

Locally Linear Embedding

Quân Phan

Quốc Mai

Tùng Phạm

Trường Đặng

PiMA 2021



Trình bày: Nhóm 4, Locally Linear Embedding

August 8, 2021

- 1 Dimensionality Reduction và Sơ lược về Locally Linear Embedding
 - Manifold Learning và Manifold
 - Tổng quan về Locally Linear Embedding
- 2 Locally Linear Embedding cơ bản
 - Xây dựng neighbor graph
 - Xây dựng ma trận trọng số
 - Xây dựng dữ liệu mới từ ma trận trọng số
- 3 Áp dụng mô hình
- 4 Kết luận

Contents

- 1 Dimensionality Reduction và Sơ lược về Locally Linear Embedding
 - Manifold Learning và Manifold
 - Tổng quan về Locally Linear Embedding
- 2 Locally Linear Embedding cơ bản
 - Xây dựng neighbor graph
 - Xây dựng ma trận trọng số
 - Xây dựng dữ liệu mới từ ma trận trọng số
- 3 Áp dụng mô hình
- 4 Kết luận

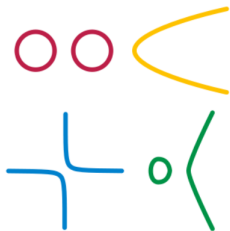


Manifold là gì?

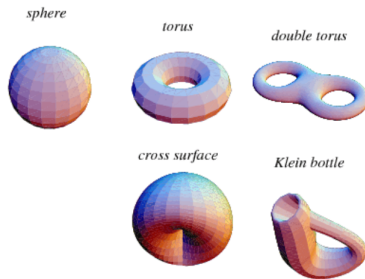
Một manifold d chiều là một cấu trúc hình học mà cục bộ tại mọi điểm của nó trông giống như một không gian Euclide d chiều.



Một số ví dụ về Manifold



Hình: Các manifold 1-chiều



Hình: Các manifold 2-chiều



Manifold Learning

Giả thiết đa tạp

Các điểm dữ liệu trong không gian D chiều đều nằm trên một manifold d chiều với $D > d$.



Manifold Learning

Định nghĩa

Một lớp các thuật toán giảm chiều dữ liệu **không giám sát**, bắt đầu với **giả thiết đa tạp** và sử dụng các **tính chất cục bộ** của manifold để tìm ra representation ít chiều của các điểm dữ liệu gốc nhiều chiều.



Contents

- 1 Dimensionality Reduction và Sơ lược về Locally Linear Embedding
 - Manifold Learning và Manifold
 - Tổng quan về Locally Linear Embedding
- 2 Locally Linear Embedding cơ bản
 - Xây dựng neighbor graph
 - Xây dựng ma trận trọng số
 - Xây dựng dữ liệu mới từ ma trận trọng số
- 3 Áp dụng mô hình
- 4 Kết luận



Ý tưởng chính

Một thuật toán giảm chiều dữ liệu thuộc lớp **Manifold Learning**, dựa vào **2** ý tưởng chính.

Gắn cho manifold (gần) chứa dữ liệu một hệ trục tọa độ.

Ý tưởng 1 - Bản chất của DR trong LLE

LLE giảm chiều dữ liệu bằng cách biến đổi **tọa độ ban đầu** của mỗi điểm dữ liệu thành **tọa độ so với hệ trục tọa độ trên manifold**.

Ý tưởng chính

Ý tưởng 2 - Local Geometry

Khi giảm chiều dữ liệu, LLE tập trung *bảo toàn* **local geometry** của các không gian tuyến tính cục bộ.

Tính chất của manifold

Mỗi điểm dữ liệu $x^{(i)}$ và tập hợp các neighbors của nó đều nằm trên một không gian tuyến tính cục bộ.



Input - Output

Input

- Tập dữ liệu cần giảm chiều $X_{m \times D}$ với m điểm dữ liệu (mỗi vector hàng của X là một điểm dữ liệu) và D features
- Tham số cho bài toán xác định neighbors
- Số chiều d cho dữ liệu đầu ra ($d < D$)

Output Tập dữ liệu đã được giảm chiều $Y_{m \times d}$ với m điểm dữ liệu (mỗi hàng của Y là một điểm dữ liệu) và d features.



Các bước thực hiện

- 1 Xác định neighbors cho từng điểm dữ liệu
- 2 Xấp xỉ mỗi điểm dữ liệu bằng một tổ hợp tuyến tính (có điều kiện) của các neighbors của nó
- 3 Sử dụng các trọng số của tổ hợp tuyến tính tìm được trong bước 2, suy ra representation (trong không gian ít chiều hơn) của điểm dữ liệu ban đầu



Ý tưởng chính ~ Các bước thực hiện

(Bước 2) Xấp xỉ mỗi điểm dữ liệu bằng một tổ hợp tuyến tính của các neighbors = *Xác định local geometry* (Ý tưởng 2)

(Bước 3) Sử dụng các trọng số tìm được, suy ra representation = *Bảo toàn local geometry* (Ý tưởng 2)

(Bước 2) Mục đích "điều kiện": Đảm bảo trọng số và output bất biến với các phép biến đổi tuyến tính (Ý tưởng 1)



Các bước thực hiện (Cụ thể!)

- 1 Xây dựng neighbor graph
- 2 Tìm ma trận trọng số W để tối ưu hóa hàm mất mát:

$$L(W) = \sum_{i=1}^m \left\| x^{(i)} - \sum_{j=1}^m w_{i,j} \cdot x^{(j)} \right\|^2$$

- 3 Giữ nguyên trọng số W , tìm tập dữ liệu đã giảm chiều Y sao cho hàm mất mát được tối ưu:

$$\Phi(Y) = \sum_{i=1}^n \left\| y^{(i)} - \sum_{j=1}^n w_{i,j} y^{(j)} \right\|^2$$



Contents

- 1 Dimensionality Reduction và Sơ lược về Locally Linear Embedding
 - Manifold Learning và Manifold
 - Tổng quan về Locally Linear Embedding
- 2 Locally Linear Embedding cơ bản
 - Xây dựng neighbor graph
 - Xây dựng ma trận trọng số
 - Xây dựng dữ liệu mới từ ma trận trọng số
- 3 Áp dụng mô hình
- 4 Kết luận



Ý tưởng của neighbor graph

- Khi phóng to vào 1 điểm x_i trên manifold, thì hình dạng vùng lân cận sẽ trông như 1 mặt phẳng
- Việc xây dựng neighbor graph sẽ giúp ta thu được thông tin về tính chất cục bộ tại mỗi điểm



Đo lường khoảng cách

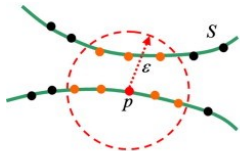
- Để so sánh khoảng cách giữa các điểm dữ liệu, ta cần 1 hàm đo lường khoảng cách
- Các khoảng cách thường dùng:

- Khoảng cách Euclid: $EUD(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$

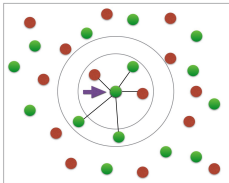
- Khoảng cách Cosine: $COD(p, q) = 1 - \cos(p, q)$



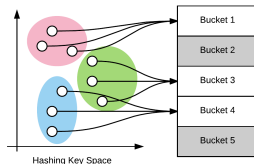
Các thuật toán dựng neighbor graph



Hình: ϵ – neighborhood



Hình: K-nearest neighbors



Hình: Locality sensitive hashing



Contents

- 1 Dimensionality Reduction và Sơ lược về Locally Linear Embedding
 - Manifold Learning và Manifold
 - Tổng quan về Locally Linear Embedding
- 2 Locally Linear Embedding cơ bản
 - Xây dựng neighbor graph
 - **Xây dựng ma trận trọng số**
 - Xây dựng dữ liệu mới từ ma trận trọng số
- 3 Áp dụng mô hình
- 4 Kết luận



Bài toán tối ưu

Mục tiêu: Xấp xỉ mỗi điểm dữ liệu bằng một tổ hợp tuyến tính của các neighbors của nó:

$$x^{(i)} \approx \sum_{j=1}^m w_{i,j} \cdot x^{(j)}$$

Tối thiểu hóa hàm mất mát:

$$L(W) = \sum_{i=1}^m \left\| x^{(i)} - \sum_{j=1}^m w_{i,j} \cdot x^{(j)} \right\|^2 \quad (1)$$



Điều kiện ràng buộc

Ta muốn các trọng số **bất biến** với các *phép quay*, *phép vị tự*, và *phép tịnh tiến*.

- Dạng của (1) cho thấy bất biến với *phép quay* và *phép vị tự*.
- Điều kiện bất biến với *phép tịnh tiến*:

$$\sum_{j=1}^m w_{i,j} = 1$$



Chứng minh bất biến với tịnh tiến

Với $\sum_{j=1}^m w_{i,j} = 1$:

$$\begin{aligned} & (x^{(i)} + t) - \sum_{j=1}^m w_{i,j} \cdot (x^{(j)} + t) \\ &= x^{(i)} + t - \left(\sum_{j=1}^m w_{i,j} \cdot x^{(j)} + \sum_{j=1}^m w_{i,j} \cdot t \right) \\ &= x^{(i)} - \sum_{j=1}^m w_{i,j} \cdot x^{(j)} + \left(t - t \cdot \sum_{j=1}^m w_{i,j} \right) \\ &= x^{(i)} - \sum_{j=1}^m w_{i,j} \cdot x^{(j)} \end{aligned}$$



Bài toán tối ưu

Tìm ma trận trọng số W sao cho:

$$W = \operatorname{argmin}_W \sum_{i=1}^m \left\| x^{(i)} - \sum_{j=1}^m w_{i,j} \cdot x^{(j)} \right\|^2$$

Thỏa mãn:

- 1 $w_{i,j} = 0$ nếu x_j không phải là neighbor của x_i
- 2 $\sum_{j=1}^m w_{i,j} = 1$ với mọi $1 \leq i \leq m$



Đơn giản hóa bài toán tối ưu

Đặt:

$$\epsilon_i = \left\| x^{(i)} - \sum_{j=1}^m w_{i,j} \cdot x^{(j)} \right\|^2$$

với mọi $i, j \in \{1, 2, \dots, m\}$.

$L(W)$ đạt cực tiểu tương đương với mọi ϵ_i được tối ưu hóa.



Một số biến đổi tương đương

Giả sử $x^{(i)}$ có k neighbors. Gọi các neighbors của $x_1^{(i)}$ lần lượt là $x_1^{(i)}, x_2^{(i)}, \dots, x_k^{(i)}$. Gọi thêm:

- Z_i là ma trận $k \times D$ có các vector hàng lần lượt là $x_1^{(i)} - x^{(i)}, x_2^{(i)} - x^{(i)}, \dots, x_k^{(i)} - x^{(i)}$
- W_i là ma trận $k \times 1$ chứa trọng số của các neighbors của $x^{(i)}$

Khi đó:

$$\epsilon_i = W_i^T Z_i Z_i^T W_i = W_i^T G_i W_i$$

với $G_i = Z_i Z_i^T$ là một ma trận Gram.



Phát biểu lại

Tối ưu hóa

$$\epsilon_i = W_i^T G_i W_i$$

với điều kiện $\mathbf{1}^T W_i - \mathbf{1} = 0$.



Giải bài toán tối ưu

$$\begin{cases} G_i W_i = \frac{\lambda}{2} \mathbf{1} \\ \mathbf{1}^T W_i - 1 = 0 \end{cases} \quad (2)$$

Nếu G_i khả nghịch, (2) tương đương:

$$\begin{cases} W_i = \frac{\lambda}{2} G_i^{-1} \mathbf{1} \\ \mathbf{1}^T W_i - 1 = 0 \end{cases}$$

Nếu G_i không khả nghịch, sử dụng kỹ thuật chính quy hóa.



Contents

- 1 Dimensionality Reduction và Sơ lược về Locally Linear Embedding
 - Manifold Learning và Manifold
 - Tổng quan về Locally Linear Embedding
- 2 Locally Linear Embedding cơ bản
 - Xây dựng neighbor graph
 - Xây dựng ma trận trọng số
 - Xây dựng dữ liệu mới từ ma trận trọng số
- 3 Áp dụng mô hình
- 4 Kết luận



Bài toán tối ưu

- **Mục tiêu:** Giữ được tính tuyến tính cục bộ của dữ liệu
- Tối thiểu hóa hàm mất mát:

$$\Phi(Y) = \sum_{i=1}^m \left\| y^{(i)} - \sum_{j=1}^m w_{i,j} y^{(j)} \right\|^2.$$

Tương tự hàm mất mát trong bước xây dựng ma trận trọng số:

$$L(W) = \sum_{i=1}^m \left\| x^{(i)} - \sum_{j=1}^m w_{i,j} \cdot x^{(j)} \right\|^2$$



Bài toán tối ưu

- **Tuy nhiên**, hàm mất mát $\Phi(Y)$ có nghiệm suy biến (các điểm dữ liệu $y^{(1)}, y^{(2)}, \dots, y^{(m)}$ trùng nhau)
 \implies Cần đặt thêm các điều kiện bổ sung.

Các điều kiện bổ sung

- $\sum_{i=1}^m y^{(i)} = 0$
- $\frac{1}{m} \sum_{i=1}^m y^{(i)} y^{(i)T} = I$

Điều kiện đầu tiên để lời giải là duy nhất theo phép tịnh tiến.

Điều kiện thứ hai đảm bảo nghiệm không bị suy biến.



Phát biểu bài toán

Tìm ma trận dữ liệu đầu ra $Y \in \mathbb{R}^{m \times d}$ để:

$$Y = \underset{Y}{\operatorname{argmin}} \Phi(Y) = \underset{Y}{\operatorname{argmin}} \operatorname{tr} \left(Y^T (I - W)^T (I - W) Y \right)$$

và thỏa mãn các điều kiện:

- $Y^T \mathbf{1} = 0$
- $\frac{1}{m} Y^T Y = I_m$

Chỉ sử dụng các đẳng thức trên đường chéo ma trận của điều kiện thứ hai, các điều kiện còn lại tự động được thỏa mãn.

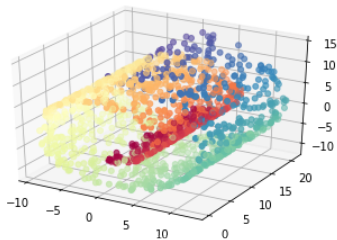


Giải bài toán tối ưu

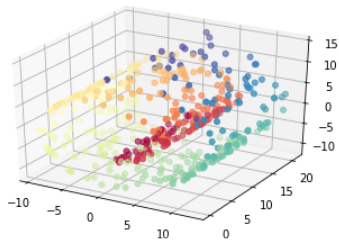
- Sử dụng phương pháp nhân tử Lagrange, nghiệm tối ưu Y sẽ có các vector cột là các vector riêng ứng với d trị riêng nhỏ nhất khác 0 của $(I - W)^T(I - W)$.
- Để ý rằng $(I - W)^T(I - W)$ là ma trận Gram nên các giá trị riêng luôn không âm, và do đó có thể chọn ra $d + 1$ giá trị riêng nhỏ nhất.



Dataset swiss roll



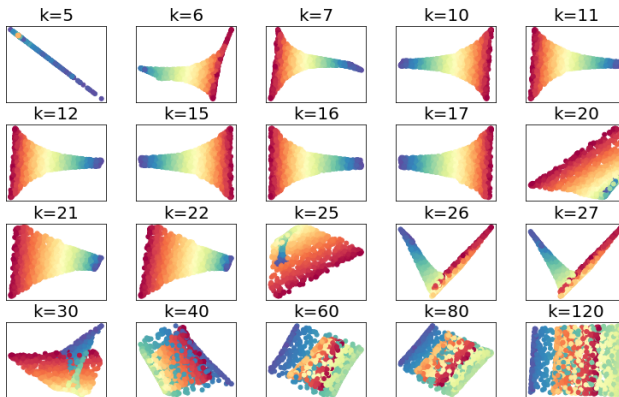
Hình: $n = 1500$



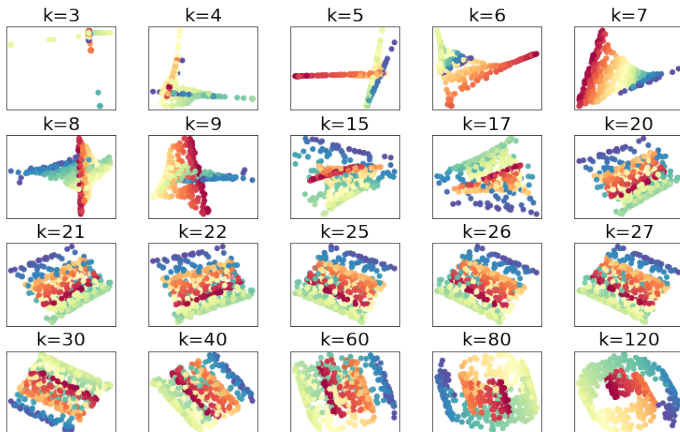
Hình: $n = 500$



K-nearest neighbor trên tập $n = 1500$



K-nearest neighbor trên tập $n = 500$



Nhận xét

- Thuật toán LLE làm không tốt với bộ dữ liệu có mật độ thưa và làm khá tốt với bộ dữ liệu phân bố đều và dày.
- Với mỗi khoảng tham số nhất định sẽ cho các kết quả tương tự nhau.
- Với các tham số quá nhỏ, neighbor graph sẽ trở nên không liên thông, khiến cho kết quả bị suy biến.
- Với các tham số quá lớn, mỗi điểm sẽ có rất nhiều neighbor, dẫn tới việc các tính chất hình học cục bộ bị mất đi.



Bài toán nén ảnh

Mục đích

Sử dụng LLE để giảm số lượng dữ liệu cần để biểu diễn 1 bức ảnh, qua đó bóc tách các tính chất đặc trưng → Ứng dụng trong các mô hình thị giác máy tính.

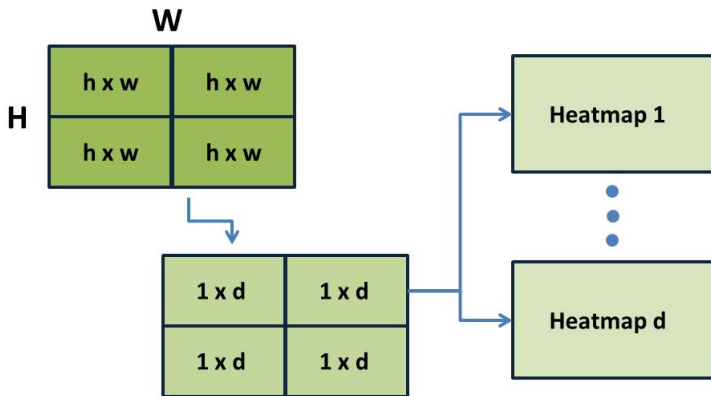


Các bước thực hiện nén ảnh

- Phân ảnh thành các patch nhỏ
- Sử dụng LLE để giảm chiều các patch
- Sắp xếp lại các feature của từng patch vào các vị trí tương ứng để tạo thành ma trận heatmap

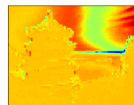
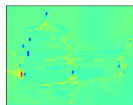
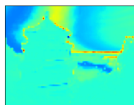
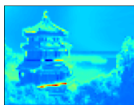
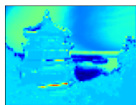
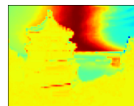
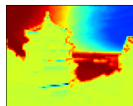
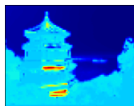
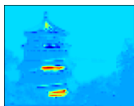
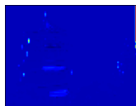


Minh họa các bước nén ảnh



Kết quả nén ảnh

The Mathematics of Data Science



Hình: Ảnh gốc và các heatmap



Tóm tắt

Một thuật toán giảm chiều dữ liệu thuộc lớp Manifold Learning.

Với nền tảng là giả thiết đa tạp, LLE trải qua 3 bước:

- 1 Xây dựng neighbor graph
- 2 Biểu diễn mỗi điểm dữ liệu bằng một tổ hợp tuyến tính có ràng buộc của các neighbors
- 3 Dùng ma trận trọng số tìm được để xác định tập dữ liệu mới ít chiều hơn.



Ưu điểm của LLE

- 1 Đảm bảo tìm được nghiệm tối ưu toàn cục
- 2 Yêu cầu ít siêu tham số
- 3 Nhu cầu lưu trữ thấp.
- 4 Nhiều ma trận thưa. Có thể tận dụng để giảm không gian lưu trữ và thời gian chạy thuật toán.



Một số lưu ý khi áp dụng

- 1 Tập dữ liệu cần phải nằm (xấp xỉ gần) trên một đa tạp
- 2 Phải chọn phương pháp và siêu tham số xây dựng neighbor graph phù hợp.
 - Với k -nearest neighbors: Thường được chọn trong thực tế. Mặc dù chọn k không dễ, LLE vẫn cho kết quả khá tốt trên một khoảng k khá rộng.
 - Với ϵ -neighborhood: khó chọn ϵ . Giá trị phù hợp phụ thuộc rất nhiều vào kích cỡ và hình dạng cục bộ của manifold dữ liệu.



References I

- [1] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. March 25, 2019. *Eigenvalue and Generalized Eigenvalue Problems: Tutorial*. <https://arxiv.org/pdf/1903.11240.pdf>.
- [2] Lê Quang Tiên. August 25, 2018. *Locally Linear Embedding (LLE)*. <https://thetalog.com/machine-learning/locally-linear-embedding/>.
- [3] Jianzhong Wang. 2012. *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*. <https://bit.ly/JianzhongWang2012>.
- [4] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. November 22, 2020. *Locally Linear Embedding and its Variants: Tutorial and Survey*. https://bit.ly/LLEtut_updated.



References II

- [5] Sam T. Roweis and Lawrence K. Saul
<https://cs.nyu.edu/~roweis/lle/algorithm.html>.
- [6] Ashwini Kumar Pal. 2018.
<https://blog.paperspace.com/dimension-reduction-with-lle/>.

