Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)

Nguyễn Tuấn Anh, Hà Văn Hoàng, Nguyễn Tr. Thịnh, Lê Châu Anh

Trường Đại học Công nghệ Thông tin

CS115.N11.KHTN

Ngày 2 tháng 1 năm 2023

Tổng quan

1. Introduction
2. Linear Algebra
3. Singular Value Decomposition (SVD)
4. Principal Component Analysis (PCA)
5. SVD vs PCA
6. PCA vs LDA

Introduction

Bây giờ, chúng ta không nhắc lại định nghĩa, tính chất, các phép toán trên ma trận cũng như các phép biến đổi sơ cấp.

Definition (Trị riêng, vector riêng)

Cho ma trận $A \in \mathbb{R}^{n \times n}$. Nếu có vector $x \neq 0$ và số vô hướng λ sao cho $Ax = \lambda x$ thì λ được gọi là trị riêng của A còn x được gọi là vector riêng của A ứng với giá trị riêng λ.

Definition (Trực giao, trực chuẩn)

Một hệ cơ sở $u_1, u_2, ..., u_m \in \mathbb{R}^m$ được gọi là trực giao nếu

$$u_i \neq 0; \quad u_i^T u_j = 0 \quad \forall \quad 1 \leq i \neq j \leq m$$

Một hệ cơ sở $u_1, u_2, ..., u_m \in \mathbb{R}^m$ được gọi là trực chuẩn nếu

$$u_i^T u_j = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases}$$

$$u_i \neq 0 \quad \forall \quad 1 \leq i, j \leq m$$

Ma trận U là ma trận trực giao nếu $U \in \mathbb{R}^{n \times n}$ bao gồm các vector cột u_i thỏa mãn tính chất:

$$||u_i||_2^2 > 0, u_i^T u_j = 0 \quad \forall \quad 1 \leq i \neq j \leq n$$

Definition (Chéo hóa)

Ma trận vuông A được gọi là chéo hóa được khi và chỉ khi tồn tại ma trận P vuông không suy biến và ma trận đường chéo D thỏa mãn $P^{-1}AP = D$

Theorem (Điều kiện chéo hóa)

Ma trận vuông $A \in \mathbb{R}^{n\times n}$ chéo hóa được khi và chỉ khi nó có n vector riêng độc lập tuyến tính.

Definition (Trace)

Trace của ma trận A, ký hiệu trace(A), là tổng của tất cả các phần tử trên đường chéo của ma trận đó.

- $\text{trace}(A) = \text{trace}(A^T)$
- $\text{trace}(AB) = \text{trace}(BA)$
- $\text{trace}(A + B) = \text{trace}(A) + \text{trace}(B)$

Definition (Đa thức đặc trưng)

Xét ma trận $A \in \mathbb{R}^{n \times n}$. Đa thức đặc trưng của A, ký hiệu $p_A(\lambda)$ được xác định bởi $p_A(\lambda) = \det(\lambda I_n - A)$.

Example

Ma trận $A = \begin{pmatrix} 0 & -9 \\ -6 & 15 \end{pmatrix}$ có đa thức đặc trưng là $p_A(\lambda) = (\lambda - 6)(\lambda - 9)$.

Nhận xét: Nghiệm λ của $p_A(\lambda)$ được gọi là giá trị riêng của A.

Ở ví dụ trên, A có 2 trị riêng phân biệt là 6 và 9.

Cho 2 cơ sở của KGVT V: $E = \{e1, e2, ..., e_n\}$ và $E' = \{e_1', e_2', ..., e_n'\}$.
$\forall x \in V$, cần tìm liên hệ mà

$$x = x_1 e_1 + x_2 e_2 + ... + x_n e_n = x_1' e_1' + x_2' e_2' + ... + x_n' e_n'$$

Ngoài ra chúng ta còn có các khai triển

$$\begin{cases} e_1' &= a_{11} e_1 + a_{21} e_2 + ... + a_{n1} e_n \\ e_2' &= a_{12} e_1 + a_{22} e_2 + ... + a_{n2} e_n \\ .... & \\ e_n' &= a_{1n} e_1 + a_{2n} e_2 + ... + a_{nn} e_n \end{cases}$$

Từ đó

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix} \begin{pmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{pmatrix} \equiv A \begin{pmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{pmatrix}$$

Definition (Ma trận chuyển cơ sở)

Ma trận A ở trên được gọi là ma trận chuyển cơ sở từ E sang E'.

Example

Trong $\mathbb{R}^3$, cho 2 cơ sở

$$E = \{(1;1;1),(1;0;1),(1;1;0)\}, E' = \{(1;1;2),(1;2;1),(1;1;1)\}$$

Khi đó ma trận chuyển cơ sở từ E sang E' là

$$P = E^{-1}E' = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Definition (Frobenius Norm)

Frobenius norm của một ma trận được định nghĩa là căn bậc hai của tổng bình phương các phần tử của ma trận. Frobenius norm của một ma trận A được ký hiệu là $||A||_F$. Được xác định bởi

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

Example

Ma trận $A = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix}$ có $||A||_F = \sqrt{5}$.

Definition (L2 - Norm)

Độ dài Euclide của một vector $x \in \mathbb{R}^m$ được gọi là L2 - Norm của vector đó, được xác định bởi

$$||x||_2 = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$$

Nhận xét: Bình phương của L2 - Norm chính là tích vô hướng của một vector với chính nó, $||x||_2^2 = x^T x$.

Example

Vector $x = (1, 2, 3)$ có L2 - Norm là $||x||_2 = \sqrt{14}$.

Theorem

Cho A, B là các ma trận vuông cấp n. Khi đó đa thức đặc trưng của AB và BA trùng nhau.

Theorem

Cho A là một ma trận trực giao, khi đó tồn tại một cơ sở trực giao sao cho ma trận của A trong cơ sở đó có dạng đường chéo khối với các khối ± 1, hoặc $\begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$

Nhận xét: Biểu thức này chính là phép quay?

Singular Value Decomposition (SVD)

Introduction

Singular Value Decomposition (SVD) là một trong những phương pháp thuộc nhóm matrix factorization.

Mục đích ban đầu của phương pháp này là tìm ra một phép xoay không gian sao cho tích vô hướng của các vector không thay đổi. Phương pháp SVD được phát triển dựa trên những tính chất của ma trận trực giao và ma trận đường chéo để tìm ra một ma trận xấp xỉ với ma trận gốc.

Phương pháp này được ứng dụng rộng rãi trong các lĩnh vực xử lý hình ảnh, clustering, các thuật toán nén và giảm chiều dữ liệu và các bài toán recommendation.

Singular Value Decomposition (SVD)

Theorem (Singular Value Decomposition - SVD)

Cho ma trận chữ nhật $A \in \mathbb{R}^{m \times n}$. Khi đó A có thể được phân tích thành:

$$A = U\Sigma V^\top$$

Giải thích:

Trong đó $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ là các ma trận trực giao, còn $\Sigma \in \mathbb{R}^{m \times n}$ là ma trận thỏa mãn $\Sigma_{ij} = 0, \forall i \neq j$ và $\sigma_i = \Sigma_{ii} \geq \Sigma_{jj} = \sigma_j \geq 0, \forall i \geq j$.

Các phần tử trên nằm đường chéo của ma trận Σ gồm các $\sigma_i, i = \overline{1, r}$ được gọi là *singular values*. Các vector cột $u_i, i = \overline{1, m}$ của ma trận U được gọi là *left-singular vectors* và các vector cột $v_j, j = \overline{1, n}$ của ma trận V được gọi là *right-singular vectors*.

Proof of the SVD theorem

Singular Value Decomposition - SVD

Cho ma trận chữ nhật $A \in \mathbb{R}^{m \times n}$. Khi đó A có thể được phân tích thành:

$$A = U\Sigma V^\top$$

Hệ quả. Từ SVD của A chúng ta có được

$$AA^\top = U\Sigma V^\top (U\Sigma V^\top)^\top = U\Sigma V^\top V\Sigma^\top U^\top = U(\Sigma\Sigma^\top)U^\top$$

$$A^\top A = (U\Sigma V^\top)^\top U\Sigma V^\top = V\Sigma^\top U^\top U\Sigma V^\top = V(\Sigma^\top\Sigma)V^\top$$

Có nghĩa là

- U là *eigenvectors matrix* của $AA^\top$.
- V là *eigenvectors matrix* của $A^\top A$.
- Các trị riêng của $AA^\top$, $A^\top A$ bằng bình phương singular values của A.

Proof of the SVD theorem

Chứng minh. Cho ma trận $A \in \mathbb{R}^{m \times n}$. Chúng ta có thể tìm ra SVD của A thông qua việc giải phương trình có các ẩn là 3 ma trận

$$A = U\Sigma V^T$$

Giả sử tồn tại phép phân tích trên. Ta có $X^T X \in \mathbb{R}^{n \times n}$ là ma trận đối xứng, bán xác định dương, theo Spectral Theorem suy ra

$$A^T A = V(\Sigma^T \Sigma)V^T$$

Trong đó, $V \in \mathbb{R}^{n \times n}$ là ma trận trực giao, $\Sigma \in \mathbb{R}^{n \times n}$ là ma trận đường chéo gồm các phần tử trên đường chéo thỏa mãn

$$\sigma_1 \geq ... \geq \sigma_r > 0 = \sigma_{r+1} = ... = \sigma_n \quad (r = \text{rank}(A) \leq n)$$

Công thức trên giúp ta tìm ra ma trận V chứa các vector riêng của $A^T A$ và ma trận Σ chứa các singular values của A. Sau khi tìm được V và Σ, ta có thể viết lại công thức trên thành

$$AV = U\Sigma$$

Proof of the SVD theorem

Hoặc dưới dạng tích các vector cột

$$A[v_1 \dots v_n] = [u_1 \dots u_n] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \\ & & \end{bmatrix}$$

So sánh cột, ta thu được

$$Av_i = \begin{cases} \sigma_i u_i, & 1 \leq i \leq r \text{ (singular values khác 0)} \\ 0, & r < i \leq n \end{cases}$$

Điều này ngụ ý rằng chúng ta cần tìm ma trận U mà

$$u_i = \frac{1}{\sigma_i} Av_i, \forall i = \overline{1, r}$$

Proof of the SVD theorem

Dễ dàng chứng minh được $u_1, ..., u_r$ là các vector trực giao do

$$u_i^T u_j = \left(\frac{1}{\sigma_i} A v_i\right)^T \left(\frac{1}{\sigma_j} A v_j\right) = \frac{1}{\sigma_i \sigma_j} v_i^T A^T A v_j$$
$$= \frac{1}{\sigma_i \sigma_j} v_i^T (\sigma_j v_j) = \frac{1}{\sigma_i \sigma_j} v_i^T v_j (\lambda_j = \sigma_j^2)$$
$$= \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

Cuối cùng, chọn $u_{r+1}, ..., u_n \in \mathbb{R}^n$ thỏa mãn

$$U = [u_1 ... u_r u_{r+1} ... u_n] \in \mathbb{R}^{n \times n}$$

là ma trận trực giao. Hoàn tất chứng minh. □

Singular Value Decomposition (SVD)

Trong Python, để tính SVD của một ma trận, chúng ta sử dụng module linalg của numpy như sau:

Calculating SVD in Python

```
import numpy as np
from numpy import linalg as LA

m, n = 2, 3
A = np.random.rand(m, n)

U, S, V = LA.svd(A)

# checking if U, V are orthogonal and S is a diagonal matrix with
# nonnegative decreasing elements
print 'Frobenius norm of (UU^T - I) =', \
    LA.norm(U.dot(U.T) - np.eye(m))
print '\n S = ', S, '\n'
print 'Frobenius norm of (VV^T - I) =', \
    LA.norm(V.dot(V.T) - np.eye(n))
```

Singular Value Decomposition (SVD)

Compact SVD

Biểu thức $A = U\Sigma V^\top$ có thể viết dưới dạng tổng các ma trận A_i rank 1

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T = \sum_{i=1}^{r} \sigma_i A_i$$

- u_i là các vector cột của ma trận trực giao U
- v_i là các vector cột của ma trận trực giao T
- Mỗi $A_i = u_i v_i^T, 1 \geq i \geq r$ là một ma trận có rank $= 1$

Singular Value Decomposition (SVD)

Compact SVD

Ta thấy ma trận A chỉ phụ thuộc vào r cột đầu tiên của U, V và r giá trị σ_i của ma trận Σ. Vậy nên ta có thể phân tích A gọn hơn (compact SVD):

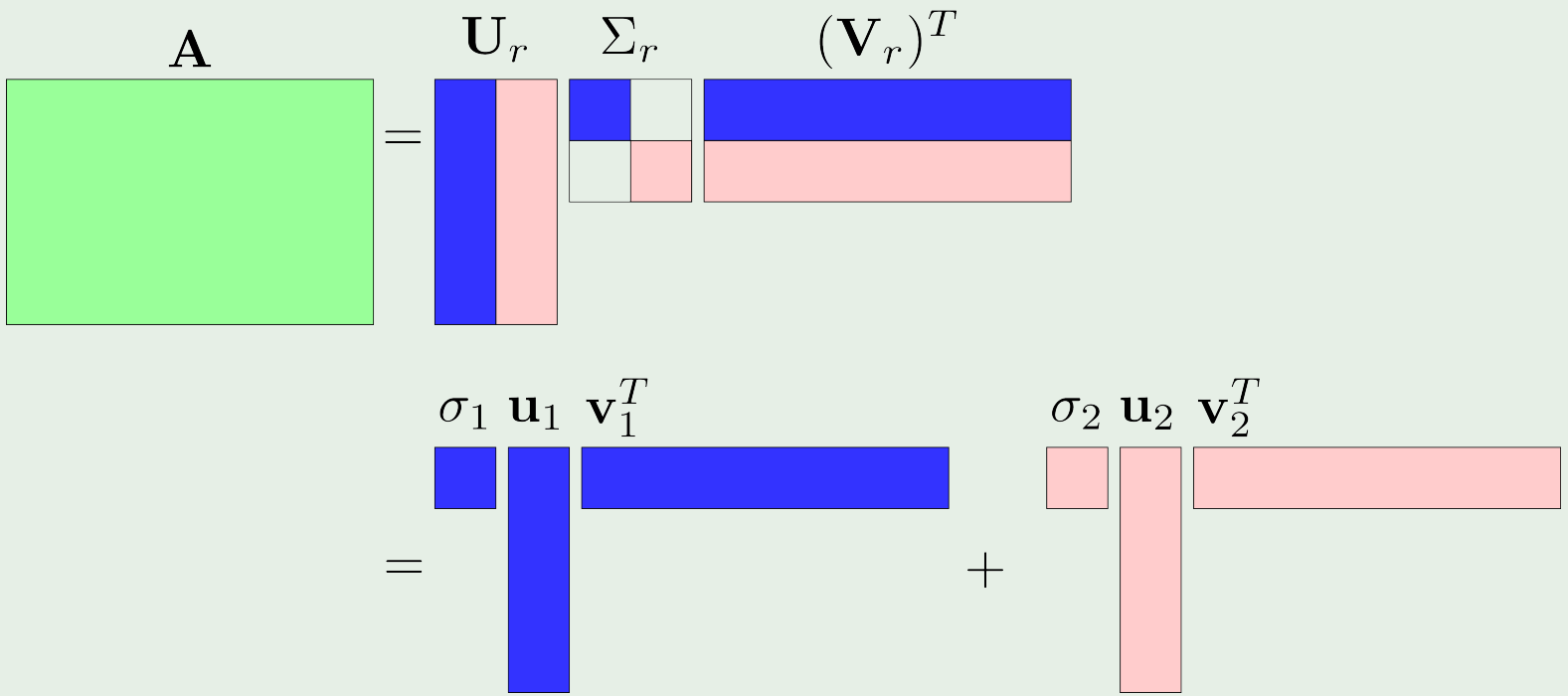
$$A = U_r \Sigma_r (V_r)^T$$

- U_r, V_r lần lượt là các ma trận tạo bởi r cột đầu tiên của U và V.
- Σ_r là ma trận tạo bởi r hàng đầu tiên và r cột đầu tiên của Σ

Singular Value Decomposition (SVD)

Example (Compact SVD)

Biểu diễn SVD dưới dạng thu gọn và dưới dạng tổng các ma trận rank 1 ($m = 4, n = 6, r = 2$)



Singular Value Decomposition (SVD)

Truncated SVD

Chú ý rằng trong ma trận Σ, các giá trị trên đường chéo không âm và giảm dần $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.
Thông thường, chỉ một lượng nhỏ các σ_i mang giá trị lớn, các giá trị còn lại thường rất nhỏ và gần 0. Khi đó, ta có thể xấp xỉ ma trận A bằng tổng của $k < r$ ma trận hạng 1:

$$A \approx A_k = U_k \Sigma_k (V_k)^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

Singular Value Decomposition (SVD)

Chúng ta đến với một định lý quan trọng của phương pháp này.

Theorem (Truncated SVD)

$$||A - A_k||_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Singular Value Decomposition (SVD)

Theorem (Truncated SVD)

$$||A - A_k||_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Nhận xét: Sai số của phép xấp xỉ trên chính là căn bậc hai của tổng bình phương các singular values mà ta đã bỏ qua ở phần cuối ma trận Σ. Ở đây sai số được định nghĩa là Frobenius norm của hiệu hai ma trận.

Proof of Truncated SVD theorem I

Theorem (Truncated SVD)

$$||A - A_k||_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Chứng minh. Bởi vì $||X||_F^2 = \text{trace}(XX^T)$ và $\text{trace}(XY) = \text{trace}(YX)$ với mọi ma trận X, Y cho nên chúng ta có các biến đổi:

$$||A - A_k||_F^2 = \left|\left| \sum_{i=k+1}^{r} \sigma_i u_i v_i^T \right|\right|_F^2 \tag{1}$$

$$= \text{trace}\left(\sum_{i=k+1}^{r} \sigma_i u_i v_i^T \right) \left(\sum_{j=k+1}^{r} \sigma_j u_j v_j^T \right)^T \tag{2}$$

Proof of Truncated SVD theorem II

$$= \text{trace} \sum_{i=k+1}^{r} \sum_{j=k+1}^{r} \sigma_i \sigma_j u_i v_i^T v_j u_j^T \tag{3}$$

$$= \text{trace} \sum_{i=k+1}^{r} \sigma_i^2 u_i u_i^T \tag{4}$$

$$= \text{trace} \sum_{i=k+1}^{r} \sigma_i^2 \tag{5}$$

$$= \sum_{i=k+1}^{r} \sigma_i^2 \tag{6}$$

Hoàn tất chứng minh. □

Singular Value Decomposition (SVD)

Nhận xét: Như vậy, sai số do xấp xỉ càng nhỏ nếu phần singular values bị truncated có giá trị càng nhỏ so với phần singular values được giữ lại. Đây là một định lý quan trọng giúp cho việc xấp xỉ ma trận dựa trên lượng thông tin muốn giữ lại.

Example

Nếu ta muốn giữ lại ít nhất 90% lượng thông tin trong A, trước hết ta tính $\sum_{i=\overline{1,r}}(\sigma_i^2)$. Sau đó chọn k nhỏ nhất sao cho

$$\sum_{i=1}^{k} \sigma_i^2 \geq 0.9 \sum_{j=1}^{r} \sigma_j^2$$

Một vài ứng dụng của SVD

Giảm chiều dữ liệu

Các ma trận A_k gần khít với A nên ta có thể dùng SVD để giảm chiều dữ liệu. Việc giảm chiều dữ liệu giúp ta có khả năng biểu diễn bộ dữ liệu đó một cách khá chính xác trên đồ thị.

Nén ảnh

Để lưu ảnh với Truncated SVD, ta lưu các ma trận $U_k \in \mathbb{R}^{m\times k}, \Sigma_k \in \mathbb{R}^{k\times k}, V_k \in \mathbb{R}^{n\times k}$. Tổng số phần tử phải lưu là $k(m+n+1)$. Giả sử mỗi phần tử được lưu bởi một số thực 4 byte, vậy số byte cần lưu trữ là $4k(m+n+1)$. Khi so giá trị này với ảnh gốc kích thước mn, mỗi giá trị là 1 số nguyên 1 byte thì ta có tỉ lệ nén là:

$$\frac{4k(m+n+1)}{mn}$$

Singular Value Decomposition (SVD)

Một vài ứng dụng của SVD

Recommendation System

Ý tưởng tương tự như nén ảnh, đó là căn cứ vào những cặp (user, item) đã được rating của Utility matrix để tìm ra một ma trận xấp xỉ tốt nhất và dự báo cho những cặp (user, item) chưa được rating.

Singular Value Decomposition (SVD)

Example

Xét biểu diễn ma trận $A = xy^\top$. Chúng ta dễ dàng nhận thấy đây là ma trận hạng 1 nên chỉ có duy nhất một trị riêng khác 0 là $\sigma_1 = ||x|| \cdot ||y||$. Các thành tố của SVD chính là

$$U = \frac{1}{||x||}x, V = \frac{1}{||y||}y, \Sigma = ||x|| \cdot ||y||$$

Singular Value Decomposition (SVD)

Example

Xét biểu diễn ma trận $A = xy^\top$. Chúng ta dễ dàng nhận thấy đây là ma trận hạng 1 nên chỉ có duy nhất một trị riêng khác 0 là $\sigma_1 = ||x|| \cdot ||y||$. Các thành tố của SVD chính là

$$U = \frac{1}{||x||}x, V = \frac{1}{||y||}y, \Sigma = ||x|| \cdot ||y||$$

Example

Tìm SVD của ma trận

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

Singular Value Decomposition (SVD)

Example

Tìm SVD của ma trận

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

Lời giải: Chúng ta biết rằng các singular values chính là căn bậc hai của các trị riêng của $AA^\top$ và $A^\top A$. Vì thế chúng ta sẽ tính

$$AA^\top = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}$$

Ma trận này có các trị riêng là 25 và 9. Nên các singular values của A là $\sigma_1 = 5, \sigma_2 = 3$. Hay là

$$\Sigma = \begin{pmatrix} 5 & 0 \\ 0 & 3 \end{pmatrix}$$

Các cột của U là các vector riêng đơn vị của $AA^\top$, có thể tìm ra bằng cách giải hệ

Singular Value Decomposition (SVD)

$$(AA^\top - 25I)u = \begin{pmatrix} -8 & 8 \\ 8 & -8 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$(AA^\top - 9I)u = \begin{pmatrix} 8 & 8 \\ 8 & 8 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Từ đó thu được

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Để ý rằng $\sigma_i v_i = A^\top u_i$, cho nên

$$V = A^\top U \Sigma^{-1} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1/3 \\ 1 & -1/3 \\ 0 & 4/3 \end{pmatrix}$$

SVD tối ưu hóa kết quả

Tại sao trị riêng và các singular value lại hữu ích trong thống kê?
Một trong những lý do đó là sự xuất hiện của chúng nhiều lúc chính là **kết quả** của các bài toán tối ưu hóa quan trọng.

Chẳng hạn, với $x \in \mathbb{R}^n$ là một biến ngẫu nhiên có $\mathbb{C}\text{ov}(x) = \Sigma \in \mathbb{R}^{n \times n}$.
Liệu rằng chúng ta có thể tìm được hình chiếu của x với phương sai cực đại hoặc cực tiểu? Tức là, chúng ta có thể tìm được a mà

$$\mathbb{V}\text{ar}(A^\top x) = a^\top \Sigma a$$

đạt cực đại hoặc cực tiểu?
Để có thể quản lý dễ dàng, chúng ta giả sử rằng $||a||_2 = 1$. Lúc này ta giải quyết bài toán tối ưu hóa sau

SVD tối ưu hóa kết quả

Bài toán

Cho a thỏa mãn $a^\top a = 1$. Tìm $\max(A^\top \Sigma A)$ và $\min(A^\top \Sigma A)$.

Ma trận Σ đối xứng, chúng ta có thể viết

$$\Sigma = V \Lambda V^\top$$

trong đó Λ là ma trận đường chéo, V là ma trận trực giao.
Nếu đặt $b = V^\top a$ thì

$$a^\top \Sigma a = b^\top \Lambda b = \sum_{i=1}^{n} (\lambda_i b_i^2)$$

lưu ý rằng $\lambda_1 \geq \lambda_2 \geq \cdots$ và

$$\sum_{i=1}^{n} b_i^2 = b^\top b = a^\top V V^\top a = a^\top a = 1$$

SVD tối ưu hóa kết quả

Ta nhận thấy giá trị **maximum** chính là λ_1 khi đặt $b = (1\ 0\ 0\ \cdots)^\top$.
Để ý rằng

$$a = VV^\top a = Vb = v_1$$

Nên giá trị lớn nhất đạt được khi $a = v_1$, các vector riêng của Σ tương ứng với trị riêng lớn nhất λ_1.
Tương tự, giá trị **minimum** là λ_n đạt được khi đặt $b = (0\ 0\ \cdots 0\ 1\ )^\top$, tương ứng với $a = v_n$.

SVD tối ưu hóa kết quả

Từ những lập luận ở trên, chúng ta đi đến 3 mệnh đề quan trọng

Mệnh đề 1

Với mọi ma trận đối xứng $\Sigma \in \mathbb{R}^{n \times n}$ và a thỏa mãn $a^\top a = 1$ thì

$$\max(a^\top \Sigma a) = \lambda_1, \quad \min(a^\top \Sigma a) = \lambda_n$$

Mệnh đề 2

Với mọi ma trận A thì $\max_{x:||x||_2=1} ||Ax||_2 = \max_x \frac{||Ax||_2}{||x||_2} = \sigma_1$

Mệnh đề 3

Với mọi ma trận A thì $\max_{a,b:||a||=||b||=1} (a^\top Ab) = \sigma_1$

Principal Component Analysis (PCA)

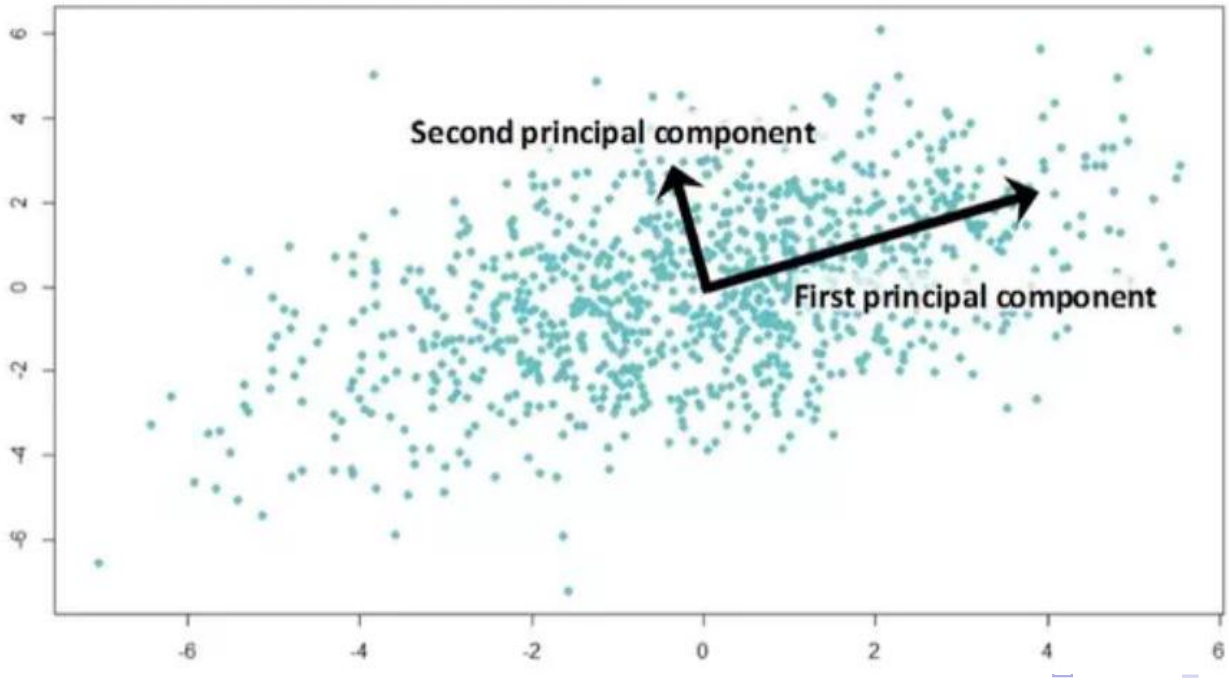
Giới thiệu

Principal Component Analysis (PCA, Pearson 1901) là một trong những nhóm kỹ thuật để lấy dữ liệu nhiều chiều và sử dụng các ràng buộc giữa các biến để biểu diễn nó ở dạng dễ xử lý hơn, *ít chiều hơn* mà không làm mất nhiều thông tin quan trọng.

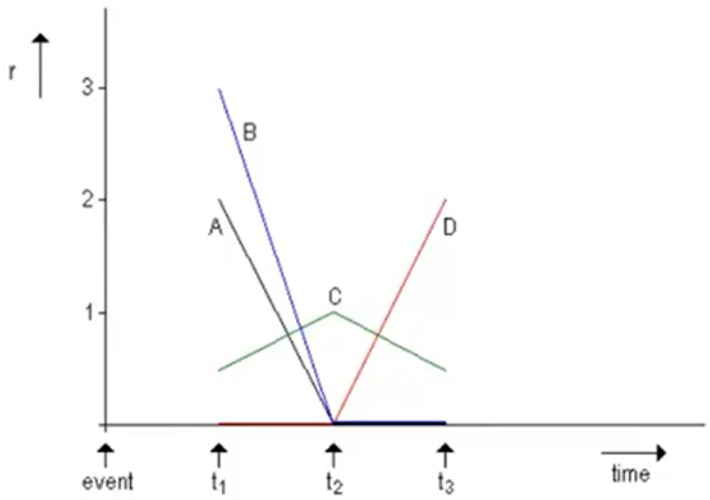
Principal Component Analysis (PCA)

Ý tưởng

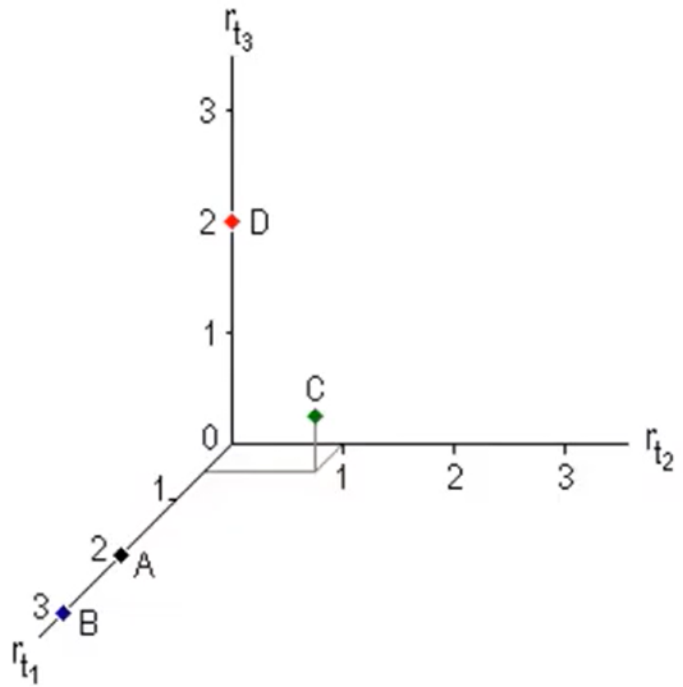
PCA còn được biết là một phương pháp trong thống kê, với mục đính tối đa hóa phương sai của các *transformed variable* y. Ta bắt đầu bằng cách chọn ra u_1 sao cho $y_1 = u_1^T x$ có phương sai lớn nhất. Sau đó, chọn u_2 sao cho $y_2 = u_2^T x$ có phương sai lớn nhất nhưng không tương quan với y_1, và tiếp tục như thế. PCA còn có mục đích là giảm thiểu độ lỗi.



Principal Component Analysis (PCA)



We are working with this representation:



gene	t_1	t_2	t_3
A	2	0	0
B	3	0	0
C	0.5	1	0.5
D	0	0	2

Principal Component Analysis (PCA)

Ý tưởng dưới góc nhìn toán học

Đặt $x_1, x_2, \cdots, x_n$ là các vector $p \times 1$ được tính toán trên n đơn vị thí nghiệm ban đầu. Ta có thể viết:

$$X = \begin{pmatrix} -x_1^T - \\ -x_2^T - \\ -\cdots- \\ -x_n^T - \end{pmatrix}$$

Ý tưởng dưới góc nhìn toán học

Đặt $x_1, x_2, \cdots, x_n$ là các vector $p \times 1$ được tính toán trên n đơn vị thí nghiệm ban đầu. Ta có thể viết:

$$X = \begin{pmatrix} -x_1^T - \\ -x_2^T - \\ - \cdots - \\ -x_n^T - \end{pmatrix}$$

Chú ý: Giả sử rằng X đã được tính toán theo giá trị trung bình của cột sao cho giá trị trung bình của mỗi cột bằng 0. Nếu không, ta có thể tính lại X bằng cách thay X bởi H_X, với H_X là *centering matrix*, hoặc thay tất cả giá trị x_i bởi $x_i - \overline{x}$.

Principal Component Analysis (PCA)

Ý tưởng dưới góc nhìn toán học

Ma trận hiệp phương sai của X (Đảm bảo chú ý trên) là:

$$S = \frac{1}{n} X^T X = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$$

Ý tưởng dưới góc nhìn toán học

Cho một số vector u, khi đó với tất cả các biến được chuyển đổi $y_i = u^T x_i$ có:

- Mean 0:

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i = \frac{1}{n}\sum_{i=1}^{n} u^T x_i = \frac{1}{n} u^T \sum_{i=1}^{n} x_i = 0$$

  do giá trị trung bình của $x_i = 0$.

- Có ma trận hiệp phương sai là $u^T S u$:

$$\frac{1}{n}\sum_{i=1}^{n} y_i^2 = \frac{1}{n}\sum_{i=1}^{n} (u^T x_i x_i^T u) = \frac{1}{n} u^T \sum_{i=1}^{n} x_i x_i^T u = u^T S u$$

Principal Component Analysis (PCA)

Ý tưởng dưới góc nhìn toán học

Chúng ta muốn tìm các vector đơn vị u ($||u|| = 1$) để tối đa hóa phương sai mẫu, $u^T Su$ trên các vector đó.
Vector riêng đầu tiên v_1 của S để tối đa hóa $u^T Su$ (cũng là right sigular vector đầu tiên của X) là:

$$\max_{u:||u||=1} u^T Su = v_1 Sv_1 = \lambda_1$$

với λ_1 là giá trị riêng lớn nhất của S.
Vì thế, *first principal component* của X là v_1, và *first transformed variable* là $y_1 = v_1^T x$. Áp dụng với tất cả n điểm, ta có giá trị mới là:

$$y_{i1} = v_1^T x_i$$

Ý tưởng dưới góc nhìn toán học

y_1 là *first transformed variable* để phương sai đạt giá trị tối đa. Tiếp tục ta sẽ chọn y_2 không tương quan với y_1. (Tại sao?)
Ta có hiệp phương sai mẫu giữa y_1 và $u_2^T x$ là:

$$s_{y_2 y_1} = \frac{1}{n} \sum_{i=1}^{n} u_2^T x_i x_i^T v_1 = u_2^T S v_1 = \lambda_1 u_2^T v_1$$

với v_1 là vector riêng của S.
Để y_2 không tương quan với y_1, ta sẽ chọn u_2 sao cho u_2 trực giao với v_1 ($u_2^T v_1 = 0$). Vì thế chọn u_2 là kết quả của bài toán tối ưu:

$$\max_u u^T S u \ (u^T v_1 = 0)$$

Ý tưởng dưới góc nhìn toán học

Giải pháp là lấy $u_2 = v_2$, suy ra được giá trị riêng của S là: $v_2^T S v_2 = \lambda_2$ *Second principal component* của X là v_2, và *second transformed variable* là $y_2 = v_2^T x$. Áp dụng với tất cả n điểm, ta có transformed variable mới là:

$$y_{i2} = v_2^T x_i$$

Cứ tiếp tục như thế, với lần biến đổi thứ j thì transformed variable mới thứ j sẽ là:

$$y_{ij} = v_j^T x_i$$

với v_j là vector riêng thứ j của S (cũng là right sugular vector thứ j của X).

Geometric interpretation

PCA chiếu các điểm x_i lên không gian con V. Các vector cơ sở cho không gian con này là các vector riêng của S (còn là các right sigular vector của X):

$$V = \text{span}(v_1, v_2, \cdots, v_r)$$

Khi đó, ma trận chiếu trực giao của V là:

$$P_V = VV^T$$

với $V^T V = I$.

Principal Component Analysis (PCA)

Geometric interpretation

Tọa độ các điểm được chiếu trên không gian con V (đối với các cơ sở của V) là các điểm thành phần chính (các transformed variable):

$$y_i = \begin{pmatrix} y_{i1} \\ \cdots \\ y_{ir} \end{pmatrix} = V^T x_i$$

với $V = \begin{pmatrix} | & & | \\ v_1 & \cdots & v_r \\ | & & | \end{pmatrix}$ là ma trận của right sigular vector từ SVD của X.

Các transformed variable là:

$$Y = \begin{pmatrix} -y_1^T - \\ - \cdots - \\ -y_n^T - \end{pmatrix}$$

Principal Component Analysis (PCA)

Geometric interpretation

Thay thế SVD cho $X = U\Sigma V^T$, ta có thể thấy các transformed variable matrix/principal component scores:

$$Y = U\Sigma$$

Y là ma trận kích thước $n \times r$, với $r < p$ thì ta đã giảm số chiều của X mà vẫn giữ được nhiều nhất các thông tin quan trọng.

Principal Component Analysis (PCA)

Mô tả chính

Cho các điểm $x_1, x_2, \cdots, x_n$ biểu thị một mẫu trong $\mathbb{R}^p$ với vector trung bình mẫu là $\overline{x}$, có ma trận hiệp phương sai S. Giả sử rằng $S = \frac{1}{n} X^T H_X$ có spectral decomposition:

$$A = V \Lambda V^T = \sum_{j=1}^{p} \lambda_j v_j v_j^T$$

khi các giá trị riêng thỏa mãn $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$ với $\Lambda = \text{diag}\{\lambda_1, \cdots, \lambda_p\}$, và V bao gồm các vector riêng của S. Nếu S đạt "full rank", $\lambda_p > 0$, còn nếu đạt được rank r, với $r < p$, thì $\lambda_{r+1} = \cdots = \lambda_p = 0$, chúng ta có thể giảm kích thước V và chỉ xem xét r cột đầu tiên.

Principal Component Analysis (PCA)

Mô tả chính

Các principal component của X được xác định tuần tự. Nếu v_k là vector của u sao cho tối đa hóa được phương sai ở lần tìm principal component thứ $k(k < j)$, thì thành principal component j là kết quả của vấn đề tối ưu sau:

$$\max_{u:||u||=1} u^T S u$$

với ràng buộc $v_k^T u = 0$ $(1 \leq k < j)$ (khi $j = 1$ thì không có ràng buộc này).

Mệnh đề 1

Giá trị lớn nhất của bài toán trên chính là λ_j, đạt được khi $u = v_j$.

Chứng minh. Ta có thể chứng minh bằng phương pháp nhân tử Lagrange. Với $j = 1$:

$$\mathcal{L} = u^T S u + \lambda(1 - u^T u)$$

Đặt đạo hàm theo u bằng 0:

$$2Su - 2\lambda u = 0$$

$$\Leftrightarrow Su = \lambda u \text{ với } u^T u = 1$$

Khi đó

$$uSu = \lambda$$

Ta phải chọn $u = v_1$, vector tương ứng với giá trị riêng lớn nhất của S.

Principal Component Analysis (PCA)

Tiếp tục như thế, chúng ta có được Lagrangian cho bài toán tìm thành phần chính thứ j:

$$\mathcal{L} = u^T S u + \lambda(1 - u^T u) + \sum_{k=1}^{j-1} \mu_k (0 - u^T v_k)$$

khi chúng ta có j Lagrange nhân tử $\lambda, \mu_1, \cdots, \mu_{j-1}$. Đặt đạo hàm theo u bằng 0:

$$2Su - 2\lambda u - \sum_{k=1}^{j-1} \mu_k v_k = 0$$

Khi nhân cả 2 bên cho v_l^T:

$$2v_l^T Su - 2\lambda v_l u - \sum_{k=1}^{j-1} \mu_k v_l^T v_k = 0$$

Principal Component Analysis (PCA)

Với v_l là một vector riêng của S và vì thế $Sv_l = \lambda_l v_l$, do đó $v_k Su = 0$, ${v_l}^T u = 0$. Cũng như:

$${v_l}^T v_k = \begin{cases} 1 \text{ nếu } k = l \\ 0 \text{ trong trường hợp còn lại} \end{cases}$$

Và chúng ta đã chỉ ra rằng $\mu_l = 0$ với $1 \leq l < j$. Vì thế, ta lại có:

$$Su = \lambda u$$

u bắt buộc là vector riêng đơn vị của S. Vì u phải trực giao với $v_1, \cdots, v_{j-1}$, và ${v_l}^T S v_l = \lambda_j$, chúng ta phải chọn $u = v_j$, vector riêng ứng với trị riêng lớn nhất thứ j.

Principal Component Analysis (PCA)

Tính chất

Với $j = 1, \cdots, p$, các scores của principal component thứ j là:

$$y_{ij} = v_j^T(x_i - \overline{x}), i = \overline{1, n}$$

Khi rank$(S) = r < p$, các scores của principal component thứ $r + 1, \cdots, p$ trở nên vô nghĩa.
Trong kí hiệu vector: $y_i = (y_{i1}, \cdots, y_{ip}) = V^T(x_i - \overline{x}), i = \overline{1, n}$
Ở dạng ma trận: $Y = [y_1, \cdots, y_n]^T = HXV$
Đặt $\tilde{X} = H_X$ là column centered data matrix với SVD $\tilde{X} = U\Sigma V^T$, thì

$$Y = \tilde{X}V = U\Sigma$$

Các transformed variable $Y = HXV$ có những tính chất quan trọng được tập hợp lại ở mệnh đề 2 dưới đây.

Mệnh đề 2

Các kết quả trên cho ta thấy:

1. Trung bình các vector mẫu của $y_1, \cdots, y_n$ là một vector không: $\overline{y} = 0_p$
2. Ma trận hiệp phương sai mẫu của $y_1, \cdots, y_n$ là: $\Lambda = diag\lambda_1, \cdots, \lambda_n$
3. $v_1^T S v_1 \geq v_2^T S v_2 \geq \cdots \geq v_p^T S v_p$
4. Tổng tất cả các phương sai mẫu bằng $trace(S)$: $\sum_{j=1}^{p} v_j^T S v_j = \sum_{j=1}^{p} \lambda_j = trace(S)$
5. Tích tất cả các phương sai mẫu bằng định thức của S:

$$\prod_{j=1}^{p} v_j^T S v_j = \prod_{j=1}^{p} \lambda_j = |S|$$

Principal Component Analysis (PCA)

Từ những tính chất này, chúng ta gọi tỉ số

$$\frac{\lambda_j}{\lambda_1 + \lambda_2 + ... + \lambda_p}$$

của sự thay đổi trong mẫu được "giải thích" bởi principal component thứ j.

Đây chính là mục tiêu ban đầu của chúng ta, đó là dữ liệu được "giải thích" nhiều nhất có thể.

Các bước thực hiện tính PCA 1

- Step 1: Chuẩn bị dữ liệu cần giảm chiều là X với kích thước (n_sample, n_feature), tương ứng mỗi hàng là 1 mẫu dữ liệu có n_feature thuộc tính
- Step 2: Trừ mỗi điểm dữ liệu cho vector kỳ vọng: X_k = X_k - X_{mean} với k = 1..n_sample và X_{mean} là vector trung bình của tất cả các điểm dữ liệu
- Step 3: Tính ma trận hiệu phương sai : S = $\frac{1}{n-sample} * X^T * X$
- Step 4: Tìm trị riêng, vector riêng của ma trận S
- Step 5: Lấy k trị riêng có giá trị lớn nhất, tạo ma trận U với các hàng là các vector riêng ứng với k trị riêng đã chọn
- Step 6: Ánh xạ không gian ban đầu sang không gian k chiều: X_{new} = X*U
- Note: Nếu không hiểu phép nhân ở Step 6 bạn có thể lấy từng mẫu dữ liệu nhân với từng vector riêng, khi đó mỗi mẫu dữ liệu ban đầu sẽ được nhân với k vector nên sẽ có k chiều.

3. Python implement:

Giả sử đã có ma trận dữ liệu X, mình sẽ thực hiện lần lược từ Step 2 đến Step 6 cho mọi người tiện theo giỏi nhé:

- Step 2: Tính vector trung bình, sau đó trừ các điểm dữ liệu cho vector đó

```
mean = np.mean(X, axis=0)
X = X - mean
```

- Step 3: Tìm ma trận hiệp phương sai

```
cov = X.T.dot(X) / X.shape[0]
```

- Step 4: Tính trị riêng, vector riêng

```
eigen_values, eigen_vectors, = np.linalg.eig(cov)
```

Các bước thực hiện tính PCA 3

- Step 4: Tính trị riêng, vector riêng

```
eigen_values, eigen_vectors, = np.linalg.eig(cov)
```

- Step 5: Ở bước này mình sẽ lấy chỉ số index của trị riêng từ lớn đến nhỏ, rồi chọn k vector riêng tạo ma trận U tương ứng với k index đã tìm được

```
select_index = np.argsort(eigen_values)[::-1][:k]
U = eigen_vectors[:, select_index]
```

- Step 6: Ánh xạ dữ liệu sang không gian mới

```
X_new = X.dot(U)
```

Principal Component Analysis (PCA)

Bản chất của Principal Component Analysis

Ngoài việc giảm chiều dữ liệu, chúng ta biết PCA còn có mục đích là giảm độ lỗi. Nhưng chúng ta nhận ra là mục đích giảm độ lỗi tương đương với mục đích tối đa hóa phương sai. Vì vậy các bài toán về tối ưu dựa trên phương sai có liên quan mật thiết đến PCA.

Principal Component Analysis (PCA)

Ứng dụng

- Nén ảnh: Hình ảnh có thể được thay đổi để xử lý thuận tiện.
- Nhận dạng: Sử dụng EigenFaces để nhận dạng khuôn mặt là một kỹ thuật phổ biến trong thị giác máy tính. PCA là trọng tâm của phương pháp này vì tập hợp các EigenFaces được tạo ra bởi PCA. Sirovich và Kirby (1987) đã chỉ ra rằng PCA có thể được sử dụng trên một bộ dữ liệu hình ảnh khuôn mặt để tạo thành một tập hợp các đặc trưng cơ bản.
- Nghiên cứu mức độ hài lòng của người dùng về một sản phẩm dựa trên các đặc trưng chính được rút ra từ tập hợp những hành vi/đánh giá của người dùng đó.
- PCA cũng được ứng dụng trong lĩnh vực y tế, nơi có nhiều nguồn dữ liệu có tương quan với nhau như tuổi thọ, căn bệnh...

Quan điểm khác về PCA

Xét mẫu $x_1, x_2, ..., x_n \in \mathbb{R}^p$ có kỳ vọng 0 (thay x_i bởi $x_i - \overline{x}$ nếu kỳ vọng khác 0).
Tóm lại, trong phương pháp PCA, để tìm r principal component đầu tiên, chúng ta phải giải quyết bài toán tối ưu hóa

Problem

$$\forall k = \overline{1, r} \text{ và } u_k^\top u_j = \begin{cases} 1 \text{ nếu } j = k \\ 0 \text{ ngược lại} \end{cases} . \text{ Cực đại hóa các } u_k^\top S u_k .$$

Quan điểm khác về PCA

Xét mẫu $x_1, x_2, ..., x_n \in \mathbb{R}^p$ có kỳ vọng 0 (thay x_i bởi $x_i - \overline{x}$ nếu kỳ vọng khác 0).
Tóm lại, trong phương pháp PCA, để tìm r principal component đầu tiên, chúng ta phải giải quyết bài toán tối ưu hóa

Problem

$$\forall k = \overline{1, r} \text{ và } u_k^\top u_j = \begin{cases} 1 \text{ nếu } j = k \\ 0 \text{ ngược lại} \end{cases} . \text{ Cực đại hóa các } u_k^\top S u_k .$$

Bởi vì

$$\text{trace}(U^\top S U) = \sum_{k=1}^{r} (u_k^\top S u_k)$$

cho nên bài toán tương đương với việc cực đại hóa $\text{trace}(U^\top S U)$ với điều kiện $U^\top U = I_r$.

Quan điểm khác về PCA

Chúng ta cần tìm best rank-r xấp xỉ tuyến tính cho ma trận dữ liệu $X = (x_1 \cdots x_n)^\top$ (giả thiết rằng dữ liệu được căn giữa cột, nếu không thì thay X bởi HX như phần trước).
Một hướng đi tự nhiên đó là tìm một ma trận $U \in \mathbb{R}^{p \times r}$ có hạng r sao cho model

$$f(y) = Uy$$

có thể được sử dụng để biểu diễn dữ liệu.
Chọn các $y_i \in \mathbb{R}^r$ và U để cực tiểu tổng bình phương độ lỗi

$$\sum_{i=1}^{n} ||x_i - Uy_i||_2^2$$

Để ý rằng nếu viết

$$Y^\top = \begin{pmatrix} | & & | \\ y_1 & \cdots & y_n \\ | & & | \end{pmatrix}$$

Quan điểm khác về PCA

thì

$$\sum_{i=1}^{n} ||x_i - Uy_i||_2^2 = \text{trace}((X^\top - UY^\top)^\top (X^\top - UY^\top))$$

$$= ||X^\top - UY^\top||_F^2$$

Chúng ta tìm ma trận X_r hạng r để cực tiểu

$$||X - X_r||_F = ||X^\top - X_r^\top||_F$$

Lưu ý rằng có thể biểu diễn ma trận hạng r bất kỳ dưới dạng $X_r^\top = UY^\top$ với $U \in \mathbb{R}^{p \times r}$, $Y \in \mathbb{R}^{n \times r}$.
Bởi vì $u \in \mathrm{C}(U)$ ($\mathrm{C}(U)$ là không gian cột của U) nên giá trị cực tiểu của $||x - u||_2$ là phép chiếu vuông góc từ x xuống $\mathrm{C}(U)$. Đó là $u = UU^\top x$.
Từ đó thu được $X_r^\top = UU^\top X^\top$ và $Y^\top = U^\top X^\top$.

Quan điểm khác về PCA

Việc còn lại là tìm U đủ tốt bằng cách cực tiểu hóa ($U^\top U = I_r$)

$$\begin{aligned} ||X^\top - UU^\top X^\top||_F^2 &= ||X - XUU^\top||_F^2 \\ &= \text{trace}((X - XUU^\top)^\top (X - XUU^\top)) \\ &= \text{trace}(X^\top X) - \text{trace}(U^\top X^\top XU) \end{aligned}$$

Cực tiểu hóa biểu thức trên tương đương với cực đại hóa $\text{trace}(U^\top SU)$.
Bài toán này đã được giới thiệu trước đó.
Tóm lại, bài toán tối ưu

$$\textbf{Minimize } ||X^\top - UU^\top X^\top||_F, \textbf{ s.t } U^\top U = I_r$$

chính là bài toán tối ưu PCA.

Mối liên hệ giữa SVD và PCA

Giả sử rằng dữ liệu đã được tiền xử lý và có kỳ vọng bằng 0.
Nói một cách đơn giản, PCA yêu cầu tính trị riêng và vector riêng của ma trận hiệp phương sai, tức là tính

$$\frac{1}{n-1}XX^\top$$

trong đó X là ma trận dữ liệu.
Bởi vì ma trận hiệp phương sai là ma trận đối xứng, cụ thể là ma trận đường chéo, các vector riêng có thể được normalize để chúng trực giao

$$\frac{1}{n-1}XX^\top = \frac{1}{n-1}WDW^\top$$

Mặt khác, khai triển SVD đối với X, nghĩa là $X = U\Sigma V^\top$.
Chúng ta xây dựng ma trận hiệp phương sai từ phép phân tích này.

Mối liên hệ giữa SVD và PCA

$$\frac{1}{n-1}XX^\top = \frac{1}{n-1}(U\Sigma V^\top)(U\Sigma V^\top)^\top = \frac{1}{n-1}(U\Sigma V^\top)(V\Sigma U^\top)$$

Bởi vì V trực giao cho nên

$$\frac{1}{n-1}XX^\top = \frac{1}{n-1}U\Sigma^2 U^\top$$

Điều này ngụ ý rằng bình phương các trị riêng của $XX^\top$ là các singular value của X.

Nói tóm lại, việc sử dụng SVD để hiện thực PCA sẽ tốt hơn về mặt số lượng tính toán so với khi bắt đầu từ việc tạo ra một ma trận hiệp phương sai, bởi vì tạo $XX^\top$ có thể không bảo toàn được độ chính xác.

Sự khác nhau giữa PCA và LDA

Principal Component Analysis (PCA) và **Linear Discriminant Analysis (LDA)** là hai trong số các kỹ thuật giảm chiều phổ biến nhất. Cả hai phương pháp đều được sử dụng để giảm số lượng đặc trưng trong tập dữ liệu trong khi vẫn giữ lại càng nhiều thông tin càng tốt.

Ngoài ra, LDA là kỹ thuật giảm chiều **có giám sát**, thông qua đó phân loại được dữ liệu. Còn PCA là kỹ thuật giảm chiều **không giám sát**, mặc dù vậy nhưng nó bỏ qua các nhãn.

Cụ thể hơn, sự khác biệt chính là:

- LDA tập trung vào việc tìm kiếm một không gian con để **tối đa hóa khả năng phân tách** giữa các nhóm.
- LCA tập trung vào việc tìm hướng để **tối đa hóa phương sai** tập dữ liệu.

Sự khác nhau giữa PCA và LDA

Như vậy, khi nào chúng ta nên sử dụng PCA, khi nào nên sử dụng LDA?

Nói chung, ta nên sử dụng LDA khi mục tiêu hướng đến là phân loại, nghĩa là khi chúng ta đã có nhãn cho các điểm dữ liệu và muốn dự đoán điểm mới sẽ có nhãn nào dựa trên đặc trưng của chúng. Mặt khác, nếu ta không có nhãn hoặc mục tiêu hướng tới không chỉ đơn giản là phân loại, thì PCA có thể sẽ hoạt động tốt hơn.

Tuy nhiên, có một số tình huống LDA có thể hoạt động tốt hơn PCA kể cả không phải vấn đề phân loại.

Chẳng hạn, nếu dữ liệu có 100 đặc trưng nhưng chỉ 10% trong số đó thực sự cung cấp thông tin (còn lại là nhiễu). Nếu chạy PCA trên tập dữ liệu này, thuật toán sẽ xác định tất cả 100 thành phần vì mục tiêu của nó chỉ đơn giản là tối đa hóa phương sai.

Sự khác nhau giữa PCA và LDA

Chỉ 10% trong số các thành phần hữu ích, còn lại 90% là vô dụng. Cho nên khi chạy LDA, nó sẽ chỉ xác định 10 thành phần vì mục tiêu tối đa khả năng phân tách lớp và loại bỏ thành phần gây nhiễu.

Do đó LDA sẽ cho ra kết quả tốt hơn ngay cả khi mục tiêu không phải là phân loại.

Storing the Presentation

GitHub

Bài thuyết trình cùng toàn bộ demo code được lưu trữ tại GitHub.

Tài liệu tham khảo

- Vũ Hữu Tiệp, *Machine Learning cơ bản*.
- Richard Wilkinson, *Multivariate Statistics*.
- Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong, *Mathematical for Machine Learning*.
- Heremy Kun, *A Programmer's Introduction to Mathematics*.

The End