In [1]:

```
import numpy as np
```

In [ ]:

```
data=[]
res=0
for i in range(0,50):

  res=i

  #print("")
  re=np.binary_repr(res, width=10)
  temp=[]
  for k in re:
      temp.append(int(k))

  data.append(temp)

data=np.reshape(data,(50,10))
#print(data)
```

In [ ]:

```
X=data[0:30]
#print(X)
```

In [ ]:

```
y=[]
for i in range(0,8):
    y.append(3)
    #print(i)
for i in range(8,16):
    y.append(5)
for i in  range(16,24):
    y.append(1)
for i in range(24,32):
    y.append(4)
for i in range(32,50):
    y.append(2)
y=np.array(y)
y=y.reshape(50,1)
#print(y)
```

In [5]:

```
w=np.random.random((len(X[0]),5))
#w=[]
#w.append([0.2,0.8])
#w.append([0.6,0.4])
#w.append([0.5,0.7])
#w.append([0.9,0.3])
#w=np.array(w)
print(w)
```

```
[[ 0.04712075  0.31685081  0.7761121   0.66474738  0.84633299]
 [ 0.96894774  0.99236894  0.5137003   0.2293932   0.69601023]
 [ 0.03346317  0.30434964  0.44214105  0.77382455  0.86021688]
 [ 0.89515097  0.38925913  0.80338236  0.63658832  0.65623198]
 [ 0.94142971  0.16511856  0.56081129  0.36821766  0.37659815]
 [ 0.8066063   0.51215952  0.3598158   0.93598298  0.70325871]
 [ 0.16487203  0.96371276  0.32659075  0.9906597   0.45456785]
 [ 0.80017157  0.84575974  0.23982207  0.11666037  0.50248769]
 [ 0.12110832  0.63872095  0.29314095  0.10055082  0.2728437 ]
 [ 0.86330746  0.66348938  0.79003666  0.11910997  0.06305289]]
```

In [ ]:

```
print(w[:,0])
print(X[0])
alpha=0.6
radius=10.0
```

In [ ]:

```
epoch=10000
distance=[]


def dist(x):
    distance2=[]
    for l in range(0,5):
        s=0
        W=w[:,l]
        for k in range(0,10):
            #print(x[k])
            #print(W[k])
            d=np.power((x[k]-W[k]),2)
            #print(d)
            s=s+d
        #print(s)
        distance.append(s)
        distance2.append(s)

    return distance2


for i in range(0,epoch):
    print("Learning Rate=",alpha)
    for j in range(0,30):
        x=X[j]
        element=(dist(x))
        m=99999999999
        print(element)
        for j in range(0,5):
            if element[j]<m:
                m=element[j]
                key=j
        uw=w[:,key]
        for p in range(0,len(uw)):
            w[p][key]=w[p][key]+alpha*(x[p]-w[p][key])


        #print(w)

    alpha=alpha/2




#print(distance)
#print(w)
```

In [ ]:

```
ww=w
for h in range(0,10):
    for q in range(0,5):
        #print(ww[h][q])
        if(ww[h][q]>0.0000009 and ww[h][q]<0.000001):
            print(0.1)
            ww[h][q]=0.1
        elif ww[h][q]>0.000001 and ww[h][q]<=0.0001:
            ww[h][q]=0.2
            print(0.2)
        elif ww[h][q]>0.0001 and ww[h][q]<=0.001:
            ww[h][q]=0.4
            print(0.4)
        elif ww[h][q]>0.001 and ww[h][q]<=0.01:
            ww[h][q]=0.5
```

```
            ww[h][q]=0.5
            print(0.5)
        elif ww[h][q]>0.01 and ww[h][q]<=0.09:
            ww[h][q]=0.6
            print(0.6)
        elif ww[h][q]>=0.1 and ww[h][q]<=0.3:
            ww[h][q]=0.7
            print(0.7)
        elif ww[h][q]>0.3 and ww[h][q]<=0.5:
            ww[h][q]=0.8
            print(0.8)
        elif ww[h][q]>0.5 and ww[h][q]<=0.8:
            ww[h][q]=0.9
            print(0.9)
        elif ww[h][q]>0.8 and ww[h][q]<=1:
            ww[h][q]=1
            print(1)
        else:
            ww[h][q]=0
            print(0)


#print(ww)
```

In [10]:

```
w=ww
cluster=[]
for j in range(0,30):
        x=X[j]
        m=99999999999
        element=dist(x)
        #print(element)
        for k in range(0,5):
            if element[k]<m:
                m=element[k]
                key=k
        cluster.append(key+1)
print(cluster)
```

```
[3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 2, 2, 2, 2, 2, 2]
```

In [ ]:

```
cluster=np.array(cluster)
cluster=cluster.reshape(30,1)
output=np.concatenate((X, cluster), axis=1)
#print(output)
```

In [21]:

```
cnt=0
for i in range(0,30):
        if(cluster[i]==y[i]):
            cnt=cnt+1

accuracy=(cnt/30)*100
print("Testing accuracy=", accuracy,'%')
```

```
Testing accuracy= 26.666666666666668 %
```

In [ ]: