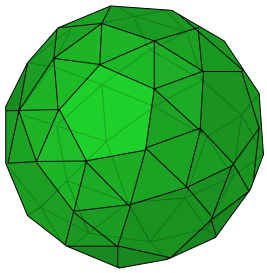
Projects in Mathematics and Applications

PRINCIPAL COMPONENT ANALYSIS

Ngày 27 tháng 8 năm 2018

Trần Thanh Bình *
Lê Quang Kỳ ‡
†Võ Thục Khánh Huyền
§Đỗ Nhật Hoàng



*Trường Phổ Thông Năng Khiếu
†Trường THPT Chuyên Lê Quý Đôn, Quảng Trị
‡Trường THPT Chuyên Hoàng Lê Kha, Tây Ninh
§Trường Phổ Thông Năng Khiếu

Lời cảm ơn

Thời gian được học tập ở Trại hè PiMA 2018 (Project in Mathematics and Applications) là quãng thời gian vô cùng ý nghĩa. Kết quả ngày hôm nay chúng tôi đạt được một phần là nhờ có sự quan tâm hỗ trợ của nhiều cá nhân và các đoàn thể. Trước hết chúng tôi xin gửi lời cảm ơn đến các anh chị Mentor, đặc biệt là anh Cấn Trần Thành Trung và Ban Tổ Chức PiMA đã giảng dạy cho chúng tôi về những kiến thức Toán thú vị, chia sẻ cho chúng tôi những cơ hội để tiếp cận với Toán ứng dụng, và giúp chúng tôi hoàn thành được đề tài một cách tốt nhất. Bên cạnh đó, chúng tôi cũng muốn bày tỏ lòng biết ơn đến Trường Đại học Khoa học Tự nhiên Thành phố Hồ Chí Minh đã tạo điều kiện tốt nhất về vật chất và tinh thần để chúng tôi có thể tập trung tối đa vào làm dự án nhóm. Và cuối cùng, chúng tôi muốn cảm ơn các bạn trại sinh đã cùng nhau học tập và vui chơi với chúng tôi trong suốt thời gian qua. Do thời gian làm dự án còn ngắn nên thiếu sót là điều không thể tránh khỏi. Nhóm chúng tôi mong sẽ nhận được đóng góp từ phía độc giả để dự án có thể hoàn thiện tốt hơn.

Tóm tắt nội dung

Những dữ liệu trong thực tế thường có số chiều và số lượng rất lớn, gây khó khăn cho việc lưu trữ và xử lý. Để giải quyết vấn đề này, người ta xây dựng các phương pháp làm giảm chiều dữ liệu nhưng vẫn giữ được những thông tin chính. Phương pháp Principal Component Analysis (PCA), phân tích thành phần chính, là một trong những phương pháp phổ biến nhất. Trong bài viết này, chúng tôi sẽ trình bày và chứng minh lại cơ sở toán học của phương pháp PCA. Do việc tìm ra các thành phần chính của bộ dữ liệu trong PCA tương đương với việc tìm trị riêng và vector riêng của những ma trận có kích thước lớn, bài viết sẽ giới thiệu thêm phương pháp Power để xấp xỉ trị riêng và vector riêng. Sau đó, chúng tôi áp dụng PCA để biểu diễn dữ liệu nhiều chiều và tìm những nét chính của bộ hình ảnh khuôn mặt người.

Mục lục

1 Giới thiệu bài toán Giảm chiều dữ liệu (Dimensionality Reduction)

1.1 Bài toán giảm chiều dữ liệu

Thông tin trong khoa học dữ liệu nói chung và Machine Learning nói riêng thường được mô tả bằng những vector có số chiều và số phần tử rất lớn, gây khó khăn trong việc lưu trữ và xử lí dữ liệu. Nếu không gian dữ liệu gốc có m tính trạng thì các vector dữ liệu sẽ là những điểm $\vec{x} \in \mathbb{R}^m$. Giảm chiều dữ liệu (Dimensionality Reduction) thực chất là tìm một hàm số $f : \mathbb{R}^m \to \mathbb{R}^k$ phù hợp, với $k < m$, để chuyển sang làm việc với các điểm dữ liệu mới $\vec{z} = f(\vec{x})$ có số chiều nhỏ hơn. Những hàm f phổ biến nhất được chọn thường là phép chiếu lên một không gian con thỏa mãn những tính chất nhất định.

1.2 Các hướng tiếp cận bài toán Giảm chiều dữ liệu

Hai hướng tiếp cận chủ yếu trong bài toán Giảm chiều dữ liệu là *Feature Selection* (lựa chọn những tính trạng liên quan) và *Feature Extraction* (xây dựng những tính trạng mới chứa nhiều thông tin hơn). Trong *Feature Extraction*, một số phương pháp thường gặp bao gồm phương pháp *Linear Discriminant Analysis (LDA)*, *Canonical Correlation Analysis (CCA)*, hay *Autoencoder*.

Phép *Phân tích thành phần chính (PCA)* có thể dùng trong cả Selection lẫn Extraction. *Phương pháp PCA* sẽ biểu diễn các vector dữ liệu trong hệ cơ sở trực chuẩn mới (tương ứng với cơ sở riêng), sao cho tầm quan trọng của mỗi chiều dữ liệu là khác nhau. Khi đó, chúng ta có thể chọn và giữ lại k *chiều quan trọng* nhất, tương ứng với việc chọn f là phép chiếu lên k chiều tương ứng. Cơ sở toán của PCA sẽ được trình bày cụ thể ở các phần sau. PCA có rất nhiều ứng dụng như nén ảnh (Image Compression) và nhận dạng khuôn mặt (Facial Recognition).

2 Cơ sở Toán học

2.1 Một số kiến thức quan trọng

Định nghĩa 2.1. (Biểu diễn trong cơ sở). Cho không gian vector U và một cơ sở hữu hạn $S = \{\vec{u}_i : i = 1, 2, \ldots, n\}$ của U. Với mỗi vector $\vec{u} \in U$, ký hiệu $\left[\vec{u}\right]_S$ là biểu diễn của $\vec{u}$ qua cơ sở S:

$$\left[\vec{u}\right]_S = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix},$$

nếu $\vec{u}$ là một vector trong $\mathbb{R}^n$ thỏa mãn:

$$\vec{u} = x_1\vec{u}_1 + x_2\vec{u}_2 + \cdots + x_n\vec{u}_n. \quad (1)$$

Định nghĩa 2.2. Cơ sở S được gọi là **trực giao** nếu $\vec{u}_i \cdot \vec{u}_j = 0$, $\forall i \neq j \in \{1, 2, \ldots, n\}$.

Định nghĩa 2.3. Cơ sở S được gọi là **chuẩn** nếu $\vec{u}_i \cdot \vec{u}_i = 1$, $\forall i \in \{1, 2, \ldots, n\}$.

Định nghĩa 2.4. Cơ sở S được gọi là **trực chuẩn** nếu nó chuẩn và trực giao.

Nhận xét 2.5. *Biểu diễn vector $\vec{u} \in U$ trong cơ sở trực chuẩn S:*

$$\begin{aligned}\vec{u} &= \text{proj}_{\vec{u}_1}(\vec{u}) + \text{proj}_{\vec{u}_2}(\vec{u}) + \cdots + \text{proj}_{\vec{u}_n}(\vec{u}) \\ &= (\vec{u} \cdot \vec{u}_1)\vec{u}_1 + (\vec{u} \cdot \vec{u}_2)\vec{u}_2 + \cdots + (\vec{u} \cdot \vec{u}_n)\vec{u}_n.\end{aligned}$$

Chứng minh. Theo định nghĩa trên, ta có:

$$\vec{u} = x_1\vec{u}_1 + x_2\vec{u}_2 + \cdots + x_n\vec{u}_n. \ (1)$$

Nhân $\vec{u}_i$ vào 2 vế của phương trình (1) ta được: $\vec{u} \cdot \vec{u}_i = x_i$ với $\forall i \in \{1, 2, \ldots, n\}$.
Do đó $\vec{u} = (\vec{u} \cdot \vec{u}_1)\vec{u}_1 + (\vec{u} \cdot \vec{u}_2)\vec{u}_2 + \cdots + (\vec{u} \cdot \vec{u}_n)\vec{u}_n$.

Mặt khác, ta có công thức hình chiếu như sau:

$$\text{proj}_{\vec{u}_i}(\vec{u}) = \frac{\vec{u} \cdot \vec{u}_i}{\|\vec{u}_i\|^2}\vec{u}_i = (\vec{u} \cdot \vec{u}_i)\vec{u}_i \text{ với } \forall i \in \{1, 2, \ldots, n\}.$$

Do đó $\vec{u} = \text{proj}_{\vec{u}_1}(\vec{u}) + \text{proj}_{\vec{u}_2}(\vec{u}) + \cdots + \text{proj}_{\vec{u}_n}(\vec{u})$. □

Nhận xét 2.6. *Từ các định nghĩa trên, ta thấy nếu biểu diễn vector $\vec{u} \in U$ trong cơ sở trực chuẩn S thì ta có thể bình phương biểu thức (1) và rút gọn để trở thành:*

$$\|\vec{u}\|^2 = x_1^2 + x_2^2 + \cdots + x_n^2.$$

Do đó, người ta thường biểu diễn tại vector dữ liệu trong cơ sở mới để thuận tiện trong việc tính toán và biểu diễn.

Định nghĩa 2.7. (Trị riêng và vector riêng của ma trận). Cho $A \in \mathbb{R}^{n\times n}$. Ta nói $\lambda \in \mathbb{R}$ là một **trị riêng** (hoặc giá trị riêng) của A nếu tồn tại $v \in \mathbb{R}^n \setminus \{0\}$ sao cho:

$$Av = \lambda v.$$

Lúc đó ta nói v là **vector riêng** ứng với trị riêng λ.
Đa thức đặc trưng của A được kí hiệu P_A: $\mathbb{R} \to \mathbb{R}$ sao cho $P_A(x) = \det(A - xI_n)$.

Định lý 2.8. *Ma trận đối xứng cấp n có n trị riêng thực.*

Định lý 2.9. *Trị riêng của ma trận $A \in \mathbb{R}^{n\times n}$ là nghiệm của phương trình $P_A(\lambda) = 0$.*

Phương Pháp Nhân Tử Lagrange (Lagrange Multiplier Method)
Ta xét bài toán tối ưu: $\min\limits_{x\in\mathbb{R}^n} f(x)$, thỏa mãn:

$$g_i(x) = 0, i \in \{1, 2, \ldots, m\},$$

với các hàm f, g_i: $\mathbb{R}^n \to \mathbb{R}$ là các hàm khả vi với đạo hàm riêng liên tục.
Để giải quyết bài toán này, ta sẽ phát biểu một điều kiện cần để điểm x^* là một cực trị địa phương của hàm f.
Ta định nghĩa **Lagrangian** $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ của bài toán tối ưu:

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

với $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m]$ được gọi là **biến đổi ngẫu** hay **vector nhân tử Lagrange**.
Nếu x^* là một điểm tối thiểu địa phương của bài toán tối ưu trên, ta có:

$$\exists \lambda^* : \bigtriangledown_{x,\lambda}\mathcal{L}(x^*, \lambda^*) = 0.$$

Để tìm cực tiểu toàn cục, ta có thể giải hệ phương trình có được từ điều kiện ở trên rồi xét thử tất cả các bộ nghiệm tìm được.

2.2 Một số kết quả cơ bản

Cho $A \in \mathbb{R}^{m\times n}$. Khi đó $AA^T \in \mathbb{R}^{m\times m}$ và $A^TA \in \mathbb{R}^{n\times n}$. Đặt $B = A^TA$.

Tính chất 2.10. Ma trận B đối xứng.

Chứng minh. Vì $B^T = (A^TA)^T = A^TA = B$ nên $B^T = B$. □

Tính chất 2.11. Chúng ta biết rằng ma trận thực đối xứng có trị riêng là các số thực. Giả sử λ, μ là 2 trị riêng khác nhau của B với các vector riêng tương ứng $\vec{v}_\lambda$, $\vec{v}_\mu$. Chứng minh rằng $\vec{v}_\lambda \cdot \vec{v}_\mu = 0$. Suy ra các vector riêng ứng với trị riêng khác nhau của B trực giao và độc lập tuyến tính.

Chứng minh. Với $B \in \mathbb{R}^{n\times n}$ và $x, y \in \mathbb{R}^n$, ta có:

1. $x \cdot y = x^Ty$.
2. $Bx \cdot y = (Bx)^T \cdot y = (x^TB^T) \cdot y = x^T \cdot (B^Ty) = x \cdot B^Ty$.

Áp dụng với λ, μ là 2 trị riêng khác nhau của B với các vector riêng tương ứng $\vec{v}_\lambda$, $\vec{v}_\mu$, ta có:

$$B\vec{v}_\lambda = \lambda\vec{v}_\lambda \text{ (1) và } B\vec{v}_\mu = \mu\vec{v}_\mu \text{ (2).}$$

Nhân $\vec{v}_\mu$ vào 2 vế của (1) ta có: $\vec{v}_\mu \cdot (B\vec{v}_\lambda) = \lambda\vec{v}_\mu \cdot \vec{v}_\lambda$.
Áp dụng công thức 2 vào biểu thức trên ta có: $(B^T\vec{v}_\mu)\vec{v}_\lambda = \lambda\vec{v}_\mu \cdot \vec{v}_\lambda$.
Mặt khác $B = B^T$ và kết hợp (2) nên biểu thức trên được viết lại thành: $\mu\vec{v}_\mu \cdot \vec{v}_\lambda = \lambda\vec{v}_\mu \cdot \vec{v}_\lambda$.
Hay $(\mu - \lambda)\vec{v}_\mu \cdot \vec{v}_\lambda = 0$, mà $\vec{v}_\mu$ khác $\vec{v}_\lambda$ nên $\vec{v}_\lambda \cdot \vec{v}_\mu = 0$.
Do đó, ta suy ra được các vector riêng ứng với trị riêng khác nhau của B **trực giao**.

□

Ma trận đối xứng $B \in \mathbb{R}^{n\times n}$ thì có đúng n trị riêng thực, gọi $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ là n vector riêng ứng với n trị riêng đó. Giả sử tồn tại các số $c_1, c_2, \ldots, c_n \in \mathbb{R}$ thỏa mãn:

$$c_1\vec{v}_1 + c_2\vec{v}_2 + \cdots + c_n\vec{v}_n = \vec{0}.$$

Bình phương hai vế của biểu thức trên và sử dụng điều kiện $\vec{v}_i \cdot \vec{v}_j = 0$ với $i \neq j$; $i, j \in \{1, 2, \ldots, n\}$, ta có:

$$\sum_{i=1}^{n} c_i^2 \|\vec{v}_i\|^2 = 0,$$

nên $c_i = 0$ với $\forall i \in \{1, 2, \ldots, n\}$.
Do đó, ta suy ra được các vector riêng ứng với trị riêng khác nhau của B **độc lập tuyến tính**, hay chúng tạo thành một hệ cơ sở.

Ý nghĩa: Ta sẽ biểu diễn bộ dữ liệu dưới cơ sở S trực giao mới mà $S = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n\}$. Cơ sở trực chuẩn giúp việc tính toán biểu diễn của một vector và tích vô hướng giữa các vector dễ dàng hơn.

Tính chất 2.12. Chứng minh rằng A^TA và AA^T có cùng giá trị riêng (Lưu ý: có thể khác nhau về số lần lặp của trị riêng 0)

Chứng minh. Gọi $\vec{v}$ là vector riêng của A^TA ứng với trị riêng λ ($\vec{v} \neq \vec{0}$, $\lambda \neq 0$). Ta có:

$$(A^TA)\vec{v} = \lambda\vec{v} \text{ hay } A^T(A\vec{v}) = \lambda\vec{v}.$$

Nhân hai vế của biểu thức cho A và gộp nhóm lại như sau:

$$AA^T(A\vec{v}) = \lambda(A\vec{v}).$$

Điều này cho chúng ta thấy $A\vec{v}$ là vector riêng của AA^T ứng với trị riêng $\lambda \neq 0$. Bây giờ chúng ta cần đi chứng minh $A\vec{v}$ khác vector 0. Nếu $A\vec{v} = \vec{0}$ thì theo biểu thức trên $\lambda\vec{v} = \vec{0}$, vô lý vì $\vec{v} \neq \vec{0}$, $\lambda \neq 0$. Do đó những trị riêng $\lambda \neq 0$ của A^TA cũng là những trị riêng của AA^T. □

Ý nghĩa: Kết quả này thực sự hiệu quả, nếu ta xét ma trận $A \in \mathbb{R}^{100\times 3}$. Ta sẽ mất rất nhiều thời gian để đi tìm trị riêng của ma trận $AA^T \in \mathbb{R}^{100\times 100}$, thay vào đó ta sẽ đi tìm trị riêng của ma trận $A^TA \in \mathbb{R}^{3\times 3}$. Đó cũng chính là trị riêng của ma trận AA^T, 97 trị riêng còn lại bằng 0.

3 Mô hình hóa bài toán PCA

Ta sẽ xét một bộ dữ liệu gồm n điểm dữ liệu, mỗi điểm dữ liệu có m tính trạng. Đối với bài toán PCA, ta sẽ biểu diễn mỗi điểm dữ liệu thành một vector m chiều, mỗi chiều sẽ ứng với một tính trạng. Ở đây, chúng ta thống nhất nếu không nói gì thêm thì $\vec{X}$ là vector cột:

$$\vec{X_i} = \begin{bmatrix} x_{1,i} \\ x_{2,i} \\ \vdots \\ x_{m,i} \end{bmatrix}, \text{ với } i = 1, 2, \ldots, n.$$

Ta xét ma trận sau:

$$X_{m\times n} = \begin{bmatrix} \vec{X_1} & \vec{X_2} & \ldots & \vec{X_n} \end{bmatrix}.$$

Viết đầy đủ, ta có:

$$X_{m\times n} = \begin{bmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,n} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \ldots & x_{m,n} \end{bmatrix}.$$

Tiếp theo, chúng ta sẽ định nghĩa thế nào là một *chiều quan trọng* của dữ liệu. *Chiều quan trọng* của dữ liệu có thể hiểu là chiều của một vector $\vec{w}$ trong không gian $\mathbb{R}^m$ mà theo phương là giá của vector đó, *độ phân tán* của các tính trạng là lớn nhất. Và để xác định *độ phân tán*, ta sẽ dùng đến phương sai. Như vậy, chiều quan trọng nhất sẽ được định nghĩa là chiều mà trên đó phương sai của bộ dữ liệu là lớn nhất.

Ta có công thức phương sai:

$$\text{Var}_{\vec{w_i}}(X) = \sigma^2_{\vec{w_i}}(X) = \frac{1}{n}\sum_{j=1}^{n}(x_{i,j} - \mu_i)^2, \text{ với } i = 1, 2, \ldots, n.$$

Trong đó, $\mu_i = \bar{x_i} = \dfrac{x_{i,1} + x_{i,2} + \ldots + x_{i,n}}{n}$ là giá trị trung bình hay giá trị kì vọng của mỗi tính trạng.

Nhận thấy rằng việc chuẩn hóa ma trận này về quanh gốc tọa độ không làm thay đổi bản chất về *độ phân tán* của bộ dữ liệu, nhưng lại làm cho việc tính toán trở nên dễ dàng hơn. Vì vậy, ta có thể xét việc biến đổi trên ma trận $\hat{X}_{m\times n}$ như sau:

$$\hat{X}_{m\times n} = [x_{i,j} - \mu_i] = \begin{bmatrix} \vec{x_1} & \vec{x_2} & \dots & \vec{x_n} \end{bmatrix}.$$

Với bộ dữ liệu $\hat{X}_{m\times n}$, ta có $\mu_{\hat{X}_{m\times n}} = \vec{0}$, ta có định nghĩa sau:

$$\text{Var}_{\vec{w_i}}(X) = \frac{1}{n}\sum_{i=1}^{n} \| \text{proj}_{\vec{w_i}} \vec{x_i}\|^2.$$

Để tìm *chiều quan trọng* nhất, ta sẽ tìm $\vec{w_i}$ sao cho $\sum_{i=1}^{n} \| \text{proj}_{\vec{w_i}} \vec{x_i}\|^2$ đạt giá trị lớn nhất.
Nhớ lại công thức tính $\| \text{proj}_{\vec{w_i}} \vec{x_i}\|$:

$$\| \text{proj}_{\vec{w_i}} \vec{x_i}\| = \| \frac{\vec{x_i}\cdot\vec{w_i}}{\|w_i\|^2}\vec{w_i}\|.$$

Như vậy, nếu ta chuẩn hóa $\vec{w_i}$ sao cho $\|\vec{w_i}\| = 1$ thì công thức tính $\text{proj}_{\vec{w_i}} \vec{x_i}$ có thể được viết lại:

$$\| \text{proj}_{\vec{w_i}} \vec{x_i}\| = |\vec{x_i}\cdot\vec{w_i}|.$$

Khi đó, công thức tính $\vec{w_1}$ có thể được viết lại thành:

$$\vec{w_1} = \underset{\|\vec{w_1}\|=1}{\text{argmax}} \sum_{i=1}^{n} (\vec{w_1}\cdot\vec{x_i})^2.$$

Hay có thể được viết gọn lại thành:

$$\vec{w_1} = \underset{\|\vec{w_1}\|=1}{\text{argmax}} [(\hat{X}^T_{m\times n} w_1)\cdot(\hat{X}^T_{m\times n} w_1)].$$

Áp dụng công thức biến đổi $x\cdot y = x^T y$, ta có:

$$\vec{w_1} = \underset{\|\vec{w_1}\|=1}{\text{argmax}} (w_1^T \hat{X}\hat{X}^T w_1).$$

Vậy, việc xác định thành phần quan trọng nhất của bộ dữ liệu đã được mô hình hóa thành việc giải quyết bài toán tối ưu trên.

Sau khi đã có thành phần quan trọng nhất, ta sẽ tiếp tục đi tìm các thành phần còn lại. Để thuận tiện cho việc quan sát và tính toán, ta sẽ tìm thành phần quan trọng thứ i đôi một vuông góc với những thành phần trước nó. Hay nói khác đi, ta sẽ tìm $\vec{w_i}$ sao cho $\text{Var}_{\vec{w_i}}(X)$ đạt max và $\vec{w_i}$ thỏa $\vec{w_i}\cdot\vec{w_j} = 0, \forall j < i$. Biến đổi tương tự như đối với chiều thứ nhất, ta sẽ có:

$$\vec{w_i} = \underset{\substack{\|\vec{w_i}\|=1 \\ \vec{w_i}\cdot\vec{w_j}=0, \forall j<i}}{\text{argmax}} (w_i^T \hat{X}\hat{X}^T w_i).$$

4 Giải quyết bài toán PCA

Bài toán PCA đã được mô hình hóa về việc tìm những chiều dữ liệu. Bây giờ ta sẽ giải bài toán tối ưu đã được nhắc đến ở phần trên. Vì đây là một bài toán tối ưu hóa có điều kiện, phương pháp đầu tiên chúng ta nghĩ đến là *phương pháp nhân tử Lagrange.*

4.1 Phương pháp nhân tử Lagrange

Ta có hàm cần tối ưu:

$$\vec{w_1} = \underset{\|\vec{w_1}\|=1}{\text{argmax}}(w_1^T \hat{X}\hat{X}^T w_1),$$

và có Lagrangian:

$$\mathcal{L}(\vec{w_1}, \lambda) = w_1^T \hat{X}\hat{X}^T w_1 + \lambda(\|w_1\|^2 - 1).$$

Đặt $A = \hat{X}\hat{X}^T$, ta có $A \in \mathbb{R}^{m\times m}$ và $A = A^T$. Viết lại công thức:

$$\mathcal{L}(\vec{w_1}, \lambda) = w_1^T A w_1 + \lambda(\|w_1\|^2 - 1).$$

Đặt $\vec{w_1} = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ \vdots \\ w_{1,m} \end{bmatrix}$, khi đó: $w_1^T A w_1 = \sum_{i=1}^{m}\sum_{j=1}^{m} w_{1,i} w_{1,j} a_{ij}$.

Vậy ta có:

$$\mathcal{L}(\vec{w_1}, \lambda) = \sum_{i=1}^{m}\sum_{j=1}^{m} w_{1,i} w_{1,j} a_{ij} + \lambda(\|w_1^2\| - 1).$$

Điều kiện cần để $\mathcal{L}$ đạt giá trị lớn nhất: $\bigtriangledown\mathcal{L}(\vec{w_1}, \lambda) = \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial w_{1,1}} \\ \frac{\partial\mathcal{L}}{\partial w_{1,2}} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial w_{1,m}} \end{bmatrix} = \vec{0}.$

Như vậy, nếu $\mathcal{L}$ đạt *max* thì $\vec{w_1}$ thỏa:

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial w_{1,i}} & = 0, \forall i = 1, 2, \ldots, m \\ \frac{\partial\mathcal{L}}{\partial\lambda} & = 0. \end{cases}$$

Ta có:

$$\frac{\partial\mathcal{L}}{\partial w_{1,i}} = 2 \cdot \sum_{j=1}^{m} a_{ij} w_{1,j} + 2\lambda w_{1,i}.$$

Như vậy, hệ có thể viết lại thành:

$$\begin{aligned} &(A + \lambda I_n)\, \vec{w_1} = 0 \\ &\quad \Leftrightarrow A\vec{w_1} = -\lambda\vec{w_1}. \end{aligned}$$

Do đó, $\vec{w_1}$ là một vector riêng của A và $-\lambda$ là một trị riêng của A.

Thế điều kiện cần vào hàm nhân tử Lagrange:

$$\begin{aligned}\mathcal{L}(\vec{w_1}, \lambda) &= w_1^T A w_1 \\ &= w_1^T (-\lambda\ w_1) \\ &= -\lambda \|w_1\|^2 \\ &= -\lambda.\end{aligned}$$

Như vậy, $\max\left(w_1^T X X^T w_1\right) = -\lambda_{max}$ khi và chỉ khi $\vec{w_1}$ là vector riêng có độ dài bằng 1 ứng với $-\lambda_{max}$ (trị riêng lớn nhất của ma trận XX^T).

Giải các bài toán tối ưu kia tương tự bằng *phương pháp nhân tử Lagrange*, ta sẽ tìm được kết quả là hệ cơ sở gồm các vector riêng có độ dài 1.

Sau khi đã giải ra hệ nghiệm vector bằng *phương pháp nhân tử Lagrange*, ta nghĩ đến một phương pháp thứ hai đẹp hơn như sau.

4.2 Phương pháp chuyển hệ cơ sở

Với chiều đầu tiên, bài toán của chúng ta là việc tối ưu hóa hàm:

$$\vec{w_1} = \underset{\|\vec{w_1}\|=1}{\operatorname{argmax}}(w_1^T \hat{X} \hat{X}^T w_1).$$

Đặt $A = \hat{X}\hat{X}^T \in \mathbb{R}^{m \times m}$. A là một ma trận đối xứng nên sẽ có m vector riêng ứng với m trị riêng. Gọi $\vec{v_1}, \vec{v_2}, \ldots, \vec{v_m}$ là m vector riêng của ma trận A ứng với m trị riêng $\lambda_1, \lambda_2, \ldots, \lambda_m$ thỏa $\|v_i\| = 1$ $(\forall i = 1, 2, \ldots, m)$. Ta thêm điều kiện $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$. Như chứng minh trên, ta có $V = \{\vec{v_1}, \vec{v_2}, \ldots, \vec{v_m}\}$ là một cơ sở trực chuẩn trong không gian $\mathbb{R}^{m \times m}$. Ta sẽ biểu diễn $\vec{w_1}$ theo cơ sở V:

$$\vec{w_1} = c_{1,1}\vec{v_1} + c_{1,2}\vec{v_2} + \ldots c_{1,m}\vec{v_m}.$$

Vì $\|\vec{w_1}\| = 1$ nên ta có điều kiện $\sum_{i=1}^{m} c_{1,i}^2 = 1$. Đồng thời, theo định nghĩa của trị riêng và vector riêng, ta có:

$$A\vec{v_i} = \lambda_i \vec{v_i}.$$

Suy ra:

$$\begin{aligned}\vec{w_1} &= \underset{\|\vec{w_1}\|=1}{\operatorname{argmax}}(w_1^T A w_1) \\ &= \underset{\|\vec{w_1}\|=1}{\operatorname{argmax}}[\vec{w_1} \cdot (A \sum_{i=1}^{m} c_{1,i}\vec{v_i})] \\ &= \underset{\|\vec{w_1}\|=1}{\operatorname{argmax}}(\vec{w_1} \cdot \sum_{i=1}^{m} \lambda_i c_{1,i}\vec{v_i}).\end{aligned}$$

Vì $\vec{v_i} \cdot \vec{v_j} = 0$, $\forall i \neq j$ nên:

$$\vec{w_1} = \underset{\|\vec{w_1}\|=1}{\operatorname{argmax}} \sum_{i=1}^{m} \lambda_i c_{1,i}^2 v_i^2.$$

Đến đây, vì ta đã chuẩn hóa cho $\|\vec{v}_i\| = 1$ nên ta có thể rút gọn được công thức như sau:

$$\vec{w}_1 = \underset{\|\vec{w}_1\|=1}{\operatorname{argmax}} \sum_{i=1}^{m} \lambda_i c_{1,i}^2.$$

Áp dụng điều kiện $\sum_{i=1}^{m} c_{1,i}^2 = 1$, và $\lambda_1 = \max\{\lambda_i\}$, ta có:

$$\sum_{i=1}^{m} \lambda_i c_{1,i}^2 \leq \lambda_1 . \sum_{i=1}^{m} c_{1,i}^2 = \lambda_1.$$

Suy ra:

$$\max \sum_{i=1}^{m} \lambda_i c_{1,i}^2 = \lambda_1.$$

Đẳng thức xảy ra khi và chỉ khi $c_{1,1} = 1$ và $c_{1,i} = 0 \ \forall i \neq 1$, hay ta có $\vec{w}_1 \equiv \vec{v}_1$.

Vậy ta đã tìm được thành phần có *độ phân tán* lớn nhất. Để đi tìm thành phân có *độ phân tán* lớn tiếp theo, ta giải lần lượt các bài toán đã nêu ở trên:

$$\vec{w}_i = \underset{\substack{\|\vec{w}_i\|=1 \\ \vec{w}_i \cdot \vec{w}_j = 0, \forall j < i}}{\operatorname{argmax}} \left(w_i^T A w_i \right).$$

Từ điều kiện $\vec{w}_i \cdot \vec{w}_j = 0 \ \forall j < i$, ta thấy rằng $\forall i \geq 2$: $\left[\vec{w}_i\right]_V = \begin{bmatrix} 0 & \dots & 0 & c_{i,i} & c_{i,i+1} & \dots & c_{i,m} \end{bmatrix}$.

Giải lần lượt các bài toán này một cách tương tự như chiều thứ nhất, ta tìm ra được *m chiều quan trọng* của bộ dữ liệu, cũng chính là *m* vector riêng ứng với *m* trị riêng của ma trận *A*. Ở đây giá trị riêng càng lớn thì chiều của vector riêng đó càng quan trọng.

4.3 Kết luận

Như vậy, *bài toán PCA* thực chất là *bài toán tìm các vector riêng của ma trận* $A = \hat{X}\hat{X}^T$. Ma trận *A* còn được gọi là *ma trận hiệp phương sai* của tập dữ liệu X.

5 Giới thiệu thuật toán tìm trị riêng

Như đã phân tích trong các phần trên, bài toán của chúng ta được đưa về bài toán tìm các *trị riêng* và các *vector riêng* của *ma trận hiệp phương sai* $\hat{X}\hat{X}^T$. Nhưng việc tìm trị riêng chính là giải phương trình $\det(A - \lambda I_n) = 0$ ($A \in \mathbb{R}^{n \times n}$). Phương trình này sẽ đưa về phương trình bậc *n*. Nhưng ta đã biết việc giải phương trình bậc cao sẽ rất khó khăn và với $n \geq 5$ thì ta không có công thức nghiệm tổng quát.

Tuy nhiên *ma trận hiệp phương sai* của chúng ta là ma trận *đối xứng* và *nửa xác định dương*, cho nên chúng ta sẽ có một vài thuật toán có thể giải được nghiệm của định thức này.

5.1 Phương pháp Power

Đây là phương pháp tìm trị riêng và vector riêng của một ma trận *nửa xác định dương* $A \in \mathbb{R}^{n \times n}$

Thuật toán: Trước tiên, ta sẽ tìm vector riêng ứng với trị riêng lớn nhất λ_1.

- Bước 1: Chọn bất kì một vector $\vec{q_0} \in \mathbb{R}^{n\times n} : \|\vec{q_0}\| = 1$.
- Bước 2: Với $k = 1, 2, 3, \ldots$, ta tính $q_k = Aq_{k-1}$.
- Bước 3: Chuẩn hóa $q_k = \dfrac{q_k}{\|q_k\|}$.
- Bước 4: Kiểm tra nếu $\|\vec{q_k} - \vec{q}_{k-1}\|$ đủ nhỏ (các vector $\vec{q}$ hội tụ) thì ta dừng lại. Nếu không, tăng k lên 1 rồi quay lại Bước 2.
- Bước 5: $\vec{q_k}$ chính là vector riêng ứng với trị riêng lớn nhất λ_1 suy ra $\lambda_1 = q_k^T A q_k$.

Ta chứng minh được rằng $\vec{q}$ sẽ hội tụ khá nhanh.

Chứng minh. Gọi S là hệ cơ sở tạo bởi các vector riêng $\vec{x}$ của ma trận A trong $\mathbb{R}^n$. Khi đó vector $\vec{q_0}$ có thể được biểu diễn dạng:

$$\vec{q_0} = a_1\vec{x_1} + a_2\vec{x_2} + a_3\vec{x_3} \ldots + a_n\vec{x_n},$$

với $a_1, a_2, a_3, \ldots$ là tọa độ của $\vec{q_0}$ trong S.

Sau k lần nhân như trên, ta có:

$$\begin{aligned} A^k q_0 &= A^k \left(a_1x_1 + a_2x_2 + a_3x_3 \ldots + a_nx_n\right) \\ &= a_1A^kx_1 + a_2A^kx_2 + a_3A^kx_3 + \ldots + a_nA^kx_n \\ &= a_1\lambda_1^kx_1 + a_2\lambda_2^kx_2 + a_3\lambda_3^kx_3 + \ldots + a_n\lambda_n^kx_n \\ &= a_1\lambda_1^k\left(x_1 + \sum_{j=2} n\frac{a_j}{a_1}\left(\frac{\lambda_j}{\lambda_1}\right)^k x_j\right). \end{aligned}$$

Nếu $|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$ thì $\left|\dfrac{\lambda_j}{\lambda_1}\right| < 1$ với $j > 1$.

Suy ra $\left(\dfrac{\lambda_j}{\lambda_1}\right)^k$ hội tụ về 0 khi k tiến đến ∞.

Vậy A^kq_0 hội tụ về $a_1\lambda_1^kx_1$. Nhưng do ta đã thực hiện chuẩn hóa liên tục Aq^{k-1} tại bước thứ k, nên ta suy ra được dãy A^kq_0 sẽ hội tụ về x_1. □

Vậy ta tính được trị riêng lớn nhất λ_1 và vector riêng ứng với nó, ta sẽ tìm được các trị riêng và vector riêng còn lại theo định lý sau.

Định lý 5.1. *Nếu ma trận nửa xác định dương A có các trị riêng $\lambda_1 > \lambda_2 > \lambda_3 > \ldots > \lambda_n > 0$ và các vector riêng tương ứng $\vec{v_1}, \vec{v_2}, \vec{v_3}, \ldots, \vec{v_n}$ hơn nữa các vector riêng này tạo thành 1 hệ trực chuẩn, thì ma trận:*

$$B = A - \lambda_1 v_1 v_1^T$$

có các trị riêng $\lambda_2 > \lambda_3 > \ldots > \lambda_n > 0$ và các vector riêng tương ứng là $v_2, v_3, \ldots, v_n$.

Chứng minh. Ta có:
• Với $i = 1$:

$$Bv_1 = \left(A - \lambda_1 v_1 v_1^T\right) v_1 = Av_1 - \lambda_1 v_1 = 0.$$

Suy ra: λ_1 không là trị riêng của B.
• Với $i > 1$:

$$\begin{aligned} Bv_i &= \left(A - \lambda_1 v_1 v_1^T\right) v_i \\ &= Av_i - \lambda_1 v_1 \left(v_1^T v_i\right) \\ &= Av_i \text{ (vì } \vec{v}_1 \text{ và } \vec{v}_i \text{ trực giao)} \\ &= \lambda_i v_i. \end{aligned}$$

Vậy khi $i > 1$ thì λ_i là trị riêng của B. Vậy ta tìm được trị riêng lớn nhất λ_2 và vector riêng $\vec{v}_2$ tương ứng của B. Tiếp tục làm như vậy, ta sẽ tìm được n trị riêng và vector riêng của ma trận A lúc đầu. □

5.2 Nhược điểm của thuật toán Power

Thuật toán Power cho ta một cách xấp xỉ trị riêng, vector riêng khá tốt và đơn giản, song, phương pháp này cũng có những hạn chế khiến nó không thường được sử dụng trong thực tế cho việc này. Trong số đó có thể kể đến như:

- Trong trường hợp không may mắn, có xác suất (rất nhỏ) việc vector $\vec{q}$ khởi tạo vuông góc với vector riêng ứng với trị riêng lớn nhất. Khi đó, thuật toán Power sẽ cho ra kết quả là vector riêng ứng với trị riêng lớn thứ 2 trong số các vector riêng của ma trận A.
- Nếu một ma trận có nhiều vector riêng ứng với một trị riêng thì thuật toán Power sẽ chỉ cho ra một vector riêng ứng với trị riêng đó.
- Nếu các trị riêng của ma trận A có giá trị xấp xỉ bằng nhau, tốc độ hội tụ của thuật toán Power sẽ khá chậm.

Tuy nhiên, thuật toán Power vẫn có những ứng dụng thực tế trong một số trường hợp cụ thể ví dụ như trong Google Pagerank để sắp xếp các website theo độ phổ biến giảm dần của chúng.

5.3 Tổng kết

Như vậy, phương pháp Power tuy là một thuật toán xấp xỉ đơn giản nhưng nó lại có những trường hợp riêng mà thuật toán hoạt động không hiệu quả. Chính vì vậy, người ta thường dùng những thuật toán khác như thuật toán **QR** hay **SVD** để thay thế.

6 Cài đặt

6.1 Các bước thực hiện Phương pháp PCA

Sau khi phân tích về PCA, chúng ta rút ra được các bước thực hiện chính như sau:

1. Tìm vector trung bình của toàn bộ dữ liệu:

$$\forall i \leq m : \overline{x}_i = \frac{1}{n} \sum_{j=1}^{n} x_j.$$

2. Chuẩn hóa lại toàn bộ dữ liệu:

$$\forall j \leq n : \text{col}_j\left(\hat{X}\right) = \text{col}_j\left(X\right) - \overline{x}.$$

3. Tính *ma trận hiệp phương sai*:

$$\text{Cov}(X) = \frac{1}{n}\hat{X}\hat{X}^T.$$

Nhưng thực tế khi lập trình, ta có thể không cần chia n (do các ma trận tương đương nhau nên không ảnh hưởng kết qua sau cùng của ta).

4. Tính các trị riêng và vector riêng có *norm* bằng 1 của ma trận $\text{Cov}(X)$, sắp xếp chúng theo thứ tự giảm dần của trị riêng.

5. Chọn k vector để giữ lại ứng với k trị riêng lớn nhất của $\text{Cov}(X)$ để tạo thành hệ cơ sở S có k chiều. Hệ cơ sở S này tạo thành không gian con mà ta cần chiếu bộ dữ liệu của mình lên.

6. Thực hiện phép chiếu $\hat{X}$ vào hệ cơ sở S, ta được dữ liệu mới chính là tọa độ của các điểm của $\hat{X}$ trong S:

$$Z = S\hat{X}.$$

6.2 Source code trên ngôn ngữ lập trình Python

Để dễ dàng hơn khi lập trình, ta quy ước ngược lại với phân tích ở trên một tí là ma trận X nằm trong $\mathbb{R}^{n\times m}$ với n hàng là các bộ dữ liệu, m cột là các đặc tính của dữ liệu.

6.2.1 Code lại thủ công toàn bộ PCA

```python
def PCA(data, k):
    import numpy as np
    X = np.matrix(data, dtype = float)
    n, m = X.shape  # n là số dữ liệu, m là số đặc tính
    # Chuẩn hóa theo vector trung bình
    X -= X.mean(axis=0)
    # Tính ma trận hiệp phương sai
    Cov = np.cov(X, rowvar=False)
    # tính trị riêng và vector riêng
    eVector = np.linalg.eigh(Cov)[1]
    # np.linalg.eigh(Cov)[0] trả ra các trị riêng sắp theo thứ tự tăng dần
    # np.linalg.eigh(Cov)[1] trả ra ma trận các vector riêng ở dạng cột
    # theo thứ tự tương ứng như trên
    eVector = (eVector.T)[::-1]
    # đổi thành ma trận vector hàng và sắp lại
    # theo thứ tự trị riêng giảm dần
    S = np.matrix(eVector[:k]) # tạo hệ cơ sở mới với k chiều đầu tiên
    return (S*(X.T)).A
    # trả ra kết quả ở dạng np.ndarray để dễ xữ lí hơn thay np.matrix
```

6.2.2 Sử dụng thư viện sklearn

```python
import numpy as np
import scipy
from sklearn.decomposition import PCA

X = # Bộ dữ liệu
k = # Số chiều muốn giữ lại
pca = PCA(n_components = k)
pca.fit(X.T)
res = pca.components_.T
```

7 Áp dụng mô hình

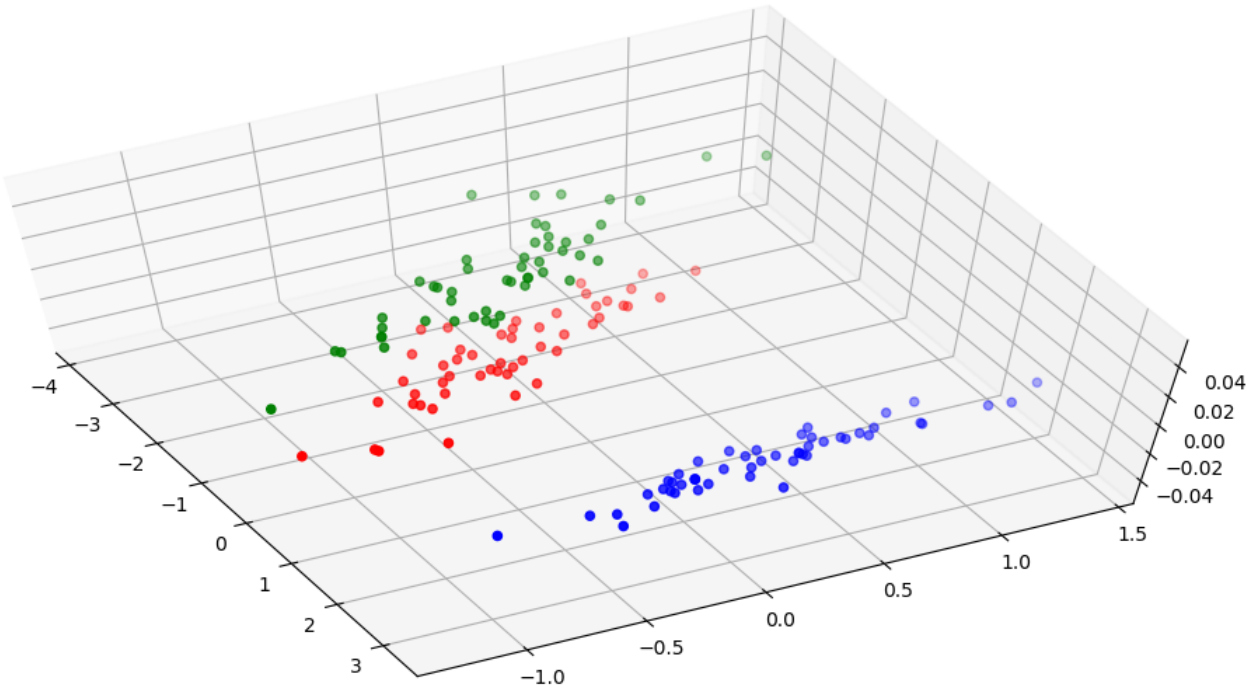
Trong thực tế, dữ liệu thường sẽ có rất nhiều tính trạng và vector biểu diễn chúng sẽ có số chiều tương ứng. Tuy vậy, con người chỉ có thể cảm nhận được vị trí và hình ảnh trong không gian nhỏ hơn hay bằng 3 chiều. Chính vì thế, nếu ta có thể áp dụng PCA để giảm chiều dữ liệu xuống còn 3 hay thấp hơn, ta có thể biểu diễn những điểm dữ liệu này trên đồ thị. Bây giờ, ta sẽ thực hiện PCA lên hai bộ dữ liệu mẫu được tải free trên mạng.

7.1 Hoa Iris

Bộ dữ liệu này gồm dữ liệu về độ dài và độ rông của đài hoa và cánh hoa của 150 bông hoa Iris thuộc 3 chủng loại Iris Setosa, Iris Versicolour, Iris Virginica. Ứng với mỗi chủng loại hoa sẽ có 50 bông hoa được khảo sát.

1	sepal length	sepal width	petal length	petal width	class
2	5,1	3,5	1,4	0,2	Iris-setosa
3	4,9	3	1,4	0,2	Iris-setosa
4	4,7	3,2	1,3	0,2	Iris-setosa
5	4,6	3,1	1,5	0,2	Iris-setosa
6	5	3,6	1,4	0,2	Iris-setosa

Kết quả sau khi PCA là:

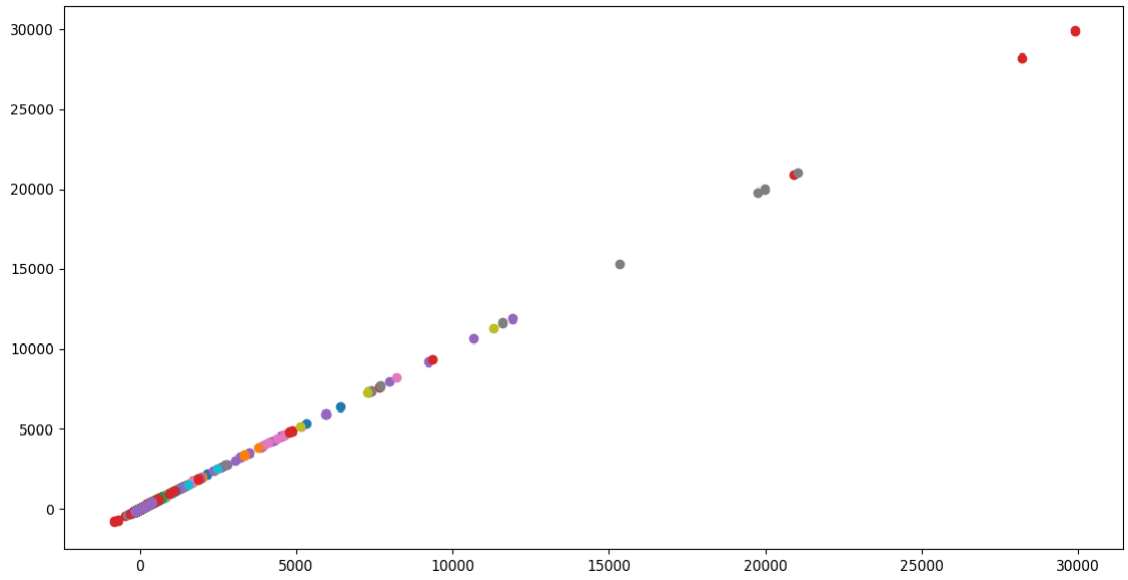


7.2 Cơ sở dữ liệu dinh dưỡng quốc gia USDA

Bộ dữ liệu này gồm danh sách các loại thức ăn và những thông số dinh dưỡng của nó như: energy kcal, protein, fat, . . .

ID	FoodGroup	ShortDescrip	Descrip	Energy_kcal	Protein_g	Fat_g	Carb_g	Sugar_g
01001	Dairy and Egg Products	BUTTER,WITH SALT	Butter, salted	717	0,85	81,11	0,06	0,06
01002	Dairy and Egg Products	BUTTER,WHIPPED,WITH SALT	Butter, whipped, with salt	717	0,85	81,11	0,06	0,06
01003	Dairy and Egg Products	BUTTER OIL,ANHYDROUS	Butter oil, anhydrous	876	0,28	99,48	0	0
01004	Dairy and Egg Products	CHEESE,BLUE	Cheese, blue	353	21,4	28,74	2,34	0,5
01005	Dairy and Egg Products	CHEESE,BRICK	Cheese, brick	371	23,24	29,68	2,79	0,51

Kết quả sau khi PCA là:



7.3 EigenFace

EigenFaces là một trong những phương pháp phổ biến trong bài toán nhận diện khuôn mặt. Ý tưởng của phương pháp này là từ một bộ ảnh khuôn mặt người có kích thước lớn và số lượng nhiều, ta sẽ tìm một không gian nhỏ hơn để biểu diễn các khuôn mặt này.

EigenFace chính là PCA: các Eigen Faces (*khuôn mặt riêng*) chính là các vector riêng ứng với các trị riêng lớn nhất. Ta có thể hiểu là các *khuôn mặt riêng* sẽ mang phần nào thông tin của các khuôn mặt ban đầu. Vì thế, khuôn mặt ban đầu có thể xấp xỉ bằng tổng có trọng số của các *khuôn mặt riêng* này.

Bây giờ chúng ta tìm EigenFaces trên bộ dữ liệu Yale face database. Bộ dữ liệu này gồm hình trắng đen khuôn mặt của 15 người khác nhau, mỗi người có 11 tấm ảnh được chụp ở điều kiện ánh sáng, cảm xúc khác nhau, gồm: *centerlight*, *glasses*, *happy*, *leftlight*, *noglasses*, *normal*, *rightlight*, *sad*, *sleepy*, *surprised* và *wink*. Các ảnh trong bộ dữ liệu này đã được chuẩn hóa cùng kích thước 116×98 và khuôn mặt nằm gọn trong ảnh.

Source code:

```python
import numpy as np
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
import imageio

# read data
states = ["centerlight", "glasses", "happy", "leftlight", "noglasses",
          "normal", "rightlight", "sad", "sleepy", "surprised", "wink"]
h = 116
w = 98
D = h * w
n = len(states) * 15
X = np.zeros((n, D))

cnt = 0
for person_id in range(1, 16):
    for state in states:
        path = "Yale/subject" + str(person_id).zfill(2) + '.' + state + ".pgm"
        X[cnt] = imageio.imread(path).reshape(D)
        cnt += 1

# Do PCA
k = int(input())
pca = PCA(n_components=k)
pca.fit(X)

# projection matrix
X_proj = pca.components_

for i in range(X_proj.shape[0]):
    plt.axis('off')
    plt.imshow(X_proj[i].reshape(h, w), interpolation="nearest")
    plt.gray()
    path = "Eigen Faces/" + str(i).zfill(2) + ".png"
    plt.savefig(path, bbox_inches='tight', pad_inches=0)

# See reconstruction of first 6 persons
for person_id in range(1, 16):
    for state in states:
        path = "Yale/subject" + str(person_id).zfill(2) + '.' + state + ".pgm"
        img = imageio.imread(path)
        # subtract mean
        x = img.reshape(D, 1) - pca.mean_.reshape(D, 1)
        # encode
        z = X_proj.dot(x)
        # decode
        x_tilde = X_proj.T.dot(z) + pca.mean_.reshape(D, 1)
        img_tilde = x_tilde.reshape(h, w)
        plt.axis('off')
        plt.imshow(img_tilde, interpolation='nearest')
        plt.gray()
        path = "reconstruction/" + str(person_id).zfill(2) + '.' + state + ".png"
        plt.savefig(path, bbox_inches='tight', pad_inches=0)
```

Dưới đây là ảnh của 9 Eigen Faces đầu tiên:

Để đánh giá độ hiệu quả của EigenFace, ta biểu diễn lại một vài ảnh để so sánh với ảnh ban đầu. Bộ dữ liệu gồm 165 ảnh này được biểu diễn lại bằng k *khuôn mặt riêng*, với k lần lượt bằng 75, 100, và 130

- Ảnh gốc:

- Tái tạo với $k = 75$:

- Tái tạo với $k = 100$:

- Tái tạo với $k = 130$:

8 Kết luận đánh giá

PCA là một cách tiếp cận phổ biến cho bài toán giảm chiều dữ liệu vì có nhiều ưu điểm như:

- PCA là một cách tiếp cận trực quan và dễ hiểu.
- PCA giúp giảm chiều của dữ liệu đi một cách đáng kể, từ đó làm cho việc xử lý và lưu trữ trở nên nhanh chóng và dễ dàng hơn. Vì vậy, trong thực tế, người ta thường thực hiện PCA kết hợp với những thuật toán khác nhằm tăng hiệu quả của thuật toán đó lên.

Tuy vậy, phương pháp này vẫn có những nhược điểm của nó:

- Độ phức tạp tính toán của PCA thường rơi vào khoảng $O(\min(n^3, m^3))$. Vì vậy, trong nhiều trường hợp, việc áp dụng PCA sẽ không hiệu quả.
- Nếu *độ phân tán* trên các chiều dữ liệu là xấp xỉ nhau, việc PCA sẽ làm cho lượng thông tin bị mất đi là khá lớn gây ảnh hưởng đến việc xử lý về sau.

9 Hướng nghiên cứu trong tương lai

Trong tương lai, nhóm sẽ đi sâu vào những kĩ thuật thực hiện như những thuật toán tìm trị riêng tối ưu hơn, như phương pháp QR factorization hay SVM factorization. Bên cạnh đó, các hướng mở rộng của PCA như Sparse PCA, Kernel PCA hay Robust PCA cũng là những hướng tìm hiểu thú vị.

Tài liệu

[1] *Tài liệu Trại hè PiMA 2018*.

[2] *Wikipedia*.

[3] *Yale face database*. http://vismod.media.mit.edu/vismod/classes/mas622-00/datasets/, 2016.

[4] data.world. *USDA National Nutrient Database data*. 2016.

[5] R.A. Fisher. *Iris Data Set*. https://archive.ics.uci.edu/ml/datasets/iris, 1936.

[6] Jeff Jauregui. *Principal component analysis with linear algebra*. 2012.

[7] Vũ Hữu Tiệp. *Machine Learning cơ bản*.