Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin

Ngành KHOA HỌC MÁY TÍNH

Môn học: TOÁN CHO KHMT

SPECTRAL CLUSTERING

Học viên:
23521704 - Trần Thị Cẩm Tú
23520520 - Lê Chí Hoàng
23521193 - Đinh Hoàng Phúc
23520526 - Ngô Lê Nhật Hoàng

Giảng viên:
Dương Việt Hằng

Contents

1 Introduction

Lý do chọn đề tài (Clustering – Spectral Clustering)

- **Tính ứng dụng cao:** Clustering là công cụ quan trọng trong việc phân tích dữ liệu không nhãn, giúp giải quyết các bài toán thực tế như phân nhóm khách hàng, phát hiện bất thường, và phân đoạn hình ảnh.
- **Hạn chế của các phương pháp truyền thống:** Các thuật toán như K-means thường không hiệu quả với cụm dữ liệu phi tuyến tính hoặc phân bố không đồng đều, điều này thúc đẩy việc tìm kiếm các phương pháp tốt hơn.
- **Ưu điểm nổi bật của Spectral Clustering:** Nhờ sử dụng lý thuyết đồ thị, Spectral Clustering xử lý tốt các cụm phức tạp và mang lại kết quả vượt trội trong các bài toán khó.
- **Phù hợp với dữ liệu hiện đại:** Trong bối cảnh dữ liệu ngày càng phức tạp, Spectral Clustering là công cụ phù hợp để phân tích dữ liệu phi cấu trúc và các tập dữ liệu lớn.

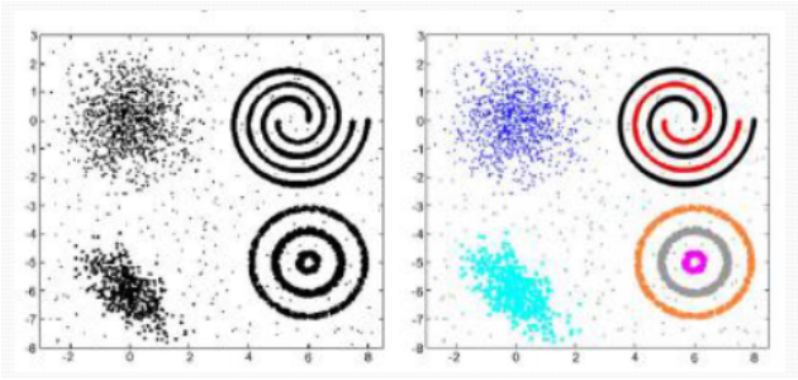
1.1 1. Clustering là gì?

Học không giám sát (Unsupervised Learning):

- Tập học (Training Data) bao gồm các quan sát, mà mỗi quan sát không có thông tin về *label* hoặc giá trị đầu ra mong muốn.
- Mục đích là tìm ra (học) các cụm, các cấu trúc, các quan hệ tồn tại ẩn trong tập dữ liệu hiện có.

Phân cụm/Phân nhóm (Clustering):

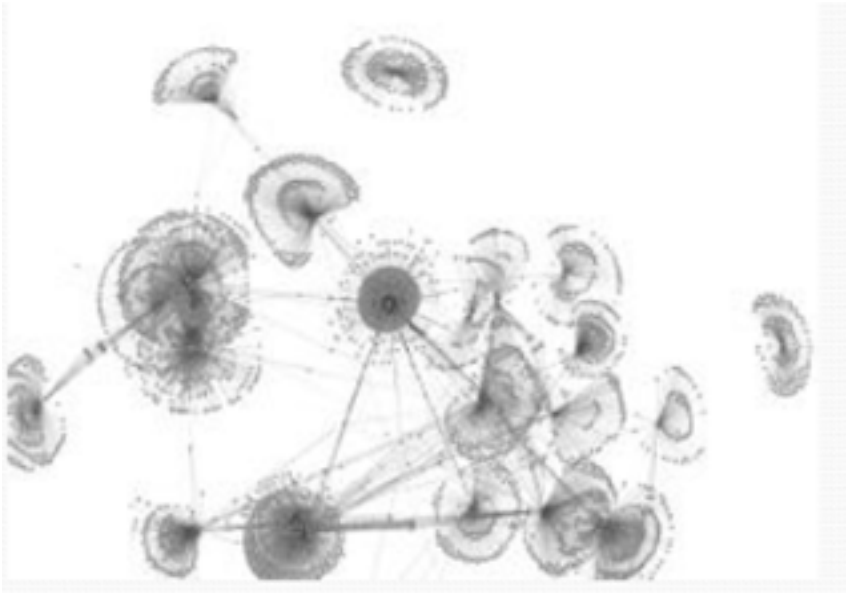
- Phát hiện các nhóm dữ liệu, nhóm tính chất.



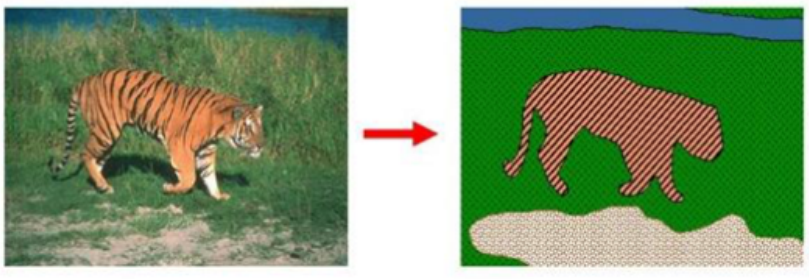
Ví dụ:

- Phân cụm ảnh.

- Community Detection (Phát hiện các cộng đồng trong xã hội).



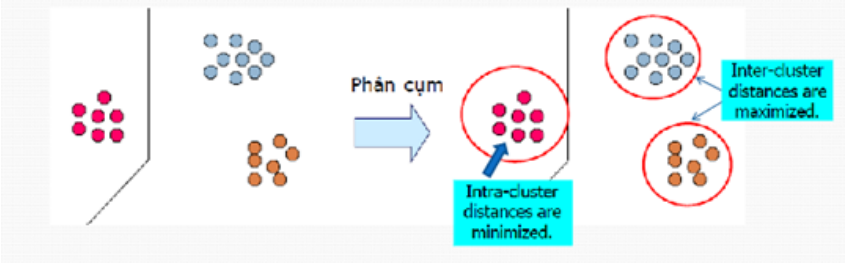
- Image Segmentation (Phân đoạn ảnh).



Clustering:

- Là quá trình phân nhóm/cụm dữ liệu/đối tượng vào các nhóm/cụm.

- Các đối tượng trong cùng một nhóm tương tự (tương đồng) với nhau hơn so với các đối tượng ở các nhóm khác.



Input:

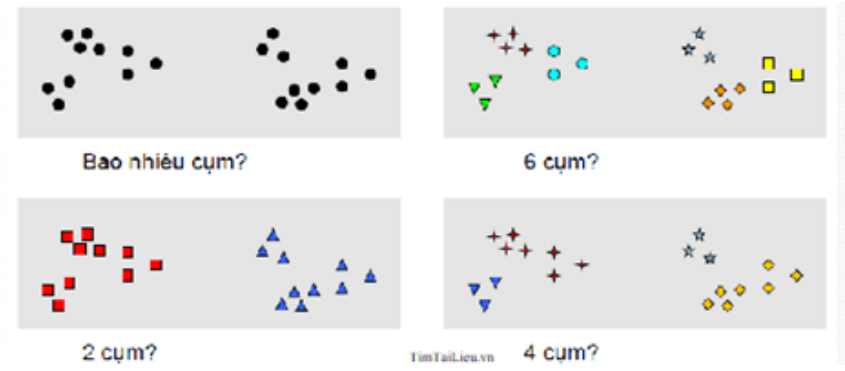
- Một tập dữ liệu $\{x_1, x_2, \ldots, x_N\}$ không có nhãn (hoặc giá trị đầu ra mong muốn).

Output:

- Các cụm (nhóm) của các quan sát.
- Một cụm (cluster) là một tập các quan sát:
  - Tương tự với nhau (theo một ý nghĩa, đánh giá nào đó).
  - Khác biệt với các quan sát thuộc các cụm khác.

Vấn đề đặt ra khi Clustering:

- Mỗi cụm/nhóm nên có bao nhiêu phần tử?
- Các phần tử nên được phân vào bao nhiêu cụm/nhóm?
- Bao nhiêu cụm/nhóm nên được tạo ra?

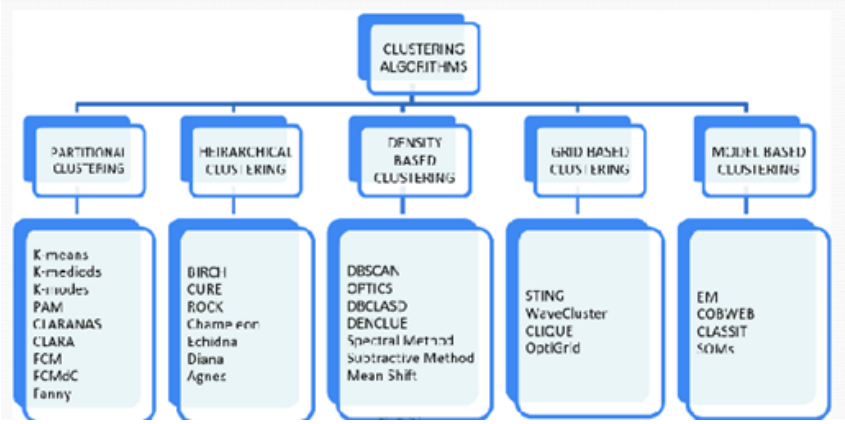


Các yêu cầu khi thiết kế thuật toán phân cụm dữ liệu:

- Có thể tương thích, hiệu quả với dữ liệu lớn, số chiều lớn.
- Có khả năng xử lý các dữ liệu khác nhau.
- Có khả năng khám phá các cụm với các dạng bất kỳ.
- Có khả năng thích nghi với dữ liệu nhiễu.
- Ít nhạy cảm với thứ tự của các dữ liệu vào.

1.2 2. Các phương pháp Clustering

- **Phân hoạch (Partitioning):** Phân hoạch tập dữ liệu n phần tử thành k cụm.
  - Ví dụ: K-Means, Fuzzy C-Means, ...
- **Phân cấp (Hierarchical):** Xây dựng các phân cấp các cụm trên cơ sở các đối tượng dữ liệu đang xem xét.
  - Ví dụ: AGNES (Agglomerative NESting), DIANA (Divisive ANAlysis), ...
- **Spectral Clustering:** Phân cụm dựa trên đồ thị.



Ví dụ:

- Phân hoạch (Partitioning).

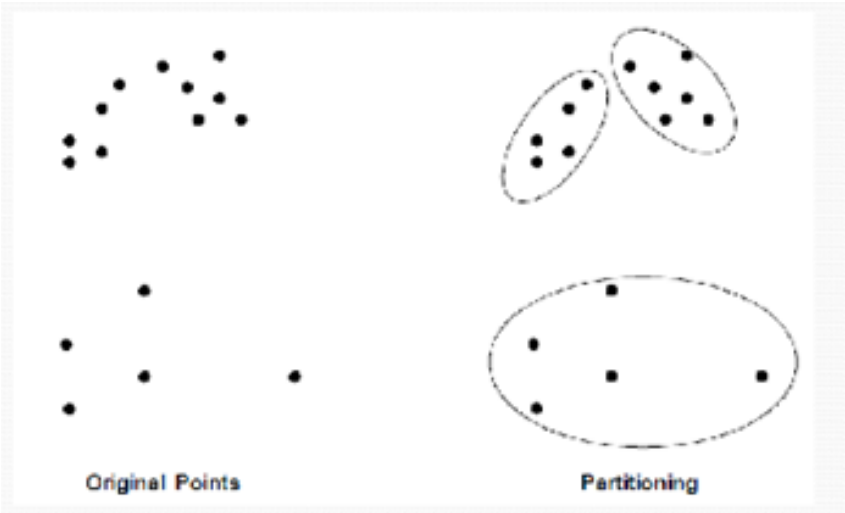


Figure 1: Ví dụ về phân hoạch (Partitioning)

- Phân cấp (Hierarchical).

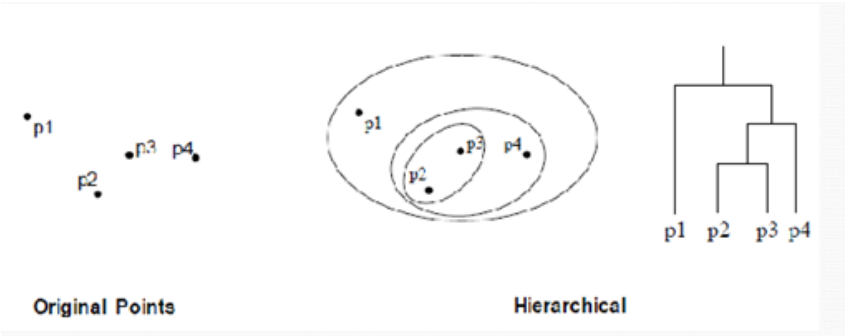


Figure 2: Ví dụ về phân cấp (Hierarchical)

- Spectral Clustering.

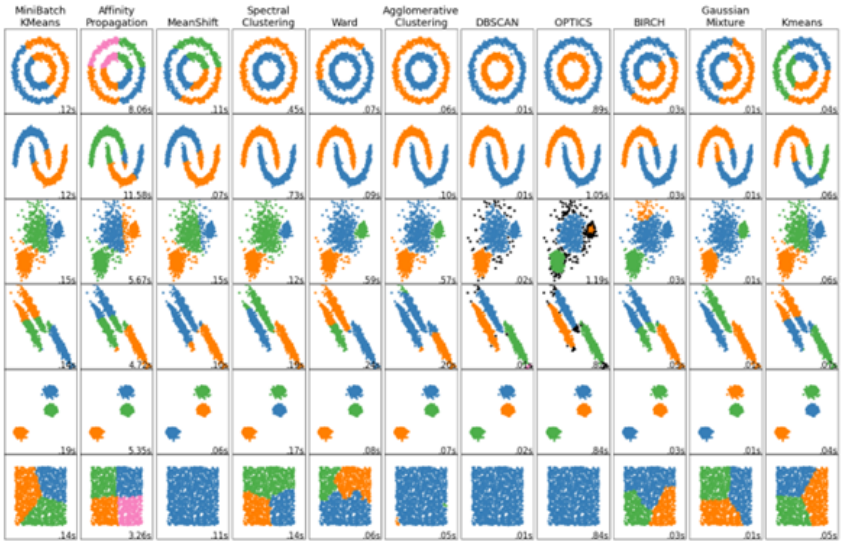


Figure 3: Các phương pháp Clustering

1.3 3. K-Means Clustering

Tóm tắt thuật toán:

- **Đầu vào:** Dữ liệu X và số lượng cluster cần tìm K.
- **Đầu ra:** Các center C và label vector cho từng điểm dữ liệu Y.

1. Chọn K điểm bất kỳ làm các center ban đầu.
2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.
3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước đó thì dừng thuật toán.
4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất cả các điểm dữ liệu đã được gán vào cluster đó sau bước 2.
5. Quay lại bước 2.

Chúng ta có thể đảm bảo rằng thuật toán sẽ dừng lại sau một số hữu hạn vòng lặp. Thật vậy:

- Hàm mất mát là một số dương và sau mỗi bước 2 hoặc 3, giá trị của hàm mất mát bị giảm đi.
- Theo kiến thức về dãy số trong chương trình cấp 3: nếu một dãy số giảm và bị chặn dưới thì nó hội tụ!

- Hơn nữa, số lượng cách phân nhóm cho toàn bộ dữ liệu là hữu hạn, nên đến một lúc nào đó, hàm mất mát sẽ không thể thay đổi, và thuật toán sẽ dừng lại.

Giả sử có N điểm dữ liệu $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in R^{d \times N}$ và $K < N$ là số cluster mà chúng ta muốn phân chia. Mục tiêu là tìm các center $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K \in R^{d \times 1}$ và label của mỗi điểm dữ liệu.

Với mỗi điểm dữ liệu $\mathbf{x}_i$, ta đặt $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iK}]$ là vector label của nó, trong đó nếu $\mathbf{x}_i$ thuộc về cluster k, thì $y_{ik} = 1$ và $y_{ij} = 0$, với $\forall j \neq k$. Đây là biểu diễn one-hot, trong đó chỉ có một phần tử bằng 1, tương ứng với cluster mà điểm dữ liệu thuộc về.

Ràng buộc của $\mathbf{y}_i$ được viết dưới dạng toán học như sau:

$$y_{ik} \in \{0, 1\}, \quad \sum_{k=1}^{K} y_{ik} = 1 \quad \forall i$$

Hàm mất mát và bài toán tối ưu

Để mô phỏng cách mỗi điểm dữ liệu được phân vào cluster có center gần nhất, ta tính sai số giữa điểm dữ liệu và center của cluster. Mục tiêu là giảm thiểu sai số này.

Sai số cho toàn bộ dữ liệu là:

$$L(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Trong đó $\mathbf{Y} = [\mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_N]$ và $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K]$ lần lượt là ma trận label vector và các center của các cluster.

Bài toán tối ưu là:

$$\mathbf{Y}, \mathbf{M} = \arg\min_{\mathbf{Y}, \mathbf{M}} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

với ràng buộc:

$$y_{ij} \in \{0, 1\}, \quad \sum_{j=1}^{K} y_{ij} = 1 \quad \forall i, j$$

Thuật toán tối ưu

Để giải quyết bài toán trên, ta có thể sử dụng phương pháp lặp, trong đó mỗi lần ta cố định một biến (các center $\mathbf{m}_j$ hoặc các label $\mathbf{y}_i$) và tìm tối ưu cho biến còn lại.

Cố định M, tìm Y

Khi các center đã cố định, tìm label vector $\mathbf{y}_i$ cho mỗi điểm dữ liệu bằng cách chọn cluster k sao cho $\|\mathbf{x}_i - \mathbf{m}_k\|_2^2$ là nhỏ nhất.

Cố định Y, tìm M

Khi label vector đã cố định, các center $\mathbf{m}_j$ được tính bằng trung bình cộng của các điểm dữ liệu trong mỗi cluster:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

Điều này có nghĩa là mỗi center là trung tâm của các điểm dữ liệu trong cluster tương ứng.

Ví dụ đơn giản về K-Means Clustering

Bắt đầu bằng một ví dụ đơn giản: trong một lớp có 50 học sinh, chúng ta cần chia thành 3 nhóm dựa trên chiều cao của các cá nhân trong lớp. Có thể tạm gọi là *"Cao"*, *"Trung Bình"*, và *"Thấp"*. Việc dán nhãn như trên chỉ mang tính minh họa và hoàn toàn không ảnh hưởng đến kết quả của bài toán.

Như vậy, chúng ta có một tập dữ liệu $\{x_i\}$, $(i = 1, \ldots, 50)$ là 50 giá trị chiều cao của các thành viên trong lớp. Vì muốn chia lớp thành 3 nhóm, nên chúng ta có $\{c_j\}$, $(j = 1, \ldots, 3)$ là kí hiệu 3 điểm trung

tâm của mỗi nhóm. c_j là một giá trị chiều cao (mét), và những cá nhân trong lớp có chiều cao gần với c_j sẽ được xếp vào nhóm j.

Tập dữ liệu $\{x_i\}$ cũng có thể chứa thêm nhiều tham số khác, như ngày tháng năm sinh, cân nặng, ...

Hàm khoảng cách: Để định nghĩa khoảng cách về chiều cao giữa một thành viên bất kì x_i đến điểm trung tâm của nhóm c_j, ta sử dụng công thức:

$$d_j(x_i, c_j) = \|x_i - c_j\|_2^2$$

trong đó:

- $d_j(x_i, c_j)$: khoảng cách (ký hiệu chữ cái đầu trong từ "distance") giữa hai điểm x_i và c_j.
- $\|\dots\|_2^2$: bình phương của hàm L2-norm (còn được gọi là vector norm).

Giải thích hàm L2-norm: Công thức toán học của L2-norm như sau:

$$\|x_i - c_j\|_2 = \sqrt{\sum_{k=1}^{n}(x_k - c_k)^2}$$

với $x_i = \langle x_1, x_2, \dots, x_k, \dots, x_n \rangle$ và $c_j = \langle c_1, c_2, \dots, c_k, \dots, c_n \rangle$.

Ví dụ: Giả sử trong ví dụ này, c_j và x_i chỉ có một tham số duy nhất. Thành viên thứ 34 trong lớp có chiều cao $x_{34} = 1.32\,\text{m}$, và nhóm $j = 2$ có giá trị trung tâm là $c_2 = 1.30\,\text{m}$. Ta tính giá trị hàm khoảng cách:

$$d_j(x_i, c_j) = \left(\sqrt{(1.32 - 1.30)^2}\right)^2 = (0.02)^2 = 0.004$$

Hàm tổng khoảng cách: Với 3 giá trị trung tâm tương ứng với 3 nhóm ($j = 1, \dots, 3$), hàm tổng khoảng cách từ một điểm x_i đến tất cả các trung tâm c_j được định nghĩa như sau:

$$\sum_{j=1}^{k} d(x_i, c_j) = \sum_{j=1}^{k} \|x_i - c_j\|_2^2$$

Đến đây, chúng ta đã tính được khoảng cách từ một điểm x_i. Tuy nhiên, vì có tất cả 50 thành viên trong lớp, hàm khoảng cách cần được mở rộng để tính tổng khoảng cách giữa mọi điểm dữ liệu và các trung tâm cụm. Hàm khoảng cách tổng quát được định nghĩa như sau:

$$\sum_{i=1}^{m}\sum_{j=1}^{k} d(x_i, c_j) = \sum_{i=1}^{m}\sum_{j=1}^{k} y_{ij}\|x_i - c_j\|_2^2$$

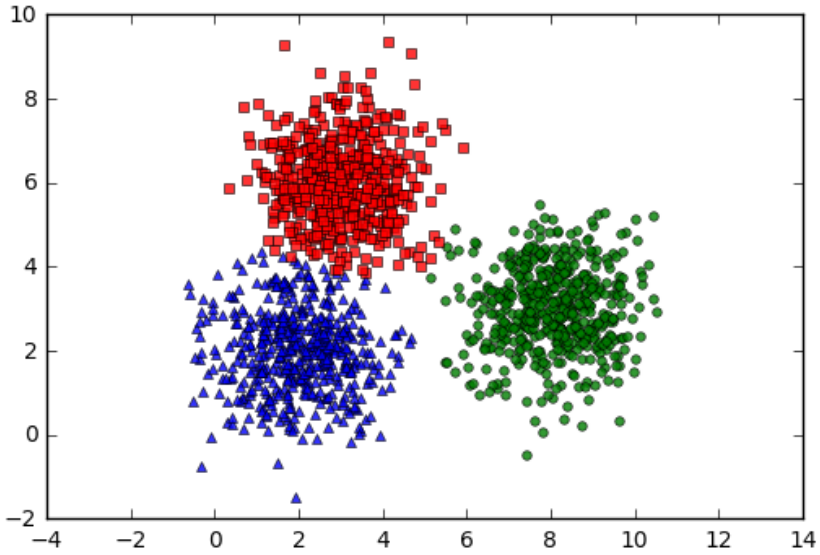
Đến đây, chúng ta đã có một hàm có thể đánh giá mức độ phù hợp của các trung tâm c_j. Càng chọn c_j sao cho giá trị hàm trên càng nhỏ, thuật toán càng hoạt động hiệu quả. Khi giá trị của x_i được cố định, **phương pháp chọn c_j chính là trung tâm của thuật toán K-Means Clustering**.

Chúng ta có thể đặt:

$$J = \sum_{i=1}^{m}\sum_{j=1}^{k} d(x_i, c_j) = \sum_{i=1}^{m}\sum_{j=1}^{k} y_{ij}\|x_i - c_j\|_2^2$$

Thuật toán K-Means Clustering đi tìm giá trị cực tiểu của J:

$$J_{\min} = \underset{c}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{j=1}^{k} d(x_i, c_j)$$



article amsmath

1.4 4. Khái niệm cơ bản:

Đồ thị (Graph): Đỉnh, cạnh, trọng số. Đồ thị được sử dụng để biểu diễn mối quan hệ giữa các điểm dữ liệu dưới dạng các đỉnh (nodes) và các cạnh (edges). Mỗi cạnh có thể có một trọng số (weight), thể hiện mức độ mạnh yếu của sự kết nối giữa các điểm. Sau đây là các khái niệm quan trọng liên quan đến đồ thị trong Spectral Clustering:

1.4.1 1. Đỉnh (Vertex/Node):

- Đỉnh (hoặc nút) là một thành phần cơ bản của đồ thị, đại diện cho một đối tượng hoặc một điểm dữ liệu trong không gian.
- Trong Spectral Clustering, mỗi đỉnh thường đại diện cho một điểm dữ liệu trong tập dữ liệu.
- **Ví dụ:** Nếu bạn có 100 điểm dữ liệu trong không gian 2D, thì bạn sẽ có 100 đỉnh trong đồ thị, mỗi đỉnh đại diện cho một điểm dữ liệu.

1.4.2 2. Cạnh (Edge):

- Cạnh (hoặc đường nối) là các kết nối giữa các đỉnh trong đồ thị, thể hiện mối quan hệ hoặc sự tương đồng giữa hai điểm dữ liệu.
- Trong Spectral Clustering, các cạnh giữa các đỉnh thường được xác định dựa trên sự tương đồng hoặc khoảng cách giữa các điểm dữ liệu.
- Nếu hai điểm dữ liệu P_i và P_j có sự tương đồng lớn (ví dụ, nếu chúng gần nhau trong không gian), thì có thể sẽ có một cạnh nối giữa chúng.
- Cạnh có thể có trọng số, tức là độ mạnh yếu của mối quan hệ giữa các điểm được biểu diễn bằng giá trị trọng số này.

1.4.3 3. Trọng Số Cạnh (Edge Weight):

- Trọng số của cạnh đại diện cho mức độ mạnh yếu của mối quan hệ giữa hai đỉnh. Trọng số này có thể được xác định dựa trên nhiều yếu tố, nhưng phổ biến nhất là sự tương đồng giữa các điểm dữ liệu.
- Sự tương đồng giữa hai điểm P_i và P_j có thể được tính bằng một số hàm hạt nhân, như hàm hạt nhân Gaussian:

$$w_{ij} = \exp\left(-\frac{\|P_i - P_j\|^2}{2\sigma^2}\right)$$

- Trọng số cao có nghĩa là các điểm dữ liệu gần nhau hoặc có sự tương đồng lớn.
- Trọng số thấp có nghĩa là các điểm dữ liệu xa nhau hoặc có sự tương đồng nhỏ.

1.5 5. Ma trận kề (Adjacency Matrix) và Ma trận tương đồng:

1.5.1 1. Ma Trận Kề (Adjacency Matrix):

Ma trận kề là một ma trận vuông mô tả mối quan hệ giữa các đỉnh trong đồ thị. Nếu đồ thị là vô hướng (undirected), ma trận kề là đối xứng. Các phần tử trong ma trận kề thể hiện mức độ kết nối (liên kết) giữa các đỉnh của đồ thị.

Định nghĩa: Cho đồ thị $G = (V, E)$ với V là tập gồm các đỉnh và E là tập gồm các cạnh. Mỗi cạnh thuộc E sẽ gồm 2 đỉnh trong tập V - một cặp đỉnh (v_i, v_j), với trọng số cạnh là w_{ij}. Ma trận kề A được định nghĩa như sau:

$$A_{ij} = \begin{cases} w_{ij}, & \text{nếu tồn tại cạnh nối giữa } v_i \text{ và } v_j, \\ 0, & \text{nếu không tồn tại cạnh nối giữa } v_i \text{ và } v_j. \end{cases}$$

Ví dụ: Nếu đồ thị có 3 đỉnh và các cạnh với trọng số tương ứng như sau:

- Cạnh giữa v_1 và v_2 có trọng số 3.
- Cạnh giữa v_2 và v_3 có trọng số 5.
- Không có cạnh nối giữa v_1 và v_3.

Ma trận kề sẽ là:

$$A = \begin{bmatrix} 0 & 3 & 0 \\ 3 & 0 & 5 \\ 0 & 5 & 0 \end{bmatrix}$$

1.5.2 2. Đồ Thị Tương Đồng (Similarity Graph):

Similarity Graphs trong Clustering: Trong học máy và đặc biệt là trong phân cụm (clustering), similarity graph (đồ thị tương đồng) là một cách mạnh mẽ để mô hình hóa mối quan hệ giữa các điểm dữ liệu. Mỗi điểm dữ liệu được biểu diễn dưới dạng một đỉnh (vertex) trong đồ thị, và mối quan hệ tương đồng giữa các điểm được biểu diễn dưới dạng các cạnh (edges) nối các đỉnh này. Mục tiêu của phân cụm là phân chia các điểm dữ liệu thành các nhóm sao cho các điểm trong cùng một nhóm có sự tương đồng cao, trong khi các điểm ở các nhóm khác nhau có sự tương đồng thấp.

Cấu Trúc của Similarity Graph: Đồ thị tương đồng có thể được mô hình hóa dưới dạng đồ thị có trọng số (weighted graph), trong đó mỗi đỉnh v_i đại diện cho một điểm dữ liệu x_i. Hai đỉnh v_i và v_j sẽ được nối với nhau nếu mức độ tương đồng s_{ij} giữa các điểm dữ liệu x_i và x_j là dương (hoặc vượt qua một ngưỡng nhất định). Trọng số của cạnh này là s_{ij}, thể hiện mức độ tương đồng giữa hai điểm dữ liệu.

Graph Notation:

- **Đỉnh (Vertex):** Mỗi đỉnh trong đồ thị đại diện cho một điểm dữ liệu.
- **Cạnh (Edge):** Hai điểm dữ liệu x_i và x_j sẽ được nối với nhau bằng một cạnh nếu sự tương đồng giữa chúng là dương.
- **Trọng Số Cạnh (Edge Weight):** Trọng số của mỗi cạnh s_{ij} thể hiện mức độ tương đồng giữa các điểm dữ liệu x_i và x_j.

Các Loại Similarity Graphs Phổ Biến:

- **ε-neighborhood graph:**
  - Đây là loại đồ thị trong đó các điểm dữ liệu x_i và x_j sẽ được nối với nhau nếu khoảng cách giữa chúng nhỏ hơn một giá trị ngưỡng ε.
  - Điều này có nghĩa là chúng chỉ kết nối các điểm có khoảng cách nhỏ hơn ε.

- **k-nearest neighbor graph (k-NN graph):**
  - Trong loại đồ thị này, mỗi điểm dữ liệu x_i được kết nối với k điểm dữ liệu gần nhất của nó.
  - Điều này có thể tạo ra một đồ thị có hướng, vì mối quan hệ k-nearest có thể không đối xứng.
  - Trọng số của các cạnh là mức độ tương đồng giữa các điểm dữ liệu.
- **Fully connected graph:**
  - Ở đây, mọi điểm dữ liệu đều được kết nối với nhau nếu mức độ tương đồng giữa chúng là dương.
  - Trọng số của các cạnh sẽ bằng mức độ tương đồng s_{ij} giữa các điểm.
  - Một ví dụ điển hình cho loại đồ thị này là đồ thị sử dụng Gaussian similarity function:

$$s_{ij} = \exp\left(-\frac{d(P_i, P_j)^2}{2\sigma^2}\right)$$

  - Trong đó:
    - P_i và P_j: Các điểm dữ liệu.
    - $d(P_i, P_j)$: Khoảng cách giữa hai điểm dữ liệu (thường là khoảng cách Euclid).
    - σ: Tham số điều chỉnh độ rộng của Gaussian kernel.

Ma Trận Tương Đồng và Ý Nghĩa của Đồ Thị Trong Mô Hình Hóa Dữ Liệu

Ma Trận Tương Đồng:

Ma trận tương đồng là một ma trận vuông, đối xứng với các phần tử có giá trị từ 0 đến 1. Các giá trị trong ma trận tương đồng được định nghĩa như sau:

- Giá trị $s_{ij} > 0$ biểu thị mức độ tương đồng giữa hai điểm dữ liệu x_i và x_j.
- Giá trị $s_{ij} = 0$ nếu hai điểm dữ liệu không có sự tương đồng (hoặc không được nối bởi một cạnh).
- Giá trị $s_{ij} \to 1$ biểu thị mức độ tương đồng rất cao.

5. Ý Nghĩa của Việc Dùng Đồ Thị Để Mô Hình Hóa Dữ Liệu:

1. Biểu Diễn Mối Quan Hệ Giữa Các Đối Tượng:

Đồ thị cho phép mô hình hóa các mối quan hệ giữa các điểm dữ liệu dưới dạng các đỉnh và các cạnh, giúp hiểu rõ hơn về cấu trúc và quan hệ giữa các đối tượng trong không gian dữ liệu.

2. Mô Hình Hóa Các Dữ Liệu Liên Kết:

Trong nhiều trường hợp, dữ liệu không phải là độc lập mà có sự liên kết hoặc ảnh hưởng qua lại giữa các phần tử. Đồ thị cung cấp một cách tốt để mô hình hóa và khai thác thông tin từ các dữ liệu liên kết này.

- **Ví dụ:**
  - Trong mạng xã hội, các đỉnh là người dùng, và các cạnh là mối quan hệ bạn bè giữa họ.
  - Trong mạng giao thông, các đỉnh là các thành phố, và các cạnh là các tuyến đường nối chúng.

3. Chia Nhóm và Phân Cụm Dữ Liệu:

Đồ thị hỗ trợ phân cụm dữ liệu (clustering) bằng cách nhóm các điểm dữ liệu có mối quan hệ mạnh mẽ với nhau. Các thuật toán như **Spectral Clustering** sử dụng đồ thị để xác định các nhóm dữ liệu.

4. Mối Quan Hệ Phi Tuyến:

Đồ thị có thể mô hình hóa các mối quan hệ phi tuyến giữa các đối tượng trong dữ liệu, điều mà các mô hình tuyến tính không thể làm được.

- **Ví dụ:** Gaussian kernel sử dụng đồ thị và ma trận tương đồng để tính toán sự tương đồng trong không gian cao chiều, giúp mô hình hóa các quan hệ phi tuyến.

5. Tính Toán Dễ Dàng Các Mối Quan Hệ Dưới Dạng Ma Trận:

Việc chuyển đổi đồ thị thành các ma trận (như ma trận kề, ma trận tương đồng hoặc ma trận Laplacian) giúp xử lý và tính toán dữ liệu hiệu quả hơn.

- **Ví dụ:**
  - Các ma trận này được sử dụng trong các thuật toán như Spectral Clustering, học sâu (deep learning) và các bài toán phân loại.
  - Chúng giúp tính toán các đặc trưng của đồ thị, như giá trị riêng (Eigenvalues) và vector riêng (Eigenvectors), để phân nhóm hoặc phân loại dữ liệu.

6. Giảm Chiều Dữ Liệu:

Đồ thị hỗ trợ giảm chiều dữ liệu (*dimensionality reduction*) bằng cách trích xuất các đặc trưng quan trọng từ dữ liệu có chiều cao, giúp tối ưu hóa hiệu suất tính toán và học máy.

2 Phương pháp Spectral Clustering

Trong bài thuyết trình ngày hôm nay, chúng em sẽ giới thiệu về các bước tiến hành phương pháp *Spectral Clustering*. Dựa trên tài liệu tham khảo từ bài báo *"A Tutorial on Spectral Clustering"* của Ulrike von Luxburg, có hai cách chính để thực hiện phương pháp này: *Normalized Spectral Clustering* và *Unnormalized Spectral Clustering*.

Nhóm chúng em đã lựa chọn áp dụng phương pháp *Normalized Spectral Clustering* dựa theo nghiên cứu của Ng, Jordan, và Weiss (2002). Phương pháp này mang lại nhiều ưu điểm mà chúng em sẽ giải thích chi tiết trong phần sau. Quy trình thực hiện được chia thành các bước cụ thể, đảm bảo tính hệ thống và hiệu quả khi triển khai.

2.1 1. Các bước thực hiện Normalized Spectral Clustering

Phương pháp *Normalized Spectral Clustering* được thực hiện thông qua các bước sau đây:

1. **Tính ma trận độ tương đồng** (*Similarity Matrix*): Xây dựng ma trận độ tương đồng $W \in R^{n \times n}$, trong đó S_{ij} thể hiện độ tương đồng giữa hai điểm dữ liệu x_i và x_j. Ví dụ, sử dụng hàm Gaussian:

$$S_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

   với σ là tham số.

2. **Tính ma trận Laplacian chuẩn hóa** (*Normalized Laplacian*): - Tính ma trận đường chéo D, trong đó $D_{ii} = \sum_{j=1}^{n} S_{ij}$. - Tính ma trận Laplacian chuẩn hóa:

$$L = D^{-1/2} W D^{-1/2}$$

3. **Tính các vector riêng** (*Eigenvectors*): Tìm k vector riêng tương ứng với k giá trị riêng lớn nhất của ma trận L. Gom các vector riêng này thành một ma trận $U \in R^{n \times k}$, trong đó mỗi cột của U là một vector riêng.

4. **Chuẩn hóa hàng của ma trận** U: Biến mỗi hàng của U thành một vector đơn vị, cụ thể:

$$U_{ij} \leftarrow \frac{U_{ij}}{\sqrt{\sum_{j=1}^{k} U_{ij}^2}}$$

5. **Áp dụng thuật toán K-means**: Sử dụng thuật toán K-means để phân cụm các hàng của ma trận chuẩn hóa U thành k cụm.

6. **Gán nhãn cụm**: Gán nhãn dữ liệu ban đầu dựa trên kết quả phân cụm.

2.2 2. Mô tả tập dữ liệu

Trong nghiên cứu này, chúng em sử dụng tập dữ liệu Iris, một tập dữ liệu phổ biến trong lĩnh vực học máy và phân cụm. Tập dữ liệu Iris có những đặc điểm nổi bật, đáp ứng tốt các tiêu chí để minh họa thuật toán Spectral Clustering cũng như so sánh với các thuật toán phân cụm khác:

- **Kích thước nhỏ gọn**: Tập dữ liệu gồm 150 mẫu, giúp dễ dàng xử lý và trực quan hóa.
- **Tính phi tuyến tính**: Các cụm trong tập dữ liệu không hoàn toàn tuyến tính, điều này tạo thách thức đáng kể cho các thuật toán như K-means vốn phù hợp hơn với dữ liệu tuyến tính.
- **Có các cụm tự nhiên nhưng không hoàn toàn tách biệt**: Tính chất này giúp minh họa ưu điểm của Spectral Clustering trong việc xử lý các cụm phức tạp và phi tuyến.

Tập dữ liệu Iris bao gồm 150 mẫu hoa thuộc ba loài Iris khác nhau: *Iris setosa*, *Iris versicolor*, và *Iris virginica*. Mỗi mẫu được mô tả bởi bốn đặc tính:

- **Độ dài của cánh hoa** (sepal length) - Đo bằng cm.
- **Độ rộng của cánh hoa** (sepal width) - Đo bằng cm.
- **Độ dài của cánh** (petal length) - Đo bằng cm.
- **Độ rộng của cánh** (petal width) - Đo bằng cm.

Các đặc tính này được sử dụng để phân loại các mẫu hoa vào ba loài khác nhau. Mục tiêu là xây dựng mô hình phân loại có thể dự đoán loài của một mẫu hoa dựa trên các đặc tính này.

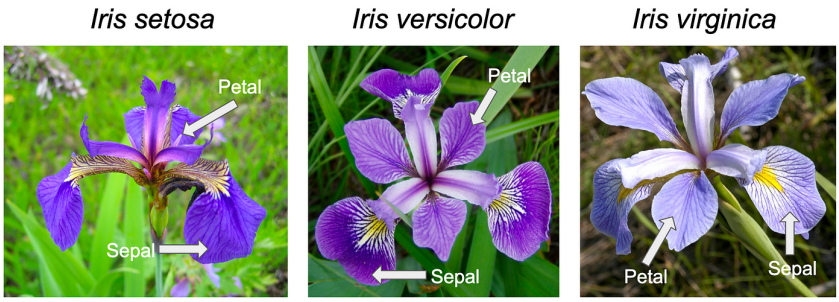


Figure 4: Minh họa tập dữ liệu Iris.

3 Tiến hành thuật toán Spectral Clustering

3.1 Bước 1: Tính ma trận độ tương đồng

Trong bước này, chúng em lựa chọn phương pháp **Fully Connected Graph** (đồ thị đầy đủ) để xây dựng ma trận độ tương đồng. Lý do lựa chọn như sau:

- Các phương pháp đồ thị khác:
  - **k-Nearest Neighbor Graph**: Yêu cầu chọn tham số k, điều này có thể khó tối ưu và ảnh hưởng đến kết quả phân cụm.
  - **ε-Neighborhood Graph**: Yêu cầu chọn tham số ε, nhưng giá trị này có thể khó xác định nếu dữ liệu có độ chênh lệch lớn.
- Với đồ thị đầy đủ (**Fully Connected Graph**):

- Không cần chọn tham số k hay ε, giảm thiểu sự phụ thuộc vào tham số.
- Phù hợp với các tập dữ liệu nhỏ, không thưa thớt như tập Iris, giúp tránh chi phí tính toán lớn.

Xây dựng ma trận độ tương đồng Với phương pháp Fully Connected Graph, chúng em sử dụng hàm độ tương đồng Gaussian để tính toán ma trận độ tương đồng S dựa trên công thức:

$$S_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Trong đó:

- S_{ij} là độ tương đồng giữa điểm dữ liệu x_i và x_j.
- σ là tham số điều chỉnh (bandwidth) cho hàm Gaussian.

Với tập dữ liệu Iris (150 mẫu), việc sử dụng Fully Connected Graph kết hợp với hàm độ tương đồng Gaussian là lựa chọn hợp lý, đảm bảo tính minh bạch và hiệu quả trong minh họa thuật toán.
Giải thích:

- $\|x_i - x_j\|^2$: Khoảng cách Euclidean giữa x_i và x_j.
- σ: Tham số kiểm soát mức độ nhạy cảm với khoảng cách.

Tính chất:

- Giá trị $s(x_i, x_j)$ nằm trong khoảng $(0, 1]$, càng gần thì giá trị càng lớn.
- Khi $\|x_i - x_j\|^2 \to 0$, thì $s(x_i, x_j) \to 1$.
- Khi $\|x_i - x_j\|^2 \to \infty$, thì $s(x_i, x_j) \to 0$.

Trong Spectral Clustering:

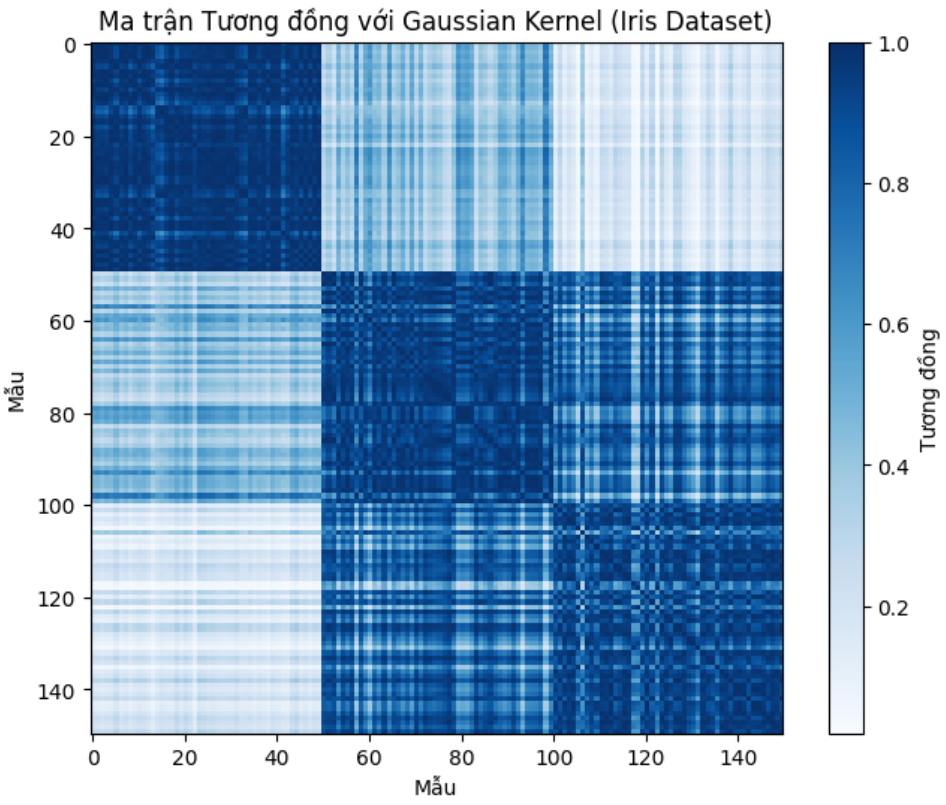
- σ nhỏ:
  - Đồ thị tương đồng W chỉ kết nối các điểm thực sự gần nhau.
  - Ma trận Laplacian phản ánh rõ cụm nhỏ, nhưng có thể bỏ sót các cụm lớn hơn.
- σ lớn:
  - Đồ thị tương đồng W trở nên quá mượt, nhiều điểm không liên quan vẫn được coi là tương đồng.
  - Ma trận Laplacian có thể không tách biệt cụm rõ ràng.

Yêu cầu đặt ra là tìm một giá trị σ phù hợp cho mô hình của chúng ta. Như đã phân tích ở trên, nếu chọn σ quá nhỏ hay quá lớn đều có thể dẫn đến một mô hình không thực sự dự đoán tốt hoàn toàn. Do đó, cần tìm giá trị σ trung bình, một giá trị mà mô hình có thể hạn chế được nhược điểm trên.

Ma trận khoảng cách K chứa tất cả các thông tin về khoảng cách giữa các điểm dữ liệu. Giá trị trung bình của ma trận khoảng cách K cung cấp một đặc trưng toàn cục của dữ liệu, đại diện cho "mức độ khoảng cách trung bình" giữa các điểm trong không gian. Việc sử dụng giá trị trung bình giúp hàm Gaussian tương thích với quy mô và phân bố tổng thể của dữ liệu.

$$\sigma = \frac{1}{n(n-1)} \sum_{i \neq j} K_{ij}$$

Từ tính toán, giá trị σ thu được là 2.527677189277037.
Dưới đây là biểu đồ ma trận tương đồng đã được trực quan hóa bằng thư viện `matplotlib`:



3.2 Bước 2: Tính toán và chuẩn hóa ma trận Laplacian

1. Ma trận tương đồng (Similarity Matrix)

Từ Bước 1, ta đã tính toán ma trận tương đồng W cho các điểm dữ liệu. Ma trận này đại diện cho mức độ tương đồng giữa các điểm dữ liệu. Một phần tử w_{ij} trong ma trận W biểu thị mức độ tương đồng giữa điểm dữ liệu i và điểm dữ liệu j, thường được tính dựa trên các đặc trưng như khoảng cách Euclid hoặc hàm Gaussian.

2. Ma trận bậc (Degree Matrix) - D

Ma trận bậc D là một ma trận đường chéo, trong đó mỗi phần tử d_i trên đường chéo là tổng trọng số của các cạnh kết nối với đỉnh i. Cụ thể, phần tử d_i được tính bằng tổng tất cả các phần tử trong hàng thứ i của ma trận tương đồng W, tức là:

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Trong đó, w_{ij} là trọng số giữa các điểm dữ liệu i và j, và n là số lượng điểm dữ liệu.

3. Ma trận Laplacian không chuẩn hóa (Unnormalized Laplacian) - L

Ma trận Laplacian không chuẩn hóa được tính bằng hiệu giữa ma trận bậc D và ma trận kề W:

$$L = D - W$$

Ma trận L này thể hiện sự kết nối giữa các điểm trong đồ thị. Các phần tử của L thể hiện mức độ kết nối giữa các điểm dữ liệu.

Phần 1

Chứng minh: Với mọi vector $f \in R^n$, ta có:

$$f^T Lf = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

Định nghĩa $d_i = \sum_j w_{ij}$, ta có:

$$f^T Lf = f^T Df - f^T Wf$$

Khai triển từng phần ta có:

$$f^T Df = \sum_{i=1}^{n} d_i f_i^2$$

và

$$f^T Wf = \sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j w_{ij}$$

Vậy ta có:

$$f^T Lf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j w_{ij}$$

Nhân $\frac{1}{2}$ ở ngoài và 2 ở trong ta có:

$$f^T Lf = \frac{1}{2}(2\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j w_{ij})$$

Chuyển chỉ số từ i thành j:

$$f^T Lf = \frac{1}{2}(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2)$$

Ta thấy d_i độc lập nhau nên ta tách 1 nửa và đổi thành d_j:

$$f^T Lf = \frac{1}{2}(\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f_i^2 - 2\sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ji} f_j^2)$$

Do ma trận W có tính đối xứng ($w_{ij} = w_{ji}$), ta có:

$$f^T Lf = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

Phần 2

Chứng minh: L là đối xứng và bán xác định dương .

Ma trận L đối xứng có thể suy ra từ tính đối xứng của W và D ($W = W^T$, $D = D^T$). Tính bán xác định dương của L là hệ quả trực tiếp từ phần 1, do $f^T Lf \geq 0$ với mọi $f \in R^n$.

Phần 3

Chứng minh: Giá trị riêng nhỏ nhất của L là 0, và một vector thuộc không gian riêng là vector có tất cả các phần tử bằng 1

Giả sử $u \in R^n$ là một vector riêng của L với giá trị riêng bằng 0, ta có:

$$L \cdot u = 0 \Rightarrow (D - W) \cdot u = 0$$

Giải hệ phương trình này:

$$\begin{bmatrix} d_1 - w_{11} & -w_{21} & \cdots & -w_{n1} \\ -w_{12} & d_2 - w_{22} & \cdots & -w_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ -w_{1n} & -w_{2n} & \cdots & d_n - w_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Xét hàng đầu tiên:

$$(d_1 - w_{11})u_1 - w_{21}u_2 - \cdots - w_{n1}u_n = 0 \Rightarrow d_1u_1 = u_1w_{11} + u_2w_{21} + \cdots + u_nw_{n1}$$

Cho tất cả $u = 1$, ta có:

$$d_i = \sum_{j=1}^{n} w_{ji}$$

Vì L là bán xác định dương nên giá trị riêng nhỏ nhất của L là 0, và một vector riêng thuộc không gian riêng tương ứng là vector có tất cả các phần tử bằng 1.

Phần 4

Chứng minh: L có n giá trị riêng không âm và là số thực.

Theo hệ quả phần 2, vì L là bán xác định dương, tất cả các giá trị riêng đều lớn hơn hoặc bằng 0, điều này khiến L chỉ có các giá trị riêng không âm và là số thực.

Phần 5

Chứng minh: Cho G là đồ thị vô hướng với trọng số không âm. Khi đó, số lượng giá trị riêng bằng 0 của L bằng số thành phần liên thông $A_1, A_2, \ldots, A_k$ trong đồ thị

Xét trường hợp đồ thị có một thành phần liên thông duy nhất. Giả sử f là một vector riêng với giá trị riêng bằng 0. Ta có:

$$0 = f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

Vì các trọng số w_{ij} không âm, tổng này chỉ có thể bằng 0 nếu tổng $w_{ij}(f_i - f_j)^2$ bằng 0. Do đó, nếu hai đỉnh nối với nhau ($w_{ij} > 0$), thì f_i và f_j phải bằng nhau. Từ đó suy ra f cần phải bằng nhau với mỗi thành phần liên thông.

Giờ ta xét trường hợp có k thành phần liên thông. Không làm mất tính tổng quát, ta giả sử rằng các đỉnh được sắp xếp theo thứ tự của thành phần liên thông

Ta thấy rằng, mỗi ma trận L_i chi chứa các đỉnh trong cùng một thành phần liên thông i là một ma trận Laplacian con ứng với thành phần liên thông thứ i.

Vì mỗi L_i là một ma trận Laplacian của một đồ thị liên thông, theo chứng minh phần 3, ta thấy mỗi L_i có giá trị riêng bằng 0 với vector riêng có giá trị bằng 1 với các đỉnh thuộc thành phần liên thông i và 0 đối với các giá trị còn lại, có thể dễ thấy các vector này độc lập tuyến tính . Do đó, số lượng giá trị riêng bằng 0 của ma trận Laplacian bằng số thành phần liên thông của đồ thị

4. Chuẩn hóa ma trận Laplacian

Để chuẩn hóa ma trận Laplacian và giảm ảnh hưởng của các điểm có mức độ kết nối cao hơn, ta chuẩn hóa ma trận Laplacian không chuẩn hóa L bằng cách nhân với $D^{-1/2}$ từ cả hai phía:

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2}$$

Đây là ma trận Laplacian chuẩn hóa theo phương pháp của Ng, Jordan và Weiss (2002), gọi là ma trận Laplacian chuẩn hóa đối xứng (L_{sym}). Mục đích của bước chuẩn hóa này là để làm cho các điểm dữ liệu có mức độ kết nối khác nhau không bị ảnh hưởng quá nhiều bởi số lượng các kết nối (bậc) của chúng, giúp cho các cụm được phân tách rõ ràng hơn.

5. Mục đích của chuẩn hóa

Việc chuẩn hóa giúp đảm bảo rằng ma trận Laplacian có thể phản ánh chính xác các mối quan hệ giữa các điểm trong đồ thị, và k-means có thể hoạt động hiệu quả trên không gian đặc trưng này mà không bị ảnh hưởng quá nhiều bởi các sự khác biệt về độ lớn của bậc các đỉnh.

Sau khi hoàn thành bước này, ta có ma trận Laplacian chuẩn hóa L_{sym}, có thể sử dụng để tiếp tục tính toán các vectơ riêng (eigenvectors) phục vụ cho bước tiếp theo trong quá trình phân cụm spectral.

Nếu không chuẩn hóa: Spectral clustering có thể không phân cụm chính xác, vì các nút có bậc cao sẽ chi phối kết quả phân cụm. Ví dụ, nút 1 (có bậc 3) có thể bị coi là trung tâm của một cụm duy nhất, dù rằng thực tế nó có thể thuộc về một cụm khác khi xét đến mối quan hệ tương đồng với các nút khác.

Nếu chuẩn hóa: Kết quả phân cụm sẽ dựa trên mối quan hệ tương đối giữa các nút, thay vì chỉ phụ thuộc vào bậc của chúng. Ví dụ, trong đồ thị, việc chuẩn hóa giúp nhận ra rằng nút 2 và nút 3 có mối quan hệ "cân bằng" với nút 1, do đó chúng có thể thuộc về các cụm khác nhau.

Trường hợp chuẩn hóa: Khi chuẩn hóa bằng $D^{-1/2}$, ma trận Laplacian chuẩn hóa L_{norm} sẽ "cân bằng" sự ảnh hưởng của các nút, bất kể bậc của chúng. Cụ thể:

- Trọng số giữa các nút được điều chỉnh bởi bậc của chúng, giúp các nút có bậc thấp (như nút 2 và nút 3) không bị "lép vế" so với các nút có bậc cao.
- Điều này làm cho việc phân cụm trở nên công bằng hơn, phản ánh chính xác cấu trúc thực sự của đồ thị.

Ma trận L_{sym} là một ma trận đối xứng và bán xác định dương, có nghĩa là tất cả các giá trị riêng của nó đều không âm. Bước chuẩn hóa này rất quan trọng vì nó giúp nắm bắt các mối quan hệ tương đồng địa phương trong dữ liệu một cách hiệu quả hơn. Ví dụ, nếu một đỉnh có bậc cao (nhiều kết nối), ảnh hưởng của nó đối với các đỉnh lân cận sẽ được giảm bớt sau khi chuẩn hóa, giúp thuật toán phân cụm chính xác hơn trong trường hợp đồ thị tương đồng có phân bố bậc không đồng đều.

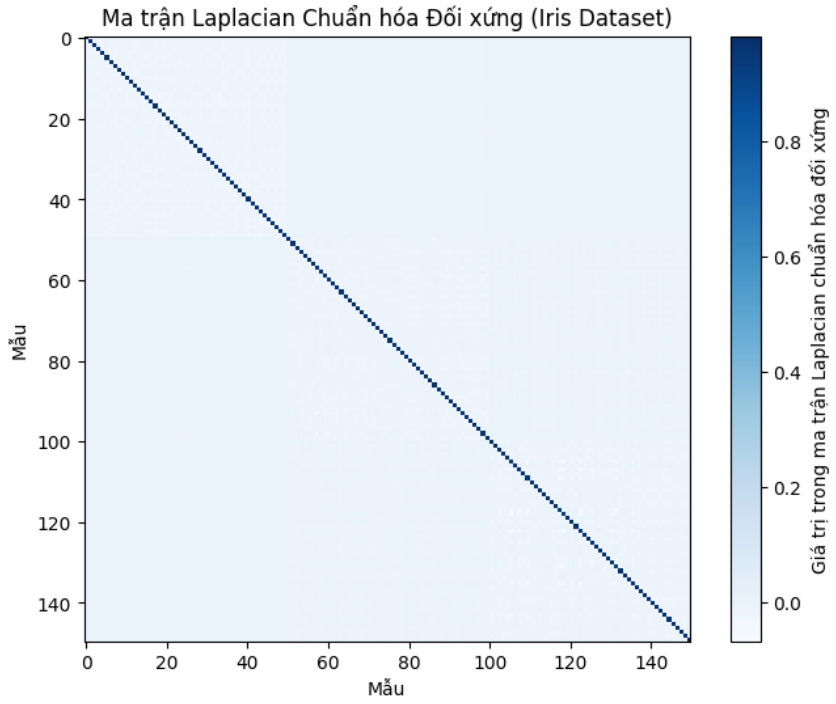


Figure 5: Trực quan hóa ma trận laplacian của tập dữ liệu Iris

3.3 Bước 3: Tính toán vector riêng và trị riêng từ ma trận Laplacian chuẩn hóa và chọn k.

Trong bước này, chúng ta tính toán vector riêng và trị riêng từ ma trận Laplacian chuẩn hóa đã được tính trong Bước 2.

Chọn k? Số k ở đây là số vector riêng (eigenvectors) đầu tiên cần tính toán từ ma trận Laplacian chuẩn hóa. k đồng thời là số cụm (clusters) mà bạn dự định phân tích trong dữ liệu. Với tập dữ liệu chúng ta đang xét, ta đã biết được số nhãn của dữ liệu là 3, tương ứng với các loài: Iris setosa, Iris virginica, Iris versicolor.

Tuy nhiên, trong một số bài toán, ta không biết trước được số lượng nhãn hay cụm. Trong trường hợp này, ta có thể ước lượng giá trị k bằng các phương pháp sau:

Phân tích giá trị riêng (Eigenvalue Gap Analysis):

- Tính tất cả các giá trị riêng của ma trận Laplacian chuẩn hóa.
- Xác định khoảng cách lớn nhất giữa hai giá trị riêng liên tiếp.
- Số lượng các giá trị riêng nhỏ nhất trước khoảng cách lớn nhất sẽ là k.

Ví dụ: Nếu các giá trị riêng lần lượt là $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots$, và $\lambda_3 - \lambda_4$ là khoảng cách lớn nhất, thì $k = 3$. Lý do là các giá trị riêng nhỏ nhất của ma trận Laplacian biểu thị số lượng cụm tốt nhất.

Sử dụng thuật toán Elbow Method:

- Tính số cụm k khác nhau (ví dụ: $k = 1, 2, 3, \ldots$).
- Với mỗi k, tính tổng khoảng cách của các điểm đến tâm cụm (inertia).
- Tìm "điểm gãy" (elbow) trong đồ thị để xác định k.

Sử dụng chỉ số Silhouette:

- Chạy Spectral Clustering với nhiều giá trị k.
- Tính chỉ số Silhouette cho từng giá trị k và chọn giá trị có chỉ số cao nhất hoặc Dunn index.

Trong trường hợp này, nếu theo phương pháp heuristic eigenvalue, ta chỉ chú ý đến trị riêng nhỏ nhất thứ hai vì giá trị trị riêng thứ nhất luôn là 0, và vectơ riêng tương ứng là một vector hằng, có ý nghĩa đại diện cho một cụm duy nhất.

Vector riêng thứ hai (tương ứng với giá trị riêng nhỏ thứ hai) chỉ ra cách phân tách đồ thị thành 2 cụm tốt nhất. Vector riêng thứ ba mở rộng thêm thông tin khi phân đồ thị thành 3 cụm, và tiếp tục như vậy cho các giá trị lớn hơn của k.

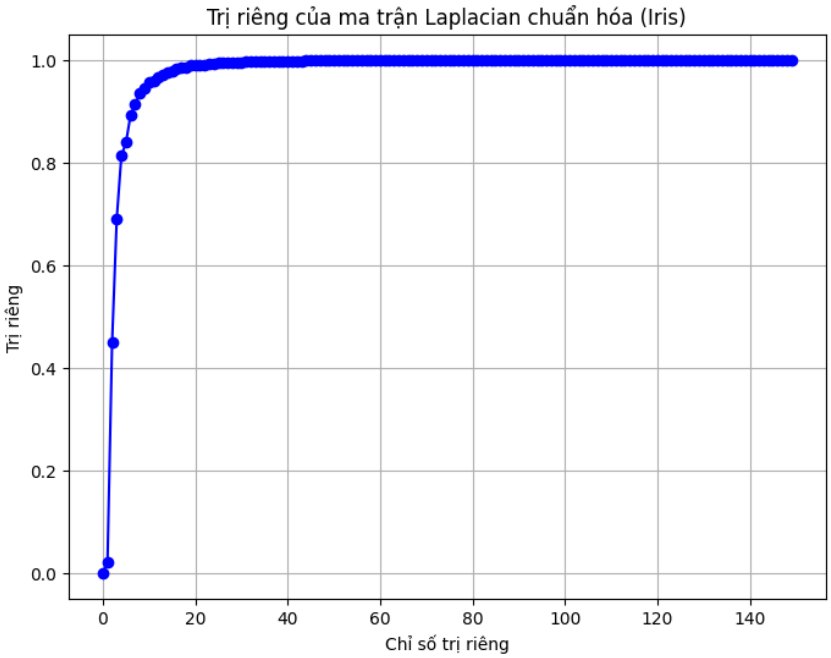


Figure 6: Trị riêng của ma trận Laplacian chuẩn hóa (Iris)

3.4 Bước 4: Tạo ma trận $U \in R^{n\times k}$ và chuẩn hóa từ k vector riêng đã chọn

Sau khi tính toán và chọn k vector riêng, ta sẽ tạo ra ma trận $U \in R^{n\times k}$ từ các vector này. Tiếp theo, ta thực hiện bước chuẩn hóa các hàng của ma trận này. Đây là một điểm khác biệt quan trọng so với thuật toán Shi-Malik.

Thuật toán tạo ra ma trận $T \in R^{n\times k}$ từ U bằng cách chuẩn hóa mỗi hàng của ma trận U sao cho độ dài (norm) của từng hàng bằng 1. Điều này có nghĩa là mỗi phần tử t_{ij} trong ma trận T sẽ được tính bằng cách chia phần tử tương ứng u_{ij} trong ma trận U cho căn bậc hai của tổng bình phương của tất cả các phần tử trong hàng thứ i của ma trận U.

Bước chuẩn hóa này rất quan trọng, giúp giải quyết các vấn đề tiềm ẩn do sự chênh lệch về độ lớn giữa các phần tử trong các vector riêng của ma trận Laplacian chuẩn hóa (L_{sym}). Điều này đảm bảo rằng tất cả các điểm dữ liệu đều được biểu diễn một cách công bằng trong quá trình phân cụm.

Cụ thể, trong không gian các vector riêng U, các điểm dữ liệu có thể có độ lớn (norm) rất khác nhau. Một số vector có độ lớn rất nhỏ, trong khi các vector khác lại có độ lớn lớn hơn nhiều. Điều này có thể dẫn đến sự thiên lệch trong quá trình phân cụm, vì thuật toán K-means sẽ ưu tiên các điểm có độ lớn lớn hơn (vì chúng có khoảng cách lớn hơn trong không gian vector).

Việc chuẩn hóa các hàng của ma trận U về độ dài 1 giúp mỗi điểm dữ liệu đóng vai trò công bằng trong việc xác định cụm. Nhờ đó, quá trình phân cụm chỉ dựa vào hướng của các vector, không bị ảnh hưởng bởi độ lớn của chúng, làm cho việc phân cụm trở nên chính xác và công bằng hơn.

Chuẩn hóa dữ liệu để sử dụng k-means hiệu quả hơn

Thuật toán k-means hoạt động tốt hơn khi dữ liệu được phân phối trên một không gian chuẩn hóa, giống như mặt cầu đơn vị. Khi các hàng của ma trận dữ liệu $\mathbf{U}$ được chuẩn hóa về độ dài bằng 1, các điểm dữ liệu sẽ được ánh xạ lên mặt cầu đơn vị trong không gian k-chiều. Điều này giúp k-means phân cụm dựa trên hướng của các vectơ, thay vì bị ảnh hưởng bởi độ lớn của chúng.

Loại bỏ sự khác biệt tỷ lệ không liên quan

Trong không gian vectơ, khoảng cách giữa các điểm dữ liệu phụ thuộc vào cả hướng và độ lớn của các vectơ. Tuy nhiên, đối với mục tiêu phân cụm, chỉ có hướng của các vectơ là quan trọng. Bằng cách chuẩn hóa, các sự khác biệt về tỷ lệ giữa các vectơ không liên quan sẽ được loại bỏ, giúp thuật toán phân cụm tập trung vào các cấu trúc quan trọng của dữ liệu.

Ảnh hưởng của chuẩn hóa đối với k-means

Khi các hàng của ma trận dữ liệu $\mathbf{U}$ được chuẩn hóa về độ dài 1, các điểm dữ liệu trong không gian vectơ riêng sẽ được ánh xạ lên mặt cầu đơn vị trong không gian k-chiều. Điều này có ý nghĩa quan trọng đối với cách thuật toán k-means hoạt động, đặc biệt trong bước cuối cùng của phương pháp phân cụm phổ chuẩn hóa (Normalized Spectral Clustering).

Tập trung vào hướng của vectơ

Trong không gian k-chiều, mỗi hàng của ma trận $\mathbf{U}$ (trước khi chuẩn hóa) là một vectơ có cả hướng và độ lớn. Tuy nhiên, để phân cụm, điều quan trọng là chỉ sử dụng hướng của các vectơ để xác định sự tương đồng giữa chúng, không bị ảnh hưởng bởi sự khác biệt về độ lớn, vốn có thể xuất phát từ dữ liệu ban đầu có khoảng cách khác nhau.

Ánh xạ lên mặt cầu đơn vị

Khi chuẩn hóa, mỗi vectơ $\mathbf{u}_i$ trong ma trận $\mathbf{U}$ trở thành $\mathbf{t}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$, với $\|\mathbf{t}_i\| = 1$. Điều này có nghĩa là mọi điểm dữ liệu sẽ được ánh xạ lên mặt cầu đơn vị trong không gian k-chiều. Trên mặt cầu, khoảng cách giữa các điểm dữ liệu chủ yếu phụ thuộc vào góc giữa các vectơ, hay còn gọi là cosine similarity, thay vì phụ thuộc vào độ lớn.

Hình học của không gian chuẩn hóa

Giả sử chúng ta có một tập dữ liệu U trước khi chuẩn hóa:

$$U = \begin{bmatrix} 1 & 2 & 2 \\ 3 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Các điểm dữ liệu này nằm trong không gian 3-chiều và không nhất thiết phải có độ dài bằng nhau. Sau khi chuẩn hóa từng hàng để có độ dài bằng 1, ta thu được ma trận T:

$$T = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{3}{5} & \frac{4}{5} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Lúc này, các điểm dữ liệu nằm trên một mặt cầu đơn vị trong không gian 3-chiều. Tất cả các điểm đều cách gốc tọa độ đúng 1 đơn vị. Trên mặt cầu, khoảng cách giữa các điểm dữ liệu phụ thuộc chủ yếu vào góc giữa các vectơ, thay vì bị ảnh hưởng bởi độ lớn.

Ví dụ minh họa

Hãy tưởng tượng 3 điểm dữ liệu trong không gian 2-chiều:

- Điểm A: $(1, 1)$
- Điểm B: $(2, 2)$
- Điểm C: $(0, 3)$

Trước khi chuẩn hóa

Khoảng cách Euclidean giữa A và B là:

$$\sqrt{(2-1)^2 + (2-1)^2} = \sqrt{2}$$

Tuy nhiên, nếu xem xét hướng của các vectơ, cả A và B đều nằm trên đường $y = x$, tức là chúng có cùng hướng.

Sau khi chuẩn hóa (mỗi điểm được chia cho độ dài của nó)

- A chuẩn hóa: $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$
- B chuẩn hóa: $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$
- C chuẩn hóa: $(0, 1)$

Trên mặt cầu, A và B trùng nhau vì chúng có cùng hướng, trong khi C khác biệt.

Tóm lại

Chuẩn hóa các hàng của ma trận U về độ dài 1 là một bước quan trọng giúp:

- Loại bỏ sự khác biệt về độ lớn giữa các điểm.
- Đảm bảo các điểm được phân cụm dựa trên hướng (góc) trong không gian vectơ riêng.
- Làm cho thuật toán k-means hoạt động hiệu quả hơn trong việc phân cụm dữ liệu chuẩn hóa trên mặt cầu đơn vị.

3.5 Bước 5: Sử dụng k-means để phân cụm trên mặt cầu đã ánh xạ

K-means, một thuật toán phân cụm phổ biến, thường hoạt động trong không gian Euclidean, nơi các điểm dữ liệu được phân nhóm dựa trên khoảng cách Euclidean giữa chúng. Tuy nhiên, khi dữ liệu được chuẩn hóa và ánh xạ lên mặt cầu đơn vị, các phép toán phân cụm của k-means sẽ hoạt động khác biệt. Cụ thể, khi dữ liệu nằm trên mặt cầu đơn vị, các phân nhóm của k-means sẽ phụ thuộc vào góc giữa các vectơ, thay vì chỉ phụ thuộc vào khoảng cách Euclidean.

Dưới đây là cách thức mà k-means hoạt động khi dữ liệu đã được chuẩn hóa và ánh xạ lên mặt cầu đơn vị:

1. Ánh xạ dữ liệu lên mặt cầu đơn vị

Khi dữ liệu được chuẩn hóa về độ dài 1, mỗi điểm dữ liệu $\mathbf{x}_i$ trở thành một vectơ đơn vị, nghĩa là $\|\mathbf{x}_i\| = 1$ với mọi i. Các điểm dữ liệu không còn được biểu diễn bởi các vectơ có độ dài tùy ý mà thay vào đó là các vectơ có độ dài bằng 1, nằm trên mặt cầu đơn vị trong không gian k-chiều.

2. Khoảng cách giữa các điểm dữ liệu

Khoảng cách giữa các điểm trên mặt cầu đơn vị không còn tính theo khoảng cách Euclidean đơn giản nữa. Thay vào đó, khoảng cách góc hoặc cosine similarity được sử dụng. Khoảng cách này được tính bằng cách đo góc giữa các vectơ. Nếu hai vectơ có góc nhỏ, tức là hướng của chúng rất giống nhau, thì chúng sẽ được coi là gần nhau.

Khoảng cách giữa hai vectơ đơn vị $\mathbf{x}_i$ và $\mathbf{x}_j$ có thể tính bằng cosine similarity:

$$\text{cosine similarity}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j = \cos(\theta)$$

Trong đó θ là góc giữa hai vectơ. Khoảng cách Euclidean giữa hai điểm trên mặt cầu đơn vị cũng có thể tính bằng:

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 = 2(1 - \cos(\theta))$$

3. Quá trình k-means trên mặt cầu đơn vị
K-means sẽ hoạt động như sau khi dữ liệu đã được chuẩn hóa và ánh xạ lên mặt cầu đơn vị:

- **Khởi tạo các tâm cụm (centroids)**: Các tâm cụm ban đầu có thể được khởi tạo ngẫu nhiên như thông thường. Tuy nhiên, sau khi dữ liệu được chuẩn hóa, các tâm cụm này phải nằm trên mặt cầu đơn vị (tức là phải có độ dài 1).
- **Gán các điểm dữ liệu vào cụm**: Mỗi điểm dữ liệu sẽ được gán vào cụm có tâm cụm gần nhất. Tuy nhiên, thay vì tính khoảng cách Euclidean thông thường, k-means sẽ sử dụng khoảng cách góc (hoặc cosine similarity) để đo sự tương đồng giữa điểm dữ liệu và các tâm cụm. Cụ thể, điểm dữ liệu sẽ được gán vào cụm có góc nhỏ nhất với tâm cụm.
- **Cập nhật các tâm cụm**: Sau khi các điểm dữ liệu đã được phân cụm, bước tiếp theo là cập nhật các tâm cụm. Trên mặt cầu đơn vị, cập nhật tâm cụm có thể được thực hiện bằng cách tính trung bình các vectơ của các điểm trong cùng một cụm. Tuy nhiên, do các vectơ trên mặt cầu có độ dài 1, nên phép tính trung bình sẽ không đơn giản như trong không gian Euclidean. Thay vào đó, trung bình của các vectơ sẽ được tính trên góc giữa chúng và sau đó được chuẩn hóa lại để đảm bảo rằng tâm cụm cuối cùng vẫn nằm trên mặt cầu đơn vị.
- **Lặp lại quá trình**: Các bước phân cụm và cập nhật tâm cụm sẽ được lặp lại cho đến khi hội tụ, tức là các tâm cụm không thay đổi hoặc thay đổi rất ít giữa các lần lặp.

4. Hình học của k-means trên mặt cầu
Khi dữ liệu nằm trên mặt cầu đơn vị, phân cụm của k-means sẽ dựa trên góc giữa các vectơ thay vì chỉ phụ thuộc vào khoảng cách Euclidean. Điều này giúp k-means phân cụm các điểm dữ liệu có cùng hướng nhưng có thể có độ dài khác nhau trong không gian ban đầu. Việc chuẩn hóa các vectơ về độ dài 1 giúp đảm bảo rằng các cụm được xác định dựa trên cấu trúc không gian của các điểm dữ liệu thay vì chỉ dựa trên khoảng cách giữa chúng.
5. Lý do k-means trên mặt cầu hiệu quả

- **Loại bỏ ảnh hưởng của độ lớn**: Các điểm dữ liệu được chuẩn hóa giúp loại bỏ ảnh hưởng của độ lớn trong quá trình phân cụm. Điều này giúp k-means chỉ phân cụm dựa trên hướng của các vectơ, tạo ra các nhóm tương tự về hướng nhưng có thể có độ lớn khác nhau.
- **Cải thiện sự phân biệt các cụm**: Việc phân cụm trên mặt cầu giúp giảm thiểu ảnh hưởng của các điểm dữ liệu có độ lớn rất khác biệt, khiến k-means dễ dàng phân biệt các nhóm hơn.

Kết luận

K-means trên mặt cầu đơn vị hoạt động bằng cách phân cụm dựa vào góc giữa các vectơ, thay vì khoảng cách Euclidean truyền thống. Việc chuẩn hóa các vectơ giúp loại bỏ sự phụ thuộc vào độ lớn và đảm bảo rằng thuật toán k-means chỉ tập trung vào hướng của các điểm, từ đó phân nhóm các điểm dữ liệu có sự tương đồng cao về

3.6 Bước 6: Gán nhãn cho các điểm dữ liệu dựa trên kết quả phân cụm

Mục tiêu: Dựa trên kết quả phân cụm từ thuật toán K-means, cần gán nhãn cho từng điểm dữ liệu ban đầu trong tập Iris. Các nhãn này sẽ tương ứng với các cụm mà thuật toán Spectral Clustering đã xác định.
Cách thực hiện:

1. **Kết quả từ bước 5:** Sau khi áp dụng K-means trên không gian vector riêng đã chuẩn hóa, ta thu được nhãn của mỗi điểm dữ liệu. Mỗi nhãn này đại diện cho một cụm (cluster) mà điểm dữ liệu đó thuộc về.
2. **Gán nhãn cho dữ liệu gốc:**

- Tập dữ liệu Iris ban đầu gồm ba nhãn: *Iris-setosa*, *Iris-versicolor*, *Iris-virginica*. Các nhãn này sẽ được ánh xạ vào kết quả phân cụm từ K-means.
- K-means không tự động liên kết các nhãn gốc với cụm, vì thuật toán chỉ xác định các cụm mà không biết thông tin nhãn thực tế. Vì vậy, cần một bước ánh xạ thủ công để so sánh và khớp các cụm với nhãn gốc.

3. **Cách ánh xạ nhãn:**
   - Dựa trên nhãn thực tế và nhãn từ K-means, ta đếm tần suất nhãn thực tế trong mỗi cụm.
   - Ánh xạ cụm có tần suất cao nhất với một nhãn thực tế. Ví dụ:
     - Cụm 0 có nhiều điểm *Iris-setosa* nhất → Gán cụm 0 thành nhãn *Iris-setosa*.
     - Cụm 1 có nhiều điểm *Iris-versicolor* nhất → Gán cụm 1 thành nhãn *Iris-versicolor*.
     - Cụm 2 có nhiều điểm *Iris-virginica* nhất → Gán cụm 2 thành nhãn *Iris-virginica*.
4. **Đánh giá hiệu suất phân cụm:**
   - So sánh các nhãn đã gán với nhãn thực tế của tập dữ liệu để đánh giá độ chính xác.
   - Các chỉ số phổ biến:
     - **Độ chính xác (Accuracy):** Tỷ lệ nhãn gán đúng so với tổng số điểm.
     - **Ma trận nhầm lẫn (Confusion Matrix):** Để kiểm tra chi tiết các nhãn bị phân cụm sai.

Áp dụng cho tập Iris:

1. **Kết quả K-means:** Tập Iris có ba cụm tương ứng với ba loài hoa: *Iris-setosa*, *Iris-versicolor*, và *Iris-virginica*. Sau khi áp dụng K-means, ta thu được nhãn phân cụm như sau:
   - Cụm 0: Nhãn K-means gán cho các điểm gần *Iris-setosa*.
   - Cụm 1: Nhãn K-means gán cho các điểm gần *Iris-versicolor*.
   - Cụm 2: Nhãn K-means gán cho các điểm gần *Iris-virginica*.
2. **Ánh xạ nhãn:**
   - So sánh các điểm trong mỗi cụm với nhãn thực tế.
   - Gán nhãn cụm theo tần suất nhãn thực tế xuất hiện nhiều nhất trong cụm đó.
3. **Đánh giá:** Với tập Iris, ta có thể dễ dàng kiểm tra nhãn đã gán và đánh giá độ chính xác, vì dữ liệu Iris có nhãn thực tế rõ ràng. Chúng em sử dụng chỉ số ARI và tính được ARI = 0.7195837484778036 cho thấy độ chính xác của Spectral Clustering trên tập Iris khá cao

Kết quả mong đợi: Sau bước này, mỗi điểm dữ liệu trong tập Iris sẽ được gán một nhãn mới dựa trên cụm của nó. Các nhãn này phản ánh kết quả phân cụm của Spectral Clustering, cho phép so sánh trực tiếp với nhãn thực tế. Kết quả có thể được trình bày bằng ma trận nhầm lẫn hoặc trực quan hóa đồ thị để minh họa rõ ràng sự khác biệt giữa phân cụm và nhãn thực tế.

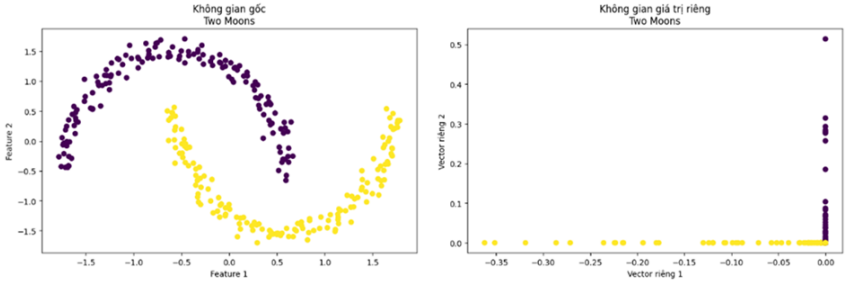
4 Ưu, nhược điểm và ứng dụng thực tiễn của Spectral Clustering

4.1 1. Ưu điểm của Spectral Clustering

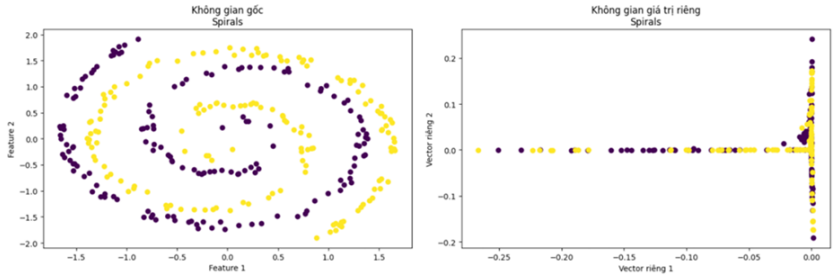
Xử lý dữ liệu phi tuyến

- Spectral Clustering chuyển dữ liệu từ không gian ban đầu sang không gian giá trị riêng (eigenvalue space).
- Trong không gian mới, các cụm dữ liệu phi tuyến có thể được phân tách rõ ràng hơn so với các thuật toán truyền thống như K-means.

- Ví dụ: Dữ liệu có dạng đường cong hoặc hình vòng cung – K-means thường không phân cụm tốt, nhưng Spectral Clustering lại hoạt động hiệu quả.



- Trong không gian dốc, ta thấy dữ liệu có hai cụm hình bán nguyệt đối xứng và giao nhau. Do đó, K-means không thể vẽ đường biên tách hai cụm này mà sẽ phân cụm sai (ví dụ: chia theo hướng ngang hoặc dọc, dẫn đến mỗi cụm có cả điểm từ hai bán nguyệt).
- Spectral Clustering chuyển dữ liệu từ không gian gốc sang không gian giá trị riêng. Trong không gian giá trị riêng, hai cụm hình bán nguyệt ban đầu được "kéo thẳng" ra theo hai trục của vector riêng, giúp thuật toán dễ dàng nhận diện và phân cụm chính xác, ngay cả khi các cụm có hình dạng phi tuyến.



- Dữ liệu có dạng hai đường xoắn ốc giao nhau khiến K-means không thể xử lý được dạng dữ liệu này vì các cụm xoắn ốc giao nhau và không có đường phân cách đơn giản nào có thể chia tách chúng. Thuật toán sẽ tạo các cụm sai lệch dựa trên khoảng cách Euclidean (ví dụ: nhóm các điểm gần nhau trong không gian gốc, bất kể chúng thuộc đường xoắn ốc nào).
- Sau khi chuyển đổi sang không gian giá trị riêng, các điểm dữ liệu thuộc hai cụm xoắn ốc được tách biệt theo trục vector riêng, giúp phân cụm chính xác ngay cả với dữ liệu có cấu trúc phức tạp như xoắn ốc.

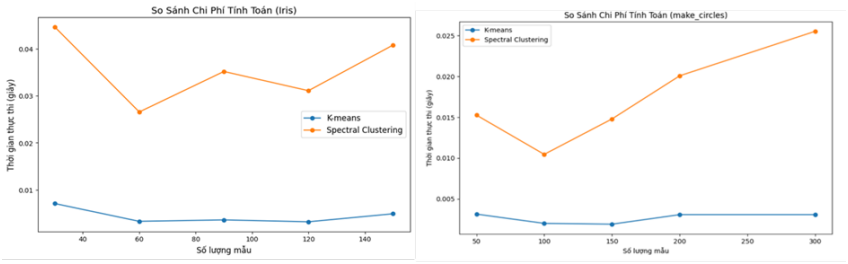
Không yêu cầu cụm có dạng hình cầu

- K-means giả định rằng các cụm phải có dạng hình cầu hoặc elip và sử dụng khoảng cách Euclidean để phân cụm. Điều này làm hạn chế khả năng xử lý dữ liệu có hình dạng phức tạp.
- Spectral Clustering không phụ thuộc vào giả định này, cho phép phân cụm dữ liệu có biên dạng đa dạng, ví dụ: các cụm hình chữ U, hình chữ C.

4.2 2. Nhược điểm của Spectral Clustering

Chi phí tính toán cao

- Tính toán giá trị riêng và vector riêng từ ma trận tương đồng có độ phức tạp $O(n^3)$.
- Với dữ liệu lớn ($n > 10,000$), thời gian tính toán sẽ rất lớn, không phù hợp cho ứng dụng thời gian thực.



Đối với bộ dữ liệu iris:

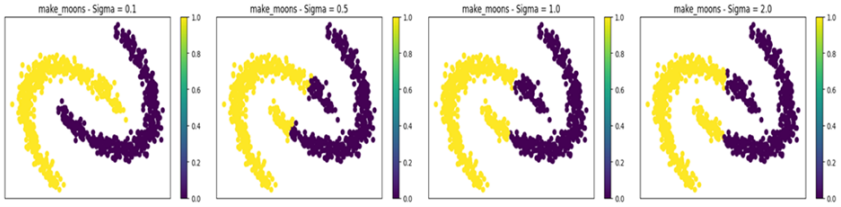
- Khi kích thước tập dữ liệu nhỏ (40–60 mẫu), K-means có thời gian thực thi rất thấp, dưới 0.01 giây, trong khi Spectral Clustering tiêu tốn khoảng 0.04 giây.
- Khi số lượng mẫu tăng lên (140 mẫu), thời gian của K-means vẫn gần như không đổi, trong khi thời gian của Spectral Clustering tăng nhẹ, gần 0.035 giây.

Đối với bộ dữ liệu make_circle:

- Với dữ liệu phi tuyến, chi phí tính toán của Spectral Clustering cao hơn hẳn so với K-means ngay từ khi số lượng mẫu nhỏ (50 mẫu, khoảng 0.025 giây).
- Khi số lượng mẫu tăng lên (300 mẫu), chi phí của Spectral Clustering tăng dần, chạm ngưỡng cao hơn 0.025 giây, trong khi thời gian của K-means vẫn giữ ổn định ở mức rất thấp.

Phụ thuộc vào tham số

- Tham số σ trong ma trận tương đồng quyết định mức độ ảnh hưởng của các điểm lân cận.



- $\sigma = 0.1$:
  - Khi giá trị quá nhỏ, thuật toán chỉ tập trung vào các mối liên kết cục bộ. Điều này có thể dẫn đến oversegmentation, khiến một nửa hình bán nguyệt bị chia thành nhiều cụm nhỏ.
  - Kết quả không tối ưu.
- $\sigma = 0.5$:
  - Giá trị này là mức trung bình hợp lý. Nó cân bằng giữa việc tập trung vào các liên kết cục bộ và toàn cục.
  - Các điểm trong cùng một nửa hình bán nguyệt sẽ được nhóm thành 1 cụm đầy đủ.
  - Đây thường là giá trị σ tối ưu cho bộ dữ liệu make_moons.
- $\sigma = 1.0$:
  - Khi σ lớn hơn, các mối quan hệ toàn cục mạnh hơn, dẫn đến việc các điểm từ các cụm khác nhau bắt đầu được nhóm chung.
  - Mặc dù đôi khi vẫn chia được 2 cụm, nhưng biên giới giữa các cụm sẽ không sắc nét, có khả năng một số điểm ở gần biên bị gán sai cụm.
  - Kết quả gần đúng, nhưng không tối ưu bằng $\sigma = 0.5$.
- $\sigma = 2.0$:
  - Giá trị rất lớn làm cho hầu hết các điểm bị coi là gần nhau. Điều này dẫn đến việc Spectral Clustering không thể phân biệt các cụm rõ ràng.
  - Kết quả sai hoàn toàn: Tất cả dữ liệu có thể bị gộp thành 1 cụm duy nhất.

4.3 3. Ứng dụng thực tế của Spectral Clustering

Phân cụm hình ảnh (Image Segmentation)

- Spectral Clustering giúp tách biệt các vùng trong hình ảnh dựa trên các tính chất tương đồng như màu sắc, kết cấu, hoặc vị trí không gian.
- Quy trình:
  - Chuyển đổi hình ảnh thành đồ thị, mỗi pixel là một đỉnh, trọng số cạnh biểu thị sự tương đồng giữa các pixel.
  - Sử dụng Spectral Clustering để nhóm các pixel tương tự vào cùng một cụm.
- Ví dụ ứng dụng:
  - Y tế: Phân đoạn ảnh MRI để hỗ trợ chẩn đoán khối u.
  - Vệ tinh: Phân vùng đất, nước, rừng trong ảnh vệ tinh.
  - Sinh học: Phân tích ảnh tế bào để nghiên cứu đặc tính sinh học.

Phân cụm văn bản (Text Clustering)

- Spectral Clustering giúp nhóm các văn bản có nội dung tương tự thành các cụm dựa trên đặc trưng từ ngữ hoặc ngữ nghĩa.
- Phương pháp biểu diễn:
  - Bag of Words (BoW): Biểu diễn văn bản thành vector tần suất từ.
  - TF-IDF: Giảm ảnh hưởng của từ phổ biến, tăng trọng số từ quan trọng.
- Quy trình:
  - Tính ma trận tương đồng giữa các văn bản.
  - Sử dụng Spectral Clustering để nhóm văn bản.
- Ví dụ ứng dụng:
  - Phân loại tài liệu: Tự động sắp xếp tài liệu theo chủ đề.
  - Tóm tắt văn bản: Rút gọn nội dung theo từng nhóm chủ đề.
  - Phân tích phản hồi khách hàng: Tìm hiểu các vấn đề phổ biến mà khách hàng gặp phải.

Phân tích mạng xã hội (Social Network Analysis)

- Spectral Clustering được sử dụng để tìm các cộng đồng liên kết (communities) trong mạng xã hội.
- Biểu diễn:
  - Mạng xã hội được biểu diễn dưới dạng đồ thị, trong đó:
    - Đỉnh: Người dùng.
    - Cạnh: Quan hệ giữa người dùng (bạn bè, tương tác).
    - Trọng số: Mức độ tương tác, số bạn chung, hoặc sự tương đồng về sở thích.
- Ví dụ ứng dụng:
  - Phát hiện cộng đồng: Tìm các nhóm có mối quan hệ chặt chẽ trên Facebook hoặc Twitter.
  - Thương mại điện tử: Phân tích mạng người dùng để tối ưu hóa đề xuất sản phẩm.
  - Nghiên cứu học thuật: Phân tích cấu trúc mạng giữa các nhà khoa học để tìm hiểu mối quan hệ hợp tác.

Kết luận

- Spectral Clustering là một công cụ mạnh mẽ để phân cụm dữ liệu, đặc biệt với dữ liệu phi tuyến hoặc có cấu trúc phức tạp.
- Tuy nhiên, nó đòi hỏi lựa chọn tham số cẩn thận và không phù hợp với dữ liệu quá lớn.

5 So sánh Spectral Clustering với các thuật toán Clustering khác

Sau đây mình sẽ so sánh Spectral Clustering với các thuật toán Clustering khác, ở đây mình sẽ so sánh với 2 thuật toán: K–mean và Hierarchical Clustering.

5.1 1. So sánh với K–mean

Sơ bộ về K–mean

Nhắc lại sơ bộ về K–mean, K–mean sẽ tạo một số tâm cụm ngẫu nhiên k (là số cụm người dùng muốn phân chia). Mỗi lần lặp, K–mean tính toán khoảng cách Euclid của mỗi điểm tới k tâm cụm, và gán từng điểm vào tâm cụm gần nó nhất. Sau đó với mỗi tâm cụm, K–mean tính trung bình khoảng cách Euclid của từng điểm trong cụm với tâm cụm, nếu có sai lệch thì tiếp tục lặp cho đến khi giá trị này không đổi hoặc đến khi tới giới hạn.

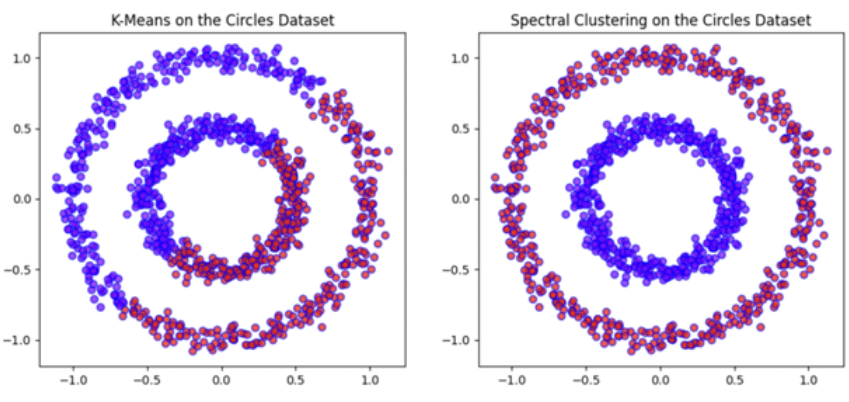
Điểm tương đồng giữa K–mean và Spectral Clustering

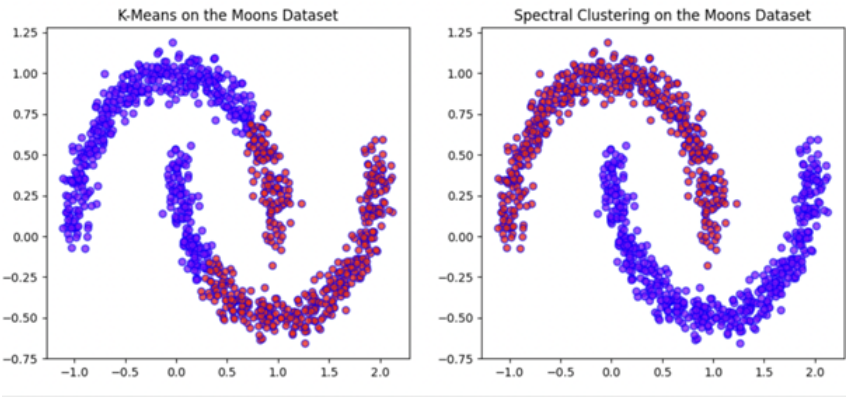
Về điểm tương đồng, chúng ta thấy Spectral Clustering và K–mean đều sử dụng thuật toán của K–mean để phân chia cụm và cần khai báo trước số cụm cần phân chia.

Điểm khác nhau giữa K–mean và Spectral Clustering

Về điểm khác nhau:

- **Cơ sở lý thuyết sử dụng:** Cơ sở lý thuyết áp dụng ở K–mean chỉ có sử dụng khoảng cách Euclid để tính toán giá trị và sai số của các lần lặp, trong khi đó Spectral Clustering còn áp dụng lý thuyết đồ thị và đại số. Spectral Clustering xem các điểm trên hệ trục tọa độ như một đồ thị có đỉnh là các điểm dữ liệu và có trọng số cạnh là khoảng cách giữa các điểm dữ liệu. Khi chuyển về đồ thị, ta tính toán được ma trận Laplacian và từ ma trận Laplacian tính các giá trị riêng và vector riêng. Từ vector riêng, ta biểu diễn các điểm vào không gian đặc trưng và thực hiện K–mean trên các điểm đã được quy chiếu vào không gian đặc trưng.
- **Hình dạng cụm được phân chia:** Về hình dạng của cụm được phân chia, K–mean thường chia các điểm dữ liệu thành hình tròn hoặc hình cầu, trong khi Spectral Clustering sẽ không có giả định trước về hình dạng của cụm. Các cụm có thể có hình dạng phi tuyến.






```python
# Import các thư viện cần thiết
import pandas as pd
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Load tập dữ liệu Iris
iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.Series(iris.target, name="True Label")

# Hiển thị dữ liệu trước phân cụm
print("Dữ liệu Iris trước phân cụm:")
print(X.head())

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Áp dụng thuật toán K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)

# Thêm nhãn cụm từ K-Means vào DataFrame
X['Cluster'] = kmeans.labels_

# Hiển thị dữ liệu sau phân cụm
print("\nDữ liệu Iris sau phân cụm:")
print(X.head())
```

Figure 7: Code K-Means Clustering trên tập dữ liệu Iris(1)

So sánh giữa K-Means và Spectral Clustering

1. Khả năng phân nhóm chính xác:

- **K-Means:** K-Means có khả năng phân nhóm rất tốt đối với lớp `setosa`, vì lớp này có sự phân tách rõ ràng và dễ dàng. Tuy nhiên, đối với hai lớp còn lại, `versicolor` và `virginica`, K-Means có thể gặp khó khăn nếu hai lớp này có sự giao thoa trong không gian đặc trưng. Khi có sự chồng lấn giữa các nhóm, K-Means có thể phân nhóm không chính xác, dẫn đến một số điểm bị phân vào nhóm sai.

```python
# Visualization trước phân cụm
from sklearn.decomposition import PCA

# Giảm chiều bằng PCA để trực quan hóa
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

X['PCA1'] = X_pca[:, 0]
X['PCA2'] = X_pca[:, 1]

# Biểu đồ scatter với nhãn thực tế
plt.figure(figsize=(14, 6))

# Trước phân cụm
plt.subplot(1, 2, 1)
sns.scatterplot(x=X['PCA1'], y=X['PCA2'], hue=y, palette='Set2', s=100)
plt.title('Dữ liệu trước phân cụm (Nhãn thực tế)')
plt.xlabel('PCA1')
plt.ylabel('PCA2')

# Sau phân cụm
plt.subplot(1, 2, 2)
sns.scatterplot(x=X['PCA1'], y=X['PCA2'], hue=X['Cluster'], palette='Set1', s=100)
plt.title('Dữ liệu sau phân cụm (Nhãn cụm)')
plt.xlabel('PCA1')
plt.ylabel('PCA2')

plt.tight_layout()
plt.show()
```

Figure 8: Code K-Means Clustering trên tập dữ liệu Iris(2)

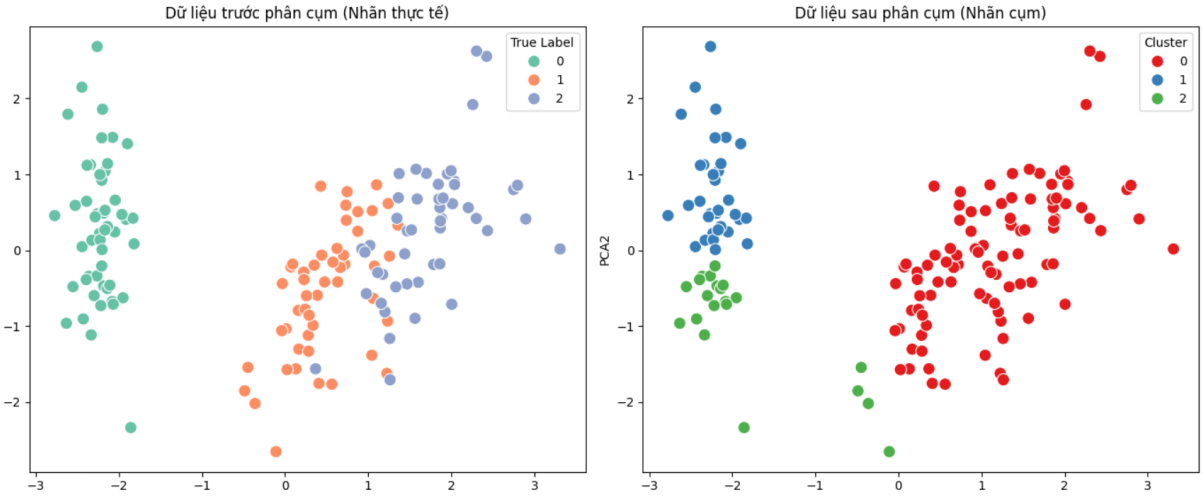


Figure 9: K-Means Clustering trên tập dữ liệu Iris

- **Spectral Clustering:** Spectral Clustering tỏ ra ưu việt hơn khi các nhóm có sự giao thoa nhẹ hoặc có hình dạng phức tạp. Tuy nhiên, trên tập dữ liệu Iris, vì các lớp đã được phân tách khá rõ ràng, kết quả phân nhóm của Spectral Clustering và K-Means không có sự khác biệt đáng kể. Dù vậy, Spectral Clustering vẫn có thể xử lý tốt hơn khi các nhóm có sự giao thoa nhẹ giữa các lớp.

2. Tốc độ tính toán:

- **K-Means:** K-Means có một ưu điểm lớn về tốc độ tính toán. Phương pháp này đơn giản và thường ít tốn tài nguyên hơn, đặc biệt khi dữ liệu không quá phức tạp. Việc tính toán chỉ yêu cầu việc xác định các centroid và phân nhóm các điểm dữ liệu về các centroid này, do đó tốc độ rất nhanh và hiệu quả.
- **Spectral Clustering:** Mặt khác, Spectral Clustering yêu cầu nhiều bước tính toán phức tạp hơn. Đầu tiên, nó phải xây dựng ma trận tương tác giữa các điểm dữ liệu, sau đó tính toán các giá trị riêng (eigenvectors) từ ma trận Laplacian hoặc adjacency. Quá trình này tốn thời gian và tài nguyên tính toán nhiều hơn so với K-Means.

3. Khả năng xử lý nhóm có hình dạng phức tạp:

- **K-Means:** Phương pháp K-Means không hiệu quả khi dữ liệu có các nhóm không có hình dạng đơn giản. Nó giả định rằng các nhóm có hình dạng hình cầu (spherical), vì vậy nếu các nhóm có hình dạng phức tạp hoặc không đều (ví dụ, có các nhóm hình elip hay xoắn), K-Means sẽ không thể phân nhóm chính xác.
- **Spectral Clustering:** Spectral Clustering không gặp phải vấn đề này. Phương pháp này có thể xử lý tốt các nhóm có hình dạng phức tạp hoặc không đều, vì nó sử dụng thông tin về mối quan hệ giữa các điểm dữ liệu thông qua ma trận tương tác, thay vì giả định về hình dạng của nhóm. Vì vậy, nếu các nhóm có hình dạng phức tạp, Spectral Clustering sẽ phân nhóm tốt hơn K-Means.

4. Sự phân tách giữa các lớp:

- **K-Means:** K-Means có thể phân tách các lớp rõ ràng rất tốt khi các lớp này phân tách rõ ràng và không có sự giao thoa. Tuy nhiên, nếu có sự giao thoa hoặc chồng chéo giữa các lớp (như trong trường hợp versicolor và virginica), K-Means có thể phân nhóm không chính xác, dẫn đến kết quả phân nhóm không tối ưu.
- **Spectral Clustering:** Trong khi đó, Spectral Clustering có khả năng phân tách các lớp có sự giao thoa tốt hơn. Nhờ vào việc xử lý mối quan hệ giữa các điểm qua ma trận tương tác, Spectral Clustering có thể tìm ra các nhóm mặc dù các lớp có sự chồng lấn hoặc giao thoa giữa chúng. Do đó, Spectral Clustering thường cho kết quả phân nhóm chính xác hơn khi các lớp không phân tách hoàn toàn.

Độ phức tạp thời gian

- K–mean có độ phức tạp thời gian là $O(KNT)$
  - K: số cụm cần phân chia
  - N: số điểm dữ liệu
  - T: số lần lặp tối đa
- Spectral Clustering có độ phức tạp là $O(N^3)$ trong trường hợp không sử dụng các biến thể tăng tốc độ xử lý:
  - $O(N^2)$: tính toán ma trận kề và ma trận Laplacian
  - $O(N^3)$: tính K giá trị riêng và vector riêng nhỏ nhất
  - $O(KNT)$: sử dụng K–mean như vừa nói

Từ đó chúng ta có thể thấy, với những bộ dữ liệu lớn, Spectral Clustering có chi phí tính toán cao hơn rất nhiều so với K–mean.

Kết luận

Với sự đơn giản, dễ triển khai, K–mean sẽ phù hợp với các dữ liệu đơn giản, tách biệt rõ ràng và hiệu quả nhất so với Spectral Clustering trong trường hợp dữ liệu lớn, nhưng Spectral Clustering sẽ mạnh hơn trong việc phân cụm dữ liệu phức tạp, có cấu trúc phi tuyến.

5.2 2. So sánh với Hierarchical clustering

Sơ bộ về Hierarchical clustering

- Hierarchical clustering và Spectral clustering là hai phương pháp Clustering khác nhau hoàn toàn. Hierarchical clustering xây dựng cây phân cấp (Dendrogram) để biểu diễn mối quan hệ tương đồng giữa các điểm dữ liệu, sử dụng việc phân chia và hợp cụm để tạo ra các cụm dữ liệu có đặc điểm tương đồng.
- Có 2 loại Hierarchical clustering: Agglomerative Hierarchical clustering và Divisive Hierarchical clustering. Ở đây chúng ta chỉ nói đến loại Agglomerative Hierarchical clustering.
- Các bước thực hiện như sau: Mỗi điểm dữ liệu sẽ được xem là một cụm phân biệt lúc đầu. Sau đó tính toán khoảng cách giữa các điểm dữ liệu, sau mỗi lần lặp hợp nhất hai điểm có khoảng cách nhỏ nhất bằng cách dùng cấu trúc dữ liệu hoặc tìm trên ma trận kề theo phương pháp liên kết nào đó (khoảng cách gần nhất, xa nhất, trung bình giữa 2 điểm thuộc 2 cụm khác nhau, ...) và kết thúc khi việc hợp nhất hai cụm bất kỳ không hiệu quả nữa (ví dụ như quá khoảng cách tối đa cho phép, chỉ còn một cụm duy nhất, ...).
- Thông tin về các cụm đã được hợp với nhau sẽ được biểu diễn qua cây Dendrogram.

Khai báo số lượng cụm

Vì có khả năng đánh giá hiệu quả của việc hợp hai cụm lại với nhau và tự động dừng khi việc hợp hai cụm không có hiệu quả, Hierarchical clustering khác Spectral Clustering ở chỗ có thể không cần khai báo trước số cụm cần phân chia.

Khả năng phát hiện cụm có cấu trúc phi tuyến

Hierarchical clustering sẽ gộp hai cụm lại nếu hai cụm có khoảng cách gần nhau hoặc đặc điểm tương tự nhau, vì vậy Hierarchical clustering cũng có thể phát hiện cụm có cấu trúc phi tuyến như Spectral Clustering.

Khả năng biểu diễn trực quan mối quan hệ giữa các cụm

- Cây phân cấp (Dendrogram) biểu diễn trực quan mối quan hệ giữa các nhánh với nhau vì nó lưu trữ thông tin một cụm đã được kết nối với cụm nào, khoảng cách, độ tương đồng của chúng là bao nhiêu, cụm hiện tại là lớp con của cụm nào và là lớp cha của cụm nào và sắp xếp các cụm có độ lớn tương đồng cùng mức trong cấu trúc cây, nên Hierarchical clustering sẽ phát hiện tốt các dữ liệu có tính phân cấp.
- Hierarchical clustering chỉ ra được mối quan hệ giữa các cụm nhờ vào biểu diễn trực quan các thông tin đã nói ở cây Dendrogram, giúp người dùng thấy rõ mối quan hệ giữa các cụm bằng cách quan sát cây Dendrogram. Điều này Spectral clustering không làm được do nó không phân chia các cụm bằng cách hợp từng cụm lại mà phân chia các điểm có quan hệ kết nối với nhau vào một số cụm đã quy ước trước và điều chỉnh khi cần thiết.

Khả năng xử lí dữ liệu phức tạp

- Trong một Dendrogram, một điểm không thể thuộc về hai cụm khác nhau trong cùng một thời điểm. Vì thế khi các điểm dữ liệu chồng lấn lên nhau, Hierarchical clustering sẽ không xử lý được vì nó sẽ gặp khó khăn khi hợp nhất hai điểm chồng lấn, nó sẽ có vùng giao nhau nên không thể tính toán được khoảng cách hay độ tương đồng giữa chúng và nếu có nhiều hơn một điểm đè lên điểm đang xét, Hierarchical clustering không thể quyết định được hợp nhất hai điểm nào.
- Trong khi đó, Spectral clustering vẫn có thể phân cụm được với dữ liệu chồng lấn do nó vẫn xét được mối quan hệ kết nối giữa các điểm thông qua việc quy chiếu các điểm qua không gian đặc trưng kể cả có sự chồng lấn dữ liệu.
- Với các dữ liệu phi tuyến có độ nhiễu, Spectral clustering cho kết quả tốt hơn do Hierarchical clustering chỉ dựa vào khoảng cách trực tiếp, độ tương đồng mà không xét đến sự kết nối giữa các cụm với nhau để gom cụm.

Độ phức tạp

- Độ phức tạp của Hierarchical clustering là từ $O(N^2 \log N)$ đến $O(N^3)$:
  - Tính toán ma trận khoảng cách ban đầu là $O(N^2)$.
  - Mỗi lần hợp nhất cụm và tính toán lại khoảng cách giữa hai cụm với nhau tùy vào phương pháp liên kết hai cụm nên mất từ $O(N \log N)$ đến $O(N^2)$ (dùng heap hoặc duyệt qua tất cả các điểm dữ liệu).
  - Có tối đa $n-1$ lần hợp nhất, nên độ phức tạp tổng thể là như trên.
- Với độ phức tạp như vậy, Hierarchical clustering hoàn toàn không phù hợp với dữ liệu có kích thước lớn, trong khi Spectral clustering mặc dù khó áp dụng trực tiếp với dữ liệu nhưng vẫn có thể dùng các biến thể để tăng tốc.

Kết luận

Tóm lại, khi áp dụng với các tập dữ liệu có cấu trúc phức tạp như phi tuyến, phân cấp, **Spectral Clustering** sẽ tốt hơn. Nếu muốn thấy rõ mối quan hệ giữa các cụm dữ liệu thì dùng **Hierarchical Clustering**, lưu ý rằng **Hierarchical Clustering** dễ bị ảnh hưởng khi dữ liệu bị nhiễu. Cả hai thuật toán **Clustering** trên nên cân nhắc sử dụng khi dữ liệu lớn.

5.3 Nên dùng Spectral Clustering khi nào?

Sau khi phân tích ưu nhược điểm của **Spectral Clustering**, chúng ta thấy **Spectral Clustering** có khả năng phân cụm mạnh, nhất là trong trường hợp dữ liệu bị nhiễu, chồng lấn lên nhau hoặc cụm có hình dạng phi tuyến nên **Spectral Clustering** nên được sử dụng khi xử lý các dữ liệu loại này. Tuy nhiên, nên cân nhắc khi sử dụng với các dữ liệu có độ phức tạp cao vì **Spectral Clustering** sẽ tốn nhiều chi phí tính toán.

6 Tổng kết

Tổng kết lại, bài thuyết trình đã giới thiệu cho các bạn về cơ sở lý thuyết, nguyên lý hoạt động của **Spectral Clustering**, chỉ ra được ưu, nhược điểm và so sánh **Spectral Clustering** với một số thuật toán **Clustering** khác. Cuối cùng, chúng ta đã biết được nên dùng **Spectral Clustering** khi nào và ứng dụng thực tế của **Spectral Clustering** trong các lĩnh vực khác nhau.

References

[1] A tutorial on Spectral Clustering written by Ulrike von Luxburg from the Max Planck Institute for Biological Cybernetics.

[2] https://github.com/taldatech/ee046202-unsupervised-learning-data-analysis/blob/master/ee046202_tutorial_12_spectral_clustering.ipynb

[3] https://machinelearningcoban.com/2017/01/01/kmeans/

[4] https://developers.google.com/machine-learning/clustering/clustering-algorithms?hl=vi

[5] https://www.youngwonks.com/blog/spectral-clustering#:~:text=Hierarchical%20clustering%20is%20typically%20used,complex%20or%20non%2Dlinear%20data