In [ ]:

```
Program-4:
Build an Artificial Neural Network by implementing the Backpropagation algorithm
and test the same using appropriate data sets.
```

In [1]:

```
import numpy as np

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X/np.amax(X,axis=0)
y = y/100

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


epoch=1000
eta =0.2
input_neurons = 2
hidden_neurons = 3
output_neurons = 1


# Weight and bias - Random initialization
wh=np.random.uniform(size=(input_neurons,hidden_neurons))
bh=np.random.uniform(size=(1,hidden_neurons))
wout=np.random.uniform(size=(hidden_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
```

In [2]:

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

#Backpropagation
    # Error at Output layer
    Eo = y-output
    outgrad = sigmoid_grad(output)
    d_output = Eo* outgrad

    # Error at Hidden later
    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

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:
 [[0.89384466]
 [0.8819897 ]
 [0.89381367]]
```

In [ ]:

```
1
```