MACHINE LEARNING

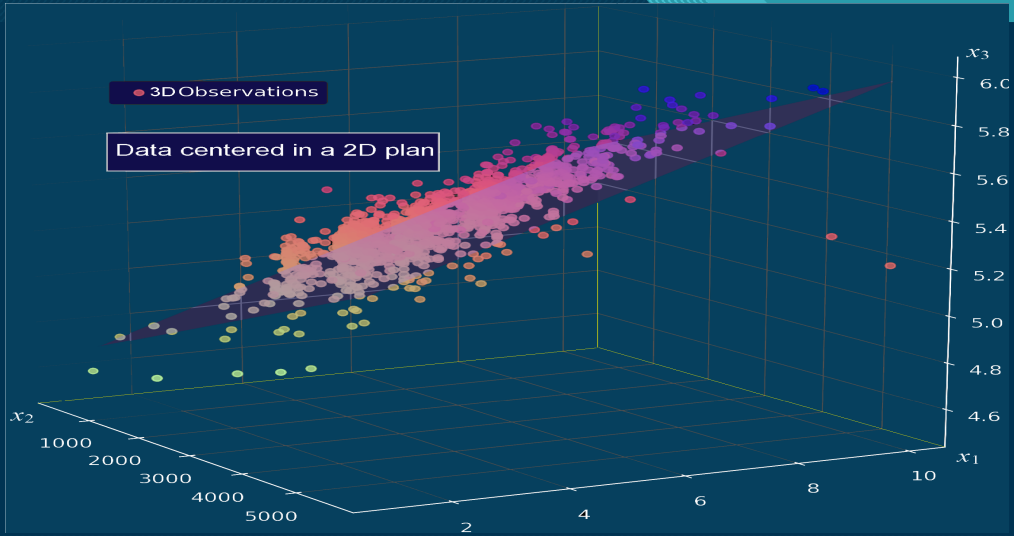
Cao Văn Chung
cvanchung@hus.edu.vn
Informatics Dept., MIM, HUS, VNU Hanoi

Principal Component Analysis

Dimensionality Reduction

- Dimensionality Reduction - Giảm chiều dữ liệu, là một trong những kỹ thuật quan trọng trong Machine Learning.
- Các bài toán thực tế có thể có số chiều của feature vectors rất lớn, lên tới hàng nghìn chiều. Hơn nữa số lượng điểm dữ liệu cũng rất lớn.
- Thực hiện lưu trữ và tính toán trực tiếp trên dữ liệu có số chiều cao sẽ rất khó khăn và tốn kém, tốc độ thấp.
- Giảm số chiều dữ liệu là một bước quan trọng và cần thiết. Đây cũng được coi là một phương pháp nén dữ liệu.

Dimensionality Reduction



Dimensionality Reduction

- Dimensionality Reduction, có thể quy về việc đi tìm một hàm số với đầu vào là một điểm dữ liệu ban đầu $x \in \mathbb{R}^d$ với d rất lớn, và tạo ra một điểm dữ liệu mới $z \in \mathbb{R}^k$ với $k < d$, mà các yêu cầu về thông tin của x không mất mát quá lớn.
- Ta sẽ chỉ tìm hiểu phương pháp đơn giản nhất trong các thuật toán Dimensionality Reduction dựa trên một mô hình tuyến tính:

  Principal Component Analysis (PCA), tức **Phân tích thành phần chính**.
- PCA dựa trên quan sát rằng dữ liệu thường không phân bố ngẫu nhiên trong không gian mà thường phân bố gần các đường/mặt đặc biệt nào đó.
- PCA xem xét một trường hợp đặc biệt khi các mặt đặc biệt đó có dạng tuyến tính là các không gian con (subspace).

Dimensionality Reduction

- **Ý tưởng**: Đơn giản nhất để giảm chiều dữ liệu từ d chiều về $k < d$ chiều là chỉ giữ lại k chiều (features) phần tử quan trọng nhất.
- Tuy nhiên để hiện thực hóa, cần vượt qua:
  - Làm thế nào để xác định thành phần nào là quan trọng hơn.
  - Trường hợp xấu: lượng thông tin mà mỗi thành phần mang là như nhau, bỏ đi thành phần nào cũng dẫn đến việc mất một lượng thông tin lớn.

Giải pháp với mô hình tuyến tính:

Tìm một hệ cơ sở mới mà trong đó, biểu diễn của các vector dữ liệu ban đầu sẽ có sự khác nhau rõ rệt về tầm quan trọng giữa các thành phần. Lúc đó ta có thể bỏ qua những thành phần ít quan trọng nhất và thông tin ít bị mất mát.

Fundamental knowledge

Người đọc cần bổ sung những kiến thức sau

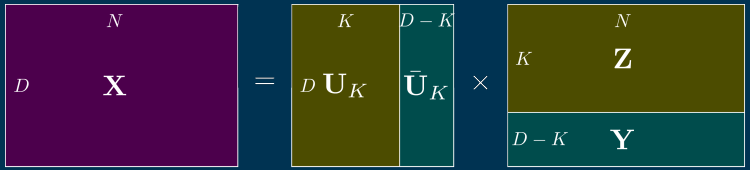
- Chuẩn 2 của ma trận ($\|A\|_2$ với $A \in \mathbb{R}^{m \times n}$).
- Hệ cơ sở của không gian tuyến tính định chuẩn, có tích vô hướng.
  - Biểu diễn vector trong các hệ cơ sở khác nhau.
  - Cơ sở trực chuẩn.
- Trace - vết của ma trận.
- Kỳ vọng và ma trận hiệp phương sai.
- Khai triển kì dị - Singular Value Decomposition (SVD)
  - Vector riêng, giá trị riêng; Khai triển kì dị - Singular Value Decomposition
  - Khai triển kì dị chặt cụt - Truncated SVD
  - Khai triển kì dị thu gọn - Compact SVD
  - Số chiều tối ưu - Optimal Rank k Approximation

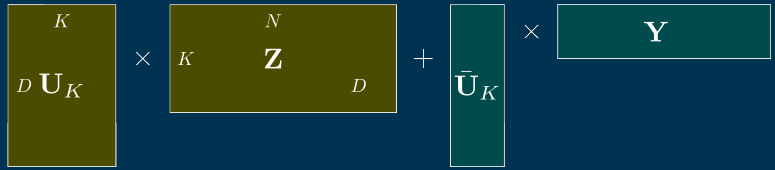
Principal Component Analysis

- Principal Component Analysis - PCA: Tìm một hệ cơ sở mới sao cho: thông tin của dữ liệu tập trung ở một số toạ độ, phần còn lại có ít thông tin.

  Để đơn giản trong tính toán, PCA sẽ tìm một hệ trực chuẩn để làm cơ sở mới.
- Giả sử hệ cơ sở trực chuẩn mới là $U = \{u_1, u_2, \ldots, u_d\}$ và chúng ta muốn giữ lại $k < d$ toạ độ trong hệ cơ sở mới này.
- Không mất tính tổng quát, ta luôn có thể giả sử đó là k thành phần đầu tiên.

Principal Component Analysis





Principal Component Analysis

- Trong hình minh họa, hệ cơ sở mới $\mathbf{U} = [\mathbf{U}_k, \bar{\mathbf{U}}_k]$ là hệ trực chuẩn với $\mathbf{U}_k$ là ma trận con tạo bởi k cột đầu tiên của $\mathbf{U}$.

  Trong hệ cơ sở mới, ma trận dữ liệu có thể được viết thành

$$\mathbf{X} = \mathbf{U}_k\mathbf{Z} + \bar{\mathbf{U}}_k\mathbf{Y}$$

- Từ đó suy ra

$$\begin{bmatrix}\mathbf{Z}\\ \mathbf{Y}\end{bmatrix} = \begin{bmatrix}\mathbf{U}_k^T\\ \bar{\mathbf{U}}_k^T\end{bmatrix}\mathbf{X} \Rightarrow \begin{matrix}\mathbf{Z} = \mathbf{U}_k^T\mathbf{X}\\ \mathbf{Y} = \bar{\mathbf{U}}_k^T\mathbf{X}\end{matrix} \quad (1)$$

- Mục đích của PCA là đi tìm ma trận trực giao $\mathbf{U}$ sao cho phần lớn thông tin được giữ lại ở phần $\mathbf{U}_k\mathbf{Z}$ và có thể lược bỏ phần $\bar{\mathbf{U}}_k\mathbf{Y}$.
- Cụ thể tìm cách thay $\mathbf{Y}$ bằng một ma trận xấp xỉ $\bar{\mathbf{Y}}$ có toàn bộ các cột như nhàu, không phụ thuộc dữ liệu test (có thể phụ thuộc dữ liệu training).

Principal Component Analysis

- Gọi mỗi cột đó là $\mathbf{b}$ và có thể coi nó là **bias**, khi đó, ta sẽ xấp xỉ $\mathbf{Y} \approx \mathbf{b}\mathbf{1}^T$, với $\mathbf{1}^T \in \mathbb{R}^{1\times N}$ là vector hàng có toàn bộ các phần tử bằng 1.

  Giả sử đã tìm được $\mathbf{U}$, ta cần tìm $\mathbf{b}$ thoả mãn

$$\mathbf{b} = \text{argmin}_{\mathbf{b}} \|\mathbf{Y} - \mathbf{b}\mathbf{1}^T\|_F^2 = \text{argmin}_{\mathbf{b}} \|\bar{\mathbf{U}}_k^T\mathbf{X} - \mathbf{b}\mathbf{1}^T\|_F^2.$$

- Giải phương trình đạo hàm theo $\mathbf{b}$ của hàm mục tiêu bằng 0:

$$(\mathbf{b}\mathbf{1}^T - \bar{\mathbf{U}}_k^T\mathbf{X})\mathbf{1} = 0 \Rightarrow N\mathbf{b} = \bar{\mathbf{U}}_k^T\mathbf{X}\mathbf{1} \Rightarrow \mathbf{b} = \bar{\mathbf{U}}_k^T\bar{\mathbf{x}}.$$

- Dễ thấy, nếu vector kỳ vọng $\bar{\mathbf{x}} = \mathbf{0}$, việc tính toán sẽ thuận tiện hơn nhiều.

Principal Component Analysis

- Có thể đạt được kỳ vọng $\bar{\mathbf{x}} = \mathbf{0}$, nếu ngay từ đầu, ta trừ mỗi vector dữ liệu đi vector kỳ vọng $\bar{\mathbf{x}}$ của toàn bộ dữ liệu.

  Với giá trị $\mathbf{b}$ tìm được này, dữ liệu ban đầu sẽ được xấp xỉ với:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_k\mathbf{Z} + \bar{\mathbf{U}}_k\bar{\mathbf{U}}_k^T\bar{\mathbf{x}}\mathbf{1}^T.$$

- Kết hợp các phương trình trên, ta định nghĩa hàm tổn thất như sau:

$$J = \frac{1}{N}||\mathbf{X} - \tilde{\mathbf{X}}||_F^2 = \frac{1}{N}||\bar{\mathbf{U}}_k\bar{\mathbf{U}}_k^T\mathbf{X} - \bar{\mathbf{U}}_k\bar{\mathbf{U}}_k^T\bar{\mathbf{x}}\mathbf{1}^T||_F^2. \quad (2)$$

- Chú ý: Nếu các cột của một ma trận $\mathbf{V}$ tạo thành một hệ trực chuẩn thì với một ma trận $\mathbf{W}$ bất kỳ, ta luôn có:

$$\|\mathbf{VW}\|_F^2 = \text{trace}(\mathbf{W}^T\mathbf{V}^T\mathbf{VW}) = \text{trace}(\mathbf{W}^T\mathbf{W}) = \|\mathbf{W}\|_F^2.$$

Principal Component Analysis

- Vì vậy hàm tổn thất trong công thức (2) có thể viết lại thành:

$$J = \frac{1}{N}\|\bar{\mathbf{U}}_k^T(\mathbf{X} - \bar{\mathbf{x}}\mathbf{1})^T\|_F^2 = \frac{1}{N}\|\bar{\mathbf{U}}_k^T\hat{\mathbf{X}}\|_F^2 = \frac{1}{N}\|\hat{\mathbf{X}}^T\bar{\mathbf{U}}_k\|_F^2$$
$$= \frac{1}{N}\sum_{i=K+1}^{D}\|\hat{\mathbf{X}}^T\mathbf{u}_i\|_2^2 = \frac{1}{N}\sum_{i=K+1}^{D}\mathbf{u}_i^T\hat{\mathbf{X}}\hat{\mathbf{X}}^T\mathbf{u}_i = \sum_{i=K+1}^{D}\mathbf{u}_i^T\mathbf{S}\mathbf{u}_i. \quad (3)$$

- Ở đây $\mathbf{S}$ là ma trận hiệp phương sai của dữ liệu và $\hat{\mathbf{X}} = \left(\hat{x}_1, \ldots, \hat{x}_N\right)$, trong đó $\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}_n$, $n = 1, \ldots, N$.
- Viết dạng ma trận $\hat{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{x}}\mathbf{1}^T$ và $\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$. Chú ý $\mathbf{S}$ là ma trận bán xác định dương.

Principal Component Analysis

- Chú ý hệ $\{\mathbf{u}_i\}$, $i = 1, \ldots, d$ trực chuẩn, cùng với tính đối xứng, nửa xác định dương của $\mathbf{S}$, ta có

$$\begin{aligned} L &= \sum_{i=1}^{D} \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i = \frac{1}{N} \|\hat{\mathbf{X}}^T \mathbf{U}\|_F^2 = \frac{1}{N} \text{trace}(\hat{\mathbf{X}}^T \mathbf{U} \mathbf{U}^T \hat{\mathbf{X}}) \\ &= \frac{1}{N} \text{trace}(\hat{\mathbf{X}}^T \hat{\mathbf{X}}) = \frac{1}{N} \text{trace}(\hat{\mathbf{X}} \hat{\mathbf{X}}^T) = \text{trace}(\mathbf{S}) = \sum_{i=1}^{D} \lambda_i \end{aligned} \tag{4}$$

- Ở đây $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$ là các giá trị riêng của ma trận đối xứng nửa xác định dương $\mathbf{S}$, do đó chúng là các số thực không âm.

Principal Component Analysis

- Từ đẳng thức cuối của (4) suy ra L không phụ thuộc vào hệ cơ sở $\mathbf{U}$ mà chỉ phụ thuộc bản thân dữ liệu. Cụ thể hơn, L chính là tổng của các phương sai theo từng thành phần của dữ liệu ban đầu.
- Bài toán cực tiểu hóa hàm tổn thất J trong (2) tương đương với cực đại hóa

$$F = L - J = \sum_{i=1}^{k} \mathbf{u}_i \mathbf{S} \mathbf{u}_i^T.$$

Principal Component Analysis

Định lý

F đạt giá trị lớn nhất $\max F = \sum_{i=1}^{k} \lambda_i$ khi các vector riêng $\{\mathbf{u}_i\}$ ứng với giá trị riêng $\{\lambda_i\}$ lập thành một hệ trực chuẩn: $\{\mathbf{u}_i\}$ trực giao và $\|\mathbf{u}_i\|_2 = 1$, $i = 1, \ldots, k$.

- Định lý trên có thể được chứng minh bằng quy nạp theo k.
- Giá trị riêng lớn nhất λ_1 được gọi là *Thành phần chính thứ nhất* (First Principal Component); trị riêng thứ hai λ_2 còn được gọi là *Thành phần chính thứ hai*.v.v.
- Chính vì vậy, phương pháp này có tên gọi là *Phân tích thành phần chính* - Principal Component Analysis.

Principal Component Analysis

- **Các bước của phương pháp PCA**
  1. Tính vector kỳ vọng của toàn bộ dữ liệu:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

  2. Tính dữ liệu chuẩn hóa $\hat{\mathbf{x}}_n$

$$\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}; \qquad n = 1, 2, \ldots, N.$$

  3. Tính ma trận hiệp phương sai:

$$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

Principal Component Analysis

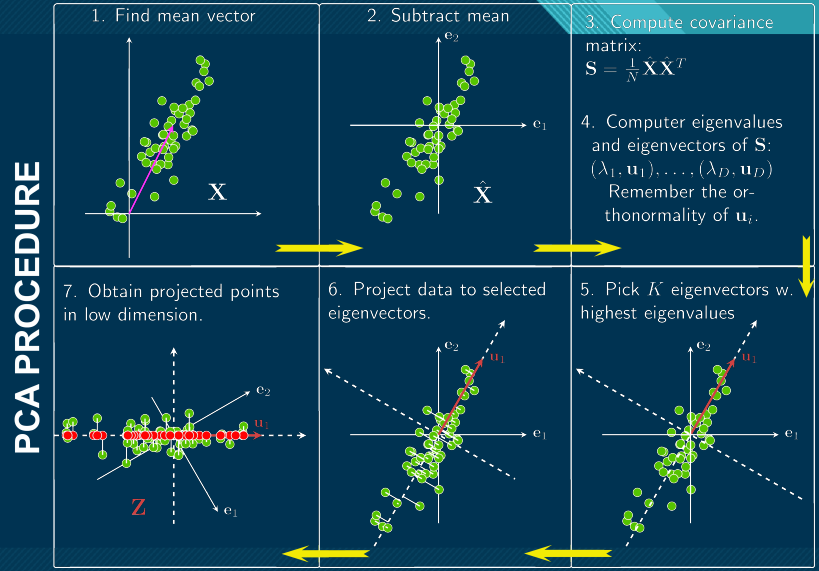
- **Các bước của phương pháp PCA**
  4. Tính các trị riêng λ_i và vector riêng $\mathbf{u}_i$ có $\|\mathbf{u}_i\|_2 = 1$ của ma trận này, sắp xếp chúng theo thứ tự giảm dần của trị riêng.
  5. Chọn k giá trị riêng lớn nhất và k vector riêng trực chuẩn tương ứng. Xây dựng ma trận $\mathbf{U}_k$.
  6. Các cột của $\{\mathbf{U}_i\}_{i=1}^{k}$ là hệ cơ sở trực chuẩn, tạo thành không gian con của k thành phần chính, it gần với phân bố của dữ liệu ban đầu đã chuẩn hoá.
  7. Chiếu dữ liệu ban đầu đã chuẩn hoá $\hat{\mathbf{X}}$ xuống không gian con nói trên. Dữ liệu mới chính là toạ độ của các điểm dữ liệu trên không gian mới

$$\mathbf{Z} = \mathbf{U}_k^T \hat{\mathbf{X}}.$$

- Dữ liệu ban đầu được xấp xỉ theo dữ liệu mới

$$\mathbf{x} \approx \mathbf{U}_k \mathbf{Z} + \bar{\mathbf{x}}.$$

Principal Component Analysis



PCA & Truncated SVD

- Cho ma trận $\mathbf{X} \in \mathbb{R}^{d \times N}$, xét bài toán xấp xỉ X bởi A với $\text{rank}(A) \leq k$

$$\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{A}\|_F \qquad \text{s.t.} \quad \text{rank}(\mathbf{A}) = K.$$

- Giả sử SVD của $\mathbf{X}$ là $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, với $\mathbf{U} \in \mathbb{R}^{D \times D}$ và $\mathbf{V} \in \mathbb{R}^{N \times N}$ là trực giao; $\Sigma \in \mathbb{R}^{D \times N}$ là ma trận đường chéo (không nhất thiết vuông) và các phần tử trên đường chéo không âm, giảm dần.
- Lúc đó nghiệm của bài toán xấp xỉ trên sẽ là

$$\mathbf{A} = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T$$

với $\mathbf{U} \in \mathbb{R}^{d \times k}$ và $\mathbf{V} \in \mathbb{R}^{N \times k}$ là k cột đầu tiên của $\mathbf{U}$, $\mathbf{V}$; $\Sigma_k \in \mathbb{R}^{k \times k}$ là ma trận đường chéo con ứng với k hàng, k cột đầu tiên của Σ.

PCA & Truncated SVD

Xét phương pháp PCA: Xét dữ liệu chuẩn hóa $\hat{\mathbf{X}}$, lúc đó kỳ vọng $\bar{\mathbf{x}} = 0$.

- Nghiệm xấp xỉ của PCA trở thành

$$\hat{\mathbf{X}} \approx \tilde{\mathbf{X}} = \mathbf{U}_k \mathbf{Z}.$$

- Bài toán tối ưu của PCA sẽ trở thành

$$[\mathbf{U}_k, \mathbf{Z}] = \min_{\mathbf{U}_k, \mathbf{Z}} \|\hat{\mathbf{X}} - \mathbf{U}_k \mathbf{Z}\|_F \quad \text{s.t.:} \quad \mathbf{U}_k^T \mathbf{U}_k = \mathbf{I}_k$$

  với $\mathbf{I}_k \in \mathbb{R}^{k \times k}$ là ma trận đơn vị trong không gian các ma trận vuông k chiều; điều kiện ràng buộc chính là hệ $\mathbf{U}_k$ trực chuẩn.

Dễ dàng thấy rằng, với dữ liệu được chuẩn hóa $\hat{\mathbf{X}}$ ($\bar{\mathbf{x}} = 0$), nghiệm bài toán tối ưu trong phương pháp PCA chính là nghiệm Truncated SVD của ma trận dữ liệu (với các điểm dữ liệu ứng với các cột, các điểm được viết thành hàng trong ma trận).

Optimal Dim k

- Số chiều k được xác định dựa trên lượng thông tin muốn giữ lại.
- PCA còn được gọi là phương pháp *tối đa tổng phương sai được giữ lại*.
  - Có thể coi tổng các phương sai được giữ lại là lượng thông tin được giữ lại.
  - Với phương sai càng lớn, tức dữ liệu có độ phân tán cao, thể hiện lượng thông tin càng lớn.
- Nhắc lại

$$L = \sum_{i=1}^{D} \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i = \frac{1}{N} \text{trace}(\hat{\mathbf{X}}^T \hat{\mathbf{X}}) = \frac{1}{N} \text{trace}(\hat{\mathbf{X}} \hat{\mathbf{X}}^T) = \text{trace}(\mathbf{S}) = \sum_{i=1}^{D} \lambda_i$$

tức là trong mọi hệ trục toạ độ, tổng phương sai của dữ liệu là như nhau và bằng tổng các trị riêng của ma trận hiệp phương sai $\sum_{i=1}^{d} \lambda_i$.

Optimal Dim k

- PCA với số chiều k giữ lại lượng thông tin (tổng các phương sai) là $\sum_{i=1}^{k} \lambda_i$.
- Tỷ lệ thông tin được giữ lại trong PCA có thể tính theo

$$r_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j}.$$

- Ví dụ: để giữ lại 99% thông tin, cần chọn số chiều k là số tự nhiên nhỏ nhất sao cho $r_k \geq 0.99$.
- Chú ý khi dữ liệu phân bố quanh một không gian con, các giá trị phương sai lớn nhất ứng với các λ_i đầu tiên lớn hơn nhiều so với các phương sai còn lại. Do đó ta có thể chọn được k khá nhỏ để đạt được $r_k \geq 0.99$.

Applying of PCA in practice

Trong phần này ta giả thiết dữ liệu đã được chuẩn hóa. Lúc đó

$$\bar{\mathbf{x}} = 0 \quad \text{và} \quad \mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^T.$$

- Có hai trường hợp trong thực tế mà chúng ta cần lưu ý về PCA
  - Lượng dữ liệu quan sát nhỏ hơn rất nhiều so với số chiều dữ liệu: $N < d$;
  - Lượng dữ liệu trong tập quan sát là rất lớn (massive dataset).
- Hậu quả: Tính toán trực tiếp ma trận hiệp phương sai và các giá trị riêng có thể trở nên bất khả thi.

Dim. $d > N$ - cardinal of data

- $\mathbf{X} \in \mathbb{R}^{d \times N}$ với $d > N$. Trước hết cần chọn $K < \text{rank}(\mathbf{X}) \leq N$.
- Để tính các giá trị riêng và vector riêng của $\mathbf{S}$ cần đến một số tính chất
  - **Tính chất 1**: Giá trị riêng của A cũng là giá trị riêng của kA với $k \neq 0$ bất kỳ.
  - **Tính chất 2**: Với $d_1, d_2 > 0$ là các số tự nhiên bất kỳ; $\mathbf{A} \in \mathbb{R}^{d_1 \times d_2}$ và $\mathbf{B} \in \mathbb{R}^{d_2 \times d_1}$, lúc đó tập các giá trị riêng của ma trận $\mathbf{AB}$ cũng là tập giá trị riêng của $\mathbf{BA}$[1].
  - **Tính chất 3**: Giả sử $(\lambda, \mathbf{u})$ là một cặp trị riêng - vector riêng của $\mathbf{T} = \mathbf{X}^T\mathbf{X}$, lúc đó $(\lambda, \mathbf{Xu})$ là cặp trị riêng - vector riêng của $\mathbf{S} = \mathbf{X}\mathbf{X}^T$. Thật vậy

$$\mathbf{X}^T\mathbf{X}\mathbf{u} = \lambda\mathbf{u} \Rightarrow (\mathbf{X}\mathbf{X}^T)(\mathbf{X}\mathbf{u}) = \lambda\mathbf{X}\mathbf{u}.$$

- Vậy thay vì tìm trị riêng của ma trận hiệp phương sai $\mathbf{S} \in \mathbb{R}^{d \times d}$, ta có thể tìm trị riêng của $\mathbf{T} = \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{N \times N}$ có số chiều $N < d$.

[1] Theorem 1.3.22, "Matrix Analysis", Horn and Johnson - the second edition.

Chuẩn hóa hệ vector riêng

Không gian con riêng: ứng với trị riêng của một ma trận là không gian sinh (*span subspace*) tạo bởi toàn bộ các vector riêng ứng với trị riêng đó.

- Ta cần chuẩn hoá các vector riêng $\{u_i\}$ tìm được sao cho chúng tạo thành một hệ trực chuẩn.
- Việc này có thể được thực hiện dựa trên một số tính chất của không gian con riêng ứng với các giá trị riêng.

Chuẩn hóa hệ vector riêng

- Tính chất của hệ riêng $\{(\lambda_i, u_i)\}$
  - **Chú ý 1**: Nếu $\mathbf{A}$ là một ma trận đối xứng, $(\lambda_1, \mathbf{x}_1), (\lambda_2, \mathbf{x}_2)$ là các căp trị riêng - vector riêng của $\mathbf{A}$ với $\lambda_1 \neq \lambda_2$, thì $\mathbf{x}_1^T \mathbf{x}_2 = 0$. Thật vậy

$$\mathbf{x}_2^T \mathbf{A} \mathbf{x}_1 = \mathbf{x}_1^T \mathbf{A} \mathbf{x}_2 = \lambda_1 \mathbf{x}_2^T \mathbf{x}_1 = \lambda_2 \mathbf{x}_1^T \mathbf{x}_2 \Rightarrow \mathbf{x}_1^T \mathbf{x}_2 = 0 \quad \text{do} \quad \lambda_1 \neq \lambda_2.$$

  - Tính chất trên suy ra, hai vector bất kỳ trong hai không gian riêng ứng với hai trị riêng khác nhau của một ma trận đối xứng thì trực giao với nhau.
  - **Chú ý 2**: với các trị riêng độc lập tìm được trong một không gian riêng, ta có thể dùng thuật toán Gram-Schmit để chuẩn hoá chúng về một hệ trực chuẩn.
- Kết hợp hai điểm trên, ta có thể thu được các vector riêng tạo thành một hệ trực chuẩn, chính là ma trận $\mathbf{U}_k$ trong phương pháp PCA.

Trường hợp large-scale

- Thực tế có những bài toán mà d, N rất lớn. Lúc đó cần giải tìm hệ riêng của ma trận với kích thước lớn.
- **Ví dụ:** Dữ liệu là có 1 triệu bức ảnh 1000×1000 pixels (số chiều là 10^6 - rất lớn).
- Trực tiếp tính toán giá trị riêng và vector riêng cho ma trận hiệp phương sai là không khả thi.
- Có thể sử dụng phương pháp lặp **Power Method**[2] để xấp xỉ giá trị riêng λ lớn nhất và vector riêng (chuẩn hóa) ứng với nó.

Thuật toán Power Method có các bước như trang sau.

[2] https://www.cs.huji.ac.il/ csip/tirgul2.pdf

Trường hợp large-scale: Power Method

$\mathbf{A} \in \mathbb{R}^{n \times n}$ - Ma trận nửa xác định dương.

1. Chọn vector $\mathbf{q}^{(0)} \in \mathbb{R}^n$ với $\|\mathbf{q}^{(0)}\|_2 = 1$ bất kỳ.
2. Tại các bước $i = 1, 2, \ldots$, tính $\mathbf{z} = \mathbf{A}\mathbf{q}^{(i-1)}$.
3. Chuẩn hóa $\mathbf{q}^{(k)} = \mathbf{z}/\|\mathbf{z}\|_2$.
4. Nếu $\|\mathbf{q}^{(k)} - \mathbf{q}^{(k-1)}\|_2$ đủ nhỏ thì sang bước 5. Ngược lại, đặt $k := k + 1$ và quay lại bước 2.)
5. Tính: $\lambda_1 = (\mathbf{q}^{(k)})^T \mathbf{A}\mathbf{q}^{(k)}$, đây là giá trị riêng lớn nhất của $\mathbf{A}$; $\mathbf{q}^{(k)}$ là vector riêng ứng với λ_1. Kết thúc.

Phương pháp Power hội tụ nhanh![2]
Sau khi tìm được trị riêng lớn nhất λ_1, có thể tìm các giá trị riêng tiếp theo dựa vào tính chất

[2] https://www.cs.huji.ac.il/ csip/tirgul2.pdf

Trường hợp large-scale

Định lý

Nếu ma trận nửa xác định dương $\mathbf{A}$ có các trị riêng $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n (\geq 0)$ và các vector riêng tương ứng $\mathbf{v}_1, \ldots, \mathbf{v}_n$ tạo thành một hệ trực chuẩn, thì ma trận

$$\mathbf{B} = \mathbf{A} - \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T$$

có các trị riêng $\lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n \geq 0$ tương ứng với các vector riêng $\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_n$.

- Thật vậy, với $i = 1$: $\mathbf{B}\mathbf{v}_1 = (\mathbf{A} - \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T)\mathbf{v} = \mathbf{A}\mathbf{v}_1 - \lambda_1 \mathbf{v}_1 = \mathbf{0}$.
- Với $i > 1$: $\mathbf{B}\mathbf{v}_i = (\mathbf{A} - \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T)\mathbf{v}_i = \mathbf{A}\mathbf{v}_i - \lambda_1 \mathbf{v}_1 (\mathbf{v}_1^T \mathbf{v}_i) = \mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$.

Trường hợp large-scale

- Đến đây ta có thủ tục lặp để tìm k cặp trị riêng - vector riêng tương ứng $(\lambda_i, u_i)_{i=1}^k$ với $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_k \geq \lambda_j \geq 0, \forall j > k$, như sau
  0. Bước đầu, $CountPair = 1$;
  1. Áp dụng Power Method để tìm cặp $(\bar{\lambda}_1, \bar{u}_1)$ với $\bar{\lambda}_1$ lớn nhất.
  2. Gán $\lambda_{CountPair} := \bar{\lambda}_1$; $u_{CountPair} := \bar{u}_1$. Nếu $CountPair = k$, sang bước 4. Ngược lại, sang bước 3.
  3. Áp dụng kết quả định lý trước, đặt: $\mathbf{A} = \mathbf{A} - \bar{\lambda}_1 \bar{u}_1 \bar{u}_1^T$.
     Đặt $CountPair = CountPair + 1$, quay lại bước 1.
  4. Hệ $\{(\lambda_i, u_i)\}_{i=1}^k$ thu được chính là hệ cần tìm.
- Với quy trình trên, ta sẽ tìm được (xấp xỉ) k trị riêng và vector riêng tương ứng của ma trận hiệp phương sai $\mathbf{S}$, với $k \leq \text{rank}(\mathbf{S})$ tùy ý.

Remarks

- PCA là một phương pháp Unsupervised.
- Việc thực hiện PCA trên toàn bộ dữ liệu không phụ thuộc vào class(nếu có) của mỗi dữ liệu.
- PCA đôi khi không hiệu quả thậm chí gây sai lệch khi áp dụng cho các bài toán classification.
- Với các bài toán Large-scale, đôi khi việc tính toán trên toàn bộ dữ liệu là không khả thi vì còn có vấn đề về bộ nhớ.
- Giải pháp là thực hiện PCA lần đầu trên một tập con dữ liệu vừa với bộ nhớ, sau đó lấy một tập con khác để (incrementally) cập nhật nghiệm của PCA tới khi nào hội tụ.
- Có nhiều hướng mở rộng của PCA, có thể tìm kiếm theo từ khoá: Sparse PCA, Kernel PCA, Robust PCA.

Eigenface

- Áp dụng phương pháp PCA để rút gọn số chiều của dữ liệu ảnh.
- Dữ liệu là ảnh khuôn mặt của 15 người, định dạng tệp pgm, được tải về từ cơ sở dữ liệu *Yale face database*.
- Khuôn mặt mỗi người được thể hiện trong 11 trạng thái: "centerlight", "glasses", "happy", "leftlight", "noglasses", "normal", "rightlight", "sad", "sleepy", "surprised", "wink" (tổng cộng có 165 ảnh).
- Tất cả các ảnh đều có kích thước 116×98 pixels, tức là dữ liệu có $116 \times 98 = 11368$ chiều.
- Ta sử dụng PCA để giảm số chiều về $k = 100$.

Eigenface

Có thể sử dụng thư viện **sklearn** của python để xây dựng đoạn lệnh cho phương pháp PCA khá đơn giản:

```
# Doing PCA:
from sklearn.decomposition import PCA
pca = PCA(n_components=K) # K = 100
pca.fit(X.T)

# projection matrix:
U = pca.components_.T
```

Eigenface

Bảng: Hàng trên: Ảnh gốc; hàng dưới: Ảnh đã giảm số chiều từ $116 \times 98 = 11368$ xuống còn $k = 100$.