淺度機器學習
PCA、SVD 及其在影像處理的應用

汪群超

March 26, 2023

數位影像是屬於大型的資料，一張 1024 × 1024 的灰階照片大小等於 1Mbytes。當影像更大、色彩更豐富、數量更多時，不管在儲存、傳輸或是特徵抽取 (feature extraction) 上都造成困擾。基於影像資料中，鄰進像素的相關性頗高（如圖 1），表示這類的資料本身即存在著一些「多餘 (redundant) 資料」，利用主成分分析的原理可以有效的去除這些多餘的資料，讓資料量變小，在影像處理上則稱為「影像壓縮」。經過壓縮處理過的影像適合保存、傳輸等用途，但被刪除的資料畢竟仍是影像的一部份，在影像復原時會有某種程度上的損失，不過相較於影像大小的縮減，在某些應用上仍是值得的。本單元將以實際的資料展現主成分分析在資料壓縮與上的功能，雖然有時候效果不是很好，卻可以當作其他方法的前置作業處理 (preprocessing)。

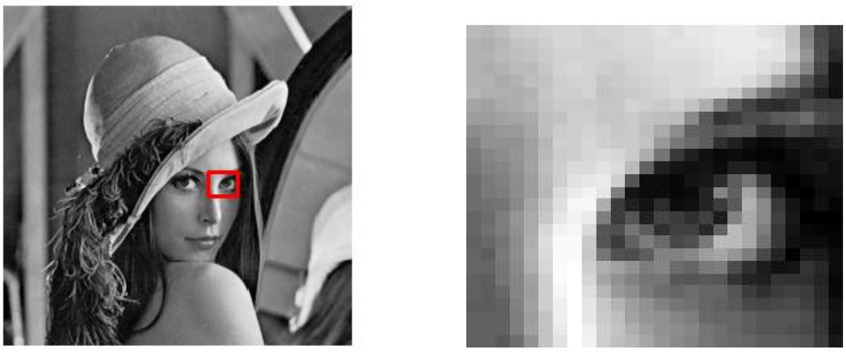

圖 1: 影像資料中鄰近像素間的高相關性：右圖為左圖 Lena 左眼部位的放大圖，可以清楚的看到每個像素及其灰度。

本章將學到關於程式設計：

〈本章關於 Python 的指令與語法〉

套件與指令：

mayplotlib: `image.imread, imshow`

sklearn.datasets: `fetch_openml`

numpy.linalg: `svd`

1 背景介紹

1.1 主成分分析

主成分分析的原理係將原變數向量 $\mathbf{x}$ （含 p 個變數），正交投射 (Orthogonal projection) 到維度較低 ($q < p$) 的子空間（Subspace）V，成為 $\mathbf{x}_q$，如圖 2 的低維度意象圖所示，其數學的表示法寫成

$$\mathbf{x}_q = P\mathbf{x} \tag{1}$$

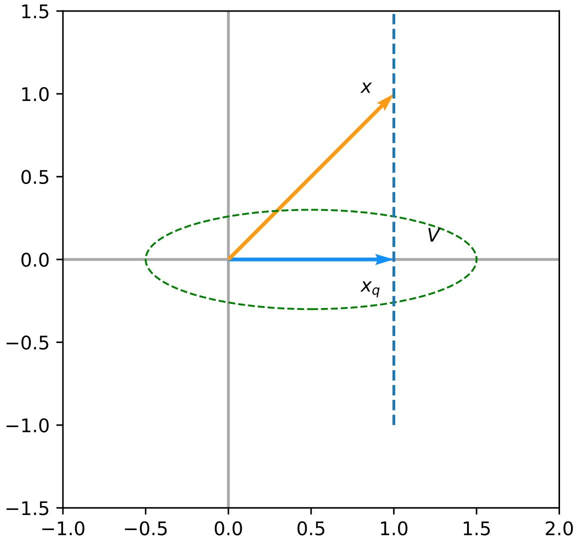

圖 2: 主成分分析的正交投射原理

其中 P 即所謂的投射矩陣，在此也稱為正交投射矩陣 (Orthogonal Projection Matrix) 。主成分分析為滿足其「主成分」的目的，定義了投射的目標區（子空間 V）及 P 的選擇。V 的定義及 P 的選擇如下：

$$\begin{aligned} V &= span\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_q\} \\ P &= V_q V_q^T \end{aligned} \tag{2}$$

其中 $V_q = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_q]$，$\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_q$ 為變數向量 $\mathbf{x}$ 的共變異矩陣 Σ_X 的前 q 個特徵向量（經標準化後）即

$$\Sigma_X \mathbf{v}_k = \lambda_k \mathbf{v}_k, \qquad 1 \le k \le q, \lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_q \tag{3}$$

其中 $p \times q$ 的矩陣 V_q 也稱為 Orthonormal matrix，因其滿足 $V_q^T V_q = I_q$。「主成分分析」將資料變數從 $\mathbf{x}$ 投射（轉置）到 $\mathbf{x}_q$ 的主要目的有二：其一，從空間幾何的角度來看，子空間的座標軸的選擇係依據資料成分（能量或變異）的分佈，資料呈現在其間的分佈與原座標軸所呈現出的分佈不同，在某些應用上，如群組分析、變數的選擇，可以得到好處。經轉換後的資料在子空間 V 的座標為

$$\mathbf{z}_q = V_q^T \mathbf{x} \tag{4}$$

z_q 為 $q \times 1$ 的向量，當 $q = 1, 2$ 時，常可從資料分佈圖中，發現一些潛藏的資訊。其二：資料量變小。原變數 $\mathbf{x}$ 與新變數 $\mathbf{x}_q$ 雖同為 $p \times 1$ 的向量，但 $\mathbf{x}_q$ 所在的空間較小，在資料的儲存上通常以下列方式來表達

$$\mathbf{x}_q = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_q \mathbf{v}_q = V_q \boldsymbol{\alpha} \tag{5}$$

其中 $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2 \ \cdots \alpha_q]^T$。每個樣本都是基底向量的線性組合，儲存上僅需為每一個樣本保留其組合係數，及一組共用的基底。當樣本數愈大，節省的空間也相對可觀。式 (5) 的 $\boldsymbol{\alpha}$ 其實就是式 (4) 的 $\mathbf{z}_q$。

當然儲存空間變小並非全無代價，根據主成分的原理，$\mathbf{x}_q$ 僅保留原變數一定比例的資訊（能量），其餘部分（假設為誤差）完全捨棄，這項誤差表示為

$$\mathbf{e} = \mathbf{x} - \mathbf{x}_q = (I - P)\mathbf{x} \in V^{\perp}, \text{where } V \oplus V^{\perp} = R^p$$

很明顯的，q 的選擇決定了誤差的大小，而且並無一定的標準，完全視應用的情況而定。譬如影像資料的壓縮若著眼於視覺感官的反應，若干資料的損失往往是視覺上所能忍受的，此時 q 的選擇常以目視來決定。主成分分析在實際資料的計算上與著名的 SVD 矩陣分解有密切的關係，先就 SVD 說明如下，再來分析之間的關係。

1.2 SVD: Singular Value Decomposition

假設 A 為一個 $m \times n$ 的矩陣，其 SVD 表示法為

$$A = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^T \tag{6}$$

其中 $r = Rank(A) \leq min(m,n)$，$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ 稱為 singular values，$\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_r$ 為 $m \times 1$ 的 orthonormal vectors，也稱為 left singular vectors，而 right singular vectors 指的是 $n \times 1$ 的 orthonormal vectors $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_r$。

這些 singular values 及 singular vectors 來自以下的 eigenvalue-eigenvector 分析：

$$\begin{aligned} AA^T\mathbf{u}_k &= \lambda_k\mathbf{u}_k = \sigma_k^2\mathbf{u}_k, \quad k = 1, 2, \cdots, m \\ A^TA\mathbf{v}_k &= \sigma_k^2\mathbf{v}_k, \quad k = 1, 2, \cdots, n \end{aligned} \tag{7}$$

式 (7) 說明 singular value σ_k 的平方是 AA^T 的特徵值，而 left singular vector $\mathbf{u}_k$ 為其相對的特徵向量。Right singular vectors $\mathbf{v}_k$ 為 A^TA 的特徵向量，相對應的特徵值也是 σ_k 的平方。式 (6) 的 SVD 表示法可以矩陣的方式改寫為

$$A = U\Sigma V^T \tag{8}$$

其中

$$\begin{aligned} U &= \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \end{bmatrix} \\ \Sigma &= \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \\ V &= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \end{aligned}$$

U, V 為 $m \times m$ 及 $n \times n$ 的 orthonormal matrix，即 $U^TU = UU^T = I_m, V^TV = VV^T = I_n$。$\Sigma$ 為一 $m \times n$ 的對角矩陣，對角的位置自 $r + 1$ 之後皆為 0。

1.3 主成分分析與 SVD

前面對於主成分分析的探討僅止於變數，現考量其實際的樣本資料。假設原始變數表示為 $p \times 1$ 的向量 $\mathbf{x}$，現有 N 個樣本，組合成一個 $p \times N$ 的資料矩陣 X，

即

$$X = [\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_N] \tag{9}$$

對變數 $\mathbf{x}$ 的共變異矩陣 Σ_X 的估計，寫成（假設 $\bar{\mathbf{x}} = \mathbf{0}$）

$$S_x = \frac{1}{N-1} X X^T \tag{10}$$

假設資料矩陣 X 的 SVD 表示為

$$X = U \Sigma V^T \tag{11}$$

根據式 (7)，上式 $p \times p$ 矩陣 U 中的 left singular vectors $\mathbf{u}_k$ 即為式 (10) 中共變異矩陣的特徵向量，當式 (11) 改寫為

$$U^T X = \Sigma V^T = Z = [\mathbf{z}_1\ \mathbf{z}_2\ \cdots \mathbf{z}_N] \tag{12}$$

式 (12) 的資料矩陣 Z 就是原始資料經過座標軸轉換過的新座標。當只取 $q (q < p)$ 個主成分時，式 (12) 寫成 $q \times N$ 矩陣

$$U_q^T X = \Sigma_q V_q^T = \begin{bmatrix} \sigma_1 \mathbf{v}_1^T \\ \sigma_2 \mathbf{v}_2^T \\ \vdots \\ \sigma_q \mathbf{v}_q^T \end{bmatrix} = Z_q \tag{13}$$

式 (13) 其實就是式 (4) 的樣本值表示法，利用 SVD 的方式可以看得更清楚，計算上也更方便。前面說過，當 $q < p$ 時，原始資料並無法完全復原，只能得到近似值 X_q ，以資料矩陣 X 的 SVD 的表示法[1]

$$X_q = U_q U_q^T X = U_q \Sigma_q V_q^T = \sum_{k=1}^{q} \sigma_k \mathbf{u}_k \mathbf{v}_k^T = U_q Z_q \tag{14}$$

稱為原始資料 X 的 *Rank q* approximation。或是針對每個樣本的近似值

[1] $X = U\Sigma V^T = U_q \Sigma_q V_q^T + U_{p-q} \Sigma_{p-q} V_{p-q}^T$, therefore, $U_q^T X = U_q^T U \Sigma V^T = U_q^T (U_q \Sigma_q V_q^T + U_{p-q} \Sigma_{p-q} V_{p-q}^T) = U_q^T U_q \Sigma_q V_q^T = \Sigma_q V_q^T$. Thus $U_q U_q^T X = U_q \Sigma V_q^T$

$$
\begin{aligned}
\tilde{\mathbf{x}}_k &= \sigma_1 \mathbf{v}_1(1)\mathbf{u}_1 + \sigma_2 \mathbf{v}_2(1)\mathbf{u}_2 + \cdots + \sigma_q \mathbf{v}_q(1)\mathbf{u}_q \\
&= U_q \tilde{\mathbf{z}}_k \\
&= \tilde{\mathbf{z}}_k(1)\mathbf{u}_1 + \tilde{\mathbf{z}}_k(2)\mathbf{u}_2 + \cdots + \tilde{\mathbf{z}}_k(q)\mathbf{u}_q, \qquad 1 \le k \le N
\end{aligned} \tag{15}
$$

當 X 代表一張影像資料時，X_q 便是其近似版本，[2]或可解釋為經壓縮後還原的影像，其失真程度與 q 的選擇有關。這裡說到「壓縮」，代表儲存 X_q 所需的容量較 X 小，而「失真」表示捨棄了部分資料（即 $X = X_q + \sum_{k=q+1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^T$）。以下的範例可以幫助理解。

2 練習

範例 1. 高解析度影像照片的檔案大，通常必須經過壓縮處理以降低儲存空間，譬如著名的 JPEG 檔。壓縮的好處是方便傳送與保存，缺點是失真的影像品質，不過如能保持一定的清晰程度，仍可被接受。不同於 JPEG 的壓縮演算法，本文介紹 SVD 矩陣解構法，也可以利用 *rank q* approximation 去除影像「多餘」的資料，做為另一種壓縮方式。本範例採影像處理學界經常使用的 Lena 黑白圖片為例（圖 3），進行壓縮處理，看看壓縮的品質與壓縮比例表現如何。

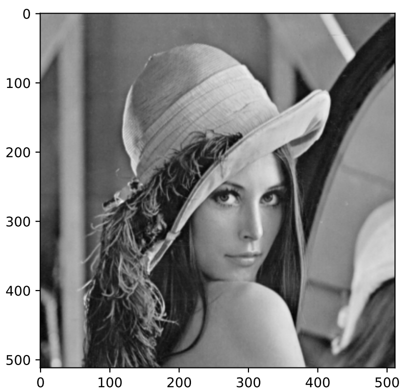

圖 3: 影像處理學界著名的 Lena 圖（512×512）

[2]當 X 的每一欄代表一張影像資料時，則第 k 欄（第 k 張影像）的近似版本為 $\tilde{\mathbf{x}}_k$。且從式 15 看得出，每張近似的影像皆為 left singular vectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_q$ 的線性組合，差別只是組合係數不同而已。當作為影像壓縮之用途時，每張影像僅需儲存這 q 個係數與一組共同的基底 $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_q\}$。

一張黑白影像資料等於一個矩陣，圖 3 可以視為一 512×512 的資料矩陣 X。根據式 (14) 的 *Rank* q approximation，取 X 矩陣的前 q 個「能量」最高的分項和（X_q）做為 X 的近似矩陣，也可視為某種壓縮圖。圖 4 展示三個不同壓縮比例的近似圖，由左至右分別取 $q = 128, 64, 32$，其壓縮倍數分別為 $2, 4, 8$ 倍。從圖可以看出當壓縮倍數為 2 時，影像目視的品質幾無差異，但儲存空間卻少了一半。壓縮倍數的計算是個約略值，以 $q = 128$ 為例，式 (14) 需要 128 個 512×1 的 $\mathbf{u}_k$ 向量與同等數量的 $\mathbf{v}_k$ 向量，加上 128 個 σ_k 值。即資料量從 512×512 減少為 $512 \times 256 + 128$，大約減少一半。

圖 4: Lena 原圖的近似（壓縮）圖，自左至右採 $q = 128, 64, 32$，壓縮倍數分別約為 $2, 4, 8$ 倍。

建立圖 4 的影像壓縮程式碼參考如下：

```python
import numpy as np
from numpy.linalg import svd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

imgfile = "Lenna.png" # 512x512x3
X = mpimg.imread(imgfile)
if len(X.shape) > 2:
    X = np.mean(X, axis=2) # convert RGB to grayscale

N, p = X.shape
U, E, VT = svd(X, full_matrices = False)
q = np.array([p/4, p/8, p/16]).astype('int')
fig, ax = plt.subplots(1, 3, figsize=(12, 4))
for i, r in enumerate(q):
    Xq = U[:, :r] @ np.diag(E[:r]) @ VT[:r, :]
    ax[i].imshow(Xq, cmap = 'gray')
    ax[i].set_title('Compression ratio: {}'.format(p/r/2))
    ax[i].set_xticks([])
    ax[i].set_yticks([])
```

```
plt.show()
```

有一個有趣的想法，如果將影像矩陣做切割、轉置、移動，重新組合成另一個矩陣（資料總量與內容不變），再對這個新矩陣做 SVD 的近似矩陣，最後還原成影像圖案，請問這個做法能得到更好的結果？還是更差？還是一樣？也就是矩陣內資料的調動，是否影響資料變異數的分布？或資料能量的配置？而這些改變對影像矩陣的影響是否不同？畢竟影像矩陣不同於一般資料矩陣，我們還是很重視最後呈現出來的視覺效果。

圖 5 來自將原來的影像矩陣做切割、轉置與重組，三張圖分別與圖 4 的壓縮比例差不多，但視覺品質卻有明顯的差距。做法如下：

1. 將原圖（512×512）自左至右，由上而下，切割出一張張 16×16 的小圖（patch），總共切出 $32 \times 32 = 1024$ 張小圖。
2. 將每張小圖轉換成（reshape）256×1 的向量，再由左而右組合成一 256×1024 的新矩陣。
3. 對新矩陣做 *Rank q* approximation。
4. 將近似矩陣的每一列重組回 16×16 的小圖，並依序組合回原圖的結構。

其實新矩陣與原矩陣的資料並沒有任何改變，只是重組而已，但經過 *Rank q* 近似後的效果卻明顯進步了（特別是壓縮 8 倍的圖）。原因是甚麼？值得想想。如此一來，切割成 8×8 或 32×32 的效果可以期待嗎？

圖 5: Lena 原圖重組後的近似（壓縮）圖，壓縮倍數約為 $2, 4, 8.2$ 倍。

接下來的範例將處理手寫數字的影像圖，包括數字影像的壓縮與生成，如圖 6 呈現的 50 張數字圖，每張大小為 28×28 灰階影像。這些手寫數字圖片來自 sklearn.dataset，可以預先下載，方便後續處理。下載方式如下：

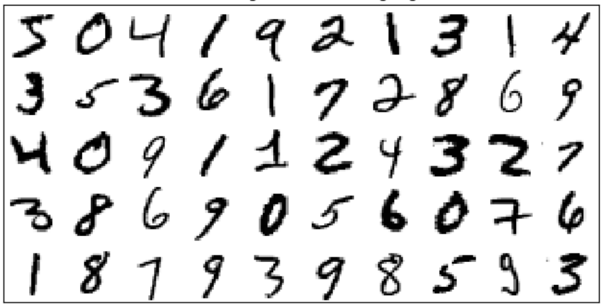

圖 6: 50 個數字樣本的影像

```
from sklearn.datasets import fetch_openml

# mnist = fetch_openml('mnist_784',parser='auto')
# X = mnist.data # data
# y = mnist.target # labels

X, y = fetch_openml('mnist_784', parser = 'auto', \
        return_X_y = True)

X = X.T # 配合本文對於圖像矩陣的定義 p x N
```

上述程式碼展示下載的兩種資料格式，都是以 pandas 的資料型態（X:Dataframe 代表影像矩陣，y:Series 為影像標籤）呈現，其中矩陣 X 內含 0 - 9 十個數字的圖形共 70000 張，每一列代表一張大小 28×28，的數字圖形（被轉換為 1×784 的向量），因此矩陣 X 大小為 70000×784。為配合本文對於圖像矩陣的定義，因此將 X 轉置為 784×70000，即 $p = 784, N = 70000$。

範例 2. 試著先寫一支小程式秀出矩陣 X 中的一張數字圖，如圖 7，熟悉影像資料的結構。成功之後，再將程式擴大為可以秀出多張數字圖合併的「大圖」(蒙太奇)，如圖 6，以便能同時觀察多張樣本影像的異同。

```
import matplotlib.pyplot as plt
import numpy as np

i = 0
img = X.iloc[:, i]
sz = np.sqrt(len(img)).astype('int')
plt.imshow(np.array(img).reshape(sz, sz), cmap='gray')
plt.show()
```

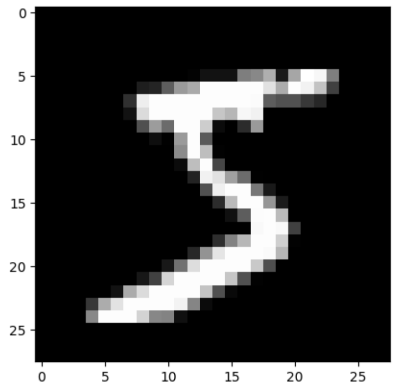

圖 7: 一張 28 × 28 數字 5 的樣本影像

圖 6 以蒙太奇方式呈現前 50 張圖，這個方式很適合將許多圖形集結在一起，方便觀賞、觀察與比較。因此可以寫成副程式，供重複使用。參考程式如下：

```
def montage(A, m, n):
    '''
    Create a montage matrix with mn images
    Inputs:
    A: original pxN image matrix with N images (p pixels), N > mn
    m, n: m rows & n columns, total mn images
    Output:
    M: montage matrix containing mn images
    '''

    sz = np.sqrt(A.shape[0]).astype('int') # image size sz x sz
    M = np.zeros((m*sz, n*sz)) # montage image

    for i in range(m) :
        for j in range(n) :
            M[i*sz: (i+1)*sz, j*sz:(j+1)*sz] = \
                A[:, i*n+j].reshape(sz, sz)

    return M
```

圖 6 繪製方式如：

```
plt.figure(figsize = (8, 12))
m, n = 5, 10  # m x n montage (total mn images)
M = montage(np.array(X), m, n)
plt.imshow(M, cmap = plt.cm.gray_r, interpolation = 'nearest')
plt.xticks([])
plt.yticks([])
```

```
plt.title('The␣Montage␣of␣handwriting␣digits')
plt.show()
```

範例 3. sklearn.dataset 的數字圖形資料以式 (9) 的方式儲存 70000 張影像在變數 X，其中數字 3 佔 7184 張，如圖 8 呈現 200 張。取所有的數字 3 做主成分分析，以 SVD 的 *Rank q* approximation（式 (14)）取前 q 個主成分，再進行影像還原，合成如圖 9 展示的 20 張樣本影像及其壓縮影像，其中原影像在第 1 列，第 2-16 列為合成 $q = 1, 3, 5, \cdots, 29$ 個主成分的還原影像。

圖 8 則是篩選出 200 個數字 3 影像，觀察手寫的差異，以便作為將來手寫辨認時的參考，譬如，數字 3 最容易與哪個數字混淆？繪圖的參考程式如下：

```
digit_to_show = '3'
idx = y[y==digit_to_show].index
Digit = X.iloc[:, idx]
plt.figure(figsize = (12, 12))
m, n = 10, 20  # A m x n montage (total mn images)
M = montage(np.array(Digit), m, n)
plt.imshow(M, cmap = 'gray', interpolation = 'nearest')
plt.xticks([])
plt.yticks([])
plt.show()
```

圖 8: 200 張數字 3 樣本的影像

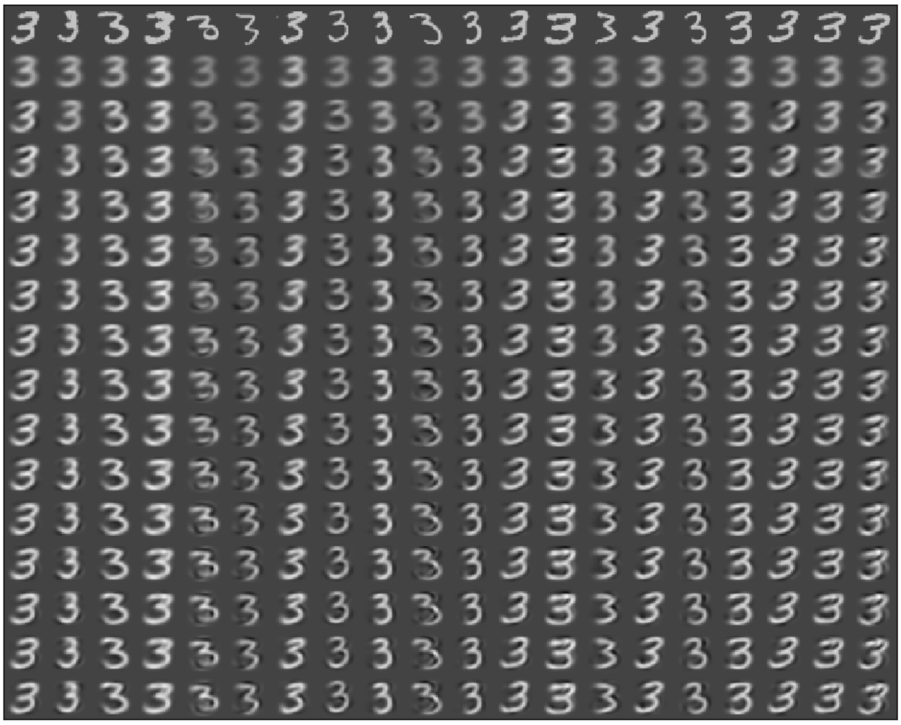

圖 9: 取 q 個主成分後的影像還原（壓縮）效果：第 1 列為原圖，第 2-16 列取 $q = 1, 3, 5, \cdots, 29$。

圖 9 將不同的 q 值所呈現的影像並列，方便研究者觀察「對於不同的影像，不同的 q 值在回復影像時的效果。」當 $q = 1$ 時，即僅取第一個成分，不論原圖的長相如何，幾乎都長的一樣。當加進來的成分愈多時（往下），影像才逐漸變化而接近原圖。換句話說，前面的成分是原影像的「主幹」，愈後面的成分呈現「修飾」效果。圖 9 幫助我們瞭解不同成分扮演的角色，並觀察從加進第幾個成分後，這 20 張圖才有明顯的不同，這或許可以幫助決定影像壓縮時 q 該取多少，才能既解省儲存空間，又能保持原圖的模樣。於是我們不免好奇每個逐漸加進來的成分究竟「長」甚麼樣子？如何能修飾第一主成分？圖 10 呈現這個想法，其中第 1 列為原始影像，第 2-7 列為第 1 至第 6 個主成分。

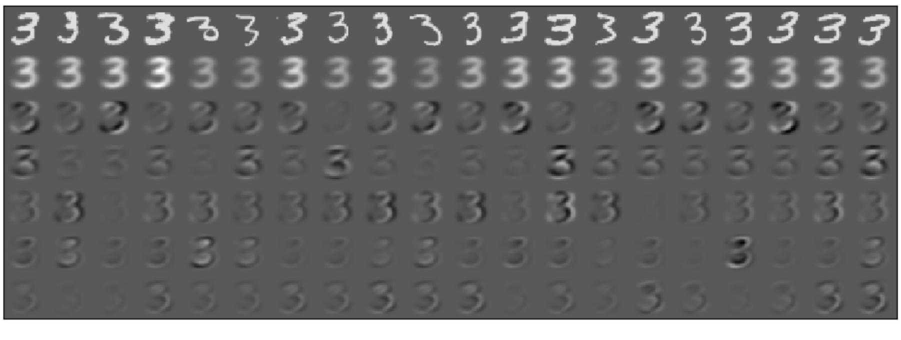

圖 10: 觀察個別主成分的影像：第 1 列為原圖，第 2-7 列取 $q = 1, 2, \cdots, 6$ 個主成分。

範例 4. 影像矩陣 X 的 SVD 分解的第 k 個主成分為 $\sigma_k \mathbf{u}_k \mathbf{v}_k^T$ 。事實上決定影像長相的關鍵是 left singular vector $\mathbf{u}_k$，式 (15) 說明了每個壓縮過的樣本 $\tilde{\mathbf{x}}_k$ 是 $\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_q$ 的線性組合。[a]這是否意味著，這些獨立成分也有「3」的味道？畫出來看看便知有沒有，如圖 11。這樣的試驗，其實也是研究工作中經常會做的驗證。有時只是證明自己的想法很天真，有時卻有令人意外的驚奇。不管結果如何，在觀念上都有深植的效果。

[a]為符合式 (15) 的說明，影像矩陣 X 必須轉置為 784×7184，即矩陣的表示法為 $p \times N$，其中 p, N 分別代表變數個數與樣本數。若維持原來的維度，則 right singular vector $\mathbf{v}_k$ 才是關鍵的成分。

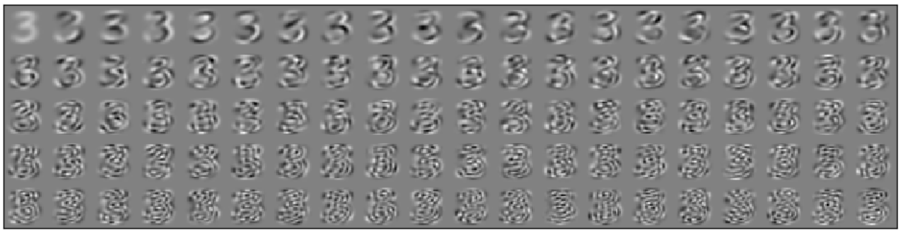

圖 11: $\mathbf{u}_k, k = 1, 2, \cdots, 100$ 的「長相」

圖 11 秀出前 100 個主成分代表的圖形，確實看出「3」的模樣，雖然愈往後的向量其影像愈模糊，不過長相卻略有不同。完整的成分應該有 256 個，也可以解釋為數字影像「3」的特徵，當然主成分分析的原理也說明，後面的特徵比較「不重要」。而組合的係數就是該樣本對於每個特徵的權重，藉以修飾初步同樣貌的影像。如此說來，我們是否可以做這樣的結論：這 256 個由 7184 張數字 3 得到的特徵向量，是否代表 3 的特徵，換句話說，「所有」的 3 都是從這些特徵向量組合而來，而前面的 q 個特徵向量最具代表性。再進一步大膽的猜測，是否能從最具代表性的 q 個特徵的任意組合，便能製造出一個人工的數字 3 影像？也或許不是任意組合，而是某些有規則的組合？再配合動手實驗看看。

範例 5. 接續上面的練習，式 (15) 的 $\tilde{\mathbf{z}}_k(1), \tilde{\mathbf{z}}_k(2), \cdots, \tilde{\mathbf{z}}_k(q)$ 的大小是否具特殊的分佈？這些係數是否也與特徵值 σ_k 呈現由大到小的分佈？是否呈現出特殊的分配模式？值得畫出來看看。如圖 12 列出其中的三組係數，左圖依序畫出其大小，右圖則是畫出相對的直方圖。這樣的分配比較稀疏（sparse），接近雙指數分配（double exponential）。

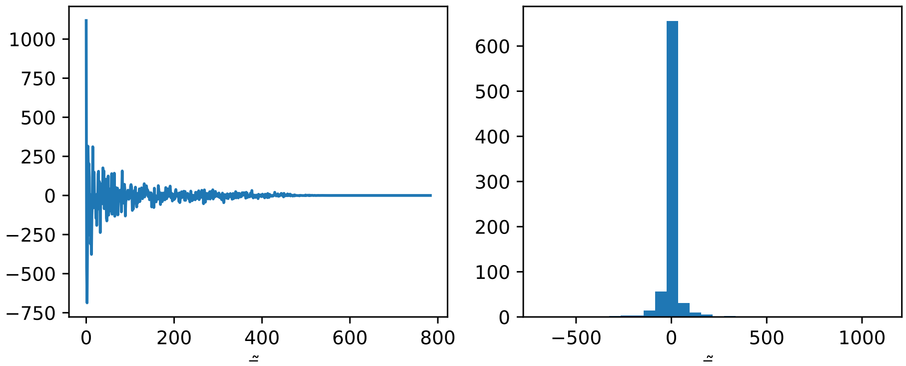

圖 12: 組合係數 $\tilde{\mathbf{z}}_k$ 的大小與分配

3 觀察與延伸

1. 式 (4) 同時代表座標軸的轉換與空間的縮減。如果不考慮空間的縮減，即子空間 V 等於原空間，或是表示為 $V = span\{\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_p\}$，則式 (4) 寫成

$$
\begin{aligned}
\mathbf{z} &= V^T\mathbf{x} \\
\mathbf{x} &= V\mathbf{z} = [V_q\ V_{p-q}] \begin{bmatrix} \mathbf{z}_q \\ \mathbf{z}_{p-q} \end{bmatrix} = V_q\mathbf{z}_q + V_{p-q}\mathbf{z}_{p-q} = \mathbf{x}_q + \mathbf{e}
\end{aligned}
$$

更明白的表示出 q 的選擇與誤差間的關係。

2. 從圖 8 觀察主成分代表的意義。五張「3」的影像除了看起來都像數字「3」的共同點外，各有其不同的特色，主成分分析中的每個成分是否都代表著組成「3」的成分？當 $q = 1, 2, \cdots, 10$ 將主成分一個個加進來時，回復出來的影像似乎也逐漸從相似 ($q = 1$：第二行的每一張圖) 到突顯出特色 ($q = 10$：最後一行的每一張圖)。

3. 式 (5) 或式 (15) 也提供一個觀點：每個樣本都來自同一組向量的線性組合，如範例 4 所示。如果我們任意的組合這些基底向量，是不是可以製造出新（人造）的樣本？換句話說，以本單元所使用的數字 3 的影像資料，從樣本分析出的基底向量，可否組成一個新的手寫「3」的人造影像？範例 5 中關於組合係數的大小與分佈，提供了些線索。以下這張圖是人工組合出來的，不與原樣本的任一張圖相同。

4. SVD 可以簡單解釋為將一個矩陣 A 分解為 r 個基底矩陣 $\mathbf{u}_k\mathbf{v}_k^T$ 相加，且每個基底矩陣配置一權重值 σ_k，由大排到小，代表該基底矩陣的重要性（貢獻度），也因此可以藉由去除貢獻低的基底矩陣，達到減量（資料壓縮）的目的。或者藉由取前幾項貢獻度高的基底矩陣作為特定目的，譬如

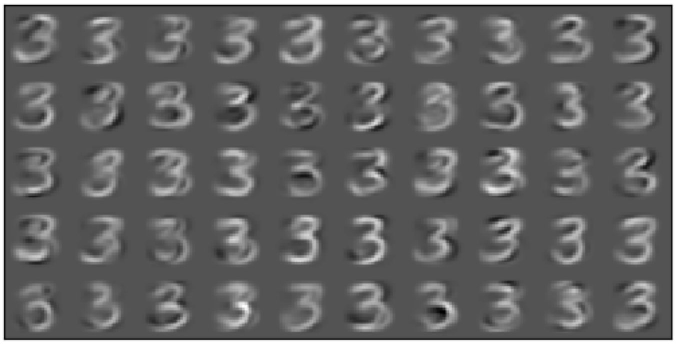

圖 13: 人造「3」

影像加密或影像辨識。類似的觀念非常多，像函數的泰勒展開式（Taylor series），便是將任意函數分解為無限項次的多項式函數。又如與本文相關的影像矩陣的二維傅立葉轉換（Two dimensional Discrete Fourier Transform, DFT），令 f 代表一 $M \times N$ 的影像矩陣，而 $f[m,n]$ 代表矩陣內第 (m,n) 位置的值，則 f 的二維傅立葉轉換寫成

$$F[k,l] = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f[m,n]e^{-i2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right)} \tag{16}$$

其中 $k = 0, 1, \cdots, M-1; l = 0, 1, \cdots, N-1$，而基底函數 $e^{-i2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right)} = \cos(2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right)) - i\sin(2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right))$ 分為實數與虛數兩個面向。這個以週期函數 $\cos(\cdot)$ 與 $\sin(\cdot)$ 組合成的新函數 $F[k,l]$ 便將影像從一般的實數空間帶到頻譜空間（spectrum），解構了影像內容的頻率變化。

傅立葉係數 $F[k,l]$ 的大小（amplitude）代表影像 f 在某個頻率的能量。一般而言，影像的「能量」只表現在非常小部分的頻率上，在其他頻譜的能量相對非常低，因此可以考慮去除，僅保留少部分頻率的能量，譬如圖 14 呈現 512×512 的 Lenna 影像前 10000 個最大的傅立葉係數 $F[k,l]$，其大小佔比約 43%。

去除傅立葉係數較小的頻率，並非直接去除，而是令 $F[k,l] = 0$，仍維持矩陣的規模（但變成稀疏矩陣 sparse matrix），準備用來進行傅立葉的逆轉換（inverse DFT）回到影像原來的空間，表示為：

$$f[m,n] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F[k,l]e^{i2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right)} \tag{17}$$

稀疏的 $F[k,l]$ 矩陣可以在儲存空間方面節省許多，[3]同時又能從 inverse

[3]稀疏矩陣的儲存方式很多種，譬如，最粗淺的方式，是儲存非 0 傅立葉係數所在的位置

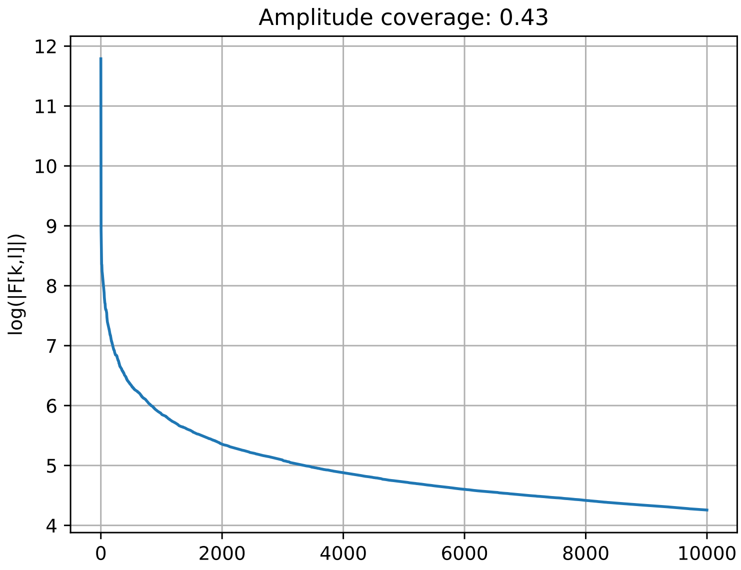

圖 14: 傅立葉係數 $F[k,l]$ 前 10000 大的 Amplitude 分布。

DFT 的過程還原影像，當然，還原的程度與保留的傅立葉係數之多寡相關。圖 15 展示幾張不同影像在相同的壓縮比下還原的面貌。圖上顯示的壓縮比是以最保守的 3 倍數計算（非 0 的傅立葉係數總數乘以 3），而還原的視覺效果則與圖的內容與大小有關。當壓縮比來到 16 倍，Lenna 影像已經顯露模糊，而差不多大小的 AFGHAN girl 較不明顯，尺寸更大的 AFGHAN girl 縮放在同樣大小的空間，看起來與原圖相差不多，這當然也是圖像壓縮的用途。下列程式碼示範對一張圖像以 FFT（Fast Fourier Transform）演算法計算 $F[k,l]$，保留傅立葉係數較大的 q%，其他則設為 0，最後以 inverse FFT 轉換回影像矩陣，成為壓縮後還原之影像。

```
import numpy as np
from numpy.fft import fft2, ifft2

# 令 img 為讀入的黑白影像矩陣
# Compute the 2D FFT of the image
fft_img = fft2(img)
# Keep only q% of the coefficients by setting the rest to zero
q = 2
n = int(q / 100 * fft_img.size)
tmp = fft_img.ravel()
tmp[np.argsort(abs(tmp))[:-n]] = 0
compressed_img = ifft2(tmp.reshape(fft_img.shape)).real
```

(k,l) 與係數值 $F[k,l]$。因此以一張 $M\times N$ 的影像為例，假設取 q 個最大的 $F[k,l]$ 值，則需儲存空間為 $3q$，壓縮比為 $\frac{MN}{3q}$。這個方式可說是最保守的方式，還有其他更精簡的方法。

(a) 512×512 Lenna

(b) 612×612 AFGHAN girl

(c) 2060×3200 AFGHAN girl

圖 15: 不同影像在相同 DFT 壓縮比下的還原效果。

4 習題

1. 證明式 (1) 中的 P 是一個正交投射矩陣，其定義如下
   定理：P is an orthogonal projection if and only if

$$P^2 = P \quad and\ P = P^T$$

2. 根據式 (4)，證明變數 $\mathbf{z}_q(i)$ 與 $\mathbf{z}_q(j)$ 不相關，即 $E(\mathbf{z}_q(i)\mathbf{z}_q(j)) = 0$。
3. 假設樣本數為 N，根據式 (4) 計算主成分分析在儲存空間上節省的比例。
4. 證明式 (4) 中的組合係數等於新的座標值，即 $\alpha = \mathbf{z}_q$。
5. 式 (7) 隱含一個事實：矩陣 AA^T 的特徵值為實數並且大於等於 0，試證明之。

6. 假設 **x** 為一 $p \times 1$ 的向量，欲將 **x** 投射到另一個維度為 $q(q < p)$ 的子空間 V，成為 $\mathbf{x}_q$，如圖 2 所示。假設 $w_1, w_2, \cdots, w_q$ 為構成子空間 V 的 orthonormal basis，即 $V = span\{w_1, w_2, \cdots, w_q\}$，證明其正交投射矩陣為 WW^T，即 $\mathbf{x}_q = WW^T\mathbf{x}$。其中 $W = [w_1\ w_2\ \cdots w_q]$

7. 製作出如圖 5 的壓縮圖像，patch size 為 16×16。不同的 patch size 是否產生不同的效果？

8. 同範例 3，但多觀察幾個不同的 q 值，直到壓縮的影像與原圖幾乎一樣。計算在不同的 q 值，壓縮的影像所需的儲存空間，及與原影像比較時，壓縮的比例為何？

9. 同上，試試不同的資料檔。

10. 如「觀察 3」所言，試著從式 (15) 模擬幾個人造的影像「3」出來。

11. 畫出 Lena 圖的前 100 個主成分影像圖。

References

[1] J. Latin, D. Carroll, P. E. Green, ”Analyzing Multivariate Data,”2003, Duxbruy.

[2] A. C. Rencher, ”Multivariate Statistical Inference and Applications,”1998, John Wily and Sons.

[3] T. Hastie, R. Tibshirani, J. Friedman, ”The Elements of Statistical Learning：Data Mining, Inference, and Prediction”.