My first AI project

Given that,

X is input matrix which is 3×2 for 3 sets of 2 parameter values,
Y is output matrix as per inputs given in X,
W_1 is matrix of weight values between input layer and first hidden layer,
Z_2 is matrix of result of multiplication of input matrix X and matrix of weight values between input layer and first hidden layer W_1,
a_2 is activation matrix after first hidden layer or input to the output matrix,
W_3 is matrix of weight values between first hidden layer and output layer,
Z_3 is matrix of result of multiplication of matrix of weight values between input layer and first hidden layer W_1 and a_2,
$\hat{y}$ is the output estimate as per inputs provided to the system.

In [1]:

```
import numpy as np
import matplotlib.pyplot as mpl
```

In [2]:

```
x = np.array([[3,5],[5,1],[10,2]],dtype=float)
y = np.array([[72],[82],[93]],dtype=float)
```

In [3]:

```
x
```

Out[3]:

```
array([[ 3.,  5.],
       [ 5.,  1.],
       [10.,  2.]])
```

In [4]:

```
x = x/np.amax(x,axis=0)
x
```

Out[4]:

```
array([[0.3, 1. ],
       [0.5, 0.2],
       [1. , 0.4]])
```

In [5]:

```
y
```

Out[5]:

```
array([[72.],
       [82.],
       [93.]])
```

In [6]:

```
y = y/100
```

In [7]:

```
y
```

Out[7]:

```
array([[0.72],
       [0.82],
       [0.93]])
```

In [8]:

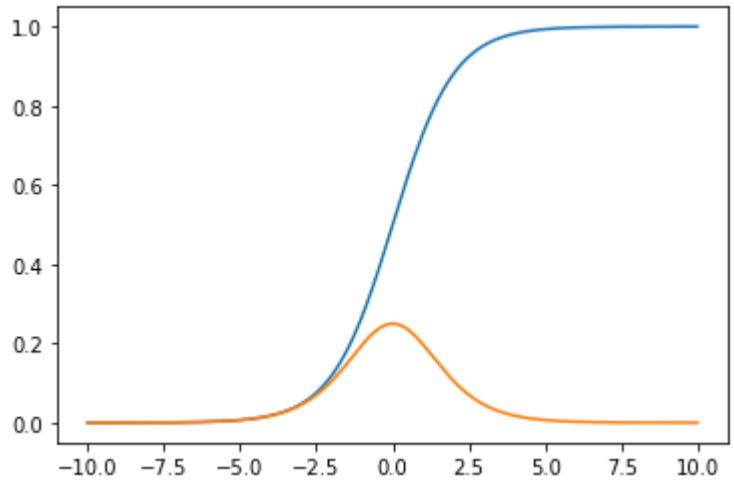
```
class Neural_Network(object):
    def __init__(self):
        self.inputLayerSize = 2
        self.outputLayerSize = 1
        self.hiddenLayerSize = 3
        self.w1 = np.random.randn(self.inputLayerSize,self.hiddenLayerSize)
        self.w2 = np.random.randn(self.hiddenLayerSize,self.outputLayerSize)

    def sigmoid(self,z):
        return 1/(1+np.exp(-z))

    def forward(self,x):
        self.z2 = np.dot(x,self.w1)
        self.a2 = self.sigmoid(self.z2)
        self.z3 = np.dot(self.a2,self.w2)
        self.yHat = self.sigmoid(self.z3)
        return self.yHat

    def sigmoidDerivative(self,z):
        return np.exp(-z)/((1+np.exp(-z))**2)

    def cost(self,x,y):
        self.yHat = self.forward(x)
        j = 0.5*np.sum((y-self.yHat)**2)
        return j
```

In [9]:

```
NN = Neural_Network()
```

In [10]:

```
test = np.arange(-10,10,0.01)
mpl.plot(test,NN.sigmoid(test))
mpl.plot(test,NN.sigmoidDerivative(test))
mpl.show()
```



In [11]:

```
yHat = NN.forward(x)
```

In [12]:

```
yHat
```

Out[12]:

```
array([[0.61102883],
       [0.57469102],
       [0.57201004]])
```

In [13]:

```
y
```

Out[13]:

```
array([[0.72],
       [0.82],
       [0.93]])
```

In [14]:

```
NN.cost(x,y)
```

Out[14]:

```
0.10010401055127335
```

In [ ]:

```

```