National University of Computer and Emerging Sciences

Lab Manual 09
CL461-Artificial Intelligence Lab

Course Instructor	Mrs. Bushra Rashid
Lab Instructor (s)	Mr. Gullsher Ali Chaudhary Ms. Sukhan Amir
Section	BCS-6G
Semester	Spring 2024

Department of Computer Science
FAST-NU, Lahore, Pakistan

Objectives

After performing this lab, students shall be able to understand the following:

- ✓ Artificial Neural Network (ANN)

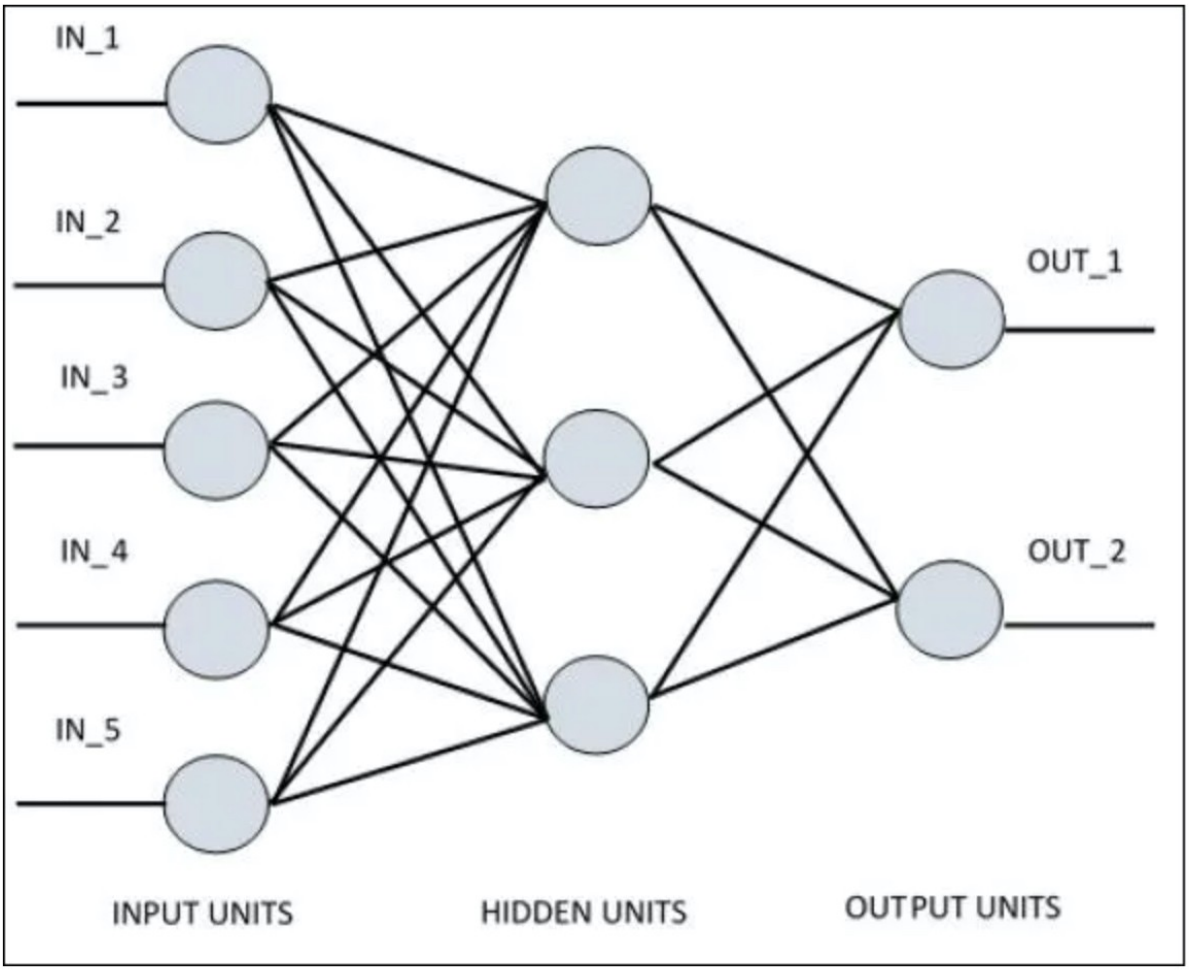
Task#1

3 Artificial Neural Network (ANN)

An Artificial neural network is a computational network, based on biological neural networks that construct the structure of the human brain. Similar to how a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks

ANNs have an input layer and output layer. Between these two layers there are other hidden layers that perform the mathematical computations that help determine the decision or action the machine should take. Ultimately, these hidden layers are in place to transform the input data into something the output unit can use.

The schematic diagram of the artificial neural network is as follows.



This figure shows that the hidden entity is communicating with the external layer. Input and output units, on the other hand, communicate only through the hidden layers of the network.

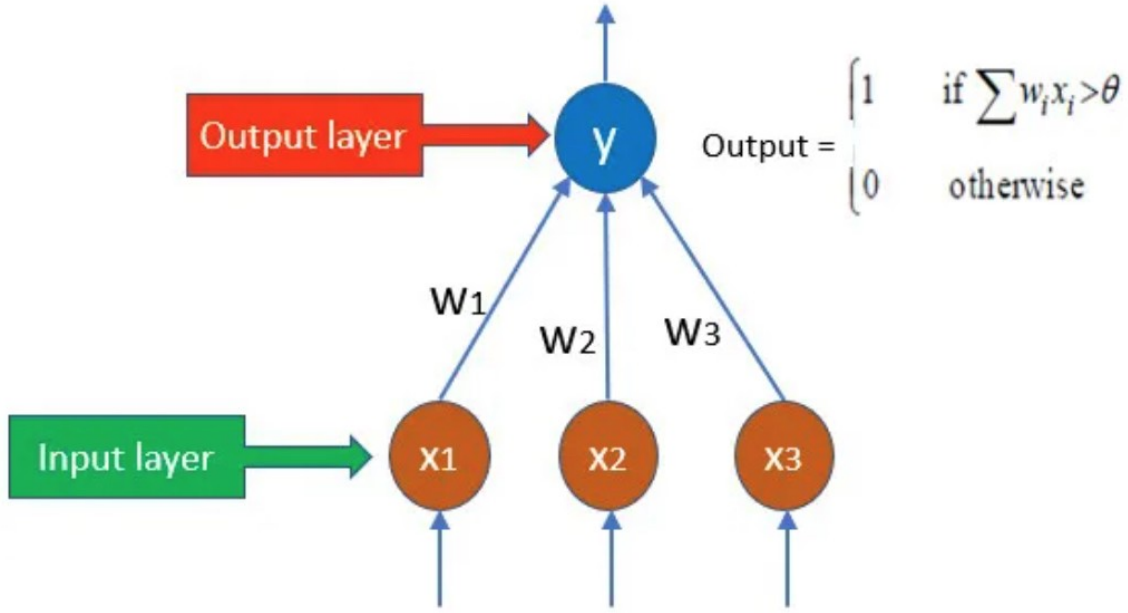
The connection pattern with the nodes, the total number of layers, the level of the nodes between the inputs and outputs, and the number of neurons per layer, define the architecture of the neural network.

There are two types of architecture. These types focus on the functionality artificial neural networks as follows −

1. Single Layer Perceptron
2. Multi-Layer Perceptron

Single Layer Perceptron:

In this lab we will go through a single-layer perceptron this is the first and basic model of the artificial neural networks. It is also called the feed-forward neural network. The working of the single-layer perceptron (SLP) is based on the threshold transfer between the nodes. This is the simplest form of ANN and it is generally used in the linearly based cases for the machine learning problems.



y = b + w1x1 + w2x2+ w3x3

The single layer perceptron does not have a priori knowledge, so the initial weights are assigned randomly. SLP sums all the weighted inputs and if the sum is above the threshold (some predetermined value), SLP is said to be activated (output=1).

The input values are

	Output
$w_1x_1 + w_2x_2 + \ldots + w_nx_n > \theta$	1
$w_1x_1 + w_2x_2 + \ldots + w_nx_n \leq \theta$	0

presented to the perceptron, and if the predicted output is the same as the desired output, then the performance is considered satisfactory and no changes to the weights are made. However, if the output does not match the desired output, then the weights need to be changed to reduce the error

Learning rate = 0.2
Threshold = 0.5
Actual output = W1X1+W2X2

Change in weight = Wn = learning rate * (desired output- actual output) * Xn
Next weight adjustment = Wn+ Wn

Because SLP is a linear classifier and if the cases are not linearly separable the learning process will never reach a point where all the cases are classified properly.

Implementing the AND gate using SLP

The objective of this lab is to implement the AND gate using a Single Layer Perceptron we first define the input and output values for the AND gate and then define the activation function. The weights and bias are initialized randomly, and the learning rate and number of epochs are set.

A complete flow of algorithm is shown below.

Flow of Algorithm:

```
import numpy as np

# Define the input data for the AND gate
input_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

# Define the target output for the AND gate
target_output = np.array([[0], [0], [0], [1]])

# Define the learning rate and the number of iterations
learning_rate = 0.1
num_iterations = 10

# Define the activation function (step function)
def activation_function(x):
    return 1 if x >= 0 else 0

# Initialize the weights and the bias
weights = np.array([0.0, 0.0])
bias = 0.0
```

```
# Train the perceptron
for i in range(num_iterations):
    for j in range(len(input_data)):
        x = input_data[j]
        y = target_output[j]
        # Calculate the weighted sum of the inputs and the bias
        # Calculate the predicted output using the activation function
        # Calculate the error
        # Update the weights and the bias

# Print learned weights
# Test the perceptron

test_inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
```

Here you are supposed to fill the missing part in above give code of SLP to train for AND gate and show learned weights and bias terms after training.

Submission Instructions

Always read the submission instructions carefully.

- Rename your Jupyter notebook to your roll number and download the notebook as **.ipynb** extension.
- To download the required file, go to **File->Download .ipynb**
- Only submit the **.ipynb** file. DO NOT **zip** or **rar** your submission file.
- Submit this file on Google Classroom under the relevant assignment.
    - Late submissions will not be accepted.