Compressing Data via Dimensionality Reduction

Team 04

Faculty of Mathematics and Computer Science
University of Science

Table of Contents

Table of Contents

Vì sao phải giảm số chiều dữ liệu ?

- Khó khăn về tính toán, thời gian thực thi
- Hạn chế về không gian lưu trữ
- Các tính chất của dữ liệu trong số chiều nhỏ có thể không còn đúng trong trường hợp số chiều lớn, dẫn đến việc phân tích dữ liệu khó khăn

Vì sao phải giảm số chiều ?

Khó khăn:

- Việc chiếu dữ liệu từ không gian có số chiều lớn xuống không gian có số chiều thấp dẫn đến các thông tin quan trọng của dữ liệu bị mất một phần (hoặc toàn bộ) (Giải quyết trong phương pháp PCA và LDA)
- Việc giảm chiều của dữ liệu có thể làm cho bài toán ban đầu có thể dẫn đến không giải quyết được

Vì sao phải giảm số chiều ?

Thuận lợi:

- Việc chiếu dữ liệu xuống không gian thấp hơn có thể dẫn đến bài toán trở nên dễ dàng hơn
- Thời gian tính toán nhanh chóng, cho ra kết quả nhanh hơn
- Tiết kiệm bộ nhớ lưu trữ dữ liệu

Vì sao phải giảm số chiều ?

Ý tưởng chung của các thuật toán sẽ giới thiệu:
Cho vectơ $\boldsymbol{X} \in \mathbb{R}^n$, ta cần chuyển dữ liệu về vectơ $\boldsymbol{Y} \in \mathbb{R}^m$ với $m << n$
Xây dựng ma trận chiếu $\boldsymbol{W} \in \mathbb{R}^{mxn}$
Khi đó: $\boldsymbol{Y} = \boldsymbol{WX} \in \mathbb{R}^m$

Table of Contents

Mở đầu về PCA

PCA chính là phương pháp đi tìm một hệ cơ sở mới sao cho thông tin của dữ liệu chủ yếu tập trung ở một vài toạ độ, phần còn lại chỉ mang một lượng nhỏ thông tin.

Ngoài ra, việc xác định mẫu hoặc mô hình của dữ liệu với số lượng chiều lớn thông qua các phương pháp trực quan bằng đồ họa thông thường sẽ rất khó khăn, do vậy PCA sẽ là công cụ hữu ích giúp chúng ta giải quyết vấn đề này.

1. Trị riêng và vectơ riêng

- Cho ma trận $\boldsymbol{A} \in M_n(\mathbb{C})$.
  Ta nói $\lambda \in \mathbb{C}$ là trị riêng của ma trận $\boldsymbol{A}$ nếu có vectơ $\boldsymbol{u} \in \mathbb{C}^n \backslash \{0\}$ sao cho:
$$\boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{u}$$
- Khi đó mọi vectơ $\boldsymbol{u}$ thỏa phương trình trên được gọi là vectơ riêng của ma trận $\boldsymbol{A}$ ứng với trị riêng λ

1. Trị riêng và vectơ riêng

- Với $\lambda \in \mathbb{C}$ là một trị riêng của $\boldsymbol{A}$, đặt:

$$
\begin{aligned}
E(\lambda) &= \{\boldsymbol{u} \in \mathbb{C}^n | \boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{u}\} \\
&= \{0\} \cup \{\text{Tập hợp các vecto riêng ứng với trị riêng } \lambda\} \\
&= \{\boldsymbol{u} \in \mathbb{C}^n | (\boldsymbol{A} - \boldsymbol{\lambda} \boldsymbol{I_n})\boldsymbol{u} = 0\} \\
&= \text{ Không gian nghiệm của hệ } (\boldsymbol{A} - \boldsymbol{\lambda} \boldsymbol{I_n})\boldsymbol{X} = 0
\end{aligned}
$$

- $E(\lambda)$ được gọi là không gian riêng của $\boldsymbol{A}$ ứng với trị riêng λ

Nhắc lại một vài kiến thức

1. Trị riêng và vectơ riêng

- Đa thức đặc trưng của $\boldsymbol{A} \in M_{nxn}(\mathbb{C})$ được định nghĩa bởi:
$$P_{\boldsymbol{A}}(\lambda) = \det(\boldsymbol{A} - \lambda \boldsymbol{I_n})$$
với biến $\lambda \in \mathbb{C}$
- **Định lí:**
$$\lambda \text{ là một trị riêng của } \boldsymbol{A} \iff P_{\boldsymbol{A}}(\lambda) = 0$$

2. Kì vọng và ma trận hiệp phương sai của vectơ ngẫu nhiên

- Một vectơ ngẫu nhiên là vectơ mà mỗi phần tử của vectơ là một biến ngẫu nhiên
- Kì vọng của vectơ ngẫu nhiên là vectơ mà ở đó mỗi phần tử là kì vọng của phần tử tương ứng trong vectơ ngẫu nhiên

$$\mu \equiv \boldsymbol{E}(\boldsymbol{X}) = \boldsymbol{E}\begin{pmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \cdots \\ \boldsymbol{X}_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{E}(\boldsymbol{X}_1) \\ \boldsymbol{E}(\boldsymbol{X}_2) \\ \cdots \\ \boldsymbol{E}(\boldsymbol{X}_n) \end{pmatrix}$$

2. Kì vọng và ma trận hiệp phương sai của vectơ ngẫu nhiên

- Ma trận ngẫu nhiên là ma trận mà mỗi phần tử là một biến ngẫu nhiên.
- Kì vọng của ma trận ngẫu nhiên là ma trận mà ở đó mỗi phần tử là kì vọng của phần tử tương ứng trong ma trận ngẫu nhiên
- Ma trận hiệp phương sai được định nghĩa bởi:

$$\Sigma \equiv \boldsymbol{E}(\boldsymbol{X}-\mu)(\boldsymbol{X}-\mu)^{\boldsymbol{T}} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{pmatrix}$$

trong đó $\sigma_{ij} = \boldsymbol{E}(\boldsymbol{X_i}-\boldsymbol{\mu_i})(\boldsymbol{X_j}-\boldsymbol{\mu_j})$

2. Kì vọng và ma trận hiệp phương sai của vectơ ngẫu nhiên
Với mẫu ngẫu nhiên $\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n} \in \mathbb{R}^D$, ta định nghĩa vectơ trung bình mẫu:[1]

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x_i} \in \mathbb{R}^D$$

và ma trận hiệp phương sai mẫu:

$$\boldsymbol{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x_i} - \bar{\boldsymbol{x}})(\boldsymbol{x_i} - \bar{\boldsymbol{x}})^T \in M_{DxD}(\mathbb{R})$$

Nhận xét: Ma trận hiệp phương sai mẫu luôn là ma trận đối xứng thực **nửa** xác định dương

[1] mỗi vector đại diện cho một điểm dữ liệu (sample) thứ i

Các kết quả của trị riêng - vectơ riêng ứng dụng trong PCA

Định lý 01

Cho $\boldsymbol{A}$ là ma trận đối xứng thực. Khi đó mọi trị riêng phức của $\boldsymbol{A}$ đều là số thực

Định lý 02

Mọi ma trận đối xứng xác định dương (t.ư nửa xác định dương) đều có các trị riêng thực dương (t.ư không âm)

Các kết quả của trị riêng - vectơ riêng ứng dụng trong PCA

Định lý 03

Cho $\boldsymbol{A}$ là ma trận đối xứng. Khi đó các không gian con riêng của $\boldsymbol{A}$ trực giao với nhau từng đôi một, tức mỗi vectơ riêng ứng trị riêng này sẽ trực giao với mọi vectơ riêng ứng với trị riêng kia

Định lý 04

$\boldsymbol{A}$ là ma trận đối xứng $\iff$ $\boldsymbol{A}$ chéo hóa trực giao được, nghĩa là có ma trận $\boldsymbol{P}$ là ma trận trực giao sao cho $\boldsymbol{P}^{-1}\boldsymbol{AP}$ là ma trận đường chéo, với các hệ số trên đường chéo là các trị riêng

Các kết quả của trị riêng - vectơ riêng ứng dụng trong PCA

Cho vectơ ngẫu nhiên $\boldsymbol{X} = (X_1, X_2, \cdots, X_p)^T \in \mathbb{R}^p$.
Ta định nghĩa thành phần chính thứ i là tổ hợp tuyến tính $\boldsymbol{a_i^T X}$ sao cho:

- $\text{Var}(\boldsymbol{a_i^T X})$ đạt giá trị lớn nhất với điều kiện $\boldsymbol{a_i^T a_i} = 1$
- $\text{Cov}(\boldsymbol{a_i^T X}, \boldsymbol{a_k^T X}) = 0\ \forall k \neq i$

Định lý 05

Cho $\boldsymbol{\Sigma}$ là ma trận hiệp phương sai của biến ngẫu nhiên $\boldsymbol{X}$, cùng với các trị riêng $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ và họ trực chuẩn vectơ riêng tương ứng: $\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e_p}$
Khi đó thành phần chính thứ i được cho bởi:

$$Y_i = \boldsymbol{e_i^T X}\ \forall i = 1, 2, \cdots, p$$

Với cách chọn đó thì:

$$\begin{aligned} \text{Var}(Y_i) &= \lambda_i,\ \forall i = 1, 2, \cdots, p \\ \text{Cov}(Y_i, Y_k) &= 0\ \forall i \neq k \end{aligned}$$

Định lý 06

Với $\boldsymbol{Y} = (Y_1, Y_2, \cdots, Y_p)^T$ là vectơ gồm các thành phần chính của vectơ ngẫu nhiên $\boldsymbol{X} = (X_1, X_2, \cdots, X_p)^T$. Khi đó:

$$\begin{aligned}\sum_{i=1}^{p} \sigma_{ii} &= \sum_{i=1}^{p} \mathrm{Var}(X_i) \\ &= \sum_{i=1}^{p} \lambda_i = \sum_{i=1}^{p} \mathrm{Var}(Y_i)\end{aligned}$$

Thuật toán PCA

Các bước của thuật toán PCA:

- Chuẩn hóa dữ liệu (d-chiều)
- Xây dựng ma trận hiệp phương sai từ dữ liệu đã được chuẩn hoá
- Tìm các vectơ riêng, trị riêng của ma trận hiệp phương sai
- Xây dựng tập trực chuẩn từ tập hợp các vecto trên
- Sắp xếp các trị riêng theo chiều giảm dần
- Chọn k vectơ riêng đầu tiên ứng với k trị riêng đầu tiên trong bộ trị riêng có thứ tự ở bước trên ($k < d$)
- Xây dựng ma trận chiếu $\boldsymbol{W} \in M_{kxd}(\mathbb{R})$ từ các vectơ riêng trên
- Tìm hình chiếu Y của ma trận dữ liệu X trong không gian mới sinh bởi ma trận $\boldsymbol{W}$, $Y = \boldsymbol{W}X$

Demo thuật toán PCA

Cho tập dữ liệu sau:

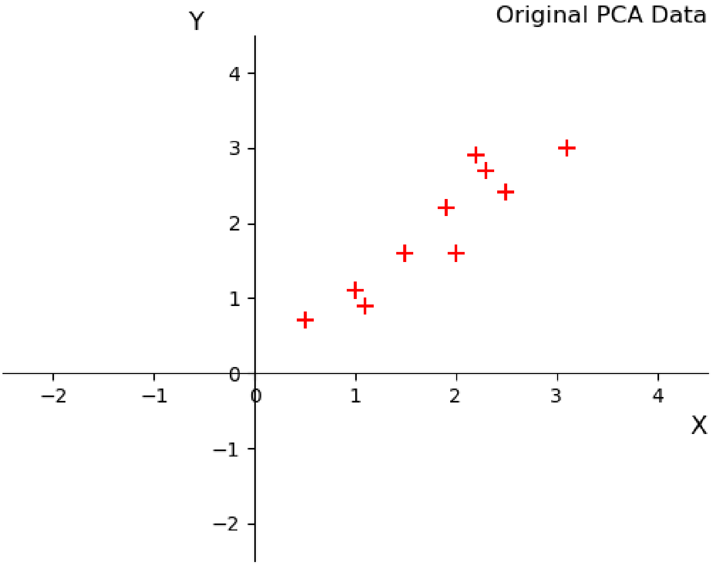
	X	Y
Data	2.5	2.4
	.5	.7
	2.2	2.9
	1.9	2.2
	3.1	3.0
	2.3	2.7
	2	1.6
	1	1.1
	1.5	1.6
	1.1	.9

Demo thuật toán PCA

Trung bình của mẫu: $\bar{x} = 1.81, \bar{y} = 1.91$

Độ lệch chuẩn của mẫu: $s_X \approx .785, s_Y \approx .846$

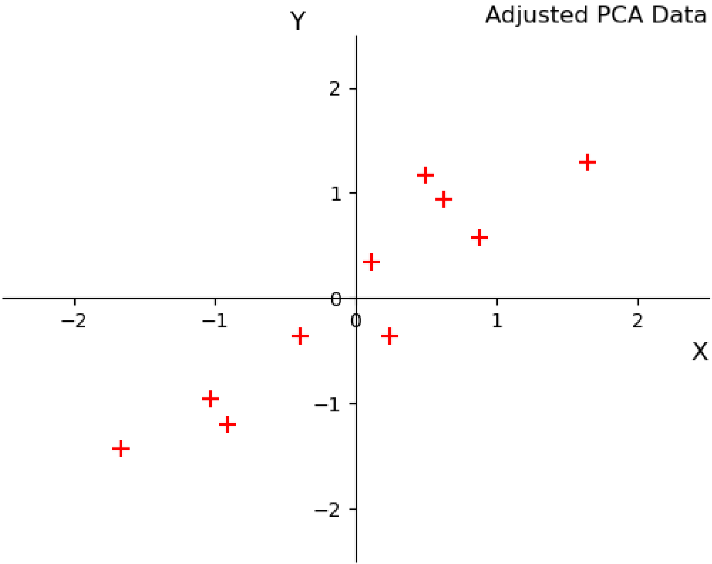
Demo thuật toán PCA



Bước 1: Chuẩn hóa dữ liệu qua phép biến đổi $\dfrac{X_i - \bar{x}}{s_X}, \dfrac{Y_i - \bar{y}}{s_Y} \ \forall i$

	X	Y
Data Adjust	.879	.579
	-1.668	-1.429
	.497	1.170
	.115	.343
	1.643	1.288
	.624	.933
	.242	-.366
	-1.032	-.957
	-.395	-.366
	-.904	-1.193

Demo thuật toán PCA



Bước 2: Xây dựng ma trận hiệp phương sai trên mẫu đã chuẩn hóa

$$\boldsymbol{S} = \frac{1}{9}\sum_{i=1}^{10} \boldsymbol{x_i x_i^T}$$

$$= \begin{pmatrix} 1 & 0.926 \\ 0.926 & 1 \end{pmatrix}$$

trong đó: $\boldsymbol{x_i} = (x_i, y_i)^T, i = 1, 2, \cdots, 10$

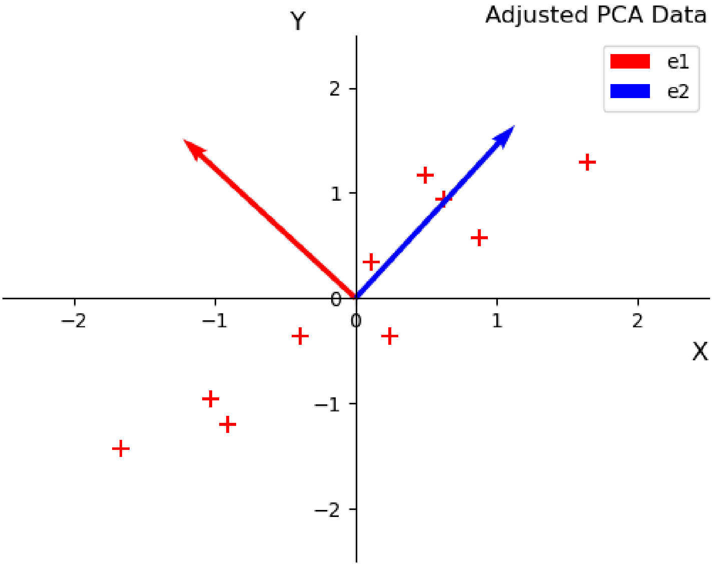
Demo thuật toán PCA

Bước 3: Tìm các vectơ riêng, trị riêng của ma trận hiệp phương sai mẫu

Trị riêng thứ nhất: $\boldsymbol{\lambda}_1 \approx .074$ ứng với vectơ riêng: $\boldsymbol{e}_1 = \begin{pmatrix} -.707 \\ .707 \end{pmatrix}$

Trị riêng thứ hai: $\boldsymbol{\lambda}_2 \approx 1.926$ ứng với vectơ riêng: $\boldsymbol{e}_2 = \begin{pmatrix} .707 \\ .707 \end{pmatrix}$

Demo thuật toán PCA



Demo thuật toán PCA

Trị riêng thứ nhất: $\boldsymbol{\lambda}_1 \approx .074$ ứng với vectơ riêng: $\boldsymbol{e}_1 = \begin{pmatrix} -.707 \\ .707 \end{pmatrix}$

Trị riêng thứ hai: $\boldsymbol{\lambda}_2 \approx 1.926$ ứng với vectơ riêng: $\boldsymbol{e}_2 = \begin{pmatrix} .707 \\ .707 \end{pmatrix}$

Chọn các vector riêng giữ lại nhiều thông tin của dữ liệu nhất. Trị riêng $\boldsymbol{\lambda}_1$ nhỏ hơn $\boldsymbol{\lambda}_2$, ứng với tỷ lệ thông tin bị mất khi loại bỏ vector riêng $\boldsymbol{e}_1$ thấp hơn khi loại bỏ vector riêng $\boldsymbol{e}_2$.

(Trường hợp này có thể chọn giữ lại $\boldsymbol{e}_2$ hoặc cả $\boldsymbol{e}_1$ và $\boldsymbol{e}_2$).

Demo thuật toán PCA

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{e}_1^{\boldsymbol{T}} \\ \boldsymbol{e}_2^{\boldsymbol{T}} \end{pmatrix} = \begin{pmatrix} -.707 & .707 \\ .707 & .707 \end{pmatrix}$$

$$\boldsymbol{X} = \begin{pmatrix} .879 & .579 \\ -1.668 & -1.429 \\ .497 & 1.170 \\ .115 & .343 \\ 1.643 & 1.288 \\ .624 & .933 \\ .242 & -.366 \\ -1.032 & -.957 \\ -.395 & -.366 \\ -.904 & -1.193 \end{pmatrix}$$

$$\boldsymbol{Y}_{\boldsymbol{output}}^{\boldsymbol{T}} = \boldsymbol{W}\boldsymbol{X}^T$$

Demo thuật toán PCA

$$\boldsymbol{Y}_{\boldsymbol{output}}^{\boldsymbol{T}} = \begin{pmatrix} -.707 & .707 \\ .707 & .707 \end{pmatrix} \begin{pmatrix} .879 & .579 \\ -1.668 & -1.429 \\ .497 & 1.170 \\ .115 & .343 \\ 1.643 & 1.288 \\ .624 & .933 \\ .242 & -.366 \\ -1.032 & -.957 \\ -.395 & -.366 \\ -.904 & -1.193 \end{pmatrix}^{\top}$$

Demo thuật toán PCA

$$\boldsymbol{Y_{output}} = \begin{pmatrix} -.212 & 1.030 \\ .169 & -2.190 \\ .476 & 1.178 \\ .161 & .323 \\ -.251 & 2.072 \\ .219 & 1.101 \\ -.430 & -.0878 \\ .053 & -1.406 \\ .020 & -.538 \\ -.204 & -1.483 \end{pmatrix}$$

Demo thuật toán PCA - Python

```
In [2]: rnd =  np.random.RandomState(0)

In [3]: data =  rnd.normal(size = 100).reshape(100,1)

In [4]: X = data.dot(data.T)

In [5]: X.shape
Out[5]: (100, 100)

In [6]: # Kiểm tra tính đối xứng của ma trận X
        (X == X.T).all()
Out[6]: True

In [7]: eigenvals, _ = np.linalg.eig(X)

In [8]: # Số phức ?
        eigenvals[0]
Out[8]: (101.94036179371047+0j)

In [9]: eigenvals.dtype
Out[9]: dtype('complex128')
```

Demo thuật toán PCA - Python

Nhận xét 1: Các vectơ trong kết quả không là họ trực chuẩn
Nhận xét 2: Các trị riêng - vectơ riêng trong kết quả đều là trị riêng - vectơ riêng trong trường số phức, với phần ảo của các vectơ xấp xỉ 0

Demo thuật toán PCA - Python

Nguyên nhân: Floating point number
Giải pháp: Sử dụng lệnh np.linalg.eigh, lệnh này lợi dụng tính đối xứng của ma trận

Demo thuật toán PCA - Python

```
In [2]: rnd =  np.random.RandomState(0)

In [3]: data =  rnd.normal(size = 100).reshape(100,1)

In [4]: X = data.dot(data.T)

In [5]: X.shape

Out[5]: (100, 100)

In [6]: # Kiểm tra tính đối xứng của ma trận X
        (X == X.T).all()

Out[6]: True

In [7]: eigenvals, _ = np.linalg.eigh(X)

In [8]: eigenvals[0]

Out[8]: -3.393482558402277e-14

In [9]: eigenvals.dtype

Out[9]: dtype('float64')
```

Demo thuật toán PCA - Python

Nhận xét 1: Các trị riêng - vectơ riêng trong kết quả là họ trực chuẩn
Nhận xét 2: Các trị riêng - vectơ riêng trong kết quả đều là trị riêng - vectơ riêng trong trường số thực (đúng theo lý thuyết)

Bắt đầu phần code

Table of Contents

Introduction to LDA

LDA cùng với PCA là hai phương pháp giúp cho việc tính toán chính xác và hiệu quả hơn.

Ý tưởng

Thuật toán LDA tìm những trục toạ độ phân biệt nhất giữa các lớp. Những trục này sau đó được sử dụng để định nghĩa mặt phẳng mà data sẽ chiếu xuống. Điểm mạnh của LDA chính là làm cho các hình chiếu của các lớp cách xa nhau nhất có thể. Nói một cách khác, LDA giúp chúng ta tìm ra phép chiếu sao cho tỉ lệ giữa between-class variance và within-class variance lớn nhất có thể. 2 thuật ngữ sẽ được mô tả chi tiết ở các slide sau.

Thuật toán LDA

Các bước của thuật toán LDA:

- Chuẩn hóa dữ liệu (d-chiều) (d là số feature).
- Với mỗi class, tính vectơ trung bình d chiều.
- Xây dựng ma trận phân tán giữa các class $\boldsymbol{S_B}$ (between-class scatter matrix) và ma trận phân tán trong class (within-class scatter matrix) $\boldsymbol{S_w}$

Thuật toán LDA

- Gọi $\boldsymbol{m}$ là trung bình của tất cả các mẫu, $\boldsymbol{m}_i$ là trung bình của các mẫu trong lớp thứ i, n_i là số lượng mẫu có trong lớp i ta có công thức cho $\boldsymbol{S_W}$ và $\boldsymbol{S_B}$ như sau

$$\begin{aligned}
\boldsymbol{S_w} &= \sum_{i=1}^{c} \boldsymbol{S_i} \\
\boldsymbol{S_B} &= \sum_{i=1}^{c} n_i(\boldsymbol{m_i} - \boldsymbol{m})(\boldsymbol{m_i} - \boldsymbol{m})^T
\end{aligned}$$

Trong đó $\boldsymbol{S_i}$ chính là ma trận phân tán của lớp thứ i, được tính như sau:

$$\boldsymbol{S_i} = \sum_{x \in D_i} (\boldsymbol{x} - \boldsymbol{m_i})(\boldsymbol{x} - \boldsymbol{m_i})^T$$

Thuật toán LDA

- Tính vectơ riêng và trị riêng tương ứng của ma trận $\boldsymbol{S_w}^{-1}\boldsymbol{S_B}$.
- Sắp xết các trị riêng giảm dần.
- Chọn k vectơ riêng tương ứng với k trị riêng lớn nhất để xây dựng một ma trận chuyển đổi $\boldsymbol{W}$ có dxk chiều.
- Chiếu mẫu lên không gian con mới sử dụng ma trận chuyển đổi $\boldsymbol{W}$.

Demo thuật toán LDA

Cho tập dữ liệu sau:

	X	Y	class
Data	2.5	2.4	1
	0.5	0.7	0
	2.2	2.9	1
	1.9	2.2	1
	3.1	3.0	1
	2.3	2.7	1
	2	1.6	0
	1	1.1	0
	1.5	1.6	0
	1.1	0.9	0

Trung bình của mẫu: $\bar{x} = 1.81, \bar{y} = 1.91$

Độ lệch chuẩn của mẫu: $s_X \approx 0.785, s_Y \approx 0.846$

Demo thuật toán LDA

Bước 1: Chuẩn hóa dữ liệu qua phép biến đổi $\dfrac{\boldsymbol{X_i} - \bar{x}}{s_X}, \dfrac{\boldsymbol{Y_i} - \bar{y}}{s_Y} \; \forall i$

	X	Y	class
Data Adjust	.879	.579	1
	-1.668	-1.429	0
	.497	1.170	1
	.115	.343	1
	1.643	1.288	1
	.624	.933	1
	.242	-.366	0
	-1.032	-.957	0
	-.395	-.366	0
	-.904	-1.193	0

Demo thuật toán LDA

Bước 2: Với mỗi class, tính vectơ trung bình

$$\boldsymbol{m_i} = \frac{1}{n_i} \sum_{x \in D_i}^{c} x_m$$

	class 0	class 1
X	-.7514	.7514
Y	-.8624	.8624

Bước 3: Xây dựng ma trận phân tán giữa các lớp ($\boldsymbol{S_B}$) và ma trận phân tán trong lớp ($\boldsymbol{S_w}$).

$$\boldsymbol{S_w} = \sum_{i=1}^{c} \boldsymbol{S_i}$$

Trong đó $\boldsymbol{S_i}$ chính là ma trận phân tán của lớp i, được tính như sau:

$$\boldsymbol{S_i} = \sum_{x \in D_i} (\boldsymbol{x} - \boldsymbol{m_i})(\boldsymbol{x} - \boldsymbol{m_i})^T$$

Demo thuật toán LDA

$$\boldsymbol{S_w} = \begin{pmatrix} 3.3541 & 1.8535 \\ 1.8535 & 1.563 \end{pmatrix}$$

Ta sẽ tính ma trận phân tán giữa các lớp $\boldsymbol{S_B}$ như sau:
Với $\boldsymbol{m}$ là trung bình của tất cả mẫu trong dữ liệu

$$\boldsymbol{S_B} = \sum_{i=1}^{c} n_i(\boldsymbol{m_i} - \boldsymbol{m})(\boldsymbol{m_i} - \boldsymbol{m})^T$$

$$\boldsymbol{S_B} = \begin{pmatrix} 5.6459 & 6.4798 \\ 6.4798 & 7.437 \end{pmatrix}$$

Bước 4: Tính vectơ riêng và trị riêng tương ứng của ma trận $S_w^{-1}S_B$.

Tìm trị riêng với ma trận $S_w^{-1}S_B = \begin{pmatrix} -1.763 & -2.0235 \\ 6.2364 & 7.1576 \end{pmatrix}$

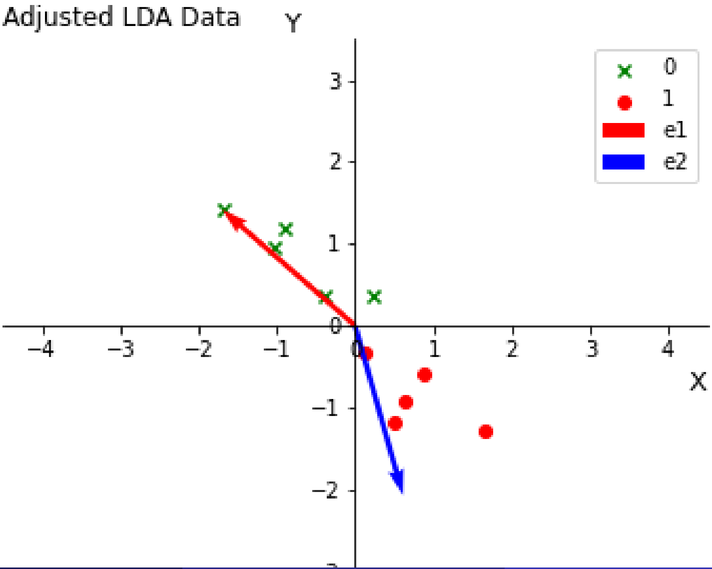
Trị riêng thứ nhất: $\lambda_1 \approx -8.881e-16$ ứng với vectơ riêng: $e_1 = \begin{pmatrix} -.754 \\ .6569 \end{pmatrix}$

Trị riêng thứ hai: $\lambda_2 \approx 5.3937$ ứng với vectơ riêng: $e_2 = \begin{pmatrix} .272 \\ -.9623 \end{pmatrix}$

Chọn các vector riêng giữ lại nhiều thông tin của dữ liệu nhất. Trị riêng λ_1 nhỏ hơn λ_2, ứng với tỷ lệ thông tin bị mất khi loại bỏ vector riêng e_1 thấp hơn khi loại bỏ vector riêng e_2.

(Trường hợp này có thể chọn giữ lại e_2.)

Demo thuật toán LDA



Demo thuật toán LDA

Chọn 1 vectơ riêng tương ứng với 1 trị riêng lớn nhất để xây dựng một ma trận chuyển đổi $\boldsymbol{W}$ có 2x1 chiều.

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{e}_2 \end{pmatrix} = \begin{pmatrix} .272 \\ -.9623 \end{pmatrix}$$

$$\boldsymbol{X} = \begin{pmatrix} .879 & .579 \\ -1.668 & -1.429 \\ .497 & 1.170 \\ .115 & .343 \\ 1.643 & 1.288 \\ .624 & .933 \\ .242 & -.366 \\ -1.032 & -.957 \\ -.395 & -.366 \\ -.904 & -1.193 \end{pmatrix} \qquad \boldsymbol{Y_{output}} = \boldsymbol{XW}$$

Demo thuật toán LDA

$$Y_{output} = \begin{pmatrix} -.318 \\ .9217 \\ -.9903 \\ -.2985 \\ -.7922 \\ -.7283 \\ .4182 \\ .6402 \\ .245 \\ .9022 \end{pmatrix}$$

Demo thuật toán LDA

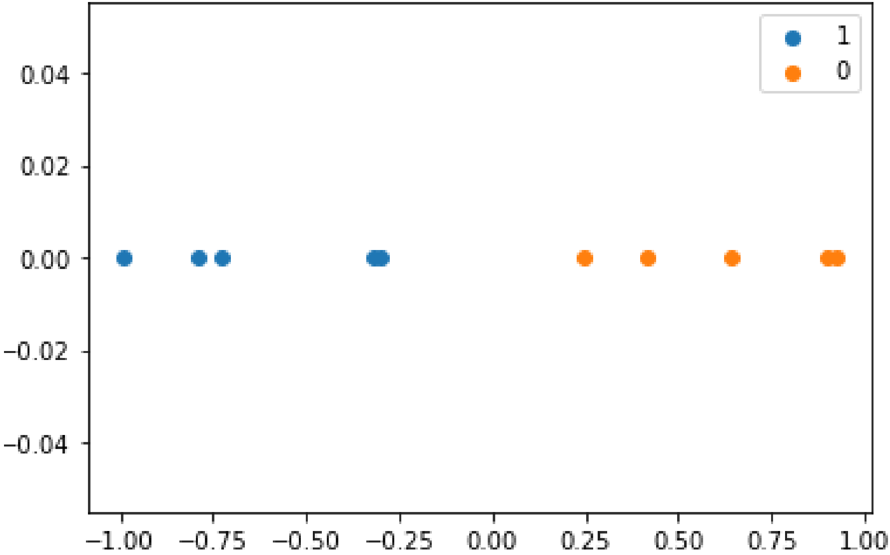
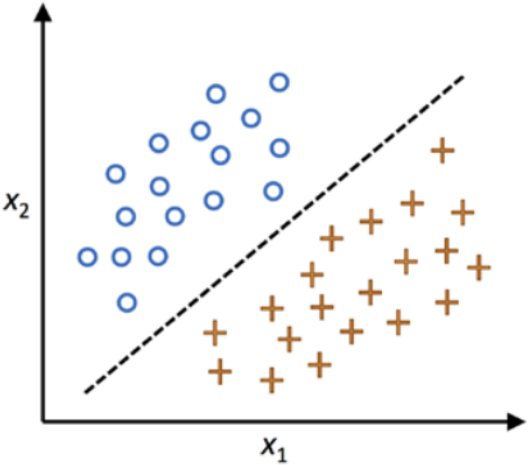


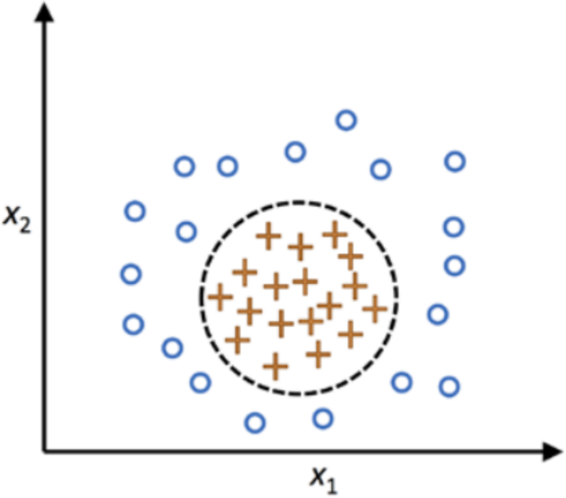
Table of Contents

Intro

Dữ liệu tuyến tính: có thể phân loại bằng các công cụ phân lớp tuyến tính (đường thẳng, mặt phẳng, siêu phẳng, ...)



Vấn đề non-linear ?



Intro to KPCA

Với các trường hợp dữ liệu thuộc dạng phi tuyến tính (non-linear), các kỹ thuật biến đổi tuyến tính cho việc giảm số chiều như PCA, LDA có vẻ không phải là sự lựa chọn tốt.

Ý tưởng thuật toán KPCA

Chiếu các dữ liệu ở dạng non-linear sang không gian mới với số chiều lớn hơn, tại đây các điểm dữ liệu sẽ trở nên tách biệt và có thể áp dụng các công cụ phân lớp tuyến tính.

Sau đó sử dụng các thuật toán giảm số chiều để chiếu các điểm dữ liệu về lại không gian với số chiều nhỏ hơn. Bài toán được giải quyết.

Định nghĩa

Cho $\boldsymbol{x} \in \mathbb{R}^d$. Định nghĩa ánh xạ ϕ:

$$\phi : \mathbb{R}^d \longrightarrow \mathbb{R}^k \qquad (k \gg d)$$

Ví dụ: Cho $\boldsymbol{x} = (x_1, x_2)^T \longrightarrow \boldsymbol{z} = \phi(\boldsymbol{x}) = (x_1^2, \sqrt{2x_1x_2}, x_2^2)^T$

Một vài kiến thức toán

Ma trận hiệp phương sai:

$$S = \frac{1}{n-1}\sum_{i=1}^{n}\phi(\mathbf{x_i})\phi(\mathbf{x_i})^T$$

Hàm và ma trận kernel

Hàm kernel:

$$\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) = \phi(\boldsymbol{x_i})^T \phi(\boldsymbol{x_j})$$

Ma trận kernel:

$$\boldsymbol{K} = \phi(\boldsymbol{X})\phi(\boldsymbol{X})^T$$

$$\boldsymbol{K} = \begin{pmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & \kappa(\boldsymbol{x}_1, \boldsymbol{x_n}) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1) & \kappa(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & \kappa(\boldsymbol{x}_2, \boldsymbol{x_n}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x_n}, \boldsymbol{x}_1) & \kappa(\boldsymbol{x_n}, \boldsymbol{x}_2) & \cdots & \kappa(\boldsymbol{x_n}, \boldsymbol{x_n}) \end{pmatrix}$$

Nhược điểm

Nhắc lại

Hàm kernel

$$\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) = \phi(\boldsymbol{x_i})^T \phi(\boldsymbol{x_j})$$

Ánh xạ ϕ

$$\phi : \mathbb{R}^d \longrightarrow \mathbb{R}^k \qquad (k \gg d)$$

Trong đó: $\phi(\boldsymbol{x})$ rất khó tính toán ở không gian với số chiều k lớn (có thể là vô hạn)

Ví dụ minh họa hàm kernel

Ví dụ: Xét phép biến đổi 1 điểm dữ liệu trong không gian hai chiều $\boldsymbol{x} = [x_1, x_2]^T$ thành một điểm trong không gian 5 chiều $\phi(\boldsymbol{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$. Ta có:

$$\begin{aligned}
\phi(\boldsymbol{x})^T\phi(\boldsymbol{z}) &= [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2] \\
&.[1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1z_2, z_2^2]^T \\
&= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2x_2^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 \\
&= (1 + x_1z_1 + x_2z_2)^2 \\
&= (1 + \boldsymbol{x}^T\boldsymbol{z})^2 \\
&= \kappa(\boldsymbol{x}, \boldsymbol{z})
\end{aligned}$$

Hàm Kernel và ma trận Kernel

Định lý

(Tiêu chuẩn Mercer) Với hàm kernel κ có dạng như trên, điều kiện cần và đủ để hàm κ là một kernel là với mọi tập $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(m)}\}, (m < \infty)$ thì ma trận kernel $\boldsymbol{K}$ tương ứng là ma trận nửa xác định dương.

$\boldsymbol{K}$ được gọi là nửa xác định dương, nếu và chỉ nếu:

$$\sum_{j=1}^{n}\sum_{i=1}^{n}\kappa(\boldsymbol{x_i}, \boldsymbol{x_j})c_j c_i \geq 0, \forall c_k \in \mathbb{R}, k = 1, 2, ..., n$$

Mở rộng

Trong thực hành, có một vài hàm số κ không thỏa mãn điều kiện Merrcer nhưng vẫn cho kết quả chấp nhận được. Những hàm số này vẫn được gọi là kernel. Trong phạm vi của bài thuyết trình này chỉ tập trung vào các hàm kernel thông dụng (có sẵn trong thư viện sklearn).

Một số hàm kernel thông dụng

- **Polynomial**

$$\kappa(\mathbf{x_i}, \mathbf{x_j}) = (\gamma \mathbf{x_i}^T \mathbf{x_j} + \theta)^d$$

Trong đó, θ là ngưỡng và d là hệ số mũ tùy chỉnh bởi người dùng. Ví dụ: $\gamma = 1, \theta = 0, d = 1$ ta được hàm kernel tuyến tính

- **Sigmoid**

$$\kappa(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\gamma \mathbf{x_i}^T \mathbf{x_j} + \theta)$$

trong đó: $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

- **Radial Basic Function (Gaussian kernel)**

$$\kappa(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|_2^2\right), \gamma > 0$$

Ví dụ cho trường hợp vô hạn

Ví dụ: Radial Basic Function (Gaussian Kernel)
Cho hai vector x, y trong $\mathbb{R}$. Chọn Gaussian kernel với $\lambda = 1$:

$$\boldsymbol{\kappa}(x, y) = \exp(-(x-y)^2)$$

Ta có:

$$\exp(-(x-y)^2) = \exp(-x^2 + 2xy - y^2) = \exp(-x^2)\exp(-y^2)\sum_{n=0}^{\infty}\frac{2^n x^n y^n}{n!}$$

Đặt:

$$\phi : \mathbb{R} \longrightarrow \ell^2$$

với $\phi(x) = (x_1, x_2, ..., \cdots, x_n, \cdots), x_n = \dfrac{2^{n/2} x^n \exp(-x^2)}{\sqrt{n!}}$

Vậy: $\boldsymbol{\kappa}(x, y) = \phi(x)^T \phi(y)$

Tóm tắt các kernel thông dụng

Tên	Công thức	Kernel	Thiết lập hệ số
polynomial	$(\gamma \boldsymbol{x_i}^T \boldsymbol{x_j} + \theta)^d$	'poly'	$d : degree, \gamma : gamma, \theta : coef0$
sigmoid	$\tanh(\gamma \boldsymbol{x_i}^T \boldsymbol{x_j} + \theta)$	'sigmoid'	γ: gamma, θ: coef0
rbf	$\exp\left(-\gamma \lVert \boldsymbol{x_i} - \boldsymbol{x_j} \rVert_2^2\right)$	'rbf'	$\gamma > 0$: gamma

Thuật toán KPCA

- Chọn hàm kernel $\kappa(\mathbf{x_i}, \mathbf{x_j})$
- Tính ma trận kernel $\mathbf{K}$

$$\mathbf{K} = \begin{pmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x_n}) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x_n}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x_n}, \mathbf{x}_1) & \kappa(\mathbf{x_n}, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x_n}, \mathbf{x_n}) \end{pmatrix}$$

- Chuẩn hóa : $\mathbf{K}' = \mathbf{K} - 1_{\mathbf{n}}\mathbf{K} - \mathbf{K}1_{\mathbf{n}} + 1_{\mathbf{n}}\mathbf{K}1_{\mathbf{n}}$
  Với $1_{\mathbf{n}}$ là ma trận có số chiều giống với ma trận kernel và có tất cả các giá trị bằng $\frac{1}{n}$
- Tính trị riêng, vector riêng của ma trận $\mathbf{K}'$
- Sắp xếp các trị riêng giảm dần
- Chọn k vector riêng tương ứng với k giá trị riêng để xây dựng một ma trận chuyển đổi $\mathbf{W}$.
- Chiếu mẫu lên không gian con mới sử dụng ma trận chuyển đổi $\mathbf{W}$

Demo kernel PCA

Cho tập dữ liệu sau:

	X	Y	class
Data	0	0	1
	-1	2	0
	0	-1	0
	2	0	1
	1	-2	1
	-2	1	1

Demo kernel PCA

Bước 1: Chọn hàm kernel

$$\boldsymbol{\kappa}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-2\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2\right) \qquad (\text{với } \gamma = 2)$$

Bước 2: Tính ma trận kernel

$$\boldsymbol{K} = \begin{pmatrix} 1 & e^{-10} & e^{-2} & e^{-8} & e^{-10} & e^{-10} \\ e^{-10} & 1 & e^{-20} & e^{-26} & e^{-40} & e^{-4} \\ e^{-2} & e^{-20} & 1 & e^{-10} & e^{-4} & e^{-16} \\ e^{-8} & e^{-26} & e^{-10} & 1 & e^{-10} & e^{-34} \\ e^{-10} & e^{-40} & e^{-4} & e^{-10} & 1 & e^{-20} \\ e^{-10} & e^{-4} & e^{-16} & e^{-34} & e^{-20} & 1 \end{pmatrix}$$

Bước 3 : Chuẩn hóa ma trận K

$$\boldsymbol{K}' = \boldsymbol{K} - 1_n\boldsymbol{K} - \boldsymbol{K}1_n + 1_n\boldsymbol{K}1_n$$

(với $n = 6$)

$$\boldsymbol{K}' = \begin{pmatrix} .798 & -.183 & -.070 & -.180 & -.183 & -.183 \\ -.183 & .837 & -.186 & -.160 & -.163 & -.145 \\ -.070 & -.186 & .791 & -.183 & -.167 & -.186 \\ -.180 & -.160 & -.183 & .843 & -.160 & -.160 \\ -.183 & -.163 & -.167 & -.160 & .837 & -.163 \\ -.183 & -.145 & -.186 & -.160 & -.163 & .837 \end{pmatrix}$$

Demo kernel PCA

Bước 4 : Tìm trị riêng, vector riêng cho ma trận K'

Ta lấy hai cặp (trị riêng, vectơ riêng) ứng với hai trị riêng lớn nhất.

- Ứng với trị riêng $\boldsymbol{\lambda}_1 \approx 8.881x10^{-16}$, ta có vector riêng :

$$\boldsymbol{e}_1 = \begin{pmatrix} -4.082x10^{-1} \\ 6.925x10^{-1} \\ -5.666x10^{-7} \\ 1.015x10^{-1} \\ -1.030x10^{-1} \\ -5.768x10^{-1} \end{pmatrix}$$

- Ứng với trị riêng $\boldsymbol{\lambda}_2 \approx .863$, ta có vector riêng :

$$\boldsymbol{e}_2 = \begin{pmatrix} -4.082x10^{-1} \\ -1.665x10^{-2} \\ -7.071x10^{-1} \\ -8.421x10^{-6} \\ -4.582x10^{-1} \\ 3.507x10^{-1} \end{pmatrix}$$

Bước 5 : Xây dựng ma trận chuyển đổi W có 6x2 chiều
Chọn 2 vectơ riêng tương ứng với 2 trị riêng lớn nhất để xây dựng một ma trận chuyển đổi $\boldsymbol{W} \in \mathbb{R}^{6x2}$).

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 \end{pmatrix} = \begin{pmatrix} -4.082x10^{-1} & -4.082x10^{-1} \\ 6.925x10^{-1} & -1.665x10^{-2} \\ -5.666x10^{-7} & -7.071x10^{-1} \\ 1.015x10^{-1} & -8.421x10^{-6} \\ -1.030x10^{-1} & -4.582x10^{-1} \\ -5.768x10^{-1} & 3.507x10^{-1} \end{pmatrix}$$

$$\boldsymbol{K'} = \begin{pmatrix} .798 & -.183 & -.070 & -.180 & -.183 & -.183 \\ -.183 & .837 & -.186 & -.160 & -.163 & -.145 \\ -.070 & -.186 & .791 & -.183 & -.167 & -.186 \\ -.180 & -.160 & -.183 & .843 & -.160 & -.160 \\ -.183 & -.163 & -.167 & -.160 & .837 & -.163 \\ -.183 & -.145 & -.186 & -.160 & -.163 & .837 \end{pmatrix}$$

$$\boldsymbol{Y_{output}} = \boldsymbol{K'W}$$

Demo kernel PCA

$$\boldsymbol{Y_{output}} = \begin{pmatrix} -.3462 & -.2535 \\ .7382 & .2160 \\ .0057 & -.5166 \\ .1568 & .2224 \\ -.0467 & -.2450 \\ -.5079 & .5767 \end{pmatrix}$$

Demo Kernel PCA - Python

Bắt đầu phần code.

Table of Contents

Tổng kết

Thank you for your attention.