_rate \* diff\_output**

4.     **Output and Convergence Check:**

·        Print total error after processing: **print("Total Error after processing all examples:", terr)**.

·        Create a DataFrame (**output\_frame**) to display input, linear combination, activation output, weights, bias, and error for each example.Print the DataFrame.

5.     **Convergence Check and Recursion:**

·        Check if the total error (**terr**) is less than the specified maximum error (**MaxErr**).

·        If not converged, print "Convergence achieved. Error:" and call the **cal\_output\_and** function recursively with updated weights and bias.

**Code:**

import numpy as np

import pandas as pd

def cal\_output\_and(w1, w2, learning\_rate, bias, MaxErr=1):

xi = [(1, 1), (1, 0), (0, 1), (0, 0)]

outputs = []

terr=0

for x in xi:

linear\_combination = w1 \* x[0] + w2 \* x[1] + bias

target\_output = x[0] or x[1]

diff\_output = target\_output - linear\_combination

error = pow(diff\_output, 2)

terr=error+terr

w1 = w1 + learning\_rate \* diff\_output \* x[0]

w2 = w2 + learning\_rate \* diff\_output \* x[1]

bias = bias + learning\_rate \* diff\_output

outputs.append([x[0], x[1], linear\_combination, target\_output,w1,w2,bias,error])

print("Total Error after processing all examples:", terr ) # Corrected total error calculation

output\_frame = pd.DataFrame(outputs, columns=['Input 1', 'Input 2', 'Actual Output', 'Target Output', 'w1', 'w2', 'bias', 'Error'])

print(output\_frame.to\_string(index=False))

if MaxErr < terr:

print("Convergence achieved. Error:", terr)

print("\nNext Epoch")

cal\_output\_and(w1, w2, learning\_rate, bias)

return

w1 = 0.1

w2 = 0.1

learning\_rate = 0.1

bias = 0.1

cal\_output\_and(w1, w2, learning\_rate, bias)

**Output:**

