



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Name: Deepali Kothari
Roll no: 09
Experiment No. 2
Implement Multilayer Perceptron algorithm to simulate XOR gate
Date of Performance: 31/7/23
Date of Submission: 7/8/23

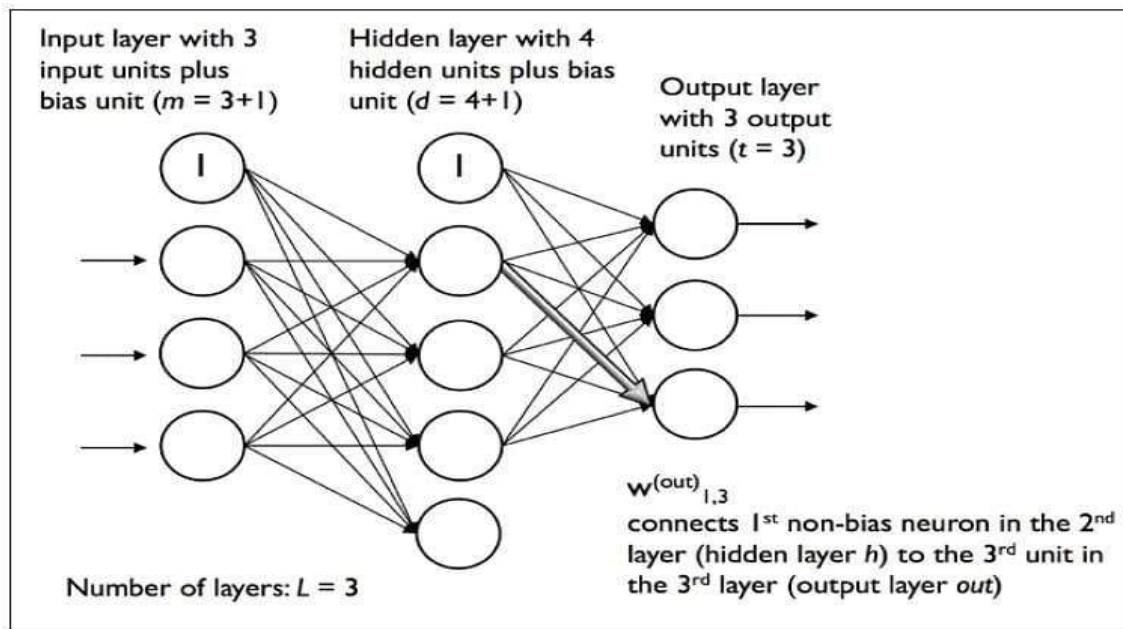


Aim: Implement Multilayer Perceptron algorithm to simulate XOR gate.

Objective: Ability to perform experiments on different architectures of multilayer perceptron.

Theory:

multilayer artificial neuron network is an integral part of deep learning. And this lesson will help you with an overview of multilayer ANN along with overfitting and underfitting.



A fully connected multi-layer neural network is called a Multilayer Perceptron (MLP).

At has 3 layers including one hidden layer. If it has more than 1 hidden layer, it is called a deep ANN. An MLP is a typical example of a feedforward artificial neural network. In this figure, the i th activation unit in the l th layer is denoted as $a_i(l)$.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The number of layers and the number of neurons are referred to as hyperparameters of a neural network, and these need tuning. Cross-validation techniques must be used to find ideal values for these.

The weight adjustment training is done via backpropagation. Deeper neural networks are better at processing data. However, deeper layers can lead to vanishing gradient problems. Special algorithms are required to solve this issue.

A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network. MLP is a deep learning method.

Code :

```
# importing Python library
```

```
import numpy as np
```

```
# define Unit Step Function
```

```
def unitStep(v):
```

```
    if v >= 0:
```

```
        return 1
```

```
    else:
```

```
        return 0
```

```
# design Perceptron Model
```

```
def perceptronModel(x, w, b):
```

```
    v = np.dot(w, x) + b
```

```
    y = unitStep(v)
```

```
    return y
```

CSL701: Deep Learning Lab



NOT Logic Function

$w_{NOT} = -1$, $b_{NOT} = 0.5$

def NOT_logicFunction(x):

$w_{NOT} = -1$

$b_{NOT} = 0.5$

 return perceptronModel(x, w_{NOT} , b_{NOT})

AND Logic Function

here $w_1 = w_{AND1} = 1$,

$w_2 = w_{AND2} = 1$, $b_{AND} = -1.5$

def AND_logicFunction(x):

$w = \text{np.array}([1, 1])$

$b_{AND} = -1.5$

 return perceptronModel(x, w, b_{AND})

OR Logic Function

$w_1 = 1$, $w_2 = 1$, $b_{OR} = -0.5$

def OR_logicFunction(x):

$w = \text{np.array}([1, 1])$

$b_{OR} = -0.5$

 return perceptronModel(x, w, b_{OR})

XOR Logic Function

with AND, OR and NOT

function calls in sequence

def XOR_logicFunction(x):

CSL701: Deep Learning Lab



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
y1 = AND_logicFunction(x)
y2 = OR_logicFunction(x)
y3 = NOT_logicFunction(y1)
final_x = np.array([y2, y3])
finalOutput = AND_logicFunction(final_x)
return finalOutput
```

testing the Perceptron Model

```
test1 = np.array([0, 0])
```

```
test2 = np.array([0, 1])
```

```
test3 = np.array([1, 0])
```

```
test4 = np.array([1, 1])
```

```
print("XOR({}, {}) = {}".format(0, 0, XOR_logicFunction(test1)))
```

```
print("XOR({}, {}) = {}".format(0, 1, XOR_logicFunction(test2)))
```

```
print("XOR({}, {}) = {}".format(1, 0, XOR_logicFunction(test3)))
```

```
print("XOR({}, {}) = {}".format(1, 1, XOR_logicFunction(test4)))
```

Output:

```
XOR(0, 0) = 0
XOR(0, 1) = 1
XOR(1, 0) = 1
XOR(1, 1) = 0
```



Conclusion:

In conclusion, the use of the Multilayer Perceptron (MLP) method to mimic the XOR gate demonstrates how network architecture and the backpropagation algorithm work together to solve non-linear classification issues. The importance of adding hidden layers and using backpropagation to get correct answers is highlighted by the XOR gate simulation, a well-known illustration of a challenge that a single-layer perceptron cannot solve.