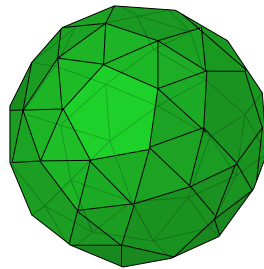


Projects in Mathematics and Applications

ISOMAP

Đình Thuận ^{*}, Phan Chí Thọ [†], Xuân Minh [‡], Khánh Linh [§]

Ngày 17 tháng 8 năm 2021



* Trường Đại học Khoa học Tự nhiên, ĐHQG TP HCM

† Trường Đại học Việt - Đức (VGU)

‡ Trường THPT Chuyên Trần Đại Nghĩa

§ Trường THPT Chuyên Nguyễn Bình Khiêm

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin bày tỏ lòng biết ơn sâu sắc đến Ban Tổ chức Trại hè PiMA vì đã trao cơ hội, tạo điều kiện tốt nhất cho chúng em được học tập, nghiên cứu các vấn đề Toán học và ứng dụng trong suốt hai tuần vừa qua. Những trải nghiệm và kiến thức chúng em tích lũy được có ý nghĩa rất lớn đối với bản thân mỗi thành viên trong nhóm, đây chắc chắn sẽ là hành trang quý báu, nâng bước chúng em trên con đường nghiên cứu khoa học, chinh phục tri thức.

Chúng em xin chân thành cảm ơn các anh chị Mentors đã nhiệt tình giảng dạy, giải đáp những thắc mắc, tận tâm hỗ trợ chúng em trong việc học tập, nghiên cứu. Cảm ơn các anh chị trong BTC đã nỗ lực làm việc để tổ chức những buổi giao lưu, vui chơi, giúp trại sinh chúng em hòa đồng, thêm hiểu biết về lẫn nhau, có những trải nghiệm tốt không kém sinh hoạt tập thể trực tiếp. Cảm ơn các anh chị diễn giả đã cung cấp cho chúng em rất nhiều thông tin bổ ích, mở rộng góc nhìn của chúng em về Data Science, giúp chúng em định hướng rõ ràng hơn về con đường tương lai. Đặc biệt, chúng em xin cảm ơn anh Trần Hoàng Bảo Linh và anh Nguyễn Nguyễn đã theo sát và giúp đỡ nhóm từ lúc bắt đầu cho đến khi hoàn thành dự án, có được bài báo cáo này.

Sau hai tuần tham gia PiMA, chúng em không chỉ thu về được một lượng lớn kiến thức bổ ích về Toán học và Khoa học dữ liệu mà còn được trải nghiệm việc nghiên cứu, thuyết trình dự án, viết báo cáo, làm việc nhóm cùng những người bạn chưa quen biết trước. Nhờ vậy, chúng em được rèn luyện và nâng cao nhiều kỹ năng cần thiết cho tương lai, được giao lưu, kết bạn với các anh chị, các bạn trại sinh tài năng, nhiệt huyết, thân thiện. Hy vọng trong những năm tiếp theo, PiMA có thể tiếp tục phát triển để lan tỏa niềm đam mê Toán học cùng những ứng dụng thú vị, bổ ích của Toán đến các bạn học sinh THPT, giúp các bạn trải nghiệm những điều tốt đẹp, ý nghĩa giống (và hơn) chúng em.

Trong quá trình làm việc, do thời gian chỉnh sửa không nhiều cũng như trình độ của nhóm tác giả có hạn, tuy đã rất cố gắng nhưng sai sót là điều khó tránh khỏi. Chúng em rất mong nhận được góp ý và chia sẻ từ các anh chị Mentors và bạn đọc để nhóm có thể hoàn thiện dự án tốt hơn!

TÓM TẮT NỘI DUNG

Trong Khoa học dữ liệu, giảm chiều dữ liệu là một bài toán rất quan trọng, với mục tiêu biểu diễn các điểm dữ liệu (ban đầu ở chiều cao) ở chiều thấp hơn sao cho một số tính chất tối ưu được thỏa mãn. Trong đó, Isomap là một thuật toán giảm chiều dữ liệu phi tuyến tính khá thông dụng. Ở bài báo cáo này, chúng tôi xin trình bày tổng quan về Isomap, mô tả, giải thích cụ thể các bước của thuật toán cùng việc áp dụng mô hình, các cải tiến của Isomap. Hy vọng sẽ cung cấp nhiều thông tin hữu ích cho bạn đọc!

MỤC LỤC

| | | |
|----------|---|-----------|
| 1 | Tổng quan về Isomap | 1 |
| 1.1 | Giới thiệu | 1 |
| 1.2 | Tầm quan trọng và cách xác định khoảng cách geodesic | 2 |
| 1.3 | Ý tưởng chính | 2 |
| 2 | Thuật toán Isomap | 3 |
| 2.1 | Xây dựng đồ thị vùng lân cận (neighbor graph) | 3 |
| 2.2 | Tính toán khoảng cách đồ thị giữa từng cặp điểm dữ liệu trên toàn dataset . | 6 |
| 2.3 | Giảm chiều dữ liệu - Áp dụng thuật toán cMDS | 6 |
| 3 | Áp dụng mô hình | 11 |
| 3.1 | Thuật toán Isomap cơ bản | 11 |
| 3.2 | Thuật toán Isomap dựa trên radius neighbor | 15 |
| 3.3 | Áp dụng thuật toán Isomap lên Handwriting Digit Dataset | 17 |
| 4 | Kết luận đánh giá | 20 |
| 4.1 | Đánh giá thuật toán Isomap | 20 |
| 4.2 | Cải tiến thuật toán | 20 |
| 5 | Hướng phát triển trong tương lai | 22 |

1 TỔNG QUAN VỀ ISOMAP

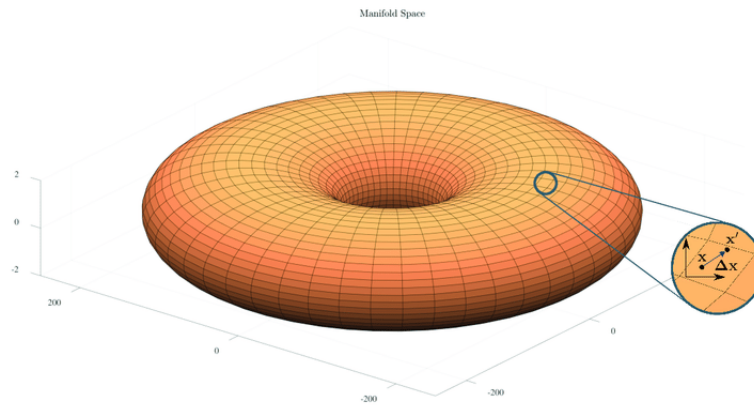
1.1 Giới thiệu

1. Isomap (tên đầy đủ: isometric mapping) là thuật toán giảm chiều dữ liệu phi tuyến tính với mục tiêu bảo toàn tốt nhất khoảng cách geodesic giữa các điểm dữ liệu trên đa tạp.

Thuật toán Isomap mở rộng các kỹ thuật cổ điển của phân tích thành phần chính (PCA) và chia tỷ lệ đa chiều (MDS) cho một lớp đa tạp phi tuyến tính. Thay vì sử dụng khoảng cách Euclidean, Isomap sử dụng khoảng cách trắc địa (Geodesic) được tạo ra bởi một biểu đồ lân cận. Thay đổi này giúp kết hợp cấu trúc đa tạp trong kết quả nhúng chiều thấp.

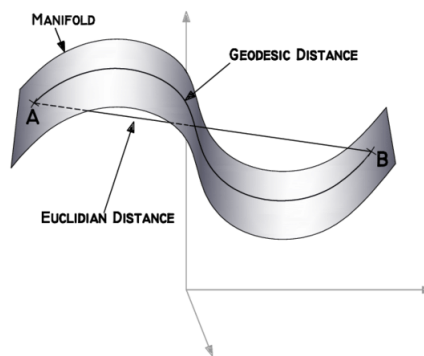
2. Các khái niệm liên quan:

- Đa tạp: một không gian topo n -chiều sao cho với mỗi điểm, ta có thể xác định được một lân cận U xung quanh điểm đó và tồn tại V thuộc không gian Euclidean n -chiều sao cho U đồng phôi với V . Hay nói nôm na, đa tạp là một không gian topo n -chiều mà cục bộ tại gần mỗi điểm, giống với không gian Euclidean n -chiều.



Hình 1: Hình xuyến là đa tạp 2 chiều

- Khoảng cách geodesic (còn gọi là khoảng cách trên đa tạp) là đường đi ngắn nhất giữa 2 điểm trên cùng 1 đa tạp khi đi dọc đa tạp đó.



Hình 2: Khoảng cách Euclidean và Geodesic

1.2 Tầm quan trọng và cách xác định khoảng cách geodesic

1. Việc bảo toàn khoảng cách geodesic quan trọng vì khoảng cách geodesic:

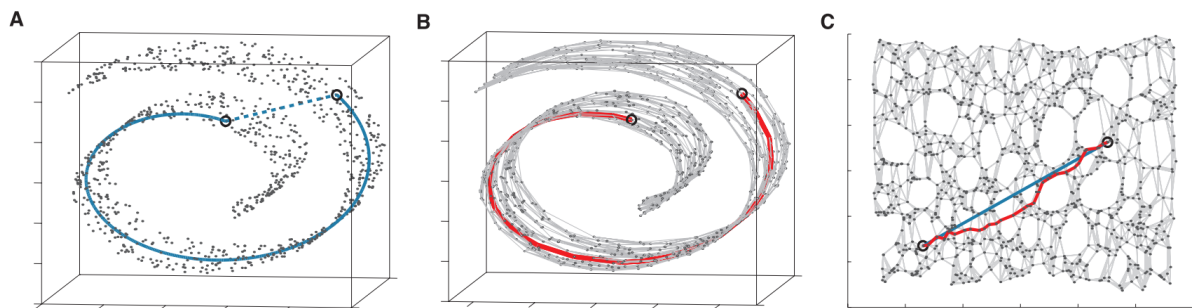
- Nắm bắt chân thực cấu trúc hình học phi tuyến tính tương ứng với chiều cong.
- Phản ánh quá trình chuyển đổi dọc theo đa tạp.

Những điểm nằm cách xa nhau dọc theo bề mặt đa tạp có thể gần nhau theo khoảng cách Euclidean. Vì vậy, bảo toàn khoảng cách Geodesic, thay vì bảo toàn khoảng cách Euclidean, giúp duy trì cấu trúc hình học của đa tạp.

2. Cách xác định khoảng cách geodesic:

Khoảng cách geodesic thường không thể xác định chính xác (trừ khi ta biết trước đa tạp đúng).

Trong thực tế, ta chỉ được cung cấp một tập dữ liệu X được lấy mẫu từ một đa tạp M chưa biết, ta có thể xấp xỉ khoảng cách geodesic bằng đường đi ngắn nhất trên đồ thị lân cận được xây dựng dựa trên tập dữ liệu.



Hình 3: Tính xấp xỉ khoảng cách geodesic bằng đường đi ngắn nhất trên đồ thị lân cận

1.3 Ý tưởng chính

Tìm các điểm trên không gian ít chiều sao cho khoảng cách Euclidean giữa chúng xấp xỉ với khoảng cách đồ thị trên đa tạp mà ta đo được ở không gian nhiều chiều.

Dựa trên ý tưởng chính, ta xác định được hai vấn đề cần giải quyết:

1. Tính khoảng cách Geodesic giữa hai điểm dữ liệu bất kỳ để lập ma trận khoảng cách: Một tập dữ liệu luôn có tính rời rạc, do đó, cơ bản nó là xấp xỉ rời rạc của một đa tạp. Vì vậy, việc tính geodesic distance giữa hai điểm bất kỳ giống như việc xấp xỉ nó bằng cách xây dựng đồ thị trên các điểm dữ liệu ban đầu. Cách thực hiện:

- Nếu hai điểm "gần nhau", thì ta sẽ xấp xỉ geodesic distance giữa hai điểm đó bằng Euclidean distance (Bước 1 - xây dựng đồ thị lân cận, việc "gần nhau" được xác định dựa trên hằng số k chọn trước)
- Thiết lập geodesic distance giữa hai điểm bất kỳ bằng shortest path search algorithms. Tương quan ở đây: geodesic là khoảng cách ngắn nhất trên đa tạp, khi ta rời rạc hóa là đường đi ngắn nhất trên đồ thị. (Bước 2 - tìm đường đi ngắn nhất trên đồ thị)

2. Giảm chiều dữ liệu sao cho khoảng cách geodesic giữa các điểm dữ liệu được bảo toàn tốt nhất. (Bước 3 - Áp dụng cMDS cho ma trận khoảng cách đã tính được)

2 THUẬT TOÁN ISOMAP

Dữ liệu đầu vào:

$$X = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_n]$$

với $\mathbf{x}_i \in \mathbb{R}^q$ và số chiều dữ liệu cần đạt được sau khi giảm chiều là p .

Dữ liệu đầu ra:

$$Y_p = [\mathbf{y}_1 \quad \mathbf{y}_2 \quad \dots \quad \mathbf{y}_n]$$

với $\mathbf{y}_i \in \mathbb{R}^p$ sao cho hàm mất mát

$$\mathcal{L}(Y_p) = \sum_{i=1}^n \sum_{j=1}^n (d_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2$$

đạt giá trị nhỏ nhất.

Lý do lựa chọn hàm mất mát: Mục tiêu của thuật toán là bảo toàn tốt nhất khoảng cách geodesic giữa các điểm dữ liệu. Ta mong muốn khoảng cách euclidean giữa các điểm dữ liệu được biểu diễn ở chiều thấp xấp xỉ khoảng cách geodesic (d_{ij}) đo được ở chiều cao tốt nhất có thể. Tương đương với việc chênh lệch giữa hai khoảng cách này càng nhỏ thì thuật toán bảo toàn càng tốt.

Thuật toán gồm 3 bước chính:

1. Xây dựng đồ thị vùng lân cận (neighbor graph).
2. Tính toán khoảng cách đồ thị giữa từng cặp điểm dữ liệu trên toàn dataset.
3. Giảm chiều dữ liệu - Áp dụng thuật toán cMDS.

Nội dung cụ thể từng bước sẽ được trình bày trong những phần sau.

2.1 Xây dựng đồ thị vùng lân cận (neighbor graph)

Dữ liệu đầu vào:

$$X = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_n]$$

với $\mathbf{x}_i \in \mathbb{R}^q \forall i \in \{1, 2, \dots, n\}$ và $k \in \mathbb{R}$ là 1 siêu tham số biểu thị số điểm gần nhất được nối với 1 điểm.

Dữ liệu đầu ra: $D = \{d_{ij}\} \in \mathbb{R}^{n \times n}$ trong đó d_{ij} là khoảng cách trực tiếp từ i tới j nếu có cạnh nối giữa i và j , nếu không thì $d_{ij} = +\infty$.

Có 3 thuật toán chính để giải quyết bài toán này là thuật toán brute force, thuật toán kd-tree và thuật toán ball tree. Và dưới đây sẽ là mô tả thuật toán brute force và thuật toán kd-tree.

2.1.1 Thuật toán brute force

Thuật toán

$\forall i \in \{1, 2, \dots, n\}$ ta làm các bước sau:

1. Tính khoảng cách từ x_i tới $x_j \forall j \in \{1, 2, \dots, n\}$ và lưu lại các giá trị này trong 1 danh sách.
2. Sắp xếp danh sách này tăng dần.

3. Duyệt qua $k + 1$ điểm x_j gần nhất với x_i , đặt t là khoảng cách từ x_i tới x_j . Ta cập nhật $d_{ij} = t$ và $d_{ji} = t$. (Lưu ý: cập nhật luôn cả d_{ji} là cần thiết vì ta muốn đồ thị của ta là vô hướng để đảm bảo ma trận khoảng cách sau khi chạy thuật toán đường đi ngắn nhất là đối xứng.)

Các cạnh không được cập nhật sẽ được gán $= +\infty$.

Độ phức tạp

Do phải xếp sắp lại n lần và độ phức tạp mỗi lần sắp xếp là $O(n \log n)$, nên độ phức tạp của thuật toán này là: $O(n^2 \log n)$.

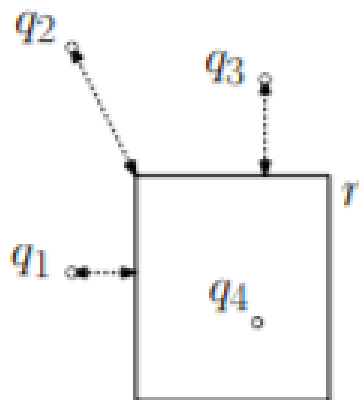
2.1.2 Thuật toán kd-tree

Thuật toán bruteforce tốn nhiều thời gian do phải tính tất cả khoảng cách giữa 2 cặp điểm. Nhưng điều này là không thật sự cần thiết vì nếu 2 điểm ở thật sự xa nhau thì ta không cần tính khoảng cách giữa chúng. Vì vậy, ta sẽ muốn xây dựng 1 cấu trúc dữ liệu để chia dữ liệu của ta thành các vùng. Sau đó ta có thể duyệt nhánh cận qua các điểm dữ liệu để tối thiểu số điểm cần phải duyệt. kd-tree chính là cấu trúc dữ liệu mà ta cần.

Tổng quan về kd-tree

1. Kd-tree là 1 cây nhị phân.
2. Mỗi đỉnh trên cây sẽ chứa 1 điểm dữ liệu. tính chất của điểm dữ liệu này sẽ được trình bày cụ thể hơn tại bước xây dựng cây.
3. Để thuận tiện cho bài toán KNN thì ta sẽ định nghĩa thêm 2 biến p^{low} và p^{high} là vị trí 2 đỉnh đối diện của 1 hình hộp chữ nhật trong không gian R^q . cụ thể, nếu điểm x thuộc hình hộp chữ nhật này thì: $p_i^{low} \leq x_i \leq p_i^{high} \forall i \in \{1, 2, \dots, q\}$. Khi đó ta cũng có khoảng cách từ y là 1 điểm bất kì trong \mathbb{R}^q tới hình hộp chữ nhật này là:

$$\frac{1}{4} \sum_{i=1}^q (|y_i - p_i^{low}| + |y_i - p_i^{high}| - p_i^{high} + p_i^{low})^2$$



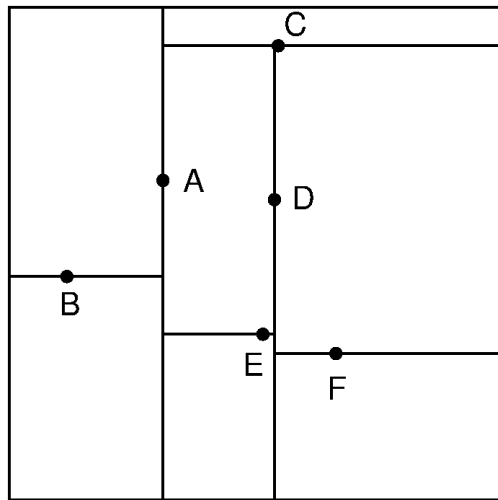
Hình 4: khoảng cách từ các điểm khác nhau tới hình hộp chữ nhật trong R^2

4. Đỉnh ưu tiên: t là 1 điểm đang thực hiện truy vấn trên cây, ta đang ở đỉnh x có tầng là h trên cây. $r = (x \bmod q) + 1$. Nếu $t_r \leq x_r$ thì đỉnh ưu tiên là đỉnh con bên trái, nếu không thì đỉnh ưu tiên là đỉnh con bên phải.

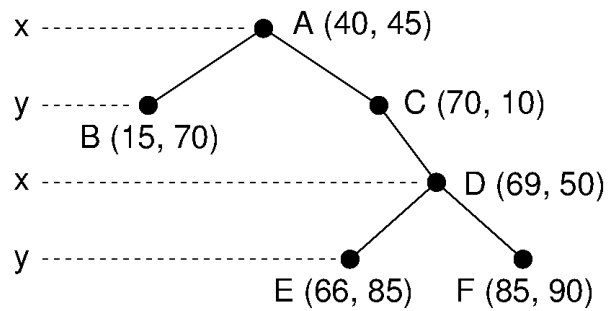
Bước 1: Xây dựng kd-tree từ dữ liệu đầu vào

Thêm lần lượt các điểm dữ liệu x_i vào cây theo các bước sau:

1. Bắt đầu từ đỉnh gốc của cây.
2. Nếu đỉnh hiện tại là NULL thì xuống bước 3, nếu không thì xuống tiếp đỉnh con ưu tiên và lặp lại bước 2.
3. Thêm đỉnh mới tại vị trí này có giá trị là x_i và lưu lại p^{low} , p^{high} là surrounding box của x_i dựa trên p^{low} , p^{high} của đỉnh cha của nó.



(a)



(b)

Hình 5: các điểm dữ liệu và cây tương ứng**Bước 2: Tìm k điểm dữ liệu gần nhất ứng với từng điểm dữ liệu x_i**

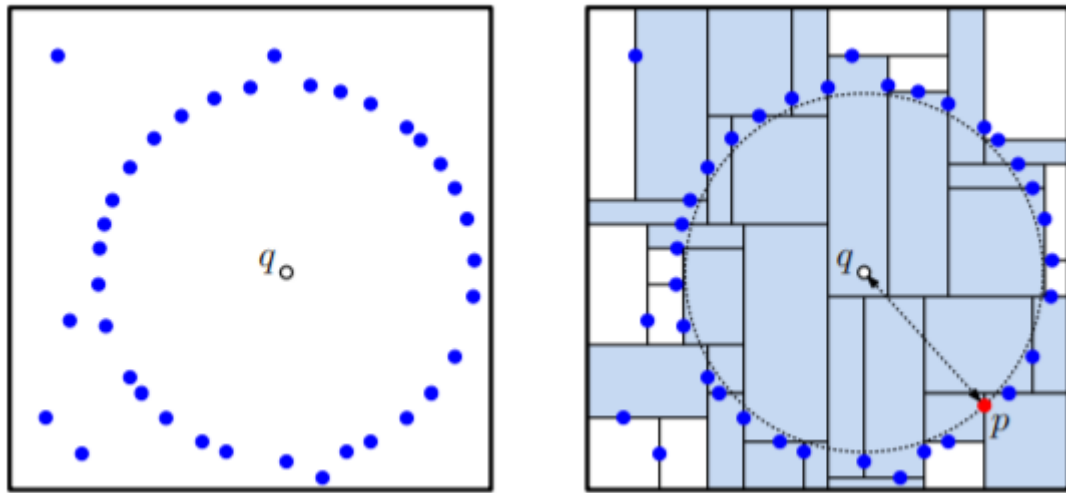
Khởi tạo D (D chứa k điểm dữ liệu gần với x_i nhất).

Tại mỗi đỉnh x_j , ta thực hiện các bước sau:

1. Nếu khoảng cách từ x_i tới hình hộp chữ nhật chứa x_j không gần hơn khoảng cách tới điểm dữ liệu xa thứ k trong D thì ngừng duyệt tiếp các con của x_j .
2. Cập nhật khoảng cách từ x_i tới x_j vào D .
3. Gọi đệ quy xuống đỉnh con ưu tiên của x_i , sau đó gọi đệ quy xuống đỉnh còn lại.

Nhận xét.

Trong 1 số trường hợp, thuật toán không thể tránh khỏi việc phải kiểm tra tất cả đỉnh trên cây. Dưới đây là 1 ví dụ.



Hình 6

Có thể thấy mặc dù p là điểm có vị trí khá xa với q trên cây, nhưng p lại là điểm gần q nhất. Vì vậy, ta có thể sẽ phải duyệt qua hầu hết các điểm để tìm được điểm gần q nhất. Do đó, độ phức tạp của thuật toán tìm kiếm trong trường hợp xấu nhất vẫn là $O(n \log k)$ ($O(n)$ nếu sử dụng fibonacci heap) dẫn đến độ phức tạp của thuật toán này là $(O(n^2 \log k) \ O(n^2))$ nếu sử dụng fibonacci heap). Tuy vậy, trong thực tế (khi cây tương đối cân bằng và các điểm dữ liệu phân bố ngẫu nhiên) thì thuật toán sẽ giảm thiểu được rất nhiều điểm dữ liệu cần kiểm tra. Do vậy mà thuật toán kd-tree vẫn nhanh hơn nhiều so với thuật toán brute force.

2.2 Tính toán khoảng cách đồ thị giữa từng cặp điểm dữ liệu trên toàn dataset

Ở bước này, ta sẽ dùng thuật toán Dijkstra. Cụ thể, ta sẽ bắt đầu bằng việc chọn 1 điểm ngẫu nhiên và hàng chờ các điểm sẽ được xét. Gọi đó là điểm A và hàng chờ Q.

Ta xét độ dài cạnh AA_i với A_i là các điểm có liên kết với A. Sau đó, ta thêm vào Q các điểm A_i kèm khoảng cách đối với A tương ứng theo thứ tự khoảng cách tăng dần.

Cuối cùng ta thực hiện vòng lặp sau:

- Xét điểm đầu tiên trong hàng chờ (gọi điểm này là L), lần lượt xác định khoảng cách từ A đến các điểm có liên kết với L bằng tổng khoảng cách từ A đến L cộng khoảng cách từ L đến điểm đó. Cập nhật khoảng cách đồ thị của các điểm này nếu tổng trên nhỏ hơn chỉ số hiện tại.
- Bổ sung các điểm cùng khoảng cách vào hàng chờ Q sao cho vẫn đảm bảo tính tăng của hàng chờ.
- Loại điểm L ra khỏi hàng chờ.

Thuật toán sẽ dừng lại khi tất cả các điểm trong dataset đã được xét.

2.3 Giảm chiều dữ liệu - Áp dụng thuật toán cMDS

Sau hai bước trên, ta có được ma trận chứa thông tin về các geodesic distances giữa các điểm dữ liệu: $D = \{d_{ij}\}_{1 \leq i, j \leq n}$. Mục tiêu của chúng ta là giảm chiều của tập dữ liệu $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ thành các điểm dữ liệu $\mathbf{y}_1, \dots, \mathbf{y}_n$ với chiều p bé hơn, sao cho tất cả các khoảng cách

$\|\mathbf{y}_i - \mathbf{y}_j\|$ là xấp xỉ khoảng cách geodesic d_{ij} giữa \mathbf{x}_i và \mathbf{x}_j tốt nhất có thể. Do đó, ta thiết lập hàm mục tiêu cần tối ưu

$$\mathcal{L}(Y_p) = \sum_{i=1}^n \sum_{j=1}^n (d_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2$$

với

$$Y_p = [\mathbf{y}_1 \quad \mathbf{y}_2 \quad \cdots \quad \mathbf{y}_n]$$

là ma trận chứa các điểm dữ liệu đã được giảm chiều, $\mathbf{y}_i \in \mathbb{R}^p$ với mọi $1 \leq i \leq n$. Để giải quyết bài toán tối ưu trên, ta sử dụng thuật toán Classical Multidimensional Scaling (cMDS).

2.3.1 Các bước áp dụng cMDS

Ta áp dụng thuật toán Classical Multidimensional Scaling (cMDS) với ma trận D , gồm 4 bước như sau:

1. Lập ma trận bình phương khoảng cách $D_{(2)} = \{d_{ij}^2\}_{1 \leq i, j \leq n}$.
2. Áp dụng double centering $K = -\frac{1}{2}HD_{(2)}H$, với H là centering matrix $H = I_n - \frac{1}{n}J_n$, trong đó $I_n \in \mathbb{R}^{n \times n}$ là ma trận đơn vị và $J_n \in \mathbb{R}^{n \times n}$ là ma trận có tất cả các phần tử là các số 1.
3. Xác định các trị riêng và hệ vector riêng trực chuẩn tương ứng của K và thực hiện chéo hóa $K = V\Lambda V^T$, trong đó:
 - Λ là ma trận đường chéo với các phần tử là n trị riêng $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ của K .
 - V là ma trận chứa các cột là hệ vector riêng trực chuẩn

$$V = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n]$$

trong đó $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ lần lượt là vector riêng (trực chuẩn) tương ứng với các trị riêng $\lambda_1, \lambda_2, \dots, \lambda_n$

4. Chọn p trị riêng lớn nhất và các vector riêng tương ứng, ta được ma trận output

$$Y_p = \Lambda_p^{\frac{1}{2}} V_p^T$$

2.3.2 Chứng minh thuật toán

1. Một số kiến thức, tính chất quan trọng

Định nghĩa 2.1. (Cơ sở). Cho không gian vector V và U là một không gian con của V . Một cơ sở (basis) của U là một tập hợp S gồm các vectors (có thể vô hạn) thuộc U thỏa mãn:

- Các vectors trong S độc lập tuyến tính với nhau.
- Tập sinh bởi các vectors này cũng là U : $\text{Span}(S) = U$

Định nghĩa 2.2. (Cơ sở trực chuẩn) Một tập hợp vectors tạo thành một tập hợp trực chuẩn nếu tất cả vectors trong tập hợp trực giao (hay vuông góc) và tất cả đều có độ dài bằng độ dài đơn vị (bằng 1). Một tập hợp trực chuẩn tạo thành một cơ sở được gọi là cơ sở trực chuẩn.

Định nghĩa 2.3. (Ma trận trực giao) Ma trận $A \in \mathbb{R}^{n \times n}$ được gọi là trực giao nếu:

$$A^T A = I_n$$

Nhận xét 2.4. Các dòng (hoặc cột) của một ma trận trực giao tương ứng với một cơ sở trực chuẩn của \mathbb{R}^n

Tính chất 2.5. Nếu ma trận $A \in \mathbb{R}^{n \times n}$ là ma trận trực giao thì: $A^T A = A A^T = I_n$ hay A^T là ma trận nghịch đảo của A.

Định nghĩa 2.6. (Ma trận đối xứng) $A \in \mathbb{R}^{n \times n}$ được gọi là ma trận đối xứng nếu:

$$A^T = A$$

Định lý 2.7. (Định lý về phổ - Spectral Theorem) Ma trận đối xứng cấp n có n trị riêng thực.

Tính chất 2.8. Ma trận đối xứng cấp n có n vectors riêng trực chuẩn.

Chứng minh: Gọi v_i, v_j là 2 vectors riêng chuẩn (độ dài bằng 1) ứng với 2 trị riêng khác nhau λ_i, λ_j ($1 \leq i < j \leq n$) của ma trận A.

$$\text{Ta có: } \langle v_i, A v_j \rangle = v_i^T A v_j = v_j^T A^T v_i = v_j^T A v_i = \langle v_j, A v_i \rangle$$

$$\text{Mà } \langle v_i, A v_j \rangle = \langle v_i, \lambda_j v_j \rangle = \lambda_j \langle v_i, v_j \rangle ; \langle v_j, A v_i \rangle = \langle v_j, \lambda_i v_i \rangle = \lambda_i \langle v_i, v_j \rangle$$

Vì $\lambda_i \neq \lambda_j$ nên $\langle v_i, v_j \rangle = 0$ hay v_i, v_j trực giao $\forall 1 \leq i < j \leq n$

Tính chất 2.9. Ma trận A đối xứng cấp n có thể biểu diễn dưới dạng:

$$A = V \Lambda V^T \quad (1)$$

với $\Lambda \in \mathbb{R}^{n \times n}$ là ma trận đường chéo gồm các phần tử trên đường chéo là các trị riêng của A, $V \in \mathbb{R}^{n \times n}$ là ma trận gồm các cột là các vector riêng (trực chuẩn) tương ứng.

Chứng minh: Ta có $AV = V\Lambda \iff AVV^T = V\Lambda V^T \iff A = V\Lambda V^T$ ($VV^T = I_n$ do V là ma trận trực giao)

Định nghĩa 2.10. Ma trận $A \in \mathbb{R}^{m \times n}$ có $\text{rank}(A) = r$, singular value decomposition (SVD) của A là $A = U \Sigma V^T$, trong đó:

- $\Sigma \in \mathbb{R}^{r \times r}$ là ma trận đường chéo gồm các single value: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$ với $\sigma_1^2, \sigma_2^2, \dots, \sigma_r^2$ là toàn bộ trị riêng khác 0 của $A^T A$
- $V \in \mathbb{R}^{n \times r}$ là ma trận gồm các cột là các vector riêng \mathbf{v}_i ($\|\mathbf{v}_i\| = 1$) tương ứng với trị riêng σ_i^2
- $U \in \mathbb{R}^{m \times r}$ là ma trận gồm các vector cột \mathbf{u}_i ($\|\mathbf{u}_i\| = 1$) thỏa $A \mathbf{v}_i = \sigma_i \mathbf{u}_i$

Tính chất 2.11. Biểu diễn (1) là SVD của ma trận $A \in \mathbb{R}^{n \times n}$.

$$\text{Chứng minh: } A = V \Lambda V^T \iff A^T = (V \Lambda V^T)^T = (\Lambda V^T)^T V^T = V \Lambda V^T$$

$$\implies A^T A = V \Lambda V^T V \Lambda V^T = V \Lambda^2 V^T \implies (A^T A) V = V \Lambda^2 V^T V = V \Lambda^2$$

Từ đó, có Λ^2 là ma trận đường chéo gồm các phần tử trên đường chéo là các trị riêng của $A^T A$, V là ma trận gồm các cột là các vector riêng (trực chuẩn) tương ứng.

Lại có: $AV = \Lambda V$ nên biểu diễn (1) chính là SVD của A. (Đpcm)

Định nghĩa 2.12. (Rank) Hạng của ma trận A là số chiều của không gian dòng của ma trận đó, ký hiệu là $\text{rank}(A)$ hoặc $r(A)$

Tính chất 2.13. $\text{rank}(A^T A) = \text{rank}(A^T) = \text{rank}(A)$

2. Chứng minh thuật toán

- Giả sử tồn tại

$$Y = [\mathbf{y}_1 \quad \mathbf{y}_2 \quad \cdots \quad \mathbf{y}_n]$$

$Y \in \mathbb{R}^{q \times n}$ sao cho $\|y_i - y_j\| = d_{ij} \forall i, j \in \{1, \dots, n\}$. Nếu Y thỏa thì:

$$Y^* = [\mathbf{y}_1 + c \quad \mathbf{y}_2 + c \quad \cdots \quad \mathbf{y}_n + c]$$

cũng thỏa $\forall c \in \mathbb{R}^q$. Không mất tính tổng quát, giả sử: $\sum_{i=1}^n (\mathbf{y}_i) = 0$

Xét $K = Y^T Y \implies K_{ij} = \mathbf{y}_i^T \mathbf{y}_j$

Ta có: $d_{ij}^2 = (\|\mathbf{y}_i - \mathbf{y}_j\|)^2 = (\mathbf{y}_i - \mathbf{y}_j)^T (\mathbf{y}_i - \mathbf{y}_j) = \mathbf{y}_i^T \mathbf{y}_i + \mathbf{y}_j^T \mathbf{y}_j - 2\mathbf{y}_i^T \mathbf{y}_j$

$$d_{ij}^2 = k_{ii} + k_{jj} - 2k_{ij} \quad (2)$$

Vì $\sum_{i=1}^n (y_i) = 0$ nên: $\sum_{i=1}^n k_{ij} = \sum_{i=1}^n y_i^T y_j = \sum_{i=1}^n \sum_{k=1}^q y_{ik} y_{jk} = \sum_{k=1}^q y_{jk} \sum_{i=1}^n y_{ik} = 0$, tức là tổng mỗi dòng, mỗi cột của K đều bằng 0.

Đặt $T = \text{trace}(B) = \sum_{i=1}^n k_{ii}$, ta có:

$$\begin{aligned} \sum_{i=1}^n d_{ij}^2 &= \sum_{i=1}^n k_{ii} + \sum_{j=1}^n k_{jj} - 2 \sum_{i=1}^n k_{ij} = T + nk_{jj} \\ \sum_{j=1}^n d_{ij}^2 &= T + nk_{ii} \\ \sum_{j=1}^n \sum_{i=1}^n d_{ij}^2 &= \sum_{j=1}^n (T + nk_{jj}) = nT + n \sum_{j=1}^n k_{jj} = 2nT \end{aligned} \quad (3)$$

Thay (3) vào (2) thu được:

$$k_{ij} = -\frac{1}{2} \left(d_{ij}^2 - \frac{1}{n} \sum_{i=1}^n d_{ij}^2 - \frac{1}{n} \sum_{j=1}^n d_{ij}^2 + \frac{1}{n^2} \sum_{j=1}^n \sum_{i=1}^n d_{ij}^2 \right)$$

hay $K = -\frac{1}{2} H D_{(2)} H$, trong đó: H là centering matrix: $H = I_n - \frac{1}{n} J_n$, J_n là ma trận một $n \times n$ và $D_{(2)} = \{d_{ij}^2\}$

Dễ thấy K là ma trận đối xứng cấp n . Vì vậy, tìm được n giá trị riêng $\lambda_1, \dots, \lambda_n$ và n vector riêng tương ứng: $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ trực chuẩn.

Đặt:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

$$V = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n]$$

V là ma trận trực giao, suy ra $V^T V = V V^T = I_n$

Ta có: $KV = V\Lambda \iff KVV^T = V\Lambda V^T \iff K = V\Lambda V^T (= Y^T Y)$

- Giảm chiều dữ liệu: Tìm Y_p sao cho:

$$Y_p = \underset{Y_p}{\operatorname{argmin}} \sum_{1 \leq i, j \leq n} (d_{ij} - \|y_i - y_j\|)^2 = \underset{Y_p}{\operatorname{argmin}} \|Y_p^T Y_p - K\|_F.$$

Lại có $\operatorname{rank}(Y_p^T Y_p) = \operatorname{rank}(Y_p) \leq p$. Áp dụng định lí:

Định lí (Low-rank approximation): Cho ma trận $A \in \mathbb{R}^{m \times n}$, $\operatorname{rank}(A) = r$, $k \leq r$

$$A_k = \underset{B}{\operatorname{argmin}} \{\|A - B\|_F^2 \mid \operatorname{rank}(B) \leq k\} \text{ với: } A_k = U_k \Sigma_k V_k^T$$

(SVD của A là: $A = U \Sigma V^T$)

Vì vậy, để bài toán tối ưu thì:

$$Y_p^T Y_p = V_p \Lambda_p V_p^T = V_p \Lambda_p^{\frac{1}{2}} \Lambda_p^{\frac{1}{2}} V_p^T \text{ nhận } Y_p = \Lambda_p^{\frac{1}{2}} V_p^T \text{ là nghiệm.}$$

Trong đó Λ_p là ma trận đường chéo gồm p trị riêng lớn nhất, V_p là ma trận gồm các cột là các vector riêng tương ứng. (Thuật toán đã được chứng minh)

2.3.3 Trường hợp không có nghiệm tối ưu

Tuy nhiên, ma trận K (Kernel matrix) không phải bao giờ cũng là ma trận bán xác định dương, nếu K có trị riêng âm thì $\Lambda_p^{\frac{1}{2}}$ không xác định thực.

Giải pháp: (Robust Kernel Isomap) Chuyển đổi ma trận K thành Mercer kernel matrix K' (ma trận này bán xác định dương) bằng cách sử dụng phương pháp constant-shifting.

Các bước thực hiện như sau:

Ký hiệu: $K(D_{(2)}) = -\frac{1}{2} H D_{(2)} H$

Tính trị riêng lớn nhất c^* của ma trận:

$$\begin{bmatrix} 0 & 2K(D_{(2)}) \\ -I & -4K(D) \end{bmatrix}$$

và thiết lập ma trận Mercer kernel matrix K' bằng công thức:

$$K' = K(D_{(2)}) + 2cK(D) + \frac{1}{2}c^2H$$

Ma trận K' bán xác định dương $\forall c \geq c^*$, sử dụng ma trận K' thay cho ma trận K và tiếp tục thực hiện các bước của bài toán tối ưu.

(Phần chứng minh ma trận K' bán xác định dương, bạn đọc có thể tham khảo tại: [2])

3 ÁP DỤNG MÔ HÌNH

3.1 Thuật toán Isomap cơ bản

Ta sẽ tiến hành áp dụng thuật toán Isomap để xem output của thuật toán phụ thuộc vào k như thế nào và giải thích lý do tại sao lại xảy ra trường hợp như vậy.

Ta sẽ xét các trường hợp ở dataset dưới đây: Swiss Roll với 1500 điểm dữ liệu, Swiss Roll với 500 điểm dữ liệu và S Curve với 1500 điểm dữ liệu

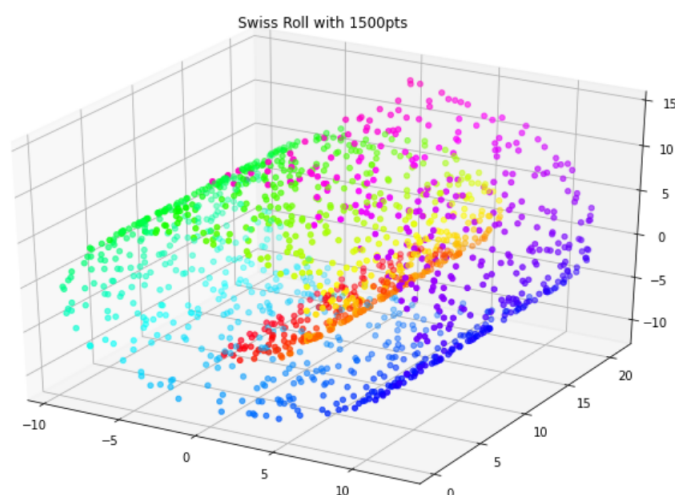
3.1.1 Tập dữ liệu *Swiss Roll* với 1500 điểm dữ liệu

```

1  import sklearn
2  from sklearn import datasets
3  import matplotlib.pyplot as plt
4  import numpy as np
5
6  num = 1500
7  X, t = sklearn.datasets.make_swiss_roll(n_samples=num, random_state=1)
8  xdata = X[:,0]
9  ydata = X[:,1]
10 zdata = X[:,2]
11
12 fig = plt.figure(figsize= (12,8))
13 ax = fig.add_subplot(projection='3d')
14 ax.scatter(xs=xdata, ys=ydata, zs=zdata,c=t, cmap=plt.cm.gist_rainbow)
15 plt.title('Swiss Roll with ' + str(num) + 'pts')
16 plt.show()

```

Sau khi chạy các dòng code ở trên ta khởi tạo được tập dữ liệu Swiss Roll với 1500 điểm dữ liệu dưới dạng 3d.



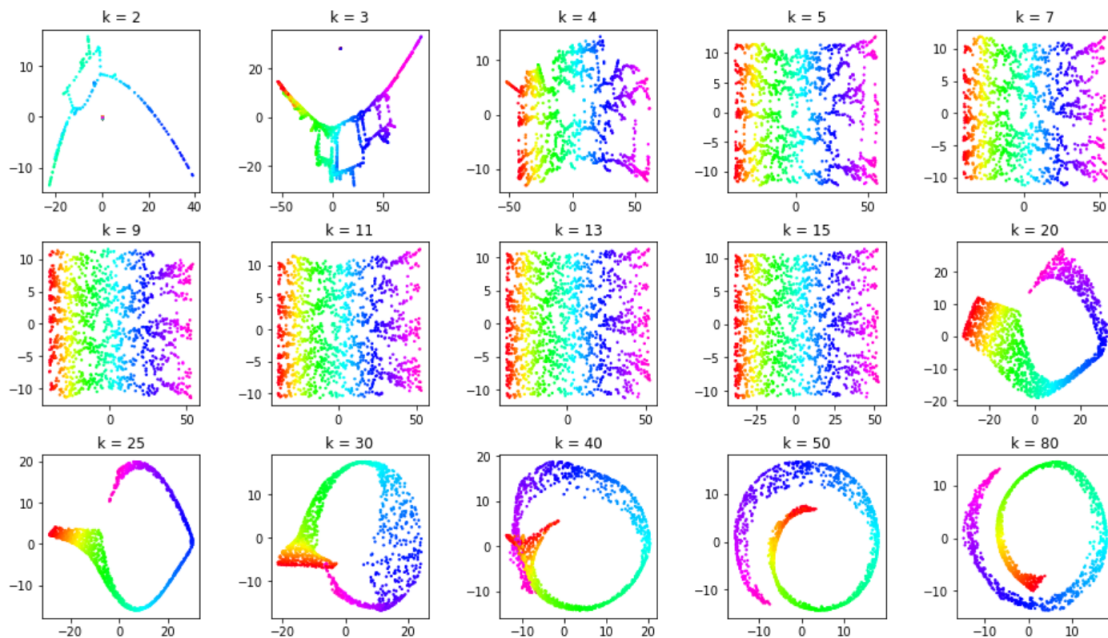
Hình 7: Tập dữ liệu Swiss Roll với 1500 điểm dữ liệu

Tiếp theo ta sẽ chạy thuật toán Isomap với từng giá trị k cho trước ở dưới đây và giảm xuống 2 chiều để đánh giá thuật toán. Sau khi chạy các dòng code ở dưới ta được hình sau và rút ra vài nhận xét:

```

1 k = [2, 3, 4, 5, 7, 9, 11, 13, 15, 20, 25, 30, 40, 50, 80]
2 f = 1
3 fig = plt.figure(figsize = (16,9))
4 fig.subplots_adjust(hspace = 0.3, wspace = 0.45)
5
6 for i in k:
7     ax2=fig.add_subplot(3,5,f)
8     embedding = Isomap(n_neighbors=i)
9     X_transformed = embedding.fit_transform(X[:num])
10    ax2.scatter(x = X_transformed[:,0], y = X_transformed[:,1], c = t,
11               cmap=plt.cm.gist_rainbow, s=2)
12    plt.title('k = ' + str(i))
13    f = f + 1
14 plt.show()

```



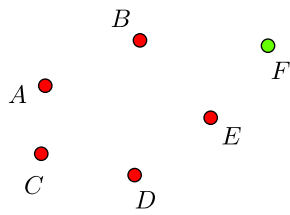
Hình 8: Tập dữ liệu Swiss Roll sau khi giảm chiều với số k tương ứng

Nhận xét 1: Thuật toán chạy tối ưu với giá trị **k** nằm trong khoảng nhất định, $k \in [4, 15]$

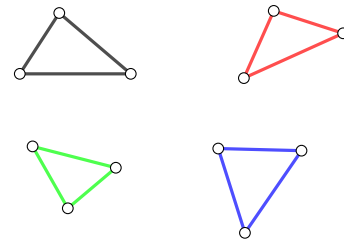
Nhận xét 2: Nếu **k** quá nhỏ ($k = 2$ hoặc $k = 3$), các điểm sau khi giảm chiều nằm san sát nhau và uốn cong thành hình vòng cung và không cho thấy thông tin gốc (1).

Hơn nữa ở $k = 2$ đồ thị không thể hiện được tất cả các màu ảnh. (2)

- **Lý giải (1):** Xét một manifold 2 chiều gồm 6 điểm ABCDEF trong dataset trên (ở hình 3), với $k = 3$ thì ta có thể thấy khoảng cách geodesic $AF = \min \{AB + BE + EF, AC + CD + DE + EF\}$. Nhưng với $k = 5$ thì khoảng cách geodesic $AF = AE + EF$, ngắn hơn và chính xác hơn so với $k = 3$
- **Lý giải (2):** Khi số lượng neighbor quá ít, sẽ xảy ra một tình trạng là các điểm sẽ không liên kết được với nhau (như hình 4). Điều đó dẫn đến việc thuật toán sẽ không xây dựng được đồ thị lân cận chính xác và thậm chí không tính được khoảng cách manifold của tất cả các điểm, do có vài điểm không được kết nối với nhau.



Hình 9: Khoảng cách giữa 2 điểm không tối ưu



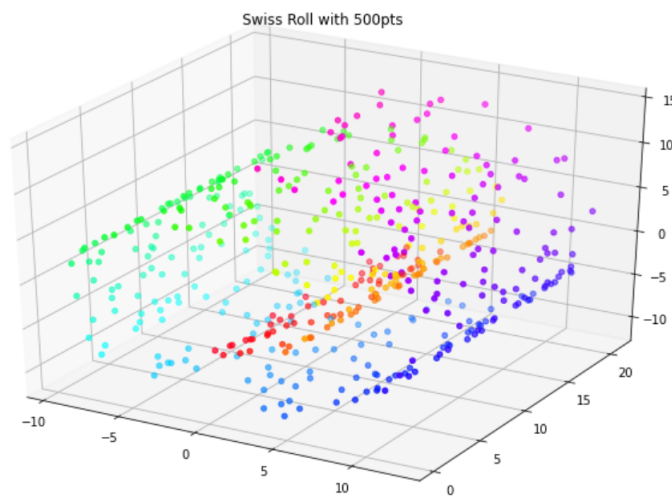
Hình 10: Các điểm bị rời rạc

Nhận xét 3: Nếu chọn **k quá lớn ($k > 20$)**, các hình không còn thể hiện đúng khoảng cách so với hình gốc nữa, và dần bị uốn cong lại giống như chiếu dataset gốc lên 1 mặt phẳng

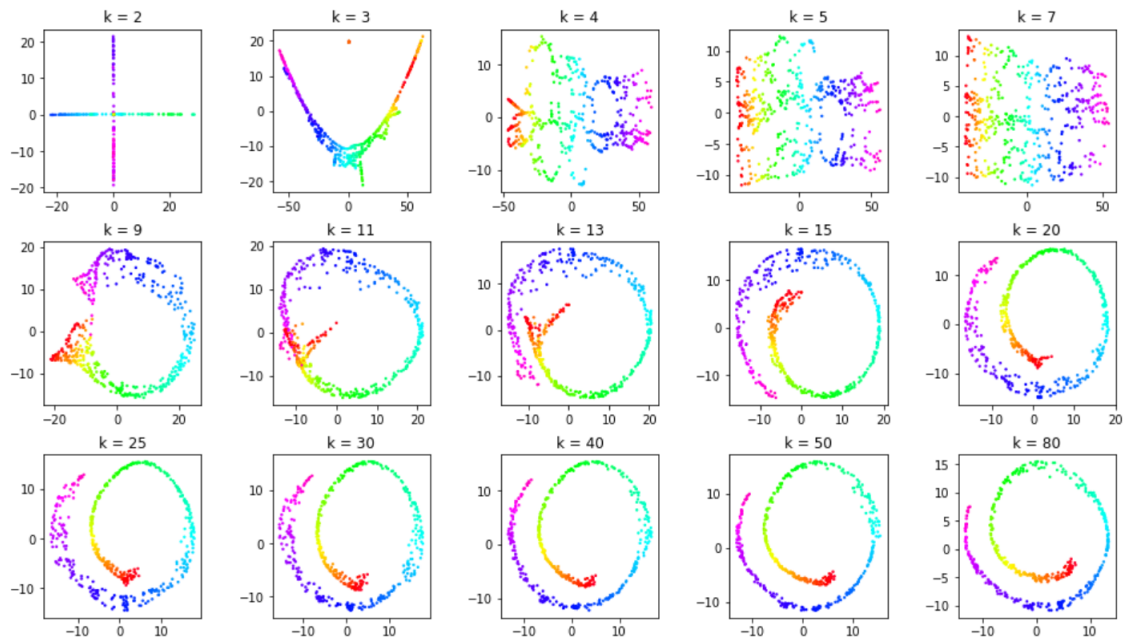
- **Lý giải:** Khi số **k** quá lớn sẽ xảy ra hiện tượng các điểm sẽ tìm các điểm lân cận khác nằm ngoài manifold. Điều đó sẽ khiến cho khoảng cách giữa 2 điểm thuộc manifold khác nhau trở thành euclid, làm mất đi ý nghĩa ban đầu của thuật toán isomap.

3.1.2 Tập dữ liệu Swiss Roll với 500 điểm dữ liệu

Ta sẽ khảo sát cùng tập dữ liệu nhưng khác số lượng điểm để xem sự ảnh hưởng của giá trị **k** lên kết quả. Thay đổi biến `num = 500` ở đoạn code trên ta được hình như sau:



Hình 11: Tập dữ liệu Swiss Roll sau khi giảm chiều với số k tương ứng



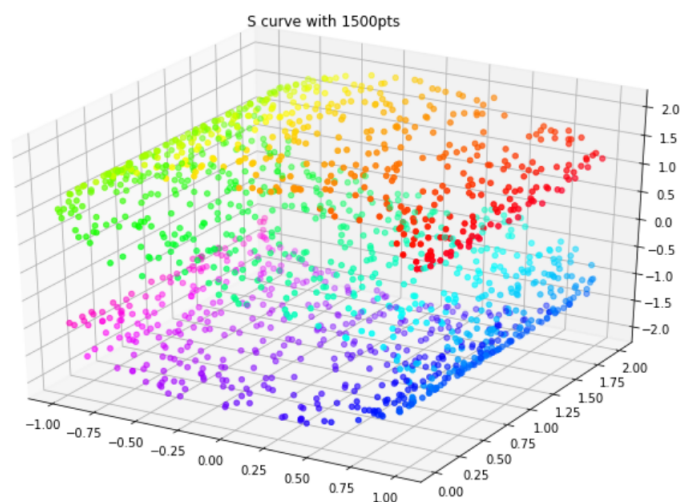
Hình 12: Tập dữ liệu 500 điểm Swiss Roll sau khi giảm chiều với số k tương ứng

Nhận xét: Khoảng k để thuật toán chạy tối ưu phụ thuộc vào số lượng data.

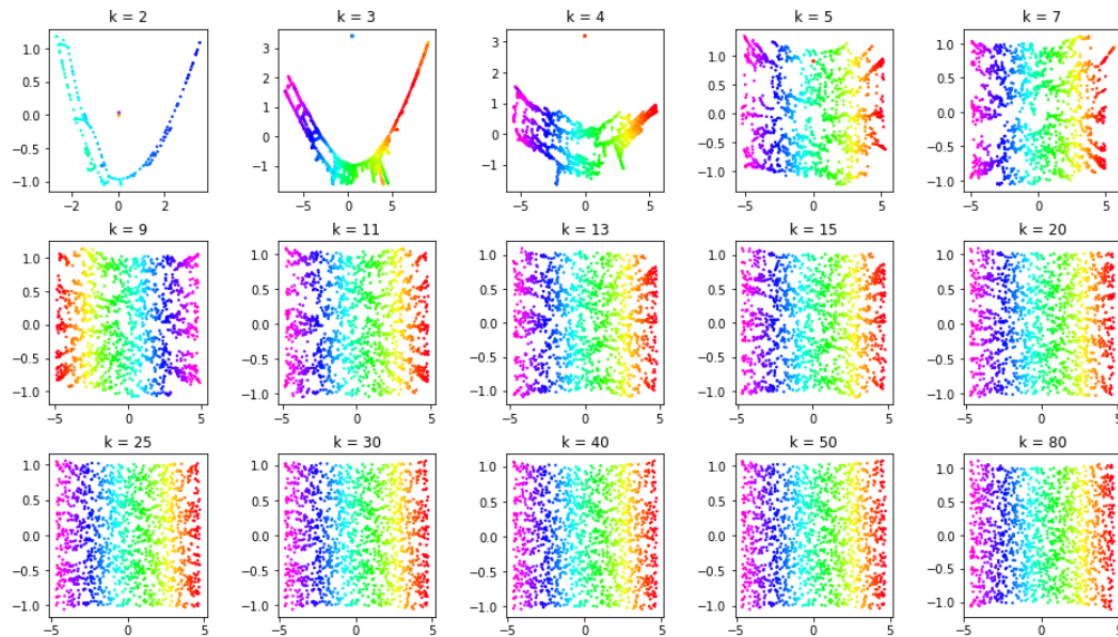
- **Lý giải:** Số lượng dataset ít thì mật độ các điểm cũng sẽ giảm dần và dẫn đến khoảng k phù hợp cũng sẽ giảm đi.

3.1.3 Tập dữ liệu S Curve với 1500 điểm dữ liệu

Ta cũng thực hiện thuật toán Isomap với số k cho trước để đánh giá sự ảnh hưởng của k lên thuật toán.



Hình 13: Tập dữ liệu S Curve với 1500 điểm dữ liệu



Hình 14: Tập dữ liệu S Curve sau khi giảm chiều với số k tương ứng

Sau khi chạy thuật toán xong ta được kết quả bất ngờ: **k = 80** thuật toán vẫn chạy cho ra kết quả rất tốt, mặc dù trường hợp **k > 20** với Swiss Roll với cùng số lượng data thì thuật toán đã không còn hiệu quả

- **Lý giải:** Do mật độ phân bố điểm cùng với cấu trúc hình học giữa dataset Swiss Roll và S Curve khác nhau. Điều đó dẫn tới việc khoảng giá trị **k** tương ứng để thuật toán chạy tốt cũng khác nhau

3.2 Thuật toán Isomap dựa trên radius neighbor

Nhận thấy rằng trong dataset như Swiss Roll sẽ có khu tập trung nhiều điểm nhưng sẽ có khu lại thưa thớt, việc chọn **k** sẽ không giải quyết được tình huống này, đặc biệt là với dataset ít dữ liệu thì việc thưa thớt sẽ xảy ra nhiều hơn. Qua việc xây dựng đồ thị lân cận bằng **k** nearest neighbor với 500 điểm Swiss Roll trên ta thấy được đồ thị chứa rất nhiều lỗ. Nhóm em đã tìm hiểu và đưa ra giải pháp đó bằng cách xây dựng đồ thị neighbor bằng **radius** thay vì chọn **k**.

Thuật toán này vẫn giống thuật toán gốc chỉ khác ở bước xây dựng đồ thị lân cận. Những điểm nào nằm trong **radius** của một điểm sẽ được coi là **neighbor** của điểm đó.

Ta chạy thuật toán đó với 500 điểm dữ liệu Swiss Roll với bán kính cho trước và được kết quả sau:

```

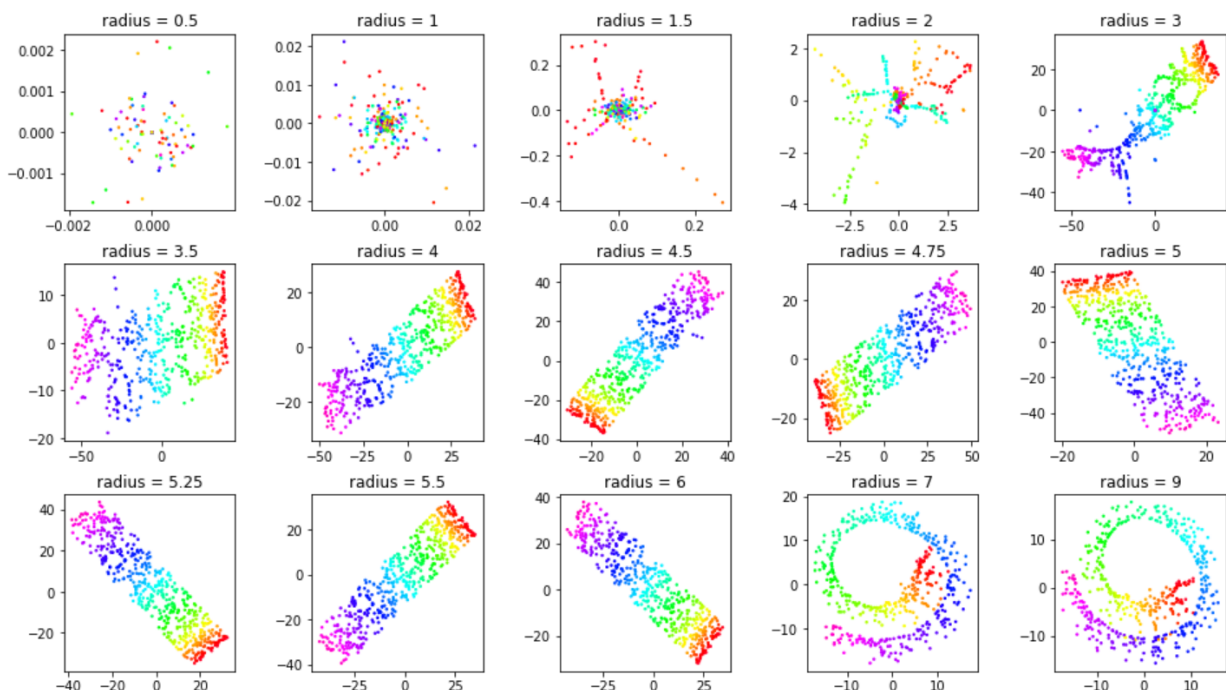
1  # Cải biến của isomap, sử dụng radius để liên kết các neighbor
2  import sklearn
3  from sklearn import datasets
4  import matplotlib.pyplot as plt
5  import numpy as np
6
7  num = 500
8  X, t = sklearn.datasets.make_swiss_roll(n_samples=num, random_state=1)
9  xdata = X[:, 0]

```

```

10 ydata = X[:,1]
11 zdata = X[:,2]
12
13 k = [0.5, 1, 1.5, 2, 3, 3.5, 4, 4.5, 4.75, 5, 5.25, 5.5, 6, 7, 9]
14 f = 1
15 fig = plt.figure(figsize = (16,9))
16 fig.subplots_adjust(hspace = 0.3, wspace = 0.45)
17
18 for i in k:
19     ax2=fig.add_subplot(3,5,f)
20     X_e = radius_neighbors_graph(X, radius=i, mode = 'distance')
21     X_path = sklearn.utils.graph_shortest_path.graph_shortest_path
22                (X_e, directed = False)
23
24     embedding = MDS(n_components=2,dissimilarity='precomputed')
25     X_radius = embedding.fit_transform(X_path[:num])
26     ax2.scatter(x=X_radius[:,0], y=X_radius[:,1],c=t,
27                cmap=plt.cm.gist_rainbow, s=2)
28     plt.title('radius = ' + str(i))
29     f = f+1
30 plt.show()

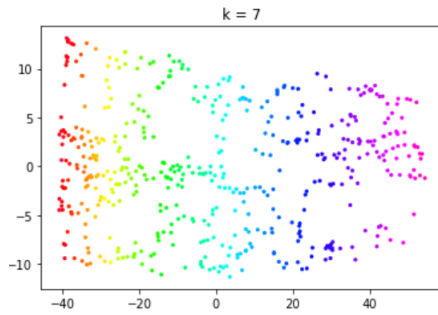
```



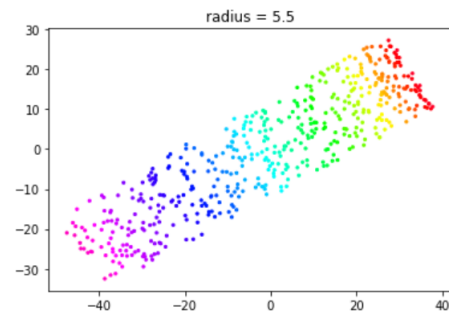
Hình 15: Tập dữ liệu Swiss Roll 500 điểm sau khi giảm chiều với radius tương ứng

Ta có thể thấy được thuật toán chạy tương đối tốt so với việc chạy k-neighbor ở trên, dữ liệu sau khi giảm chiều tương đối đều nhau và không sinh ra nhiều lỗ trống như thuật toán k-neighbor.

Dưới đây là 2 bức ảnh giữa 2 thuật toán qua tham số được cho là tương đối tối ưu.



Hình 16: Giảm chiều bằng KNN



Hình 17: Giảm chiều bằng Radius neighbor

Cả 2 thuật toán đều tốt trong việc bảo toàn khoảng cách geodesic, nhưng ở Radius neighbor các điểm được phân bố đều hơn và ít lỗi hơn so với KNN.

3.3 Áp dụng thuật toán Isomap lên Handwriting Digit Dataset

Nhận thấy việc giảm chiều của thuật toán Isomap có thể giúp ta thấy được các **feature ẩn** sau khi giảm chiều, nên nhóm em sẽ ứng dụng Isomap lên **Handwriting Digit Dataset** và nhận xét các feature nổi bật sau khi giảm chiều bằng Isomap.

Việc thực hiện đó sẽ được tóm tắt như sau:

1. Nhập dataset của Mnist
2. Lọc ra data số cần phân tích (ở đây nhóm em chọn Lucky 7)
3. Thực hiện Isomap lên data đó và plot lên 2d
4. Chọn random các điểm ở trong plot đó
5. Hiển thị những điểm random trong plot đó rồi phân tích feature

Dưới đây sẽ là phần code cụ thể

```

1  # Import các thư viện
2  import sklearn
3  import matplotlib.pyplot as plt
4  import numpy as np
5  from sklearn.manifold import Isomap
6  from keras.datasets import mnist
7
8  # nhập Handwriting dataset từ Mnist
9  (train_X, train_y), (test_X, test_y) = mnist.load_data()
10 train_Xnew = train_X.reshape(60000, 784)
11 data_num7 = train_Xnew[train_y == 7]
12 data_num7_iso = Isomap(n_neighbors=10, n_components=2).fit_transform(data_num7)
13
14 # Chọn random các điểm trong dataset đó để phân tích
15 def find_landmarks(Y, n, m):
16     xr = np.linspace(np.min(Y[:, 0]), np.max(Y[:, 0]), n)
17     yr = np.linspace(np.min(Y[:, 1]), np.max(Y[:, 1]), m)
18     xg, yg = np.meshgrid(xr, yr)
19

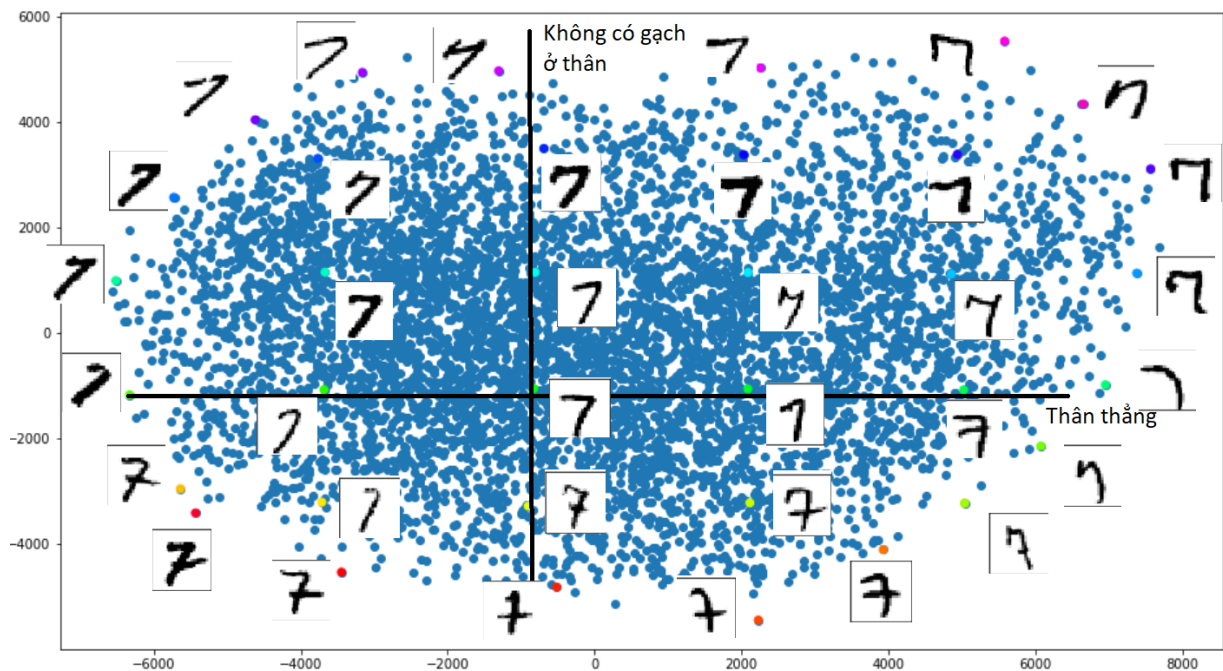
```

```

20     idx = [0]*(n*m)
21
22     for i, x, y in zip(range(n*m), xg.flatten(), yg.flatten()):
23         idx[i] = int(np.sum(np.abs(Y-np.array([x,y]))**2, axis = -1).argmin())
24
25     return idx
26
27     # Plot các điểm đó lên đồ thị
28     landmarks = find_landmarks(data_num7_iso, 6, 6)
29     t = [2*i +1 for i in range(36)]
30     fig = plt.figure(figsize = (16,9))
31     plt.scatter(data_num7_iso[:,0], data_num7_iso[:,1] )
32     plt.scatter(data_num7_iso[landmarks, 0], data_num7_iso[landmarks, 1], c= t,
33                 cmap=plt.cm.gist_rainbow)
34
35     # hiển thị chữ viết tay của các điểm random đó
36     fig = plt.figure(figsize = (7,7))
37     for i in range(len(landmarks)):
38         ax = fig.add_subplot(7,7, i + 1)
39         imgplot = ax.imshow(np.reshape(data_num7[landmarks[i]], (28,28)),
40                               cmap = plt.cm.get_cmap("Greys"))
41         imgplot.set_interpolation("nearest")
42         plt.xticks([])
43         plt.yticks([])
44     plt.show()

```

Sau khi chạy các dòng code đó xong, thêm một xíu chỉnh sửa để chúng ta dễ hình dung xu hướng của các điểm qua hình sau.



Hình 18: Tập dữ liệu chữ viết số 7 sau khi giảm chiều và hình ảnh các hình được chọn random

Như hình trên thì ta sẽ thấy được 2 feature chính của các điểm dữ liệu đó là **gạch ở thân** và **độ nghiêng của thân** của số 7.

Dựa vào trục mà nhóm em vẽ qua việc phân tích xu hướng, ta có thể thấy được càng lên cao số 7 sẽ **không có gạch ở thân** và càng qua phía bên phải thì thân của số 7 ít nghiêng hơn. Càng xuống dưới thì số 7 sẽ có **gạch ở thân**, và càng qua phía bên trái thì **thân** số 7 bị **ngiêng** nhiều hơn

4 KẾT LUẬN ĐÁNH GIÁ

4.1 Đánh giá thuật toán Isomap

Thuật toán Isomap phụ thuộc lớn vào việc xây dựng lân cận nên việc chọn k và cấu trúc dữ liệu sẽ ảnh hưởng lớn đến kết quả cuối cùng.

- Về việc chọn k , thực nghiệm chứng minh rằng ta không thể chọn k một cách tùy ý.
- Nếu lân cận được chọn tồn tại các lỗ, khoảng cách euclid sẽ không xấp xỉ khoảng cách đồ thị nên dẫn đến kết quả sai.

4.2 Cải tiến thuật toán

4.2.1 Phương pháp chọn K - số nearest neighbors hợp lý

Lựa chọn vùng lân cận trong Isomap khá quan trọng. Nếu vùng lân cận tại mỗi điểm quá lớn, các vùng lân cận cục bộ sẽ bao gồm các điểm dữ liệu từ các nhánh khác của đa tạp, làm tắt chúng và dẫn đến lỗi phụ trong quá trình nhúng cuối cùng. Nếu nó quá nhỏ, sẽ dẫn đến gián đoạn, làm cho đa tạp phân mảnh thành một số lượng lớn các cụm bị ngắt kết nối. Vì vậy, khi có phương pháp chọn số k nearest neighbors hợp lý, cấu trúc của đa tạp sẽ được bảo toàn tốt hơn.

Thuật toán lựa chọn tham số tối ưu gồm 4 bước:

1. Chọn khoảng giá trị có thể nhận của K : $K_{opt} \in [K_{min}, K_{max}]$
 - K_{min} là giá trị nhỏ nhất của K , với giá trị này, đồ thị lân cận (kể từ bước 2 của thuật toán Isomap) sẽ không bị rời rạc.
 - K_{max} là giá trị nguyên lớn nhất thỏa $\frac{2P}{N} \leq K + 2$ trong đó P là số cạnh và N là số nút trong đồ thị kể từ bước 2 của thuật toán.

Giải thích:

- K_{max} là một giá trị hữu hạn, vì khi K tăng, P (số cạnh của đồ thị) sẽ tăng lên rất nhanh (vì mỗi điểm dữ liệu sẽ được kết nối thêm với một số điểm khác ở bước 1). Vì vậy, K tăng đến một lúc nào đó sẽ không còn thỏa mãn bất đẳng thức nữa.
- Sau khi hoàn thành các thử nghiệm và phân tích dữ liệu kết quả, nhận thấy rằng trong tất cả các trường hợp được xem xét, nếu giá trị trung bình của biểu đồ (được tính với một số giá trị của K) lớn hơn $K + 2$, thì thường nhận các lỗi tắt không chạy theo bề mặt của đa tạp.

Do đó ta chọn giới hạn trên của tập các giá trị K thỏa bất đẳng thức là K_{max}

2. Tính hàm mất mát $L(K)$ với mỗi $K_{opt} \in [K_{min}, K_{max}]$ (Hàm mất mát L trong bài toán giảm chiều dữ liệu tối ưu ở bước 3 - Isomap)
3. Ước tính tất cả các cực tiểu của $L(K)$ và các số K tương ứng để tạo ra tập S_K gồm các ứng cử viên ban đầu cho giá trị K tối ưu
4. Với mỗi $K \in S_K$, chạy thuật toán Isomap và xác định K_{opt} bằng công thức:

$$K_{opt} = \operatorname{argmin}_K (1 - p_{D_x D_y}^2)$$

Trong đó:

- D_x là ma trận geodesic distances của tập dữ liệu đầu vào X
- D_y là ma trận khoảng cách euclide của tập dữ liệu đầu ra Y
- p là hệ số tương quan tuyến tính tiêu chuẩn giữa tất cả các mục của D_x và D_y
 p nhận giá trị trong khoảng $[-1;1]$, $(1 - p^2)$ nhận giá trị càng nhỏ thì $\|p\|$ càng gần 1, D_x và D_y càng tương quan mật thiết.

Để tìm hiểu sâu hơn về vấn đề này, bạn đọc có thể tham khảo ở tài liệu:[11]

4.2.2 Landmark-Isomap

Giới thiệu về L-Isomap

Khi số lượng dữ liệu đầu vào N quá lớn, Isomap có thể trở nên quá tốn thời gian về cấu trúc đường dẫn ngắn nhất ($O(kN^2 \log(N))$) và phân rã eigenvalue MDS ($O(N^3)$).

Tăng tốc hai tính toán này, L-Isomap được đề xuất. L-Isomap chọn ngẫu nhiên n điểm từ X , gọi là các điểm mốc. Thay vì tính toán đường đi ngắn nhất giữa tất cả các điểm dữ liệu, L-Isomap chỉ tính toán đường đi ngắn nhất từ mỗi điểm dữ liệu đến các điểm mốc. Sau đó, cMDS được áp dụng cho ma trận khoảng cách trắc địa $n \times N$ thu được để tìm cách nhúng chiều thấp của các điểm mốc. Việc nhúng các điểm còn lại thu được bằng một phép biến đổi tuyến tính cố định của khoảng cách trắc địa của chúng đến các điểm mốc.

Bằng cách này, độ phức tạp về thời gian của đường đi ngắn nhất và tính toán MDS lần lượt được giảm xuống $O(knN \log(N))$ và $O(n^2N)$.

Các bước của thuật toán Landmark-Isomap

1. Chọn n điểm mốc từ tập dữ liệu X ($n \geq p + 1$, với p là số chiều cần giảm xuống)
2. Tính khoảng cách geodesic từ tất cả các điểm dữ liệu đến các điểm mốc đã chọn.
3. Áp dụng cMDS cho các điểm mốc (sử dụng ma trận khoảng cách geodesic Δ giữa các điểm mốc) và thu được ma trận biểu diễn chiều thấp (p -chiều) của các điểm mốc L_p
4. Đối với mỗi điểm, xác định vị trí của nó bằng cách sử dụng các khoảng cách trắc địa đã biết đến từng điểm mốc:
 - Tính δ_u trong đó δ_{ui} là trung bình hàng thứ i của ma trận Δ
 - Với mỗi điểm x_a , tính δ_a trong đó δ_{ai} là bình phương khoảng cách giữa điểm x_a và điểm mốc i
 - Biểu diễn chiều thấp của x_a là $y_a = -\frac{1}{2}L_p^{-1}(\delta_a - \delta_u)$ với L_p^{-1} là nghịch đảo penrose moore của L_p .

Vấn đề của L-Isomap

1. Cách chọn các điểm mốc
2. Sự không ổn định về cấu trúc liên kết, gây ra bởi các cạnh ngắn mạch

Hiện nay, đã có đề xuất về phương pháp chọn mốc và phương pháp loại trừ các cạnh ngắn mạch. Phương pháp tích hợp L-Isomap với hai cải tiến này được gọi là Robust L-Isomap (RL-Isomap). Bạn đọc có thể tham khảo tài liệu: [5] để biết thêm thông tin về RL-Isomap.

5 HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI

Dựa vào những nội dung được nêu trên, ta có thể thấy vẫn còn tiềm năng nghiên cứu ở những khía cạnh như sau:

- Ở khoản tính toán khoảng cách trên toàn dataset, việc sử dụng thuật toán Dijkstra có thể được cải thiện bằng cách sử dụng Fibonacci Heap.
- Cấu trúc dataset chưa được khai thác tối đa. Thuật toán Isomap vẫn hoạt động tương tự với những trường hợp đối lập về cấu trúc như khi dataset có phân bố đều và khi tồn tại nhiều "lỗ", trường hợp các tầng dữ liệu được tách biệt tốt và trường hợp nằm gần nhau.

Giải pháp: phân loại thuật toán Isomap dựa vào dataset để áp dụng những bước tính toán tối ưu hơn cho cấu trúc cụ thể của dataset đó.

TÀI LIỆU

- [1] Dr. Guangliang Chen. *Mathematical Methods for Data Visualization: Isometric Feature Mapping (ISOMap)*. PhD thesis, San José State University. URL: <https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec10ISOMap.pdf>.
- [2] Heeyoul Choi and Seungjin Choi. Robust kernel isomap. 2007. URL: https://www.researchgate.net/publication/222403996_Robust_kernel_Isomap".
- [3] Gopal Das. helo minh, May 2017. URL: <https://gopalcda.com/2017/05/24/construction-of-k-d-tree-and-using-it-for-nearest-neighbour-search/>.
- [4] Benyamin Ghogh, Ali Ghodsi, and Mark Crowley Fakhri Karray. Multidimensional scaling, sammon mapping, and isomap: Tutorial and survey. 2020. URL: <https://arxiv.org/pdf/2009.08136.pdf>.
- [5] HaoShi, Baoqun Yin, Yu Kang, Chao Shao, and Jie Gui. Robust l-isomap with a novel landmark selection method. 2017. URL: <https://www.hindawi.com/journals/mpe/2017/3930957/>.
- [6] Sungkyu Jung and Xingye Qiao. Stat 2221-lecture 8: Multidimensional scaling-advanced applied multivariate analysis, Spring 2015. URL: https://www.stat.pitt.edu/sungkyu/course/2221Spring15/lec8_mds.pdf.
- [7] Scikit learn Developers. *Scikit-learn Machine Learning in Python*. URL: <https://scikit-learn.org/>.
- [8] Yann LeCun and Corinna Cortes. *THE MNIST DATABASE of handwritten digits*. URL: <http://yann.lecun.com/exdb/mnist/>.
- [9] Dave Mount. Cmsc 420: Lecture 14: Answering queries with kd-trees, Fall 2019. URL: <https://www.cs.umd.edu/class/fall2019/cmsc420-0201/Lects/lect14-kd-query.pdf>.
- [10] Ashwini Kumar Pal. Dimension reduction - isomap, 2018. URL: <https://blog.paperspace.com/dimension-reduction-with-isomap/>.
- [11] O. Samko, A.D. Marshall, and P.L. Rosin. Selection of the optimal parameter value for the isomap algorithm. 2006. URL: https://www.researchgate.net/publication/223824802_Selection_of_the_optimal_parameter_value_for_the_Isomap_algorithm.