Fuzzy Logic & Neural Networks (CS-514)

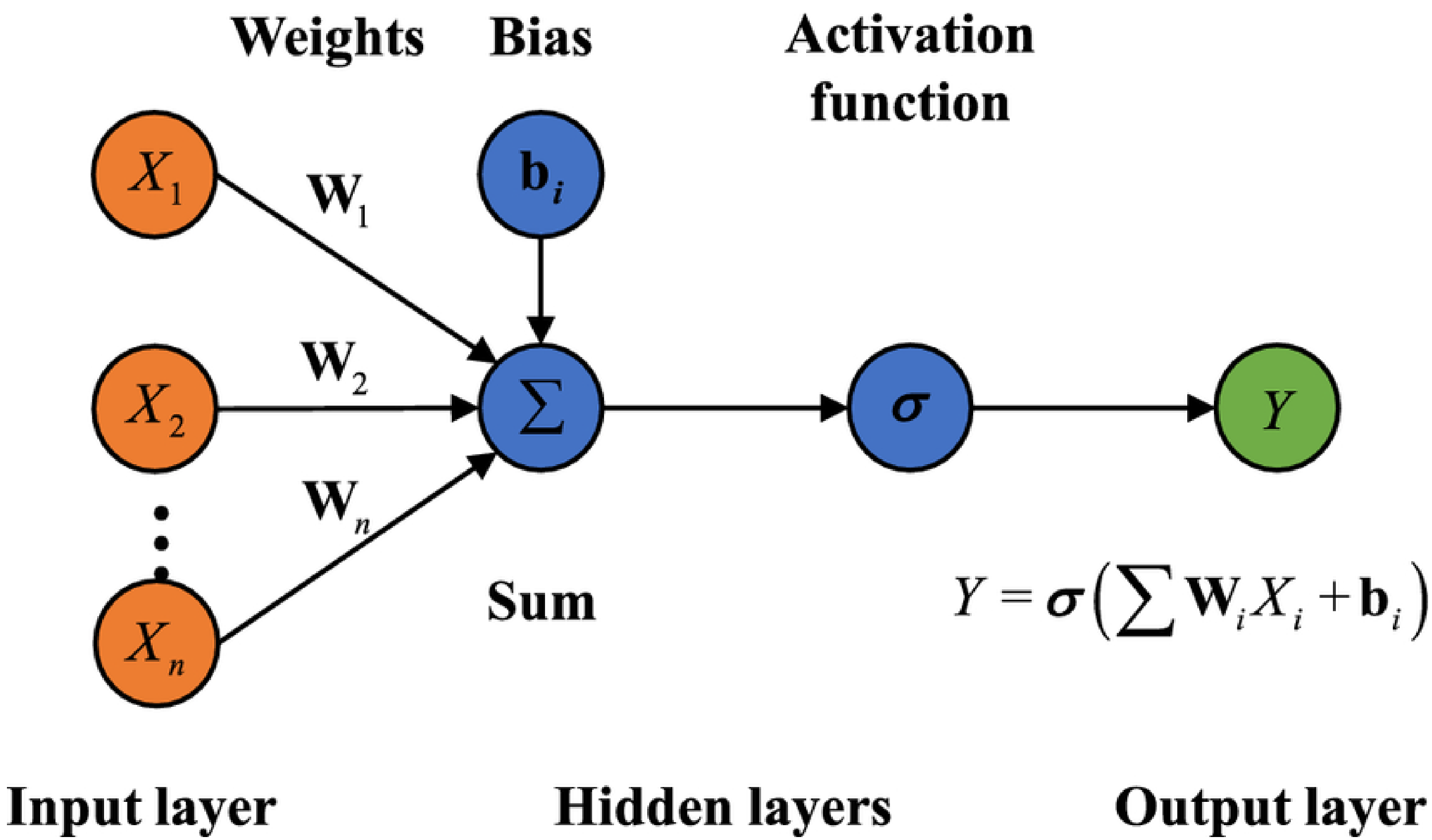
Dr. Sudeep Sharma

IIIT Surat

sudeep.sharma@iiitdmj.ac.in

Artificial Neuron Model

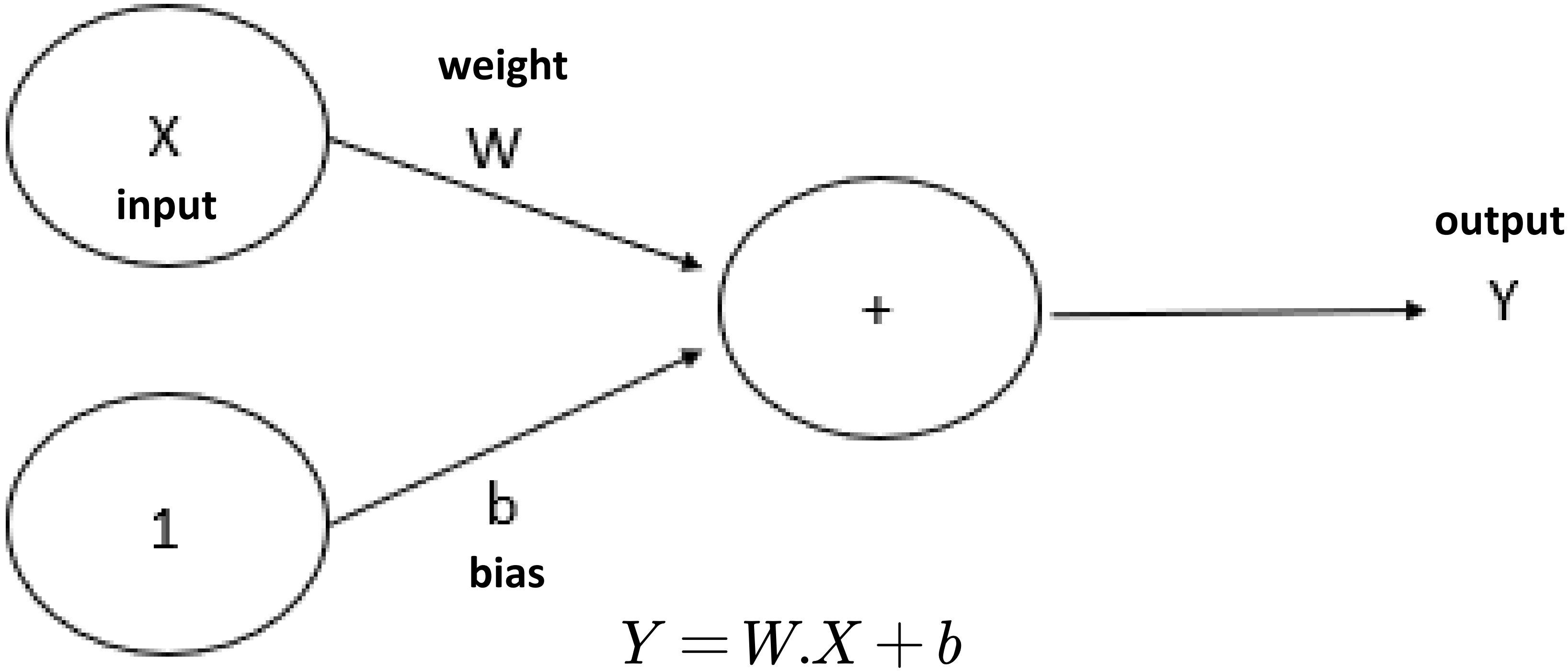
Model of an artificial neuron



Source: Google Images

Artificial Neuron Model

Simplest model of an artificial neuron



$$Y = W.X + b$$

$$y = m.x + c$$

Source: GI

Artificial Neuron Model

Simplest model of an artificial neuron

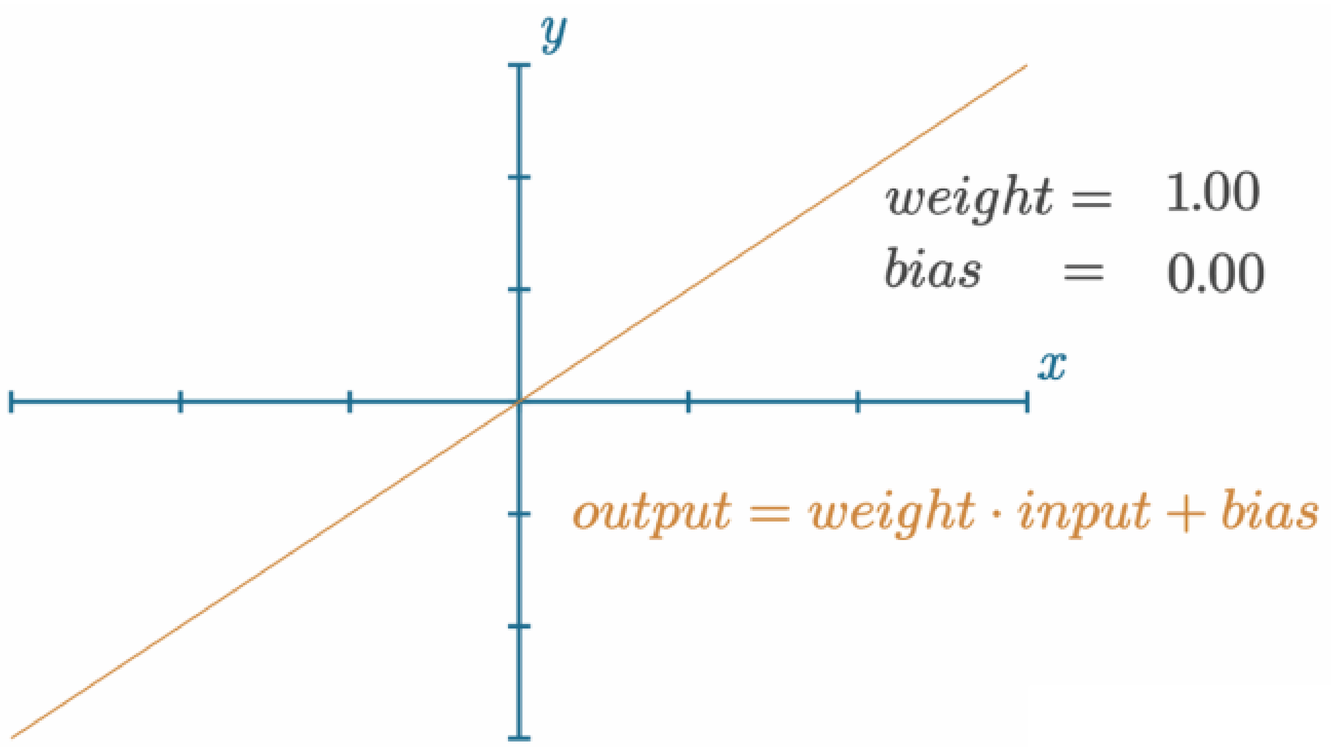


Fig: Graph of a single-input neuron's output with a weight of 1, bias of 0 and input *x*

Source: Google Images

Artificial Neuron Model

Simplest model of an artificial neuron

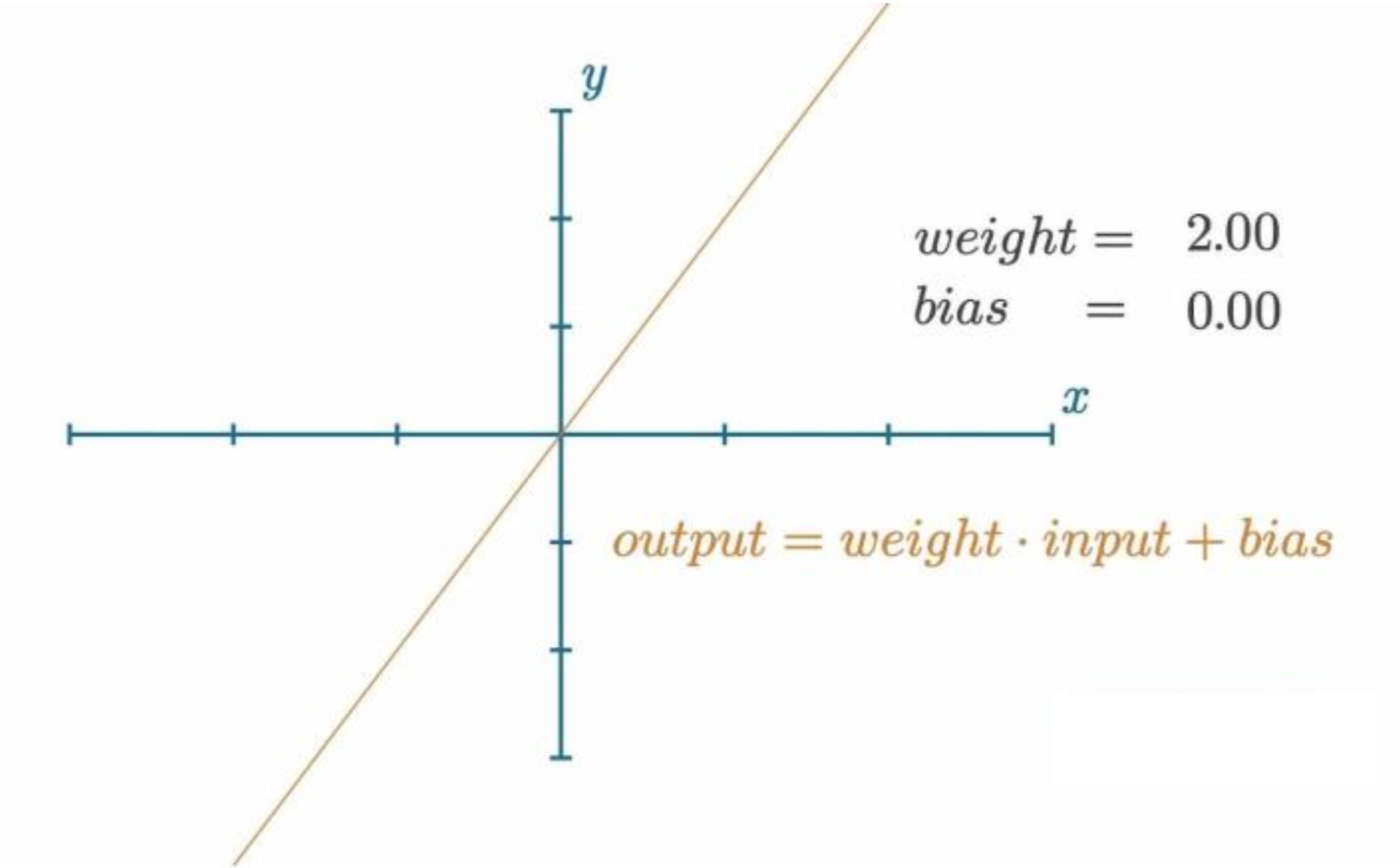


Fig: Graph of a single-input neuron's output with a given weight, bias and input *x*

Source: Google Images

Artificial Neuron Model

Simplest model of an artificial neuron

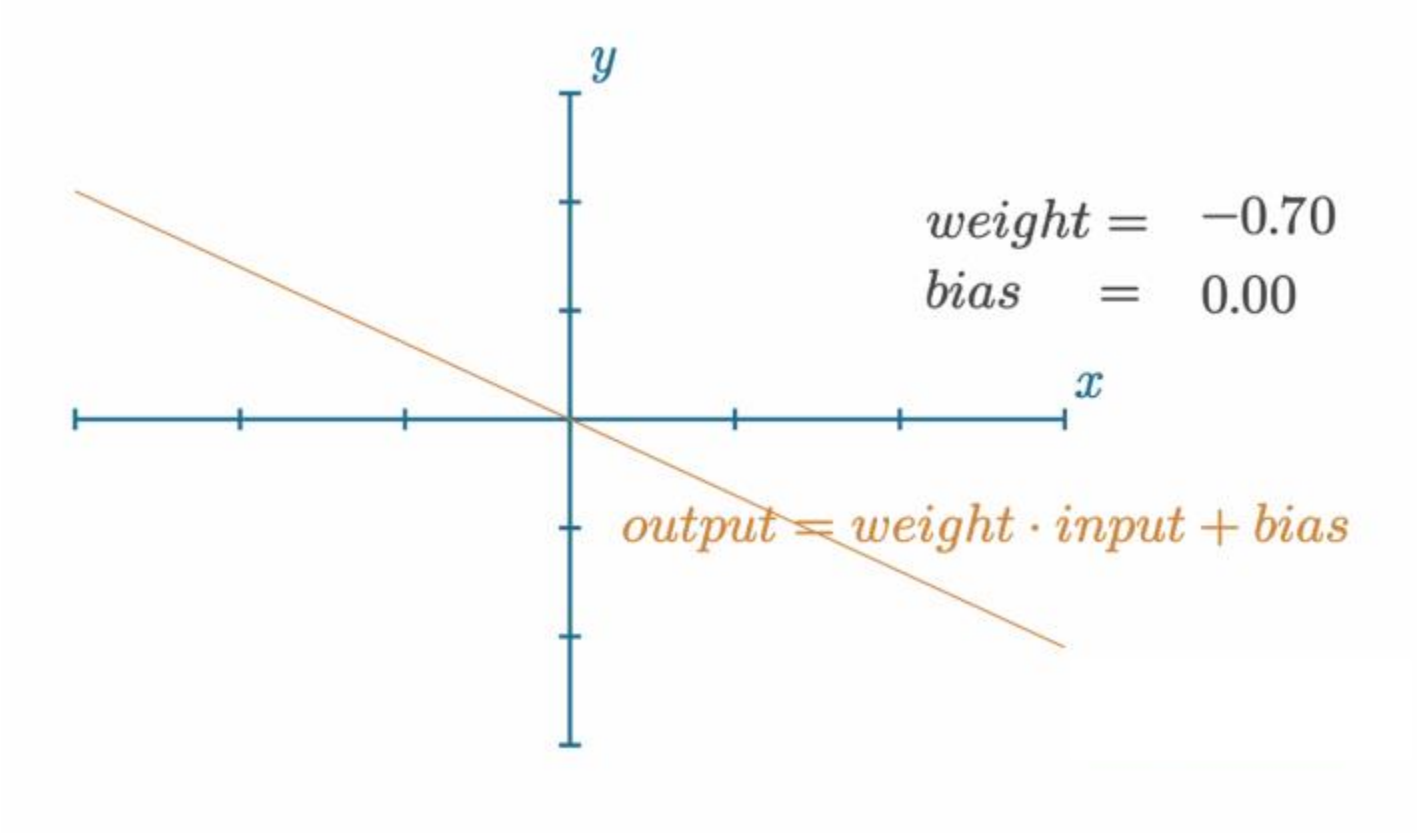


Fig: Graph of a single-input neuron's output with a given weight, bias and input *x*

Source: Google Images

Artificial Neuron Model

Simplest model of an artificial neuron

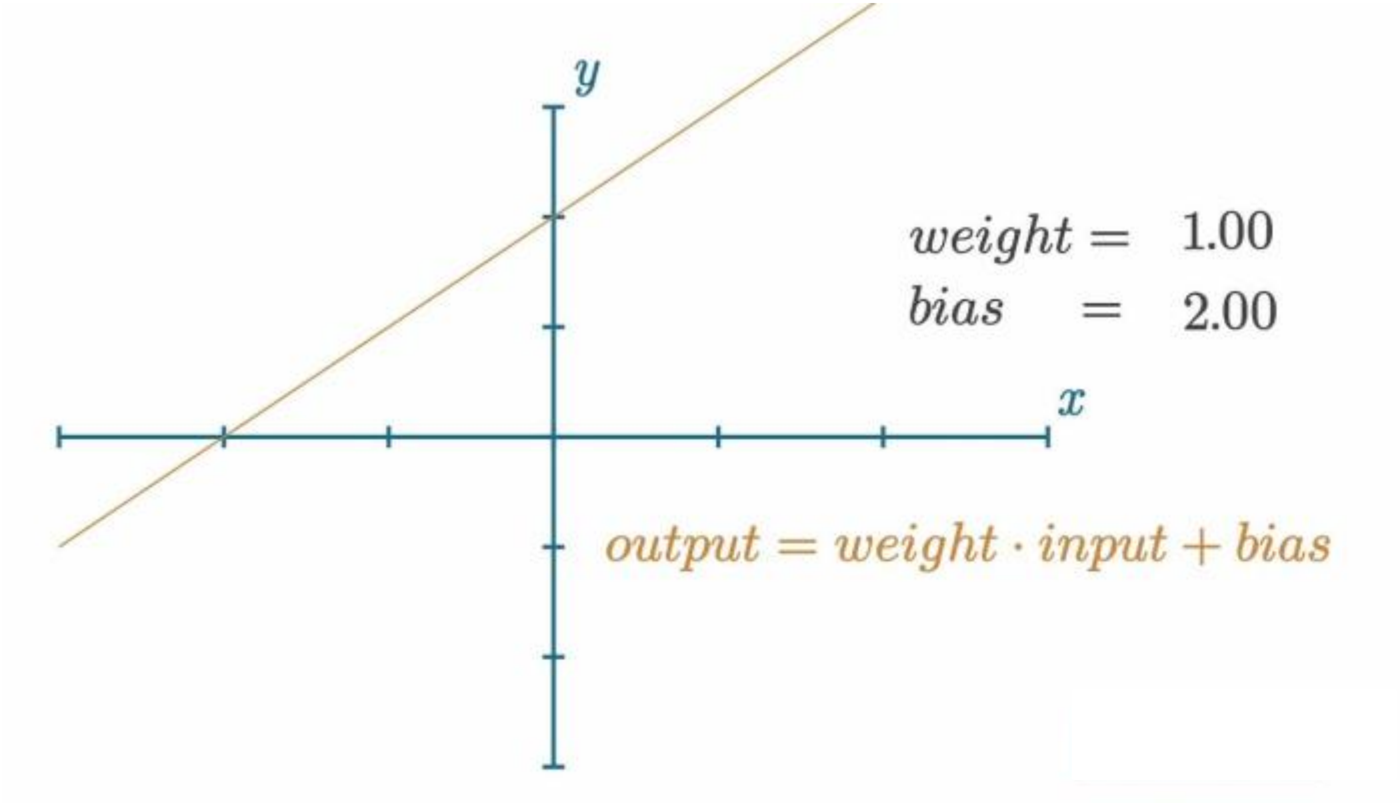


Fig: Graph of a single-input neuron's output with a given weight, bias and input *x*

Source: Google Images

Artificial Neuron Model

Simplest model of an artificial neuron

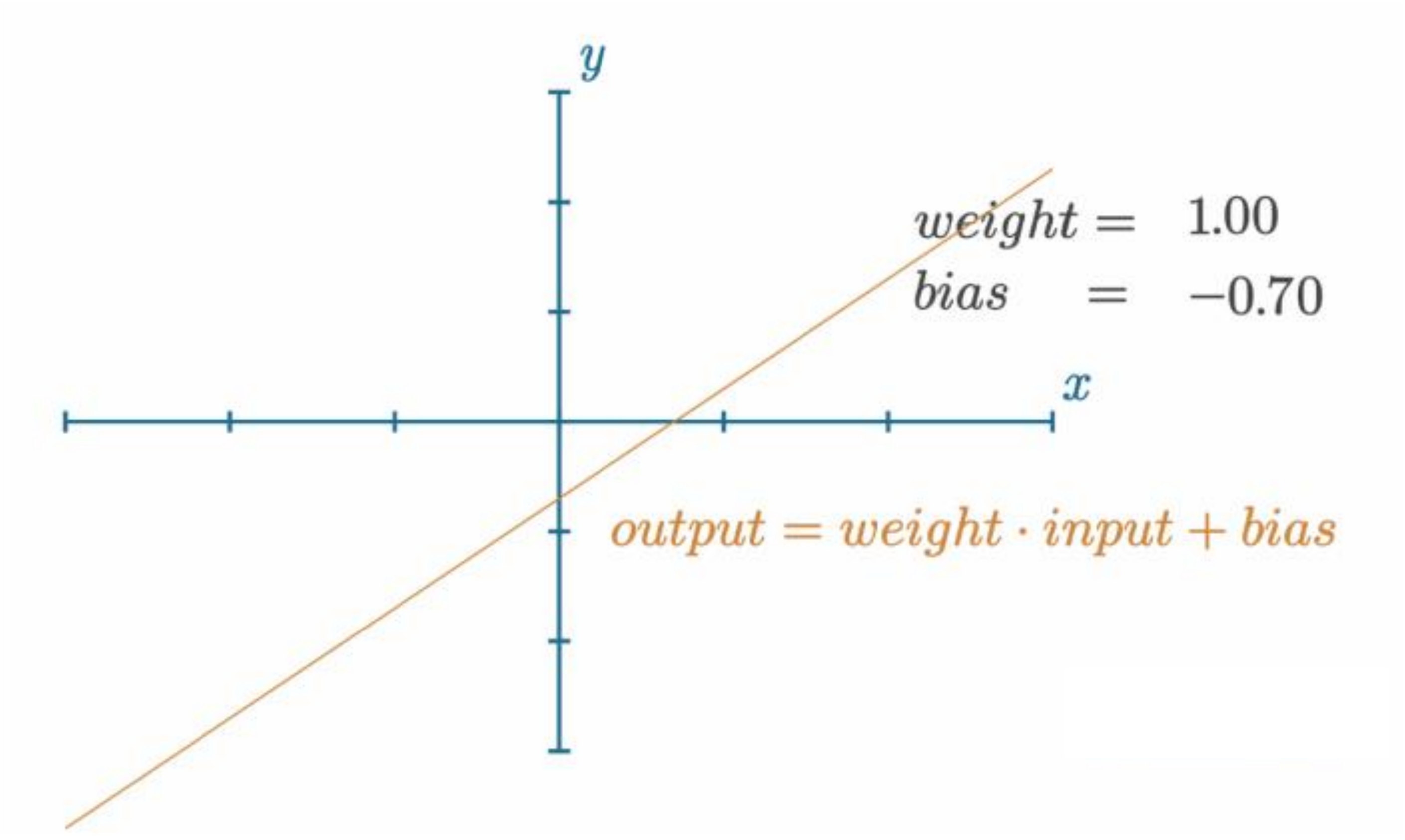
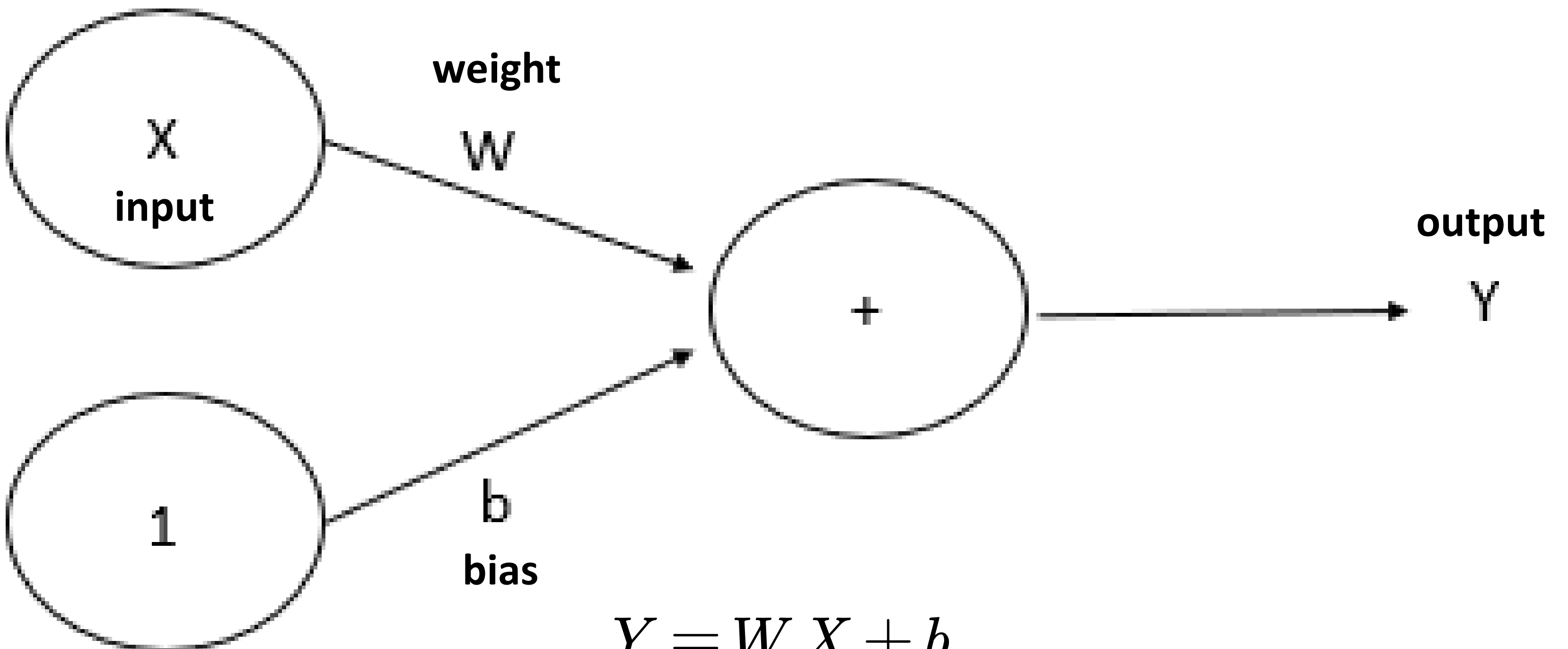


Fig: Graph of a single-input neuron's output with a given weight, bias and input *x*

Source: Google Images

Coding our First Neuron

A Single Neuron



$$Y = W.X + b$$

$$y = m.x + c$$

$$output = weight.input + bias$$

Coding our First Neuron

A Single Neuron

- Let’s say we have a single neuron

- To start with consider as:

```
input = 1
```

The weights are the parameters that we will tune to get the desired output.

```
weight = 0.5
```

The bias is an additional tunable value

```
bias = 1
```

Coding our First Neuron

A Single Neuron

- This neuron sums the input multiplied by that input's weight, then adds the bias.

```
output = input * weight + bias

print (output)
```

Coding our First Neuron

A Single Neuron

- If we have a single neuron, and there are three inputs to this neuron.

```
inputs = [ 1 , 2 , 3 ]
```

The weights are the parameters that we will tune to get the desired output.

```
weights = [ 0.2 , 0.8 , - 0.5 ]
```

The bias is an additional tunable value

```
bias = 2
```

Coding our First Neuron

A Single Neuron

- This neuron sums each input multiplied by that input's weight, then adds the bias.

```
output = (inputs[ 0 ] * weights[ 0 ] +
              inputs[ 1 ] * weights[ 1 ] +
              inputs[ 2 ] * weights[ 2 ] + bias)

print (output)
```

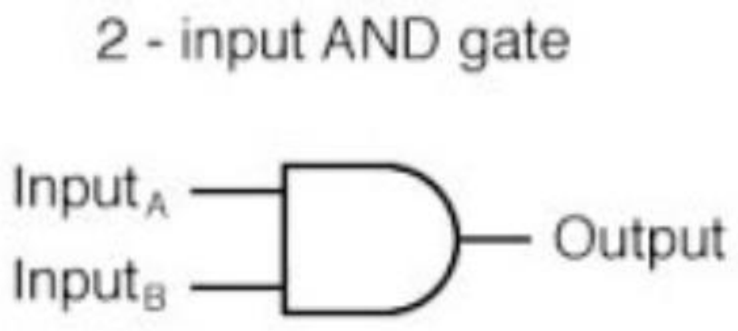
- **Exercise:** Compute the output of the neuron for the given data as:

- ```
  inputs = [ 1.0 , 2.0 , 3.0 , 2.5 ]
  weights = [ 0.2 , 0.8 , - 0.5 , 1.0 ]
  bias = 2.0
  ```

- Ans: 4.8
  ```

Artificial Neuron Model

Implementing Simple Logic Circuits: AND Gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Fig: Two Input AND Gate

Source: Google Images

Coding our First Neuron

A Single Neuron: Implementing the AND Gate:

```
input_A = 1

input_B = 1

w_A = 0.5

w_B = 0.5

bias = -0.7

out = w_A*input_A + w_B*input_B + bias

if out<0:
  out = 0
else:
  out = 1

print(out)
```

Coding our First Neuron

A Single Neuron

➢ **Implementing the AND Gate as a function:**

```
def AND_gate(input_A,input_B):

    w_A = 0.5

    w_B = 0.5

    bias = -0.7

    out = w_A*input_A + w_B*input_B + bias

    if out<0:
      return 0
    else:
      return 1
```

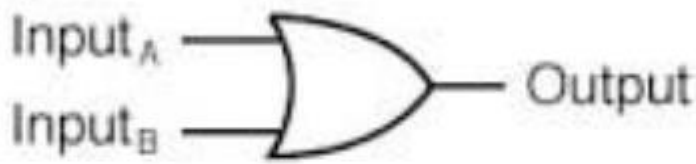
```
out = AND_gate(0,1)

print(out)
```

Artificial Neuron Model

Implementing Simple Logic Circuits: OR Gate

2 - input OR gate



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Fig: Two Input OR Gate

Source: Google Images

Coding our First Neuron

A Single Neuron: Implementing the OR Gate:

```
input_A = 1

input_B = 1

w_A = 0.5

w_B = 0.5

bias = -0.4

out = w_A*input_A + w_B*input_B + bias

if out<0:
  out = 0
else:
  out = 1

print(out)
```

Coding our First Neuron

A Single Neuron

➢ **Implementing the OR Gate as a function:**

```
def OR_gate(input_A,input_B):

    w_A = 0.5

    w_B = 0.5

    bias = -0.4

    out = w_A*input_A + w_B*input_B + bias

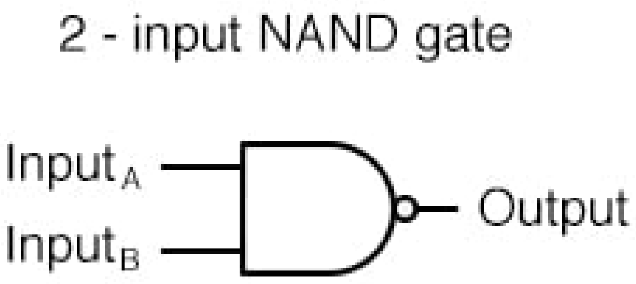
    if out<0:
      return 0
    else:
      return 1
```

```
out = OR_gate(0,1)

print(out)
```

Artificial Neuron Model

Implementing Simple Logic Circuits: NAND Gate



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Fig: Two Input NAND Gate

Source: Google Images

Coding our First Neuron

A Single Neuron: Implementing the NAND Gate:

```
input_A = 1

input_B = 1

w_A = -0.5

w_B = -0.5

bias = 0.7

out = w_A*input_A + w_B*input_B + bias

if out<0:
  out = 0
else:
  out = 1

print(out)
```

Coding our First Neuron

A Single Neuron

- Implementing the NAND Gate as a function:

```
def NAND_gate(input_A,input_B):

   w_A = -0.5

   w_B = -0.5

   bias = 0.7

   out = w_A*input_A + w_B*input_B + bias

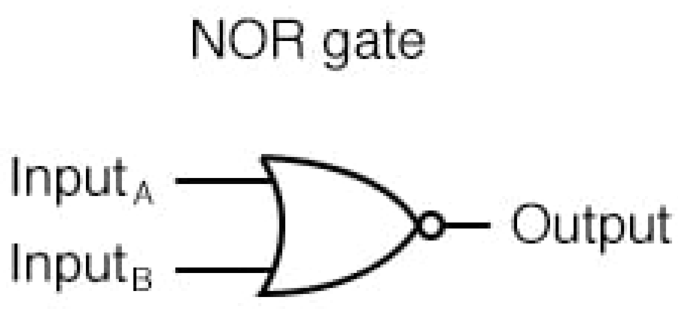
   if out<0:
     return 0
   else:
     return 1
```

```
out = NAND_gate(1,1)

print(out)
```

Artificial Neuron Model

Implementing Simple Logic Circuits: NOR Gate



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

Fig: Two Input NOR Gate

Source: Google Images

Coding our First Neuron

A Single Neuron: Implementing the NOR Gate:

```
input_A = 1

input_B = 1

w_A = -0.5

w_B = -0.5

bias = 0.4

out = w_A*input_A + w_B*input_B + bias

if out<0:
  out = 0
else:
  out = 1

print(out)
```

Coding our First Neuron

A Single Neuron

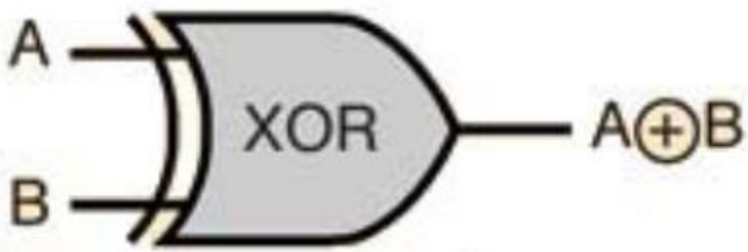
➢ Implementing the NOR Gate as a function:

```python
def NOR_gate(input_A,input_B):

    w_A = -0.5

    w_B = -0.5

    bias = 0.4

    out = w_A*input_A + w_B*input_B + bias

    if out<0:
      return 0
    else:
      return 1
```

```python
out = NOR_gate(0,1)

print(out)
```

Artificial Neuron Model

Implementing Simple Logic Circuits: XOR Gate



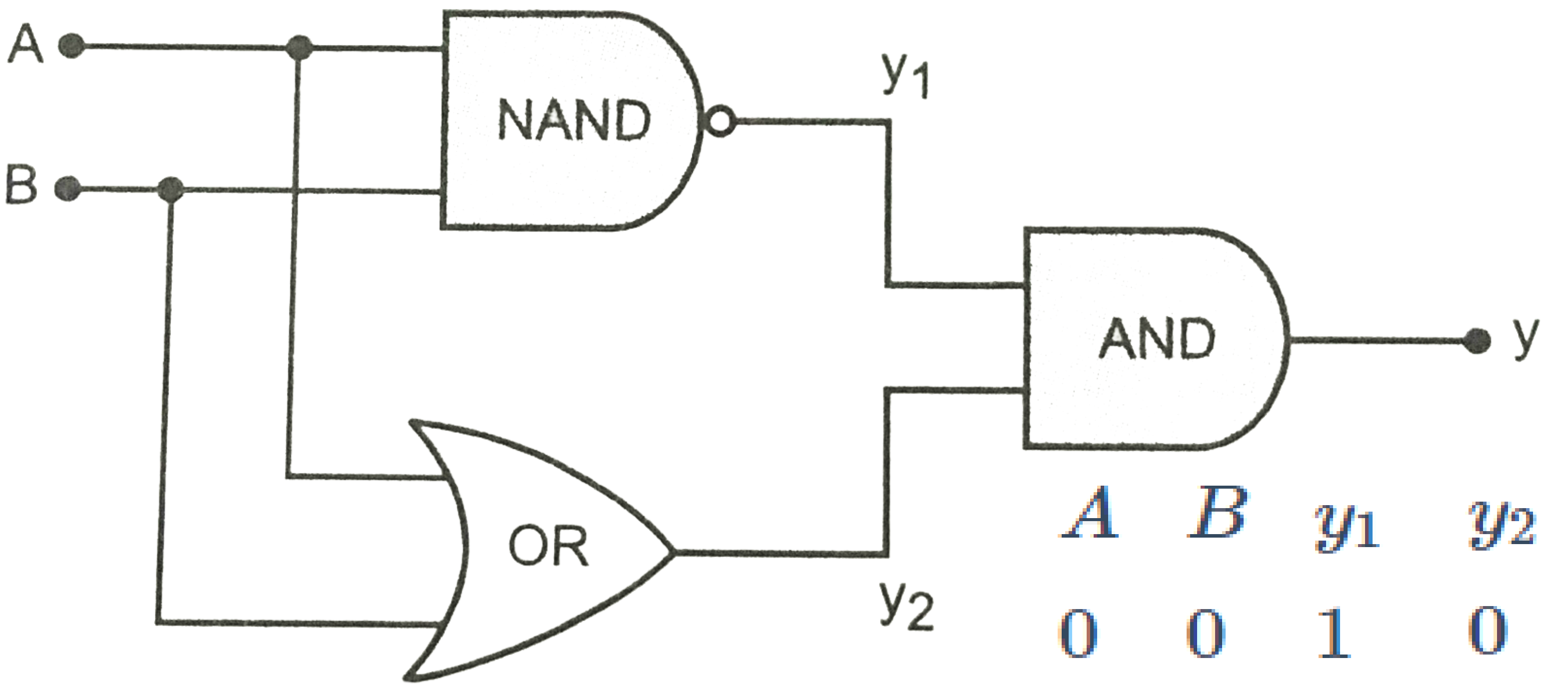
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Fig: Two Input XOR Gate

Source: Google Images

Artificial Neuron Model

Implementing Simple Logic Circuits: XOR Gate



A	B	y_1	y_2	y
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	1	0

Fig: Two Input XOR Gate using NAND, OR and AND gates

Source: Google Images

Coding our First Neuron

A Single Neuron

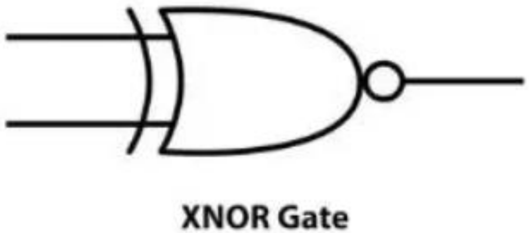
- Implementing the XOR Gate as a function:

```
def XOR_gate(input_A,input_B):

    y1 = NAND_gate(input_A,input_B)

    y2 = OR_gate(input_A,input_B)

    y = AND_gate(y1,y2)

    return y
```

```
out = XOR_gate(1,1)

print(out)
```

Artificial Neuron Model

Implementing Simple Logic Circuits: XNOR Gate



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

Fig: Two Input XNOR Gate

Source: Google Images

Coding our First Neuron

Assignment: Implement the XNOR gate using simple gates.

Artificial Neuron Model

Implementing Simple Logic Circuits: XNOR Gate

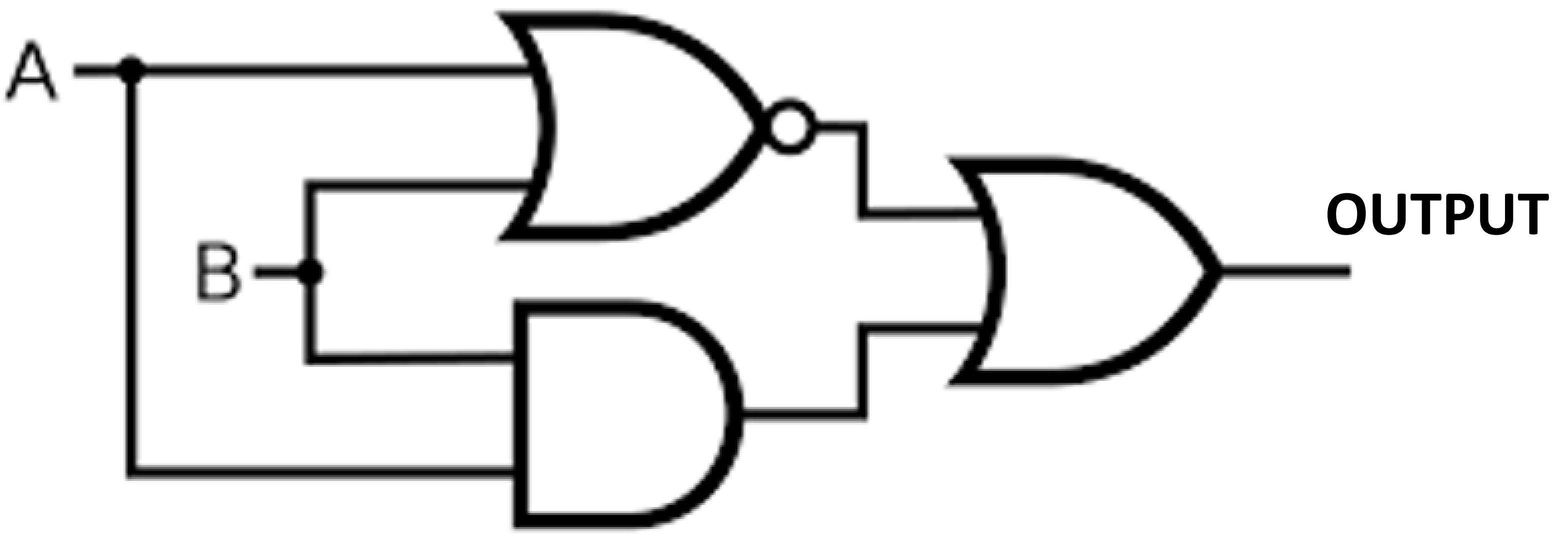


Fig: Two Input XNOR Gate using NOR, AND and OR gates

Source: Google Images