Aprenentatge computacional

PAC1

Luis Enrique Arribas Zapater 17745245D

Abril 2018

Exercici 1

1. Los datos son todos de tipo numérico y de muy diferentes órdenes de magnitud como vemos a continuación.

```
             mean         sd n
ALCOHOL    13.01500  0.8833055 8
ASH         2.22750  0.4300747 8
CLASS       0.50000  0.5345225 8
COLOR       3.83875  1.6180450 8
MAGNESIUM 105.25000 14.5773797 8
PHENOLS     2.60875  0.4769378 8
```

Debemos normalizar de los datos, ya que si no aquellos de mayores magnitudes tendrían un peso desproporcionado. Si usásemos por ejemplo un espacio bidimensional los puntos quedarían distribuidos a lo largo de uno de los ejes sin prácticamente variación perceptible en el otro.

Vemos los datos normalizados por estandarización (restando a la media muestral y dividiendo por la desviación típica):

```
                      mean sd n
Z.ALCOHOL    -2.636780e-16  1 8
Z.ASH        -4.510281e-17  1 8
Z.CLASS       0.000000e+00  1 8
Z.COLOR      -1.110223e-16  1 8
Z.MAGNESIUM   3.469447e-18  1 8
Z.PHENOLS    -3.538836e-16  1 8
```

2.

```
> kmeans (Dataset, 2, iter.max=10)
K-means clustering with 2 clusters of sizes 5, 3
```

El algoritmo crea dos conjuntos en torno a dos centroides aleatorios sobre dos de los elementos (filas) y calcula para cada elemento el centroide más próximo y recalculando el centroide a continuación. Este proceso se repite hasta la convergencia en la que se obtienen los centroides finales:

```
Cluster means:
   Z.ALCOHOL      Z.ASH    Z.CLASS    Z.COLOR Z.MAGNESIUM  Z.PHENOLS

1 -0.6351144 -0.3104112  0.5612486 -0.6555751  -0.6757044 -0.3034987
2  1.0585239  0.5173519 -0.9354143  1.0926251   1.1261740  0.5058312
```

Y las categorías

```
Clustering vector:
[1] 2 1 1 2 1 1 2 1
```

Categoría 1:{2, 3, 5, 6, 8}
Categoría 2:{1, 4, 7}

Para calcular la precisión comparamos el atributo “class” con la categoría asignada por el algoritmo k-means.

n	Z.CLASS	Cluster
1	2	2
2	1	1
3	1	1
4	2	2
5	1	1
6	2	1
7	2	2
8	1	1

Por tanto la precisión del algoritmo es:

$$precisión = \frac{4+3}{8} \cdot 100 = 87{,}5\%$$

3. Aplicamos el análisis de componentes principales (PCA) y obtenemos los porcentaje de las varianzas que corresponden a cada componente principal.

```
0.6796859 0.1713183 0.1134574 0.03191806 0.003440555 0.000179866
```

El porcentaje de varianza acumulado es

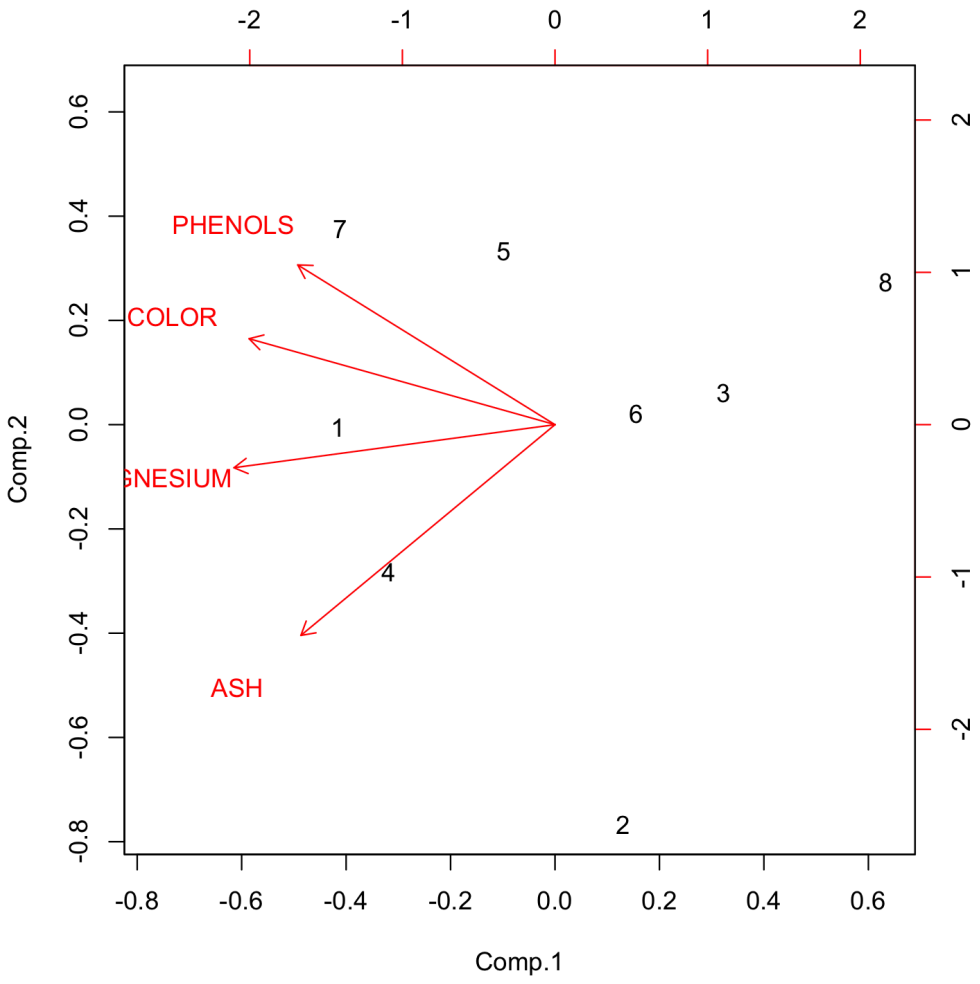
```
0.6796859 0.8510041 0.9644615 0.99637958 0.999820134 1.000000000
```

Por tanto las **tres primeras componentes** (en rojo) conservan más del 95% de la varianza, luego descartamos las componentes 4 a 6 y obtenemos la matriz:

```
Component loadings:
                Comp.1      Comp.2      Comp.3
Z.ALCOHOL    -0.4444856  0.41986771  0.09318856
Z.ASH        -0.3096503 -0.44785306  0.74569516
Z.CLASS       0.4393383 -0.36895687  0.12284238
Z.COLOR      -0.4744324  0.05355777 -0.31084711
Z.MAGNESIUM -0.4615742 -0.10053666  0.16944406
Z.PHENOLS    -0.2745801 -0.68850092 -0.54297989
```

Donde podemos observar destacadas en esas tres primeras componentes principales los valores absolutos de sus coeficientes de mayor valor, o sea los más representativos que son COLOR y MAGNESIUM en la primera componente (que tiene 67% de la varianza) y PHENOLS y ASH en la segunda y tercera respectivamente.

En la gráfica vemos como vector cada uno de los coeficientes (PHENOLS; COLOR,…) Las componentes de cada vector indican el valor para dicho coeficiente en cada una de las dos primeras componentes principales representadas.



Aplicando k-Means a la matriz reducida de datos obtenemos unos nuevos centroides

```
> kmeans(Dataset, 2, iter.max=2)
K-means clustering with 2 clusters of sizes 3, 5

Cluster means:
        PC1          PC2         PC3
1 -2.371934  0.16559666 -0.05767648
2  1.423160 -0.09935799  0.03460589

Clustering vector:
[1] 1 2 2 1 2 2 1 2
```

Las categorías son entonces

Cat 1: {2, 3, 5, 6, 8}
Cat2: {1, 4, 7}

Y comparando con el atributo “class”

n	Z.CLASS	Cluster
1	2	2
2	1	1
3	1	1
4	2	2
5	1	1
6	2	1
7	2	2
8	1	1

Como conclusión vemos que la reducción de dimensionalidad ha preservado la información con suficiente precisión como para devolver el mismo resultado que realizando el k-means sobre el doble de columnas.

Exercici 2

1. En primer lugar, con los datos anteriormente estandarizados, creamos una matriz de distancias y especificamos que la medida de distancia se tome de forma euclidea.

```
> list( dist(Dataset, method="euclidean"))
[[1]]
          1         2         3         4         5         6         7
2 4.1892842
3 4.4602143 2.0384770
4 0.8309208 3.5772196 4.0726456
5 4.1365700 2.8079428 2.4169482 3.9397857
6 2.9124248 2.8613314 2.4153417 2.4501406 3.2560150
7 1.3243970 4.1726324 4.0890206 1.6302332 3.3875722 2.6496014
8 5.4257157 3.3603798 1.6650899 5.1081715 3.8743796 3.2203335 5.1810874
```

El proceso de agrupación por **enlace simple** es:

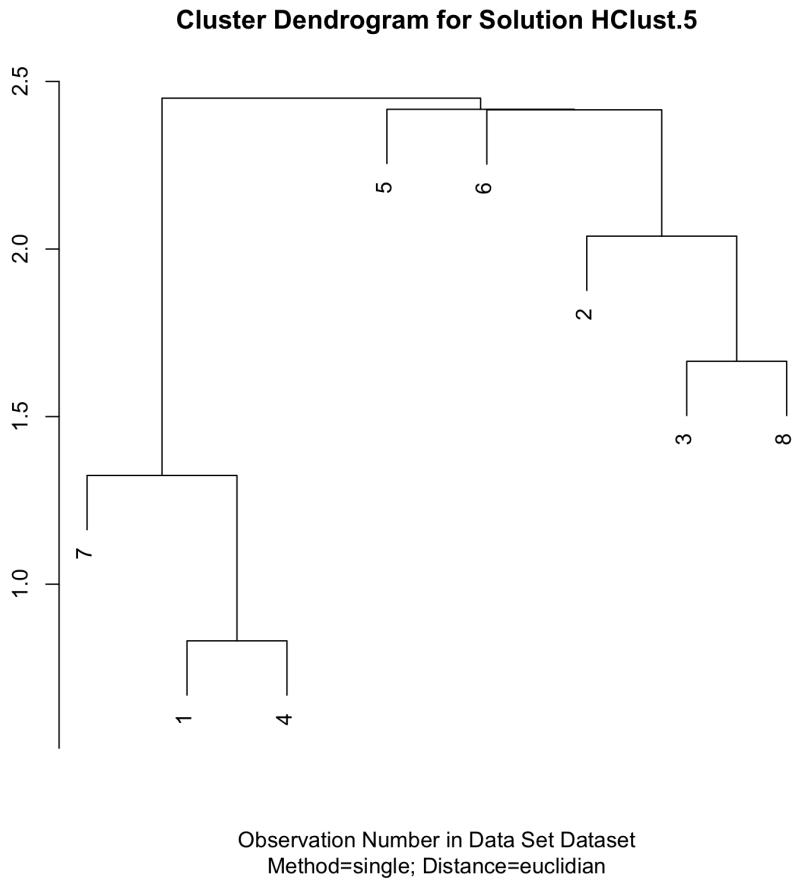
{1}
{1, 4}
{1, 4, 7}
{3, 8}
{2, 3, 8}
{6, 2, 3, 8}
{5, 6, 2, 3, 8}
{1, 4, 7, 5, 6, 2, 3, 8}

Y para obtener dos categorías el orden de agrupación nos da el resultado:

Cat 1: {2, 3 ,5, 6, 8}
Cat 2: {1, 4, 7}

Resultado que coincide con las categorizaciones del ejercicio 1, por lo que tiene una **precisión del 87,5%**

A continuación vemos en dendrograma



El proceso de agrupación por **enlace completo** es:

{1}
{1, 4}
{7, 1, 4}
{3, 8}
{2, 5}
{6, 7, 1, 4}
{2, 5, 3, 8}
{6, 7, 1, 4, 2, 5, 3, 8}

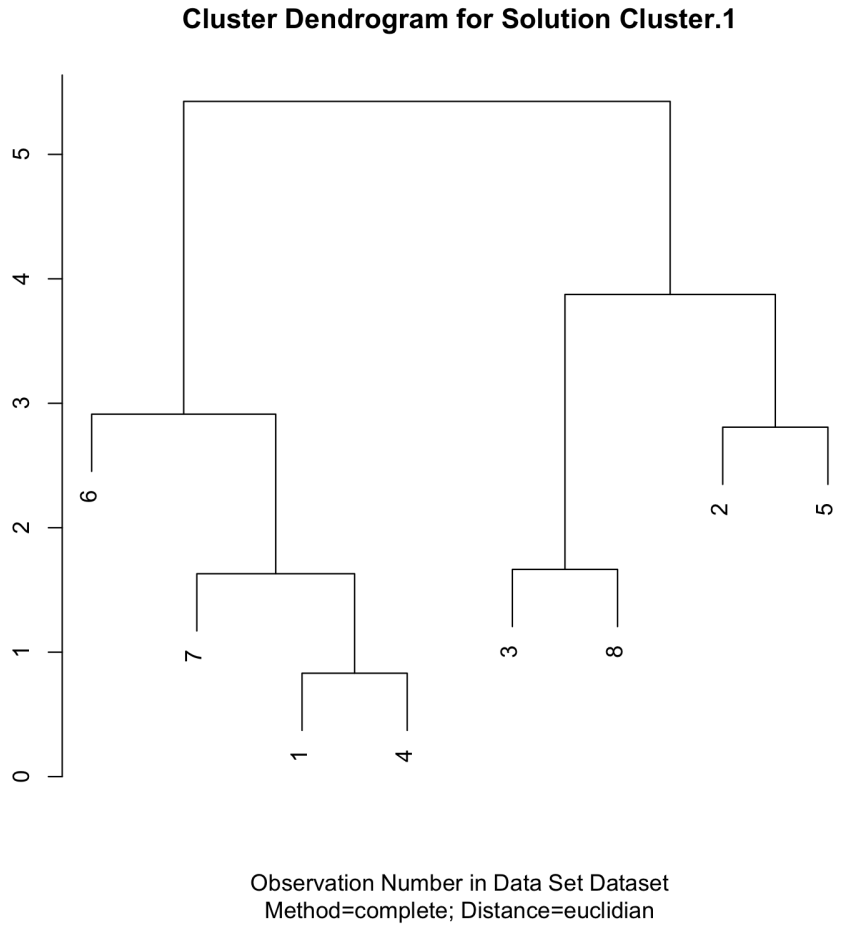
Y para obtener dos categorías el orden de agrupación nos da el resultado:

Cat 1: {2, 3, 5, 8}
Cat 2: {1, 4, 6, 7}

Calculamos la precisión del algoritmo

$$precisión = \frac{4+34}{8} \cdot 100 = \mathbf{100\%}$$

A continuación vemos en dendrograma



Finalmente ejecutamos el algoritmo usando el criterio de distancia por medias aritméticas. El algoritmo realiza las agrupaciones de la siguiente forma:

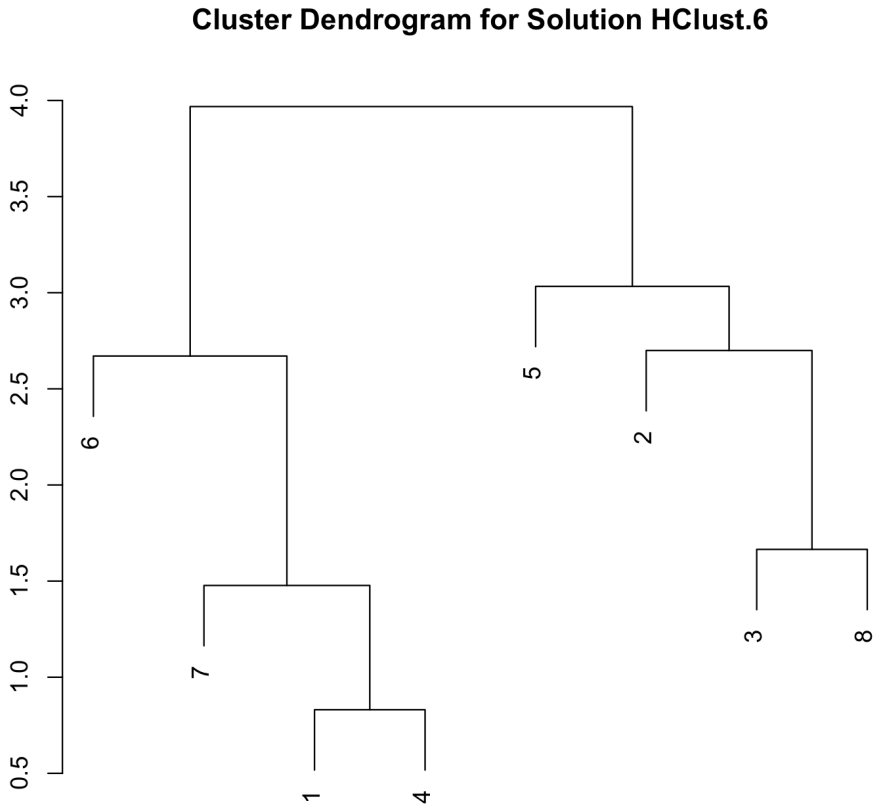
{1}
{1, 4}
{7, 1, 4}
{3, 8}
{6, 7, 1, 4}
{2, 3, 8}
{5, 2, 3, 8}
{6, 7, 1, 4, 5, 2, 3, 8}

La categorización queda de la siguiente forma:

Cat 1: {2, 3, 5, 8}
Cat 2: {1, 4, 6, 7}

Por lo que la precisión es del 100% como en el caso anterior.

A continuación vemos el dendrograma



Exercici 3

1. Con el valor 'random' como segundo parámetro los centroides iniciales son escogidos aleatoriamente entre la población. El algoritmo tiene una precisión de 6,15%

```
kMeansOut=KMeans(n_clusters=2,init='random',max_iter=25).fit(sampleData)
```

```
runfile('/Users/luisarribas/Desktop/kmeans y FuzzyCmeans/
kmeans_sklearn.py', wdir='/Users/luisarribas/Desktop/kmeans y
FuzzyCmeans')
[K-MEANS OUTPUT]
 - CLASS 0 MEMBERS          : [60, 61, 62, 63, 65, 66, 68, 69, 70,
71, 73, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94, 95, 97, 98, 100, 101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119,
120, 121, 123, 124, 125, 126, 127, 128, 129, 130]
 - CLASS 0 CENTROID          : [-0.80 0.04 -0.38 0.36 -0.46 -0.70
-0.74 0.44 -0.34 -0.76 -0.08 -0.45
 -0.82]
 - CLASS 1 MEMBERS          : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
64, 67, 72, 74, 75, 96, 99, 122]
 - CLASS 1 CENTROID         : [0.75 -0.04 0.35 -0.34 0.43 0.66 0.70
-0.41 0.32 0.72 0.07 0.42 0.77]
[K-MEANS PERFORMANCE]
 - ACCURACY                 : 6.153846153846154%
```

2. Con el valor 'k.means++' como segundo parámetro el algoritmo se ejecuta mejorando el método de selección de centroides iniciales aleatorio y acelera la convergencia. Además vemos que la precisión mejora hasta 93,84

```
kMeansOut=KMeans(n_clusters=2,init='k-means'
+',max_iter=25).fit(sampleData)
```

```
runfile('/Users/luisarribas/Desktop/kmeans y FuzzyCmeans/
kmeans_sklearn.py', wdir='/Users/luisarribas/Desktop/kmeans y
FuzzyCmeans')
[K-MEANS OUTPUT]
 - CLASS 0 MEMBERS          : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 64, 67, 72, 74, 75, 96, 99, 122]
```

```
 - CLASS 0 CENTROID        : [0.75 -0.04 0.35 -0.34 0.43 0.66 0.70 -0.41
0.32 0.72 0.07 0.42 0.77]
 - CLASS 1 MEMBERS         : [60, 61, 62, 63, 65, 66, 68, 69, 70, 71, 73,
76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
94, 95, 97, 98, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 123, 124, 125,
126, 127, 128, 129, 130]
 - CLASS 1 CENTROID        : [-0.80 0.04 -0.38 0.36 -0.46 -0.70 -0.74 0.44
-0.34 -0.76 -0.08 -0.45
 -0.82]
[K-MEANS PERFORMANCE]
 - ACCURACY                : 93.84615384615384%
```

3. Finalmente categorizamos en 3 clusters. La precisión baja a 41,53%.

```
kMeansOut=KMeans(n_clusters=3,init='k-
means+',max_iter=10).fit(sampleData)
```

```
[K-MEANS OUTPUT]
 - CLASS 0 MEMBERS         : [5, 22, 25, 26, 42, 44, 64, 72, 74, 75, 80,
82, 85, 94, 95, 96, 99, 100, 103, 110, 111, 121, 122, 123, 124, 125, 126,
127]
 - CLASS 0 CENTROID        : [-0.53 0.70 0.34 0.68 0.05 0.42 0.28 -0.25
0.47 -0.52 -0.32 0.46 -0.57]
 - CLASS 1 MEMBERS         : [60, 61, 62, 63, 65, 66, 68, 69, 70, 71, 73,
76, 77, 78, 79, 81, 83, 84, 86, 87, 88, 89, 90, 91, 92, 93, 97, 98, 101,
102, 104, 105, 106, 107, 108, 109, 112, 113, 114, 115, 116, 117, 118,
119, 120, 128, 129, 130]
 - CLASS 1 CENTROID        : [-0.76 -0.26 -0.49 0.33 -0.45 -1.00 -1.02
0.63 -0.63 -0.75 0.11 -0.71
 -0.78]
 - CLASS 2 MEMBERS         : [1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 23, 24, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 67]
 - CLASS 2 CENTROID        : [0.93 -0.12 0.26 -0.62 0.37 0.66 0.75 -0.42
0.32 0.91 0.06 0.40 0.97]
[K-MEANS PERFORMANCE]
 - ACCURACY                : 41.53846153846154%
```

Exercici 4

K.Means nítido VS. método aglomerativo.

En los ejercicio 1 y 3 se ha usado el algoritmo k-means (nítido) para clasificar un conjunto de datos n-dimensionales, mientras que en el ejercicio 2 hemos usado un método aglomerativo para construir los clusters. Veamos brevemente el funcionamiento de ambos algoritmos para entender las diferencias.

1. K-means selecciona (mediante diferentes métodos posibles) dos o más elementos del conjunto de datos donde fija los centroides de los respectivos clusters o categorías. Así para n clusters se seleccionarán n centroides.

2. Para cada elemento e se calcula la distancia a cada uno de los n centroides. e es categorizado en el cluster a menor distancia (con métrica “euclidea” o “manhattan” según sea el caso) y se recalcula el centroide del cluster teniendo en cuenta el nuevo componente para el centro de masas común.

3. De esta forma los n centroides se desplazan en cada iteración y debe por tanto recalcularse para todos lo elementos la pertenencia a cada cluster. Para grandes volúmenes de datos esto puede suponer problemas de computación por lo que se puede limitar (parámetros iter.max y n_jobs el número de iteraciones a costa de perder precisión en favor de la tratabilidad del problema).
   Cuando se alcanza un estado en el que los centroides no se desplazan más y se alcanza una situación “estable” se ha alcanzado la convergencia y el algoritmo finaliza.

El método aglomerativo se fundamenta en la construcción de una matriz de semejanzas (o de distancias como es el caso del exercici 2).

1. Se selecciona el campo (distancia mínima) de la matriz y se agregan por orden al cluster los elementos correspondientes a dicha distancia mínima e.

2. Dependiendo del criterio de cálculo de semejanzas, que veremos más adelante, se realiza la misma operación para otro nuevo elemento que se adhiere al cluster o forma uno nuevo. Los cluster creados mediante este proceso actúan ahora de forma atómica como si fuesen un único elemento. Esta operación se repite iterativamente hasta que todos los elementos son categorizados en función de sus semejanzas.

3. Finalmente el algoritmo termina cuando todos los elementos están categorizados en alguno de los n clusters que queremos.

El cálculo de las distancias entre dos elementos cuando al menos uno de ellos es un cluster formado por dos o más elementos atómicos viene definido por el criterio que usemos para ubicar en el espacio el cluster.
Dichos criterios pueden ser simple linkage (distancia entre los dos elementos más próximos de ambos clusters), complete linkage (distancia entre los dos elementos mas alejados de ambos clusters), arithmetic average (distancia media de las distancias entre todos los elementos de

ambos clusters) y centroide (distancia entre los dos centros de masas de ambos clusters). En el ejercicio hemos usado tres de ellos con resultados diferentes.

Obsérvese que k-means ubica n centroides que cambian de posición a medida que se agregan elementos al cluster. Esto implica que a lo largo de la ejecución del algoritmo un elemento puede ser categorizado en múltiples clusters y que a medida que los centro de masa se desplazan este elemento cambia de categoría.

Por otro lado en el método aglomerativo esto no sucede, ya que el cluster “crece” ya con las distancias mínimas posibles entre sus elementos y el crecimiento o aglomeración de nuevos elementos se detiene cuando no existen elementos para categorizar. Dicho de otro modo k-means crea n categorías y busca las semejanzas al centro de esas categorías y el método aglomerativo agrupa en función de las semejanzas primero y crea las categorías después en función del número de categorías necesarias.

K-Means nítido sobre un conjunto de dimensionalidad reducida por PCA

En el punto 3 del exercici 1 hemos reducido la dimensionalidad del conjunto conservando el 96% de la información. Dicho de otra forma se han proyectado 6 dimensiones en 3 y se ha aplicado el algoritmo k-means sobre el resultado con una precisión igual a la ejecución sobre el cojunto de información completo.

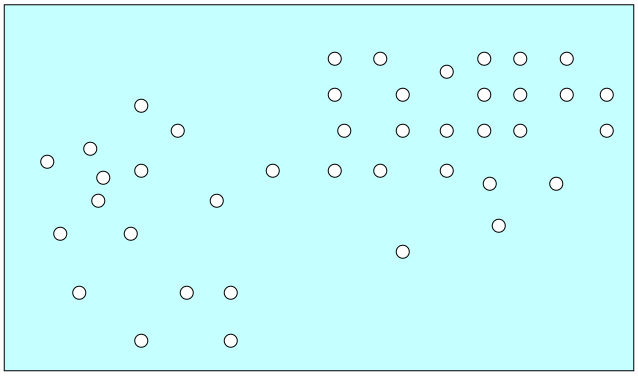
La reducción de dimensionalidad es una cuestión de gran importancia computacional, ya que el crecimiento en complejidad es exponencial y puede hacer problemas intratables computacionalmente.

El hecho de que se pueda reducir la dimensionalidad sin variar el resultado final es de gran valor.

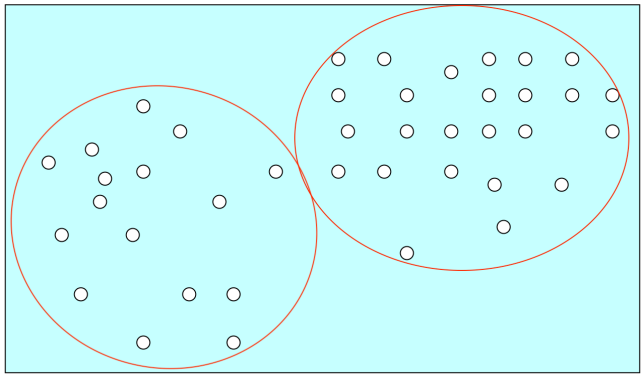
Elección mejorada de centroides iniciales en k-means con k-means_sklearny

Para concluir estudiemos el algoritmo k-means utilizado en el exercici 3 y la importancia de la selección de los n centroides iniciales.

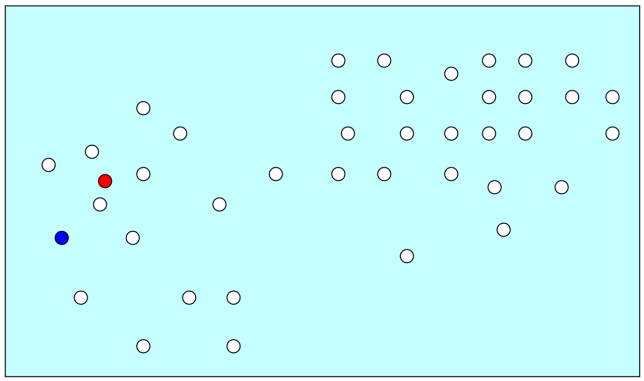
Sea un conjunto de elementos que deseamos agrupar en 2 clusters



Intuitivamente podemos suponer que el clustering terminará con una categorización mas o menos así

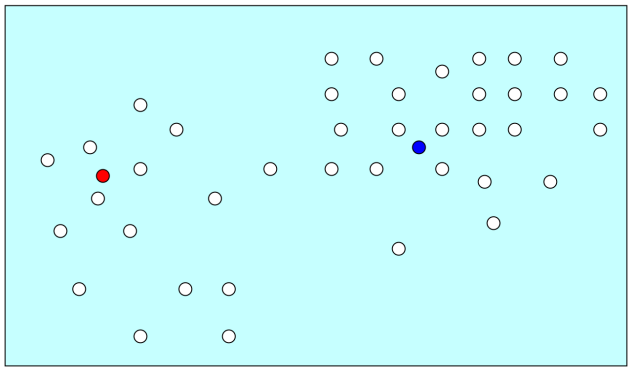


¿Pero, que sucedería si la elección de centroides iniciales fuese esta?



Probablemente el algoritmo no pueda categorizar correctamente los elementos del conjunto, máxime cuando por cuestión relativa al coste computacional hemos limitado el número de iteraciones del algoritmo.

Por tanto sería muy interesante de cara a mejorar la precisión del algoritmo obtener centroides iniciales del tipo



Enel ejercicio 3 la selección de centroides iniciales se pasa comom parámetro a la función kmeans, que admite un vector,'random' o 'k-means++'. La diferencia de esto dos últimos arroja una diferencia en la precisión de más del 87% con el conjunto de datos que estamos usando.