In [6]:

```
import numpy as np
x= np.array(([2,9],[1,5],[3,6]),dtype= float)
y=np.array(([92],[86],[89]),dtype= float)
c= np.amax(x,axis=0)
print(c)
x=x/c
y=y/100
print(x)
print(y)
```

```
[3. 9.]
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
[[0.92]
 [0.86]
 [0.89]]
```

In [7]:

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
def sigmoid_grad(x):
    return x*(1-x)
```

In [8]:

```
epoch=1000
eta=0.1
input_neurons=2
hidden_neurons=3
output_neurons=1

wh=np.random.uniform(size=(input_neurons,hidden_neurons))
print(wh)
bh=np.random.uniform(size=(1,hidden_neurons))
print(bh)
wout=np.random.uniform(size=(hidden_neurons,output_neurons))
print(wout)
bout=np.random.uniform(size=(1,output_neurons))
print(bout)
```

```
[[0.65264821 0.62887702 0.5970444 ]
 [0.59087162 0.18987399 0.74788479]]
[[0.00750303 0.31406397 0.53450128]]
[[0.64697465]
 [0.48613899]
 [0.25438819]]
[[0.79651886]]
```

In [9]:

```
for i in range(epoch):
    h_ip=np.dot(x,wh)+bh
    print(h_ip)
    h_act= sigmoid(h_ip)
    o_ip=np.dot(h_act,wout)+bout
    output= sigmoid(o_ip)

    Eo=y-output
    outgrad= sigmoid_grad(output)
    d_output = Eo* outgrad
    print("the d_output is \n",d_output)

    Eh=d_output.dot(wout.T)
    hiddengrad= sigmoid_grad(h_act)
    d_hidden= Eh *hiddengrad
    wout += h_act.T.dot(d_output)*eta
    wh += x.T.dot(d_hidden)*eta
```

```
 [1.06611355 1.07898871 1.63367929]]
the d_output is
 [[ 0.00466684]
 [-0.00020244]
 [ 0.0012267 ]]
[[1.04613638 0.93315458 1.68413881]
 [0.56010201 0.63451776 1.1510031 ]
 [1.06621843 1.07907263 1.63371177]]
the d_output is
 [[ 0.00465104]
 [-0.00021537]
 [ 0.00121378]]
[[1.04624553 0.9332421  1.68417259]
 [0.56016055 0.6345647  1.15102121]
 [1.06632269 1.07915608 1.63374408]]
the d_output is
 [[ 0.00463538]
 [-0.0002282 ]
 [ 0.00120098]]
```

In [10]:

```
print("Normalized input: \n"+str(x))
print("Actual output: \n"+ str(y))
print("Predicted output: \n",output)
```

```
Normalized input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual output:
[[0.92]
 [0.86]
 [0.89]]
Predicted output:
 [[0.89451311]
 [0.87906677]
 [0.89607708]]
```

In [ ]:

```

```