El método de k-means

In [1]:
```
import numpy as np
```

In [2]:
```
data = np.random.random(90).reshape(30,3)
data
```

Out[2]:
```
array([[0.92109987, 0.56906258, 0.64727939],
       [0.55919123, 0.05172538, 0.21932999],
       [0.90805456, 0.5383852 , 0.16381645],
       [0.8090016 , 0.42125809, 0.04388641],
       [0.23760919, 0.32983236, 0.54654162],
       [0.68431052, 0.64381587, 0.97410997],
       [0.26730938, 0.9336947 , 0.79864046],
       [0.44484324, 0.0634494 , 0.80993724],
       [0.99441921, 0.38040239, 0.66742724],
       [0.57553982, 0.31124665, 0.0971625 ],
       [0.61281466, 0.9922274 , 0.50339431],
       [0.25428251, 0.95374404, 0.43875459],
       [0.43561642, 0.99890634, 0.58883071],
       [0.32330891, 0.17585399, 0.74128556],
       [0.24546038, 0.4298249 , 0.64882829],
       [0.40286689, 0.10936101, 0.65226585],
       [0.6201188 , 0.15101004, 0.92115459],
       [0.21303982, 0.75591633, 0.44083139],
       [0.93205942, 0.61510666, 0.21089914],
       [0.55640153, 0.2802549 , 0.49530155],
       [0.21831764, 0.37198329, 0.16882207],
       [0.54363578, 0.56587905, 0.09779374],
       [0.54558932, 0.67453735, 0.92377643],
       [0.08621554, 0.39581937, 0.41102462],
       [0.32108886, 0.86298699, 0.82819526],
       [0.99993061, 0.69584234, 0.29320292],
       [0.68524438, 0.58492163, 0.39674957],
       [0.8128288 , 0.82506168, 0.66255047],
       [0.27363081, 0.2597944 , 0.25849375],
       [0.9944207 , 0.34777552, 0.23274862]])
```

In [23]:
```
c1 = np.random.choice(range(len(data)))
c2 = np.random.choice(range(len(data)))
clust_centers = np.vstack([data[c1], data[c2]])
clust_centers, c1, c2, data[c1]
```

Out[23]:
```
(array([[0.08621554, 0.39581937, 0.41102462],
        [0.40286689, 0.10936101, 0.65226585]]),
 23,
 15,
 array([0.08621554, 0.39581937, 0.41102462]))
```

In [24]:
```
from scipy.cluster.vq import vq
```

In [25]:
```
vq(data, clust_centers)
```

Out[25]:
```
(array([1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0,
        1, 0, 0, 1, 1, 1, 0, 1]),
 array([0.69275957, 0.46388858, 0.82332202, 0.79520484, 0.21363329,
```

```
0.68442396, 0.68727789, 0.16949973, 0.65086668, 0.58745232,
0.80096275, 0.58334844, 0.71931208, 0.13665761, 0.28821152,
0.        , 0.348187  , 0.3829395 , 0.85476779, 0.27823622,
0.2769137 , 0.57988546, 0.64304897, 0.        , 0.66891137,
0.91069842, 0.60924832, 0.82486483, 0.27729577, 0.76339511]))
```

Como se ve se vom el método **vq** se generan dos arrays, donde el primero, indica a qué cluster pertenece cada elemento, y el segundo indica la distancia que está cada elemento al varicentro de su clúster correspondiente, ej: 1-0.34444813 y así. **NOTA:** aquellos valores culla distancia sea 0 es porque **son** los varicentros.

In [26]:
```
from scipy.cluster.vq import kmeans
```

In [27]:
```
kmeans(data, clust_centers)
```

Out[27]:
```
(array([[0.33704242, 0.62706695, 0.5908235 ],
        [0.73497853, 0.40692141, 0.4132191 ]]),
 0.3930267540534248)
```

Con el método kmeans de scipy tenemos los dos varicentros de cada cluster, así como también un número al final que representa la suma de la diferencia de los cuadrados de las distancias normalizado, dividida la distancia total.

In [34]:
```
kmeans(data,2)
```

Out[34]:
```
(array([[0.79601851, 0.5306842 , 0.32586467],
        [0.36058881, 0.4935756 , 0.626282  ]]),
 0.37578415402644)
```

In [ ]:
```

```