**3.1.1. Ôn tập lý thuyết**

**1. Giải thuật K-Means hoạt động như thế nào? Các bước chính trong quy trình phân cụm**

**K-Means** là thuật toán phân cụm (clustering) dựa trên khoảng cách, nhằm phân chia dữ liệu thành **K cụm** sao cho các điểm trong cùng một cụm càng giống nhau càng tốt, và khác biệt so với các cụm khác.

**Quy trình chính:**

1. **Chọn số lượng cụm K**: Người dùng xác định trước số cụm cần phân loại.
2. **Khởi tạo centroid**: Chọn ngẫu nhiên K điểm làm tâm cụm ban đầu.
3. **Gán điểm vào cụm gần nhất**: Tính khoảng cách từ mỗi điểm đến các centroid, gán điểm vào cụm có centroid gần nhất.
4. **Cập nhật centroid**: Tính trung bình các điểm trong mỗi cụm để cập nhật vị trí centroid mới.
5. **Lặp lại bước 3 và 4**: Cho đến khi centroid ổn định hoặc đạt số vòng lặp tối đa.

**2. Tại sao cần chọn số lượng cụm K trước khi chạy K-Means? Làm thế nào để xác định giá trị K tối ưu?**

* **Tại sao cần chọn K:** K-Means là thuật toán giám sát số cụm, nghĩa là **không tự động xác định K**. Nếu K quá lớn hoặc quá nhỏ, phân cụm sẽ không phản ánh đúng cấu trúc dữ liệu.
* **Cách xác định K tối ưu:**
  1. **Elbow Method:** Vẽ đồ thị **WCSS (Within-Cluster Sum of Squares)** theo K; điểm "khuỷu" (elbow) là K tối ưu.
  2. **Silhouette Score:** Tính chỉ số silhouette cho mỗi K, chọn K có giá trị silhouette cao nhất.
  3. **Gap Statistic:** So sánh WCSS thực với dữ liệu ngẫu nhiên.

**3. Hàm mục tiêu (objective function) của K-Means**

Hàm mục tiêu của K-Means là **tổng bình phương khoảng cách từ các điểm tới centroid của cụm**:

* : cụm i
* : centroid của cụm i
* : khoảng cách bình phương từ điểm x đến centroid

**Ý nghĩa:** Giảm thiểu tổng khoảng cách này giúp các điểm trong cùng cụm càng gần nhau càng tốt.

**4. Hạn chế của K-Means**

* Nhạy cảm với **giá trị khởi tạo centroid** → kết quả khác nhau khi chạy nhiều lần.
* Phù hợp với cụm **hình cầu** và **kích thước tương đồng**; không tốt cho dữ liệu phi hình cầu hoặc mật độ khác nhau.
* Nhạy cảm với **outliers**; giá trị ngoại lai làm centroid lệch.
* **Cần xác định K trước**, không tự động chọn số cụm.

**5. Code mẫu triển khai K-Means bằng Python**

# Import thư viện

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

# Tạo dữ liệu mẫu

X, y = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.6, random\_state=42)

# Khởi tạo và chạy K-Means

kmeans = KMeans(n\_clusters=4, init='k-means++', n\_init=10, random\_state=42)

kmeans.fit(X)

# Lấy nhãn và centroid

labels = kmeans.labels\_

centroids = kmeans.cluster\_centers\_

# Vẽ kết quả

plt.scatter(X[:,0], X[:,1], c=labels, cmap='viridis', s=50)

plt.scatter(centroids[:,0], centroids[:,1], color='red', marker='X', s=200)

plt.title('K-Means Clustering')

plt.show()

**Các bước thực hiện:**

1. Chuẩn bị dữ liệu.
2. Khởi tạo K-Means với K cụm, sử dụng k-means++ để chọn centroid tốt.
3. Huấn luyện mô hình với .fit().
4. Lấy nhãn cụm và centroid.
5. Vẽ kết quả trực quan.

**6. Chọn số cụm K tối ưu trong Python**

**Ví dụ: Elbow Method và Silhouette Score**

from sklearn.metrics import silhouette\_score

wcss = []

sil\_scores = []

K\_range = range(2, 11)

for k in K\_range:

kmeans = KMeans(n\_clusters=k, init='k-means++', n\_init=10, random\_state=42)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

sil\_scores.append(silhouette\_score(X, kmeans.labels\_))

# Elbow Method

plt.plot(K\_range, wcss, 'bx-')

plt.xlabel('Number of clusters K')

plt.ylabel('WCSS')

plt.title('Elbow Method')

plt.show()

# Silhouette Score

plt.plot(K\_range, sil\_scores, 'ro-')

plt.xlabel('Number of clusters K')

plt.ylabel('Silhouette Score')

plt.title('Silhouette Analysis')

plt.show()

**7. Giải quyết nhạy cảm với giá trị khởi tạo**

* **Vấn đề:** K-Means ngẫu nhiên chọn centroid → kết quả khác nhau.
* **Giải pháp:**
  + Sử dụng **K-Means++** (mặc định trong Scikit-learn) để chọn centroid thông minh.
  + Chạy **nhiều lần** (n\_init=10–50) và chọn kết quả tốt nhất (hàm mục tiêu nhỏ nhất).

**8. Đánh giá chất lượng các cụm**

* **WCSS / Inertia:** Tổng bình phương khoảng cách giữa điểm và centroid trong cùng cụm (nhỏ hơn = tốt).
* **Silhouette Score:** Đo mức độ tách biệt giữa các cụm, giá trị từ -1 đến 1 (gần 1 = tốt).
* **Davies-Bouldin Index (DBI):** Giá trị nhỏ hơn = cụm tốt hơn.

Ví dụ:

score = silhouette\_score(X, labels)

print(f"Silhouette Score: {score:.3f}")

**3.2.1. Ôn tập lý thuyết**

**1. Giải thuật phân cụm đa cấp hoạt động như thế nào?**

**Hierarchical Clustering (Phân cụm đa cấp)** là thuật toán phân nhóm dữ liệu theo cấu trúc cây, không cần xác định số cụm K trước.  
Có hai loại chính:

1. **Phân cụm hợp nhất (Agglomerative Clustering – bottom-up):**
   * Mỗi điểm dữ liệu bắt đầu là một cụm riêng lẻ.
   * Lặp đi lặp lại: gộp hai cụm gần nhau nhất dựa trên khoảng cách.
   * Tiếp tục cho đến khi chỉ còn 1 cụm duy nhất hoặc đạt điều kiện dừng.
2. **Phân cụm phân tách (Divisive Clustering – top-down):**
   * Bắt đầu với toàn bộ dữ liệu là một cụm lớn.
   * Chia cụm lớn thành các cụm con nhỏ hơn, dựa trên sự khác biệt lớn nhất giữa các điểm.
   * Tiếp tục tách đến khi mỗi điểm là một cụm riêng hoặc đạt điều kiện dừng.

**Khác biệt chính:**

* Agglomerative: **bottom-up**, gộp dần.
* Divisive: **top-down**, tách dần.

**2. Các phương pháp liên kết (linkage)**

Phương pháp liên kết xác định **khoảng cách giữa các cụm**:

| **Phương pháp** | **Công thức/Đặc điểm** | **Khi nên sử dụng** |
| --- | --- | --- |
| **Single linkage** | Khoảng cách nhỏ nhất giữa 2 điểm thuộc 2 cụm | Dữ liệu có hình dạng chuỗi, muốn phát hiện cụm dài và hẹp |
| **Complete linkage** | Khoảng cách lớn nhất giữa 2 điểm thuộc 2 cụm | Khi muốn cụm đồng đều, tránh cụm kéo dài |
| **Average linkage** | Trung bình khoảng cách giữa tất cả các điểm của 2 cụm | Cân bằng giữa single và complete linkage |
| **Ward’s method** | Giảm thiểu tổng bình phương khoảng cách trong cụm (giống K-Means) | Khi muốn cụm có dạng hình cầu, đồng đều, phổ biến trong phân tích số liệu |

**3. Dendrogram và cách sử dụng để chọn số cụm**

* **Dendrogram** là biểu đồ cây thể hiện thứ tự hợp nhất hoặc tách cụm.
* Trục **x:** các điểm dữ liệu; trục **y:** khoảng cách giữa các cụm khi gộp.
* **Chọn số cụm:** vẽ một đường ngang trên dendrogram; số cụm bằng số đường cắt dendrogram qua đường ngang đó.

**4. Áp dụng phân cụm đa cấp cho dữ liệu phi số (non-numeric data)**

* Dữ liệu phi số không thể tính khoảng cách Euclidean trực tiếp.
* Giải pháp: tính **ma trận khoảng cách (distance matrix)** bằng các phương pháp khác:
  + Khoảng cách Hamming cho dữ liệu nhị phân
  + Khoảng cách Jaccard cho tập hợp
  + Khoảng cách cosine cho dữ liệu văn bản
* Sau đó áp dụng Agglomerative Clustering dựa trên ma trận khoảng cách.

**5. Code mẫu Agglomerative Clustering bằng Python (Scikit-learn)**

# Import thư viện

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import AgglomerativeClustering

# Tạo dữ liệu mẫu

X, y = make\_blobs(n\_samples=200, centers=3, cluster\_std=0.6, random\_state=42)

# Khởi tạo Agglomerative Clustering

agg = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')

labels = agg.fit\_predict(X)

# Vẽ kết quả

plt.scatter(X[:,0], X[:,1], c=labels, cmap='rainbow', s=50)

plt.title('Agglomerative Clustering')

plt.show()

**Các bước thực hiện:**

1. Chuẩn bị dữ liệu.
2. Khởi tạo mô hình AgglomerativeClustering, chọn **linkage** và số cụm nếu muốn.
3. Áp dụng .fit\_predict() để phân cụm.
4. Vẽ kết quả trực quan.

**6. Vẽ dendrogram trong Python (sử dụng Scipy)**

from scipy.cluster.hierarchy import dendrogram, linkage

# Tính linkage

Z = linkage(X, method='ward') # hoặc 'single', 'complete', 'average'

# Vẽ dendrogram

plt.figure(figsize=(10, 6))

dendrogram(Z, truncate\_mode='lastp', p=30, leaf\_rotation=90., leaf\_font\_size=10.)

plt.title('Dendrogram')

plt.xlabel('Sample index')

plt.ylabel('Distance')

plt.show()

* truncate\_mode='lastp' chỉ hiển thị các cụm lớn nhất.
* Chọn **ngưỡng cắt (threshold)** trên trục y để xác định số cụm.

**7. Các lớp trong gói Scipy hỗ trợ phân cụm đa cấp**

* scipy.cluster.hierarchy.linkage(): tạo ma trận liên kết giữa các điểm.
* scipy.cluster.hierarchy.dendrogram(): vẽ dendrogram.
* scipy.cluster.hierarchy.fcluster(): cắt dendrogram để tạo các cụm cuối cùng.

**So sánh Scikit-learn vs Scipy:**

| **Đặc điểm** | **Scikit-learn** | **Scipy** |
| --- | --- | --- |
| Cách triển khai | Thẳng tiến, .fit() hoặc .fit\_predict() | Dựa trên ma trận khoảng cách và linkage |
| Vẽ dendrogram | Không trực tiếp, cần thư viện khác | Hỗ trợ trực tiếp với dendrogram() |
| Tính linh hoạt | Dễ sử dụng, tích hợp ML pipelines | Linh hoạt hơn, có thể dùng với dữ liệu phi số, tuỳ biến linkage |