In [61]:
```
import numpy as np
```

In [ ]:
```

```

In [69]:
```
# Initial Weight Matrix is:-
w = np.array([[0.2, 0.6, 0.4, 0.9, 0.2],
              [0.3, 0.5, 0.7, 0.6, 0.8]])


# Units
x = [0.3, 0.4]

# Learning Rate(a):
a = 0.3
```

In [ ]:
```

```

Finding Winning Cluster or Cluster having minimum Euclidian Distance

In [64]:
```
def winningCluster(w, x):
    r = w.shape[0]  # Numbers of Rows or Units
    c = w.shape[1]  # Number of Cols or Clusters
    miniDist = 999
    miniIndex = -1

    for i in range(c):
        d = 0
        for j in range(r):
            d += (w[j][i]-x[j])**2

        if(d<miniDist):
            miniDist = d
            miniIndex = i

    return miniIndex
```

In [ ]:
```

```

Updation of Weights

In [66]:
```
# Here cNum is Cluster Number
def updateWeights(w, cNum, a, x):
    r = w.shape[0]  # Numbers of Rows or Units
    for i in range(r):
        w[i][cNum] = w[i][cNum] + a*(x[i]-w[i][cNum])
```

In [ ]:
```
```

Applying SOM

In [70]:
```
cj = winningCluster(w, x)
print("Winning Cluster Number is: ", cj)
updateWeights(w, cj, a, x)
print("\nUpdated weights of Winning Cluster i.e. Cluster Number ", cj, " i

if((cj-1) < 1):
    print("\nCluster Number ", cj-1, " is not Available")
else:
    updateWeights(w, cj-1, a, x)
    print("\nUpdated weights of Cluster Number ", cj-1, " is: \n", w)

if((cj+1) >= w.shape[1]):
    print("\nCluster Number ", cj+1, " is not Available")
else:
    updateWeights(w, cj+1, a, x)
    print("\nUpdated weights of Cluster Number ", cj+1, " is: \n", w)
```

```
Winning Cluster Number is:  0

Updated weights of Winning Cluster i.e. Cluster Number  0  is:
 [[0.23 0.6  0.4  0.9  0.2 ]
 [0.33 0.5  0.7  0.6  0.8 ]]

Cluster Number  -1  is not Available

Updated weights of Cluster Number  1  is:
 [[0.23 0.51 0.4  0.9  0.2 ]
 [0.33 0.47 0.7  0.6  0.8 ]]
```