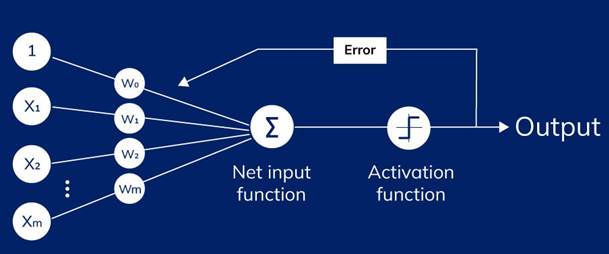
**Experiment No.: 04**

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



**Steps:**

1.     **Initialization:**

·        Initialize weights (**w1** and **w2**) and bias (**bias**) to some values (provided as arguments to the function).

·        Define the input pairs **xi** for the logical OR operation.

2.     **Iterate through Input Pairs:**

·        For each input pair **(x[0], x[1])** in **xi**:

·        Calculate the linear combination of inputs and weights:

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

·        Determine the output based on a threshold:

**output = 1 if linear\_combination >= threshold else 0**.

·        Set the target output as the logical OR of the inputs: **target\_output = x[0] or x[1]**.

**Weight Update:**

If the output does not match the target output:

·        Update weights using the perceptron learning rule:

·        **w1 = w1 + learning\_rate \* (target\_output - output) \* x[0]**

·        **w2 = w2 + learning\_rate \* (target\_output - output) \* x[1]**

·        Clear the **outputs** list to start fresh for the next iteration.

·        Recursively call the **cal\_output\_and** function with updated weights to continue the learning process.

3.     **Output Results:**

·        Store the results (inputs, linear combination, and activation output) in a DataFrame for visualization.

·        Print the results, including weights and learning rate.

**Code:**

import numpy as np

import pandas as pd

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

bias = 0

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

outputs = []

for x in xi:

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

output = int(linear\_combination >= threshold)

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

if output != target\_output:

w1 = w1 + learning\_rate \* (target\_output - output) \* x[0]

w2 = w2 + learning\_rate \* (target\_output - output) \* x[1]

outputs = [] # Clear outputs when weights are updated

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

return

outputs.append([x[0], x[1], linear\_combination, output])

output\_frame = pd.DataFrame(outputs, columns=['Input 1', 'Input 2', 'Linear Combination', 'Activation Output'])

print(f'Results for threshold: {threshold} \n')

print(f'Results for w1: {w1} \n')

print(f'Results for w2: {w2} \n')

print(f'Results for learning\_rate: {learning\_rate} \n')

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

w1 = 0.6

w2 = 0.6

learning\_rate = 0.5

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

**Output:**

