

# **NON-LINEAR DIMENSION REDUCTION**

# **ISOMAP**

A final report for Applied Multivariate Analysis

Submitted by:

110024521 邱婉如

110024516 邱繼賢

107020010 張育豪

109024513 趙唯毓

106020015 羅靖雅

**2022 / 06 / 08**

## ABSTRACT

---

Isomap 為針對流形資料的學習方法，是透過非線性的降維方式，利用低維度資料模擬資料點在流形上的實際距離，盡可能保留資料間距離的資訊，相較於 PCA、classical MDS 等線性的降維方式，更適合用來處理具有流形結構的資料。

我們針對具有流形結構的資料以 Isomap 降維分析，與 classical MDS、PCA 比較得到的結論是，Isomap 能有效保留資料在流形上的幾何性質，則對於資料的分群能力更顯著。

# Contents

<b>Abstract</b>	<b>i</b>
<b>CHAPTER 1: INTRODUCTION</b>	<b>1</b>
1.1 Manifolds	1
1.2 Geodesic distance	1
1.3 Isomap idea	2
<b>CHAPTER 2: ALGORITHM</b>	<b>3</b>
2.1 Construct neighborhood graph	3
2.2 Compute graph distances	4
2.3 MDS with graph distances	7
2.4 Result	7
<b>CHAPTER 3: EXAMPLES</b>	<b>8</b>
3.1 Swiss Roll	8
3.2 Handwritten digits	12
<b>CHAPTER 4: CONCLUSION</b>	<b>15</b>
<b>CHAPTER 5: QUESTIONS</b>	<b>16</b>
<b>References</b>	<b>19</b>

# Chapter 1

## INTRODUCTION

### 1.1 Manifolds

流形 (manifold) 的定義含括了非常複雜的數學概念在此不談，但簡單來說，它是一個拓樸空間。可以看作是曲線跟曲面的推廣，整體看來是個柔軟的幾何形體，得以扭轉變形；不過局部卻是硬的結構，近似歐氏空間。

在流形上同樣可以進行微分跟積分的運算，而無窮連續可微的流形我們稱之為平滑流形 (smooth manifold)，根據 Nash's embedding theorem，我們可以把任一低維的平滑流形鑲嵌進高維度的歐氏空間  $\mathcal{X}$  之中，而  $\mathcal{X}$  在此就是我們資料點所在的空間  $\mathbb{R}^r$ 。例如 3.1 的瑞士捲就是將 2 維的平滑流形鑲嵌進  $\mathbb{R}^3$  之中。

如果賦予平滑流形  $\mathcal{M}$  一個用來測量點與點之間距離的度量  $d^{\mathcal{M}}$ ， $(\mathcal{M}, d^{\mathcal{M}})$  則被稱為黎曼流形 (Riemannian manifold)。

### 1.2 Geodesic distance

當黎曼流形  $(\mathcal{M}, d^{\mathcal{M}})$  是連通的，自然地它就成為一個度量空間，而  $d^{\mathcal{M}}$  決定了它的結構。在黎曼流形上直線的概念依然存在，那就是測地線 (geodesic)。

假設黎曼流形  $(\mathcal{M}, d^{\mathcal{M}})$  上有兩點  $y$  跟  $y'$ ，而  $C(y, y')$  是一個集合，它蒐集了所有在  $\mathcal{M}$  上連接  $y$  跟  $y'$  兩點的可微分曲線， $L(\cdot)$  計算曲線的弧長，接著我們就可以定義兩點間的距離：

$$d^{\mathcal{M}}(y, y') = \inf_{c \in C(y, y')} L(c)$$

由此可見  $d^{\mathcal{M}}$  找了兩點間最短的可微分曲線計算弧長來當作距離，這條曲線也就是測地線，而  $d^{\mathcal{M}}(y, y')$  則稱為  $y$  跟  $y'$  兩點間的測地距離 (geodesic distance)，可以想成是兩點在曲面上的最短路徑，相當於平面上的直線距離。

### 1.3 Isomap idea

首先 Isomap 是一種流形學習的方法，它假設資料點  $\{x_i\}$  是從有限個  $\{y_i\}$  透過非線性的平滑函數  $\psi$  映射而來， $\{x_i\}$  落在  $r$  維的歐氏空間  $(\mathcal{X} = \mathbb{R}^r, \|\cdot\|_{\mathcal{X}})$  之中，而  $\{y_i\}$  落在  $t$  維的黎曼流形  $(\mathcal{M}, d^{\mathcal{M}})$  上，通常  $t \ll r$ ，且存在  $\phi = \psi^{-1}$  使得  $\phi(x) = y$ 。由於我們不知道資料點  $\{x_i\}$  背後真實的黎曼流形以及映射函數，所以 Isomap 提供了一個將資料點  $\{x_i\} \subset \mathbb{R}^r$  降維映射到  $\{\hat{y}_i\} \subset \mathbb{R}^{t'}$  的方式，試圖用這些  $\{\hat{y}_i\}$  來估計  $\{y_i\}$ ，並用  $(\mathbb{R}^{t'}, \|\cdot\|_{\mathcal{X}})$  來還原  $t$  維的黎曼流形  $(\mathcal{M}, d^{\mathcal{M}})$ 。

所謂用  $(\mathbb{R}^{t'}, \|\cdot\|_{\mathcal{X}})$  來還原  $t$  綴的黎曼流形  $(\mathcal{M}, d^{\mathcal{M}})$ ，換句話說就是希望我們對資料降維後所建構出的空間得以保留某些黎曼流形  $(\mathcal{M}, d^{\mathcal{M}})$  的性質，而 Isomap 保留了什麼性質？這就得談到它背後的重要假設：

Isomap 的全名是 **isometric feature mapping algorithm**  
顧名思義它是一種等距映射的演算法，利用等距特徵來保留整體的幾何性質。  
背後假設平滑流形  $\mathcal{M}$  是  $\mathbb{R}^t$  空間中的凸集合且  $\psi: \mathcal{M} \rightarrow \mathcal{X}$  是等距映射函數。

簡單來說，Isomap 先是引入圖論的結構，定義資料點是否為鄰居來描述流形局部的範圍，如果兩個資料點是鄰居，那麼它就會反映流形局部的特徵，也就是兩點間的歐氏距離近似映射回流形上的測地距離；如果兩個資料點不是鄰居，那麼我們就會利用 shortest path algorithm 來計算兩點間的距離，並用該距離來估計真實流形的測地距離。建構完資料點間的距離矩陣，我們就等同於還原出黎曼流形的幾何結構，剩下的就是要如何在保留幾何結構的同時，把資料降維到  $(\mathbb{R}^{t'}, \|\cdot\|_{\mathcal{X}})$  來還原  $t$  綴的黎曼流形  $(\mathcal{M}, d^{\mathcal{M}})$ ，對此 Isomap 利用了 classical MDS 的概念，把我們所建構的距離矩陣透過它來將資料降到低維度，並且它會盡可能地保留距離資訊，使得我們可以用低維度中兩點的歐氏距離來還原它們在真實黎曼流形上的遠近關係。為了視覺化，通常會把  $t'$  設為 2。

最後統整一下 Isomap 的想法，它其實可以視為 classical MDS 的延伸，重點在於它假設資料點的背後存在一個流形結構，所以比起資料點間的歐氏距離，在流形結構上的測地距離更能夠描述彼此的遠近關係，但由於我們不知道真實的流形結構為何，所以它提供了方法來估計測地距離，然後再利用 classical MDS 保留距離的資訊將資料降維並視覺化。

# Chapter 2

## ALGORITHM

### 2.1 Construct neighborhood graph

在執行 Isomap 時，第一個步驟是建立一個 neighborhood graph 來描述鄰近資料點間的 graph distance 為何，而首先必須定義何謂資料點的鄰居，常用來定義的方法有兩種，分別為  $\varepsilon$ -rule 以及 KNN-rule。

- $\varepsilon$ -rule：設定一個固定距離  $\varepsilon$ ，若任兩點間的歐氏距離小於或等於  $\varepsilon$ ，則令該兩點為鄰居。
  - 優點：時間複雜度較低，只需要  $O(dn^2)$  即可建立 neighborhood graph，其  $d$  為資料的維度， $n$  為資料數量。
  - 缺點：在建立 neighborhood graph 時需確保每個資料點至少有一個鄰居才能夠定義資料點間的 graph distance 為何，因此若資料點太稀疏或是有離群值，就需要取較大的  $\varepsilon$  來確保每個點都有鄰居，但可能會造成用歐氏距離逼近 graph distance 的效果沒有太好。
- KNN-rule：設定一個  $k$  值，令資料點的鄰居為最靠近該資料點的  $k$  個點。
  - 優點：即使資料中有離群值也不會影響到其他資料點找鄰居，因為在 KNN 中找鄰居的方法是在各個資料點中找最靠近該點的其他點為哪些，因此不會受到離群值影響。
  - 缺點：在找鄰居時，其時間複雜度較高，需要  $O(kdn^2)$ ，但能夠用 KD-tree，或是 Ball-tree 的結構，將時間複雜度降低到  $O((dn+k)\log(n))$ 。

在每個資料點找到其鄰居後，就可以建立 neighborhood graph，若兩資料點為鄰居，就在兩資料點間連接一條邊 (edge)，上方的權重為 graph distance，也就是利用鄰居間的歐氏距離來逼近，而能夠逼近的原因來自於前面提過流形局部的結構會近似於歐氏空間。

## 2.2 Compute graph distances

建立完 neighborhood graph 後，我們得到了局部的 graph distance，接著我們將使用 shortest path algorithm，建構整個資料點間的 graph distance，得到資料點的 distance matrix。

常用的 shortest path algorithm 有 Floyd's algorithm 以及 Dijkstra's algorithm，而因為在建立完 neighborhood graph 後，資料點間相連的 edges 較少，Dijkstra's algorithm 的時間複雜度會較 Floyd's algorithm 低了不少，因此在 Isomap 中大多使用 Dijkstra's algorithm 來找任兩點間的 shortest path，接下來也只會針對 Dijkstra's algorithm 說明。

### Dijkstra's Algorithm

Dijkstra's algorithm 是一個 greedy algorithm，專門用來處理單一資料點到其餘資料點的最短路徑，他的 pseudo code 如下：

---

**Input:** A directed graph  $G=(V,E)$  and a source vertex  $v_0$   
**Output:** For each  $v \in V$ , the length of a shortest path from  $v_0$  to  $v$

```
1:  $S=v_0$  and  $L(v_0)=0$ 
2: for  $i = 1$  to  $n$  do /* Initialization */
3:   if  $(v_0, v_i) \in E$  then  $L(v_i)=c(v_0, v_i)$ .
4:   else  $L(v_i)=\infty$ .
5:   end if
6: end for
7: for  $i = 1$  to  $n$  do /*Find  $i$ th nearest neighbor of  $v_0$ */
8:   Choose  $u \in V \setminus S$  such that  $L(u)$  is the smallest.
9:    $S = S \cup u$ 
10:  for all  $w \in V \setminus S$  do
11:     $L(w) = \min\{L(w), L(u) + c(u, w)\}$ . /* Relaxation */
12:  end for
13: end for
```

---

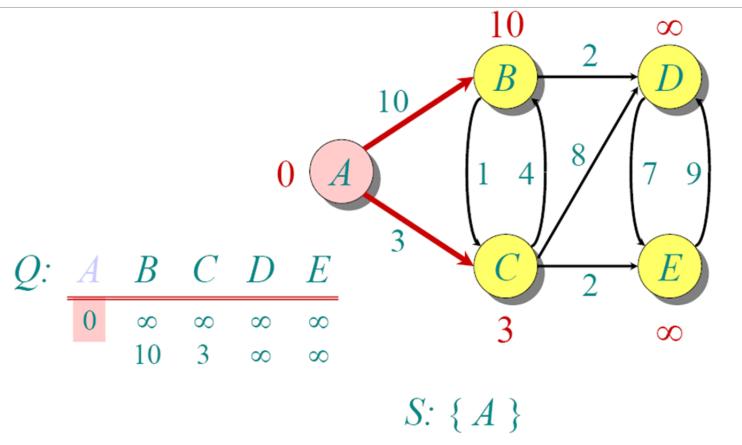
在 Dijkstra's algorithm 中，Input 是一個 graph，也就是上一個步驟建構完的 neighborhood graph：其中  $V$  為 vertex，指的是 graph 中的資料點； $E$  為 edge，指的是 graph 中連接點和點的邊。在執行 Dijkstra's algorithm 以前，還需先定義  $v_0$ ，在 output 中就能夠得到  $v_0$  到其他資料點的 shortest path 為何。而在 pseudo code 中出現的  $L(v)$  指的是資料點  $v$  和目標資料點  $v_0$  的距離。

在 Dijkstra's algorithm 中，最主要的概念就是從目標點  $v_0$  附近開始找起，由近到遠更新所有點的 shortest path，而尋找方式則是利用兩個 sets： $S$  以及  $V \setminus S$ ， $S$  放的是已經找到和  $v_0$  的 shortest path 為何的資料點， $V \setminus S$  則是放還沒找到 shortest path 的資料點，而在每次更新時也只要針對  $V \setminus S$  進行更新即可。

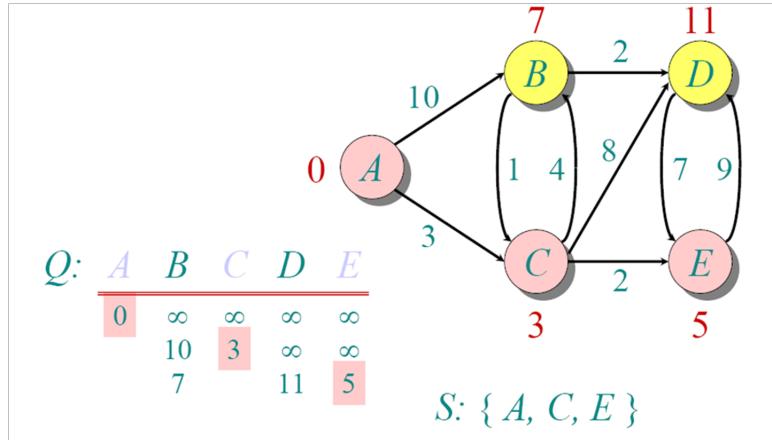
首先要做的就是初始化設定，因為目標點  $v_0$  到自己的距離一定是 0，因此可以先設定  $L(v_0) = 0$ ，且將  $v_0$  放入  $S$  內。接著執行第一個 for loop (pseudo code 第二行到第六行)，若資料點和  $v_0$  有 edge 連接，也就是兩者為鄰居，則令  $L(v_i) = c(v_0, v_i)$ ，若不是鄰居，則令  $L(v_i) = \infty$ 。

初始化之後，在  $V \setminus S$  中找哪個資料點最靠近  $v_0$ ，將該點放入  $S$  中，其距離  $L(u)$  即為點  $u$  到目標點  $v_0$  的 shortest path，接著再針對點  $u$  對其他點  $w \in V \setminus S$  進行更新，更新式子為  $L(w) = \min\{L(w), L(u) + c(u, w)\}$ ，執行到最後就能夠獲得  $v_0$  到其餘資料點的 shortest path 為何。（以上找 shortest path 及更新的過程為 pseudo code 第 7 行到第 12 行）

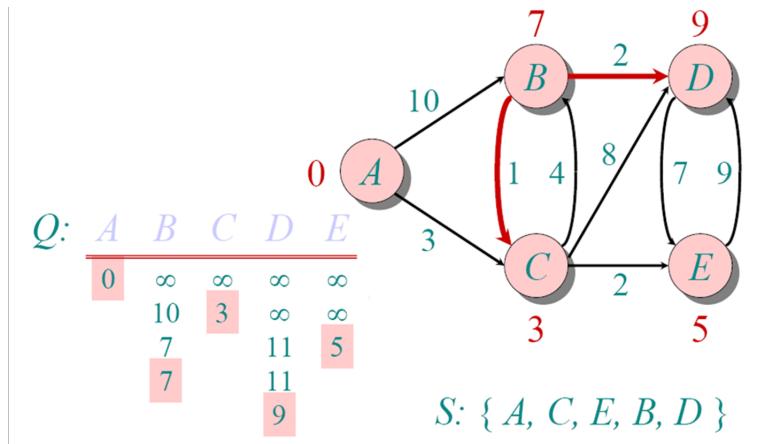
## Example



首先先做初始化，因為目標點為 A，因此將  $L(A)$  設為零，並將點 A 放入  $S$ ，接著根據 graph 的資訊，若資料點和點 A 為鄰居，則令  $L(A, v_i) = c(A, v_i)$ ，其餘的距離則令為無限大。做完後得到  $A \rightarrow B$  的距離為 10 及  $A \rightarrow C$  的距離為 3，因為 3 較小，因此我們會將 C 放入  $S$ ，並用  $L(C)$  的資訊針對其他在  $V \setminus S$  的點進行更新。



因為和資料點 C 為鄰居的點有 B、D、E，因此可以將這 3 點和 A 的距離進行更新。更新過後，A 和 B 的距離變為 7，因為  $L(C) + c(B,C) = 7 < 10$ 。而因為 D 和 E 原先都不是 A 的鄰居，因此直接令  $L(D) = L(C) + c(C,D)$  和  $L(E) = L(C) + c(C,E)$ ，再確認  $L(B)$ 、 $L(D)$  和  $L(E)$  後，得到  $L(E)$  距離最短，因此將 5 作為 A 和 E 的 shortest path，並將 E 放入 S，接著用  $L(E)$  的資訊對其餘在  $V \setminus S$  的點進行更新。



重複上面動作後，能夠得到 A 到其他所有點的 shortest path 為何，即是 A 到其他資料點的 graph distance。因此，若對於所有資料點都執行一次 Dijkstra's algorithm，我們就能夠從 neighborhood graph 中得到所有資料點間的 graph distance，也就等同得到了資料點間的 distance matrix。

## 2.3 MDS with graph distances

透過 Dijkstra's algorithm 得到 distance matrix 之後，我們能夠將 distance matrix 代入 classical MDS，如同下式：

$$A_n^G = -\frac{1}{2}HSH$$

其中  $H = I_n - n^{-1}J_n$ ， $J_n = I_nI_n'$ ，S 則為前面得到的 distance matrix。接著再對  $A_n^G$  做 eigenvalue decomposition，得到 eigenvalue 和 eigenvectors，若降維至 t 維度，其對應座標： $\hat{Y} = (\sqrt{\lambda_1}v_1, \dots, \sqrt{\lambda_t}v_t)$ ，其中  $\lambda_i$  為  $A_n^G$  第  $i$ th 大的 eigenvalue， $v_i$  為對應的 eigenvector，且  $\lambda_1 > \dots > \lambda_t$ ，如此一來就能夠將高維度的流形結構資料降低到低維度上。

## 2.4 Result

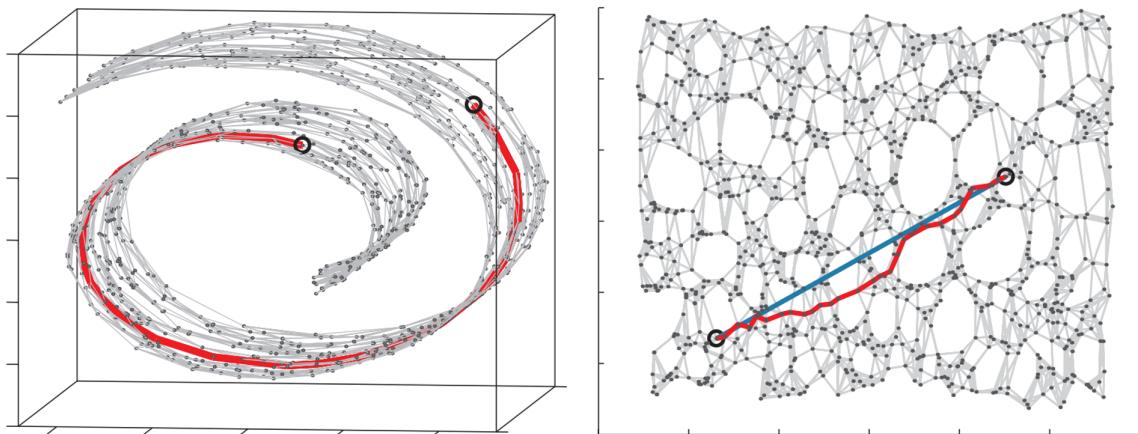


Figure 2.1: Results of ISOMAP (Source from JB Tenenbaum et al.(2000))

上圖為降維前和利用 Isomap 降維後的比較，左圖紅線為將資料點利用 Isomap 前兩步驟所得到的最短路徑，也就是我們拿來近似 geodesic distance 的 graph distance，右圖則是將三維空間中的二維流形展開到二維平面上，其中紅線即是利用 Isomap 得到的結果，藍線則是在二維平面上兩點的直線距離，也就是流形上的 geodesic distance，因此能夠看到在經由 Isomap 後得到的結果，雖然相較藍色線較曲折一點，但整體而言，使用 Isomap 降維後，成功保存了流形的結構，近似的效果也沒有太差。

# Chapter 3

## EXAMPLES

### 3.1 Swiss Roll

接下來讓我們來看一個瑞士捲 (Swiss Roll) 的例子，瑞士捲的結構可以從以下變數變換得到：

$$(X_i, Y_i) \stackrel{iid}{\sim} U((0, 4\pi) \times (0, 4\pi)) \rightarrow \left( \frac{X_i}{5} \cos(X_i), \frac{X_i}{5} \sin(X_i), Y_i \right)$$

其中變數  $X_i$  控制圖型的旋轉，變數  $Y_i$  控制圖型的厚度。

```
data = read.table("swiss_roll.dat.txt")
sol = read.table("swiss_roll_sol.dat.txt")
# 會跑非常久，可以只選取部分資料進行
# set.seed(1220)
# choose = sample(1:3000, 1000)
# data = data[choose,]
# sol = sol[choose,]
library(scatterplot3d)
scatterplot3d(data[,1], data[,3], data[,2], color = sol[,1],
pch = 16, angle = 40, scale.y = 0.5)
```

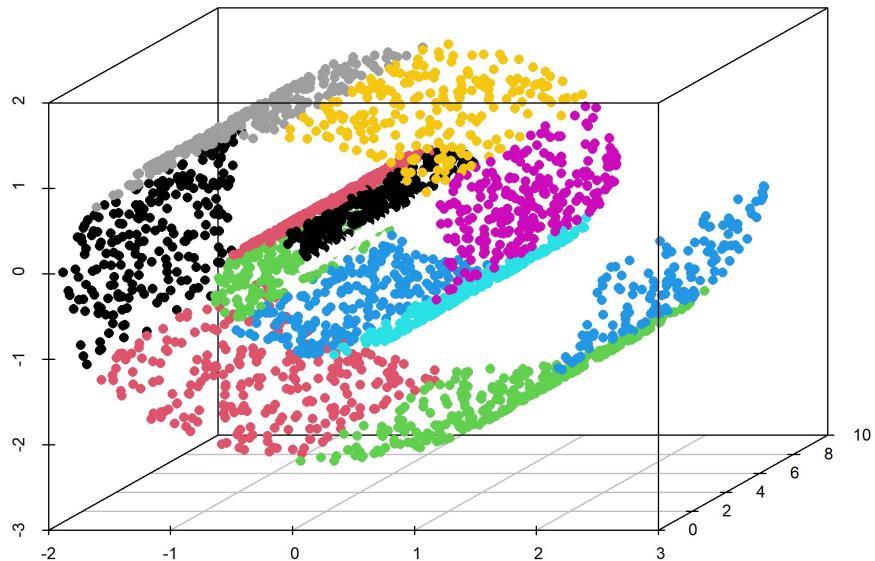


Figure 3.1: Swiss Roll

可以看出瑞士捲呈現為一個在三維空間中的二維流形，上圖中相同的顏色代表  $X_i$  數值相近的資料點，接下來先以使用歐氏距離 MDS 對資料進行降維處理：

```
D = dist(data, method = "euclidean")
mds = cmdscale(D, k=2, eig=T)
mds.point = mds$points
scatterplot3d(mds.point[,2], rep(0,3000), mds.point[,1],
color = sol[,1], pch = 16, scale.y = 0, ylab = "")
```

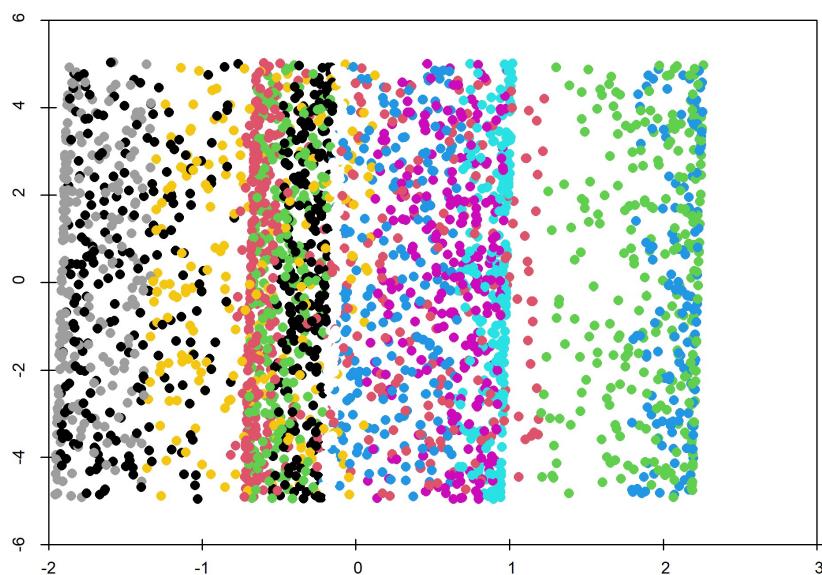


Figure 3.2: MDS

降至二維視覺化後各種顏色都混雜在一起，因為歐氏距離 MDS 的目的是希望保留資料點之間的歐氏距離，可以把 Figure 3.2 想像成將 Figure 3.1 從上往下壓扁至二維平面，但這樣的降維方式並沒有辦法保留原本資料有著的非線性流形特徵，接下來使用 Isomap 的方式對資料進行降維處理：

```
library(vegan)
iso = isomap(D, ndim = 2, k = 5)
iso.point = data.matrix(iso$points)
scatterplot3d(-iso.point[,1], rep(0,3000), iso.point[,2],
color = sol[,1], pch = 16, scale.y = 0, ylab = "")
```

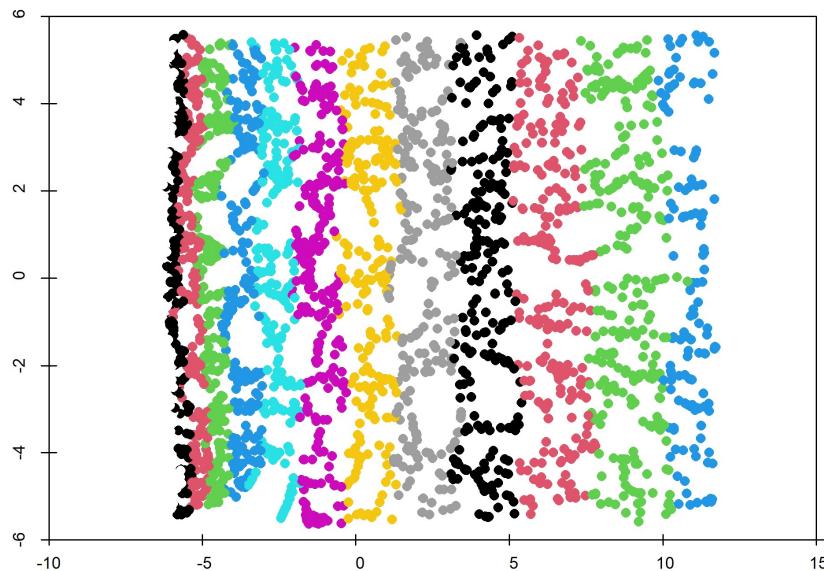


Figure 3.3: ISOMAP

透過 Isomap 降至二維平面後，可以看出各種顏色的分布被清楚的區分開來，因為 Isomap 是一種非線性的降維方式，保留了資料點之間的 graph distance，進而保留了資料點原本的流形特徵，Figure 3.3 可以理解成將 Figure 3.1 中的瑞士捲攤平成二維平面。

因為 Isomap 這種降維方式和歐氏距離 MDS 差別只在於希望保留原始資料之間的距離類型不同，前者為 graph distance，後者為歐氏距離，觀察兩種類型的距離矩陣：

```
round(as.matrix(D)[1:6,1:6],2)
```

	1	2	3	4	5	6
1	0.00	7.80	7.21	3.82	3.11	5.87
2	7.80	0.00	1.02	4.05	10.29	2.66
3	7.21	1.02	0.00	3.47	9.60	2.75
4	3.82	4.05	3.47	0.00	6.52	2.45
5	3.11	10.29	9.60	6.52	0.00	8.70
6	5.87	2.66	2.75	2.45	8.70	0.00

Figure 3.4: Euclidean distance

```
graph.dist = isomapdist(D, k = 5)
round(as.matrix(graph.dist)[1:6,1:6],2)
```

	1	2	3	4	5	6
1	0.00	8.95	8.65	4.58	7.68	14.56
2	8.95	0.00	1.43	4.60	12.82	11.53
3	8.65	1.43	0.00	4.30	11.61	10.77
4	4.58	4.60	4.30	0.00	8.78	11.98
5	7.68	12.82	11.61	8.78	0.00	10.31
6	14.56	11.53	10.77	11.98	10.31	0.00

Figure 3.5: Graph distance

很明顯可以看出，兩兩資料點之間的 graph distance 基本上都是大於該兩點之間的歐氏距離，這是因為 graph distance 的計算方式是先判斷兩資料點是否為 neighbor，若為 neighbor 則使用歐式距離，若不為 neighbor 則使用 shortest path，這樣的計算方式就導致 graph distance 的數值大小一定會「大於等於」歐氏距離。

再來觀察此筆資料進行 Isomap 時的 Scree plot：

```
plot(iso$eig[1:10], type = "bl")
```

可以看出前兩個 eigenvalues 就已經佔了總變異中很大的比例，因此此筆資料使用 Isomap 降維至二維仍然可以很好的保留原始資料大部分的變異。

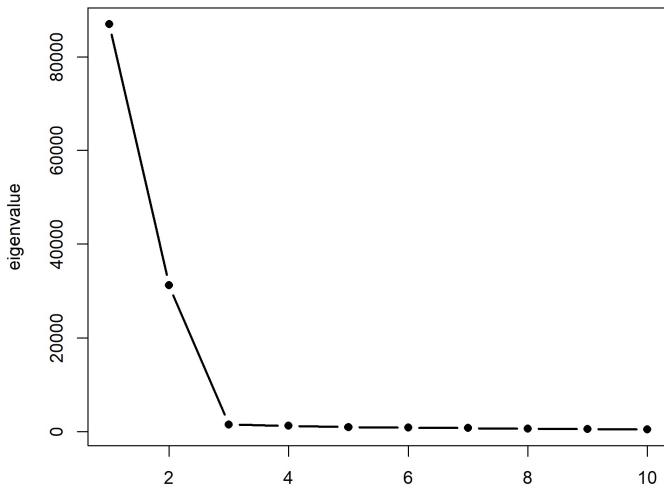


Figure 3.6: Scree plot

## 3.2 Handwritten digits

最後，我們將 ISOMAP 應用在真實的資料上。



Figure 3.7: digits

Figure 3.7 表示我們取自 Scikit-Learn 之手寫數字圖像數據集，共有 1,797 個觀測值，每一筆觀測值為  $8 \times 8$  像素的矩陣，即為一張具有 64 個變數的手寫數字圖像。此數據集包含 10 個相異值：0 至 9，而我們選取數字 0 至 4 來執行 Isomap。

首先，我們設定 KNN-rule 中不同的 neighbor 個數  $k$  構建 neighborhood graph，比較 Isomap 的結果。

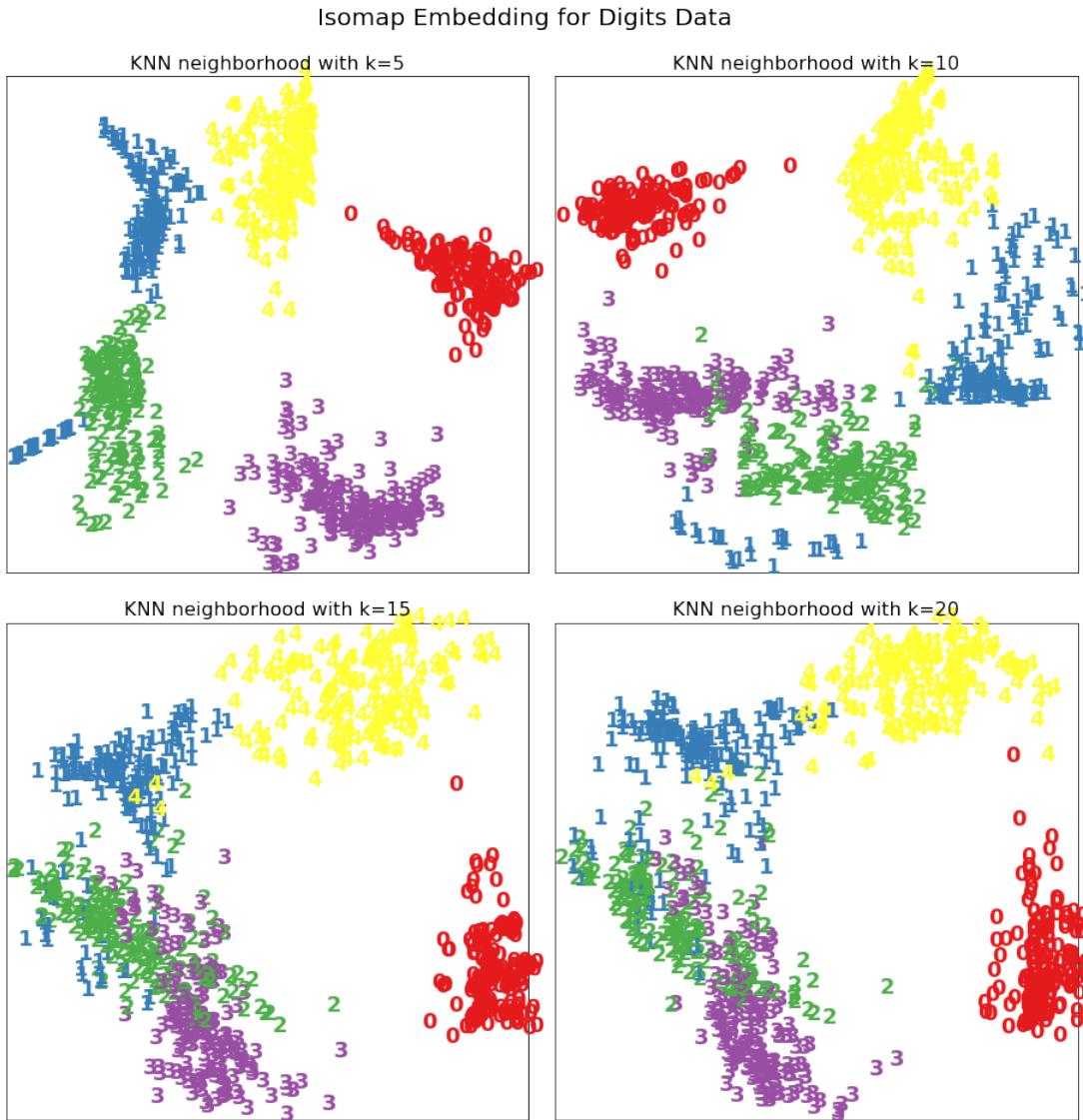


Figure 3.8: Comparison of different number of neighbors

由上圖可看出不同的  $k$  值可能會使 Isomap 的降維效果有所不同， $k$  值設至 15 或 20 時，數字 2 及 3 有重疊的現象出現，這是因為過大的  $k$  值，可能會將相距較遠的兩個點設為鄰居，導致降維後無法保留某些資料點的特徵。故我們取  $k$  值為 10 的結果進行以下比較。

與線性的降維方式做比較，如：PCA、MDS：

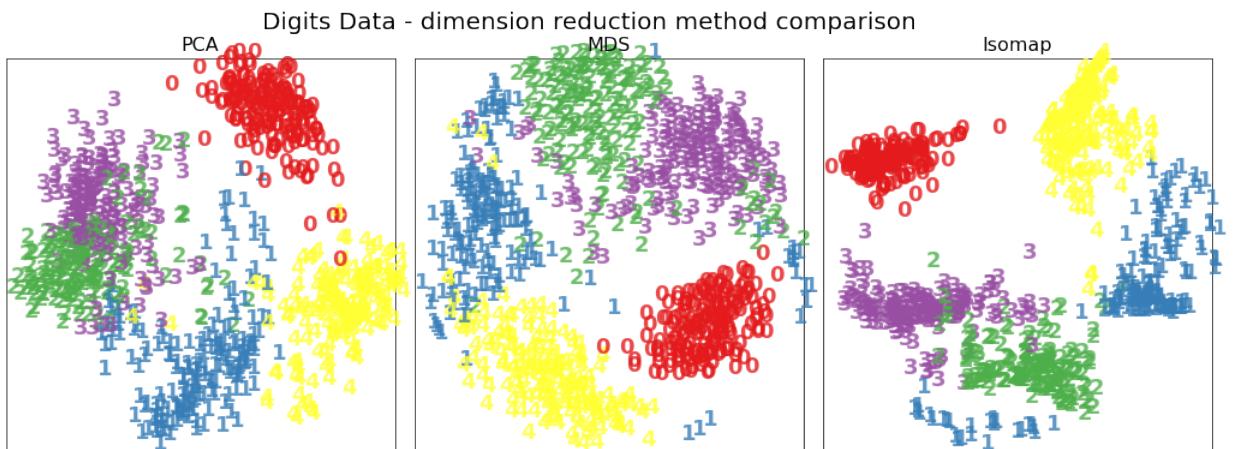


Figure 3.9: Comparison with different Dimension Reduction Method

這三個降維方法確實都有依數字分成 5 群，但是 Isomap 的降維結果較可以清楚地看出這些數字的分界。另外可得知因為此筆資料背後可能具有流形結構，所以 Isomap 比較好的反應了數據在該流形結構上的測地距離，外觀相差越多的數據會相距越遠，像是 0 與其他數字樣子明顯不同，結果亦與其他數字分離較遠。

# Chapter 4

## CONCLUSION

Isomap 主要是用來處理流形資料，目標是將資料降維，並且盡可能保留資料點間的距離資訊，是透過低維度資料點間歐氏距離估計它們在流形上的真實距離的學習方法。首先要建立 neighborhood graph，利用  $\varepsilon$ -rule 或 KNN-rule 定義資料點的鄰居，得到局部的 graph distance，再透過 shortest path algorithm 計算非鄰居資料點間的 graph distance，最後得到整體資料的 distance matrix，再利用 classical MDS 保留距離的資訊將資料降維並視覺化。

再來我們分別利用 classical MDS 與 Isomap 處理瑞士捲形態的資料，在將資料降維後，只有 Isomap 較能夠保留資料點在流形上的測地距離，且 scree plot 顯示資料經由 Isomap 降至二維後仍能保留原始資料的大部分變異，所以在這個例子中，相較於 classical MDS 而言，Isomap 更適合用來處理流形資料。再來利用 Isomap 分析手寫數字圖像，在運用 shortest path algorithm 中的 KNN-rule 時，改變不同 k 值，觀察對數字圖像的降維效果，發現在 k 值較大時，容易導致資料降維後無法保留完整的特徵，使得數字資料群間有重疊的現象。再來比較 k 值為 10 時，Isomap、PCA、MDS 的降維效果，Isomap 將數字分群的效果較佳，且會根據外觀差距越大而分離的越遠。

Isomap 提供一個處理流形資料的降維方法，能夠有效保留資料點在流形上完整的距離資訊，但因為得到 neighborhood graph 的時間複雜度相對較高，當資料過大時會消耗大量時間，所以需要注意使用的時機以及目的。另外，為了解決消耗時間較多的問題，發展出 Landmark Isomap，只針對資料中部分的資料點進行距離的估計，以達到降低時間複雜度的目的。

# Chapter 5

## QUESTIONS

- How to obtain the graph distance matrix?
- What is the difference between Euclidean distance MDS and ISOMAP?

以 homework 4 的 car price data 為例：

1. 求得 graph distance matrix :

```
car_data = read.csv("CarPrice_Assignment.csv", header = T)
car_data.2 = car_data[,c(10:14,17,19:25)]
D_car = dist(scale(car_data.2))
graph_dis = as.matrix(isomapdist(D_car, k=10))
round(graph_dis[1:6,1:6],2)
```

	1	2	3	4	5	6
1	0.00	0.00	7.79	6.51	7.47	6.54
2	0.00	0.00	7.79	6.51	7.47	6.54
3	7.79	7.79	0.00	4.22	2.81	2.78
4	6.51	6.51	4.22	0.00	1.96	1.44
5	7.47	7.47	2.81	1.96	0.00	0.93
6	6.54	6.54	2.78	1.44	0.93	0.00

## 2. 與歐氏距離 MDS 做比較：

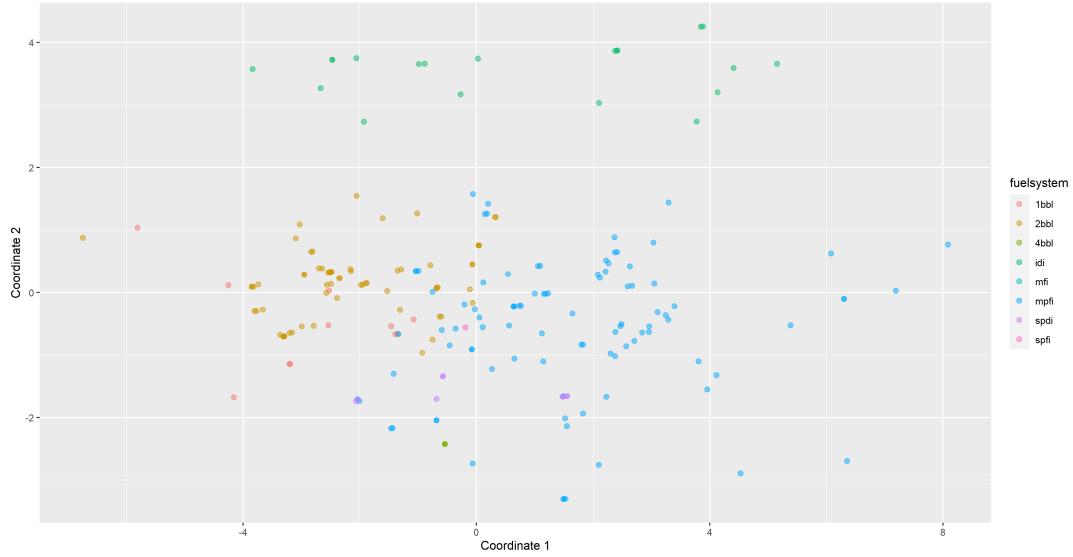


Figure 5.1: Standard MDS

```
iso = isomap(D_car, ndim = 2, k = 10)
ggplot(as.data.frame(iso$points)) +
  geom_point(aes(-Dim1, Dim2, color = car_data$fuelsystem),
             size = 2, alpha = 0.5) +
  labs(x = "Coordinate 1", y = "Coordinate 2", color = "fuelsystem")
```

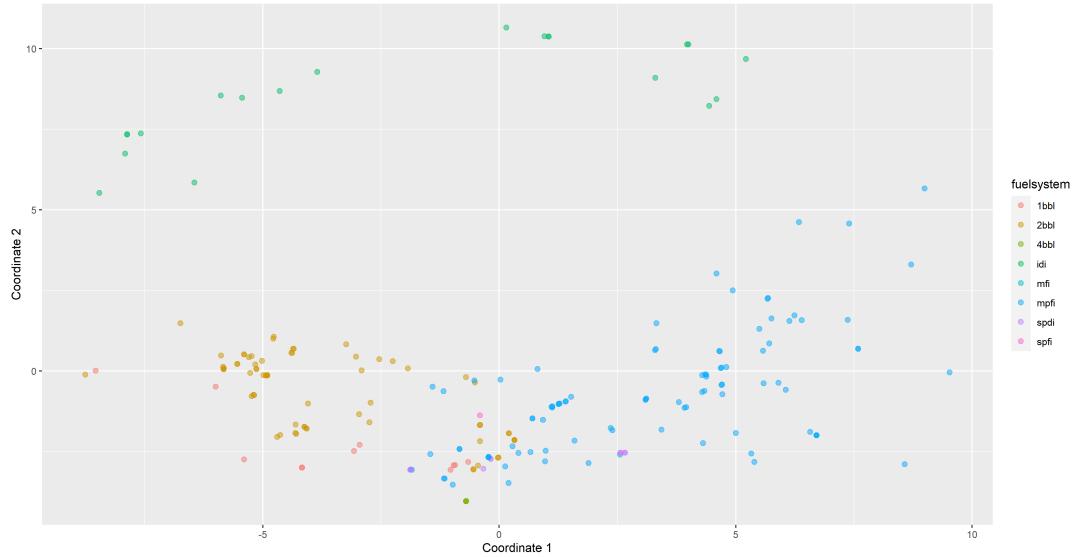


Figure 5.2: ISOMAP

可以看出使用 Isomap 的方法時，降維後資料分布的 scale 較大，這是因為 graph distance 的大小一定「大於等於」歐氏距離，在此筆資料看起來，用 Isomap

可以將不同 fuelsystem 的資料點區分得稍微較分散，但並沒有像是前面瑞士捲的例子那麼明顯，像是下方的藍色和棕色交界處還是跟 MDS 時一樣有混雜的情況，雖然比例上有所減少，可以推測原始資料沒有特別明顯的流形分佈。

# Bibliography

- [1] Izenman, A. J. Modern multivariate statistical techniques. Regression, classification and manifold learning. (2008).
- [2] J.B. Tenenbaum, V. de Silva and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. Science, vol. 290, pp. 2319–2323, 2000.
- [3] Dijkstra's Algorithm by Laksman Veeravagu and Luis Barrera :  
<https://reurl.cc/j1G201>
- [4] TD 1 - Dimensionality Reduction : <https://reurl.cc/M0NVR4>
- [5] Scikit-Learn Manifold learning on handwritten digits : <https://scikit-learn.org/>