LAB 06

Implementation of Perceptron and Activation Function

Demonstration with implementation of logic gates AND, OR

Activation Function

The activation function determines whether the neuron will "fire" (output 1) or not (output 0) based on the weighted sum of the inputs. For logic gates, the common activation function used is the Step function (also called the Heaviside function):

f(x)={1if x≥00if x<0f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}f(x)={10 if x≥0if x<0

Perceptron Implementation for AND Gate:

Weights: w1=1,w2=1w_1 = 1, w_2 = 1w1 =1,w2 =1

Bias: b=−1.5b =-1.5b=−1.5 This bias ensures that the weighted sum needs to be greater than 1.5 for the output to be 1, which only happens when both inputs are 1.

Perceptron Implementation for OR Gate:

Weights: w1=1,w2=1w_1 = 1, w_2 = 1w1 =1,w2 =1

Bias: b=−0.5b =-0.5b=−0.5 This bias ensures that the output is 1 if either input is 1, since the weighted sum needs to be greater than-0.5 for the output to be 1

Code for AND Gate Perceptron

```
# 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

# AND Logic Function
# w1 = 1, w2 = 1, b = -1.5
def AND_logicFunction(x):
    w = np.array([1, 1])
    b = -1.5
    return perceptronModel(x, w, b)

# testing the Perceptron Model
test1 = np.array([0, 1])
test2 = np.array([1, 1])
test3 = np.array([0, 0])
test4 = np.array([1, 0])

print("AND({}, {}) = {}".format(0, 1, AND_logicFunction(test1)))
print("AND({}, {}) = {}".format(1, 1, AND_logicFunction(test2)))
print("AND({}, {}) = {}".format(0, 0, AND_logicFunction(test3)))
print("AND({}, {}) = {}".format(1, 0, AND_logicFunction(test4)))
```

OUTPUT:

```
AND(0,                  1)                  =                  0
AND(1,                  1)                  =                  1
AND(0,                    0)                      =                    0
AND(1, 0) = 0
```

Code for OR Gate Perceptron:

```
# 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



# OR Logic Function

# w1 = 1, w2 = 1, b = -0.5

def OR_logicFunction(x):
```

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

b = -0.5

return perceptronModel(x, w, b)

 # testing the Perceptron Model

test1 = np.array([0, 1])

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

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

test4 = np.array([1, 0])


print("OR({}, {}) = {}".format(0, 1, OR_logicFunction(test1)))

print("OR({}, {}) = {}".format(1, 1, OR_logicFunction(test2)))

print("OR({}, {}) = {}".format(0, 0, OR_logicFunction(test3)))

print("OR({}, {}) = {}".format(1, 0, OR_logicFunction(test4)))
```

OUTPUT:

```
OR(0,                    1)                    =                    1
OR(1,                    1)                    =                    1
OR(0,                    0)                    =                    0
OR(1, 0) = 1
```