T4 Clustering: algoritmo K-medias

Índice

1 Clustering particional

Clustering particional: dado un conjunto de N datos $\mathcal{D}$ y un número de clusters $K,$ el clustering particional consiste en optimizar alguna función criterio $J(\Pi)$ para evaluar la calidad de cualquier partición Π de los datos en K clústeres

$$\Pi^* = \underset{\Pi}{\operatorname{argopt}}\, J(\Pi)$$

Intractabilidad: el clustering particional es en general un problema intratable puesto que el número de particiones a explorar crece exponencialmente con N y K (ver números de Stirling del segundo tipo)

Aproximación usual: hacemos uso de algoritmos aproximados para optimizar un criterio particular como por ejemplo la suma de errores cuadráticos

2 Criterio suma de errores cuadráticos

Suma de errores cuadráticos (SEC): de una partición $\Pi = \{X_1, \ldots, X_K\}$

$$J(\Pi) = \sum_{k=1}^{K} J_k \quad \text{con} \quad J_k = \sum_{\boldsymbol{x} \in X_k} \|\boldsymbol{x} - \boldsymbol{m}_k\|_2^2 \quad \text{y} \quad \boldsymbol{m}_k = \frac{1}{|X_k|} \sum_{\boldsymbol{x} \in X_k} \boldsymbol{x}$$

Interpretación:

- Cada clúster k se representa por su **centroide** o **media** $\boldsymbol{m}_k$
- Si $\boldsymbol{x}$ pertenece al clúster k, $\boldsymbol{x} - \boldsymbol{m}_k$ es el **vector error** obtenido al representar $\boldsymbol{x}$ con $\boldsymbol{m}_k$
- El error asociado a $\boldsymbol{x}$ se mide con la norma Euclidiana de su vector error, $\|\boldsymbol{x} - \boldsymbol{m}_k\|_2$
- Denominamos **distorsión** del clúster k, J_k, a la suma de errores al cuadrado de sus datos
- El criterio SEC es la suma de las distorsiones de todos los clusters i, obviamente, es un criterio a minimizar
- Idealmente, esperamos clusters hiper-esféricos compactos y de tamaño parecido, sobre K medias bien separadas
- Si la partición natural de los datos es distinta a la esperada, es probable que la minimización de la SEC no la encuentre

Ejemplo: cálculo de la SEC para $\Pi = \{X_1 = \{(1,7)^t, (4,2)^t, (4,6)^t\}, X_2 = \{(8,2)^t, (8,6)^t\}\}$

$$\boldsymbol{m}_1 = (3,5)^t \qquad J_1 = 8 + 10 + 2 = 20$$
$$\boldsymbol{m}_2 = (8,4)^t \qquad J_2 = 4 + 4 = 8$$

$$J = J_1 + J_2 = 28$$

In [1]:
```
import numpy as np; np.set_printoptions(precision=2)
def SEQ(X, y): # labels from 0 to K-1 for simplicity
    N, D = X.shape; K = np.max(y)+1; J = 0.0; m = np.zeros((K, D)); S = np.zeros(K).astype(int)
    for k in range(K):
        Xk = np.squeeze(X[np.where(y==k),:]); S[k] = Xk.shape[0];
        m[k] = Xk.mean(axis=0); J += np.square(Xk - m[k]).sum()
    return J, m, S
X = np.array([[1, 7], [4, 2], [4, 6], [8, 2], [8, 6]]); y = np.array([0, 0, 0, 1, 1])
J, m, S = SEQ(X, y); print(f'J = {J:.2f} m = {m.reshape(1, -1)} S = {S}')
```

```
J = 28.00 m = [[3. 5. 8. 4.]] S = [3 2]
```

3 Algoritmo K-medias de Duda y Hart

Incremento de SEC al transferir un dato de clúster: si se transfiere un dato $\boldsymbol{x}$ del clúster i al j, el incremento de SEC es

$$\Delta J = \frac{|X_j|}{|X_j|+1}\|\boldsymbol{x}-\boldsymbol{m}_j\|_2^2 - \frac{|X_i|}{|X_i|-1}\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2$$

Condición DH: conviene transferir si $\Delta J < 0$, es decir, si se incrementa menos J en X_j del que se decrementa en X_i

$$\frac{|X_j|}{|X_j|+1}\|\boldsymbol{x}-\boldsymbol{m}_j\|_2^2 < \frac{|X_i|}{|X_i|-1}\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2$$

Algoritmo K-medias de Duda y Hart: para cada dato, busca la transferencia de menor ΔJ y la aplica si cumple DH

> **Entrada:** una partición inicial, $\Pi = \{X_1, \ldots, X_K\}$
> **Salida:** una partición optimizada, $\Pi^* = \{X_1, \ldots, X_K\}$
> Calcular medias y J
> `repetir`
>
> > `para todo` dato $\boldsymbol{x}$
> >
> > > Sea i el clúster en el que se encuentra $\boldsymbol{x}$
> > > Encontrar un $j \neq i$ que minimice $\triangle J$ al transferir $\boldsymbol{x}$ de i a j
> > > `si` $\triangle J < 0$: transferir $\boldsymbol{x}$ de i a j y actualizar medias y J
>
> `hasta que` no se encuentre ninguna transferencia provechosa

Implementación: función para problemas sencillos

In [2]:
```python
import numpy as np; np.set_printoptions(precision=2, linewidth=np.inf)
def kmeansDH(X, y, max_iter=10, verbose=0):
    N = X.shape[0]; J, m, S = SEQ(X, y); z = y.copy(); notransfer = 0
    for iter in range(max_iter):
        for n in range(N):
            x = X[n, :]; i = z[n]
            if S[i] == 1: continue
            D = np.square(x - m).sum(axis=1); Di = S[i] / (S[i] - 1.0) * D[i]
            D = S / (S + 1.0) * D; D[i] = np.inf; j = np.argmin(D); Dj = D[j]; DJ = Dj - Di
            if verbose > 0: print(f'{iter} {x} {Di:.2f} {Dj:.2f} {DJ:.2f}', end=" ")
            if DJ < 0.0:
                z[n] = j; S[i] -= 1; S[j] += 1; J += DJ; notransfer = 1
                m[i] = m[i] - (x - m[i]) / S[i]; m[j] = m[j] + (x - m[j]) / S[j]
                if verbose > 0: print(f'=> z ={z} m = {m.reshape(1, -1)} J = {J:.2f}')
            else: print("=> no transfer"); notransfer += 1
            if notransfer == N: break
        if notransfer == N: break
    return J, m, z
```

Ejemplo (cont.): $X_1 = \{\boldsymbol{x}_1 = (1,7)^t, \boldsymbol{x}_2 = (4,2)^t, \boldsymbol{x}_3 = (4,6)^t\}$ $X_2 = \{\boldsymbol{x}_4 = (8,2)^t, \boldsymbol{x}_5 = (8,6)^t\}$

$\boldsymbol{x}$	i	j	$\frac{\lvert X_i\rvert}{\lvert X_i\rvert-1}\lVert\boldsymbol{x}-\boldsymbol{m}_i\rVert_2^2$	$\frac{\lvert X_j\rvert}{\lvert X_j\rvert+1}\lVert\boldsymbol{x}-\boldsymbol{m}_j\rVert_2^2$	$\triangle J$	X_1	X_2	$\boldsymbol{m}_1$	$\boldsymbol{m}_2$	J
						$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$	$\{\boldsymbol{x}_4, \boldsymbol{x}_5\}$	$(3,5)^t$	$(8,4)^t$	28
$\boldsymbol{x}_1$	1	2	$\frac{3}{2}\cdot 8 = 12$	$\frac{2}{3}\cdot 58 = 38.67$	$\frac{80}{3} = 26.67$					
$\boldsymbol{x}_2$	1	2	$\frac{3}{2}\cdot 10 = 15$	$\frac{2}{3}\cdot 20 = 13.33$	$-\frac{5}{3} = -1.67$	$\{\boldsymbol{x}_1, \boldsymbol{x}_3\}$	$\{\boldsymbol{x}_2, \boldsymbol{x}_4, \boldsymbol{x}_5\}$	$\left(\frac{5}{2}, \frac{13}{2}\right)^t$	$\left(\frac{2}{3}, \frac{10}{3}\right)^t$	26.33
$\boldsymbol{x}_3$	1	2	$\frac{2}{1}\cdot\frac{10}{4} = 5$	$\frac{3}{4}\cdot\frac{128}{9} = 10.67$	$\frac{17}{3} = 5.67$					
$\boldsymbol{x}_4$	2	1	$\frac{3}{2}\cdot\frac{32}{9} = 5.33$	$\frac{2}{3}\cdot\frac{101}{2} = 33.67$	$\frac{85}{3} = 28.33$					
$\boldsymbol{x}_5$	2	1	$\frac{3}{2}\cdot\frac{80}{9} = 13.33$	$\frac{2}{3}\cdot\frac{61}{2} = 20.33$	7					
$\boldsymbol{x}_1$	1	2	$\frac{2}{1}\cdot\frac{5}{2} = 5$	$\frac{3}{4}\cdot\frac{401}{9} = 34.17$	$\frac{175}{6} = 29.17$					

In [3]:
```
X = np.array([[1, 7], [4, 2], [4, 6], [8, 2], [8, 6]]); y = np.array([0, 0, 0, 1, 1])
J, m, z = kmeansDH(X, y, max_iter=3, verbose=1)
print(f'{z} {m.reshape(1, -1)} {J:.2f}')
```

```
0 [1 7] 12.00 38.67 26.67 => no transfer
0 [4 2] 15.00 13.33 -1.67 => z =[0 1 0 1 1] m = [[2.5  6.5  6.67 3.33]] J = 26.33
0 [4 6] 5.00 10.67 5.67 => no transfer
0 [8 2] 5.33 33.67 28.33 => no transfer
0 [8 6] 13.33 20.33 7.00 => no transfer
1 [1 7] 5.00 34.17 29.17 => no transfer
[0 1 0 1 1] [[2.5  6.5  6.67 3.33]] 26.33
```

4 Algoritmo K-medias convencional

Condición convencional: conviene transferir $\boldsymbol{x}$ del clúster i al j si

$$\|\boldsymbol{x} - \boldsymbol{m}_j\|_2^2 < \|\boldsymbol{x} - \boldsymbol{m}_i\|_2^2$$

Relación con la condición DH: la convencional es suficiente (pero no necesaria; ver ejemplo)

$$\frac{|X_j|}{|X_j|+1}\|\boldsymbol{x} - \boldsymbol{m}_j\|_2^2 < \|\boldsymbol{x} - \boldsymbol{m}_j\|_2^2 \overset{?}{<} \|\boldsymbol{x} - \boldsymbol{m}_i\|_2^2 < \frac{|X_i|}{|X_i|-1}\|\boldsymbol{x} - \boldsymbol{m}_i\|_2^2$$

Algoritmo K-medias convencional:

> **Entrada:** una partición inicial, $\Pi = \{X_1, \ldots, X_K\}$
> **Salida:** una partición optimizada, $\Pi^* = \{X_1, \ldots, X_K\}$
> `repetir`
>
> > Calcular las medias de los clusters
> > Reclasificar los datos según las medias más próximas
>
> `hasta que` no se reclasifique ningún dato

Implementación: función para problemas sencillos

In [4]:
```python
import numpy as np; np.set_printoptions(precision=2, linewidth=np.inf)
def kmeans(X, y, max_iter=10, verbose=0):
    N = X.shape[0]; z = y.copy()
    for iter in range(max_iter):
        J, m, _ = SEQ(X, z); transfers = 0
        for n in range(N):
            x = X[n, :]; i = z[n]; D = np.square(x - m).sum(axis=1)
            Di = D[i]; D[i] = np.inf; j = np.argmin(D); Dj = D[j]
            if verbose > 0: print(f'{iter} {x} {Di:.2f} {Dj:.2f}', end=" ")
            if Dj < Di:
                z[n] = j; transfers += 1
                if verbose > 0: print(f'=> z ={z}')
            else: print("=> no transfer");
        if transfers == 0: break
    return J, m, z
```

Ejemplo (cont.): $X_1 = \{\boldsymbol{x}_1 = (1,7)^t, \boldsymbol{x}_2 = (4,2)^t, \boldsymbol{x}_3 = (4,6)^t\} \quad X_2 = \{\boldsymbol{x}_4 = (8,2)^t, \boldsymbol{x}_5 = (8,6)^t\}$

$\boldsymbol{x}$	i	j	$\|\boldsymbol{x} - \boldsymbol{m}_i\|_2^2$	$\|\boldsymbol{x} - \boldsymbol{m}_j\|_2^2$	X_1	X_2	$\boldsymbol{m}_1$	$\boldsymbol{m}_2$	J
					$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$	$\{\boldsymbol{x}_4, \boldsymbol{x}_5\}$	$(3,5)^t$	$(8,4)^t$	28
$\boldsymbol{x}_1$	1	2	8	58					
$\boldsymbol{x}_2$	1	2	10	20					
$\boldsymbol{x}_3$	1	2	2	20					
$\boldsymbol{x}_4$	2	1	34	4					
$\boldsymbol{x}_5$	2	1	26	4					

In [5]:
```
X = np.array([[1, 7], [4, 2], [4, 6], [8, 2], [8, 6]]); y = np.array([0, 0, 0, 1, 1])
J, m, z = kmeans(X, y, max_iter=1, verbose=1)
print(f'{z} {m.reshape(1, -1)} {J:.2f}')
```

```
0 [1 7] 8.00 58.00 => no transfer
0 [4 2] 10.00 20.00 => no transfer
0 [4 6] 2.00 20.00 => no transfer
0 [8 2] 4.00 34.00 => no transfer
0 [8 6] 4.00 26.00 => no transfer
[0 0 0 1 1] [[3. 5. 8. 4.]] 28.00
```

Ejercicios T4 Clustering: algoritmo K-medias

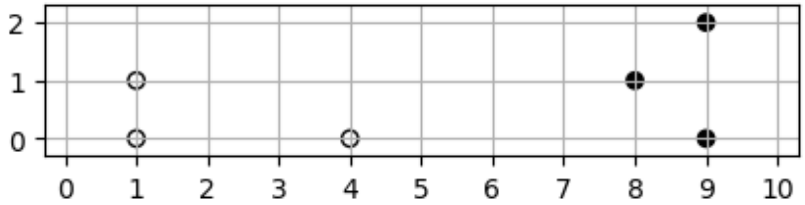
2023_01_26_Cuestión_4: Tenemos una partición de un conjunto de datos 3-dimensionales en un número de clústeres dado, $C \geq 2$. Considerad la transferencia del dato $\boldsymbol{x} = (3, 6, 4)^t$ de un clúster j a otro y, $j \neq y$. Se sabe que el clúster j contiene 3 datos (contando $\boldsymbol{x}$) e y 3. Así mismo, se sabe que la media del clúster j es $\boldsymbol{m}_j = (3, 3, 2)^t$ y la del y, $\boldsymbol{m}_y = (7, 6, 9)^t$. Si se realiza dicha transferencia, se producirá un incremento de la suma de errores cuadráticos, ΔJ, tal que:

1. $\Delta J < -70$
2. $-70 \leq \Delta J < -30$
3. $-30 \leq \Delta J < 0$
4. $\Delta J \geq 0$

Solución: la 4 ya que $\Delta J = 11.2$

2023_01_17_Cuestión 7: La figura siguiente muestra una partición de 6 puntos bidimensionales en dos clústeres, ○ y ●:

In [1]:
```
import numpy as np; import matplotlib.pyplot as plt;
fig = plt.figure(figsize=(5, 1)); plt.xlim([-.3, 10.3]); plt.ylim([-.3, 2.3])
plt.xticks(np.arange(0, 11)); plt.yticks(np.arange(0, 3)); plt.grid()
X = np.array([[1,0], [1,1], [4,0], [8,1], [9,0], [9,2]])
y = np.array([1, 1, 1, 2, 2, 2])
plt.scatter(*X.T, c=y, cmap=plt.cm.binary, edgecolors='black');
```



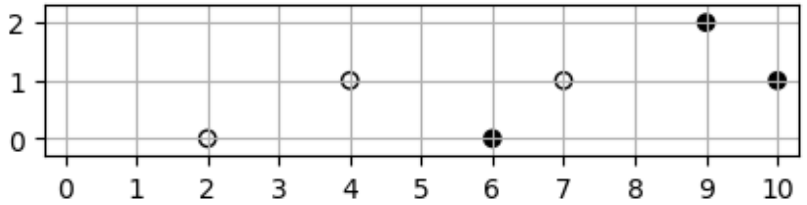
Si transferimos de clúster el punto $(1, 0)^t$, se produce una variación de la suma de errores cuadráticos (SEQ), $\Delta J = J - J'$ (SEQ después del intercambio menos SEQ antes del intercambio), tal que:

1. $\Delta J < -7$
2. $-7 \leq \Delta J < 0$
3. $0 \leq \Delta J < 7$
4. $\Delta J \geq 7$

Solución: la 4 ya que $\Delta J = 52.5 - 9.3 = 43.2$

2022_01_27_Cuestión 6: La figura siguiente muestra una partición de 6 puntos bidimensionales en dos clusters, ○ y ●:

In [2]:
```
import numpy as np; import matplotlib.pyplot as plt;
fig = plt.figure(figsize=(5, 1)); plt.xlim([-.3, 10.3]); plt.ylim([-.3, 2.3])
plt.xticks(np.arange(0, 11)); plt.yticks(np.arange(0, 3)); plt.grid()
X = np.array([[2,0], [4,1], [6,0], [7,1], [9,2], [10,1]])
y = np.array([1, 1, 2, 1, 2, 2])
plt.scatter(*X.T, c=y, cmap=plt.cm.binary, edgecolors='black');
```



Si intercambiamos de clúster los puntos $(10, 1)^t$ y $(7, 1)^t$, se produce una variación de la suma de errores cuadráticos (SEQ), $\Delta J = J - J'$ (SEQ después del intercambio menos SEQ antes del intercambio), tal que:

1. $\Delta J < -7$
2. $-7 \leq \Delta J < 0$
3. $0 \leq \Delta J < 7$
4. $\Delta J \geq 7$

Solución: la 4 ya que $\Delta J = 42.0 - 24.0 = 18.0$