Clustering with Local Density Peaks-Based Minimum Spanning Tree

Dongdong Cheng, Qingsheng Zhu, *Member, IEEE*, Jinlong Huang, Quanwang Wu, *Member, IEEE*, and Lijun Yang

Clustering with Local Density Peaks-Based Minimum Spanning Tree. IEEE Trans. Knowl. Data Eng. 33(2): 374-387 (2021)

報告者:闕中一、吳海韜

這是一篇將 MST 應用在 clustering 問題的論文。防疫期間我們會以文字代替口頭報告的方式向大家介紹

Outline

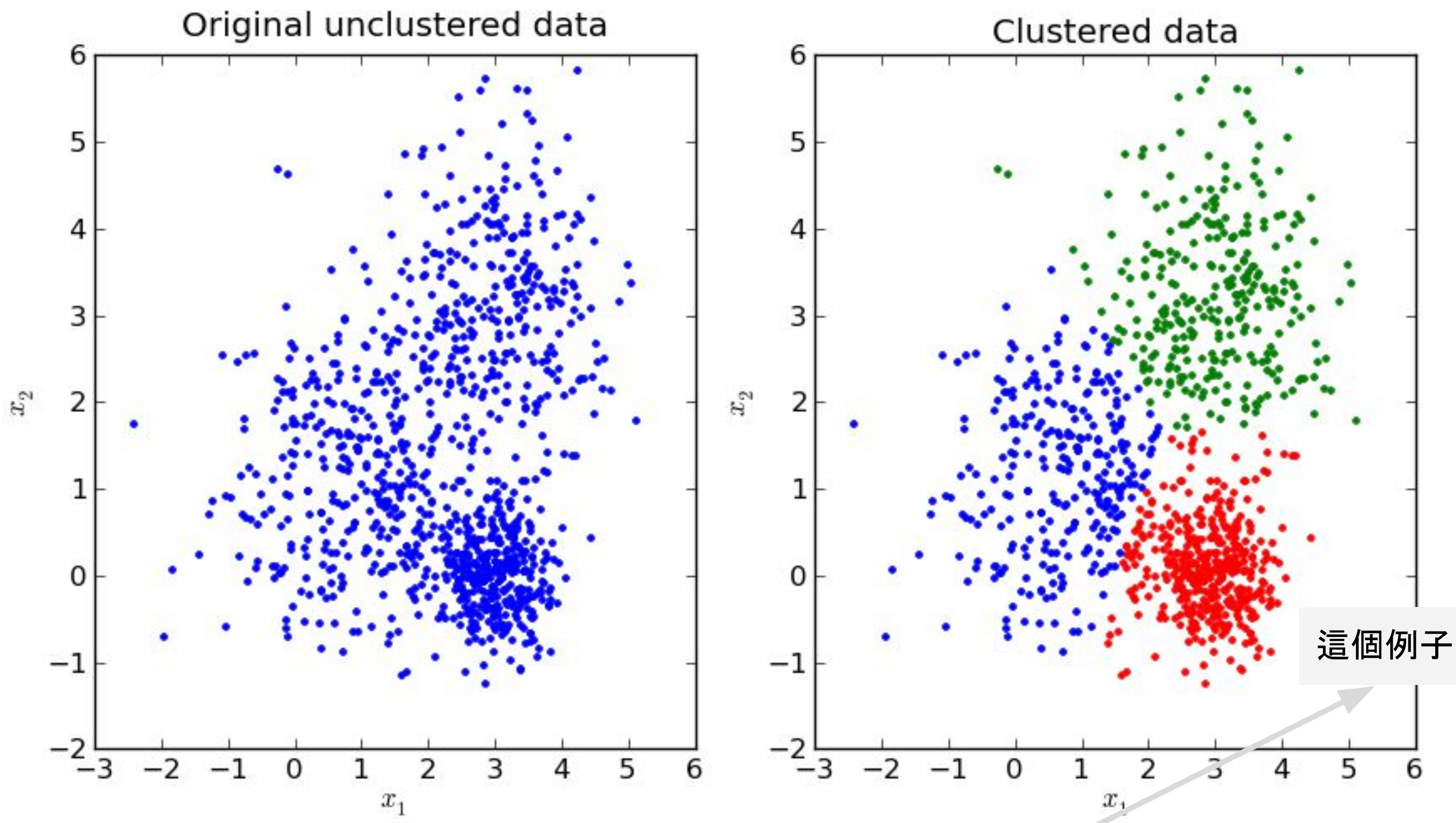
1. Why clustering?
2. How to generate clusters?
    a. Center-Based
    b. Density-Based
    c. MST-Based
3. **Local Density Peaks-Based Minimum Spanning Tree (LDP-MST)**
4. Experiments
    a. Synthetic Datasets
    b. Real Datasets
    c. Evaluation on Running Time

第一和第二部分介紹 clustering 這個應用問題的背景。第三部分介紹論文提出的方法、最後呈現實驗結果

Why clustering?

Why clustering?

https://github.com/bheemnitd/2D-KMeans-Clustering

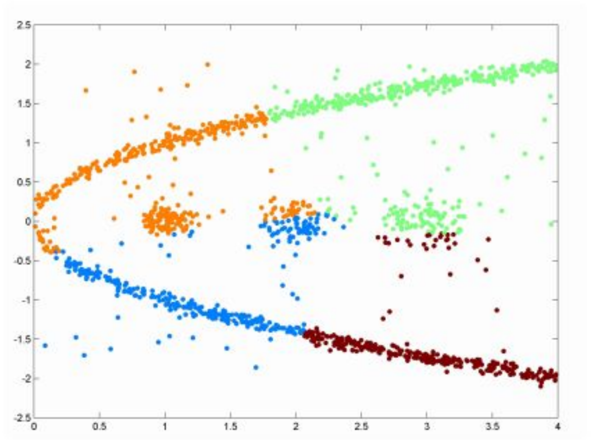


這個例子中 k=3

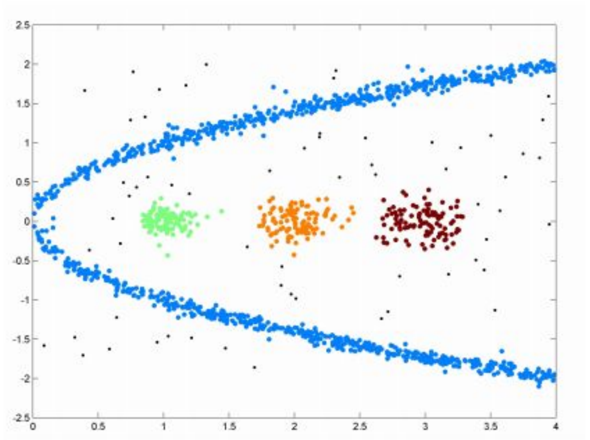
所謂的 clustering 就是輸入 n 個樣本點，將它們以某種規則 (演算法、參數)劃分為 k 群。其中分群的數目 k 值可以是事先給定，或是由演算法本身自己去發現。本文的討論大部分基於 k 給定的情形

Why clustering?

1. Unsupervised learning: Label is not needed
    a. Especially useful when the label is expensive
2. Discover how many categories are in your sample
3. Anomaly detection
4. Find useful intuitions to help supervise learning



(a) Kmeans

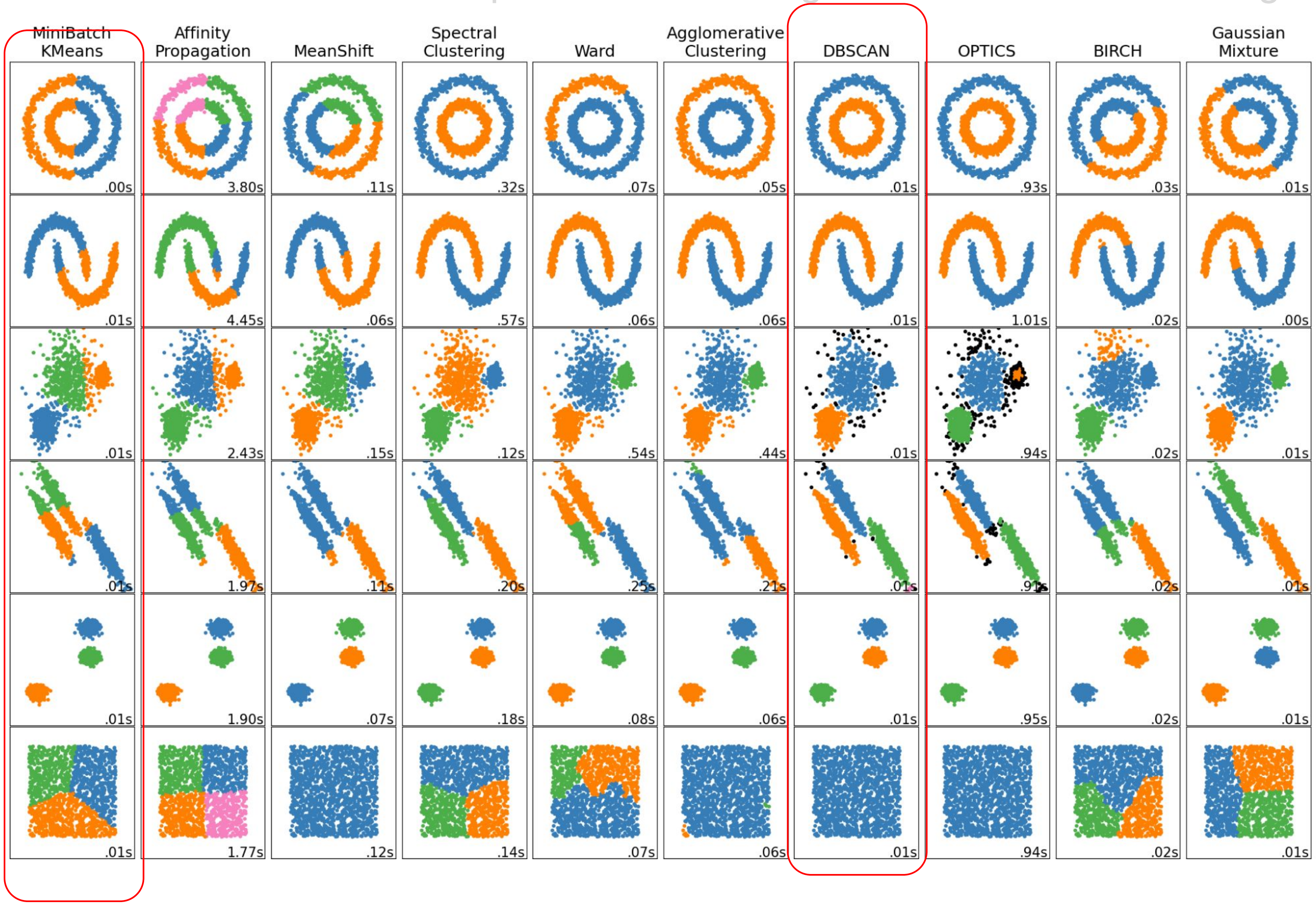


(b) DBSCAN

善用 clustering 可以幫助我們瞭解資料中隱藏的規律，這裡列舉兩種常見的方法 Kmeans 和 DBSCAN，相同顏色的樣本點稱為一個 cluster。上圖我們也注意到不同演算法得到的結果可能大相逕庭，在這個例子中 DBSCAN 的結果可能更接近我們人類的「直覺」

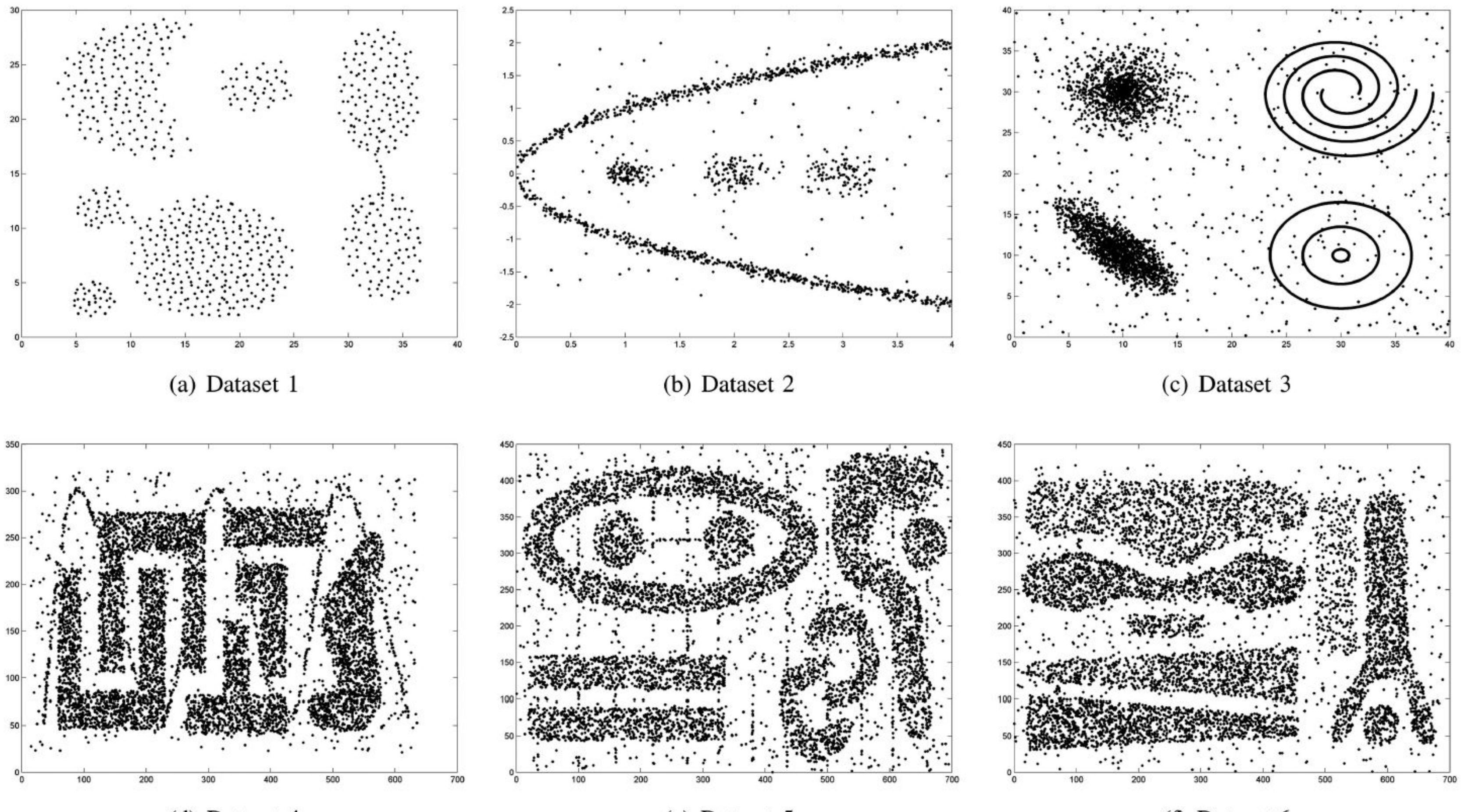
Clustering algorithm matters

https://scikit-learn.org/stable/modules/clustering.html



這是 scikit-learn 套件說明文檔提供的例子。比較紅框 KMeans 和 DBSCAN 兩者的結果，可以看出不同演算法在不同 datasets 上會有很不一樣的行為

How to tell is it good or bad?


(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

(d) Dataset 4 (e) Dataset 5 (f) Dataset 6

Fig. 5. The original synthetic data sets.

這是作者在實驗中用到人工合成的六組 datasets。直覺上它們都有一些明顯的結構，那麼接下來關鍵就是如何評價一個 clustering 方法對我們有沒有幫助，量化比較這些演算法的好壞

How to tell is it good or bad?

Label helps

TABLE 4
Synthetic Datasets

Datasets	Instances	Clusters	Source
Dataset 1	788	7	[43]
Dataset 2	1400	4	[44]
Dataset 3	8537	7	[45]
Dataset 4	8000	6	[8]
Dataset 5	8000	9	[8]
Dataset 6	8000	8	[8]

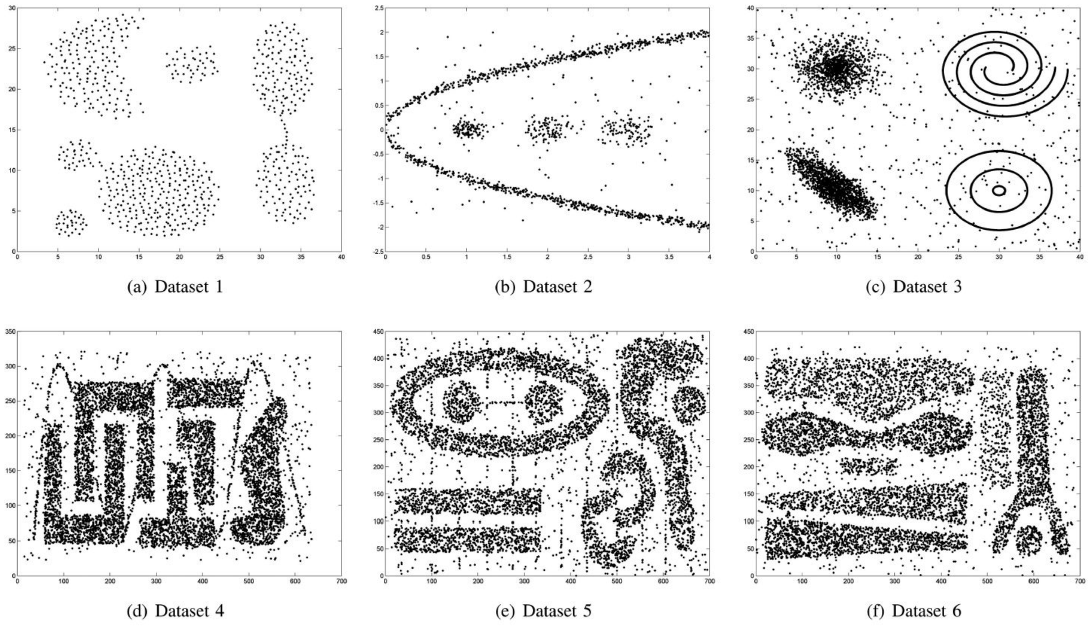

(a) Dataset 1 (b) Dataset 2 (c) Dataset 3
(d) Dataset 4 (e) Dataset 5 (f) Dataset 6

Fig. 5. The original synthetic data sets.

一個直接的想法就是根據當初人工生成這些 cluster 的規則當作 ground truth，用一般機器學習作分類問題的角度去評價 clustering 的結果

How to tell is it good or bad?

Metrics

1. Clustering Accuracy (ACC)

$$ACC = \frac{1}{n}\sum_{i=1}^{n} \delta(y_i, map(c_i)), \qquad \delta(x,y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

2. Normalized Mutual Information (NMI)

$$NMI = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} n_{i,j} \log\left(\frac{n \cdot n_{i,j}}{n_i \cdot n_j}\right)}{\sqrt{\left(\sum_i n_i \log \frac{n_i}{n}\right)\left(\sum_j n_j \log \frac{n_j}{n}\right)}}$$

n_i	# in category i
n_j	# in cluster j
$n_{i,j}$	# in category i & cluster j

在有 ground truth 的前提下，這是作者所採用的兩種量化指標，其中 ACC 便可以想成是分類問題的準確率。這兩個指標都是越大越好。

How to tell is it good or bad?

Running Time / Time Complexity

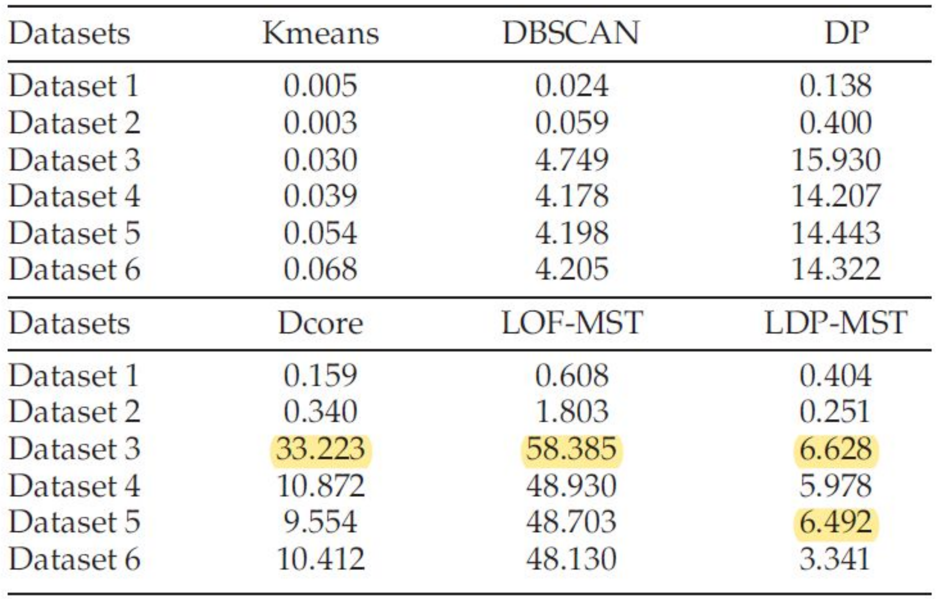


TABLE 7
The Running Time of Each Clustering Algorithm on Synthetic Data Sets (s)

Datasets	Kmeans	DBSCAN	DP
Dataset 1	0.005	0.024	0.138
Dataset 2	0.003	0.059	0.400
Dataset 3	0.030	4.749	15.930
Dataset 4	0.039	4.178	14.207
Dataset 5	0.054	4.198	14.443
Dataset 6	0.068	4.205	14.322
Datasets	**Dcore**	**LOF-MST**	**LDP-MST**
Dataset 1	0.159	0.608	0.404
Dataset 2	0.340	1.803	0.251
Dataset 3	33.223	58.385	6.628
Dataset 4	10.872	48.930	5.978
Dataset 5	9.554	48.703	6.492
Dataset 6	10.412	48.130	3.341

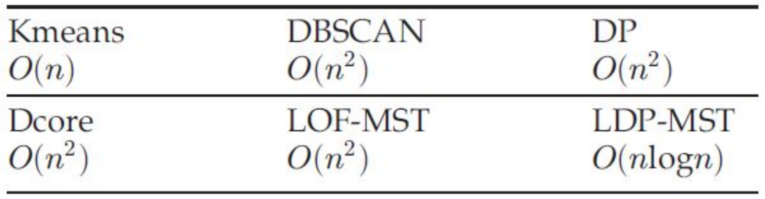


TABLE 3
The Time Complexity of Algorithms

Kmeans	DBSCAN	DP
$O(n)$	$O(n^2)$	$O(n^2)$
Dcore	LOF-MST	LDP-MST
$O(n^2)$	$O(n^2)$	$O(n\log n)$

演算法的複雜度以及實際執行時間也是我們關心的。這兩張表格為示意圖，後續會有深入的 說明

How to generate clusters?

接下來簡單介紹現有的 clustering 方法

How to generate clusters?

1. Center-Based
   a. Kmeans
   b. DP
   c. DCore
2. Density-Based
   a. DBSCAN
3. MST-Based
   a. LOF-MST
   b. <u>LDP-MST</u>

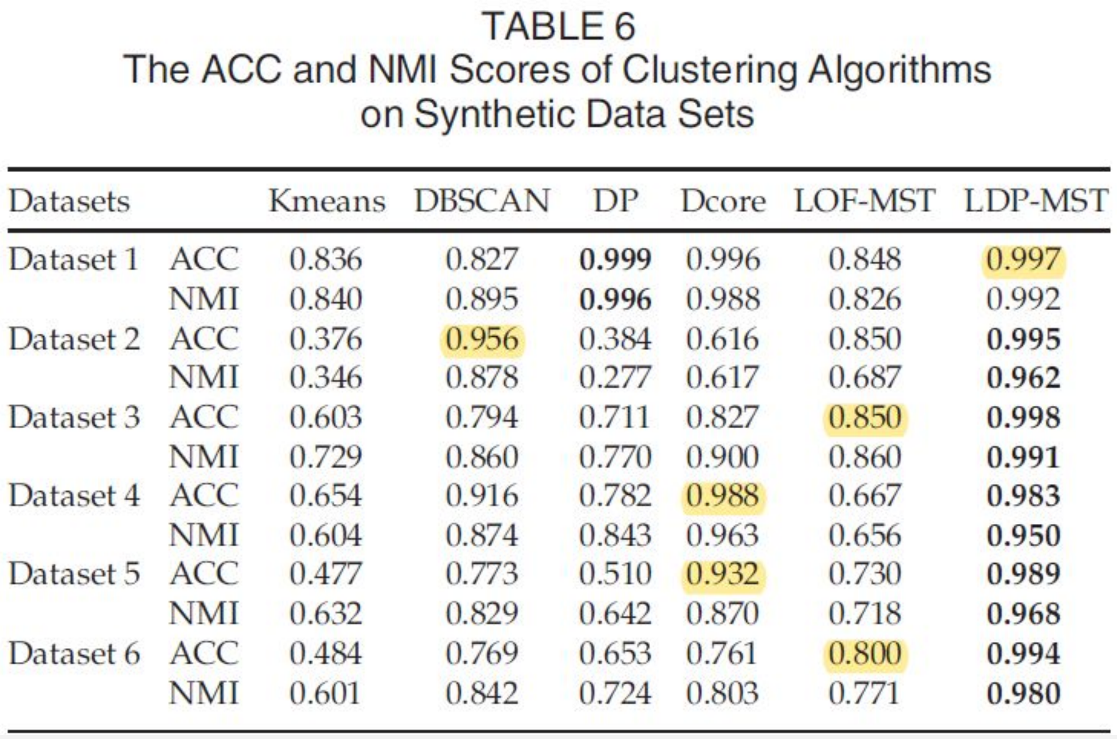


TABLE 6
The ACC and NMI Scores of Clustering Algorithms on Synthetic Data Sets

Datasets		Kmeans	DBSCAN	DP	Dcore	LOF-MST	LDP-MST
Dataset 1	ACC	0.836	0.827	**0.999**	0.996	0.848	0.997
	NMI	0.840	0.895	**0.996**	0.988	0.826	0.992
Dataset 2	ACC	0.376	0.956	0.384	0.616	0.850	**0.995**
	NMI	0.346	0.878	0.277	0.617	0.687	**0.962**
Dataset 3	ACC	0.603	0.794	0.711	0.827	0.850	**0.998**
	NMI	0.729	0.860	0.770	0.900	0.860	**0.991**
Dataset 4	ACC	0.654	0.916	0.782	0.988	0.667	**0.983**
	NMI	0.604	0.874	0.843	0.963	0.656	**0.950**
Dataset 5	ACC	0.477	0.773	0.510	0.932	0.730	**0.989**
	NMI	0.632	0.829	0.642	0.870	0.718	**0.968**
Dataset 6	ACC	0.484	0.769	0.653	0.761	0.800	**0.994**
	NMI	0.601	0.842	0.724	0.803	0.771	**0.980**

(示意圖)這篇論文探討的六種演算法和六個 datasets，後續會詳細說明

這是作者提出三大分類中有被實驗囊括探討的六個演算法。其中我們會介紹 Kmeans, DBSCAN 以及簡化版的 MST

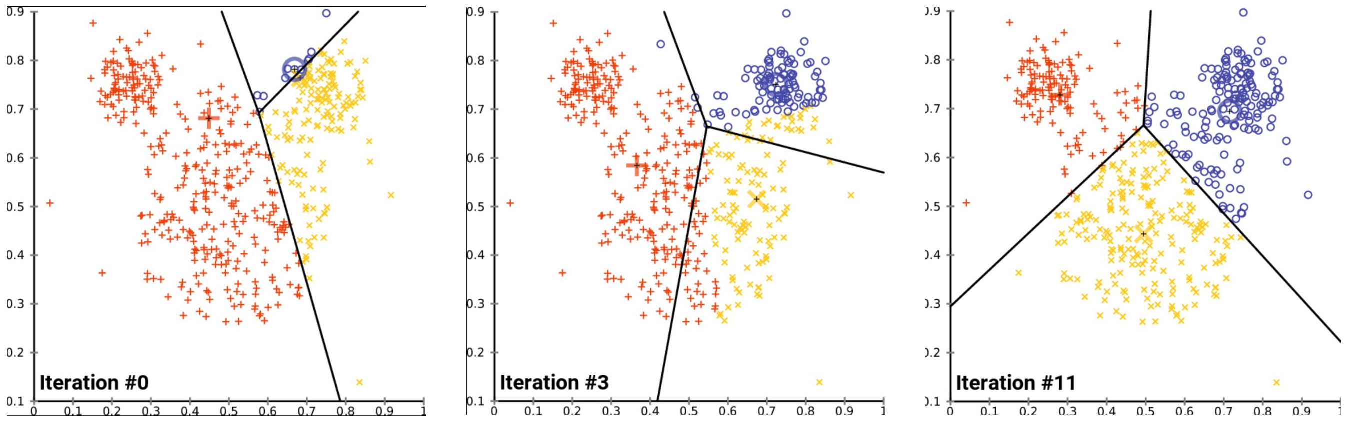
Center-Based: Kmeans

https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

1. Specify number of clusters K
2. Initialize centroids by first shuffling the dataset and then **randomly selecting K data points** for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
   a. Recalculate means (centroids) for observations assigned to each cluster.

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

https://en.wikipedia.org/wiki/K-means_clustering



Kmeans 是最常見的 clustering 方法。一開始隨機選定中心點 (centroids), 對所屬中心點的樣本取平均迭代得到新的中心點直到收斂

Density-Based: DBSCAN

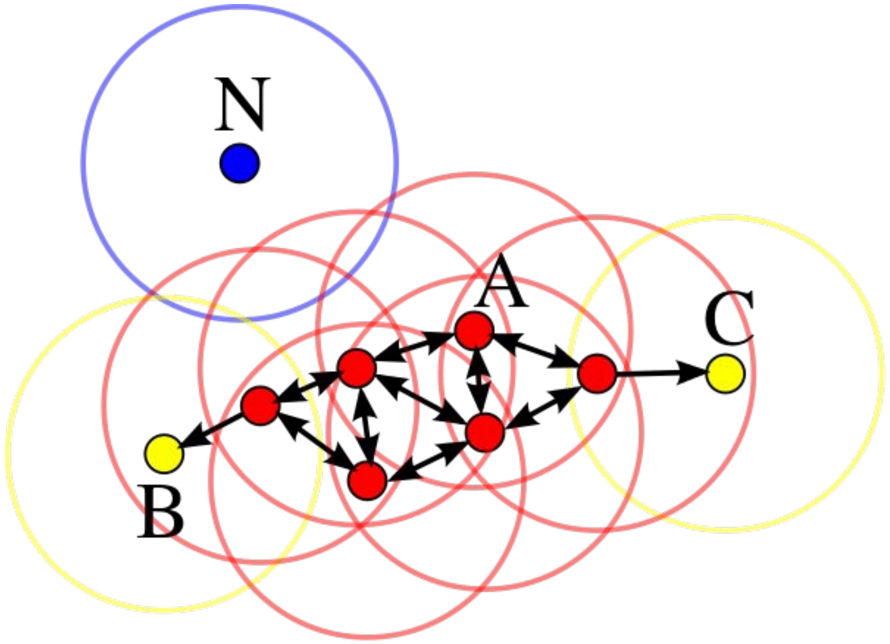
https://en.wikipedia.org/wiki/DBSCAN

Density-based spatial clustering of applications with noise

Parameters: ε, minPts

Definitions

- A point p is a **core point** if at least minPts points are within distance ε of it
- A point q is **directly reachable** from p if point q is within distance ε from core point p. Points are only said to be directly reachable from core points.
- A point q is **reachable** from p if there is a path p1, ..., pn with p1 = p and pn = q, where each pi+1 is directly reachable from pi
- All points not reachable from any other point are outliers or noise points



DBSCAN 是一種常用的 clustering 方法。右示意圖紅色核心點 (core points) 至少有 minPts=4 個鄰居在eps半徑之內，黃點不是核心但屬於能被核心點直達 (directly reachable)。藍點則視為雜訊點

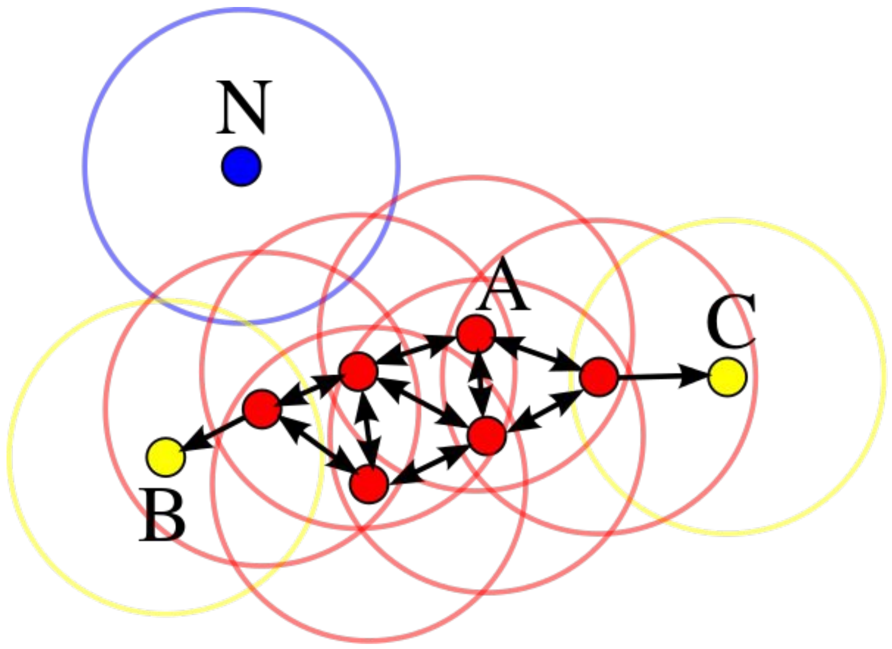
Density-Based: DBSCAN

https://en.wikipedia.org/wiki/DBSCAN

Density-based spatial clustering of applications with noise

The DBSCAN algorithm can be abstracted into the following steps:

1. Find the points in the ε (eps) neighborhood of every point, and identify the core points with more than minPts neighbors
2. Find the connected components of core points on the neighbor graph, ignoring all non-core points
3. Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor, otherwise assign it to noise



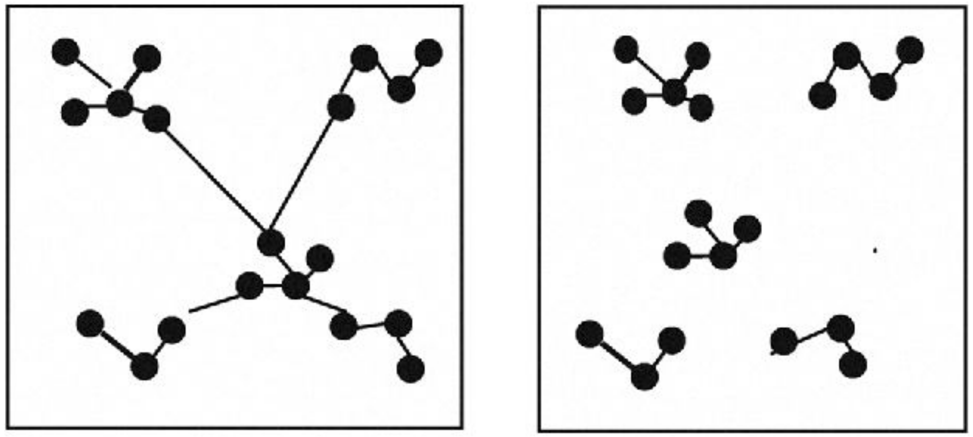
DBSCAN 算法很直覺地先找出所有核心點，互相可達的自成一個 cluster (也就是說 clusters 數目是演算法自行決定的)。其餘能被直達的點就歸屬該 cluster，否則視為雜訊

MST-Based

Clustering with Local Density Peaks-Based Minimum Spanning Tree. IEEE Trans. Knowl. Data Eng. 33(2): 374-387 (2021)

https://chih-ling-hsu.github.io/2017/09/01/Divisive-Clustering

1. Traditional MST-based clustering algorithms first construct an MST according to a distance measure
2. Continually remove **inconsistent edges** to get a set of connected components (clusters)
3. Do step 2 until the **terminal condition** is satisfied


(a) An MST connecting all the data points (b) Clusters after removal of longest edges

一般 MST-Based 方法可以歸納成三步：對資料點建 MST (一般來說是歐氏距離)、移除 inconsistent edge 來形成 cluster (例如移除最長的邊)、檢查終止條件 (例如分到有 k 群為止)。今天報告的論文 LDP-MST 也適用這個框架。然而在有雜訊點的情況下，歐氏距離不一定是最理想的設定。想像 k=3 卻剛好有兩個雜訊點 p1, p2 離我們主要關心的群特別遠，得到 p1, p2 各一群、正常點全部自成一群，而這不是我們想要的結果。所以定義 inconsistent edge 的方法相當重要

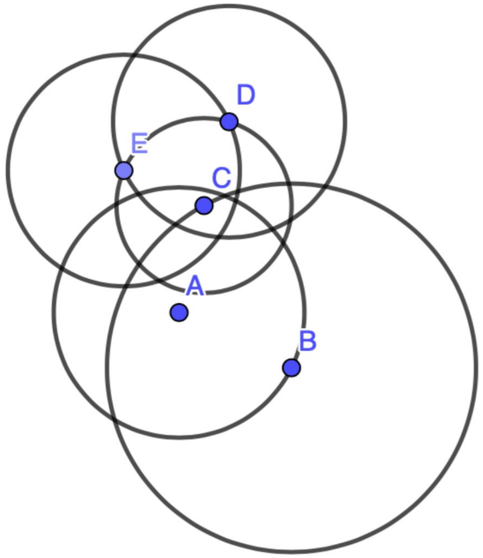
Local Density Peaks-Based Minimum Spanning Tree

(LDP-MST)

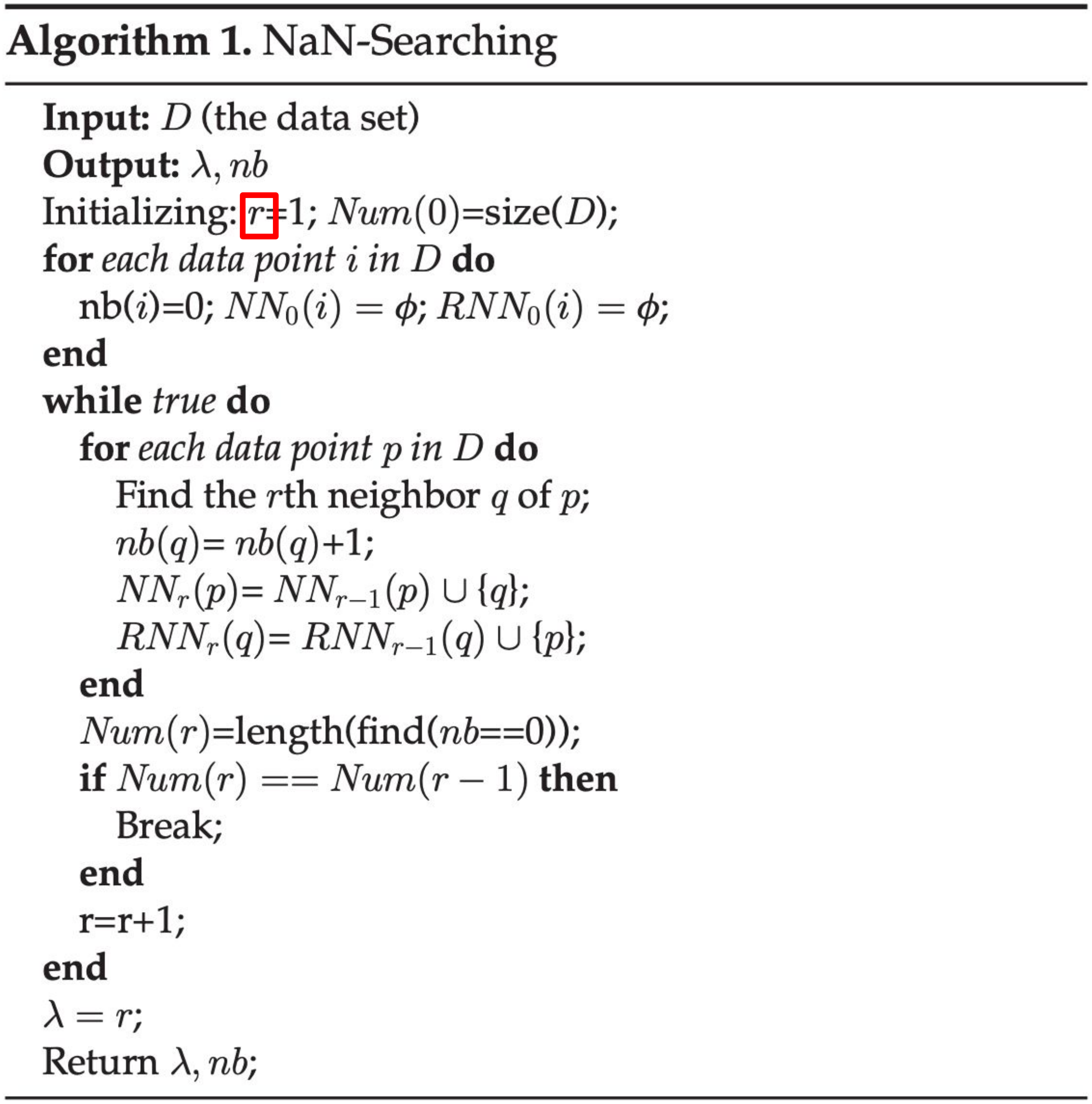
Nature Neighbor

Definition 1 (k Nearest Neighbors). *The k nearest neighbors of point p are a set of points x in D with $d(p,x) \leq d(p,o)$, i.e., $NN_k(p) = \{x \in D | d(p,x) \leq d(p,o)\}$.*

Definition 2 (Reverse k Nearest Neighbors). *The reverse k nearest neighbors of point p are a set of points x in D that consider p as its one of the k nearest neighbors, i.e., $RNN_k(p) = \{x \in D | p \in NN_k(x)\}$.*



在開始介紹具體方法之前，先帶大家了解兩個關於 nature neighbor的基本概念。右圖為一 k=3的例子，NN(C)={C,D,E}，RNN(C)={A,B,C,D,E}。在計算點p的kNN時點p自己需算在內，因此點p也本應在RNN(p)中。但注意，後續Algorithm 1 NaN-Searching的NN(p)與RNN(p)將不再考慮p點自身。



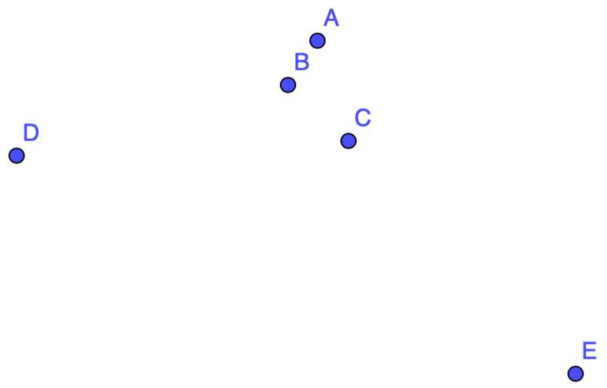
Algorithm 1. NaN-Searching

```
Input: $D$ (the data set)
Output: $\lambda, nb$
Initializing: $\boxed{r}$=1; $Num(0)$=size($D$);
for each data point $i$ in $D$ do
    nb($i$)=0; $NN_0(i) = \phi$; $RNN_0(i) = \phi$;
end
while true do
    for each data point $p$ in $D$ do
        Find the $r$th neighbor $q$ of $p$;
        $nb(q) = nb(q)+1$;
        $NN_r(p) = NN_{r-1}(p) \cup \{q\}$;
        $RNN_r(q) = RNN_{r-1}(q) \cup \{p\}$;
    end
    $Num(r)$=length(find($nb$==0));
    if $Num(r) == Num(r-1)$ then
        Break;
    end
    r=r+1;
end
$\lambda = r$;
Return $\lambda, nb$;
```

r為neighbor searching range，自r=1開始進行迭代。詳見後續例子幫助理解。

Algorithm 1. NaN-Searching

```
Input: $D$ (the data set)
Output: $\lambda$, $nb$
Initializing: $r$=1; $Num(0)$=size($D$);
for each data point $i$ in $D$ do
    nb($i$)=0; $NN_0(i) = \phi$; $RNN_0(i) = \phi$;
end
while true do
    for each data point $p$ in $D$ do
        Find the $r$th neighbor $q$ of $p$;
        $nb(q)= nb(q)+1$;
        $NN_r(p)= NN_{r-1}(p) \cup \{q\}$;
        $RNN_r(q)= RNN_{r-1}(q) \cup \{p\}$;
    end
    $Num(r)$=length(find($nb$==0));
    if $Num(r) == Num(r-1)$ then
        Break;
    end
    r=r+1;
end
$\lambda = r$;
Return $\lambda$, $nb$;
```

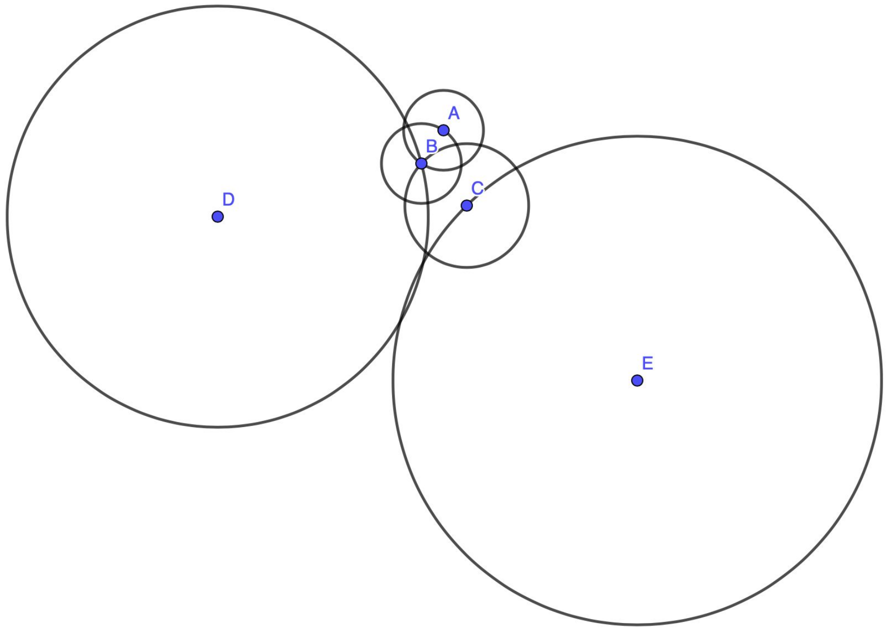


r=1	A	B	C	D	E
nb	0	0	0	0	0

nb(q)指當searching range為r時，點q被別的點計入r nearest neighbor的次數，初始均為 0。

Algorithm 1. NaN-Searching

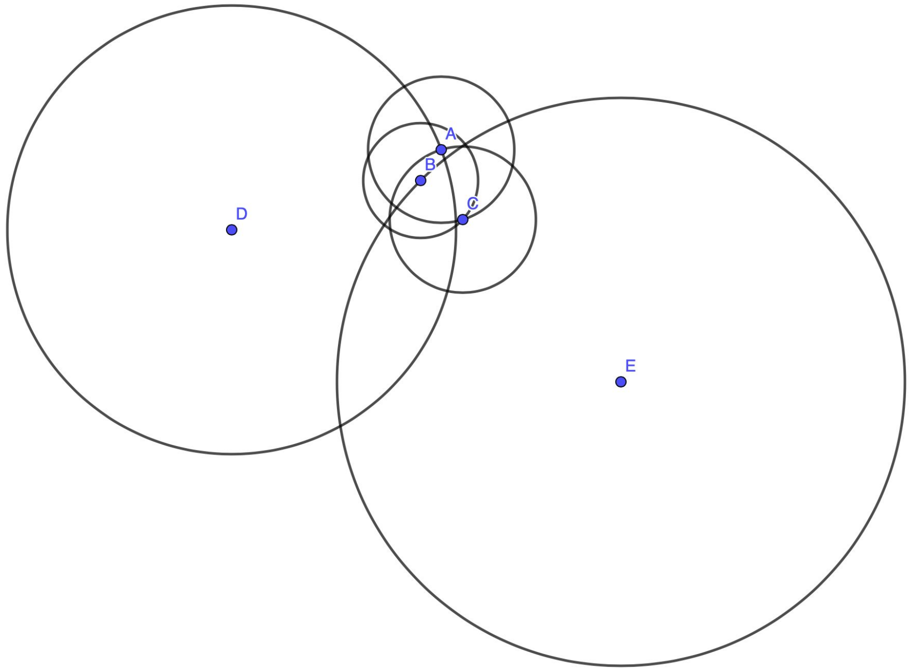
```
Input: $D$ (the data set)
Output: $\lambda, nb$
Initializing: $r$=1; $Num(0)$=size($D$);
for each data point $i$ in $D$ do
    nb($i$)=0; $NN_0(i) = \phi$; $RNN_0(i) = \phi$;
end
while true do
    for each data point $p$ in $D$ do
        Find the $r$th neighbor $q$ of $p$;
        $nb(q) = nb(q)+1$;
        $NN_r(p) = NN_{r-1}(p) \cup \{q\}$;
        $RNN_r(q) = RNN_{r-1}(q) \cup \{p\}$;
    end
    $Num(r)$=length(find($nb$==0));
    if $Num(r) == Num(r-1)$ then
        Break;
    end
    r=r+1;
end
$\lambda = r$;
Return $\lambda, nb$;
```



r=2	A	B	C	D	E
nb	1	3	1	0	0

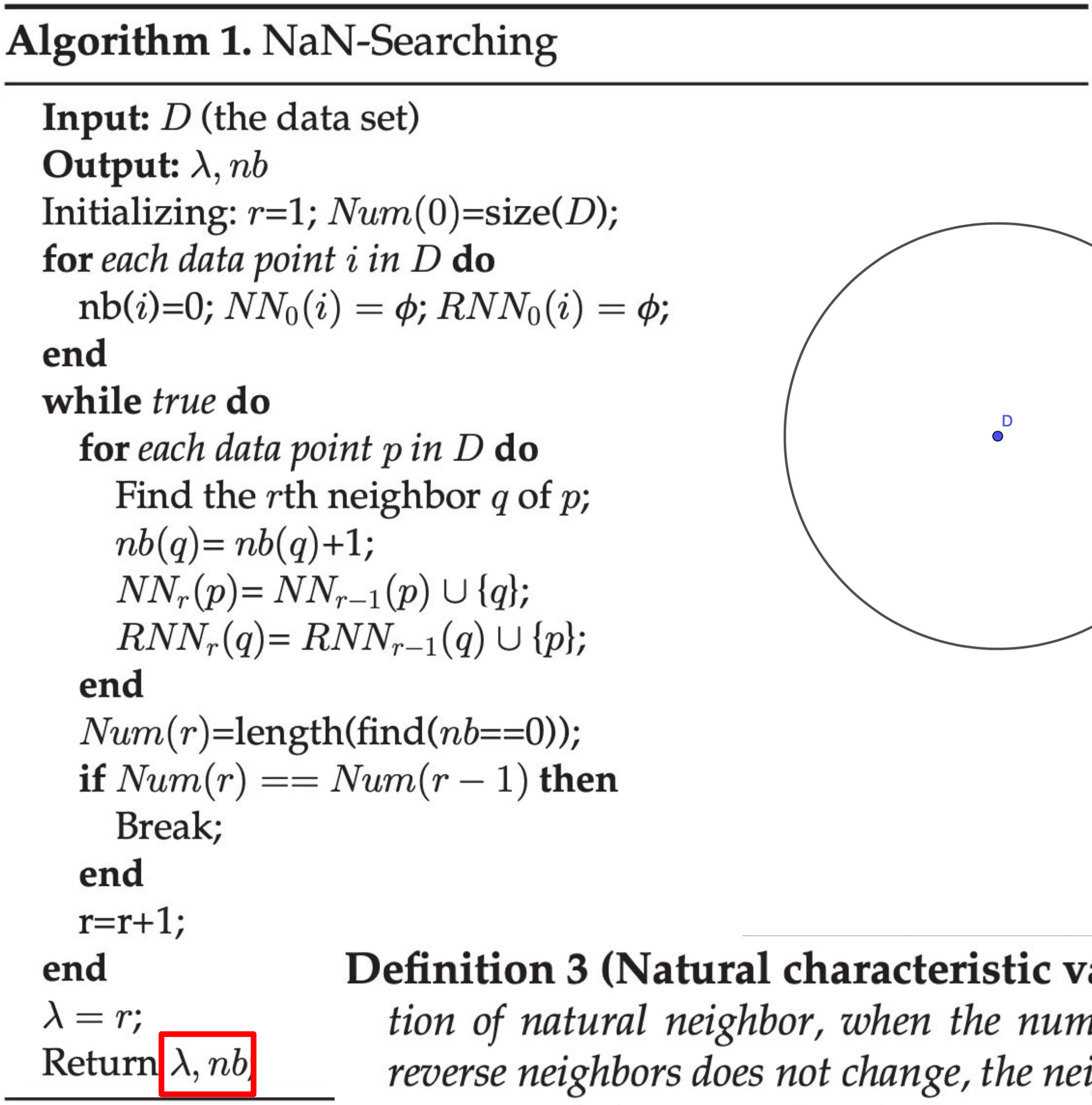
Algorithm 1. NaN-Searching

```
Input: $D$ (the data set)
Output: $\lambda, nb$
Initializing: $r$=1; $Num(0)$=size($D$);
for each data point $i$ in $D$ do
    nb($i$)=0; $NN_0(i) = \phi$; $RNN_0(i) = \phi$;
end
while true do
    for each data point $p$ in $D$ do
        Find the $r$th neighbor $q$ of $p$;
        $nb(q)= nb(q)+1$;
        $NN_r(p)= NN_{r-1}(p) \cup \{q\}$;
        $RNN_r(q)= RNN_{r-1}(q) \cup \{p\}$;
    end
    $Num(r)$=length(find($nb$==0));
    if $Num(r) == Num(r-1)$ then
        Break;
    end
    r=r+1;
end
$\lambda = r$;
Return $\boxed{\lambda, nb}$
```



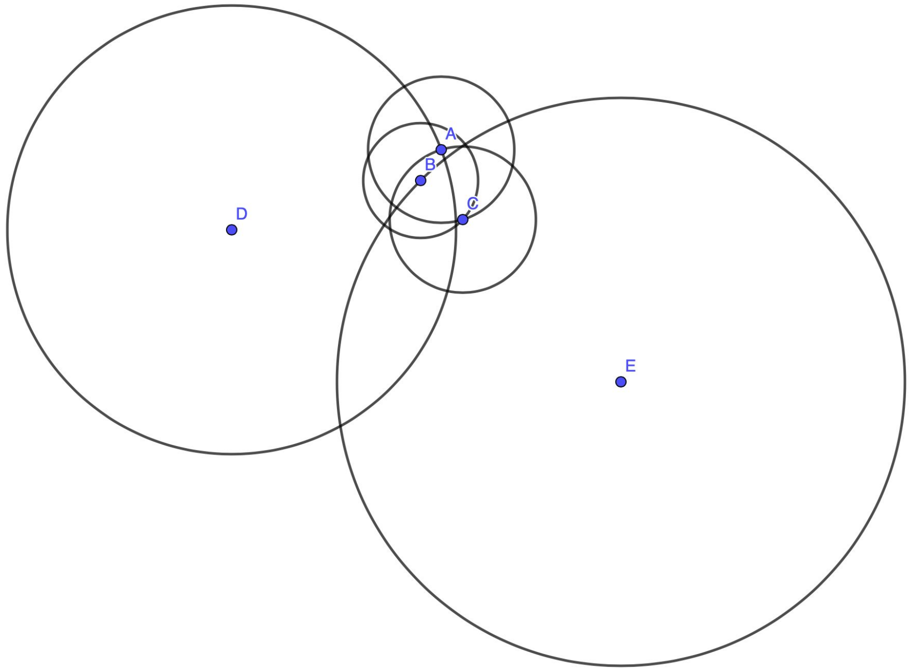
r=3	A	B	C	D	E
nb	3	4	3	0	0

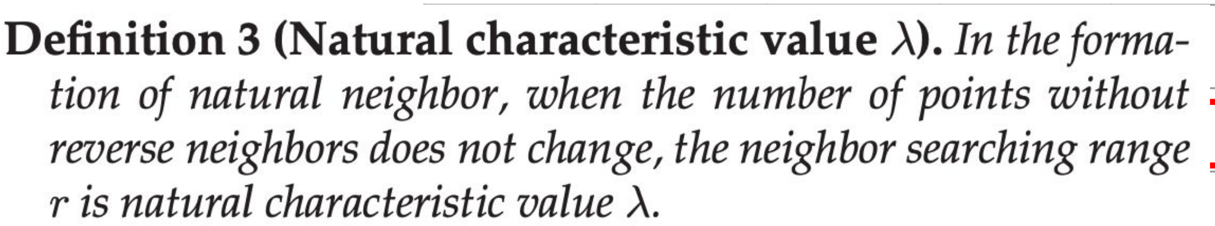
這一演算法的停止條件為，新一輪迭代得到的 nb中為0的個數與上一輪相等。



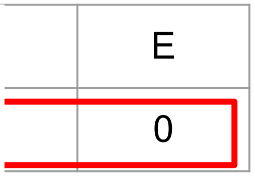
Algorithm 1. NaN-Searching

```
Input: $D$ (the data set)
Output: $\lambda, nb$
Initializing: $r$=1; $Num(0)$=size($D$);
for each data point $i$ in $D$ do
    nb($i$)=0; $NN_0(i) = \phi$; $RNN_0(i) = \phi$;
end
while true do
    for each data point $p$ in $D$ do
        Find the $r$th neighbor $q$ of $p$;
        $nb(q) = nb(q)+1$;
        $NN_r(p) = NN_{r-1}(p) \cup \{q\}$;
        $RNN_r(q) = RNN_{r-1}(q) \cup \{p\}$;
    end
    $Num(r)$=length(find($nb$==0));
    if $Num(r) == Num(r-1)$ then
        Break;
    end
    r=r+1;
end
$\lambda = r$;
Return $\boxed{\lambda, nb}$
```





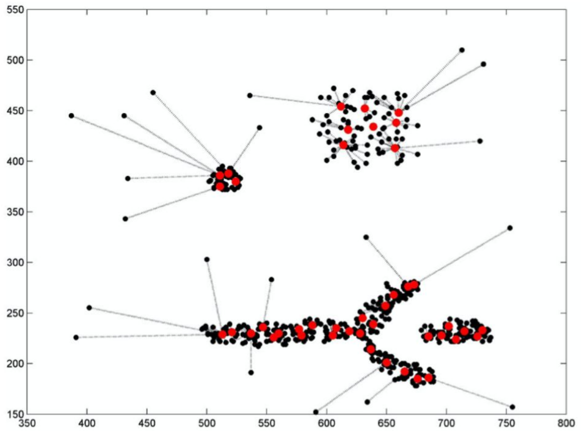
Definition 3 (Natural characteristic value λ). *In the formation of natural neighbor, when the number of points without reverse neighbors does not change, the neighbor searching range r is natural characteristic value λ.*

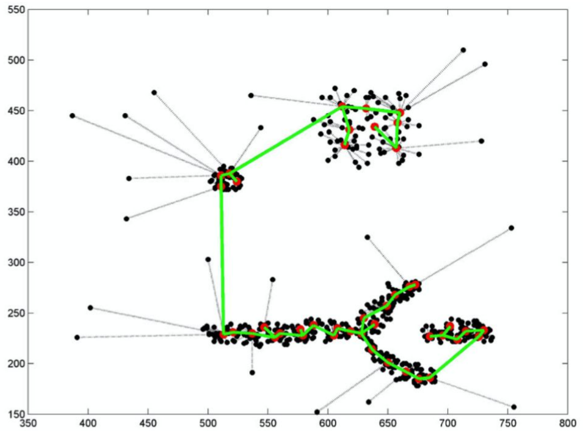


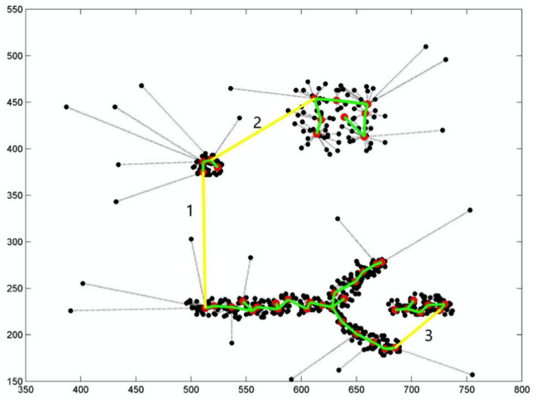
	E
	$\boxed{0}$

迭代停止時的r也被稱作nature characteristic value。

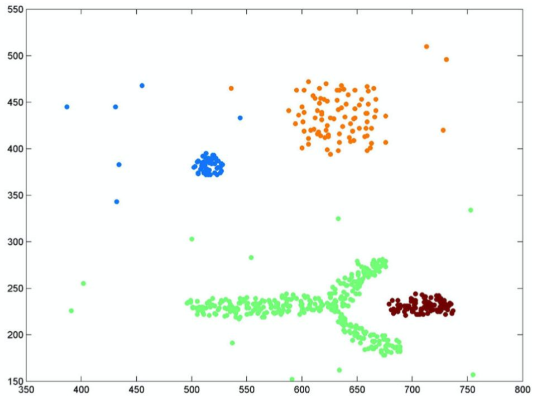
Main Idea of LDP-MST


(a)


(b)



(c)


(d)

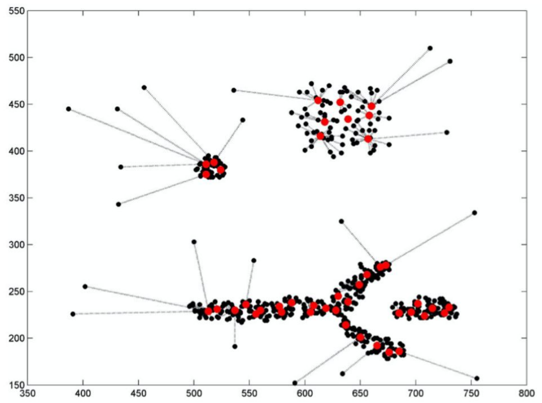
此為本文提出方法的概要，大致分為三步：(a)使用local density peaks作為代表，見圖中紅點。(b)依照一種新定義的距離為代表們構造 MST，見圖中綠線。(c)從最長的edge開始切割，直至得到目標數量的 clusters，見圖中黃線。

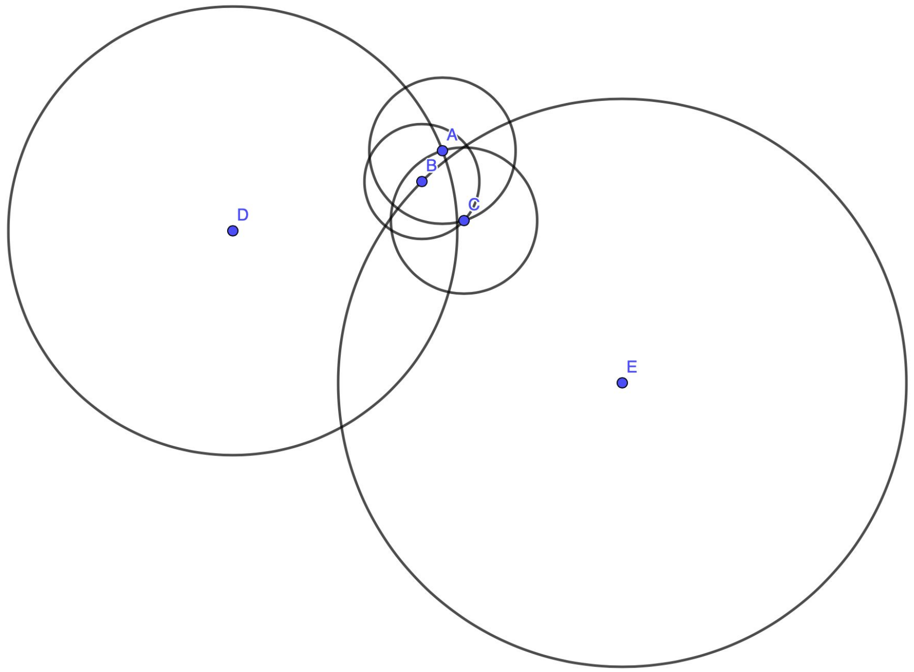
Local Density Peaks

Local density

$$\rho(p) = \frac{nb(p)}{\sum_{q \in NN_k(p)} d(p,q)}$$

Where *k*=max(*nb*).

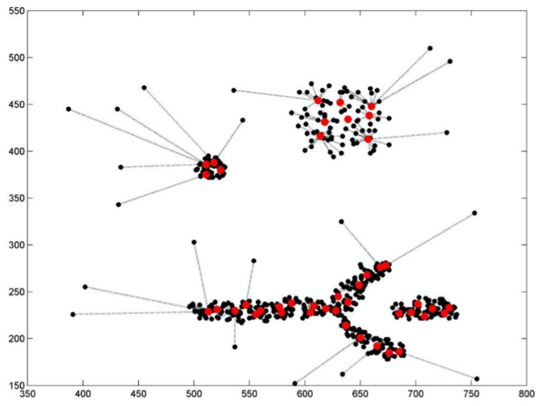




r=3	A	B	C	D	E
nb	3	$\boxed{4}$	3	0	0

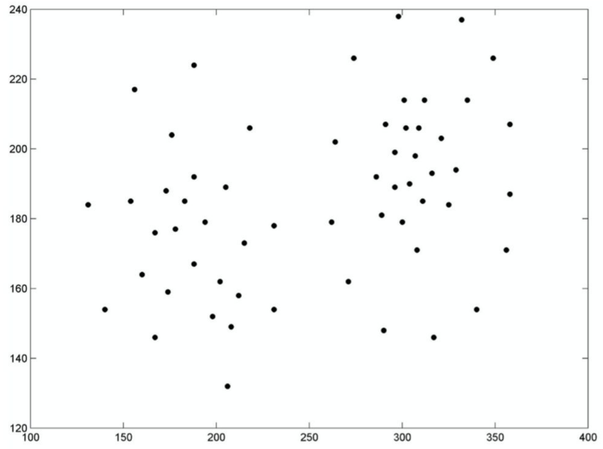
在選代表時主要參考每個點的 local density。其中k為由Algorithm 1獲得的nb決定。以之前的例子為例，k應為4。

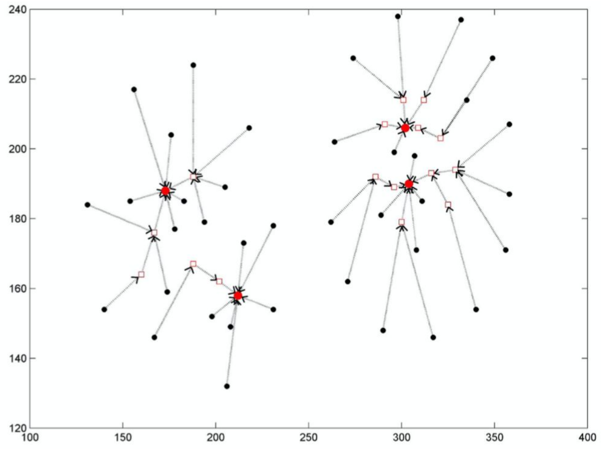
Local Density Peaks


(a)

Definition 4 (Representative). *If the local density of point q is maximum among p and the k nearest neighbors of p, then q is the representative of p, denoted as Rep(p) = q.*

Definition 5 (Local Density Peak, LDP). *A point p is a local density peak if Rep(p) = p.*

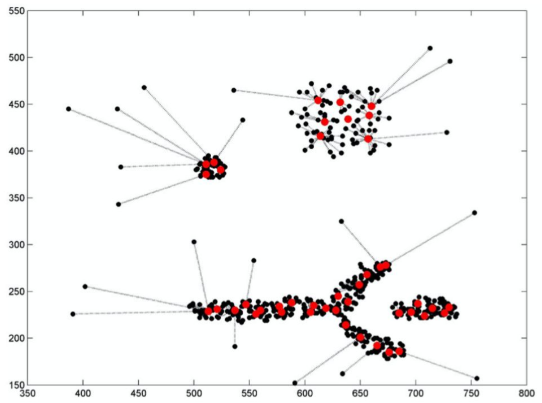




Acyclic

融合local density和kNN的概念，使每個點都找到自己的代表。能代表自己的點即為 LDP。

Local Density Peaks

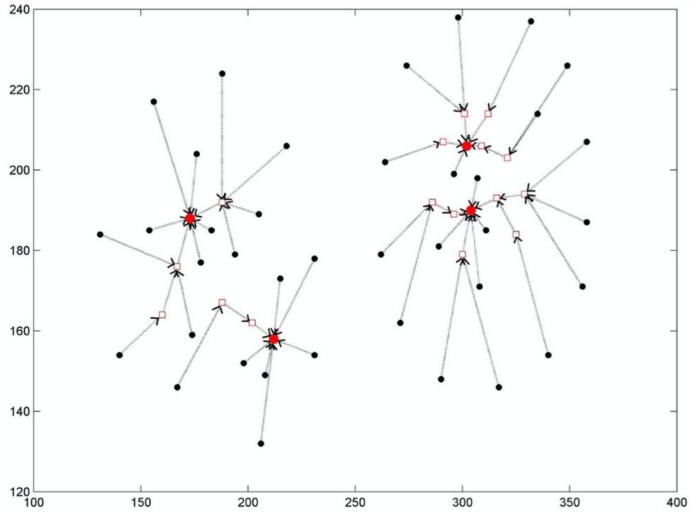


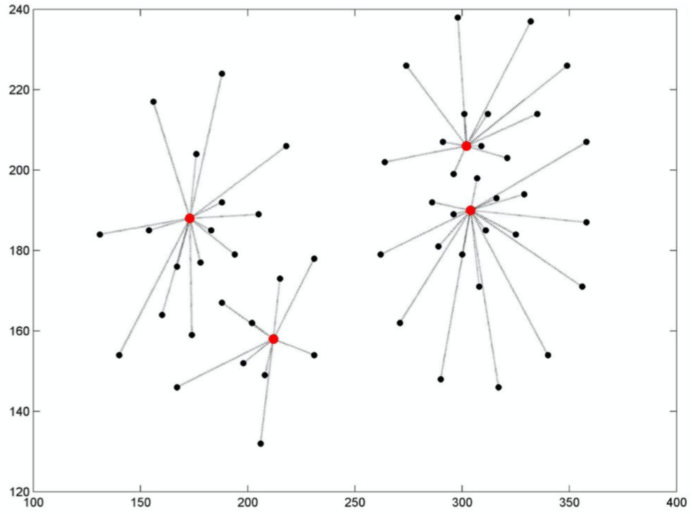
(a)

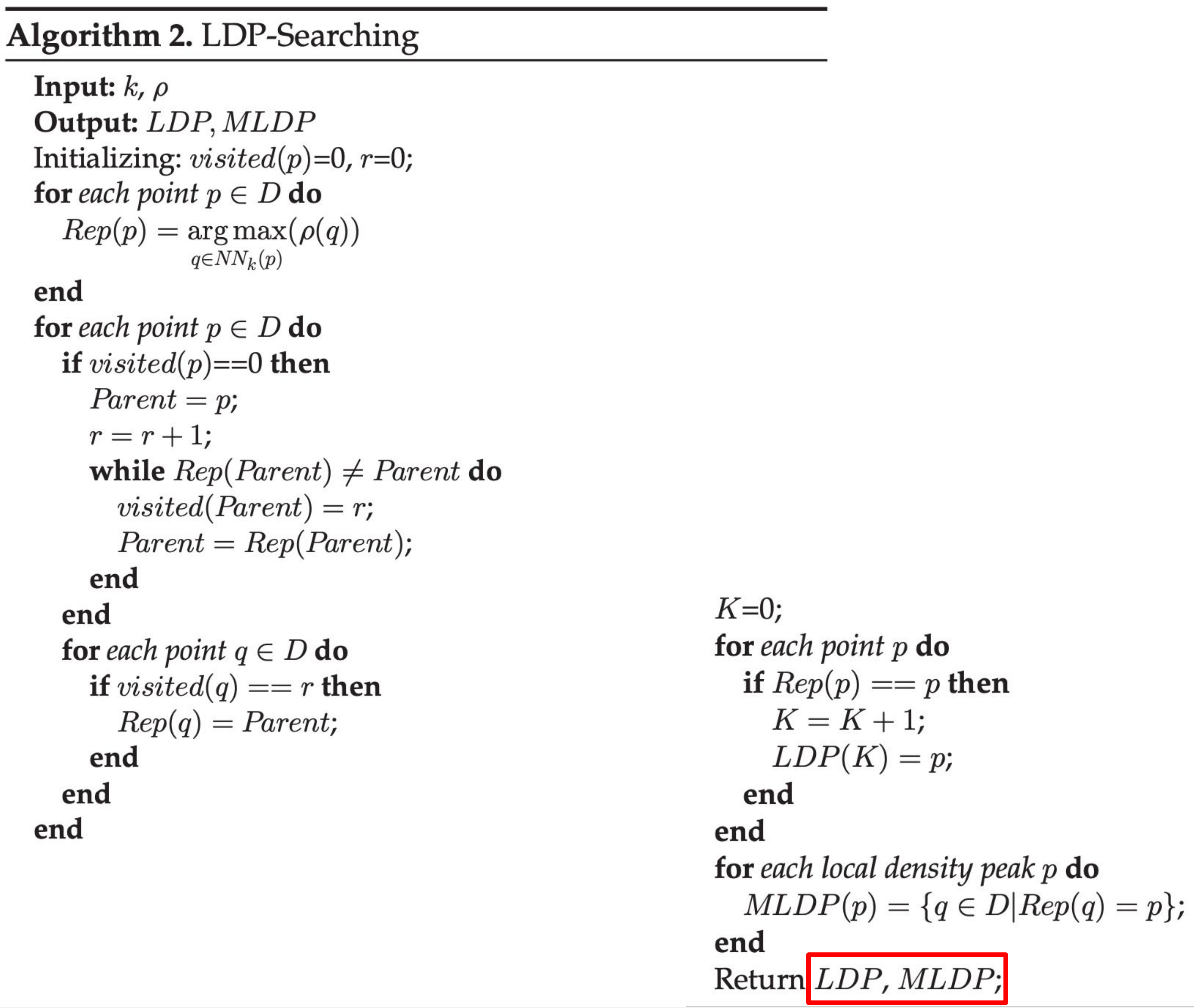
Definition 6 (Representative Transfer Rule, RTR). *If $Rep(p) = q$ and $Rep(q) = r$, then $Rep(p) = r$.*

Definition 7 (Members of Local Density Peak, MLDP). *For a local density peak p, its members are points whose representatives are p. Formally, it is defined as:*

$$MLDP(p) = \{q \in D | Rep(q) = p\}. \quad (2)$$







```
Algorithm 2. LDP-Searching
Input: $k$, $\rho$
Output: $LDP$, $MLDP$
Initializing: $visited(p)$=0, $r$=0;
for each point $p \in D$ do
    $Rep(p) = \arg\max_{q \in NN_k(p)}(\rho(q))$
end
for each point $p \in D$ do
    if $visited(p)$==0 then
        $Parent = p$;
        $r = r + 1$;
        while $Rep(Parent) \neq Parent$ do
            $visited(Parent) = r$;
            $Parent = Rep(Parent)$;
        end
    end
    for each point $q \in D$ do
        if $visited(q) == r$ then
            $Rep(q) = Parent$;
        end
    end
end
$K$=0;
for each point $p$ do
    if $Rep(p) == p$ then
        $K = K + 1$;
        $LDP(K) = p$;
    end
end
for each local density peak $p$ do
    $MLDP(p) = \{q \in D | Rep(q) = p\}$;
end
Return $LDP$, $MLDP$;
```

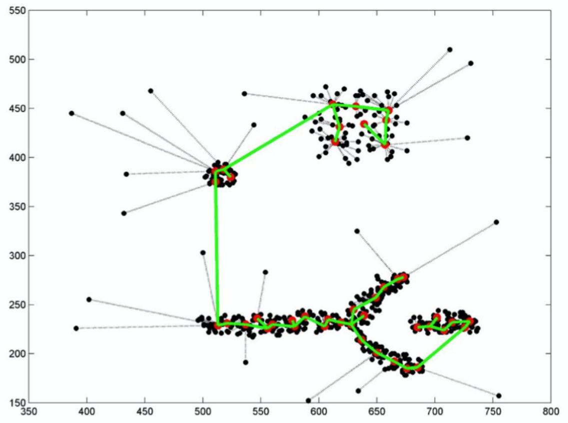
尋找LDP和MLDP的具體演算法。第一步完成。

Construct MST

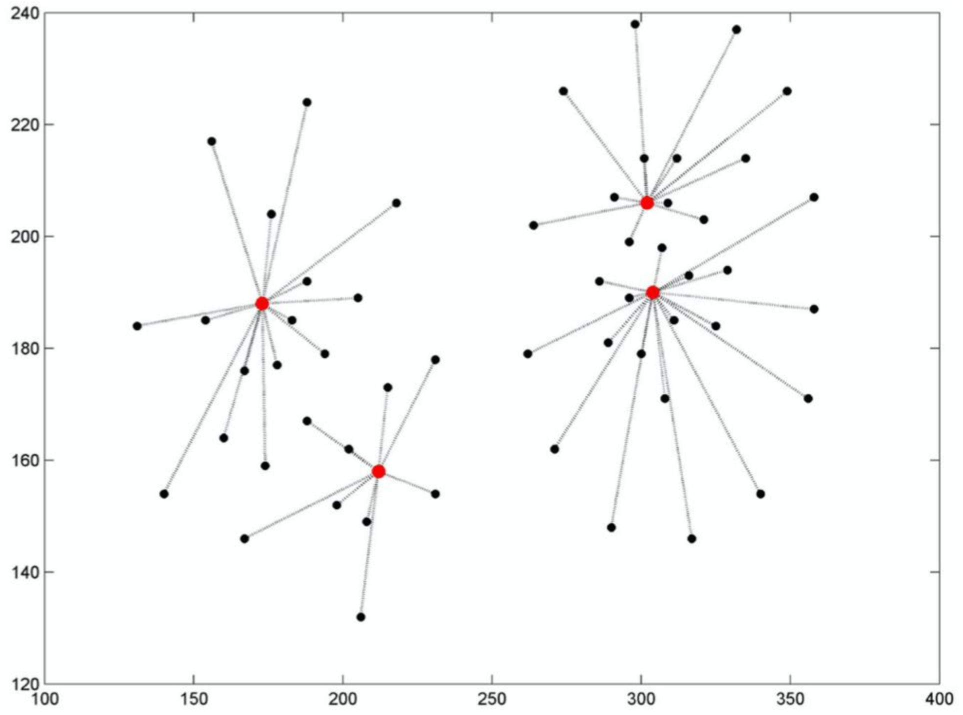
Euclidean distance?

Geodesic distance?

Shared neighbors-based distance!

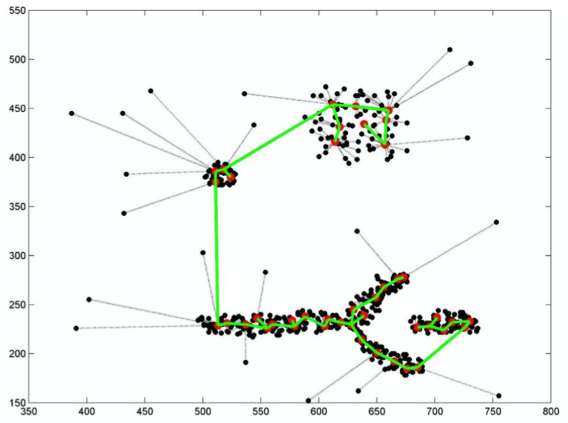


(b)



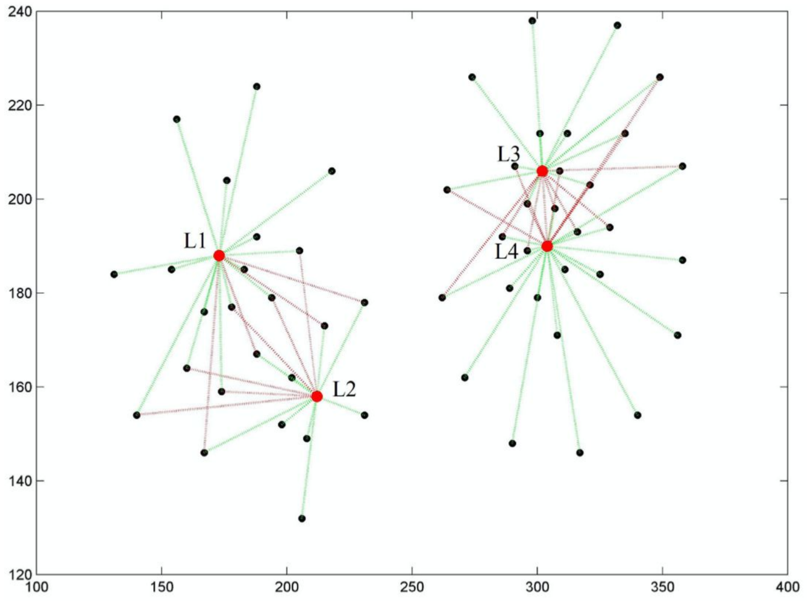
依據過往經驗，考慮到諸如 curse of dimensionality之類的問題，Euclidean distance並不是構造MST的好選擇。而理論上比較合適的Geodesic distance計算起來時間複雜度又過高。因此本文提出了一種新的定義距離的方法。

Construct MST

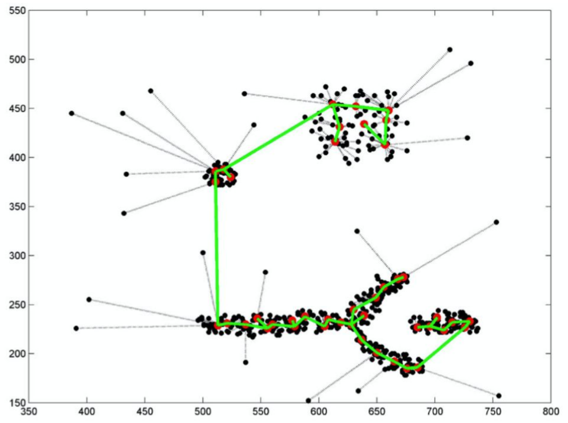

(b)

Definition 8 (Neighbors of Local Density Peaks, NLDP). *For a local density peak p, its neighbors are the union of λ nearest neighbors of all members of p, where λ is the natural characteristic value. It is defined as:*

$$NLDP(p) = \bigcup_{q \in MLDP(p)} NN_{\lambda}(q).$$

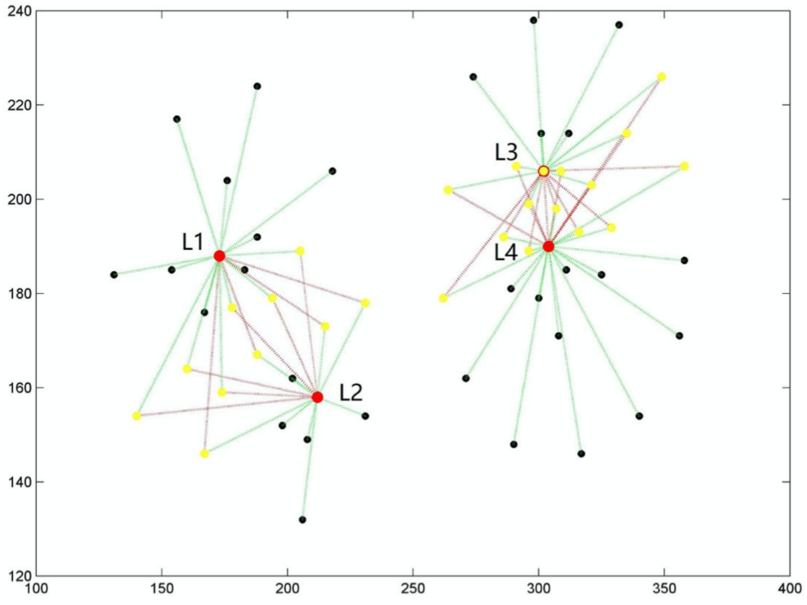


Construct MST

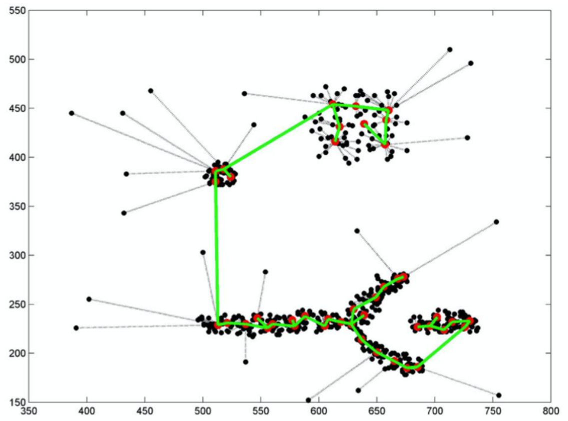

(b)

Definition 9 (Shared Neighbors between Two Local Density Peaks, SLDP). *Shared neighbors of two local density peaks p and q are the intersection of their neighbors. It is defined as:*

$$SLDP(p, q) = NLDP(p) \cap NLDP(q).$$



Construct MST

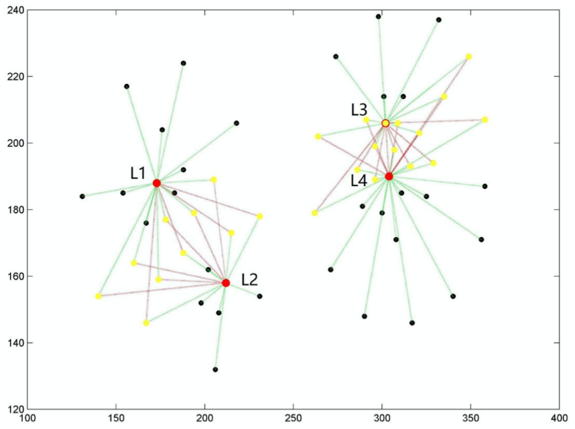


(b)

Definition 10 (Shared Neighbors-Based Distance between Local Density Peaks, SD). *For local density peaks p and q, their shared neighbors-based distance is computed as:*

$$SD(p,q) = \begin{cases} \frac{d(p,q)}{|SLDP(p,q)| \times \sum_{o \in SLDP(p,q)} \rho(o)}, & if |SLDP(p,q)| \neq 0 \\ maxd \times (1 + d(p,q)), & otherwise \end{cases} \quad (5)$$

where $d(p,q)$ is the distance between local density peaks p and q, $\rho(o)$ is the local density of point o and maxd is the maximum value of distance between all pairs of local density peaks.



SD根據兩個LDP間有無shared neighbors對照不同的計算方式。

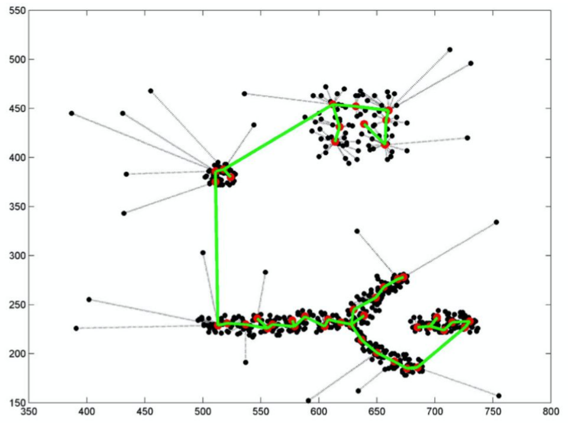
Construct MST

TABLE 1
The euclidean Distance between Local Density Peaks

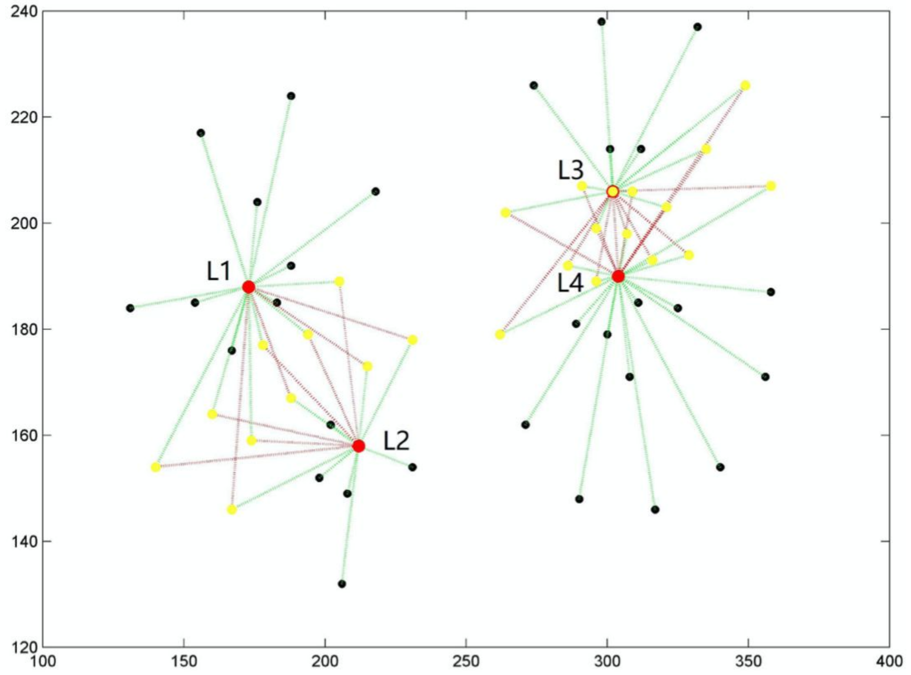
	L1	L2	L3	L4
L1	0	49	130	131
L2	49	0	102	97
L3	130	102	0	16
L4	131	97	16	0

TABLE 2
The SD between Local Density Peaks

	L1	L2	L3	L4
L1	0	17	17196	17296
L2	17	0	13495	12893
L3	17196	13495	0	2
L4	17296	12893	2	0

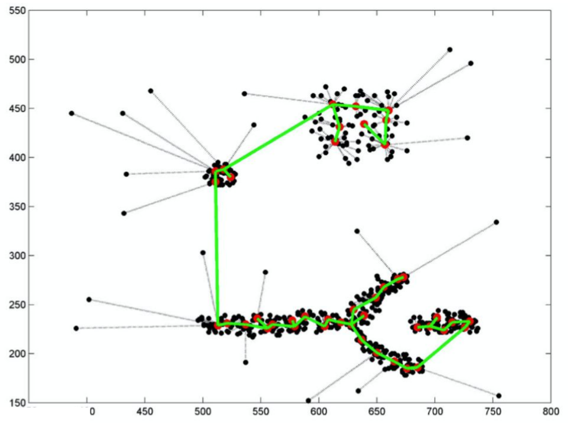


(b)

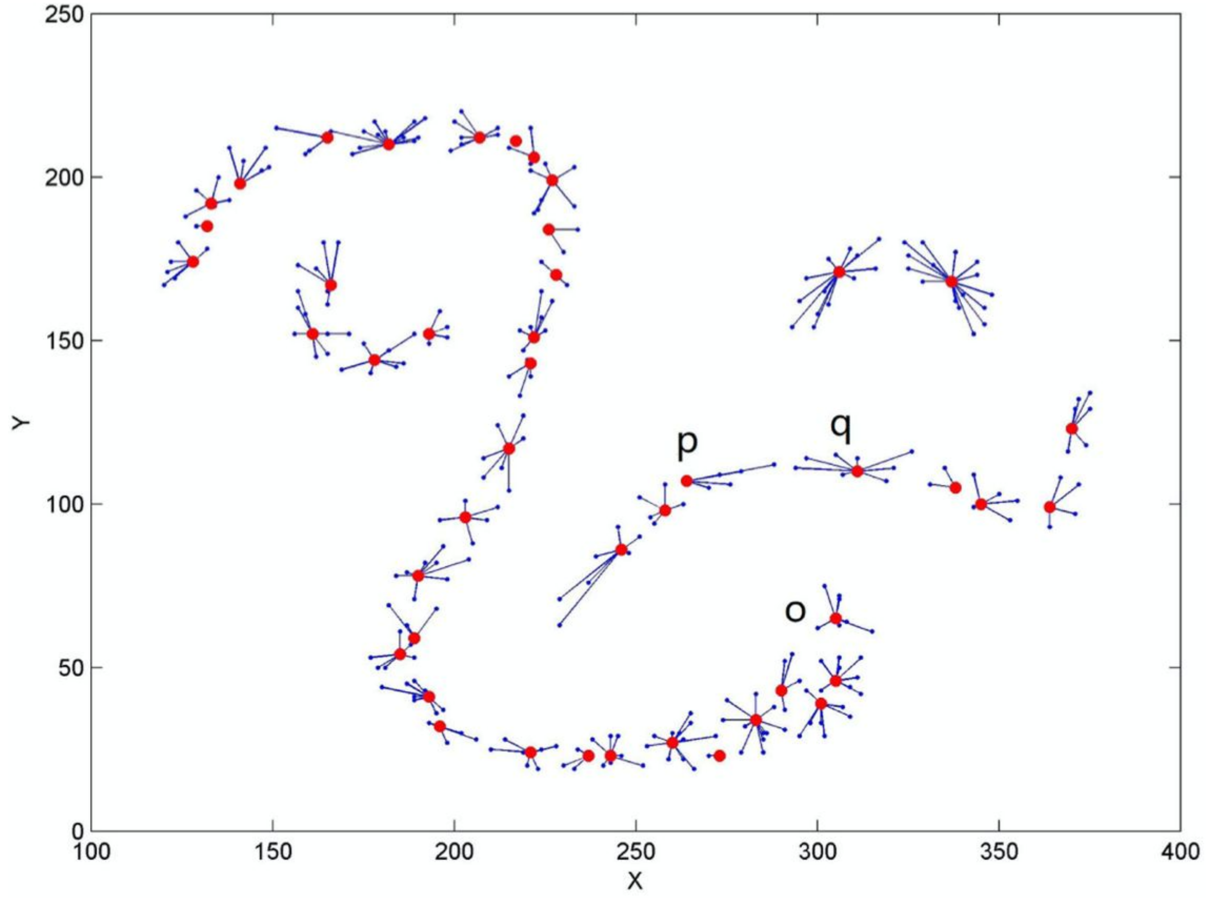


照此方法，若兩個LDP間的關係緊密，則距離會被相對縮小，若關係疏遠，則距離會被拉大。

Construct MST

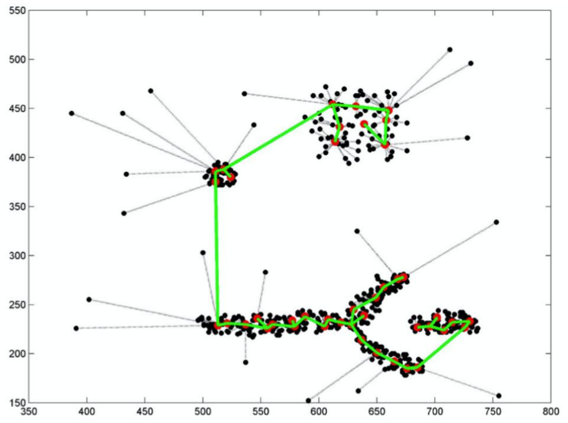


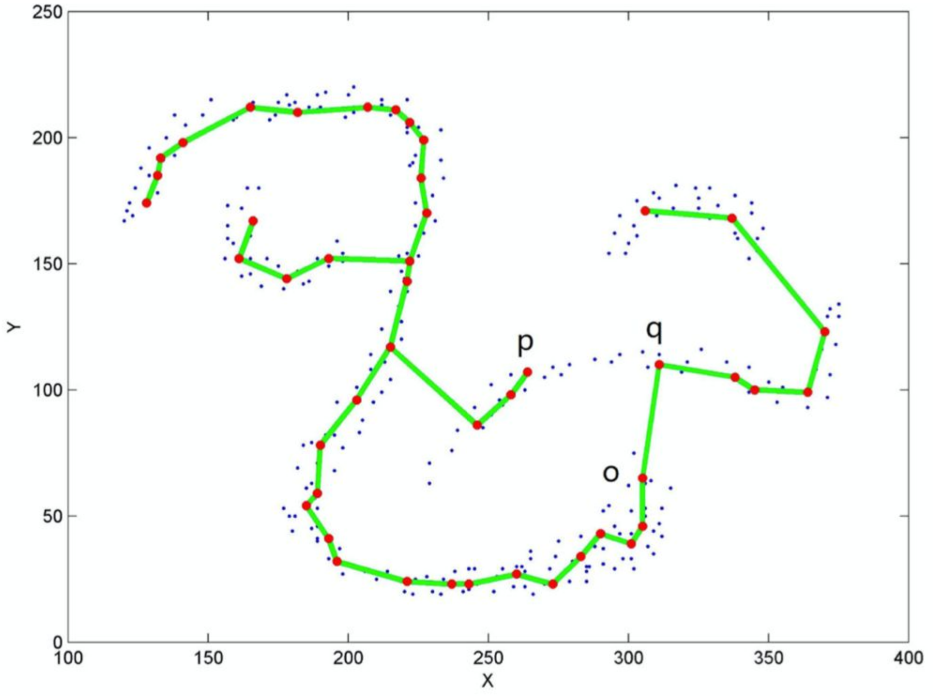
(b)



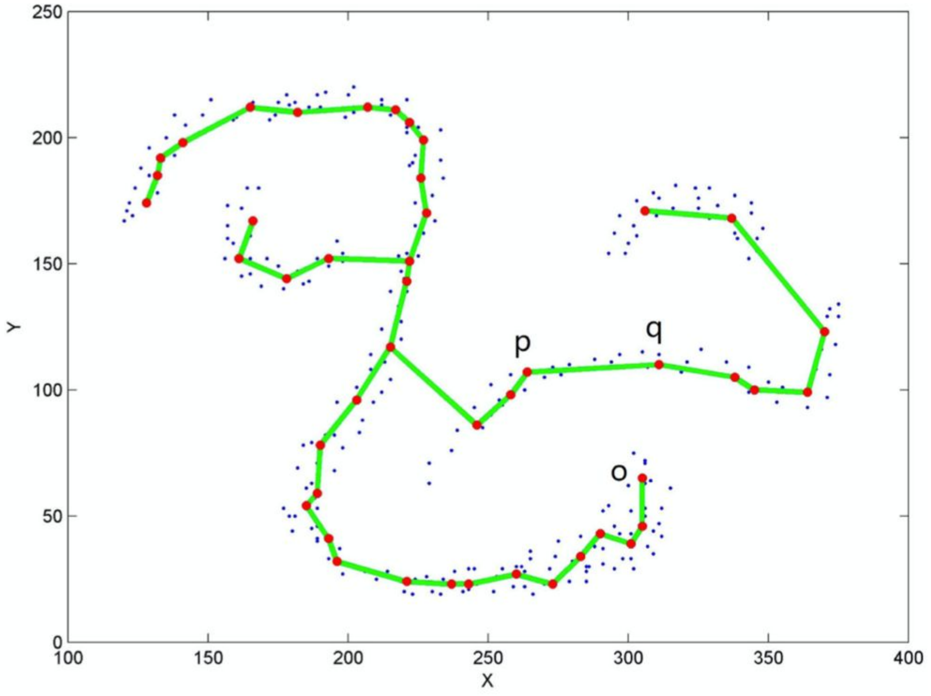
以這個data set為例，主要關注p，q和o這三個LDP根據不同的distance構造出MST的結果。

Construct MST


(b)



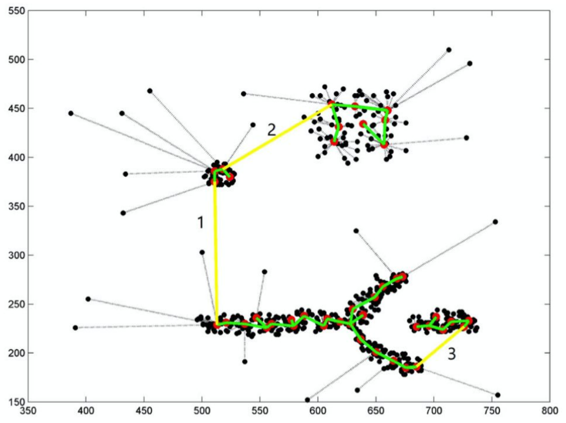
Euclidean distance.



Shared neighbors-based distance (SD).

可以看到用Euclidean distance構造的MST不符合我們對這一data set的預期。而SD表現正常。第二步完成。

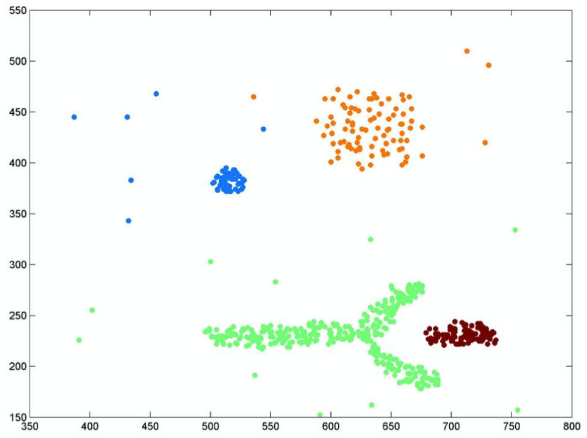
Clustering Based on LDP and MST



(c)

Repeatedly cut the longest edge (SD).

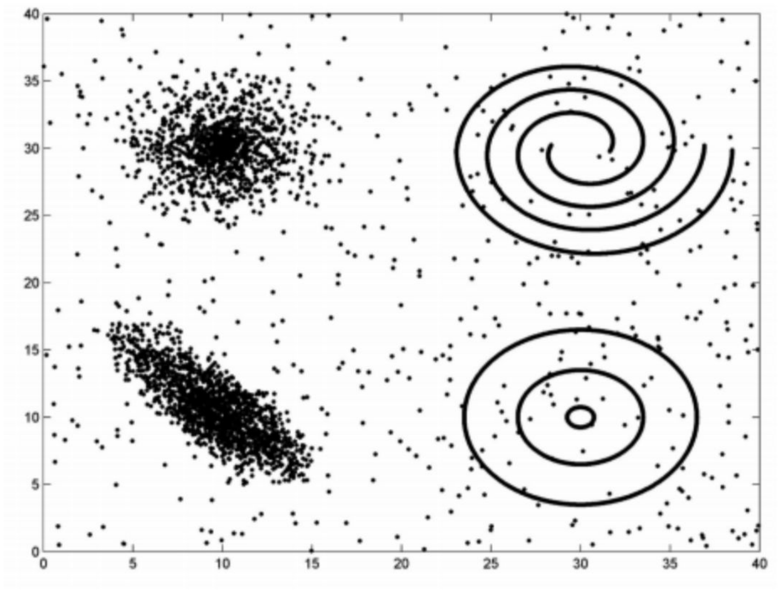
Make sure the sizes of clusters are large than a loosely estimated minimum number of points (*MinSize*).

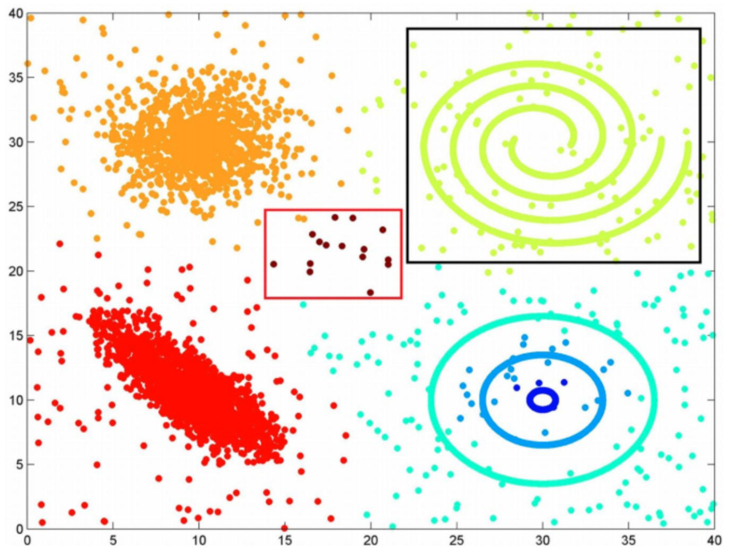


(d)

Minsize主要用於避免將距離相對較近的 outliers當做一個單獨的cluster。

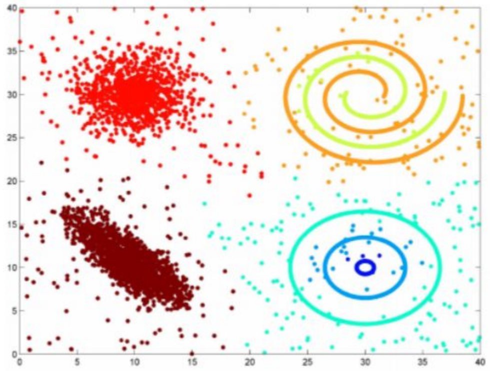
MinSize

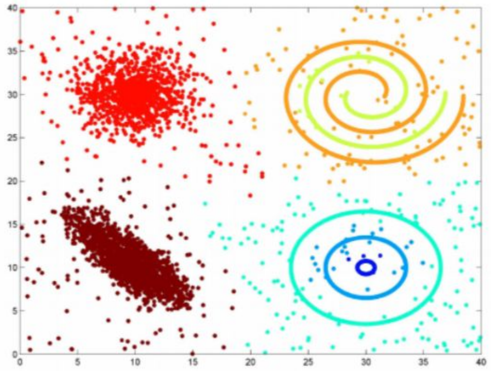

Dataset 3.

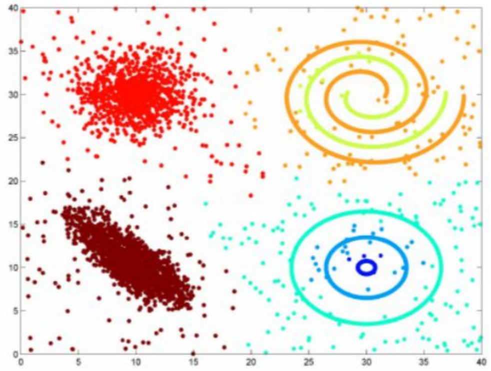

LDP-MST without setting MinSize.

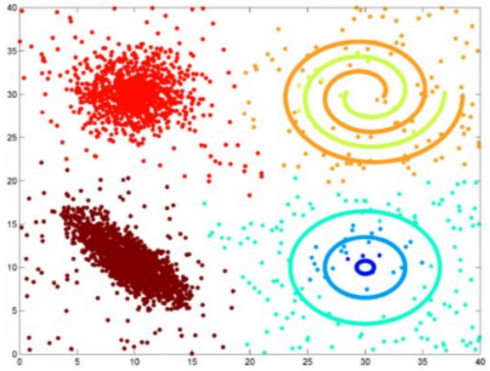
若data set當中沒有outlier或只有很少的outlier，則是否設定MinSize都不會影響結果。不然，以 Dataset 3為例，不設定MinSize導致結果中有一個純 outlier組成的cluster（右圖紅框），而右圖黑框的兩個 cluster沒有區分開。

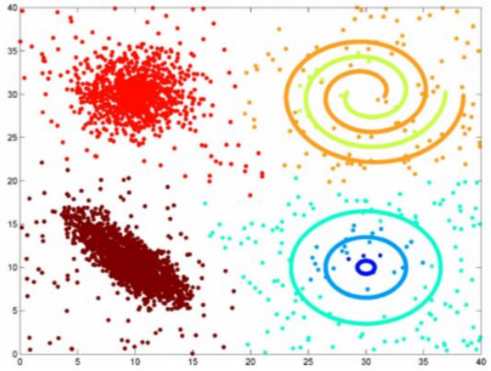
MinSize

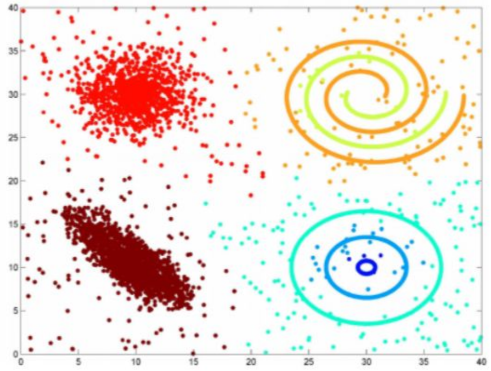

(a) 0.010


(b) 0.012

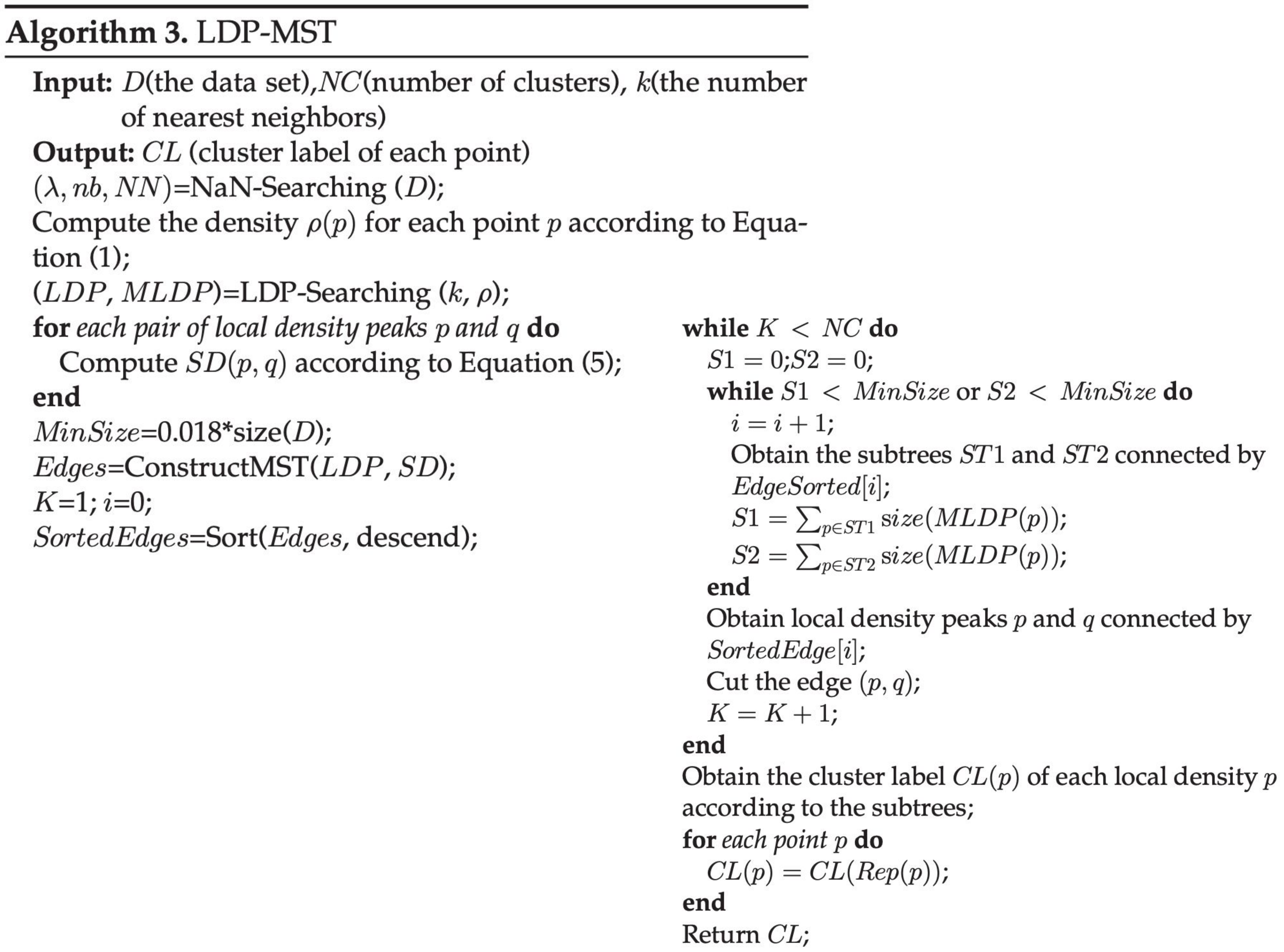

(c) 0.014


(d) 0.016


(e) 0.018


(f) 0.020

若能使MinSize大於outlier的比例並且小於最小 cluster的比例(理想狀態)，MinSize的值在小範圍內改動並不會改變clustering的結果。



Algorithm 3. LDP-MST

```
Input: $D$(the data set),$NC$(number of clusters), $k$(the number
       of nearest neighbors)
Output: $CL$ (cluster label of each point)
($\lambda, nb, NN$)=NaN-Searching ($D$);
Compute the density $\rho(p)$ for each point $p$ according to Equation (1);
($LDP$, $MLDP$)=LDP-Searching ($k$, $\rho$);
for each pair of local density peaks $p$ and $q$ do
  Compute $SD(p, q)$ according to Equation (5);
end
$MinSize$=0.018*size($D$);
$Edges$=ConstructMST($LDP$, $SD$);
$K$=1; $i$=0;
$SortedEdges$=Sort($Edges$, descend);
while $K < NC$ do
  $S1 = 0;S2 = 0;$
  while $S1 < MinSize$ or $S2 < MinSize$ do
    $i = i + 1;$
    Obtain the subtrees $ST1$ and $ST2$ connected by
    $EdgeSorted[i]$;
    $S1 = \sum_{p \in ST1} size(MLDP(p))$;
    $S2 = \sum_{p \in ST2} size(MLDP(p))$;
  end
  Obtain local density peaks $p$ and $q$ connected by
  $SortedEdge[i]$;
  Cut the edge $(p, q)$;
  $K = K + 1;$
end
Obtain the cluster label $CL(p)$ of each local density $p$
according to the subtrees;
for each point $p$ do
  $CL(p) = CL(Rep(p))$;
end
Return $CL$;
```

獲取最終cluster label的具體方法。至此 LDP-MST完成。

Time Complexity $O(n\log n)$

Search local density peaks.

NaN-searching $O(n^2)$ -> $O(n\log n)$ by KD-tree

Calculating densities and searching LDPs $O(n)$

Compute the shared neighbors-based distance between LDPs.

$O(n+n_{ldp}^2)$ ($n_{ldp} << n$)

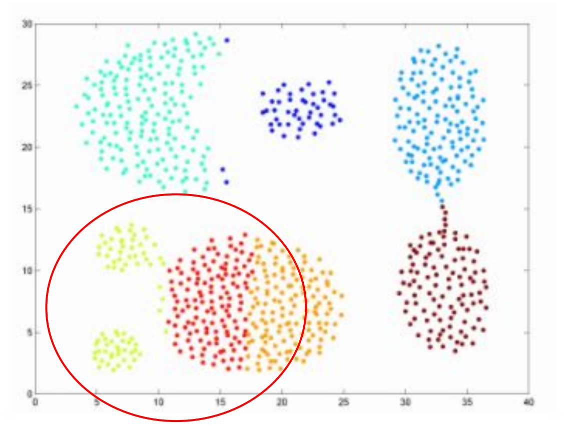
Employ the MST-based clustering algorithm to cluster the LDPs.

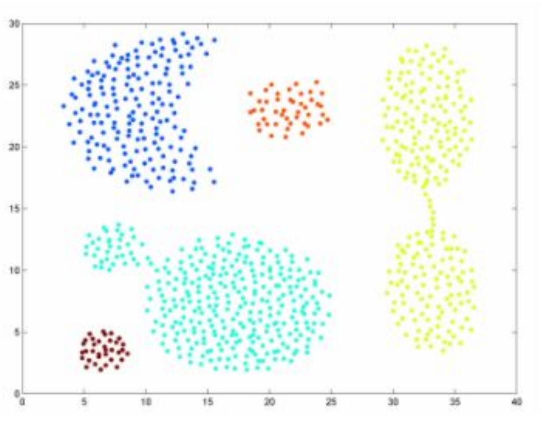
$O(n_{ldp}^2)$

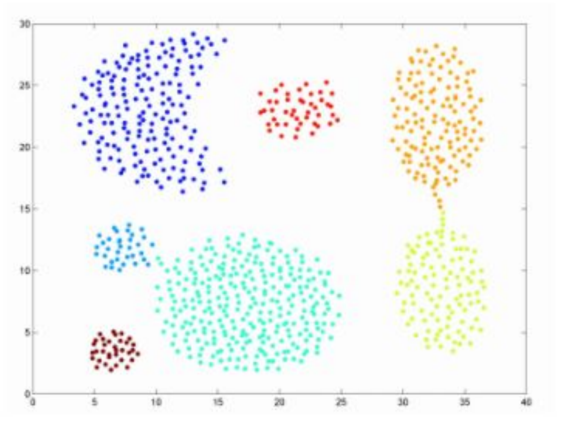
n_{ldp}為local density peaks的數量。

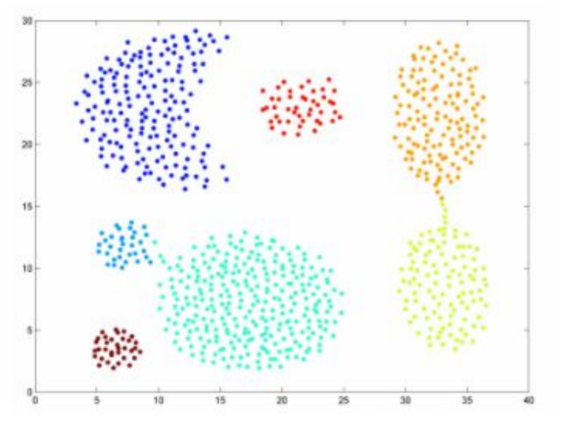
Experiments

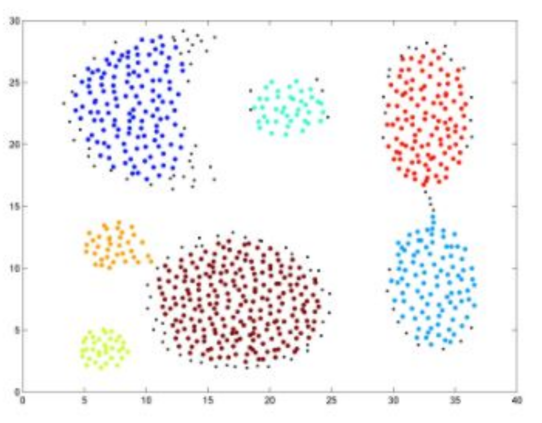
Clustering on Synthetic Datasets


(a) Kmeans


(b) DBSCAN


(c) DP


(d) Dcore


(e) LOF-MST

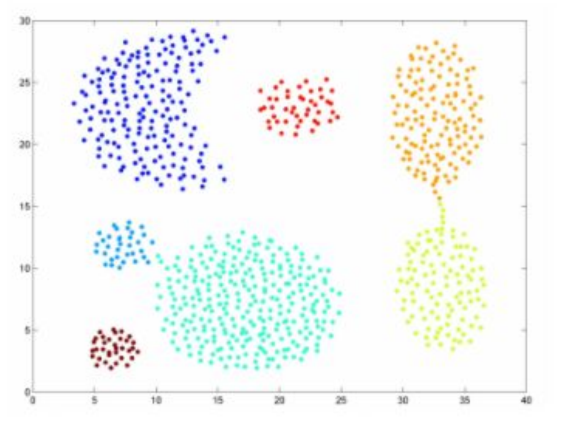
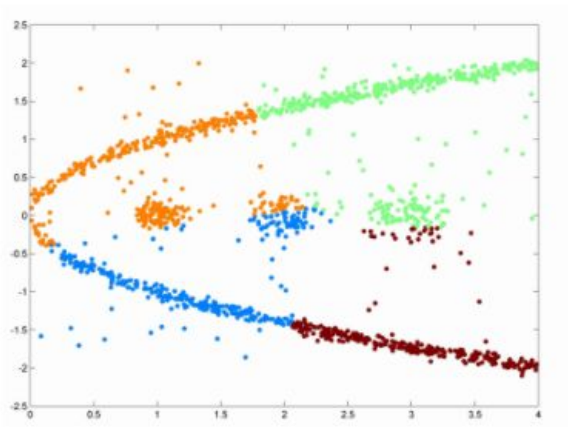
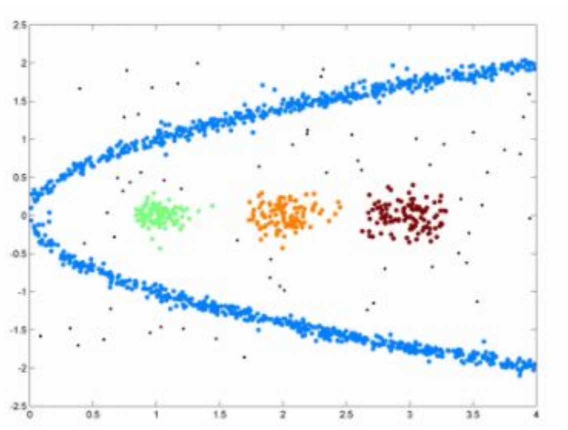
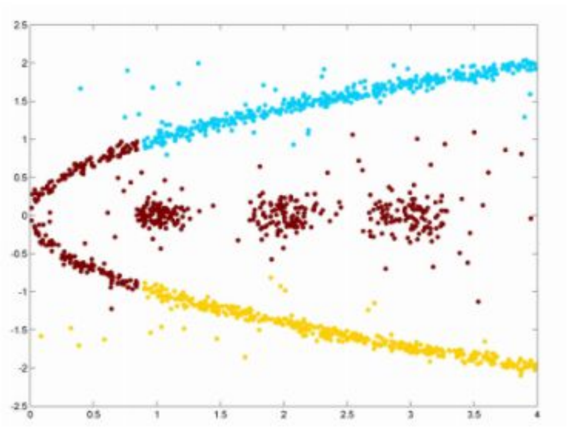

(f) LDP-MST

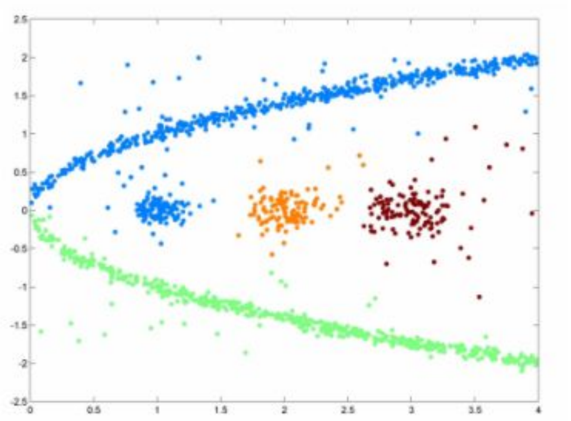
Figure 1: The clustering results on Dataset 1.

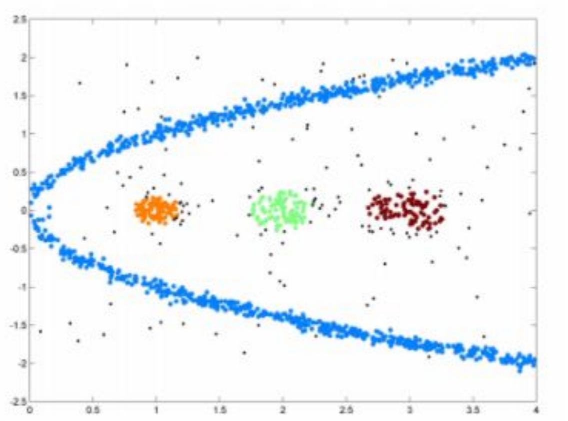
這是第一組六個演算法的分類結果。這個例子很符合我們對 clustering 問題的想像，其中 center-based 方法對這類情形相對容易解決。可以注意到 Kmeans 雖然快速容易實作，從結果來看卻有些不盡如人意 (紅圈)


(a) Kmeans


(b) DBSCAN


(c) DP


(d) Dcore


(e) LOF-MST

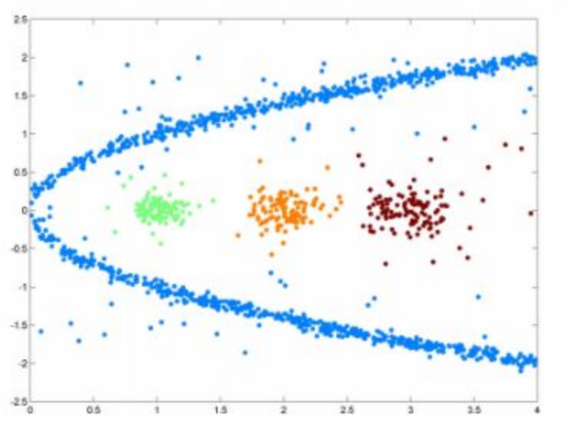
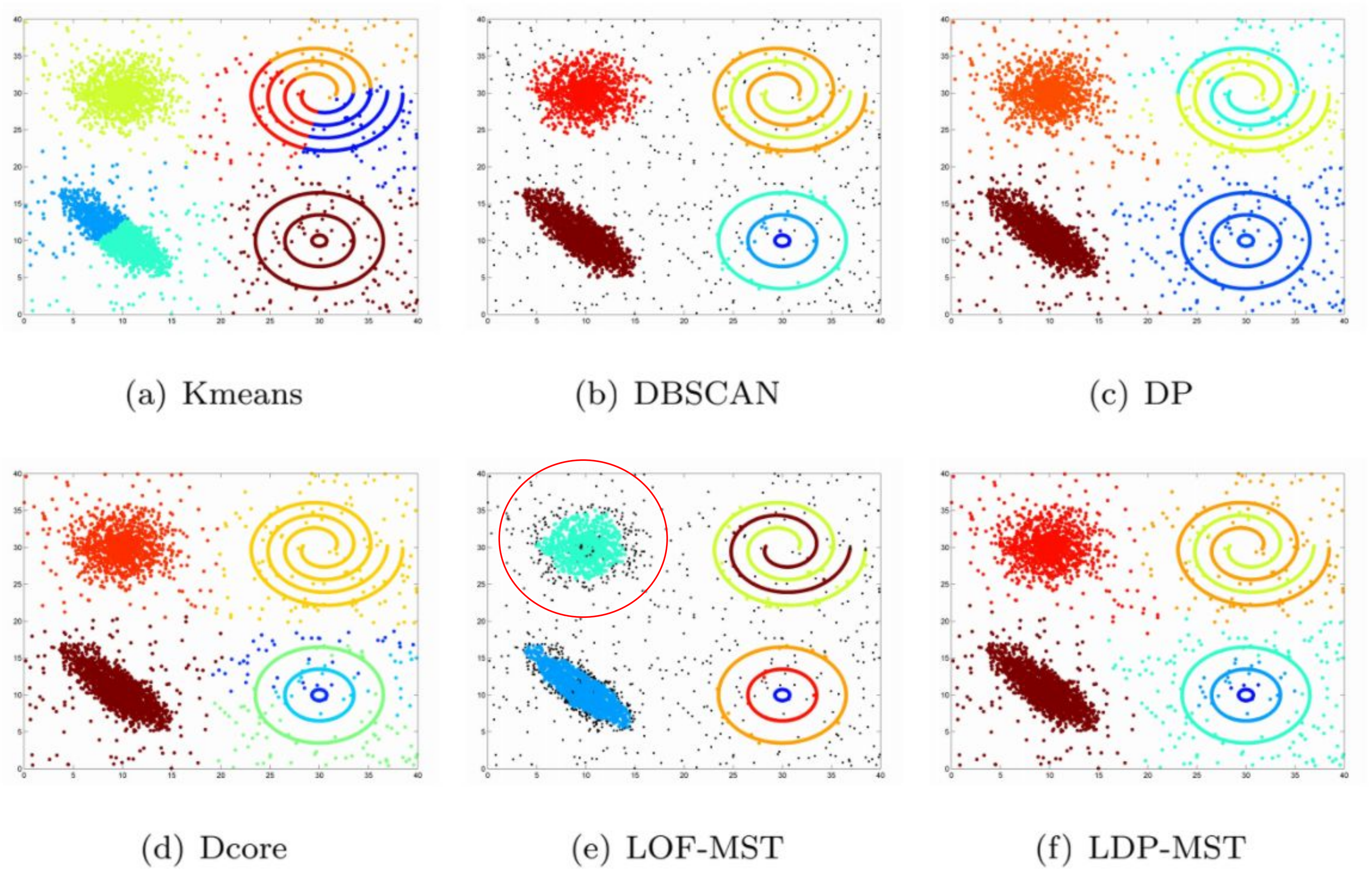

(f) LDP-MST

Figure 2: The clustering results on Dataset 2.

第二組資料和第一組呈現明顯的對比，可以看出 center-based 方法並不能識別出 拋物線是一個 cluster。DBSCAN, LOF-MST 和這篇論文的方法更符合直覺

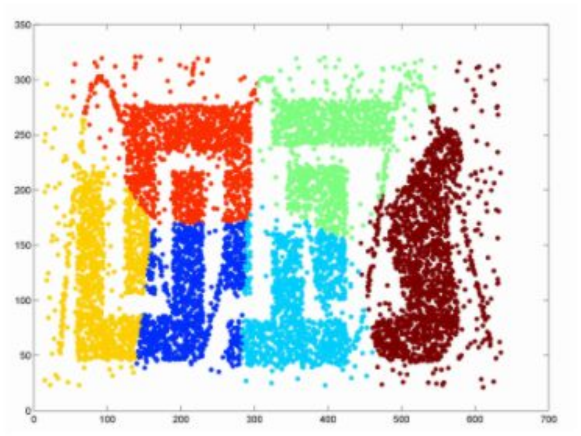


(a) Kmeans (b) DBSCAN (c) DP

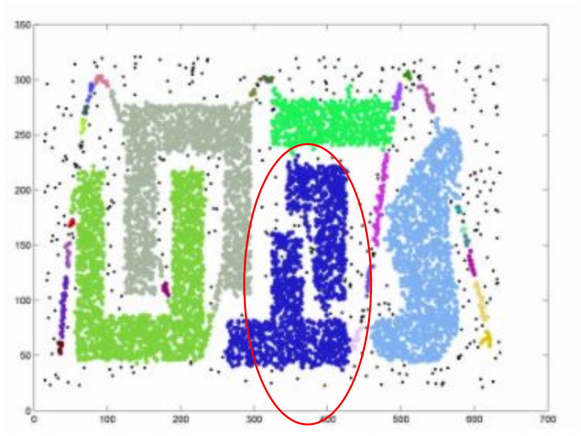
(d) Dcore (e) LOF-MST (f) LDP-MST

Figure 3: The clustering results on Dataset 3.

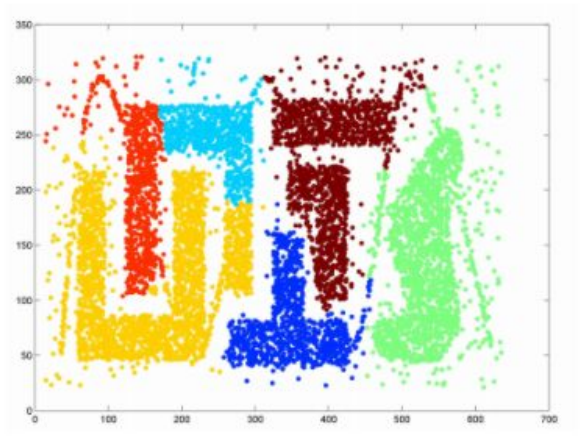
對大部分方法而言第三組的執行時間較長。這個例子中 DBSCAN 和 LDP-MST 表現都不錯。值得注意的是 LOF-MST 標示黑點代表雜訊，從紅圈可以看出 LOF-MST 傾向把正常的點認為是 outliers



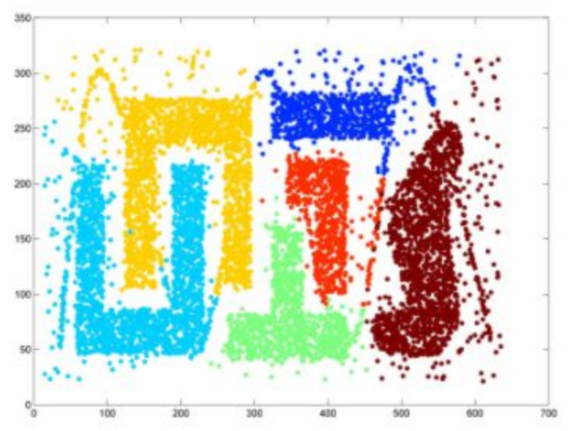
(a) Kmeans



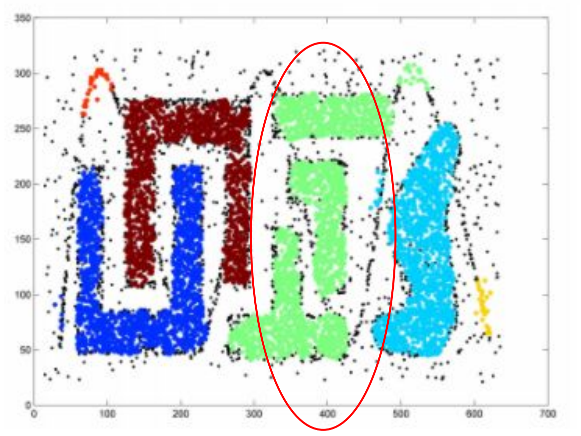
(b) DBSCAN



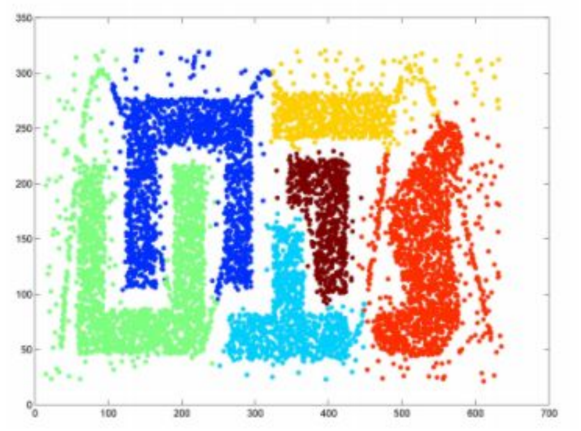
(c) DP



(d) Dcore



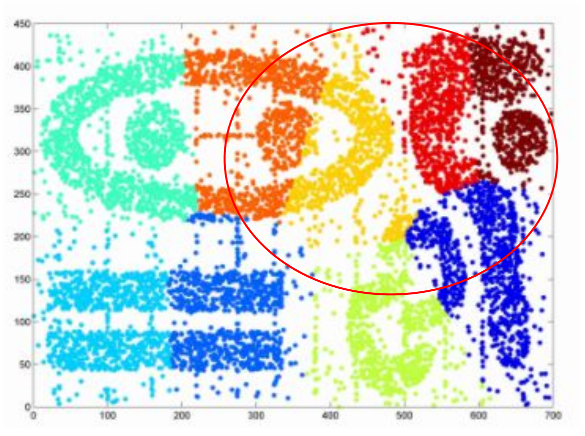
(e) LOF-MST

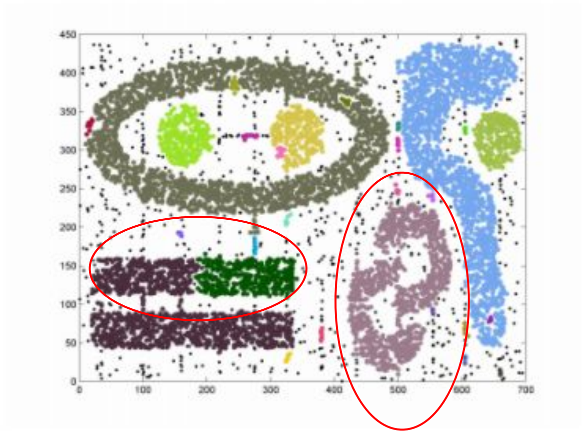


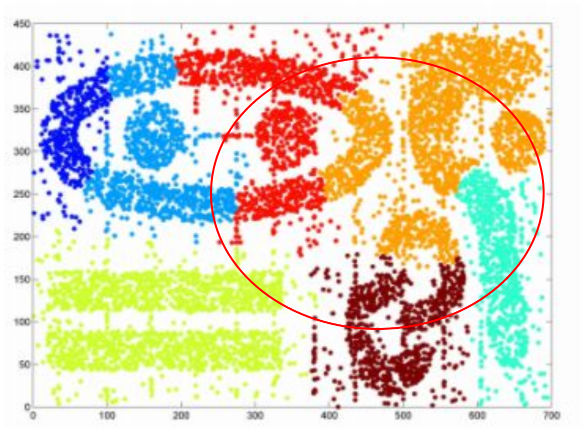
(f) LDP-MST

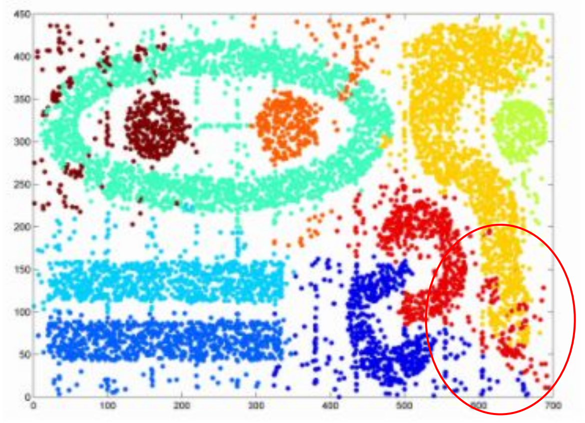
Figure 4: The clustering results on Dataset 4.

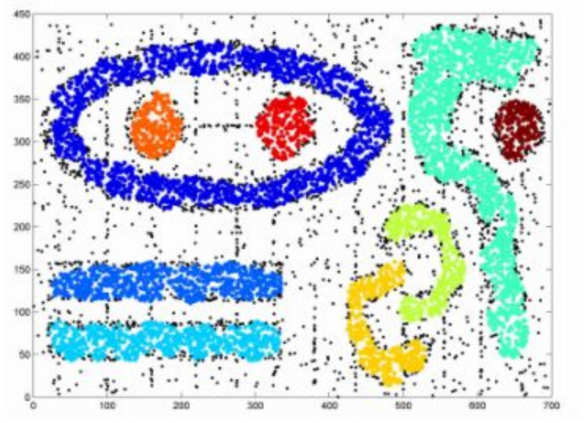
第四組資料中 DBSCAN, LOF-MST 分群結果都和 ground truth 有一些出入(紅圈)。LDP-MST, Dcore 在這組上表現不錯。同時可以看出 LDP-MST 在這組資料上相對優於同為 MST-Based 方法的 LOF-MST


(a) Kmeans


(b) DBSCAN


(c) DP


(d) Dcore


(e) LOF-MST

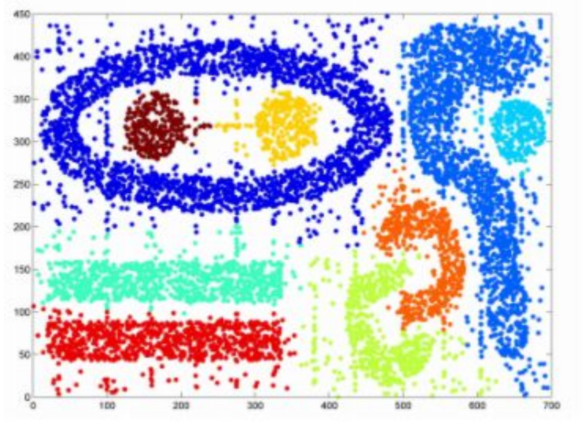
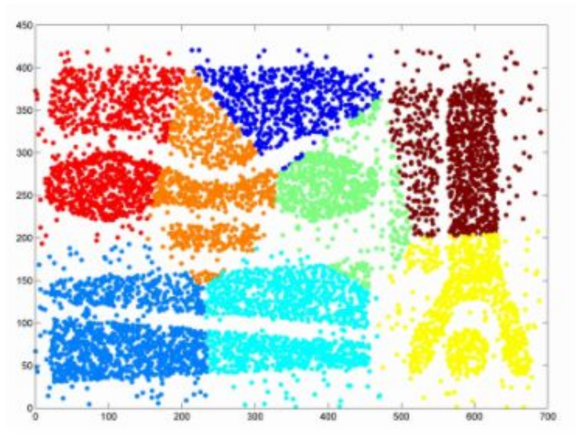
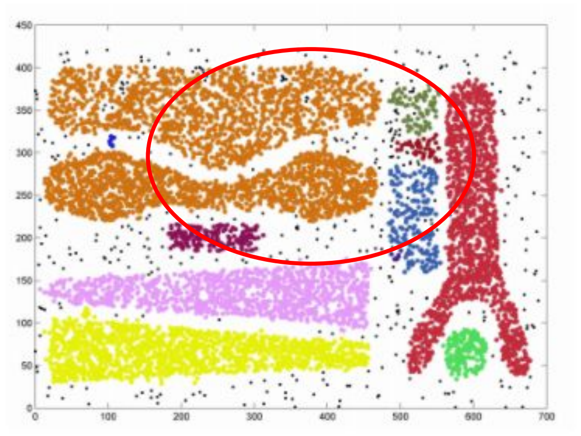

(f) LDP-MST

Figure 5: The clustering results on Dataset 5.

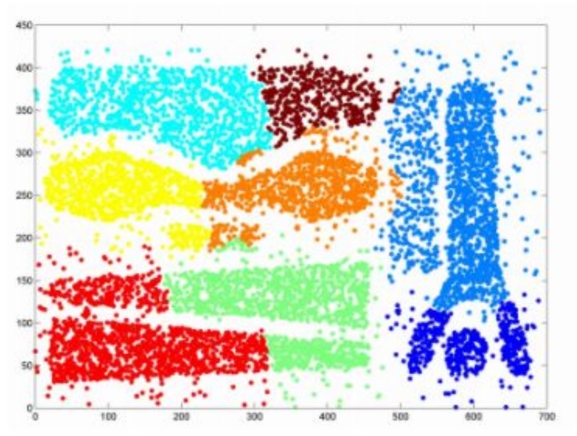
第五組資料凸顯 MST-Based 對於 clusters 多種不同形狀也能應對的優勢，其他方法都各自有一些奇怪的地方



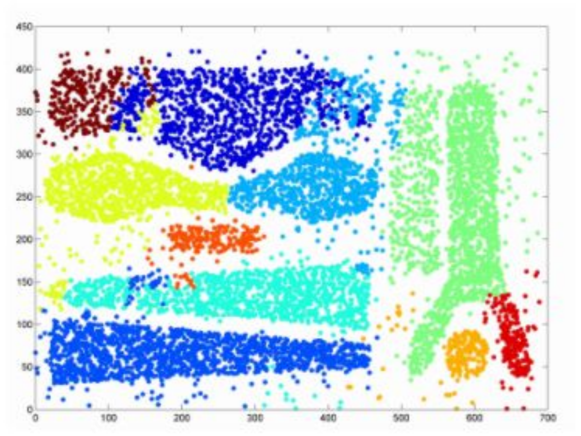
(a) Kmeans



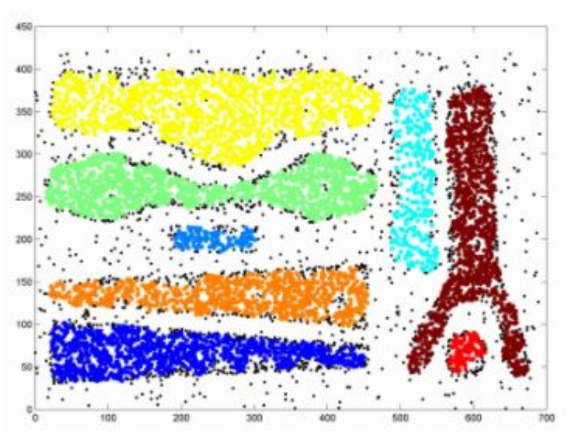
(b) DBSCAN



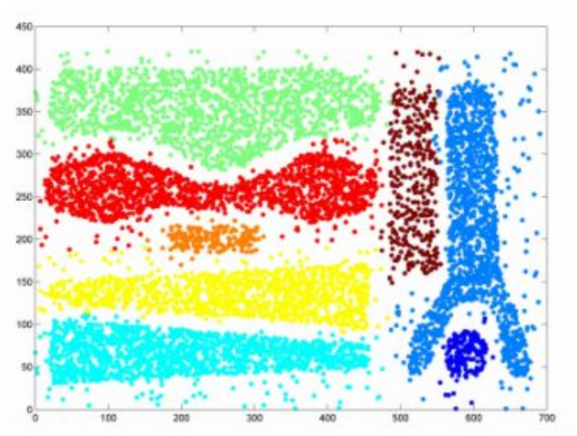
(c) DP



(d) Dcore



(e) LOF-MST



(f) LDP-MST

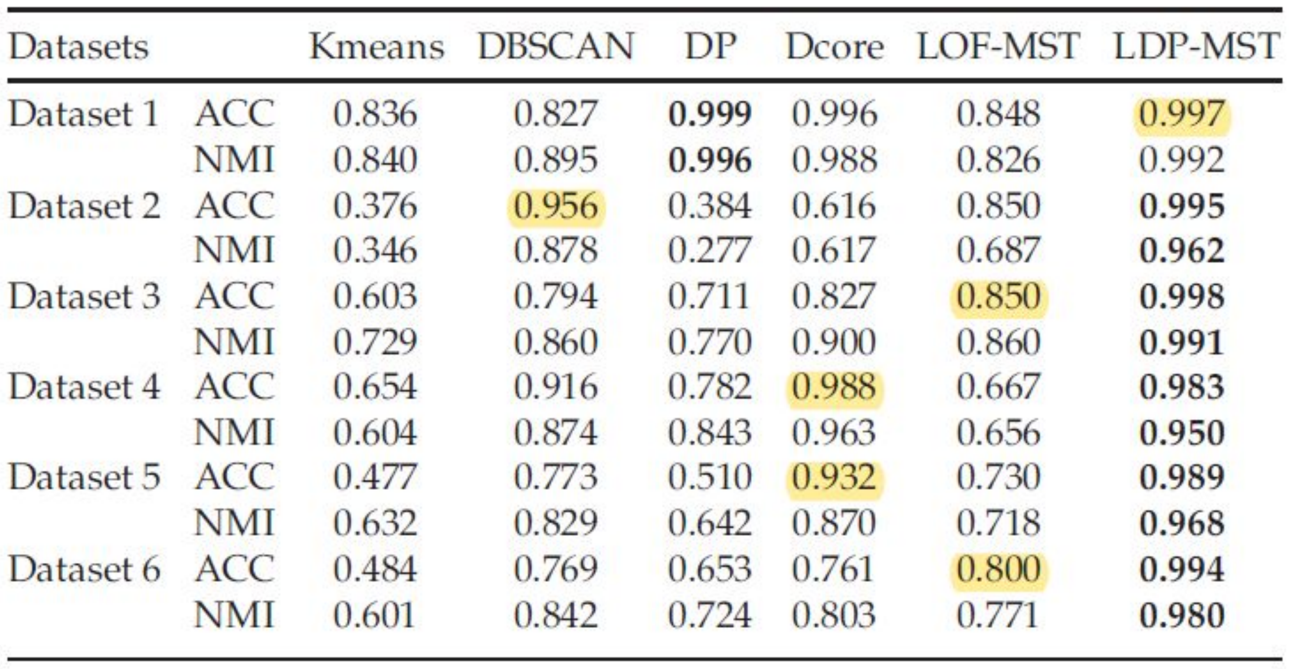
Figure 6: The clustering results on Dataset 6.

第六組資料不但群的形狀各異、各群的樣本數也相差較大，對一般分群方法來 說這種資料相對棘手。這組資料兩個 MST-Based 方法表現都不錯

Results

In terms of accuracy, the LDP-MST algorithm is very competitive

TABLE 6
The ACC and NMI Scores of Clustering Algorithms on Synthetic Data Sets



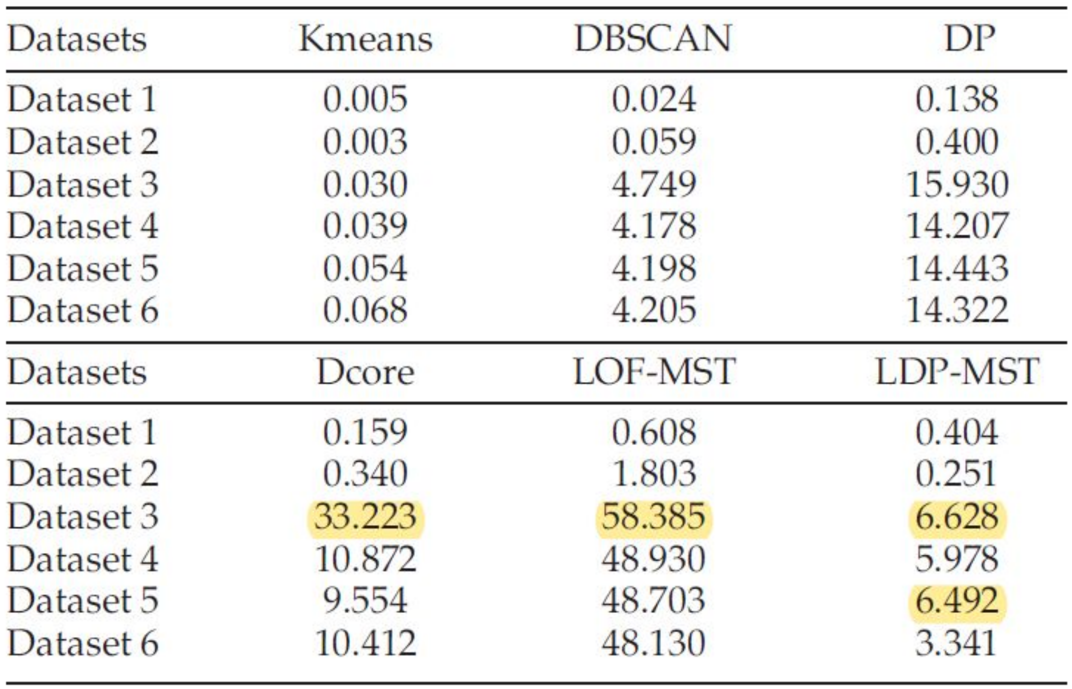
Datasets		Kmeans	DBSCAN	DP	Dcore	LOF-MST	LDP-MST
Dataset 1	ACC	0.836	0.827	**0.999**	0.996	0.848	0.997
	NMI	0.840	0.895	**0.996**	0.988	0.826	0.992
Dataset 2	ACC	0.376	0.956	0.384	0.616	0.850	**0.995**
	NMI	0.346	0.878	0.277	0.617	0.687	**0.962**
Dataset 3	ACC	0.603	0.794	0.711	0.827	0.850	**0.998**
	NMI	0.729	0.860	0.770	0.900	0.860	**0.991**
Dataset 4	ACC	0.654	0.916	0.782	0.988	0.667	**0.983**
	NMI	0.604	0.874	0.843	0.963	0.656	**0.950**
Dataset 5	ACC	0.477	0.773	0.510	0.932	0.730	**0.989**
	NMI	0.632	0.829	0.642	0.870	0.718	**0.968**
Dataset 6	ACC	0.484	0.769	0.653	0.761	0.800	**0.994**
	NMI	0.601	0.842	0.724	0.803	0.771	**0.980**

這是分群實驗的量化結果，ACC 和 NMI 指標都是越高越好。螢光標註的部分是 ACC 中該列第二高的方法，可以注意到其中 LDP-MST 在第三組和第六組的表現明顯贏過其他方法

Results

Under the premise of good classification performance, the actual execution speed of LDP-MST is quite fast

TABLE 7
The Running Time of Each Clustering Algorithm on Synthetic Data Sets (s)



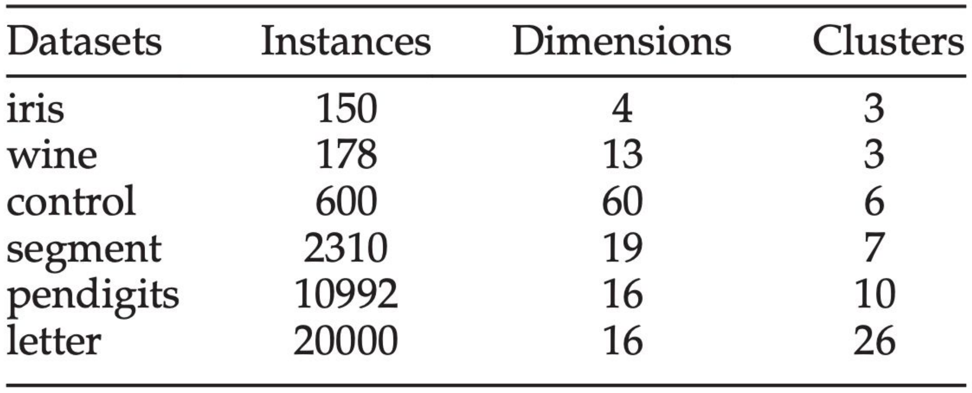
Datasets	Kmeans	DBSCAN	DP
Dataset 1	0.005	0.024	0.138
Dataset 2	0.003	0.059	0.400
Dataset 3	0.030	4.749	15.930
Dataset 4	0.039	4.178	14.207
Dataset 5	0.054	4.198	14.443
Dataset 6	0.068	4.205	14.322
Datasets	**Dcore**	**LOF-MST**	**LDP-MST**
Dataset 1	0.159	0.608	0.404
Dataset 2	0.340	1.803	0.251
Dataset 3	33.223	58.385	6.628
Dataset 4	10.872	48.930	5.978
Dataset 5	9.554	48.703	6.492
Dataset 6	10.412	48.130	3.341

螢光標註的是幾個我認為跑特別久的情形，整題而言第三組資料跑最久。針對 LDP-MST 第五組的群數比較多 (9 群) 也會稍微增加需要的時間

Clustering on Real Data Sets

Real Data Sets

TABLE 8
Real Datasets from UCI



Datasets	Instances	Dimensions	Clusters
iris	150	4	3
wine	178	13	3
control	600	60	6
segment	2310	19	7
pendigits	10992	16	10
letter	20000	16	26

由於KD-tree的限制，維度大於 10的data sets都使用PCA做了降維。PCA作為前序準備工作故耗費時間多少不考慮在內。此外還選取Olivetti Face Database中的部分進行了試驗，但由於計算兩張臉的 distance的方法直接引述於其他文章且較為複雜，在此不進行詳細解釋。

Results

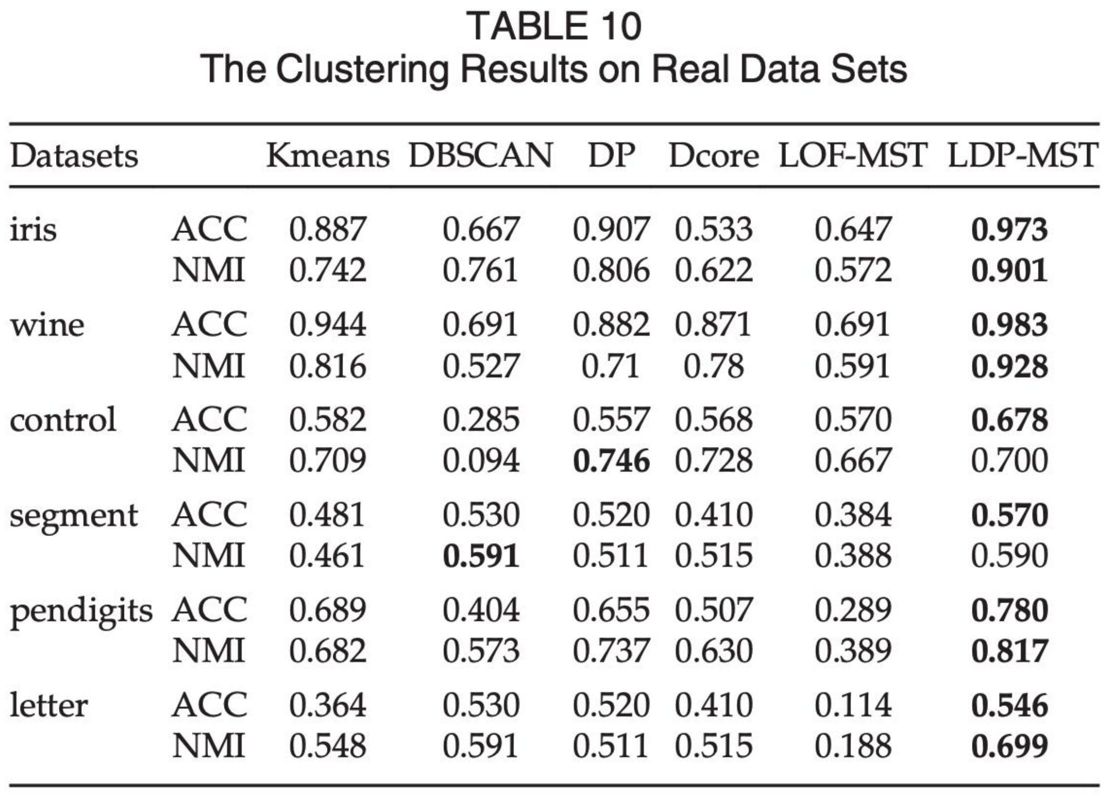


TABLE 10
The Clustering Results on Real Data Sets

Datasets		Kmeans	DBSCAN	DP	Dcore	LOF-MST	LDP-MST
iris	ACC	0.887	0.667	0.907	0.533	0.647	**0.973**
	NMI	0.742	0.761	0.806	0.622	0.572	**0.901**
wine	ACC	0.944	0.691	0.882	0.871	0.691	**0.983**
	NMI	0.816	0.527	0.71	0.78	0.591	**0.928**
control	ACC	0.582	0.285	0.557	0.568	0.570	**0.678**
	NMI	0.709	0.094	**0.746**	0.728	0.667	0.700
segment	ACC	0.481	0.530	0.520	0.410	0.384	**0.570**
	NMI	0.461	**0.591**	0.511	0.515	0.388	0.590
pendigits	ACC	0.689	0.404	0.655	0.507	0.289	**0.780**
	NMI	0.682	0.573	0.737	0.630	0.389	**0.817**
letter	ACC	0.364	0.530	0.520	0.410	0.114	**0.546**
	NMI	0.548	0.591	0.511	0.515	0.188	**0.699**

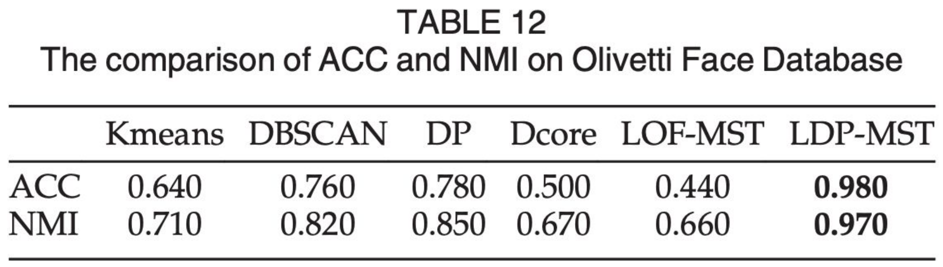


TABLE 12
The comparison of ACC and NMI on Olivetti Face Database

	Kmeans	DBSCAN	DP	Dcore	LOF-MST	LDP-MST
ACC	0.640	0.760	0.780	0.500	0.440	**0.980**
NMI	0.710	0.820	0.850	0.670	0.660	**0.970**

相比其他幾種方法，LDP-MST的accuracy基本處於領先狀態。

Results

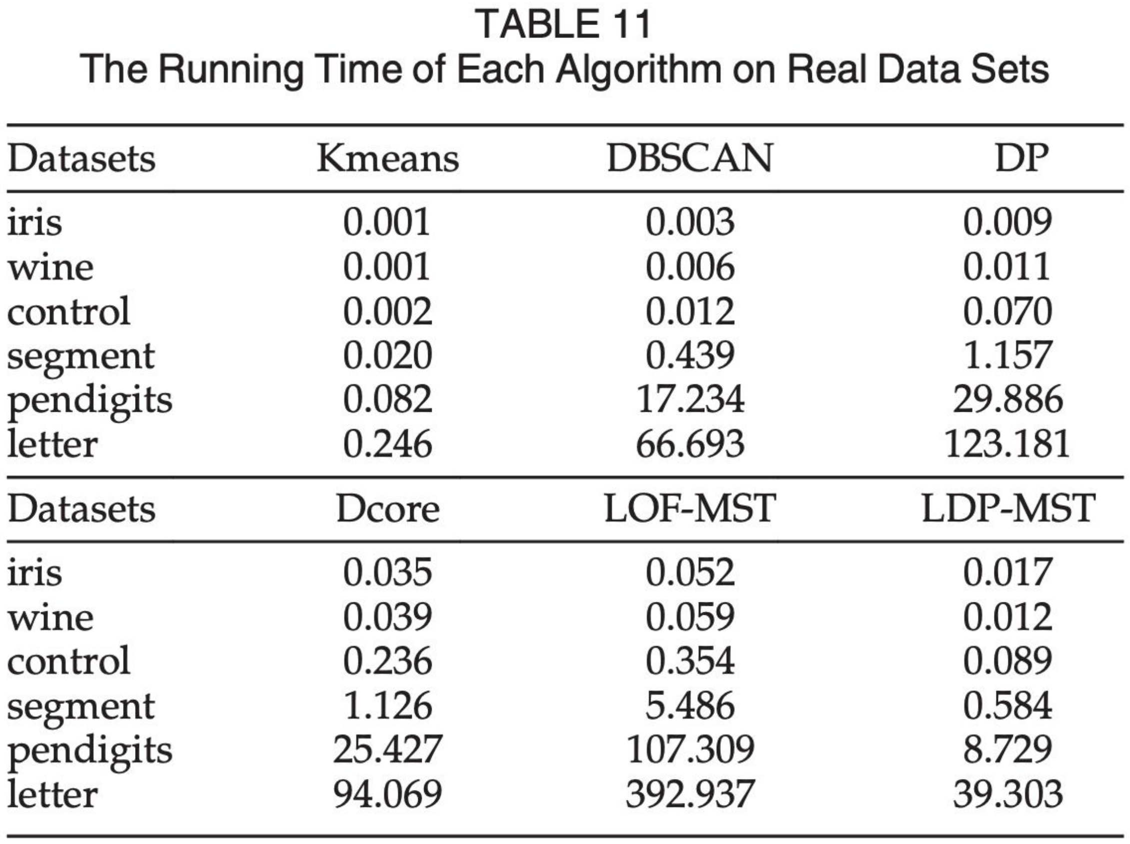


TABLE 11
The Running Time of Each Algorithm on Real Data Sets

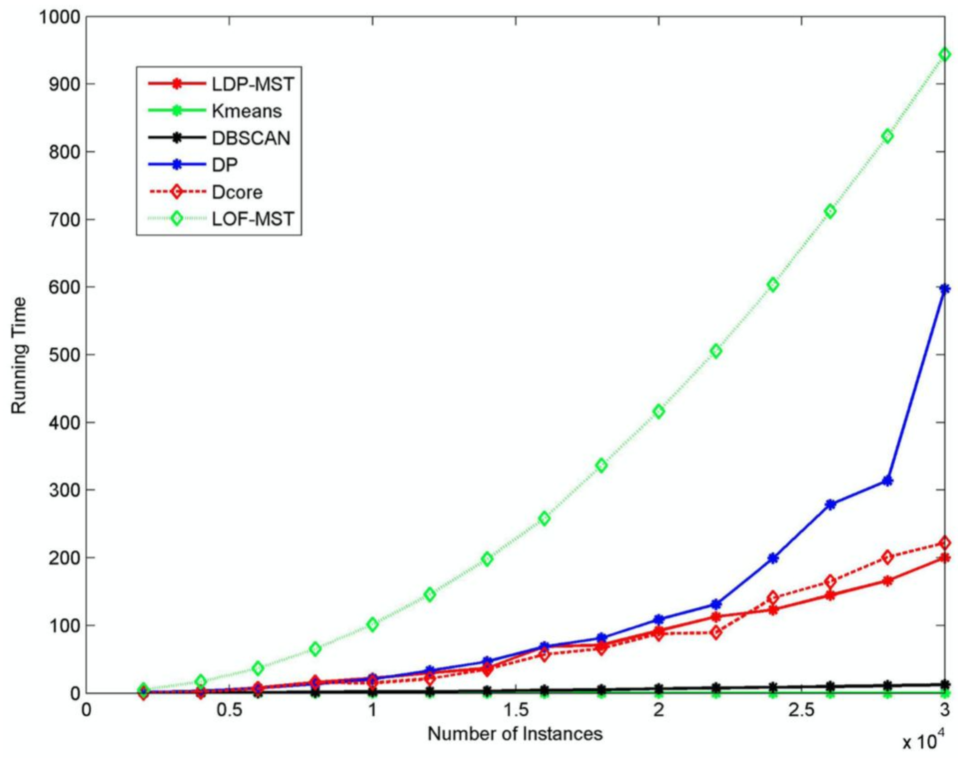
Datasets	Kmeans	DBSCAN	DP	Dcore	LOF-MST	LDP-MST
iris	0.001	0.003	0.009	0.035	0.052	0.017
wine	0.001	0.006	0.011	0.039	0.059	0.012
control	0.002	0.012	0.070	0.236	0.354	0.089
segment	0.020	0.439	1.157	1.126	5.486	0.584
pendigits	0.082	17.234	29.886	25.427	107.309	8.729
letter	0.246	66.693	123.181	94.069	392.937	39.303

Evaluation on Running Time

The Impact of the Number of Instances

Two-dimensional data sets.

Instances are randomly generated by two different Gaussian distribution.

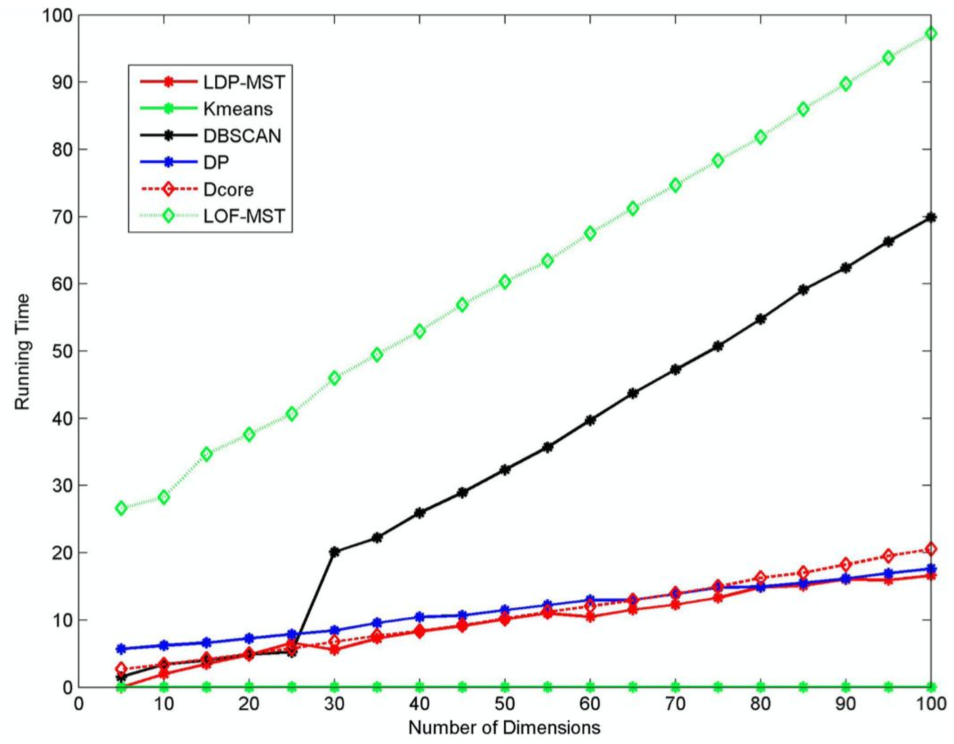


LDP-MST耗時雖長於Kmeans和DBSCAN，但遠優於LOF-MST。

The Impact of the Number of Dimensions

5000 instances.

Instances are randomly generated by two different Gaussian distribution.



當維度大過30時DBSCAN耗時大幅度增加。而 Kmeans，DP，Dcore和LDP-MST的耗時受維度增加的影響較小。

:)

謝謝大家

分工

1. Why clustering?
2. How to generate clusters?
    a. Center-Based
    b. Density-Based
    c. MST-Based
3. **Local Density Peaks-Based Minimum Spanning Tree (LDP-MST)**
4. Experiments
    a. Synthetic Datasets
    b. Real Datasets
    c. Evaluation on Running Time

吳海韜　　　　闕中一