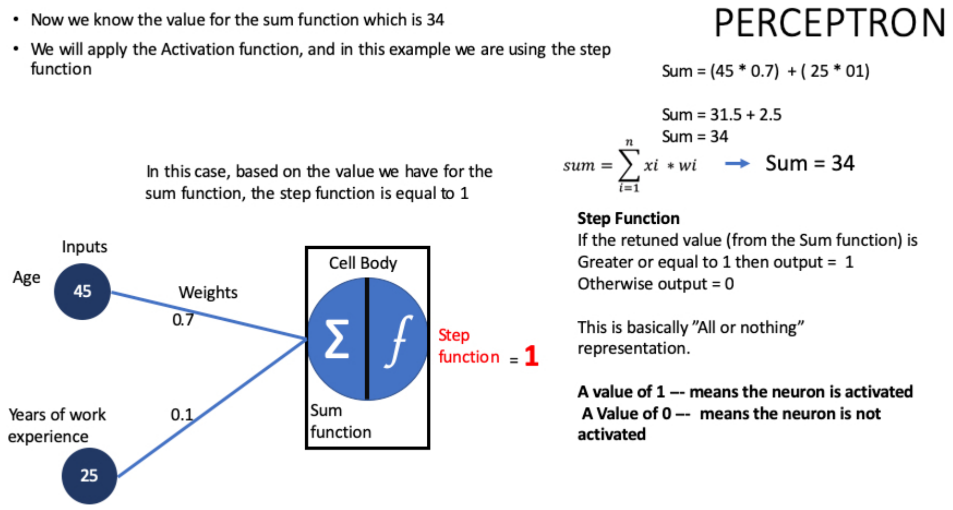
Author: Dr Mike Lakoju, CardiffMet

We are re-implementing the last code. With Neural networks, most implementation uses large datasets, and it is important to write our code in a more optimized manner. We will be using Numpy Library to achieve this.



- Now we know the value for the sum function which is 34
- We will apply the Activation function, and in this example we are using the step function

PERCEPTRON

Sum = (45 * 0.7) + (25 * 01)

Sum = 31.5 + 2.5
Sum = 34

$sum = \sum_{i=1}^{n} xi * wi$ → Sum = 34

In this case, based on the value we have for the sum function, the step function is equal to 1

Step Function
If the retuned value (from the Sum function) is
Greater or equal to 1 then output = 1
Otherwise output = 0

This is basically ”All or nothing” representation.

A value of 1 –- means the neuron is activated
A Value of 0 –- means the neuron is not activated

Import Library

```
In [ ]:
import numpy as np
```

Lets define the Inputs and weights

With NumPy, we work with arrays. Hence we will need to define our inputs as arrays

```
In [ ]:
inputs = np.array([45, 25])
```

```
In [ ]:
# Check the type of the inputs

type(inputs)
```

```
In [ ]:
# check the value at index position 0
inputs[0]
```

Lets define the weights

```
In [ ]:
# creating the weights as Numpy array

weights = np.array([0.7, 0.1])
```

```
In [ ]:
# Check the value at index 0

weights[0]
```

Create the Sum Function

The dot function is called the dot product from linear algebra. If you are dealing with a huge dataset, The processing difference between the for loop used in the last notebook and this dot product will significantly be different.

```
In [ ]:
def sum_func(inputs, weights):
    return inputs.dot(weights)
```

```
In [ ]:
# for weights = [0.7, 0.1]

s_prob1 = sum_func(inputs, weights)
s_prob1
```

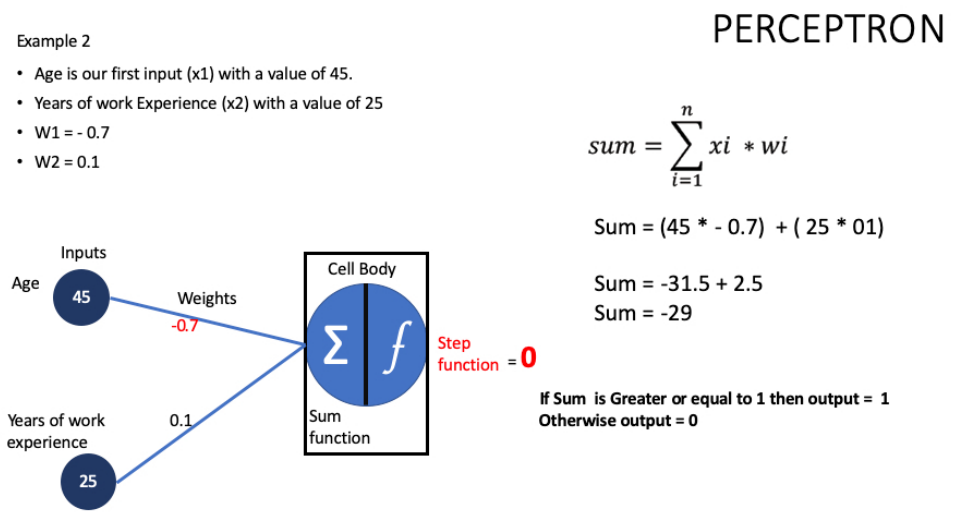
Create Step function

```
In [ ]:
def step_function(sum_func):
  if (sum_func >= 1):
    print(f'The Sum Function is greater than or equal to 1')
    return 1
  else:
        print(f'The Sum Function is NOT greater')
        return 0
```

Result

```
In [ ]:
step_function(s_prob1 )
```

If the is weights = [- 0.7, 0.1]



Example 2

- Age is our first input (x1) with a value of 45.
- Years of work Experience (x2) with a value of 25
- W1 = - 0.7
- W2 = 0.1

PERCEPTRON

$$sum = \sum_{i=1}^{n} xi * wi$$

Sum = (45 * - 0.7) + (25 * 01)

Sum = -31.5 + 2.5

Sum = -29

If Sum is Greater or equal to 1 then output = 1

Otherwise output = 0

```
weights = [-0.7, 0.1]
```

```
# for weights = [- 0.7, 0.1]

s_prob2 = sum_func(inputs, weights)

round(s_prob2, 2)  #round to 2 decimal places
```

Result

```
step_function(s_prob2 )
```

By changing the input values and weights observe different results