**Experiment No. 5-** **Implementing Perceptron Model for Logical Functions**

**Aim:** To implement basic supervised neural network learning rules for a problem. Design a NN using a leaning method to generate knowledge for classification.

**Learning Objective:** Understand and implement the basic operations of a neuron within a neural network.

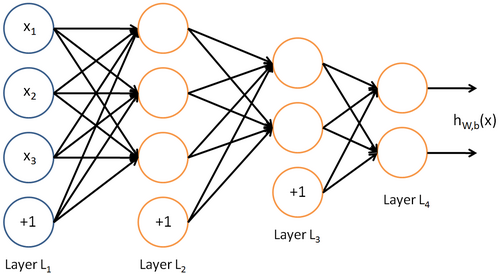
Design a simple Perceptron neural network to classify data based on given inputs and outputs.

**Tools:** Python 3 language and Jupyter Notebook

**Theory:**

A neural network is a computational model inspired by the way biological neural networks in the brain process information. It consists of interconnected neurons that pass information through layers: an input layer, one or more hidden layers, and an output layer. Each neuron performs a simple operation: it takes weighted inputs, adds a bias, applies an activation function, and passes the result to the next layer. Learning in neural networks involves adjusting the weights based on errors using methods like backpropagation.

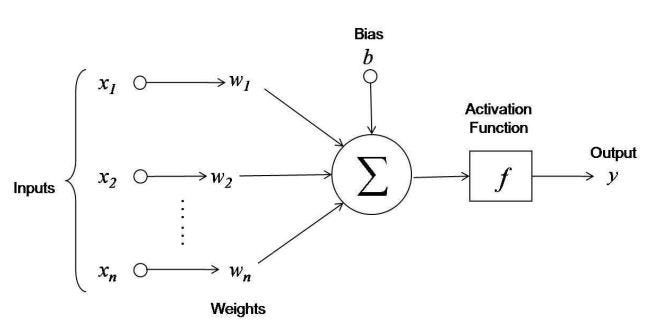
So, we can represent an artificial neural network like that :



**What does a neuron do ?**

The operations done by each neurons are pretty simple :

* Summation: Adds up the values from all connected neurons from the previous layer. Each value is multiplied by a unique weight specific to that connection.
* Bias Addition: Adds a bias term to the sum, which helps the network learn more effectively.
* Activation Function: Applies an activation function (e.g., sigmoid or ReLU) to the total value. This function transforms the output into a specific range, typically between 0 and 1 for sigmoid or 0 and positive infinity for ReLU.



**Perceptron:**

A simple neural network with 2 input neurons and 1 output neuron. It can classify based on rules like the inclusive OR:

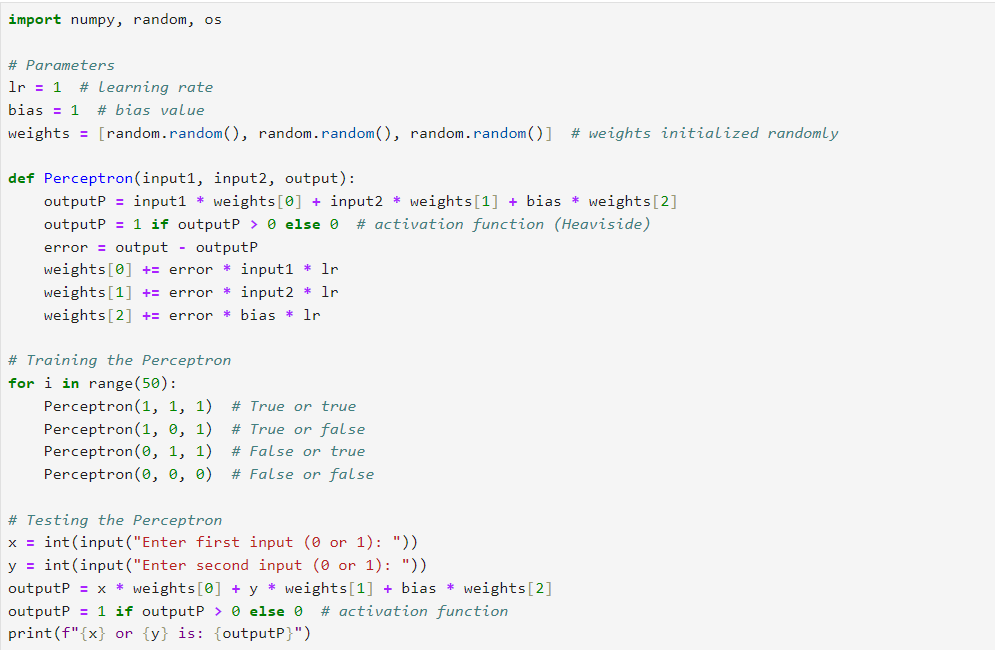
A OR B: Outputs 1 if either A or B is 1, otherwise 0.

Example

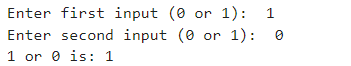
Inputs: True (1) or False (0).

Output: Classification based on the inclusive OR logic.

**Code & Output:**



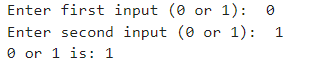
Ouput 1:



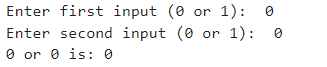
Ouput 2:



Ouput 3:



Ouput 4:



**Result & Discussion:**

1. Training Phase:

The Perceptron was trained using the inclusive OR logic. The training data consisted of all possible combinations of two binary inputs (0 and 1), with their corresponding OR outputs.

2. Test Phase:

The Perceptron was tested with user-provided inputs to classify based on the trained OR logic.

Results:

* True or True (1, 1): The model correctly outputs 1.
* True or False (1, 0): The model correctly outputs 1.
* False or True (0, 1): The model correctly outputs 1.
* False or False (0, 0): The model correctly outputs 0.

The Perceptron successfully learned to perform the inclusive OR operation, demonstrating its ability to classify binary inputs according to the OR logic. The simplicity and effectiveness of the Perceptron make it a fundamental building block for understanding more complex neural networks. Adjustments in learning rate and iterations can further refine the model's performance.

**Learning Outcomes:** Students gain a foundational understanding of neural computation using the Perceptron model and learn to implement and solve basic logic functions (such as OR) using this simple neural network.

**Course Outcomes:** Neural Computation Principles: Upon completion of the activity, students will be able to understand and use the Perceptron model to grasp basic neural computation principles.

Solving Logical Functions: Students will be able to implement and solve logical functions (OR) using a threshold logic unit.

**Conclusion:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Correction** | **Formative** | **Timely** | **Attendance /** |  |
| **Parameters** | **Assessment** | **completion of** | **Learning** |
|  | **[40%]** | **Practical [ 40%]** | **Attitude** |
|  |  |  | **[20%]** |
| **Marks** |  |  |  |
| **Obtained** |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For Faculty Use: