ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP ÔN TẬP 5

GIẢNG VIÊN: *ThS. Nguyễn Thị Anh Thư*

LỚP: *CS313.N21*

NHÓM 4:

Nguyễn Duy Đạt - 20520435

Nguyễn Quốc Huy Hoàng - 20520051

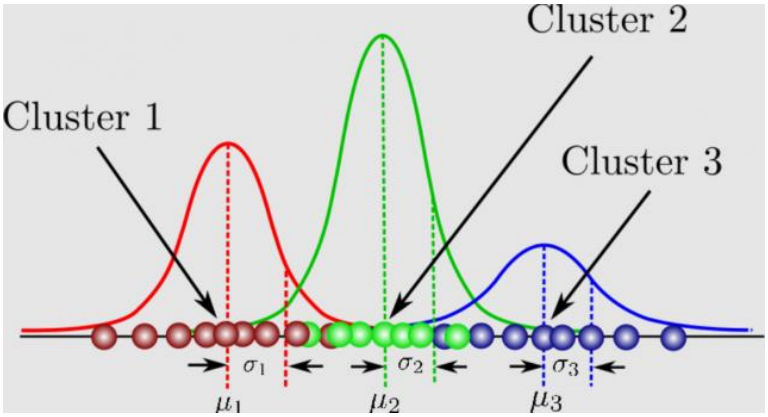
Lê Đoàn Phúc Minh - 20520243

Huỳnh Hoàng Vũ - 20520864

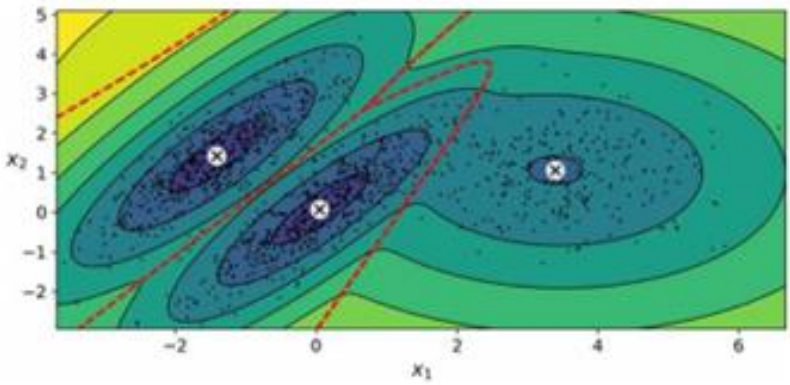
Học kỳ 2 - Năm học 2022-2023

1. Gaussian Mixture Model (GMM)

* GMM là mô hình phân cụm thuộc lớp bài toán học không giám sát mà phân phối xác suất của mỗi một cụm được giả định là *phân phối Gassian đa chiều*. Mô hình được gọi là *Mixture* là vì xác suất của mỗi điểm dữ liệu không chỉ phụ thuộc vào một phân phối *Gaussian* duy nhất mà là kết hợp từ nhiều phân phối *Gaussian* khác nhau từ mỗi cụm..



- Mỗi Cluster elip có hình dạng, kích thước, mật độ và hướng khác nhau.



*Ước lượng hợp lí tối đa:

$$\theta^* = \arg\max_{\theta} p(\mathbf{X}|\theta) = \arg\max_{\theta} \prod_{i=1}^{N} p(\mathbf{x}_i|\theta)$$

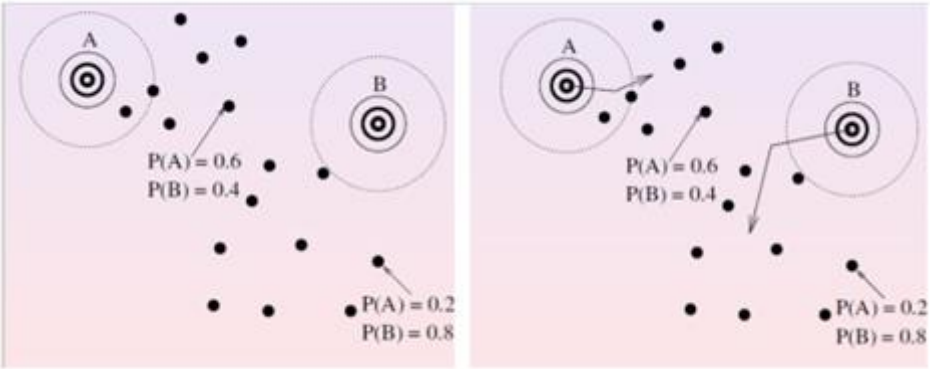
+ θ* là nghiệm tối ưu sao cho giả thiết mô hình GMM khớp nhất với bộ dữ liệu

+ Giải bằng phương pháp EM (Expectation Maximization) để cập nhật dần dần nghiệm θ. MLE bất khả thi trong trường hợp nhiều cụm

*Trong EM, mỗi vòng lặp gồm 2 bước là E-step và M-step

+ E-step: ước lượng phân phối biến ẩn z thể hiện phân phối xác suất của các cluster tương ứng với data và parameters.

+ M-step: Cực đại hóa phân phối xác suất đồng thời (join distribution probability) của data và biến ẩn z.



+ Việc cập nhật lại tham số ở bước M cần xét hàm auxilary

-GMM như 1 dạng tổng quát của K-Means clustering , GMM kết hợp với thông tin về hiệp phương sai của dữ liệu cũng như tâm các phân phối Gaussian tiềm ẩn.

* Đánh giá chất lượng mô hình GMM dựa trên chỉ số BIC (Bayesian Informaion Criteria). Chỉ số này đo lường mức độ hợp lí của model với một bộ tham số được tính dựa trên giá trị tối đa của hàm hợp lí sau:

$$\mathrm{BIC} = k\ln(n) - 2\ln(\hat{L})$$

+ k là số lượng tham số được ước lượng từ model

+ n là số lượng quan sát của bộ dữ liệu

+ L là giá trị ước lượng tối đa của hàm hợp lí

-> BIC càng nhỏ thì mức độ hợp lí của model đối với bộ dữ liệu càng cao

* Ứng dụng của GMM:

+ Gom nhóm khách hàng

+ Phát hiện bất thường, nhận diện lỗi sai sản phẩm

+ Dự đoán giá chứng khoán

+ Dùng cho tác vụ data augmentation vì tính chất sinh dữ liệu

* Demo:

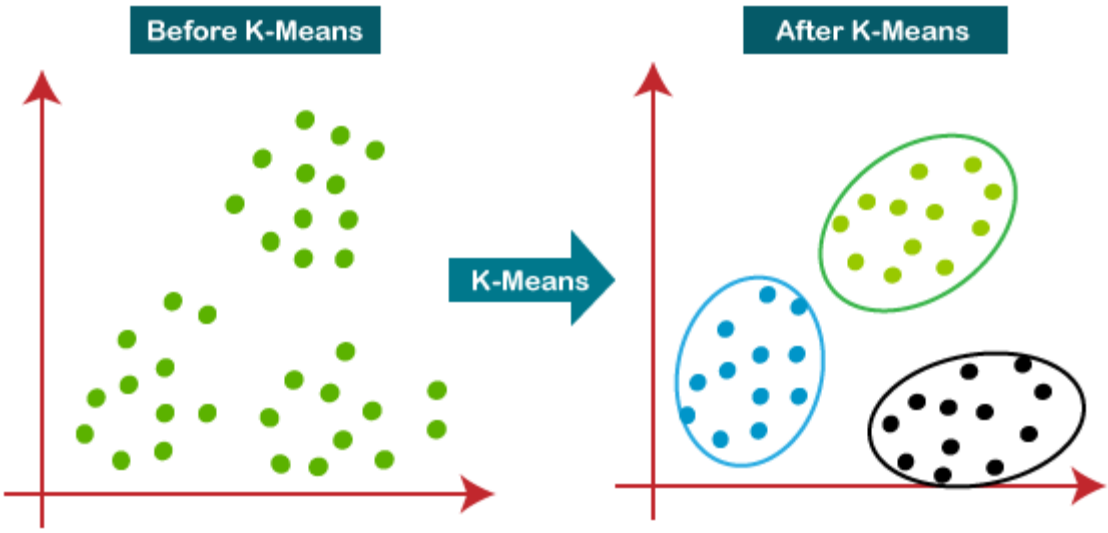
https://colab.research.google.com/drive/1P5fVhDcsrOdCGlQdM_dbeRj-sgXahbQA?usp=sharing#scrollTo=GFSqzyVfMpdH

```
import numpy as np
from sklearn.datasets import load_digits

data, labels = load_digits(return_X_y=True)
(n_samples, n_features), n_digits = data.shape, np.unique(labels).si
ze

print(f"# digits: {n_digits}; # samples: {n_samples}; # features {n_
features}")

from sklearn.mixture import GaussianMixture
X = np.array(data)
gm = GaussianMixture(n_components=10, random_state=0).fit(X)

gm.means_

gm.predict([[ 0.,  0.,  6., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13.,
15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,
4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,
8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,
5.,
       10., 100.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.]]
)
```

➔ Bộ dữ liệu có 10 lớp khi chạy predict thì dự đoán ra 3 tức là thuộc về class 3

2. Thuật toán K-mean

Thuật toán K-Means là một trong những thuật toán phân cụm dữ liệu phổ biến nhất. Tư tưởng chính của thuật toán K-Means là tìm cách phân nhóm các đối tượng đã cho vào K cụm (K là số các cụm được xác định trước, K nguyên dương) sao cho tổng bình phương khoảng cách giữa các đối tượng đến tâm nhóm (centroid) là nhỏ nhất.



Ý tưởng thuật toán K-Means:

1. Đầu tiên, chọn số lượng K cluster cần tạo.
2. Tiếp theo, chọn K điểm từ tập dữ liệu ban đầu để làm trung tâm của K cluster.
3. Gán từng điểm dữ liệu vào cluster gần nhất dựa trên khoảng cách Euclidean giữa điểm dữ liệu và các trung tâm cluster.
4. Tính lại trung tâm của mỗi cluster bằng cách lấy trung bình các điểm dữ liệu trong cluster đó.
5. Lặp lại quá trình gán điểm dữ liệu vào cluster và tính lại trung tâm cho đến khi không có điểm dữ liệu nào thay đổi cluster nữa hoặc đạt đến số lần lặp tối đa.

Thuật toán K-Means có rất nhiều ứng dụng trong các lĩnh vực khác nhau như là:

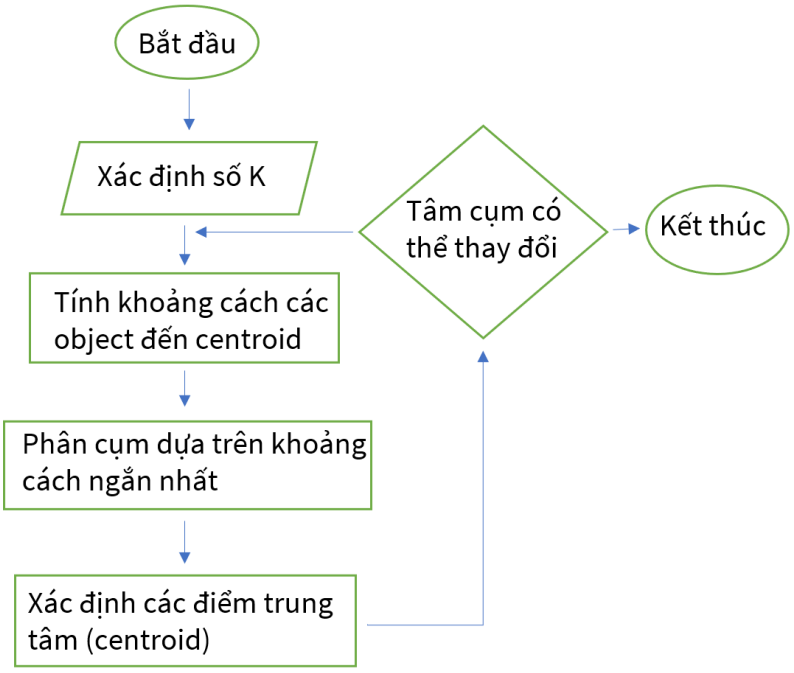
- Phân tích hành vi khách hàng: Khi có dữ liệu về hành vi khách hàng, ta có thể áp dụng thuật toán K-Means để phân nhóm khách hàng và hiểu rõ hơn về sở thích, nhu cầu của từng nhóm khách hàng. Điều này có thể giúp cho các doanh nghiệp xây dựng chiến lược kinh doanh phù hợp với từng nhóm khách hàng.
- Xử lý ảnh và video: K-Means có thể được sử dụng để phân nhóm các đối tượng trong ảnh hoặc video. Ví dụ, khi chụp ảnh một cảnh quan, ta có thể sử dụng K-Means để phân nhóm các màu sắc trong ảnh và thay đổi các thông số màu sắc để tạo ra hiệu ứng ảnh đẹp hơn.
- Phân tích dữ liệu y tế: K-Means có thể được sử dụng để phân nhóm các bệnh nhân dựa trên các chỉ số y tế như huyết áp, đường huyết, cân nặng và chiều cao.

Điều này có thể giúp các chuyên gia y tế đưa ra quyết định chẩn đoán và điều trị phù hợp cho từng bệnh nhân.

- Phân nhóm các quan sát trong khoa học: K-Means có thể được sử dụng để phân nhóm các quan sát trong nghiên cứu khoa học. Ví dụ, trong lĩnh vực sinh học phân tử, K-Means có thể được sử dụng để phân nhóm các tế bào dựa trên các thông số gen di truyền.
- Phân nhóm sản phẩm: Khi có dữ liệu về sản phẩm, ta có thể sử dụng K-Means để phân nhóm các sản phẩm dựa trên các đặc tính chung như giá, chất lượng và tính năng. Điều này giúp cho các doanh nghiệp đưa ra quyết định về chiến lược giá cả và tiếp cận thị trường phù hợp.

Thuật toán K-Means thực hiện qua các bước chính sau:

1. Chọn ngẫu nhiên K tâm (centroid) cho K cụm (cluster). Mỗi cụm được đại diện bằng các tâm của cụm.
2. Tính khoảng cách giữa các đối tượng (objects) đến K tâm (thường dùng khoảng cách Euclidean).
3. Nhóm các đối tượng vào cụm có tâm gần nó nhất.
4. Cập nhật lại tâm của các cụm mới được hình thành.
5. Lặp lại bước 2-4 cho đến khi không có sự thay đổi nào trong việc phân nhóm các đối tượng



Demo: https://colab.research.google.com/drive/1P5fVhDcsrOdCGlQdM_dbeRj-sgXahbQA#scrollTo=p5hNCYbKMwwC

```
from sklearn.cluster import KMeans

# Sử dụng thuật toán K-Means với số cụm (clusters) là 10
kmeans = KMeans(n_clusters=10, random_state=0)
kmeans.fit(X)

kmeans.predict([[ 0.,  0.,  6., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13.,
 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 100.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

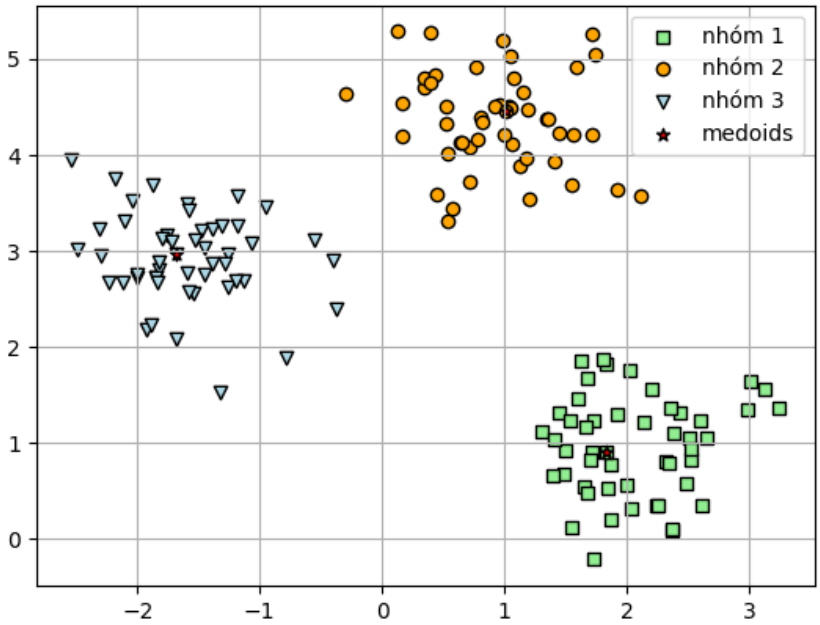
Output:

```
array([5], dtype=int32)
```

=> Bộ dữ liệu có 10 lớp, chạy predict ra 5 có nghĩa là thuộc class 5.

3. Phân cụm K-Medoids

Phân cụm K-Medoids là một kỹ thuật phân chia một tập hợp các điểm dữ liệu thành k nhóm, trong đó mỗi nhóm có một điểm đại diện gọi là medoid. Một medoid là điểm nằm ở vị trí trung tâm nhất trong nhóm, có khoảng cách trung bình đến tất cả các điểm khác trong nhóm là nhỏ nhất. Khoảng cách có thể được đo bằng bất kỳ độ đo nào, chẳng hạn như khoảng cách Euclid, khoảng cách Manhattan, v.v.



Ý tưởng chính của phân cụm K-Medoids

Ý tưởng chính của phân cụm K-Medoids là tìm k medoids sao cho tối thiểu hóa sự bất tương đồng (hay tổng chi phí) của việc phân cụm, được xác định là tổng khoảng cách của tất cả các điểm dữ liệu đến medoid gần nhất của chúng. Thuật toán bắt đầu với việc khởi tạo ngẫu nhiên hoặc tham lam k medoids, sau đó lặp lại việc hoán đổi một medoid với một điểm không phải medoid nếu việc hoán đổi giảm chi phí. Thuật toán kết thúc khi không có hoán đổi nào có thể cải thiện chi phí.

Cách thức hoạt động của phân cụm K-Medoids

Một trong những thuật toán phổ biến nhất cho phân cụm K-Medoids là Partitioning Around Medoids (PAM), được đề xuất bởi Leonard Kaufman và Peter J. Rousseeuw vào năm 1987. Mã giả của PAM có thể được viết như sau:

```
Đầu vào: tập hợp dữ liệu D, số lượng cụm k
Đầu ra: k cụm với các medoid

# Khởi tạo
M <- Chọn ngẫu nhiên k điểm dữ liệu từ D xem như các medoid
Gán các điểm dữ liệu trong D vào medoid gần chúng nhất
Tính toán chi phí ban đầu C

# Tuần tự
Lặp lại
  Với mỗi medoid m trong M
    Với mỗi non-medoid o trong D
      Hoán đổi m và o
      Gán các điểm dữ liệu trong D vào medoid gần chúng nhất
      Tính toán chi phí mới C'
      Nếu C' < C
        Chấp nhận thay đổi, C <- C'
      không thì
        Từ chối thay đổi, hoán đổi lại m và o
cho đến khi C không còn thay đổi

# Trả về
Trả về k cụm với các medoid
```

So sánh với phân cụm K-Means

- K-Medoids sử dụng các điểm dữ liệu thực tế làm tâm của nhóm, trong khi K-Means sử dụng các điểm trung bình làm tâm của nhóm.

- K-Means nhạy cảm với nhiễu và ngoại lệ, vì giá trị trung bình dễ bị ảnh hưởng bởi các giá trị cực đoan. Với K-Medoids, giá trị cực đoan chỉ ảnh hưởng đến khoảng cách của nó và medoid.

- K-Means có khả năng mở rộng hơn K-Medoids vì nó nhanh hơn và hiệu quả hơn trong việc tính toán trung bình của một cụm hơn là tìm ra trung vị của một cụm

- K-Medoids có thể sử dụng bất kỳ độ đo khoảng cách nào để đo khoảng cách, trong khi K-Means thường yêu cầu khoảng cách Euclid để có được các giải pháp hiệu quả. Điều này làm cho K-Medoids linh hoạt và phù hợp với các loại dữ liệu khác nhau hơn K-Means.

- K-Means được ưa chuộng khi tập dữ liệu lớn, các cụm có hình dạng gần cầu hay được tách biệt rõ ràng bởi các khoảng trống lớn, ngoại lệ và nhiễu là tối thiểu.

- K-Medoids được ưa chuộng khi tập dữ liệu nhỏ hoặc vừa, các cụm không được phân biệt rõ ràng hoặc có hình dạng phức tạp, ngoại lệ và nhiễu là đáng kể.

Ứng dụng của phân cụm K-Medoids

Phân cụm K-Medoids có thể được áp dụng cho các loại dữ liệu khác nhau yêu cầu các phương pháp phân cụm mạnh mẽ và linh hoạt. Một số ví dụ là:

- Phân đoạn ảnh: Phân cụm K-Medoids có thể được sử dụng để chia một ảnh thành các vùng gồm các điểm ảnh tương tự nhau, dựa trên màu sắc, cường độ, kết cấu, v.v. Medoid của mỗi vùng có thể được sử dụng làm đặc trưng cho việc xử lý hoặc phân tích tiếp theo.

- Phân cụm tài liệu: Phân cụm K-Medoids có thể được sử dụng để nhóm các tài liệu vào các chủ đề hoặc danh mục, dựa trên nội dung văn bản, từ khóa, siêu dữ liệu, v.v. Medoid của mỗi nhóm có thể được sử dụng làm tóm tắt hoặc tài liệu đại diện cho nhóm.

- Sinh học thông tin (Bioinformatics): Phân cụm K-Medoids có thể được sử dụng để nhóm các chuỗi sinh học, chẳng hạn như DNA, RNA, hoặc protein, dựa trên các độ đo tương đồng hoặc khác biệt của chúng, chẳng hạn như khoảng cách chỉnh sửa, điểm căn chỉnh, v.v. Medoid của mỗi nhóm có thể được sử dụng làm chuỗi đại diện cho nhóm.

- Tiếp thị: phân khúc khách hàng dựa trên sở thích hoặc hành vi của họ.

Demo

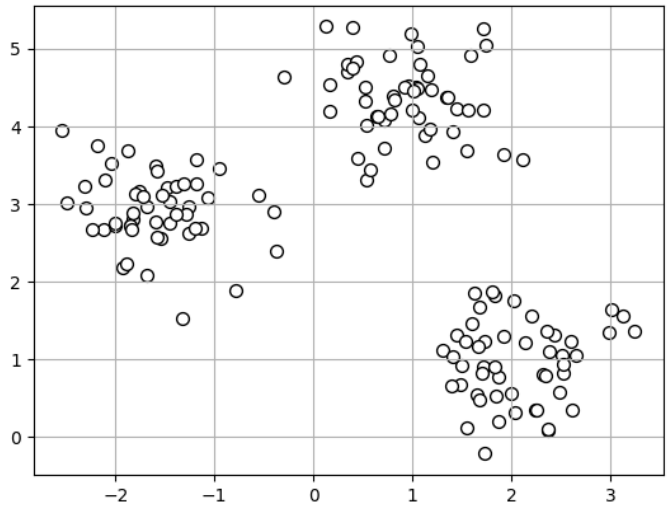
https://colab.research.google.com/drive/1P5fVhDcsrOdCGlQdM_dbeRj-sgXahbQA#scrollTo=lFsDff3WfU1s

Để minh họa và đánh giá phương pháp phân cụm K-Medoids bằng Python, chúng ta sẽ sử dụng một tập dữ liệu đơn giản của các điểm 2D tạo thành ba nhóm. Chúng ta sẽ sử dụng thư viện scikit-learn để tạo ra dữ liệu và vẽ kết quả, và thư viện scikit-learn-extra để thực hiện thuật toán K-Medoids.

Đầu tiên, chúng ta nhập các thư viện cần thiết và tạo ra 150 điểm thuộc ba nhóm với một số nhiễu.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn_extra.cluster import KMedoids

# Tạo ra 150 điểm với 3 tâm và một số nhiễu
X, y = make_blobs(n_samples=150, n_features=2,
                  centers=3, cluster_std=0.5,
                  shuffle=True, random_state=0)

# Vẽ dữ liệu đã tạo ra
plt.scatter(X[:,    0],    X[:,    1],    c='white',    marker='o',
edgecolor='black', s=50)
plt.grid()
plt.show()
```

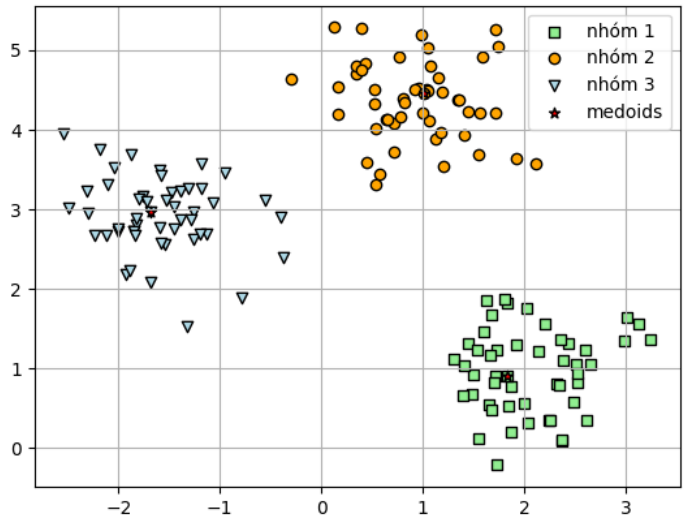
Kết quả



Tiếp theo, chúng ta áp dụng thuật toán phân cụm K-Medoids với k=3 và khoảng cách Euclid làm độ đo khoảng cách. Chúng ta cũng vẽ các nhóm kết quả và medoids của chúng.

```
# Áp dụng phân cụm K-Medoids với k=3 và khoảng cách Euclid
kmeds = KMedoids(n_clusters=3, metric='euclidean', random_state=0)
y_kmeds = kmeds.fit_predict(X)

# Vẽ dữ liệu đã được phân cụm
plt.scatter(X[y_kmeds == 0, 0], X[y_kmeds == 0, 1],
  c='lightgreen', marker='s', edgecolor='black',
  label='nhóm 1')
plt.scatter(X[y_kmeds == 1, 0], X[y_kmeds == 1, 1],
  c='orange', marker='o', edgecolor='black',
  label='nhóm 2')
plt.scatter(X[y_kmeds == 2, 0], X[y_kmeds == 2, 1],
  c='lightblue', marker='v', edgecolor='black',
  label='nhóm 3')
plt.scatter(kmeds.cluster_centers_[:, 0], kmeds.cluster_centers_[:,
1],
  c='red', marker='*', edgecolor='black',
  label='medoids')
plt.legend(scatterpoints=1)
plt.grid()
plt.show()
```

Kết quả



Chúng ta có thể thấy rằng thuật toán phân cụm K-Medoids đã phân cụm thành công dữ liệu thành ba nhóm và tìm ra medoids cho mỗi nhóm. Các medoids được đánh dấu bằng các ngôi sao màu đỏ trong biểu đồ.

```
from sklearn.metrics import silhouette_score, adjusted_rand_score

# Tính toán hệ số silhouette
silhouette = silhouette_score(X, y_kmeds)
print('Hệ số silhouette: %.3f' % silhouette)

# Tính toán chỉ số Rand điều chỉnh
ari = adjusted_rand_score(y, y_kmeds)
print('Chỉ số Rand điều chỉnh: %.3f' % ari)
```

Kết quả

```
Hệ số silhouette: 0.714
Chỉ số Rand điều chỉnh: 1.000
```

Chúng ta có thể thấy rằng cả hệ số silhouette và ARI đều gần bằng 1, cho thấy một kết quả phân cụm rất tốt phù hợp với các nhãn sự thật.

4. Thuật toán DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Thuật toán DBSCAN (hay tên đầy đủ là Density-Based Spatial Clustering of Applications with Noise) là thuật toán gom nhóm hoạt động dựa trên giả thuyết các nhóm, cụm (Cluster) là các vùng dày đặc trong không gian được phân tách

bằng các vùng có mật độ thấp hơn. Thuật toán này được đề nghị bởi Martin Ester et al. vào năm 1996.

Thuật toán này tìm và gom nhóm các điểm dữ liệu đứng gần nhau ở một khoảng cách đủ nhỏ và khiến cho vùng đó trở nên dày đặc. Nó có thể gom nhóm trong bộ dữ liệu có không gian lớn bằng cách kiểm tra mật độ của các điểm dữ liệu. Thuật toán này linh hoạt với các điểm Outlier, không cần phải biết trước số Centroid ban đầu như K-Means.

Các trọng số trong thuật toán:

- **minPoint:** Theo quy tắc chung, *minPoint* tối thiểu có thể được tính theo số chiều *D* trong tập dữ liệu đó là $minPoint \geq D + 1$. *minPoint* phải được chọn ít nhất là 3. Tuy nhiên, các giá trị lớn hơn thường tốt hơn cho các tập dữ liệu có nhiễu và kết quả phân cụm thường hợp lý hơn. Theo quy tắc chung thì thường chọn *minPoint=2*dim*. Trong trường hợp dữ liệu có nhiễu hoặc có nhiều quan sát lặp lại thì cần lựa chọn giá trị *minPoint* lớn hơn nữa tương ứng với những bộ dữ liệu lớn.
- **ε:** Giá trị ε *(epsilon)* có thể được chọn bằng cách vẽ một biểu đồ k-distance. Đây là biểu đồ thể hiện giá trị khoảng cách trong thuật toán K-Means clustering đến $k = minPoint-1$ điểm láng giềng gần nhất. Ứng với mỗi điểm chúng ta chỉ lựa chọn ra khoảng cách lớn nhất trong *k* khoảng cách. Những khoảng cách này trên đồ thị được sắp xếp theo thứ tự giảm dần. Các giá trị tốt của ε là vị trí mà biểu đồ này cho thấy xuất hiện một điểm khuỷ tay (elbow point): Nếu ε được chọn quá nhỏ, một phần lớn dữ liệu sẽ không được phân cụm và được xem là nhiễu; trong khi đối với giá trị ε quá cao, các cụm sẽ hợp nhất và phần lớn các điểm sẽ nằm trong cùng một cụm. Nói chung, các giá trị nhỏ của ε được ưu tiên hơn và theo quy tắc chung, chỉ một phần nhỏ các điểm nên nằm trong vùng lân cận.
- **Hàm khoảng cách:** Việc lựa chọn hàm khoảng cách có mối liên hệ chặt chẽ với lựa chọn ε và tạo ra ảnh hưởng lớn tới kết quả. Điểm quan trọng trước tiên đó là chúng ta cần xác định một thước đo hợp lý về độ khác biệt (disimilarity) cho tập dữ liệu trước khi có thể chọn tham số ε. Khoảng cách được sử dụng phổ biến nhất là euclidean distance.

Ý tưởng thuật toán DBSCAN:

- **Bước 1:** Thuật toán lựa chọn một điểm dữ liệu bất kì. Sau đó tiến hành xác định các điểm lõi và điểm biên thông qua vùng lân cận ε bằng cách lan truyền theo liên kết chuỗi các điểm thuộc cùng một cụm. Trong quá trình

xác định cụm, nếu đạt đủ số lượng điểm minPoint trong vùng thì sẽ cho các điểm đó vào cụm.

- **Bước 2:** Cụm hoàn toàn được xác định khi không thể mở rộng được thêm. Khi đó lặp lại đệ qui toàn bộ quá trình với điểm khởi tạo trong số các điểm dữ liệu còn lại để xác định một cụm mới.

Một số ứng dụng của DBSCAN:

- Phân cụm, gom nhóm văn bản (Document Clustering)
- Lõi hệ thống khuyến nghị (Recommendation Engine)
- Phân vùng ảnh (Image Segmentation)
- Phân vùng thị trường (Market Segmentation)
- Gom nhóm kết quả tìm kiếm (Search Result Grouping)
- Phát hiện bất thường (Anomaly Detection)

Ưu điểm và nhược điểm của DBSCAN:

Ưu điểm:

- Có thể tự loại bỏ được các dữ liệu nhiễu nhờ có tính linh hoạt
- Hoạt động tốt đối với những dữ liệu có hình dạng phân phối đặc thù
- Tốc độ tính toán nhanh

Nhược điểm:

- Thường không hiệu quả đối với những dữ liệu có phân phối đều khắp nơi
- Các trọng số có ảnh hưởng rất lớn đến kết quả phân cụm

*Demo với bộ dữ liệu ‘digits’:

https://colab.research.google.com/drive/1P5fVhDcsrOdCGlQdM_dbeRj-sgXahbQA?usp=sharing

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=20, min_samples=3).fit(X)

labels = dbscan.labels_
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Estimated number of clusters: %d" % n_clusters_)
print("Estimated number of noise points: %d" % n_noise_)

dbscan.fit_predict(
```

```
  [[ 0.,  0.,  6., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
  15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
  12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
  0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
  10., 100.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0. ]]
)
```

Output:

```
Estimated number of clusters: 24
Estimated number of noise points: 329

array[-1]
```

Như vậy, với parameter được cấu hình ở trên, bộ dữ liệu có 24 lớp với số điểm nhiễu là 329, dữ liệu này ở lớp -1 tức là bị xem như dữ liệu nhiễu

Demo: https://colab.research.google.com/drive/1P5fVhDcsrOdCGlQdM_dbeRj-sgXahbQA#scrollTo=svmuScBEF0c7

5. Chỉ số phân cụm

Phân cụm là một cách để tìm các nhóm điểm dữ liệu tương tự trong một tập dữ liệu mà không sử dụng bất kỳ nhãn nào. Nó có thể giúp với khám phá dữ liệu, giảm chiều, phát hiện bất thường và nhiều hơn nữa. Nhưng làm thế nào để chúng ta biết nếu kết quả phân cụm là tốt hay không? Làm thế nào để chọn thuật toán phân cụm hoặc tham số tốt nhất cho một tập dữ liệu? Làm thế nào để quyết định số lượng cụm để sử dụng? Các chỉ số phân cụm có thể giúp chúng ta trả lời những câu hỏi này.

Các chỉ số phân cụm là các số đo một số khía cạnh của kết quả phân cụm, chẳng hạn như độ hợp lệ, ổn định hoặc khả năng diễn giải của nó. Có nhiều loại chỉ số phân cụm khác nhau và mỗi loại có những ưu và nhược điểm riêng. Một số chỉ số phổ biến nhất là:

- **Các chỉ số nội bộ**: Các chỉ số này chỉ sử dụng dữ liệu và các cụm để đo lường độ tốt của các cụm. Chúng không cần bất kỳ nhãn hoặc thông tin bên ngoài nào. Ví dụ về các chỉ số nội bộ là silhouette coefficient (hệ số bóng ma), Davies-Bouldin index, Calinski-Harabasz index, v.v. Các chỉ số nội bộ tốt cho việc so sánh các thuật toán hoặc tham số khác nhau trên cùng một tập dữ liệu hoặc để tìm số lượng cụm tốt nhất bằng các phương pháp như phương pháp khuỷu tay (elbow method) hoặc thống kê khoảng trống (gap statistic).

- **Các chỉ số bên ngoài**: Các chỉ số này sử dụng nhãn hoặc thông tin bên ngoài để đo lường mức độ khớp của các cụm với cấu trúc thực sự của dữ liệu. Chúng tốt cho việc kiểm tra hiệu suất của thuật toán phân cụm trên các tập dữ liệu có nhãn, chẳng hạn như các tác vụ phân loại hoặc phân đoạn. Ví dụ về các chỉ số bên ngoài là adjusted Rand index, normalized mutual information (thông tin tương hỗ chuẩn hóa), F1-score, v.v. Các chỉ số bên ngoài thường cao hơn khi các cụm là sạch và rõ ràng hơn.

- **Các chỉ số tương đối**: Các chỉ số này so sánh hai hoặc nhiều kết quả phân cụm trên cùng một tập dữ liệu và đo lường sự giống hoặc khác nhau của chúng. Chúng tốt cho việc kiểm tra độ ổn định hoặc của thuật toán phân cụm dưới các điều kiện khác nhau, chẳng hạn như nhiễu, ngoại lệ hoặc các khởi tạo khác nhau. Ví dụ về các chỉ số tương đối là variation of information (biến thiên của thông tin), cluster matching index (chỉ số phù hợp cụm), cophenetic correlation coefficient (hệ số tương quan cophenetic), v.v. Các chỉ số tương đối thường thấp hơn khi các kết quả phân cụm giống nhau hơn.

Demo

https://colab.research.google.com/drive/1P5fVhDcsrOdCGlQdM_dbeRj-sgXahbQA#scrollTo=-vmvzCazMX0i

Chúng ta sẽ minh họa cách sử dụng một số chỉ số phân cụm trong Python với các ví dụ từ thư viện scikit-learn. Chúng ta sẽ sử dụng tập dữ liệu Iris làm ví dụ, bao gồm 150 mẫu của ba loại hoa iris khác nhau, mỗi loại có bốn đặc trưng: chiều dài đài hoa, chiều rộng đài hoa, chiều dài cánh hoa và chiều rộng cánh hoa.

Đầu tiên, chúng ta cần nhập một số thư viện và tải tập dữ liệu:

```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score,
calinski_harabasz_score
from sklearn.metrics import adjusted_rand_score,
normalized_mutual_info_score, f1_score
from sklearn.metrics import pairwise_distances
```

```
# Load Iris dataset
iris = datasets.load_iris()
X = iris.data # Features
y = iris.target # Labels
```

Tiếp theo, chúng ta sẽ sử dụng thuật toán K-means để phân cụm dữ liệu thành ba cụm (giả sử chúng ta biết số lượng cụm thực):

```
# Use K-means with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(X)
y_pred = kmeans.labels_ # Predicted labels
```

Bây giờ chúng ta có thể sử dụng một số chỉ số nội bộ để đo lường độ tốt của các cụm:

```
# Calculate silhouette coefficient
sil = silhouette_score(X, y_pred)
print(f"Silhouette coefficient: {sil:.3f}")

# Calculate Davies-Bouldin index
db = davies_bouldin_score(X, y_pred)
print(f"Davies-Bouldin index: {db:.3f}")

# Calculate Calinski-Harabasz index
ch = calinski_harabasz_score(X, y_pred)
print(f"Calinski-Harabasz index: {ch:.3f}")
```

Kết quả

```
Silhouette coefficient: 0.553
Davies-Bouldin index: 0.662
Calinski-Harabasz index: 561.628
```

Silhouette coefficient nằm trong khoảng từ -1 đến 1, trong đó các giá trị cao hơn có nghĩa là các cụm tốt hơn. Một giá trị gần với 0 có nghĩa là một số điểm nằm gần biên giới giữa hai cụm. Một giá trị gần với -1 có nghĩa là một số điểm nằm trong cụm sai.

Davies-Bouldin index nằm trong khoảng từ 0 đến vô cùng, trong đó các giá trị thấp hơn có nghĩa là các cụm tốt hơn. Nó đo lường sự tương đồng trung bình giữa mỗi cụm và cụm tương tự nhất của nó, trong đó sự tương đồng được định nghĩa là tỷ lệ của khoảng cách trong cụm với khoảng cách giữa các cụm. Một giá trị cao có nghĩa là một số cụm đang chồng lấn hoặc không được phân tách tốt.

Calinski-Harabasz index nằm trong khoảng từ 0 đến vô cùng, trong đó các giá trị cao hơn có nghĩa là các cụm tốt hơn. Nó đo lường tỷ lệ của phương sai giữa các cụm với phương sai trong các cụm. Một giá trị cao có nghĩa là các cụm là gọn gàng và được phân tách tốt.

Chúng ta cũng có thể sử dụng một số chỉ số bên ngoài để đo lường mức độ khớp của các cụm với nhãn thực:

```
# Calculate adjusted Rand index
ari = adjusted_rand_score(y, y_pred)
print(f"Adjusted Rand index: {ari:.3f}")

# Calculate normalized mutual information
nmi = normalized_mutual_info_score(y, y_pred)
print(f"Normalized mutual information: {nmi:.3f}")

# Calculate F1-score
f1 = f1_score(y, y_pred, average='macro')
print(f"F1-score: {f1:.3f}")
```

Kết quả

```
Adjusted Rand index: 0.730
Normalized mutual information: 0.758
F1-score: 0.273
```

Adjusted Rand index nằm trong khoảng từ -1 đến 1, trong đó các giá trị cao hơn có nghĩa là các cụm tốt hơn. Nó đo lường sự tương đồng giữa hai bộ nhãn, điều chỉnh cho ngẫu nhiên. Một giá trị gần với 0 có nghĩa là các nhãn là ngẫu nhiên. Một giá trị gần với 1 có nghĩa là các nhãn là giống nhau.

Normalized mutual information nằm trong khoảng từ 0 đến 1, trong đó các giá trị cao hơn có nghĩa là các cụm tốt hơn. Nó đo lường lượng thông tin chung giữa hai bộ nhãn, chuẩn hóa bởi entropy của chúng. Một giá trị gần với 0 có nghĩa là các nhãn là ngẫu nhiên. Một giá trị gần với 1 có nghĩa là các nhãn là giống nhau.

F1-score nằm trong khoảng từ 0 đến 1, trong đó các giá trị cao hơn có nghĩa là các cụm tốt hơn. Nó là trung bình điều hoà của độ chính xác và độ bao phủ, trong đó độ chính xác là phần trăm của điểm được gán đúng trong mỗi cụm, và độ bao phủ là phần trăm của điểm trong mỗi lớp thực được gán đúng. Một giá trị gần với 0 có nghĩa là các cụm là trống hoặc sai. Một giá trị gần với 1 có nghĩa là các cụm là hoàn hảo.

Cuối cùng, chúng ta có thể sử dụng một số chỉ số tương đối để so sánh hai hoặc nhiều kết quả phân cụm trên cùng một tập dữ liệu:

```
# Use K-means with 2 clusters
kmeans2 = KMeans(n_clusters=2, random_state=0)
kmeans2.fit(X)
y_pred2 = kmeans2.labels_ # Predicted labels

# Calculate variation of information
vi = pairwise_distances(y_pred.reshape(-1, 1), y_pred2.reshape(-1,
1), metric='hamming')[0][0]
print(f"Variation of information: {vi:.3f}")

# Calculate cluster matching index
cmi = np.sum(y_pred == y_pred2) / len(y_pred)
print(f"Cluster matching index: {cmi:.3f}")

# Calculate cophenetic correlation coefficient
coph = np.corrcoef(pairwise_distances(X),
pairwise_distances(X[y_pred.argsort()]))[0][1]
print(f"Cophenetic correlation coefficient: {coph:.3f}")
```

Kết quả

```
Variation of information: 1.000
Cluster matching index: 0.020
Cophenetic correlation coefficient: 0.993
```

Variation of information nằm trong khoảng từ 0 đến vô cùng, trong đó các giá trị thấp hơn có nghĩa là các kết quả phân cụm giống nhau hơn. Nó đo lường lượng thông tin bị mất hoặc được thêm vào khi chuyển từ một kết quả phân cụm sang một kết quả khác.

Cluster matching index nằm trong khoảng từ 0 đến 1, trong đó các giá trị cao hơn có nghĩa là các kết quả phân cụm giống nhau hơn. Nó đo lường tỷ lệ của điểm có cùng nhãn cụm trong cả hai kết quả phân cụm.

Cophenetic correlation coefficient nằm trong khoảng từ -1 đến 1, trong đó các giá trị cao hơn có nghĩa là các kết quả phân cụm giống nhau hơn. Nó đo lường mối tương quan giữa các khoảng cách ban đầu giữa các điểm và các khoảng cách giữa các trung tâm cụm của chúng.

Để kết luận, các chỉ số phân cụm là những công cụ hữu ích để đánh giá và so sánh các kết quả phân cụm khác nhau trên một tập dữ liệu cho trước. Tuy nhiên, chúng không phải là hoàn hảo và chúng có thể không luôn luôn thống nhất với nhau hoặc với trực giác của con người. Do đó, điều quan trọng là phải hiểu được các giả định và hạn chế của chúng và sử dụng chúng một cách thận trọng và thông minh.