lab

July 10, 2025

0.1 Creating A Neural Network for AND, OR, XOR Gate

AND Gate

[1]:
```
import numpy as np
import math

w1 = 10
w2 = 10
bias = -15

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

### define the step activation function
def step(z):
        if z>0:
                return 1
        else:
                return 0

### define the sigmoid activation function
def sigmoid(z):
        if (1/(1+(math.e)**(-z)))>0.5:
                return 1
        else:
                return 0

print(f"Using Step function ")
print(f"X1| X2| Output ")
for x1,x2 in inputs:

        ### Calculation of linear function
        z = (x1*w1 +x2*w2) +bias

        ### calling the activation function
        output = step(z)

        print(f"{x1} | {x2} | {output}")
```

```
print("\n")
print(f"Using sigmoid function ")
print(f"X1| X2| Output ")
for x1,x2 in inputs:

        ### Calculation of linear function
        z = (x1*w1 +x2*w2) +bias

        ### calling the activation function
        output = sigmoid(z)

        print(f"{x1} | {x2} | {output}")
```

```
Using Step function
X1| X2| Output
0 | 0 | 0
0 | 1 | 0
1 | 0 | 0
1 | 1 | 1


Using sigmoid function
X1| X2| Output
0 | 0 | 0
0 | 1 | 0
1 | 0 | 0
1 | 1 | 1
```

0.1.1 OR Gate

[2]:
```
import numpy as np
import math

w1 = 10
w2 = 10
bias = 0

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

### define the step activation function
def step(z):
        if z>0:
                return 1
        else:
                return 0

### define the sigmoid activation function
```

```
def sigmoid(z):
        if (1/(1+(math.e)**(-z)))>0.5:
                return 1
        else:
                return 0

print(f"Using Step function ")
print(f"X1| X2| Output ")
for x1,x2 in inputs:

        ### Calculation of linear function
        z = (x1*w1 +x2*w2) +bias

        ### calling the activation function
        output = step(z)

        print(f"{x1} | {x2} | {output}")


print("\n")
print(f"Using sigmoid function ")
print(f"X1| X2| Output ")
for x1,x2 in inputs:

        ### Calculation of linear function
        z = (x1*w1 +x2*w2) +bias

        ### calling the activation function
        output = sigmoid(z)

        print(f"{x1} | {x2} | {output}")
```

```
Using Step function
X1| X2| Output
0 | 0 | 0
0 | 1 | 1
1 | 0 | 1
1 | 1 | 1


Using sigmoid function
X1| X2| Output
0 | 0 | 0
0 | 1 | 1
1 | 0 | 1
1 | 1 | 1
```

0.1.2 XOR Gate

[3]:
```python
import numpy as np
import math

# Simple XOR using two layers
inputs = [(0,0), (0,1), (1,0), (1,1)]

def step(z):
    return 1 if z > 0 else 0

def sigmoid(z):
    return 1 if (1/(1+(math.e)**(-z))) > 0.5 else 0

print("Using Step Function")
print("X1| X2| Output")

for x1, x2 in inputs:
    # Layer 1: Create two helper neurons
    # Neuron 1: OR gate (x1 OR x2)
    or_result = step(x1*1 + x2*1 - 0.5)

    # Neuron 2: AND gate (x1 AND x2)
    and_result = step(x1*1 + x2*1 - 1.5)

    # Layer 2: Final XOR = OR AND (NOT AND)
    # XOR = (x1 OR x2) AND NOT(x1 AND x2)
    xor_result = step(or_result*1 + and_result*(-1) - 0.5)

    print(f"{x1} | {x2} | {xor_result}")

print("\nUsing Sigmoid Function")
print("X1| X2| Output")

for x1, x2 in inputs:
    # Layer 1: Create two helper neurons
    # Neuron 1: OR gate (x1 OR x2)
    or_result = sigmoid(x1*1 + x2*1 - 0.5)

    # Neuron 2: AND gate (x1 AND x2)
    and_result = sigmoid(x1*1 + x2*1 - 1.5)

    # Layer 2: Final XOR = OR AND (NOT AND)
    # XOR = (x1 OR x2) AND NOT(x1 AND x2)
    xor_result = sigmoid(or_result*1 + and_result*(-1) - 0.5)

    print(f"{x1} | {x2} | {xor_result}")
```

```
Using Step Function
X1| X2| Output
0 | 0 | 0
0 | 1 | 1
1 | 0 | 1
1 | 1 | 0

Using Sigmoid Function
X1| X2| Output
0 | 0 | 0
0 | 1 | 1
1 | 0 | 1
1 | 1 | 0
```