CHƯƠNG 11: PHÂN CỤM (CLUSTERING)

Khoa Khoa học và Kỹ thuật thông tin
Bộ môn Khoa học dữ liệu

NỘI DUNG

1. Bài toán phân cụm
2. Các dạng giải thuật phân cụm
3. Giải thuật K-Means.
4. Giải thuật DBScan.
5. Mô hình hỗn hợp Gauss.

Phân cụm là gì ?

- Phân cụm (clustering) là phương pháp phân chia dữ liệu X ban đầu thành các cụm (cluster) có ý nghĩa dựa trên mức độ liên quan của các dữ liệu.

VD: Phân nhóm khách hàng dựa trên hành vi mua hàng.

- Để xác định mức độ liên quan của dữ liệu cần dựa vào đặc điểm giá trị và cấu trúc của dữ liệu.
- Phân cụm thuộc dạng **học máy không giám sát** (Unsupervised learning)
  + Dữ liệu đầu vào **không có nhãn (label).**

Một số ứng dụng của phân cụm

- Phát hiện bất thường.
- Hệ khuyến nghị.
- Giảm chiều dữ liệu.
- Tích hợp dữ liệu.
- ...

Các dạng của cụm

- Không có định nghĩa thống nhất cho thế nào là một cụm.
- Các thuật toán gom cụm có cách tiếp cận khác nhau về cụm:
  + Cụm có tâm cụm (centroid), các mẫu khác xếp quanh tâm cụm. Phương pháp này đôi khi còn được gọi là phân hoạch.
  + Cụm là một vùng có nhiều mẫu xếp dày đặc, gần nhau, có hình dạng bất kỳ (mật độ).
  + Cụm phân cấp: thuật toán xác định được cụm của cụm.

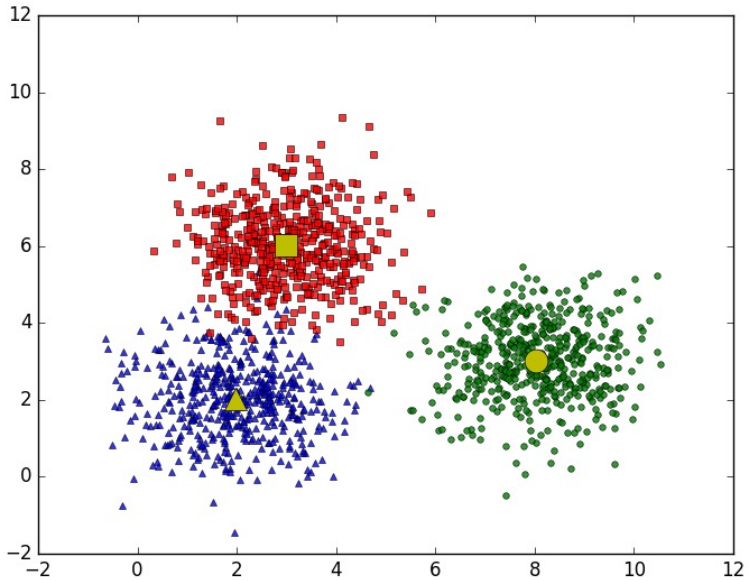
Các phương pháp phân cụm

- Phân hoạch.
  + Thuật toán minh hoạ: k-Means
- Phân cấp.
  + Thuật toán minh hoạ: Hierarchical Agglomerative Clustering (HAC)
- Dựa trên mật độ:
  + Thuật toán minh hoạ: DBSCAN

GIẢI THUẬT K-MEANS

Giới thiệu

– Là kỹ thuật gom nhóm các điểm dữ liệu có đặc tính giống nhau thành các cụm. Các điểm dữ liệu được gọi là giống nhau nếu như chúng ở gần nhau trong một không gian nhất định.



Một điểm điểm ở gần "tâm cụm" (*center*) – là các điểm đại diện cho cụm sẽ được gom về cụm đó.

➔ **Vấn đề đặt ra: xác định tâm cụm như thế nào ?**

Giải quyết vấn đề

– Mục tiêu của K-means:

+ Input: X – dữ liệu đầu vào và k – số cụm (cluster).

$X = [x_1, x_2, ..., x_m]$, $k < m$

+ Output: k center của k cụm và các điểm dữ liệu đã được phân vào các cụm dự trên các center.

- Số cụm: $m_1, m_2, ..., m_k$

- $Y = [y_{i1}, y_{i2}, ..., y_{ik}]$, i = 1...m – Nhãn của mỗi điểm dữ liệu thuộc về một cụm.

Giải thuật K-means

- Với điểm dữ liệu x_i, sai số để điểm đó thuộc cluster k (m_k là center của cluster) bất kỳ là:

$$\|x_i - m_k\|^2$$

- Với mỗi điểm dữ liệu xi sẽ được phân vào cluster k, nên $y_{ik} = 1$ (ở đây, 1 có nghĩa là thuộc cluster k). Do đó, ta nhân thêm y_{ik} vào:

$$y_{ik}\|x_i - m_k\|^2$$

- Hàm mất mát được viết lại như sau:

$$L(Y, K) = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ik} \left\| x_i - m_j \right\|_2^2$$

Giải thuật K-means

– Tối ưu hàm mất mát:

$$\text{argmin } L_{Y,K} = \text{argmin}_{Y,K} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ik} \left\| x_i - m_j \right\|_2^2$$

Ta có $\text{argmin}_j \left\| x_i - m_j \right\|_2^2$ ➔ x_i sẽ thuộc về cluster gần nó nhất (khoảng cách là nhỏ nhất).

Tuy nhiên, việc chọn cluster nào sẽ phụ thuộc vào việc *chọn điểm m làm center* ➔ chọn điểm m sao cho L đạt giá trị nhỏ nhất:

$$M = \text{argmin}_{m_j} \sum_{i=1}^{N} y_{ij} \left\| x_i - m_j \right\|_2^2$$

Giải thuật K-means

– Tính đạo hàm cho L theo m_j ➔ tìm ra center tối ưu:

$$\frac{\partial L}{\partial m_j} = 2\sum_{i=1}^{N} y_{ij}(m_j - x_i)$$

$$\frac{\partial L}{\partial m_j} = 0 \rightarrow 2\sum_{i=1}^{N} y_{ij}(m_j - x_i) = 0$$

$$\Leftrightarrow m_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij} x_i$$

$$\rightarrow m_j = \frac{\sum_{i=1}^{N} y_{ij} x_i}{\sum_{i=1}^{N} y_{ij}}$$

Tóm tắt giải thuật K-Means

- Input: Dữ liệu X + số cụm k.
- Output: k – số cụm và dữ liệu đã được gán vào từng cụm.
- Các bước thực hiện:
  + *Bước 1: Khởi tạo ngẫu nhiên các tâm cụm.*
  + *Bước 2: gán dữ liệu vào từng cụm có tâm gần nhất.*
  + *Bước 3: Cập nhật lại tâm cụm.*
  + *Bước 4: Lặp lại Bước 2 và Bước 3 cho đến khi tâm cụm không đổi.*

Ví dụ về K-Means

– Tạo dữ liệu:

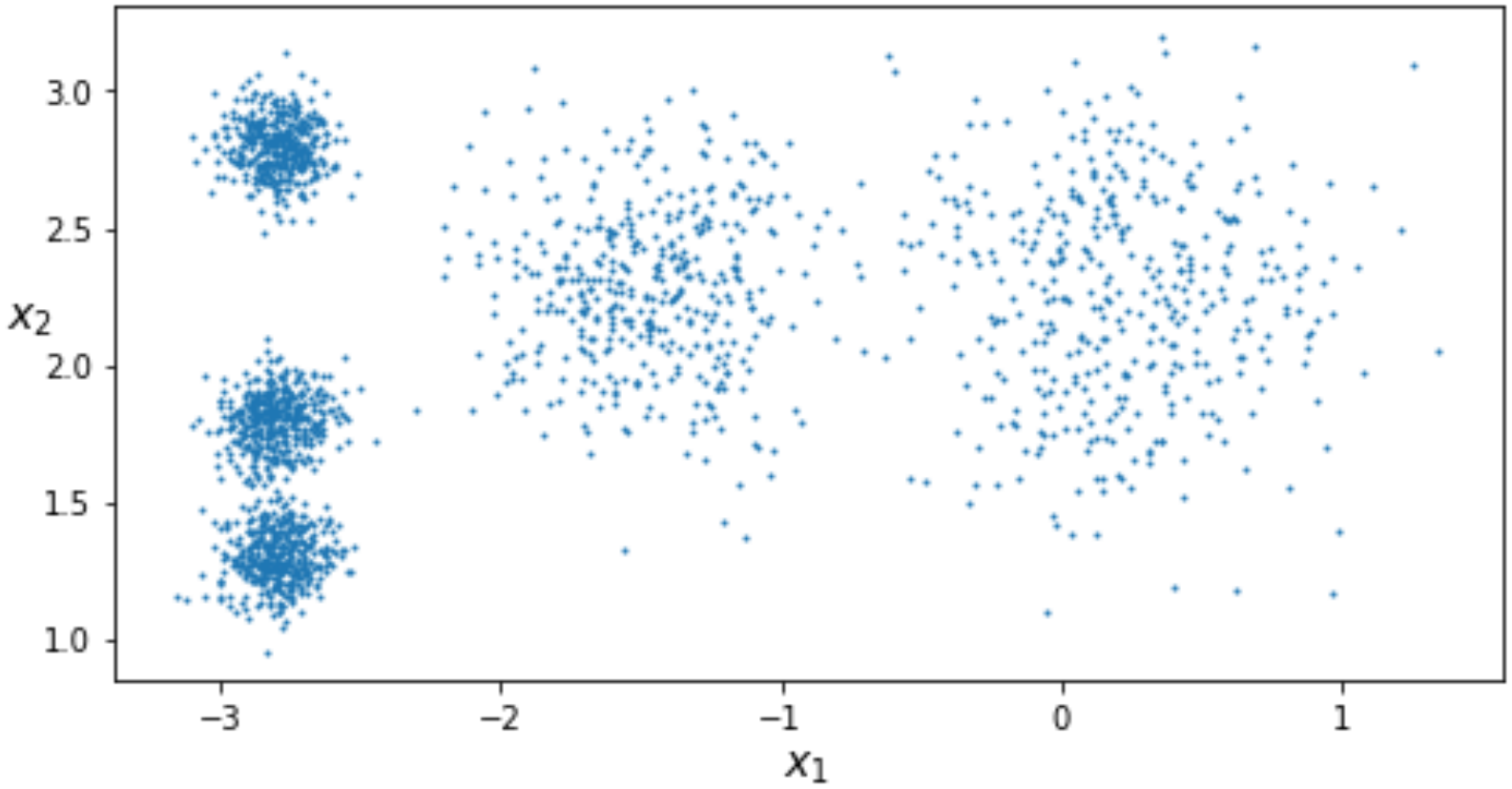
```
1. import numpy as np
2. blob_centers = np.array(
3.   [[ 0.2, 2.3],
4.   [-1.5 , 2.3],
5.   [-2.8, 1.8],
6.   [-2.8, 2.8],
7.   [-2.8, 1.3]])
8. blob_std = np.array([0.4, 0.3, 0.1, 0.1, 0.1])
```

Ví dụ về K-Means

– Phân chia dữ liệu thành X,y:

```
1. from sklearn.datasets import make_blobs


2. X, y = make_blobs(n_samples=2000,
   centers=blob_centers, cluster_std=blob_std,
   random_state=7)
```

Kết quả



Ví dụ về K-Means

– Sử dụng Kmeans từ thư viện sklearn:

```
from sklearn.cluster import KMeans
```

– Thực thi Kmeans với số cụm k = 5:

```
1. k = 5
2. kmeans = KMeans(n_clusters=k, random_state=42)
3. y_pred = kmeans.fit_predict(X)
```

Ví dụ về K-Means

– Xem số lượng centroids của từng cụm:

```
kmeans.cluster_centers_
```

– Kết quả:

```
array([
       [-2.80389616, 1.80117999], [ 0.20876306, 2.25551336],
       [-2.79290307, 2.79641063], [-1.46679593, 2.28585348],
       [-2.80037642, 1.30082566]
])
```

Ví dụ về K-Means

– Lấy kết quả phân cụm:

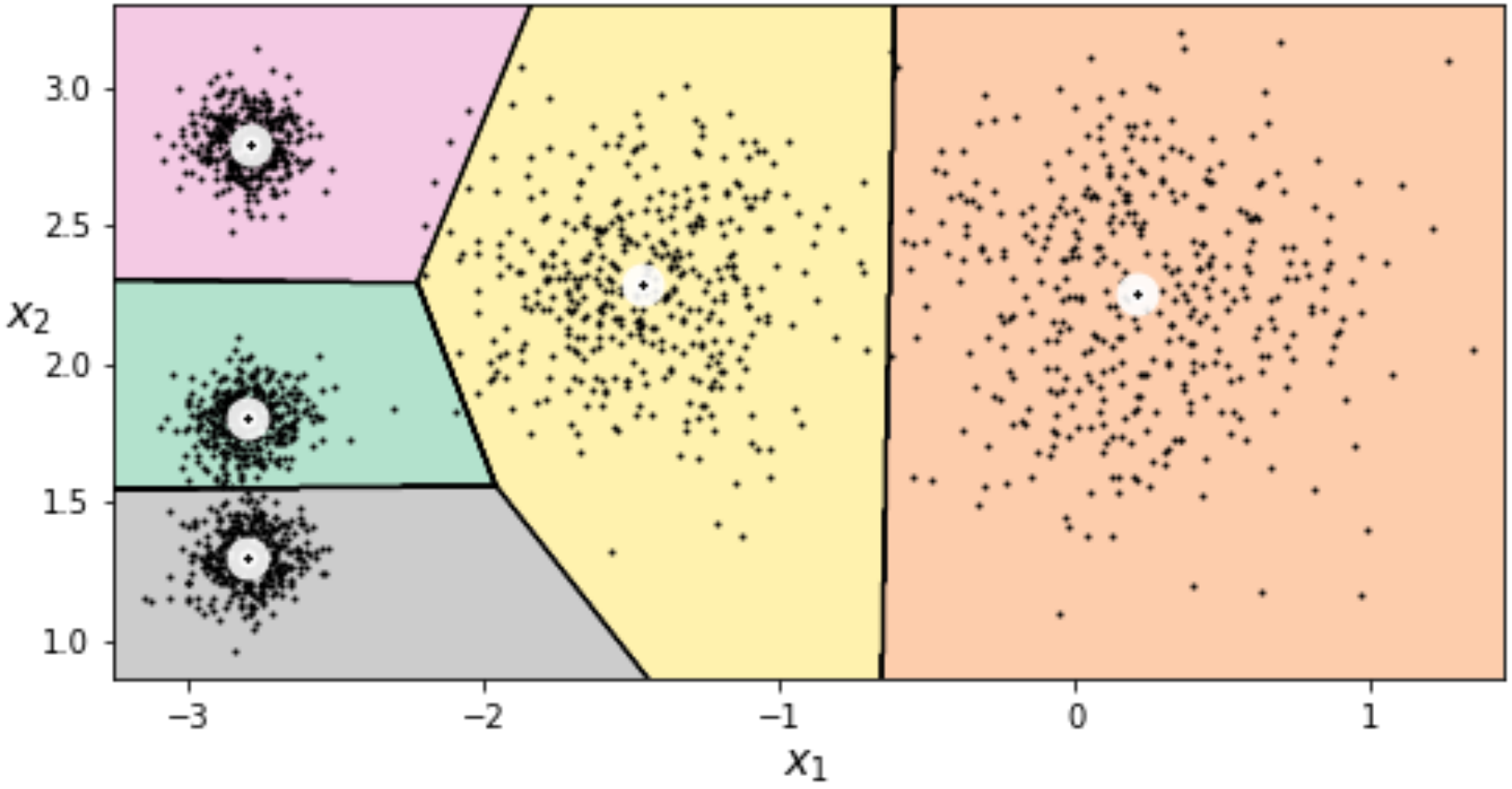
```
kmeans.labels_
```

– Kết quả:

```
array([4, 0, 1, ..., 2, 1, 0], dtype=int32)
```

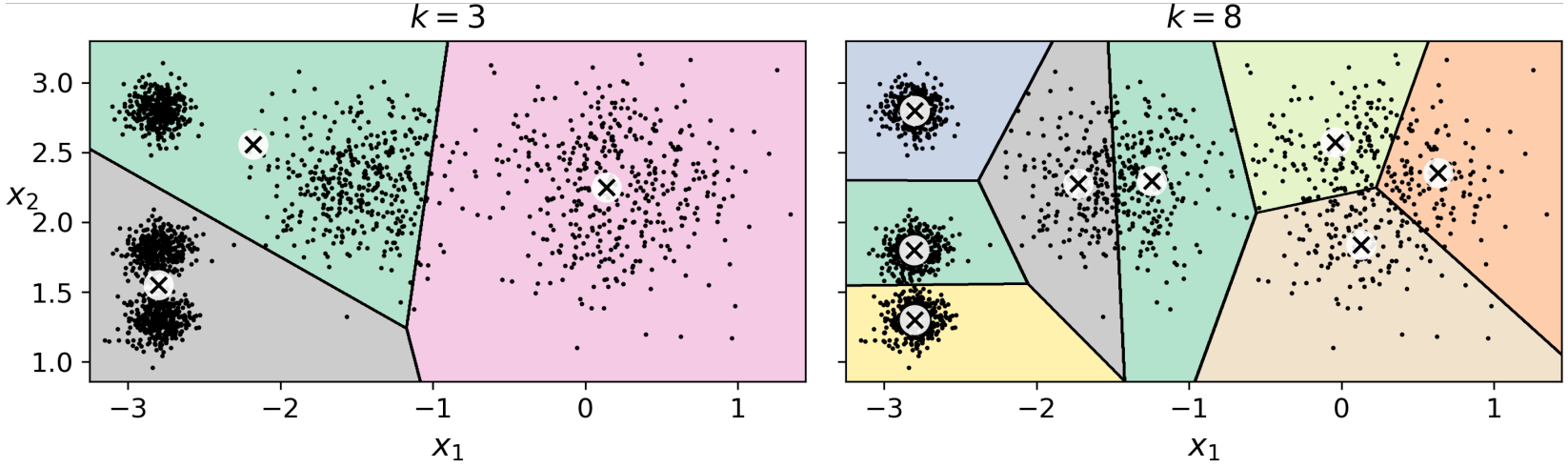
Mỗi giá trị trong kết quả tương ứng với cụm (được đánh số thứ tự từ 0 đến 4) của mỗi điểm dữ liệu.

Kết quả sau khi phân cụm



Nhận xét về K-Means

- Khó xác định số cụm k.
- Hoạt động không tốt khi các cụm có số lượng, mật độ mẫu khác nhau nhiều và không phân bố theo dạng cầu (spatial).
- Cần phải thực hiện nhiều lần để tìm được số cụm tối ưu.

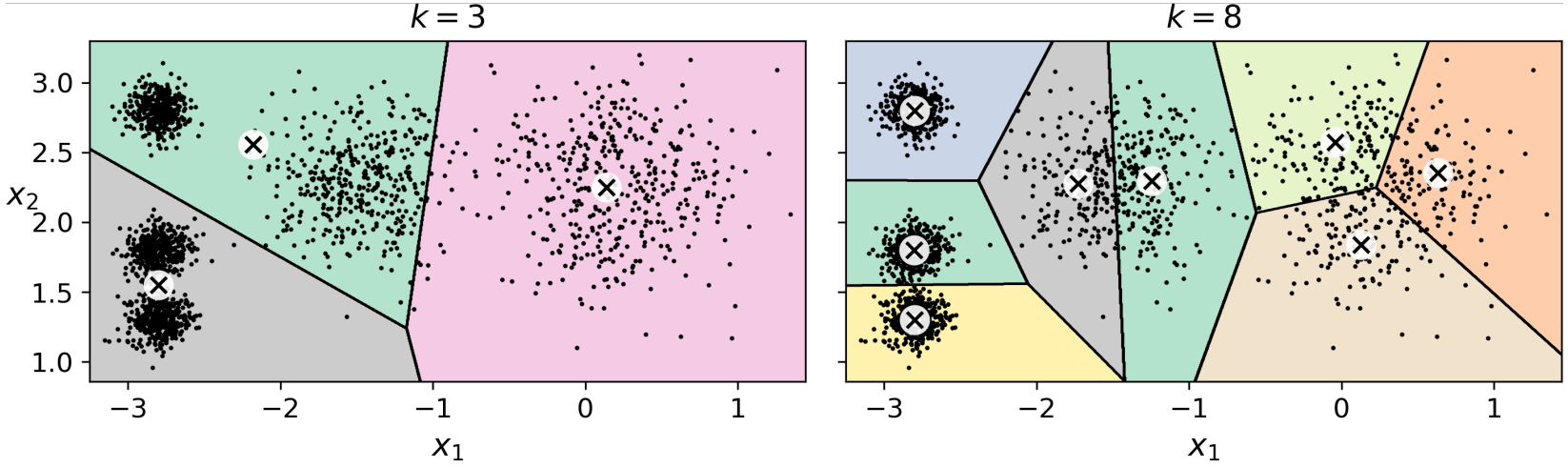


Nhận xét về K-Means

- Khó xác định số cụm k.

Hệ số Sihouette

- Hoạt động không tốt khi các cụm có số lượng, mật độ mẫu khác nhau nhiều và không phân bố theo dạng cầu (spatial).
- Cần phải thực hiện nhiều lần để tìm được số cụm tối ưu.



Mini-batch KMeans

Hệ số Silhouette

- Là một hệ số dùng để xác định tính nhất quán (consistency) trong các cụm dữ liệu.
- Hệ số silhoutette của 1 điểm dữ liệu x(i) được xác định như sau:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Trong đó:

- a(i): khoảng cách trung bình đến những mẫu khác trong 1 cụm

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

- b(i): khoảng cách trung bình đến những mẫu thuộc cụm gần cụm chứa mẫu đang xét.

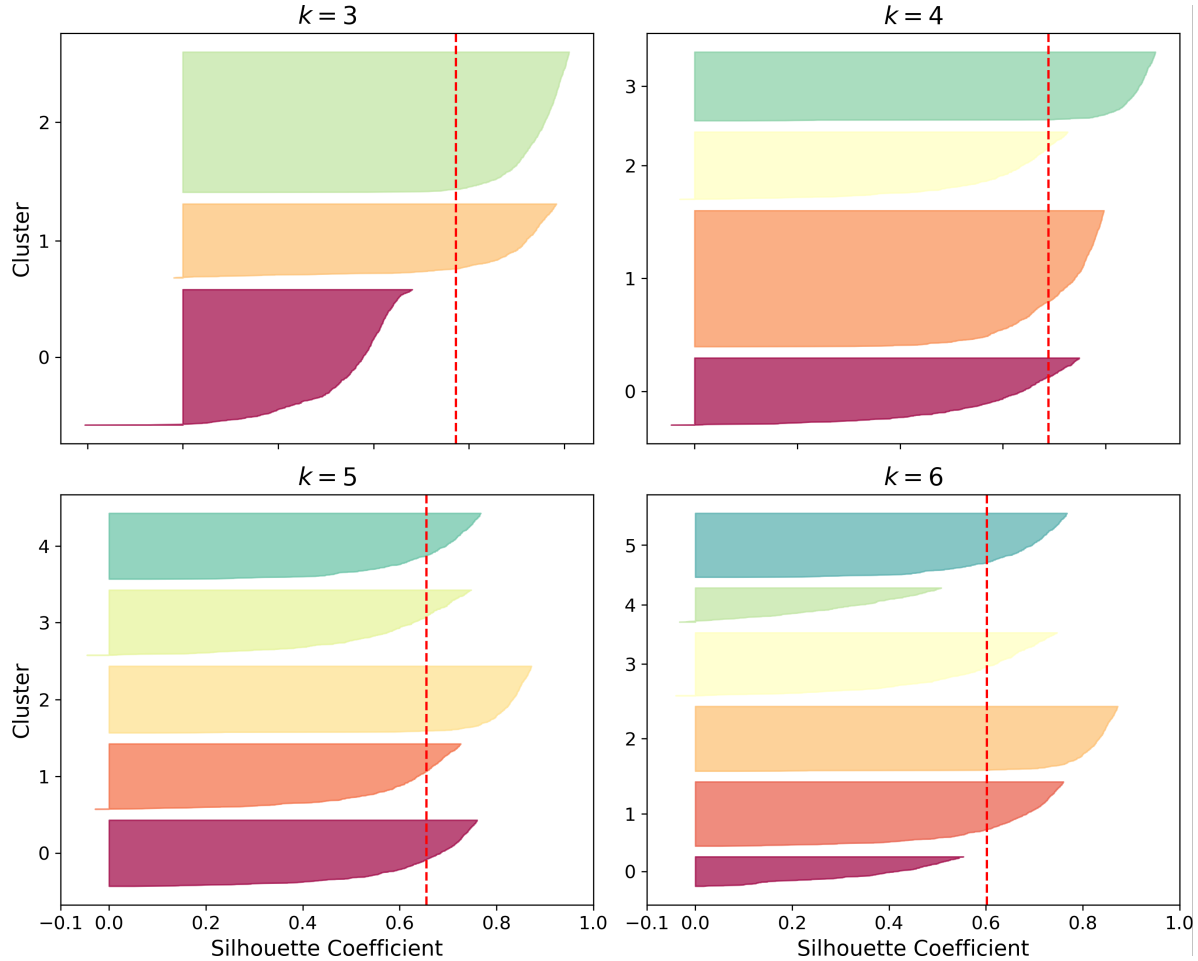
$$b(i) = \min_{J \neq i} \frac{1}{|C_I|} \sum_{j \in C_I} d(i, j)$$

Hệ số Silhouette

- Giá trị của hệ số silhouette nằm trong khoảng từ -1 đến 1:
  + Giá trị tiến đến 1: Mẫu nằm gần tâm cụm.
  + Giá trị tiến đến 0: Mẫu nằm ở biên của cụm.
  + Giá trị tiến đến -1: Mẫu nằm ngoài cụm (thuộc cụm khác).
- Sử dụng hệ số silhouttee, ta có thể xác định được số lượng cụm tối ưu (xác định được k tối ưu).
  + Cách xác định: dựa vào lược đồ sihouttee.

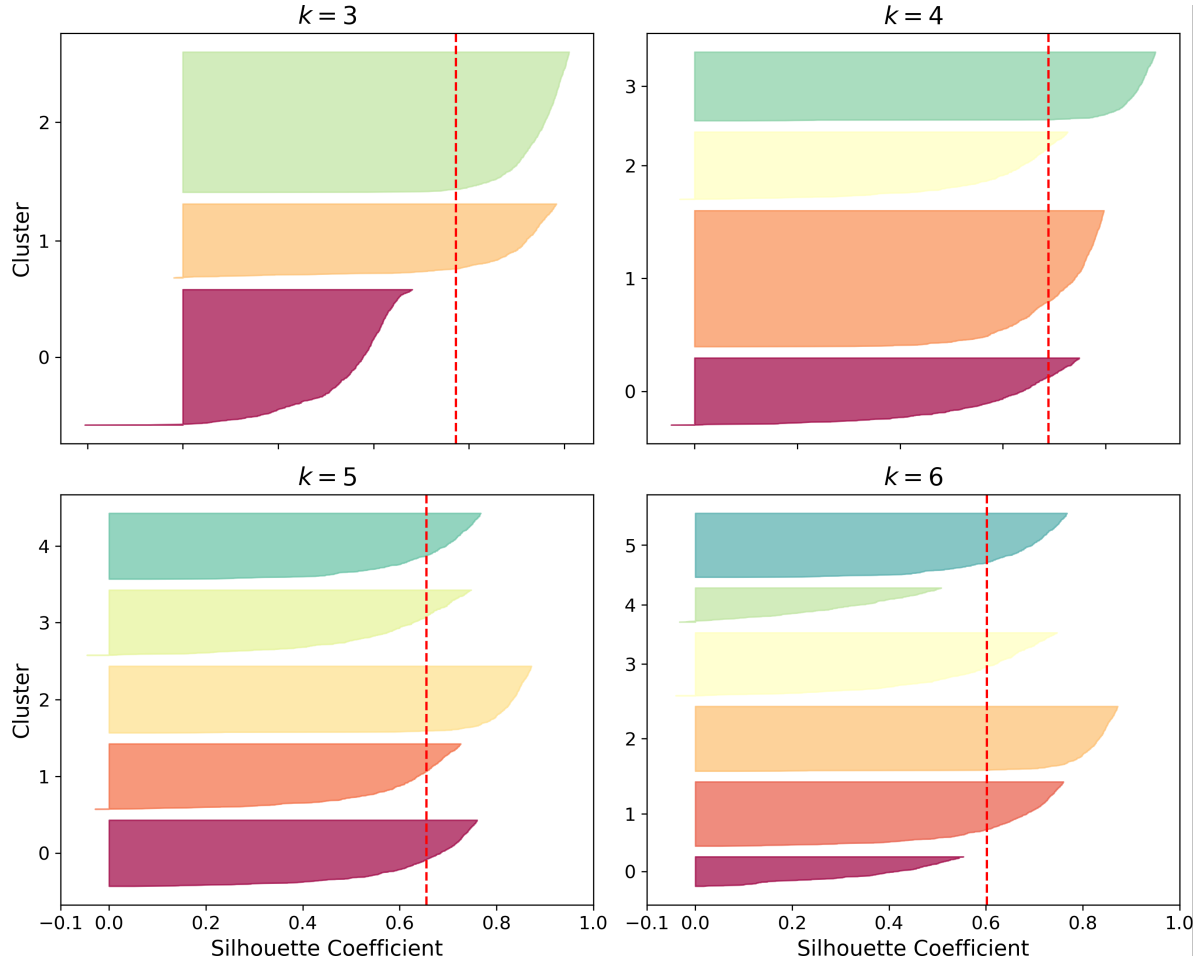
Lược đồ Silhouettee

- Đường gạch đỏ: hệ số silhouette trung bình.
- Trục ngang: hệ số silhouette cho từng cụm khác nhau.
- Trục dọc: số lượng cụm.
- Độ rộng từng mẫu thể hiện số lượng dữ liệu trong từng cụm.



Lược đồ Silhouettee

- Nếu cụm nào có các mẫu nằm dưới đường gạch đỏ thì cụm đó không tốt (phân cụm sai).
- Như hình bên phải thì:
  + Các cụm k = 3, k = 6 không tốt vì có các mẫu nằm dưới đường gạch đỏ.
  + Các cụm k = 4, k = 5 tốt hơn vì các mẫu đều nằm gần đường trung bình.



Sử dụng hệ số sihouette

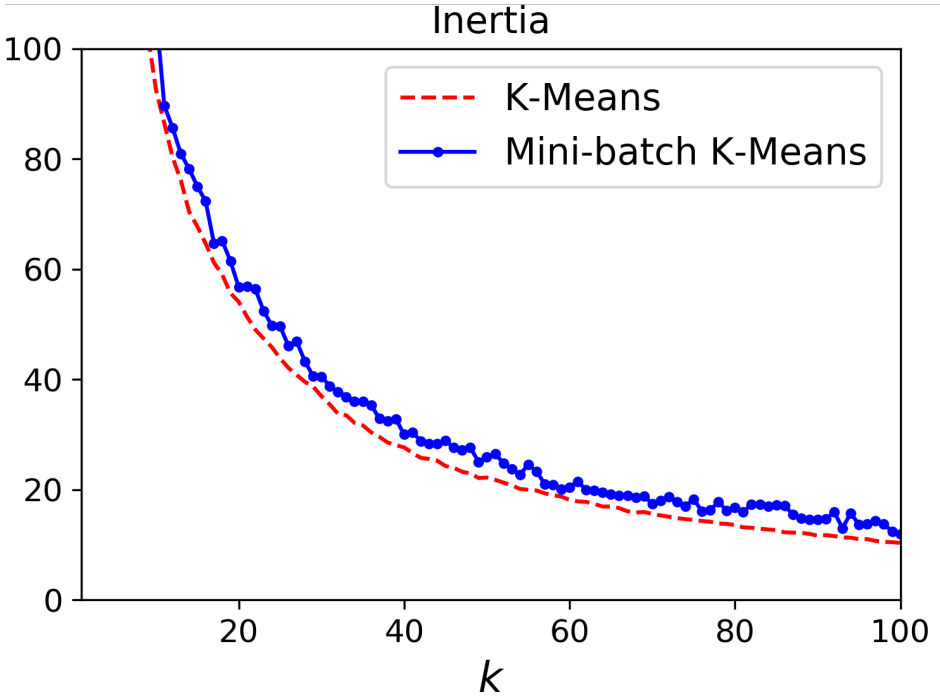
- Sử dụng hệ số sihouette từ modules metrics của thư viện sklearn.
- X là dữ liệu ban đầu, kmeans.labels là cụm ứng với từng điểm dữ liệu trong dữ liệu X.

```
1. from sklearn.metrics import silhouette_score


2. silhouette_score(X, kmeans.labels_)
```

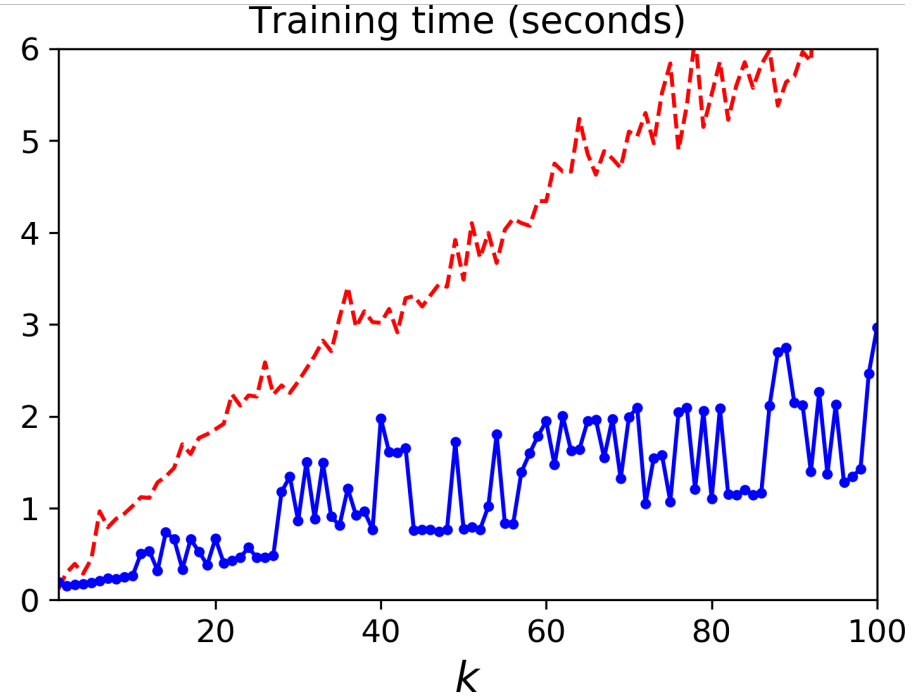
Mini-batch KMeans

- Thay vì tính trọng tâm (centroid) trên toàn bộ dataset, giải thuật này sẽ tìm centroid dựa trên từng batch của dữ liệu.
  + *Batch*: chia dữ liệu ban đầu thành từng phần nhỏ ngẫu nhiên (xem lại Gradient descent để nhớ lại khái niệm batch).
- Giải thuật Mini-batch Kmeans sẽ tăng tốc độ thực thi hơn nhiều lần so với giải thuật gốc, và phù hợp để xử lý trên các bộ dữ liệu có kích thước khá lớn.
- Tuy nhiên, giải thuật này sẽ không tốt khi số cụm k tăng dần.

So sánh giữa Kmeans và Mini-batch Kmeans



tốc độ hội tụ về tâm



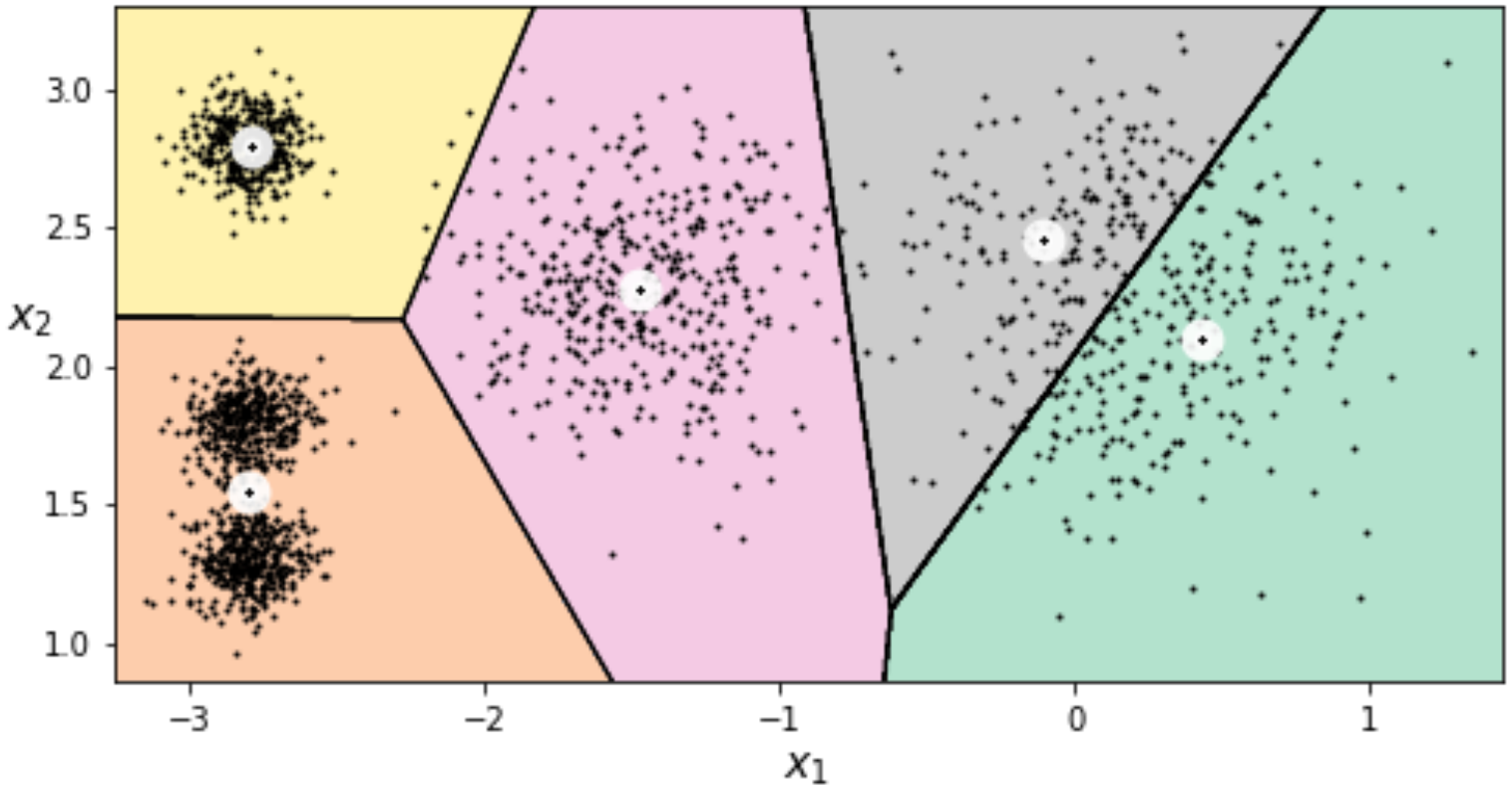
Thời gian thực thi

Sử dụng mini-batch Kmeans

– Sử dụng lớp MiniBatchKMeans trong thư viện sklearns, phân cụm với k = 5.

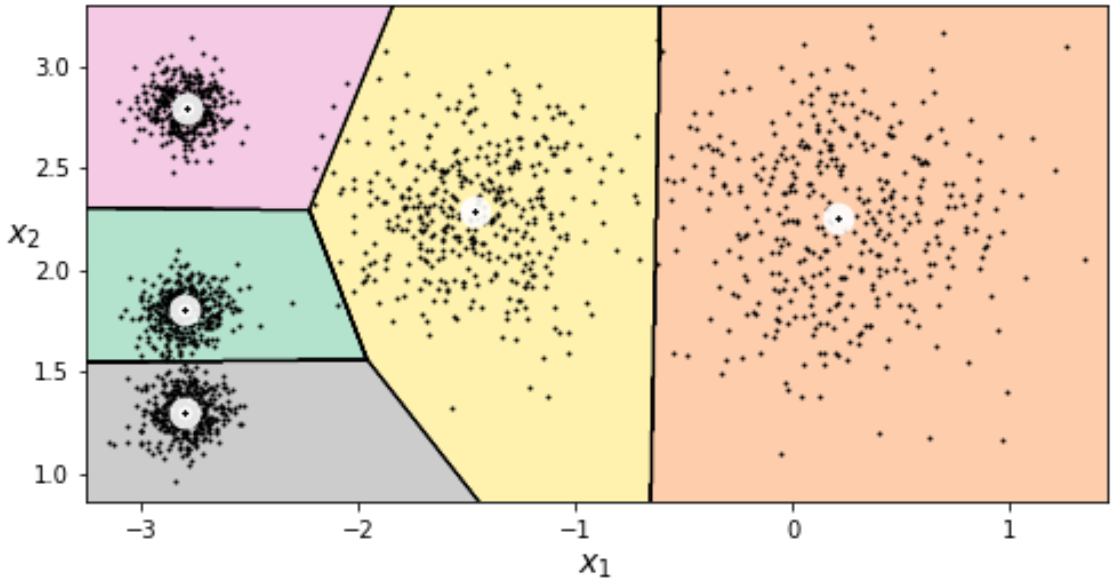
```
1. from sklearn.cluster import MiniBatchKMeans


2. minibatch_kmeans = MiniBatchKMeans(n_clusters=5)
3. minibatch_kmeans.fit(X)
```

Kết quả sau khi phân cụm

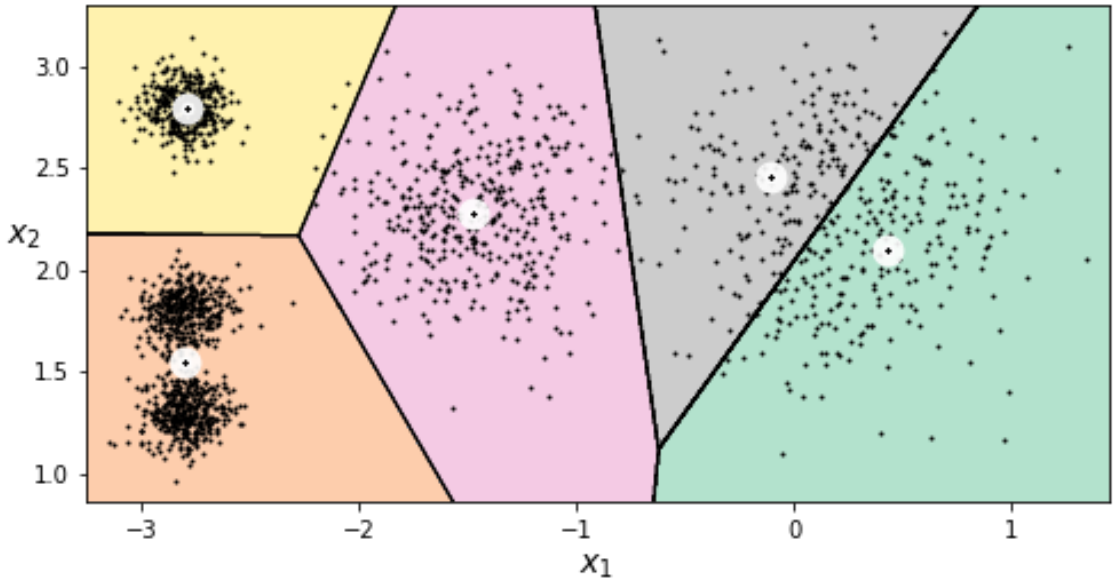


So sánh kết quả (k=5)

K-Means



Mini-batch KMeans



MỘT SỐ DẠNG PHÂN CỤM KHÁC

- Phân cụm tích tụ (*Agglomerative clustering)*: Một dạng phân cụm phân cấp nhưng theo hướng bottom-up.
- Phân cụm Birch.
- Mean-shift.
- DBScan.
- Fuzzy C-Means

GIẢI THUẬT DBSCAN

Khái niệm Active learning

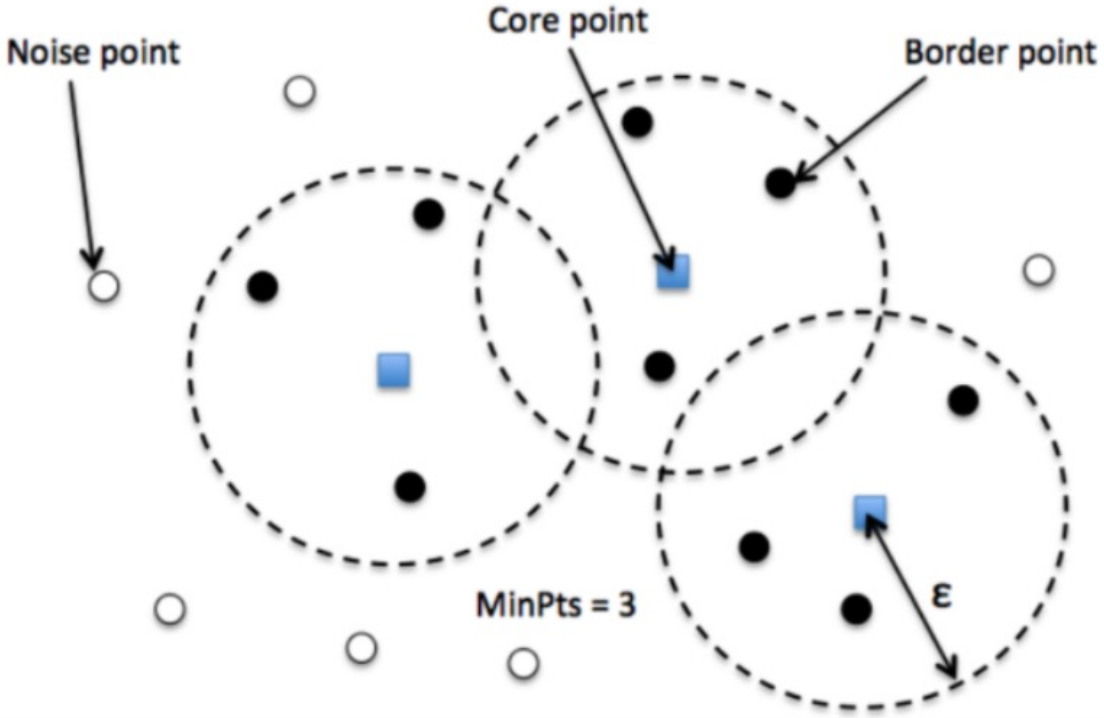
- Ý tưởng: con người (cụ thể là chuyên gia) sẽ can thiệp vào quá trình học của thuật toán.
- Các bước cụ thể:
  + Bước 1: Huấn luyện mô hình trên dữ liệu có nhãn. Sau đó dùng mô hình đã huấn luyện dự đoán cho dữ liệu mới.
  + Bước 2: Các dữ liệu nào mà mô hình dự đoán không chắn chắn thì con người sẽ tiến hành gán nhãn cho dữ liệu đó.
  + Bước 3: Lặp lại Bước 1 và Bước 2 cho tới khi mô hình đạt được hiệu suất tối ưu.

DBSCAN

- Là thuật toán phân cụm dựa trên mật độ.
  - + Phương pháp dựa trên mật độ (density): mở rộng các nhóm cho đến khi mật độ của đối tượng dữ liệu trong vùng lân cận vượt qua ngưỡng (threshold).
  - + Mật độ: số đối tượng nằm trong một bán kính xác định Eps
- Giải thuật DBSCAN sẽ lần lượt mở rộng vùng chứa dữ liệu (cụm) cho tới khi mật độ điểm dữ liệu trong vùng thấp (qua cụm mới).
  - + Ý tưởng chính: một cụm trong không gian dữ liệu là một vùng có mật độ điểm cao được ngăn cách với các cụm khác bằng các vùng liền kề có mật độ điểm thấp

DBScan

- Các siêu tham số chính do người dùng xác định:
  - + Eps: Bán kính lớn nhất của vùng lân cận
  - + MinPts: Số nhỏ nhất các điểm trong vùng lân cận bán kính Eps
- Các điểm đặc biệt:
  - + Core instance: điểm có ít nhất *minPts* điểm trong *vùng lân cận epsilon* của chính nó.
  - + Border instance: điểm có ít nhất một *core instance* nằm ở *vùng lân cận epsilon* nhưng mật độ không đủ minPts điểm.
  - + Noise instance: không thuộc cả 2 điểm trên.



Các bước thực hiện

- Bước 1: Chọn điểm dữ liệu p ngẫu nhiên.
- Bước 2: Tìm tất cả các điểm dữ liệu có mật độ có thể đạt được dựa theo Eps, MinPts.
- Bước 3:
  + Bước 3.1: Nếu p là core instance thì hình thành nhóm.
  + Bước 3.2: Nếu p là border instance thì xét đối tượng tiếp theo.
- Bước 4: Tiếp tục lặp lại cho đến khi xử lý hết các điểm dữ liệu trong bộ dữ liệu.

Ví dụ

- Bộ dữ liệu: make_moon dataset.
- Load dữ liệu:

```
1. from sklearn.datasets import make_moons


2. X, y = make_moons(n_samples=1000, noise=0.05,
   random_state=42)
```

Ví dụ

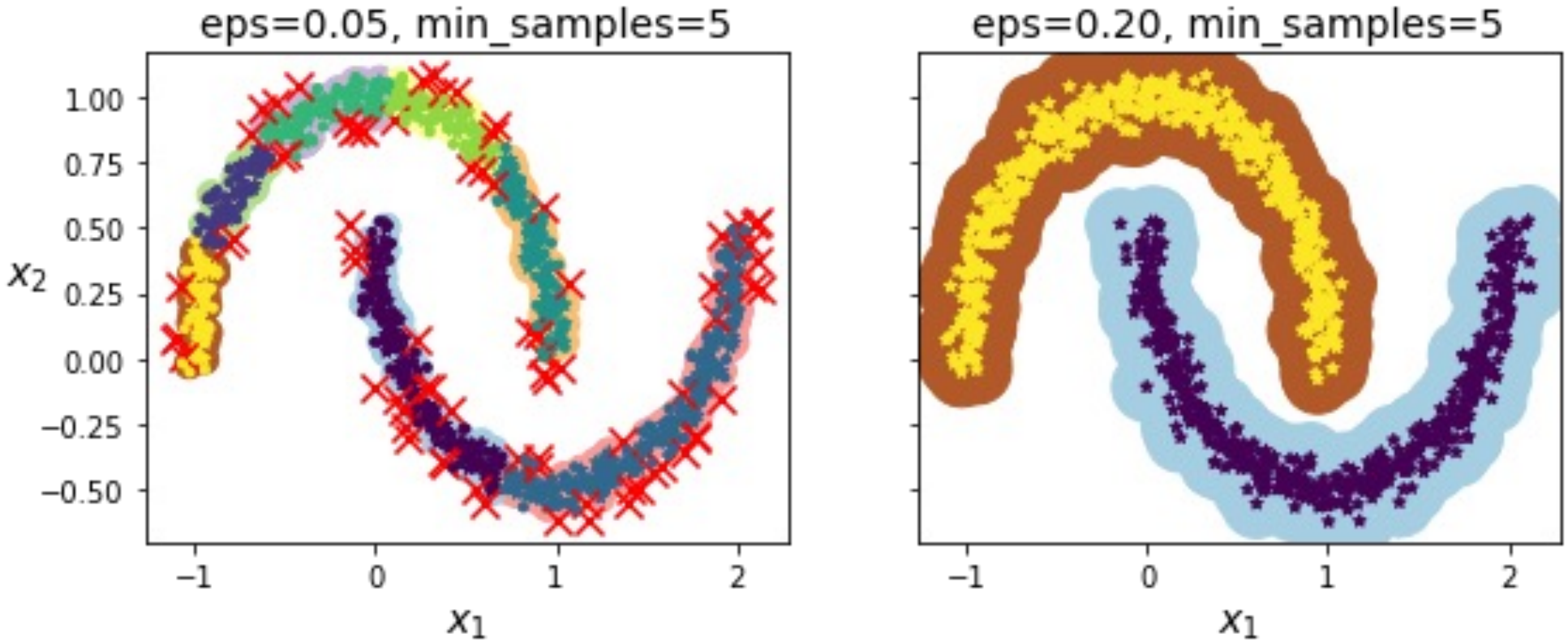
– Thực thi giải thuật DBScan với eps = 0.05 và minpts = 5

```
1. from sklearn.cluster import DBSCAN
2. dbscan = DBSCAN(eps=0.05, min_samples=5)
3. dbscan.fit(X)
```

– Thực thi giải thuật DBScan với eps = 0.2 và minpts = 5

```
1. dbscan2 = DBSCAN(eps=0.2)
2. dbscan2.fit(X)
```

Kết quả


eps=0.05, min_samples=5
eps=0.20, min_samples=5
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
x_2
−1
0
1
2
x_1

Đánh giá DBScan

- Ưu điểm:
  + Không cần xác định trước k.
  + Có khả năng xử lý dữ liệu nhiễu.
  + Xử lý được với dữ liệu phi tuyến và có phân phối đặc biệt.
- Nhược điểm:
  + Phụ thuộc vào giá trị Eps, MinPts.
  + Độ phức tạp lớn đối với dữ liệu có nhiều chiều.

MÔ HÌNH HỖN HỢP GAUSS

Mô hình hỗn hợp Gauss

- Là mô hình dựa trên xác suất, giả định rằng một instance được tạo ra từ một hỗn hợp (mixture) k các phân phối Gaussian (phân phối chuẩn).
- Các tham số chính của mô hình:
  + Giá trị trung bình: μ
  + Ma trận hiệp phương sai $\sum$.
  + Xác suất $\emptyset_k$ cho biết độ lớn của hàm Gauss (cluster weight).

Ý tưởng chính của mô hình

– Tập dữ liệu được tạo ra theo một quá trình xác suất:

$$x^{(i)} \sim N(\mu^{(j)}, \Sigma^{(j)})$$

+ Mỗi mẫu được sinh ngẫu nhiên từ 1 cụm trong k cụm. Xác suất cụm thứ j được chọn xác định bởi trọng số $\varphi^{(j)}$. Chỉ số của cụm được chọn cho mẫu thứ i là $z^{(i)}$.
+ Nếu $z^{(i)} = j$ thì mẫu thứ i đã được gán cho cụm thứ j. Vị trí $x^{(i)}$ của mẫu này được lấy mẫu ngẫu nhiên từ 1 phân phối Gauss với giá trị trung bình $\mu^{(j)}$ và ma trận hiệp phương sai $\Sigma(j)$.

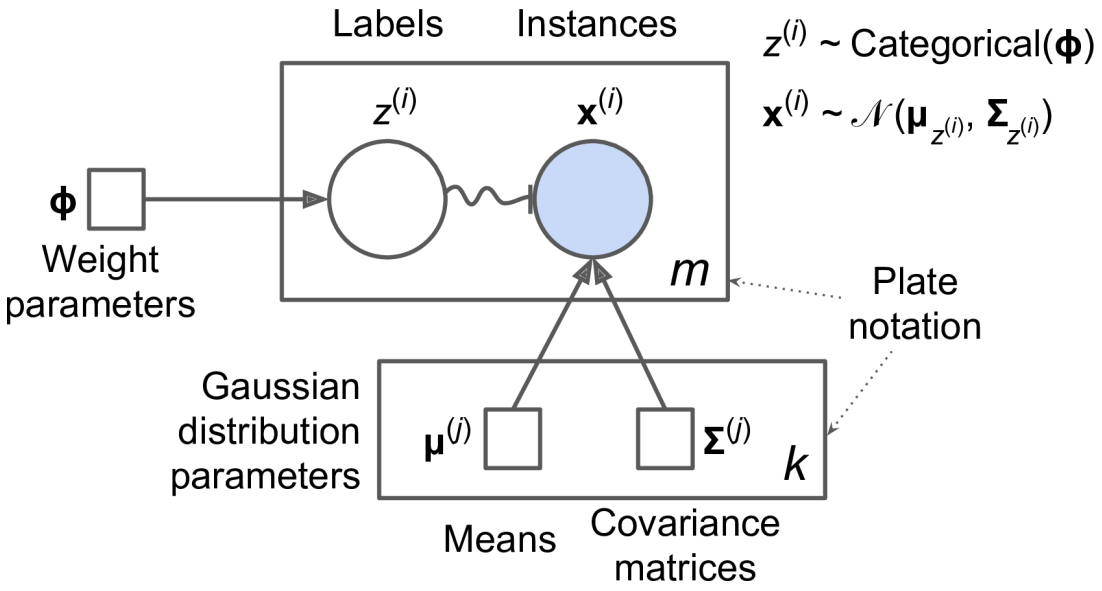
Mô tả thành phần trong mô hình hỗn hợp Gauss

- Hình vuông: tham số
- Hình tròn: biến ngẫu nhiên
- Hình chữ nhật (plate): nội dung được lặp nhiều lần với lần xác định bởi m, k.
- Biến ngẫu nhiên $z^{(i)}$ được rút từ phân phối phân loại (categorical distribution)

$$z^{(i)} \sim \text{Categorical}(\emptyset)$$

- Biến ngẫu nhiên $x^{(i)}$ được rút từ phân phối chuẩn xác định bởi giá trị trung bình và mà trận hiệp phương sai.

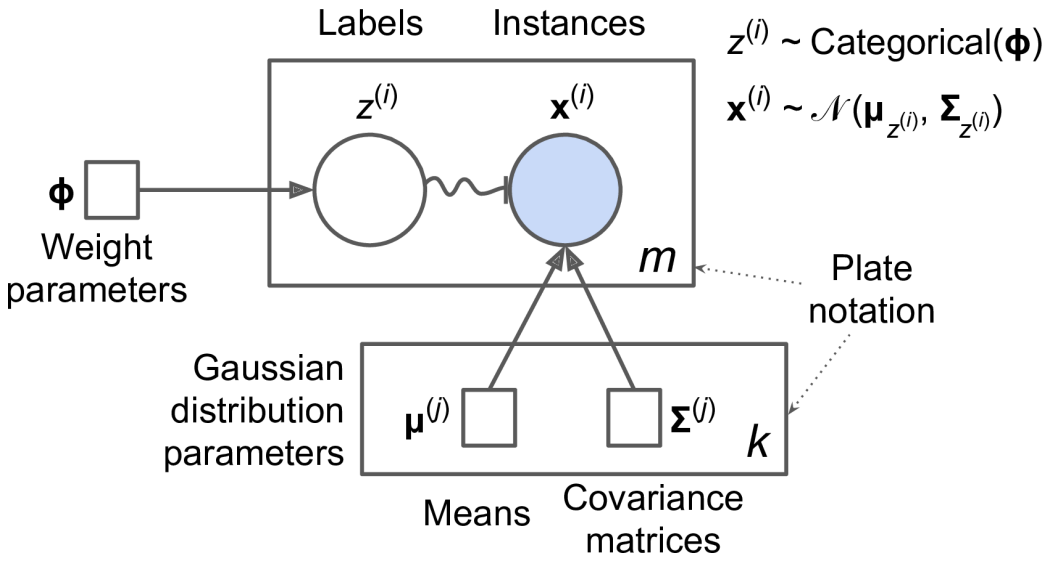
$$x^{(i)} \sim N(\mu, \sum)$$



Graphical model – mô hình đồ thị xác suất

Mô tả quan hệ trong mô hình Gauss

- Mũi tên liền nét: quan hệ phụ thuộc có điều kiện.
- Mũi tên thẳng từ plate này đến plate khác: áp dụng cho tất cả các plate
- Mũi tên uốn cong (công tắc - switch): tùy thuộc vào giá trị của $z^{(i)}$ mà x được rút từ phân phối Gauss nào.
- Nút có màu: là biến quan sát được hay giá trị đã biết (observed variable), biến còn lại là biến ẩn (latent variable).



Hoạt động của mô hình hỗn hợp Gauss

- Mục tiêu: Dựa vào dataset đầu vào X, ước lượng tham số phân cụm $\emptyset, \boldsymbol{\mu}, \sum$ sao cho tìm được phân phối chuẩn xấp xỉ với dữ liệu ban đầu ➔ xác định được cụm chứa dữ liệu.
- Các tham số (k là số cụm):
  + $\emptyset = \{\emptyset_1, \dots \emptyset_k\}$: tham số phân cụm (clustering weight).
  + $\mu = \{\mu_1, \dots \mu_k\}$: các centroids của cụm.
  + $\sum = \{\sum_1, \dots \sum_k\}$: kích thước, hình dạng của các cụm.

Hoạt động của mô hình hỗn hợp Gauss

– Cách thực hiện: sử dụng thuật toán EM (Expectation Maximum):

+ *Bước 1*: Khởi tạo tham số.
+ *Bước 2*: Ước lượng – E. Tính xác suất mẫu $x^{(i)}$ thuộc về từng cụm.
+ *Bước 3*: Cực đại hoá – M. Cập nhật các cụm sử dụng tất cả các mẫu thuộc về.
+ *Bước 4*: Lặp lại Bước 2 và Bước 3 cho đến khi hội tụ.

➔ Giống K-mean, nhưng không chỉ tìm được cụm (μ), mà còn tìm được kích thước và hình dáng của cụm ($\sum$) và trọng số của cụm ($\emptyset$).

Sử dụng mô hình hỗn hợp Gauss

– Tạo dữ liệu ban đầu:

```
1. from sklearn.datasets import make_blobs
2. import numpy as np
3. X1, y1 = make_blobs(n_samples=1000, centers=((4, -4), (0, 0)),
   random_state=42)
4. X1 = X1.dot(np.array([[0.374, 0.95], [0.732, 0.598]]))
5. X2, y2 = make_blobs(n_samples=250, centers=1, random_state=42)
6. X2 = X2 + [6, -8]
7. X = np.r_[X1, X2]
8. y = np.r_[y1, y2]
```

Sử dụng mô hình hỗn hợp Gauss

- Sử dụng thư viện GaussianMixture trong sklearn:

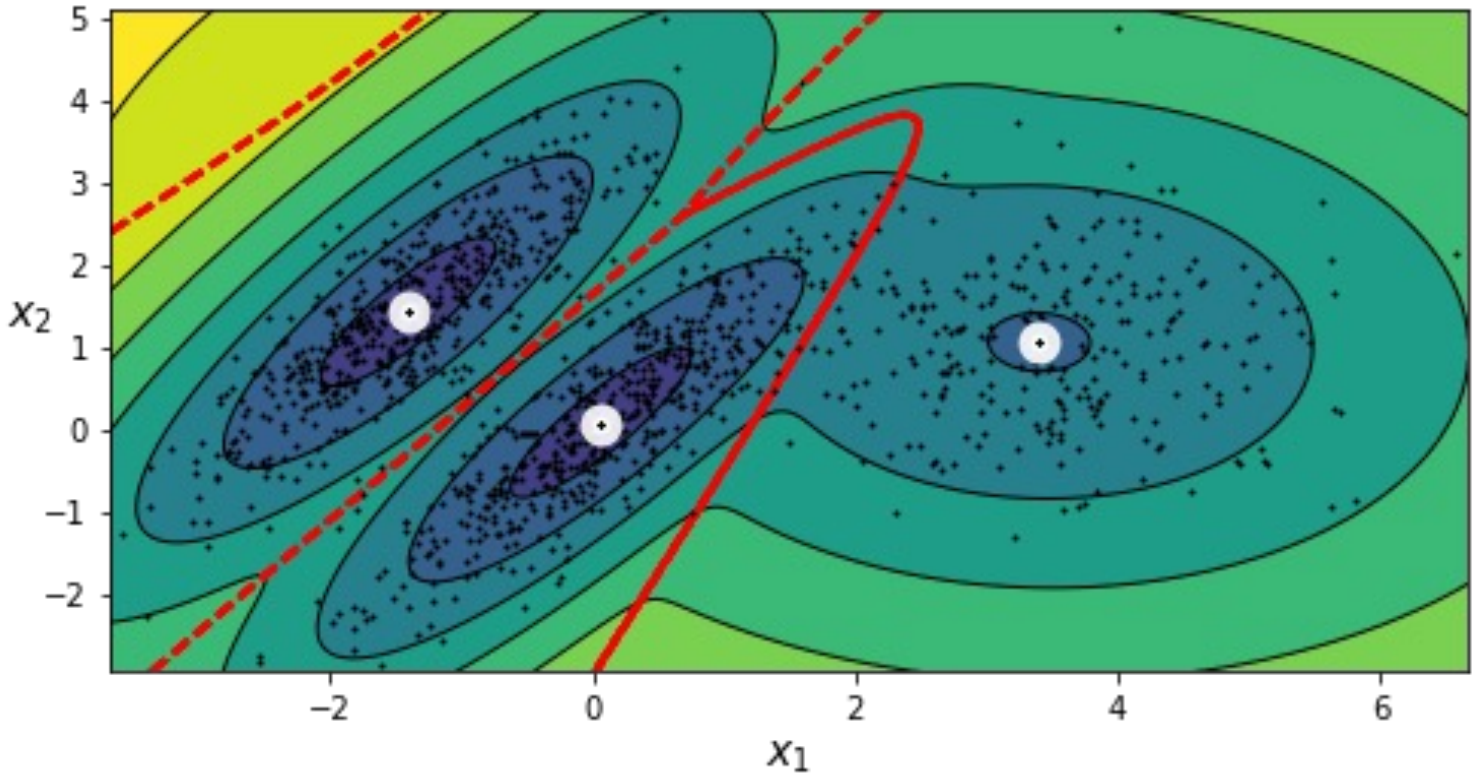
```
1. from sklearn.mixture import GaussianMixture
2. gm = GaussianMixture(n_components=3, n_init=10,
   random_state=42)
3. gm.fit(X)
```

- Kết quả lần lượt các tham số: $\emptyset$, $\boldsymbol{\mu}$, $\sum$

```
1. gm.weights_
2. gm.means_
3. gm.covariances_
```

Kết quả khi phân cụm

```
y_pred = gm.predict(X)
```



Điều chỉnh hình dạng của cụm

- Khi các cụm có hình dạng khác nhau nhiều thì EM sẽ chạy lâu.
- Do đó, để tăng tốc có thể đặt hạn chế dạng cụm bằng cách chọn giá trị siêu tham số *covariance_type*:
  - + spherical: các cụm đều có dạng cầu, có thể kích thước khác nhau.
  - + diag: các cụm dạng elip và các trục elip phải song song với trục tọa độ.
  - + tied: tất cả các cụm phải có cùng hình dạng, kích thước và hướng.

Lựa chọn số cụm cho mô hình GMM

– Không thể sử dụng hệ số Sihouette vì các cụm có hình dạng khác nhau.

➔ Sử dụng 2 hệ số BIC (Bayesian information criterion) và AIC (Akaike information criterion – AIC) để tìm ra số cụm tối ưu dựa trên khái niệm về lý thuyết thông tin.

+ $\boldsymbol{BIC = log(m)p - 2log(\hat{L})}$.

+ $\boldsymbol{AIC = 2p - 2log(\hat{L})}$.

– Trong đó:

+ m: số lượng mẫu

+ p: số lượng tham số của mô hình được học.

+ $\hat{L}$: Giá trị cực đại hóa của hàm likelihood của mô hình

Nhận xét

- BIC và AIC đều không đánh giá cao mô hình có nhiều tham số.
- BIC và AIC thường cho cùng kết quả (cùng k).
- Nếu BIC và AIC cho kết quả khác nhau (khác k) thì mô hình BIC thường đơn giản hơn AIC nhưng thường không tốt bằng AIC, đặc biệt khi tập dữ liệu lớn.

Ứng dụng của phân cụm

- Bài toán phát hiện bất thường (**Anomaly Detection**).
  - + Ý tưởng: Mẫu nào nằm trong vùng mật độ thấp thì được coi là bất thường.
- Các bước thực hiện:
  - + *Bước 1*: Xác định ngưỡng mật độ (*threshold*) ➔ thường ngưỡng này được định sẵn bởi chuyên gia.
  - + *Bước 2*: Xác định những mẫu bất thường dựa vào ngưỡng mật độ.
  - + *Bước 3*: Nếu nhiều mẫu bình thường bị xác định thành bất thường thì điều chỉnh ngưỡng (tăng hoặc giảm giá trị ngưỡng).

TÀI LIỆU THAM KHẢO

Chương 9 của sách: *Hands-on Machine Learning with ScikitLearn, Keras & TensorFlow (2nd).*