U3 Clustering: K-means algorithm

Index

1 Partitional clustering

Partitional clustering: given a dataset $\mathcal{D}$ of N samples and a number of clusters $K,$ the partitional clustering consists of optimizing some criterion function $J(\Pi)$ to evaluate the quality of any partition Π of the dataset into K clusters

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

Intractability: partitional clustering is in general an intractable problem since the number of partitions to explore grows exponentially with N and K (see Stirling numbers of the second kind)

Usual approximation: we use approximate algorithms to optimize a particular criterion such as the sum of squared errors

2 Criterion: sum of squared errors

Sum of squared errors (SSE): of a partition $\Pi = \{X_1, \dots, X_K\}$

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

Interpretation:

- Each cluster k is represented by its **centroid** or **mean** $\boldsymbol{m}_k$
- If $\boldsymbol{x}$ belongs to cluster k, $\boldsymbol{x} - \boldsymbol{m}_k$ is the **error vector** obtained by representing $\boldsymbol{x}$ with $\boldsymbol{m}_k$
- The error associated with $\boldsymbol{x}$ is measured by the Euclidean norm of its error vector, $\|\boldsymbol{x} - \boldsymbol{m}_k\|_2$
- We call the sum of the squared errors of the cluster k, J_k, the **distortion** of cluster k
- The SSE criterion is the sum of the distortions of all clusters and is obviously a criterion to be minimized
- Ideally, we expect compact hyper-spherical clusters of similar size, around K well-separated averages
- If the natural partition of the data is different from that expected, it is likely that SSE minimization will not find it

Example: calculation of the SSE for $\Pi = \{X_1 = \{(1,7)^t, (4,2)^t, (4,6)^t\}, X_2 = \{(8,2)^t, (8,6)^t\}\}$

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

$$J = J_1 + J_2 = 28$$

In [1]:
```python
import numpy as np; np.set_printoptions(precision=2)
def SSE(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 = SSE(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 K-means algorithm by Duda and Hart

SSE increment when transferring a data sample: if a data $\boldsymbol{x}$ is transferred from cluster i to j, the SSE increment is

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

DH condition: should be transferred if $\Delta J < 0$, that is, if J is increased less if $\boldsymbol{x}$ is moved into X_j than J is decreased by leaving $\boldsymbol{x}$ in X_i

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

K-means algorithm by Duda and Hart: for each data sample, find the transfer of least ΔJ and apply it if it satisfies DH

> **Input:** an initial partition, $\Pi = \{X_1, \dots, X_K\}$
> **Output:** an optimized partition, $\Pi^* = \{X_1, \dots, X_K\}$
> Calculate means and J
> `repeat`
>
> > `for all` $\boldsymbol{x}$
> >
> > > Let i be the cluster in which $\boldsymbol{x}$ is located
> > > Find a $j \neq i$ that minimizes $\triangle J$ by transferring $\boldsymbol{x}$ from i to j
> > > `if` $\triangle J < 0:$ transfer $\boldsymbol{x}$ from i to j and update means and J
>
> `until` no profitable transfer is found

Implementation: function for simple problems

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 = SSE(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
```

Example (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	$\frac{\lvert X_i\rvert}{\lvert X_i\rvert-1}\lVert\boldsymbol{x}-\boldsymbol{m}_i\rVert_2^2$	$\frac{\lvert X_j\rvert}{\lvert X_j\rvert+1}\lVert\boldsymbol{x}-\boldsymbol{m}_j\rVert_2^2$	$\triangle J$	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 Conventional K-means algorithm

Condition for conventional algorithm: $\boldsymbol{x}$ should be transfered from cluster i to j if

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

Relation to the DH condition: the conventional one is sufficient (but not necessary; see example)

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

Conventional K-means algorithm:

> **Input:** an initial partition, $\Pi = \{X_1, \ldots, X_K\}$
> **Output:** an optimized partition, $\Pi^* = \{X_1, \ldots, X_K\}$
> `repeat`
>
> > Calculate the mean of each clusters
> > Reclassify the data according to the nearest mean
>
> `until` no data is reclassified

Implementation: function for simple problems

In [4]:
```
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, _ = SSE(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
```

Example (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	$\lVert\boldsymbol{x}-\boldsymbol{m}_i\rVert_2^2$	$\lVert\boldsymbol{x}-\boldsymbol{m}_j\rVert_2^2$	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
```

U3 Clustering: K-means algorithm

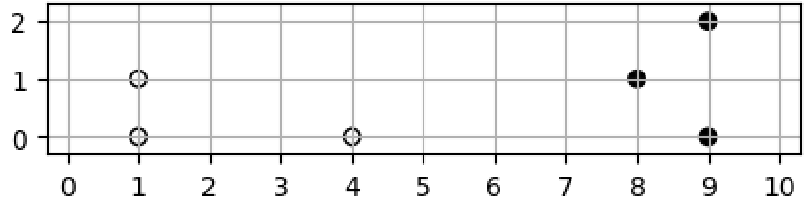
2023_01_26_Question_4: We have a partition of a 3-dimensional dataset into a given number of clusters, $C \geq 2$. Consider the transfer of the data $\boldsymbol{x} = (3, 6, 4)^t$ from one cluster j to another i, $j \neq i$. Cluster j is known to contain 3 data samples (including $\boldsymbol{x}$) and the cluster i contains 3. Likewise, it is known that the mean of cluster j is $\boldsymbol{m}_j = (3, 3, 2)^t$ and that of i, $\boldsymbol{m}_i = (7, 6, 9)^t$. If said transfer is carried out, there will be an increase in the sum of quadratic errors, ΔJ, such that:

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

Solution: option 4 since $\Delta J = 11.2$

2023_01_17_Question 7: The figure below shows a partition of 6 two-dimensional points into two clusters, ○ and ●:

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');
```



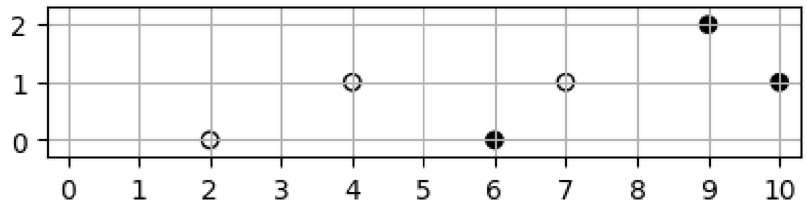
If we transfer the data sample $(1,0)^t$ to the other cluster, there is a variation of the sum of squared errors (SSE), $\Delta J = J - J'$ (SSE after exchange minus SSE before the exchange), such that:

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

Solution: option 4 since $\Delta J = 52.5 - 9.3 = 43.2$

2022_01_27_Question 6: The figure below shows a partition of 6 two-dimensional points into two clusters, ○ and ●:

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([[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');
```



If we exchange clusters the data samples $(10,1)^t$ and $(7,1)^t$, a variation of the sum of squared errors (SSE) occurs, $\Delta J = J - J'$ SSE after exchange minus SSE before exchange), such that:

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

Solution: option 4 since $\Delta J = 42.0 - 24.0 = 18.0$