

## Mục lục

<b>1</b>	<b>Lời nói đầu</b>	<b>2</b>
1.1	Đặt vấn đề	2
1.2	Lời cảm ơn	3
1.3	Bảng các ký hiệu	3
<b>2</b>	<b>Phân tích thành phần chính</b>	<b>5</b>
2.1	Phân tích thành phần chính	5
2.2	Các bước thực hiện phân tích thành phần chính	10
2.3	Liên hệ với phân tích giá trị suy biến	11
2.4	Làm thế nào để chọn số chiều của dữ liệu mới	13
2.5	Lưu ý về tính toán phân tích thành phần chính	13
2.6	Ứng dụng trong nén dữ liệu đa phương tiện dạng ảnh	14
2.7	Một số ứng dụng khác	18
2.8	Kết luận	22
	<b>Tài liệu tham khảo</b>	<b>23</b>
	<b>Index</b>	<b>24</b>

# Lời nói đầu

---

## 1.1. Đặt vấn đề

Các bài toán quy mô lớn trên thực tế có lượng điểm dữ liệu lớn và dữ liệu nhiều chiều. Nếu thực hiện lưu trữ và tính toán trực tiếp trên dữ liệu có số chiều lớn thì sẽ gặp khó khăn về lưu trữ và tính toán. Vì vậy, *giảm chiều dữ liệu* (dimensionality reduction hoặc dimension reduction) là một bước quan trọng trong nhiều bài toán học máy.

Dưới góc độ toán học, giảm chiều dữ liệu là việc đi tìm một hàm số  $f: \mathbb{R}^D \rightarrow \mathbb{R}^K$  với  $K < D$  biến một điểm dữ liệu  $\mathbf{x}$  trong không gian có số chiều lớn  $\mathbb{R}^D$  thành một điểm  $\mathbf{z}$  trong không gian có số chiều nhỏ hơn  $\mathbb{R}^D$ . Giảm chiều dữ liệu có thể áp dụng vào các bài toán nén thông tin. Nó cũng hữu ích trong việc chọn ra những đặc trưng quan trọng hoặc tạo ra các đặc trưng mới từ đặc trưng cũ phù hợp với từng bài toán. Trong nhiều trường hợp, làm việc trên dữ liệu đã giảm chiều cho kết quả tốt hơn dữ liệu trong không gian ban đầu.

Phương pháp phân tích thành phần chính được ra đời và là 1 trong những phương pháp được dùng để giảm chiều dữ liệu phổ biến nhất mà vẫn giữ tối đa được thông tin.

## 1.2. Lời cảm ơn

Lời đầu tiên, em xin bày tỏ sự cảm ơn chân thành đối với Cô giáo, TS. Nguyễn Thị Mỹ Bình – giáo viên hướng dẫn trực tiếp em.

Em cũng xin gửi lời cảm ơn tới các thầy cô trong khoa Công nghệ thông tin, trường Đại học Công Nghiệp Hà Nội đã hướng dẫn, chỉ bảo và tạo điều kiện cho em học tập cũng như nghiên cứu trong thời gian qua.







Cảm ơn Câu lạc bộ HIT, Đội Olympic Tin học khoa Công nghệ thông tin đã đồng hành cùng em trong suốt quãng thời gian học tập, làm việc tại trường.

Mặc dù đã cố gắng hoàn thành Báo cáo thực tập tốt nghiệp nhưng chắc chắn sẽ không tránh khỏi những sai sót, em kính mong nhận được sự thông cảm và chỉ bảo của các thầy cô và các bạn.

## 1.3. Bảng các ký hiệu

Các ký hiệu sử dụng trong sách được liệt kê trong Bảng 1.1 ở trang tiếp theo.

Bảng 1.1: Các quy ước ký hiệu và tên gọi được sử dụng trong sách

Ký hiệu	Ý nghĩa
$x, y, N, k$	in nghiêng, thường hoặc hoa, là các số vô hướng
$\mathbf{x}, \mathbf{y}$	in đậm, chữ thường, là các vector
$\mathbf{X}, \mathbf{Y}$	in đậm, chữ hoa, là các ma trận
$\mathbb{R}$	tập hợp các số thực
$\mathbb{N}$	tập hợp các số tự nhiên
$\mathbb{C}$	tập hợp các số phức
$\mathbb{R}^m$	tập hợp các vector thực có $m$ phần tử
$\mathbb{R}^{m \times n}$	tập hợp các ma trận thực có $m$ hàng, $n$ cột
$\mathbb{S}^n$	tập hợp các ma trận vuông đối xứng bậc $n$
$\mathbb{S}_+^n$	tập hợp các ma trận nửa xác định dương bậc $n$
$\mathbb{S}_{++}^n$	tập hợp các ma trận xác định dương bậc $n$
$\in$	phần tử thuộc tập hợp
$\exists$	tồn tại
$\forall$	mọi
$\triangleq$	ký hiệu là/bởi. Ví dụ $a \triangleq f(x)$ nghĩa là “ký hiệu $f(x)$ bởi $a$ ”.
$x_i$	phần tử thứ $i$ (tính từ 1) của vector $\mathbf{x}$
$\text{sgn}(x)$	hàm xác định dấu. Bằng 1 nếu $x \geq 0$ , bằng -1 nếu $x < 0$ .
$\exp(x)$	$e^x$
$\log(x)$	logarit <i>tự nhiên</i> của số thực dương $x$
$\underset{x}{\operatorname{argmin}} f(x)$	giá trị của $x$ để hàm $f(x)$ đạt giá trị nhỏ nhất
$\underset{x}{\operatorname{argmax}} f(x)$	giá trị của $x$ để hàm $f(x)$ đạt giá trị lớn nhất
$a_{ij}$	phần tử hàng thứ $i$ , cột thứ $j$ của ma trận $\mathbf{A}$
$\mathbf{A}^T$	chuyển vị của ma trận $\mathbf{A}$
$\mathbf{A}^H$	chuyển vị liên hợp (Hermitian) của ma trận phức $\mathbf{A}$
$\mathbf{A}^{-1}$	nghịch đảo của ma trận vuông $\mathbf{A}$ , nếu tồn tại
$\mathbf{A}^\dagger$	giả nghịch đảo của ma trận không nhất thiết vuông $\mathbf{A}$
$\mathbf{A}^{-T}$	chuyển vị của nghịch đảo của ma trận $\mathbf{A}$ , nếu tồn tại
$\ \mathbf{x}\ _p$	$\ell_p$ norm của vector $\mathbf{x}$
$\ \mathbf{A}\ _F$	Frobenius norm của ma trận $\mathbf{A}$
$\operatorname{diag}(\mathbf{A})$	đường chéo chính của ma trận $\mathbf{A}$
$\operatorname{trace}(\mathbf{A})$	trace của ma trận $\mathbf{A}$
$\det(\mathbf{A})$	định thức của ma trận vuông $\mathbf{A}$
$\operatorname{rank}(\mathbf{A})$	hạng của ma trận $\mathbf{A}$
o.w	<i>otherwise</i> – trong các trường hợp còn lại
$\frac{\partial f}{\partial x}$	đạo hàm của hàm số $f$ theo $x \in \mathbb{R}$
$\nabla_{\mathbf{x}} f$	gradient của hàm số $f$ theo $\mathbf{x}$ ( $\mathbf{x}$ là vector hoặc ma trận)
$\nabla_{\mathbf{x}}^2 f$	gradient bậc hai của hàm số $f$ theo $\mathbf{x}$ , còn được gọi là <i>Hesse</i>
$\odot$	Hadamard product (elementwise product). Phép nhân từng phần tử của hai vector hoặc ma trận cùng kích thước.
$\propto$	tỉ lệ với
	đường nét liền
	đường nét đứt
	đường nét chấm (đường chấm chấm)
	đường chấm gạch
	nền chấm
	nền sọc chéo

---

**Chương 2**

---

---

# Phân tích thành phần chính

---

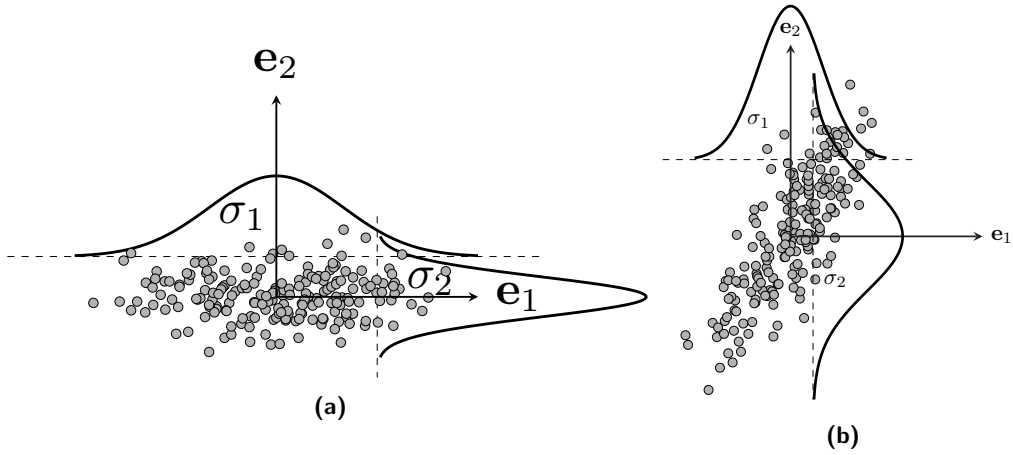
## 2.1. Phân tích thành phần chính

### 2.1.1. Ý tưởng

Giả sử vector dữ liệu ban đầu  $\mathbf{x} \in \mathbb{R}^D$  được giảm chiều trở thành  $\mathbf{z} \in \mathbb{R}^K$  với  $K < D$ . Một cách đơn giản để giảm chiều dữ liệu từ  $D$  về  $K < D$  là chỉ giữ lại  $K$  phần tử quan trọng nhất. Có hai câu hỏi lập tức được đặt ra. Thứ nhất, làm thế nào để xác định tầm quan trọng của mỗi phần tử? Thứ hai, nếu tầm quan trọng của các phần tử là như nhau, ta cần bỏ đi những phần tử nào?

Để trả lời câu hỏi thứ nhất, hãy quan sát Hình 2.1a. Giả sử các điểm dữ liệu có thành phần thứ hai (phương đứng) giống hệt nhau hoặc sai khác nhau không đáng kể (phương sai nhỏ). Khi đó, thành phần này hoàn toàn có thể được lược bỏ, và ta ngầm hiểu rằng nó sẽ được xấp xỉ bằng kỳ vọng của thành phần đó trên toàn bộ dữ liệu. Ngược lại, nếu áp dụng phương pháp này lên chiều thứ nhất (phương ngang), lượng thông tin bị mất đi đáng kể do sai số xấp xỉ quá lớn. Vì vậy, lượng thông tin theo mỗi thành phần có thể được đo bằng phương sai của dữ liệu trên thành phần đó. Tổng lượng thông tin là tổng phương sai trên toàn bộ các thành phần. Lấy một ví dụ về việc có hai camera được đặt dùng để chụp cùng một người, một camera phía trước và một camera đặt trên đầu. Rõ ràng, hình ảnh thu được từ camera đặt phía trước mang nhiều thông tin hơn so với hình ảnh nhìn từ phía trên đầu. Vì vậy, bức ảnh chụp từ phía trên đầu có thể được bỏ qua mà không làm mất đi quá nhiều thông tin về hình dáng của người đó.

Câu hỏi thứ hai tương ứng với trường hợp Hình 2.1b. Trong cả hai chiều, phương sai của dữ liệu đều lớn; việc bỏ đi một trong hai chiều đều khiến một lượng thông tin đáng kể bị mất đi. Tuy nhiên, nếu xoay trục tọa độ đi một góc



**Hình 2.1.** Ví dụ về phương sai của dữ liệu trong không gian hai chiều. (a) Phương sai của chiều thứ hai (tỉ lệ với độ rộng của đường hình chuông) nhỏ hơn phương sai của chiều thứ nhất. (b) Cả hai chiều có phương sai đáng kể. Phương sai của mỗi chiều là phương sai của thành phần tương ứng được lấy trên toàn bộ dữ liệu. Phương sai tỉ lệ thuận với độ phân tán của dữ liệu.

$$\begin{array}{c}
 \begin{array}{|c|c|} \hline N \\ \hline D & \mathbf{X} \\ \hline \end{array} & = & \begin{array}{|c|c|} \hline K & D-K \\ \hline D & \mathbf{U}_K & \hat{\mathbf{U}}_K \\ \hline \end{array} \times \begin{array}{|c|} \hline N \\ \hline K & \mathbf{Z} \\ \hline D-K & \mathbf{Y} \\ \hline \end{array} \\
 \text{Dữ liệu ban đầu} & & \text{Ma trận trực giao} \quad \text{Toạ độ trong hệ cơ sở mới} \\
 \\
 & = & \begin{array}{|c|} \hline K \\ \hline D & \mathbf{U}_K \\ \hline \end{array} \times \begin{array}{|c|c|} \hline N & \\ \hline K & \mathbf{Z} & D \\ \hline \end{array} + \begin{array}{|c|} \hline \hat{\mathbf{U}}_K \\ \hline \end{array} \times \begin{array}{|c|} \hline \mathbf{Y} \\ \hline \end{array}
 \end{array}$$

**Hình 2.2.** Ý tưởng chính của PCA: Tìm một hệ trục chuẩn mới sao cho trong hệ này, các thành phần quan trọng nhất nằm trong  $K$  thành phần đầu tiên.

phù hợp, một trong hai chiều dữ liệu có thể được lược bỏ vì dữ liệu có xu hướng phân bố xung quanh một đường thẳng.

*Phân tích thành phần chính* (principle component analysis, PCA) là phương pháp đi tìm một phép xoay trục toạ độ để được một hệ trục toạ độ mới sao cho trong hệ mới này, thông tin của dữ liệu chủ yếu tập trung ở một vài thành phần. Phần còn lại chứa ít thông tin hơn có thể được lược bỏ.

Phép xoay trục toạ độ có liên hệ chặt chẽ tới hệ trục chuẩn và ma trận trực giao (xem Mục ?? và ??). Giả sử hệ cơ sở trục chuẩn mới là  $\mathbf{U}$  (mỗi cột của  $\mathbf{U}$  là một vector đơn vị cho một chiều) và ta muốn giữ lại  $K$  toạ độ trong hệ cơ sở mới này. Không mất tính tổng quát, giả sử đó là  $K$  thành phần đầu tiên. Quan sát Hình 2.2 với cơ sở mới  $\mathbf{U} = [\mathbf{U}_K, \hat{\mathbf{U}}_K]$  là một hệ trục chuẩn với  $\mathbf{U}_K$  là ma

trận con tạo bởi  $K$  cột đầu tiên của  $\mathbf{U}$ . Trong hệ cơ sở mới này, ma trận dữ liệu có thể được viết thành

$$\mathbf{X} = \mathbf{U}_K \mathbf{Z} + \hat{\mathbf{U}}_K \mathbf{Y} \quad (2.1)$$

Từ đây ta cũng suy ra

$$\begin{bmatrix} \mathbf{Z} \\ \mathbf{Y} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_K^T \\ \hat{\mathbf{U}}_K^T \end{bmatrix} \mathbf{X} \Rightarrow \begin{aligned} \mathbf{Z} &= \mathbf{U}_K^T \mathbf{X} \\ \mathbf{Y} &= \hat{\mathbf{U}}_K^T \mathbf{X} \end{aligned} \quad (2.2)$$

Mục đích của PCA là đi tìm ma trận trực giao  $\mathbf{U}$  sao cho phần lớn thông tin nằm ở  $\mathbf{U}_K \mathbf{Z}$ , phần nhỏ thông tin nằm ở  $\hat{\mathbf{U}}_K \mathbf{Y}$ . Phần nhỏ này sẽ được lược bỏ và xấp xỉ bằng một ma trận có các cột như nhau. Gọi mỗi cột đó là  $\mathbf{b}$ , 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 một vector hàng có toàn bộ các phần tử bằng một. Giả sử đã tìm được  $\mathbf{U}$ , ta cần tìm  $\mathbf{b}$  thoả mãn:

$$\mathbf{b} = \operatorname{argmin}_{\mathbf{b}} \|\mathbf{Y} - \mathbf{b} \mathbf{1}^T\|_F^2 = \operatorname{argmin}_{\mathbf{b}} \|\hat{\mathbf{U}}_K^T \mathbf{X} - \mathbf{b} \mathbf{1}^T\|_F^2 \quad (2.3)$$

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

$$(\mathbf{b} \mathbf{1}^T - \hat{\mathbf{U}}_K^T \mathbf{X}) \mathbf{1} = \mathbf{0} \Rightarrow N \mathbf{b} = \hat{\mathbf{U}}_K^T \mathbf{X} \mathbf{1} \Rightarrow \mathbf{b} = \hat{\mathbf{U}}_K^T \bar{\mathbf{x}}. \quad (2.4)$$

Ở đây ta đã sử dụng  $\mathbf{1}^T \mathbf{1} = N$  và  $\bar{\mathbf{x}} = \frac{1}{N} \mathbf{X} \mathbf{1}$  là vector trung bình các cột của  $\mathbf{X}$ . Với giá trị  $\mathbf{b}$  tìm được này, dữ liệu ban đầu sẽ được xấp xỉ bởi

$$\mathbf{X} = \mathbf{U}_K \mathbf{Z} + \hat{\mathbf{U}}_K \mathbf{Y} \approx \mathbf{U}_K \mathbf{Z} + \hat{\mathbf{U}}_K \mathbf{b} \mathbf{1}^T = \mathbf{U}_K \mathbf{Z} + \hat{\mathbf{U}}_K \hat{\mathbf{U}}_K^T \bar{\mathbf{x}} \mathbf{1}^T \triangleq \tilde{\mathbf{X}} \quad (2.5)$$

### 2.1.2. Hàm mất mát

Hàm mất mát của PCA được coi như sai số của phép xấp xỉ, được định nghĩa là

$$\begin{aligned} \frac{1}{N} \|\mathbf{X} - \tilde{\mathbf{X}}\|_F^2 &= \frac{1}{N} \|\hat{\mathbf{U}}_K \mathbf{Y} - \hat{\mathbf{U}}_K \hat{\mathbf{U}}_K^T \bar{\mathbf{x}} \mathbf{1}^T\|_F^2 = \frac{1}{N} \|\hat{\mathbf{U}}_K \hat{\mathbf{U}}_K^T \mathbf{X} - \hat{\mathbf{U}}_K \hat{\mathbf{U}}_K^T \bar{\mathbf{x}} \mathbf{1}^T\|_F^2 \\ &= \frac{1}{N} \|\hat{\mathbf{U}}_K \hat{\mathbf{U}}_K^T (\mathbf{X} - \bar{\mathbf{x}} \mathbf{1}^T)\|_F^2 \triangleq J(\mathbf{U}) \end{aligned} \quad (2.6)$$

Chú ý rằng, 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

$$\|\mathbf{V} \mathbf{W}\|_F^2 = \operatorname{trace}(\mathbf{W}^T \mathbf{V}^T \mathbf{V} \mathbf{W}) = \operatorname{trace}(\mathbf{W}^T \mathbf{W}) = \|\mathbf{W}\|_F^2 \quad (2.7)$$

Đặt  $\hat{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{x}} \mathbf{1}^T$ . Ma trận này có được bằng cách trừ mỗi cột của  $\mathbf{X}$  đi trung bình các cột của nó. Ta gọi  $\hat{\mathbf{X}}$  là ma trận dữ liệu đã được chuẩn hoá. Có thể thấy  $\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}, \forall n = 1, 2, \dots, N$ .

Vì vậy hàm mất mát trong (2.6) có thể được viết lại thành:

$$J(\mathbf{U}) = \frac{1}{N} \|\widehat{\mathbf{U}}_K^T \widehat{\mathbf{X}}\|_F^2 = \frac{1}{N} \|\widehat{\mathbf{X}}^T \widehat{\mathbf{U}}_K\|_F^2 = \frac{1}{N} \sum_{i=K+1}^D \|\widehat{\mathbf{X}}^T \mathbf{u}_i\|_2^2 \quad (2.8)$$

$$= \frac{1}{N} \sum_{i=K+1}^D \mathbf{u}_i^T \widehat{\mathbf{X}} \widehat{\mathbf{X}}^T \mathbf{u}_i = \sum_{i=K+1}^D \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i \quad (2.9)$$

với  $\mathbf{S} = \frac{1}{N} \widehat{\mathbf{X}} \widehat{\mathbf{X}}^T$  là ma trận hiệp phương sai của dữ liệu và luôn là một ma trận nửa xác định dương (xem Mục ??).

Công việc còn lại là tìm các  $\mathbf{u}_i$  để mất mát là nhỏ nhất.

Với ma trận  $\mathbf{U}$  trực giao bất kỳ, thay  $K = 0$  vào (2.9) ta có

$$L = \sum_{i=1}^D \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i = \frac{1}{N} \|\widehat{\mathbf{X}}^T \mathbf{U}\|_F^2 = \frac{1}{N} \text{trace}(\widehat{\mathbf{X}}^T \mathbf{U} \mathbf{U}^T \widehat{\mathbf{X}}) \quad (2.10)$$

$$= \frac{1}{N} \text{trace}(\widehat{\mathbf{X}}^T \widehat{\mathbf{X}}) = \frac{1}{N} \text{trace}(\widehat{\mathbf{X}} \widehat{\mathbf{X}}^T) = \text{trace}(\mathbf{S}) = \sum_{i=1}^D \lambda_i \quad (2.11)$$

Với  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_D \geq 0$  là các trị riêng của ma trận nửa xác định dương  $\mathbf{S}$ . Chú ý rằng các trị riêng này là thực và không âm<sup>1</sup>.

Như vậy  $L$  không phụ thuộc vào cách chọn ma trận trực giao  $\mathbf{U}$  và bằng tổng các phần tử trên đường chéo của  $\mathbf{S}$ . Nói cách khác,  $L$  chính là tổng các phương sai theo từng thành phần của dữ liệu ban đầu<sup>2</sup>.

Vì vậy, việc tối thiểu hàm mất mát  $J$  được cho bởi (2.9) tương đương với việc tối đa biểu thức

$$F = L - J = \sum_{i=1}^K \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i \quad (2.12)$$

### 2.1.3. Tối ưu hàm mất mát

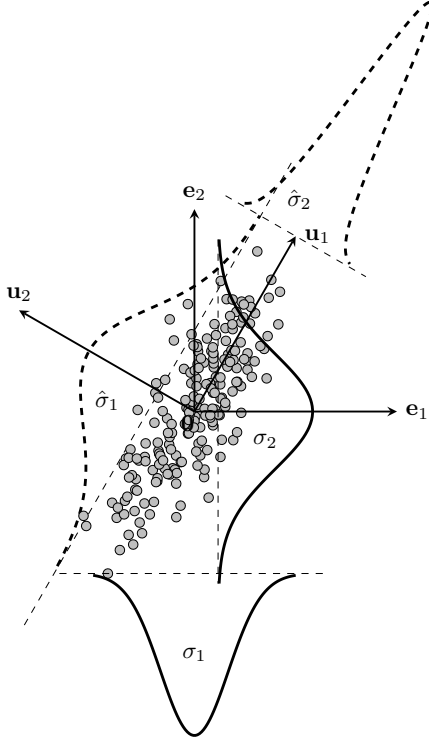
Nghiệm của bài toán tối ưu hàm mất mát PCA được tìm dựa trên khẳng định sau đây:

---

<sup>1</sup> Tổng các trị riêng của một ma trận vuông bất kỳ luôn bằng vết của ma trận đó.

<sup>2</sup> Mỗi thành phần trên đường chéo chính của ma trận hiệp phương sai chính là phương sai của thành phần dữ liệu tương ứng.





**Hình 2.3.** PCA có thể được coi là phương pháp đi tìm một hệ cơ sở trực chuẩn đóng vai trò một phép xoay, sao cho trong hệ cơ sở mới này, phương sai theo một số chiều nào đó là không đáng kể và có thể lược bỏ. Trong hệ cơ sở ban đầu  $\mathbf{O}\mathbf{e}_1\mathbf{e}_2$ , phương sai theo mỗi chiều (độ rộng của các đường hình chuông nét liền) đều lớn. Trong không gian mới với hệ cơ sở  $\mathbf{O}\mathbf{u}_1\mathbf{u}_2$ , phương sai theo hai chiều (độ rộng của các đường hình chuông nét đứt) chênh lệch nhau đáng kể. Chiều dữ liệu có phương sai nhỏ có thể được lược bỏ vì dữ liệu theo chiều này ít phân tán.

Nếu  $\mathbf{S}$  là một ma trận nửa xác định dương, bài toán tối ưu

$$\max_{\mathbf{U}_K} \sum_{i=1}^K \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i \quad (2.13)$$

$$\text{thỏa mãn: } \mathbf{U}_K^T \mathbf{U}_K = \mathbf{I} \quad (2.14)$$

có nghiệm  $\mathbf{u}_1, \dots, \mathbf{u}_K$  là các vector riêng ứng với  $K$  trị riêng (kể cả lặp) lớn nhất của  $\mathbf{S}$ . Khi đó, giá trị lớn nhất của hàm mục tiêu là  $\sum_{i=1}^K \lambda_i$ , với  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_D$  là các trị riêng của  $\mathbf{S}$ .

Khẳng định này có thể được chứng minh bằng quy nạp

Trị riêng lớn nhất  $\lambda_1$  của ma trận hiệp phương sai  $\mathbf{S}$  còn được gọi là *thành phần chính thứ nhất* (the first principal component), trị riêng thứ hai  $\lambda_2$  được gọi là *thành phần chính thứ hai*,... Tên gọi *phân tích thành phần chính* (principal component analysis) bắt nguồn từ đây. Ta chỉ giữ lại  $K$  thành phần chính đầu tiên khi giảm chiều dữ liệu dùng PCA.

Hình 2.3 minh họa các thành phần chính với dữ liệu hai chiều. Trong không gian ban đầu với các vector cơ sở  $\mathbf{e}_1, \mathbf{e}_2$ , phương sai theo mỗi chiều dữ liệu (tỉ lệ với độ rộng của các hình chuông nét liền) đều lớn. Trong hệ cơ sở mới  $\mathbf{O}\mathbf{u}_1\mathbf{u}_2$ , phương sai theo chiều thứ hai  $\hat{\sigma}_2^2$  nhỏ so với  $\hat{\sigma}_1^2$ . Điều này chỉ ra rằng khi chiếu

dữ liệu lên  $\mathbf{u}_2$ , ta được các điểm rất gần nhau và gần với giá trị trung bình theo chiều đó. Trong trường hợp này, vì giá trị trung bình theo mọi chiều bằng 0, ta có thể thay thế toạ độ theo chiều  $\mathbf{u}_2$  bằng 0. Rõ ràng là nếu dữ liệu có phương sai càng nhỏ theo một chiều nào đó thì khi xấp xỉ chiều đó bằng một hằng số, sai số xấp xỉ càng nhỏ. PCA thực chất là đi tìm một phép xoay tương ứng với một ma trận trực giao sao cho trong hệ toạ độ mới, tồn tại các chiều có phương sai nhỏ có thể được bỏ qua; ta chỉ cần giữ lại các chiều/thành phần khác quan trọng hơn. Như đã khẳng định ở trên, tổng phương sai theo toàn bộ các chiều trong một hệ cơ sở bất kỳ là như nhau và bằng tổng các trị riêng của ma trận hiệp phương sai. Vì vậy, PCA còn được coi là phương pháp giảm số chiều dữ liệu sao tổng phương sai còn lại là lớn nhất.

## 2.2. Các bước thực hiện phân tích thành phần chính

Từ các suy luận trên, ta có thể tóm tắt lại các bước trong PCA như sau:

- 1) Tính vector trung bình của toàn bộ dữ liệu:  $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$ .
- 2) Trừ mỗi điểm dữ liệu đi vector trung bình của toàn bộ dữ liệu để được dữ liệu chuẩn hoá:

$$\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}} \quad (2.15)$$

- 3) Đặt  $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_D]$  là ma trận dữ liệu chuẩn hoá, tính ma trận hiệp phương sai:

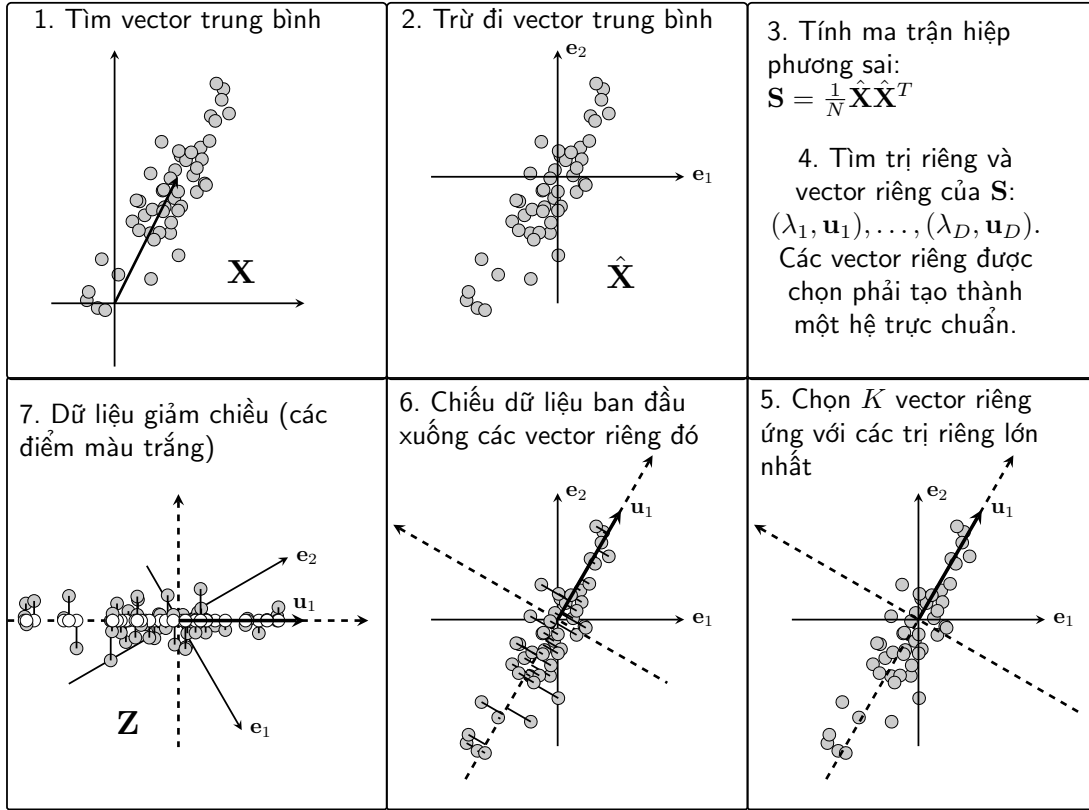
$$\mathbf{S} = \frac{1}{N} \hat{\mathbf{X}} \hat{\mathbf{X}}^T \quad (2.16)$$

- 4) Tính các trị riêng và vector riêng tương ứng có  $\ell_2$  norm bằng 1 của ma trận này, sắp xếp chúng theo thứ tự giảm dần của trị riêng.
- 5) Chọn  $K$  vector riêng ứng với  $K$  trị riêng lớn nhất để xây dựng ma trận  $\mathbf{U}_K$  có các cột tạo thành một hệ trực giao.  $K$  vector này được gọi là các thành phần chính, tạo thành một không gian con gần với phân bố của dữ liệu ban đầu đã chuẩn hoá.
- 6) Chiếu dữ liệu ban đầu đã chuẩn hoá  $\hat{\mathbf{X}}$  xuống không gian con tìm được.
- 7) Dữ liệu mới là toạ độ của các điểm dữ liệu trên không gian mới:  $\mathbf{Z} = \mathbf{U}_K^T \hat{\mathbf{X}}$ .

Như vậy, PCA là kết hợp của phép tịnh tiến, xoay trục toạ độ và chiếu dữ liệu lên hệ toạ độ mới.

Dữ liệu ban đầu có thể tính được xấp xỉ theo dữ liệu mới bởi  $\mathbf{x} \approx \mathbf{U}_K \mathbf{Z} + \bar{\mathbf{x}}$ .

## Quy trình thực hiện PCA



Hình 2.4. Các bước thực hiện PCA.

Một điểm dữ liệu mới  $\mathbf{v} \in \mathbb{R}^D$  sẽ được giảm chiều bằng PCA theo công thức  $\mathbf{w} = \mathbf{U}_K^T(\mathbf{v} - \bar{\mathbf{x}}) \in \mathbb{R}^K$ . Ngược lại, nếu biết  $\mathbf{w}$ , ta có thể xấp xỉ  $\mathbf{v}$  bởi  $\mathbf{U}_K \mathbf{w} + \bar{\mathbf{x}}$ . Các bước thực hiện PCA được minh họa trong Hình 2.4.

### 2.3. Liên hệ với phân tích giá trị suy biến

PCA và SVD có mối quan hệ đặc biệt với nhau. Xin phép nhắc lại hai điểm đã trình bày dưới đây:

#### 2.3.1. SVD cho bài toán xấp xỉ hạng thấp tốt nhất

Nhiệm của bài toán xấp xỉ một ma trận bởi một ma trận có hạng không vượt quá  $k$ :

$$\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{A}\|_F \quad (2.17)$$

thoả mãn:  $\text{rank}(\mathbf{A}) = K$

chính là SVD cắt ngọn của  $\mathbf{A}$ .

Cụ thể, nếu SVD của  $\mathbf{X} \in \mathbb{R}^{D \times N}$  là

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad (2.18)$$

với  $\mathbf{U} \in \mathbb{R}^{D \times D}$  và  $\mathbf{V} \in \mathbb{R}^{N \times N}$  là các ma trận trực giao và  $\mathbf{\Sigma} \in \mathbb{R}^{D \times N}$  là ma trận đường chéo (không nhất thiết vuông) với các phần tử trên đường chéo không âm giảm dần. Nghiệm của bài toán (2.17) chính là:

$$\mathbf{A} = \mathbf{U}_K \mathbf{\Sigma}_K \mathbf{V}_K^T \quad (2.19)$$

với  $\mathbf{U} \in \mathbb{R}^{D \times K}$  và  $\mathbf{V} \in \mathbb{R}^{N \times K}$  là các ma trận tạo bởi  $K$  cột đầu tiên của  $\mathbf{U}$  và  $\mathbf{V}$ ,  $\mathbf{\Sigma}_K \in \mathbb{R}^{K \times K}$  là ma trận đường chéo con ứng với  $K$  hàng đầu tiên và  $K$  cột đầu tiên của  $\mathbf{\Sigma}$ .

### 2.3.2. Ý tưởng của PCA

Như đã chứng minh ở (2.5), PCA là bài toán đi tìm ma trận trực giao  $\mathbf{U}$  và ma trận mô tả dữ liệu ở không gian thấp chiều  $\mathbf{Z}$  sao cho việc xấp xỉ sau đây là tốt nhất:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_K \mathbf{Z} + \hat{\mathbf{U}}_K \hat{\mathbf{U}}_K^T \bar{\mathbf{x}} \mathbf{1}^T \quad (2.20)$$

với  $\mathbf{U}_K, \hat{\mathbf{U}}_K$  lần lượt là các ma trận được tạo bởi  $K$  cột đầu tiên và  $D - K$  cột cuối cùng của ma trận trực giao  $\mathbf{U}$ , và  $\bar{\mathbf{x}}$  là vector trung bình của dữ liệu.

Giả sử rằng vector trung bình  $\bar{\mathbf{x}} = \mathbf{0}$ . Khi đó, (2.20) tương đương với

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_K \mathbf{Z} \quad (2.21)$$

Bài toán tối ưu của PCA sẽ trở thành:

$$\begin{aligned} \mathbf{U}_K, \mathbf{Z} = \arg \min_{\mathbf{U}_K, \mathbf{Z}} \|\mathbf{X} - \mathbf{U}_K \mathbf{Z}\|_F \\ \text{thoả mãn:} \quad \mathbf{U}_K^T \mathbf{U}_K = \mathbf{I}_K \end{aligned} \quad (2.22)$$

với  $\mathbf{I}_K \in \mathbb{R}^{K \times K}$  là ma trận đơn vị trong không gian  $K$  chiều và điều kiện ràng buộc để đảm bảo các cột của  $\mathbf{U}_K$  tạo thành một hệ trực chuẩn.

### 2.3.3. Quan hệ giữa hai phương pháp

Có thể nhận ra nghiệm của bài toán (2.22) chính là

$$\begin{aligned} \mathbf{U}_K \text{ trong (2.22)} &= \mathbf{U}_K \text{ trong (2.19)} \\ \mathbf{Z} \text{ trong (2.22)} &= \mathbf{\Sigma}_K \mathbf{V}_K^T \text{ trong (2.19)} \end{aligned}$$

Như vậy, nếu các điểm dữ liệu được biểu diễn bởi các cột của một ma trận, và trung bình các cột của ma trận đó là vector không thì nghiệm của bài toán PCA được rút ra trực tiếp từ SVD cắt ngọn của ma trận đó. Nói cách khác, việc đi tìm nghiệm cho PCA chính là việc giải một bài toán phân tích ma trận thông qua SVD.

## 2.4. Làm thế nào để chọn số chiều của dữ liệu mới

Một câu hỏi được đặt ra là, làm thế nào để chọn giá trị  $K$  – chiều của dữ liệu mới – với từng dữ liệu cụ thể?

Thông thường,  $K$  được chọn dựa trên việc lượng thông tin muốn giữ lại. Ở đây, toàn bộ thông tin chính là tổng phương sai của toàn bộ các chiều dữ liệu. Lượng dữ liệu muốn giữ lại là tổng phương sai của dữ liệu trong hệ trục toạ độ mới.

Nhắc lại rằng 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  $\sum_{i=1}^D \lambda_i$ . Thêm nữa, PCA giúp giữ lại lượng thông tin (tổng các phương sai) là  $\sum_{i=1}^K \lambda_i$ . Vậy ta có thể coi biểu thức:

$$r_K = \frac{\sum_{i=1}^K \lambda_i}{\sum_{j=1}^D \lambda_j} \quad (2.23)$$

là tỉ lệ thông tin được giữ lại khi số chiều dữ liệu mới sau PCA là  $K$ . Như vậy, giả sử ta muốn giữ lại 99% dữ liệu, ta chỉ cần chọn  $K$  là số tự nhiên nhỏ nhất sao cho  $r_K \geq 0.99$ .

Khi dữ liệu phân bố quanh một không gian con, các giá trị phương sai lớn nhất ứng với các  $\lambda_i$  đầu tiên cao gấp nhiều lần các phương sai còn lại. Khi đó, ta có thể chọn được  $K$  khá nhỏ để đạt được  $r_K \geq 0.99$ .

## 2.5. Lưu ý về tính toán phân tích thành phần chính

Có hai trường hợp trong thực tế mà chúng ta cần lưu ý về PCA. Trường hợp thứ nhất là lượng dữ liệu có được nhỏ hơn rất nhiều so với số chiều dữ liệu. Trường hợp thứ hai là khi lượng dữ liệu trong tập huấn luyện rất lớn, việc tính toán ma trận hiệp phương sai và trị riêng đôi khi trở nên bất khả thi. Có những hướng giải quyết hiệu quả cho các trường hợp này.

Trong mục này, ta sẽ coi như dữ liệu đã được chuẩn hoá, tức đã được trừ đi vector kỳ vọng. Khi đó, ma trận hiệp phương sai sẽ là  $\mathbf{S} = \frac{1}{N} \mathbf{X} \mathbf{X}^T$ .

### 2.5.1. Số chiều dữ liệu nhiều hơn số điểm dữ liệu

Đó là trường hợp  $D > N$ , tức ma trận dữ liệu  $\mathbf{X}$  là một *ma trận cao*. Khi đó, số trị riêng khác không của ma trận hiệp phương sai  $\mathbf{S}$  sẽ không vượt quá hạng của nó, tức không vượt quá  $N$ . Vậy ta cần chọn  $K \leq N$  vì không thể chọn ra được nhiều hơn  $N$  trị riêng khác không của một ma trận có hạng bằng  $N$ .

Việc tính toán các trị riêng và vector riêng cũng có thể được thực hiện một cách hiệu quả dựa trên các tính chất sau đây:

- a. Trị riêng của  $\mathbf{A}$  cũng là trị riêng của  $k\mathbf{A}$  với  $k \neq 0$  bất kỳ. Điều này có thể được suy ra trực tiếp từ định nghĩa của trị riêng và vector riêng.
- b. Trị riêng của  $\mathbf{AB}$  cũng là trị riêng của  $\mathbf{BA}$  với  $\mathbf{A} \in \mathbb{R}^{d_1 \times d_2}$ ,  $\mathbf{B} \in \mathbb{R}^{d_2 \times d_1}$  là các ma trận bất kỳ và  $d_1, d_2$  là các số tự nhiên khác không bất kỳ.

Như vậy, thay vì tìm trị riêng của ma trận hiệp phương sai  $\mathbf{S} \in \mathbb{R}^{D \times D}$ , ta đi tìm trị riêng của ma trận  $\mathbf{T} = \mathbf{X}^T \mathbf{X} \in \mathbb{R}^{N \times N}$  có số chiều nhỏ hơn (vì  $N < D$ ).

- c. Nếu  $(\lambda, \mathbf{u})$  là một cặp trị riêng, vector riêng của  $\mathbf{T}$  thì  $(\lambda, \mathbf{Xu})$  là một cặp trị riêng, vector riêng của  $\mathbf{S}$ . Thật vậy:

$$\mathbf{X}^T \mathbf{X} \mathbf{u} = \mathbf{T} \mathbf{u} = \lambda \mathbf{u} \Rightarrow (\mathbf{X} \mathbf{X}^T)(\mathbf{X} \mathbf{u}) = \lambda (\mathbf{X} \mathbf{u}) \quad (2.24)$$

Dấu bằng thứ nhất xảy ra theo định nghĩa của trị riêng và vector riêng.

Như vậy, ta có thể hoàn toàn tính được trị riêng và vector riêng của ma trận hiệp phương sai  $\mathbf{S}$  dựa trên một ma trận  $\mathbf{T}$  có kích thước nhỏ hơn. Việc này trong nhiều trường hợp khiến thời gian tính toán giảm đi đáng kể.

### 2.5.2. Với các bài toán quy mô lớn

Trong rất nhiều bài toán quy mô lớn, ma trận hiệp phương sai là một ma trận rất lớn. Ví dụ, có một triệu bức ảnh  $1000 \times 1000$  pixel, như vậy  $D = N = 10^6$  là các số rất lớn, việc trực tiếp tính toán trị riêng và vector riêng cho ma trận hiệp phương sai là không khả thi. Lúc này, các trị riêng và vector riêng của ma trận hiệp phương sai thường được tính thông qua *power method* (<https://goo.gl/eBRPxH>).

## 2.6. Ứng dụng trong nén dữ liệu đa phương tiện dạng ảnh

### 2.6.1. Triển khai thuật toán PCA bằng ngôn ngữ python

Để thực hiện triển khai thuật toán chúng ta sẽ thực hiện bằng 6 bước sau :

- Tính hiệu giá trị của giá trị mỗi cột và giá trị trung bình của cột đó
- Tính ma trận hiệp phương sai dữ liệu
- Tính toán giá trị riêng và vector riêng của ma trận hiệp phương sai
- Sắp xếp các giá trị riêng theo chiều tăng dần
- Chọn tập con trong tập giá trị riêng đã sắp xếp
- Chuyển đổi dữ liệu

Đây là đoạn code thực hiện các bước trên sử dụng ngôn ngữ python và thư viện numpy (1 thư viện hỗ trợ tính toán)

```
import numpy as np

def PCA(X , num_components):

    #Step-1
    X_meaned = X - np.mean(X , axis = 0)

    #Step-2
    cov_mat = np.cov(X_meaned , rowvar = False)

    #Step-3
    eigen_values , eigen_vectors = np.linalg.eigh(cov_mat)

    #Step-4
    sorted_index = np.argsort(eigen_values)[::-1]
    sorted_eigenvalue = eigen_values[sorted_index]
    sorted_eigenvectors = eigen_vectors[:,sorted_index]

    #Step-5
    eigenvector_subset = sorted_eigenvectors[:,0:num_components]

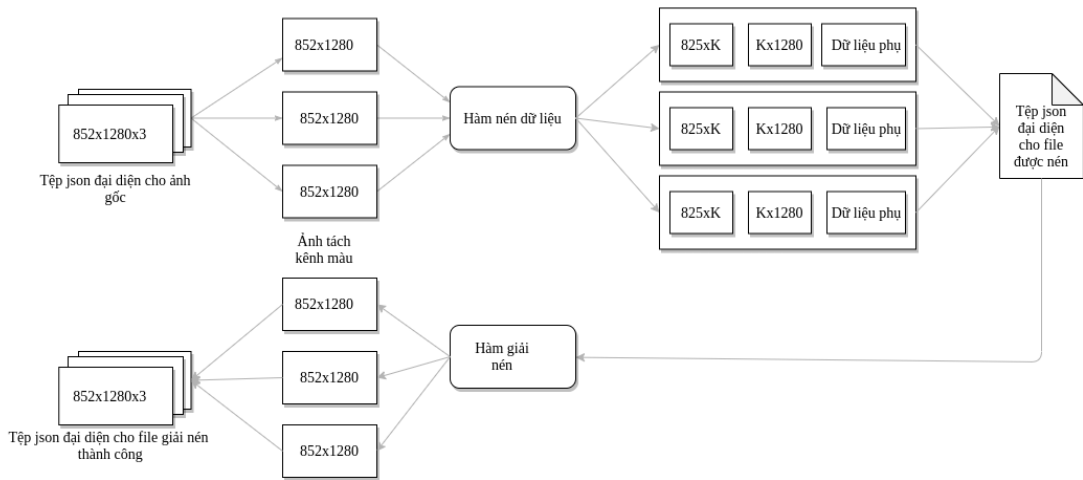
    #Step-6
    X_reduced = np.dot(eigenvector_subset.transpose() , X_meaned.
transpose() ).transpose()

    return X_reduced
```

### 2.6.2. Chương trình nén sử dụng PCA bằng ngôn ngữ python

#### *Mô hình tổng quan của chương trình*

Mô hình tập trung vào chứng minh tính giảm chiều dữ liệu của thuật toán PCA



**Hình 2.5.** Mô tả chương trình

#### *Nén và lưu trữ dữ liệu nén*

Hàm sử dụng để nén các ma trận ảnh riêng biệt kích thước mxn:

Hàm này được thực thi với các tham số là ma trận các điểm ảnh và phần trăm nén ảnh.



```
def compress_img(img, percen):
    '''compress image with percent'''

    pca = PCA().fit(img)
    var_cumu = np.cumsum(pca.explained_variance_ratio_)*100
    k = np.argmax(var_cumu > percen)

    ipca = IncrementalPCA(n_components=k)
    img_compressed = ipca.fit_transform(img)
    list_att = ['components_', 'mean_', 'explained_variance_', 'whiten']
    ipca_att = {}
    for att in list_att:
        ipca_att[att] = ipca.__getattr__(att)
    return k, img_compressed, ipca_att
```

Kết quả các giá trị mà hàm này trả về :

- Số lượng thành phần chính cần thiết để có phần trăm nén tương ứng
- Ma trận rút gọn theo các thành phần ở trên
- Các tham số cần thiết cho việc phục hồi lại ma trận ban đầu (đương nhiên là ma trận phục hồi sẽ bị mất mát dữ liệu)

### *Giải nén từ dữ liệu nén*

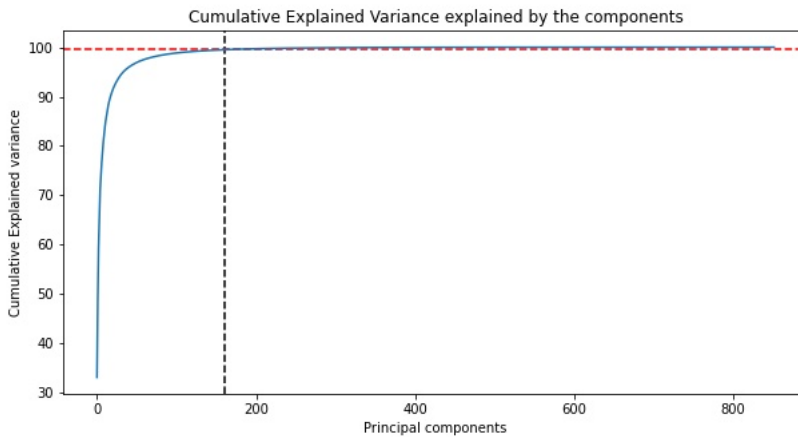
Hàm giải nén được thực thi với các tham số truyền vào là các giá trị kết quả của hàm nén

Hàm này chịu trách nhiệm phục hồi lại ảnh với kích thước ban đầu

```
def extract_img(result_compressed):
    k = result_compressed[0]
    ipca = IncrementalPCA(n_components=k)
    img_compressed = result_compressed[1]
    dict_att = result_compressed[2]
    for key in dict_att.keys():
        ipca.__setattr__(key, dict_att[key])
    # print((dict_att['mean_']))
    img_extracted = ipca.inverse_transform(img_compressed)
    return img_extracted
```

### *Phục hồi ảnh từ các ma trận riêng biệt*

### *Biểu diễn hiệu quả nén*



**Hình 2.6.** Đường thể hiện tổng cộng dồn của các giá trị riêng thành phần chính

Biểu đồ trên cho thấy khi ta lựa chọn số phần trăm nén là gần 100% thì số lượng thành phần được chọn cũng chỉ bằng 1/4 tổng số thành phần.

Nên khi gặp những ảnh có đường biểu diễn như thế thì hiệu quả nén được sẽ lớn nhưng mất mát thì nhỏ.

## 2.7. Một số ứng dụng khác

Ứng dụng đầu tiên của PCA chính là việc giảm chiều dữ liệu, giúp việc lưu trữ và tính toán được thuận tiện hơn. Thực tế cho thấy, nhiều khi làm việc trên dữ liệu đã được giảm chiều mang lại kết quả tốt hơn so với dữ liệu gốc. Thứ nhất, có thể phần dữ liệu mang thông tin nhỏ bị lược đi chính là phần gây nhiễu, những thông tin quan trọng hơn đã được giữ lại. Thứ hai, số điểm dữ liệu nhiều khi ít hơn số chiều dữ liệu. Khi có quá ít dữ liệu và số chiều dữ liệu quá lớn, quá khớp rất dễ xảy ra. Việc giảm chiều dữ liệu phần nào giúp khắc phục hiện tượng này.

Dưới đây là hai ví dụ về ứng dụng của PCA trong bài toán phân loại khuôn mặt và dò điểm bất thường.

### 2.7.1. Khuôn mặt riêng

*Khuôn mặt riêng* (eigenface) từng là một trong những kỹ thuật phổ biến trong bài toán nhận dạng khuôn mặt. Ý tưởng của khuôn mặt riêng là đi tìm một không gian có số chiều nhỏ hơn để mô tả mỗi khuôn mặt, từ đó sử dụng vector trong



**Hình 2.7.** Ví dụ về ảnh của một người trong Yale Face Database.

không gian thấp chiều này như vector đặc trưng cho bộ phân loại. Điều đáng nói là một bức ảnh khuôn mặt có kích thước khoảng  $200 \times 200$  sẽ có số chiều là 40k – một số rất lớn, trong khi đó, vector đặc trưng thường chỉ có số chiều bằng vài trăm hoặc vài nghìn. Khuôn mặt riêng thực ra chính là PCA. Các khuôn mặt riêng chính là các vector riêng ứng với những trị riêng lớn nhất của ma trận hiệp phương sai.

Trong phần này, chúng ta làm một thí nghiệm nhỏ trên *cơ sở dữ liệu khuôn mặt Yale* (<https://goo.gl/LNg8LS>). Các bức ảnh trong thí nghiệm này đã được căn chỉnh cho cùng với kích thước và khuôn mặt nằm trọn vẹn trong một hình chữ nhật có kích thước  $116 \times 98$  điểm ảnh. Có tất cả 15 người khác nhau, mỗi người có 11 bức ảnh được chụp ở các điều kiện ánh sáng và cảm xúc khác nhau, bao gồm 'centerlight', 'glasses', 'happy', 'leftlight', 'noglasses', 'normal', 'rightlight', 'sad', 'sleepy', 'surprised', và 'wink'. Hình 2.7 minh họa các bức ảnh của người có id là 10.

Ta thấy rằng số chiều dữ liệu  $116 \times 98 = 11368$  là một số khá lớn. Tuy nhiên, vì chỉ có tổng cộng  $15 \times 11 = 165$  bức ảnh nên ta có thể nén các bức ảnh này về dữ liệu mới có chiều nhỏ hơn 165. Trong ví dụ này, chúng ta chọn  $K = 100$ .

Dưới đây là đoạn code thực hiện PCA cho toàn bộ dữ liệu. Ở đây, PCA trong `sklearn` được sử dụng:

```
from sklearn.decomposition import PCA
import numpy as np
import scipy.misc # for loading image
from matplotlib.pyplot import imread
np.random.seed(1)

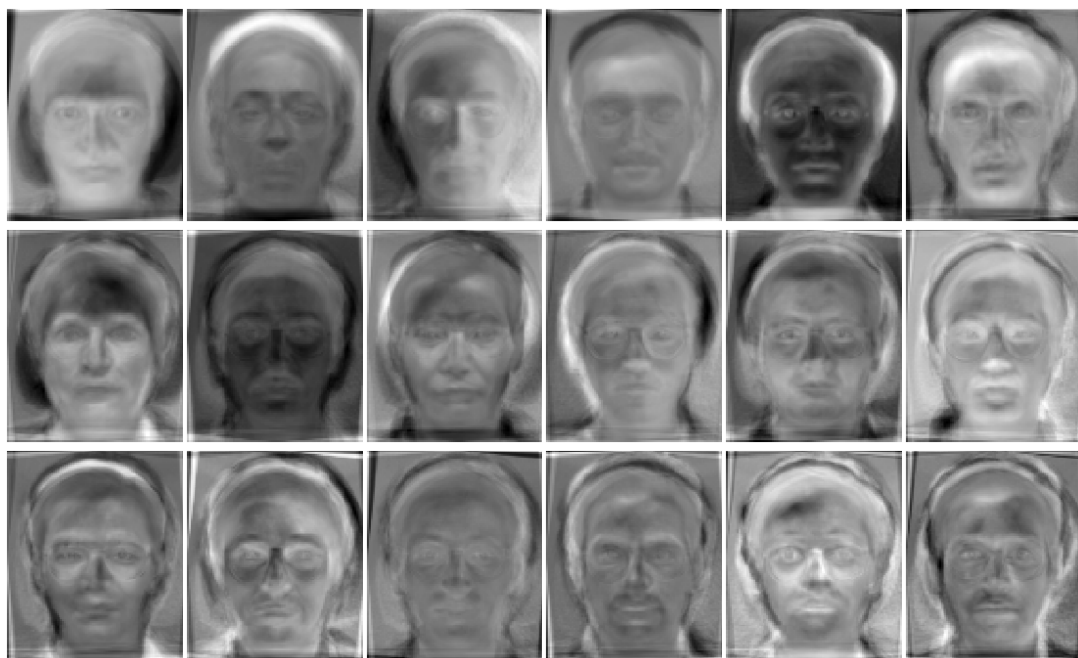
# filename structure
path = 'YALE/unpadded/' # path to the database
ids = range(1, 16) # 15 persons
states = ['centerlight', 'glasses', 'happy', 'leftlight',
          'noglasses', 'normal', 'rightlight', 'sad',
          'sleepy', 'surprised', 'wink']
prefix = 'subject'
surfix = '.pgm'
# data dimension
h, w, K = 116, 98, 100 # hight, weight, new dim
D = h * w
N = len(states)*15
# collect all data
X = np.zeros((D, N))
cnt = 0
for person_id in range(1, 16):
    for state in states:
        fn = path + prefix + str(person_id).zfill(2) + '.' + state +
surfix
        print(fn)
        X[:, cnt] = imread(fn).reshape(D)
        # misc.imread
        cnt += 1

# Doing PCA, note that each row is a datapoint
pca = PCA(n_components=K) # K = 100
pca.fit(X.T)
# projection matrix
U = pca.components_.T
```

Trong dòng `pca = PCA(n_components=K)`, nếu `n_components` là một số thực trong khoảng  $(0, 1)$ , PCA sẽ thực hiện việc tìm  $K$  dựa trên biểu thức (2.23).

Hình 2.8 biểu diễn 18 vector riêng đầu tiên (18 cột đầu tiên của  $\mathbf{U}_k$ ) tìm được bằng PCA. Các vector đã được **reshape** về cùng kích thước như các bức ảnh gốc. Nhận thấy các vector thu được ít nhiều mang thông tin của mặt người. Thực tế, một khuôn mặt gốc sẽ được xấp xỉ như tổng có trọng số của các khuôn mặt này. Vì các vector riêng này đóng vai trò như cơ sở của không gian mới với ít chiều hơn, chúng còn được gọi là *khuôn mặt riêng* hoặc *khuôn mặt chính*. Từ *chính* được dùng vì nó đi kèm với văn cảnh của *phân tích thành phần chính*.

Để xem mức độ hiệu quả của phương pháp này, chúng ta minh họa các bức ảnh gốc và các bức ảnh được xấp xỉ bằng PCA như trên Hình 2.9. Các khuôn mặt nhận được vẫn mang khá đầy đủ thông tin của các khuôn mặt gốc. Đáng

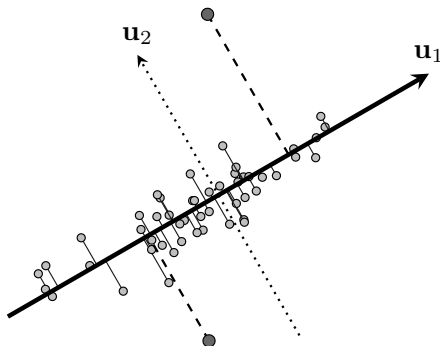


**Hình 2.8.** Các eigenfaces tìm được bằng PCA.



**Hình 2.9.** Hàng trên: các ảnh gốc. Hàng dưới: các ảnh được tái tạo dùng khuôn mặt riêng. Ảnh ở hàng dưới có nhiều nhưng vẫn mang những đặc điểm riêng mà mắt người có thể phân biệt được.

chú ý hơn, các khuôn mặt trong hàng dưới được suy ra từ một vector 100 chiều, so với 11368 chiều như ở hàng trên.



**Hình 2.10.** PCA cho bài toán dò tìm điểm bất thường. Giả sử các sự kiện bình thường chiếm đa số và nằm gần một không gian con nào đó. Khi đó, nếu làm PCA trên toàn bộ dữ liệu, không gian con thu được gần với không gian con của tập các sự kiện bình thường. Lúc này, các điểm hình tròn to đậm hơn có thể được coi là các sự kiện bất thường vì chúng nằm xa không gian con chính.

### 2.7.2. Dò tìm điểm bất thường

Ngoài các ứng dụng về nén và phân loại, PCA còn được sử dụng trong nhiều lĩnh vực khác. *Dò tìm điểm bất thường* (abnormal detection hoặc outlier detection) là một trong số đó [SCSC03, LCD04].

Ý tưởng cơ bản là giả sử tồn tại một không gian con mà các sự kiện bình thường nằm gần trong khi các sự kiện bất thường nằm xa không gian con đó. Hơn nữa, số sự kiện bất thường có một tỉ lệ nhỏ. Như vậy, PCA có thể được sử dụng trên toàn bộ dữ liệu để tìm ra các thành phần chính, từ đó suy ra không gian con mà các điểm bình thường nằm gần. Việc xác định một điểm là bình thường hay bất thường được xác định bằng cách đo khoảng cách từ điểm đó tới không gian con tìm được. Hình 2.10 minh họa cho việc xác định các sự kiện bất thường bằng PCA.

## 2.8. Kết luận

- PCA là phương pháp giảm chiều dữ liệu dựa trên việc tối đa lượng thông tin được giữ lại. Lượng thông tin được giữ lại được đo bằng tổng các phương sai trên mỗi thành phần của dữ liệu. Lượng dữ liệu sẽ được giữ lại nhiều nhất khi các chiều dữ liệu còn lại tương ứng với các vector riêng của trị riêng lớn nhất của ma trận hiệp phương sai.
- Với các bài toán quy mô lớn, đôi khi việc tính toán trên toàn bộ dữ liệu là không khả thi vì vấn đề 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 từ từ (*incrementally*) cập nhật nghiệm của PCA tới khi hội tụ. Ý tưởng này khá giống với mini-batch gradient descent, và được gọi là incremental PCA [ZYK06].

## Tài liệu tham khảo

- [LCD04] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 219–230. ACM, 2004.
- [SCSC03] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, 2003.
- [ZYK06] Haitao Zhao, Pong Chi Yuen, and James T Kwok. A novel incremental principal component analysis and its application for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(4):873–886, 2006.

---

## Index

- cơ sở dữ liệu khuôn mặt Yale – Yale face database, 19
- dimensionality reduction – giảm chiều dữ liệu, 2
- eigenface – khuôn mặt riêng, 18
- feature extraction – trích chọn đặc trưng, 2
- feature selection – lựa chọn đặc trưng, 2
- giảm chiều dữ liệu – dimensionality reduction, 2
- khuôn mặt riêng – eigenface, 18
- lựa chọn đặc trưng – feature selection, 2
- PCA, 5
- phân tích thành phần chính – principle component analysis, 5
- principle component analysis – phân tích thành phần chính, 5
- trích chọn đặc trưng – feature extraction, 2
- Yale face database – cơ sở dữ liệu khuôn mặt Yale, 19