数 据 挖 掘

Data Mining

主讲: 张仲楠 教授

厦門大學
XIAMEN UNIVERSITY

陆

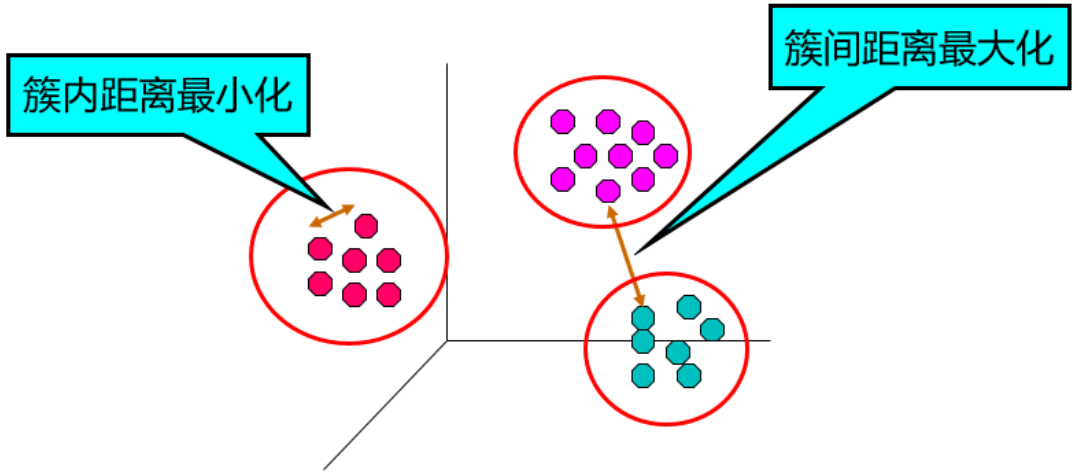
聚类分析

目 录

1.概述

1. 什么是聚类分析

■ 聚类分析仅根据在数据中发现的描述对象及其关系的信息，将数据对象分组（称为簇）。其目标是，组内的对象是相似的(相关的)，而不同组中的对象是不同的(不相关的)。组内的相似性(同质性)越大，组间差别越大，聚类就越好。



1. 什么是聚类分析

■ 在许多应用中，簇都没有很好的定义。

几个簇?

两个簇

四个簇

六个簇

1. 概述

1. 什么是聚类分析

- 聚类分析与其他将数据对象分组的技术相关。
- 聚类可以看作一种分类，它用类别(簇)标签创建对象的标记。然而，只能从数据导出这些标签。
- 分类是有监督分类(supervised classification)，即使用由类别标签已知的对象开发的模型，对新的、无标记的对象赋予类别标签。
- 聚类分析被视为无监督分类(unsupervised classification)。
- 在数据挖掘中，不附加任何条件使用术语分类时，通常是指有监督分类。

1. 概 述

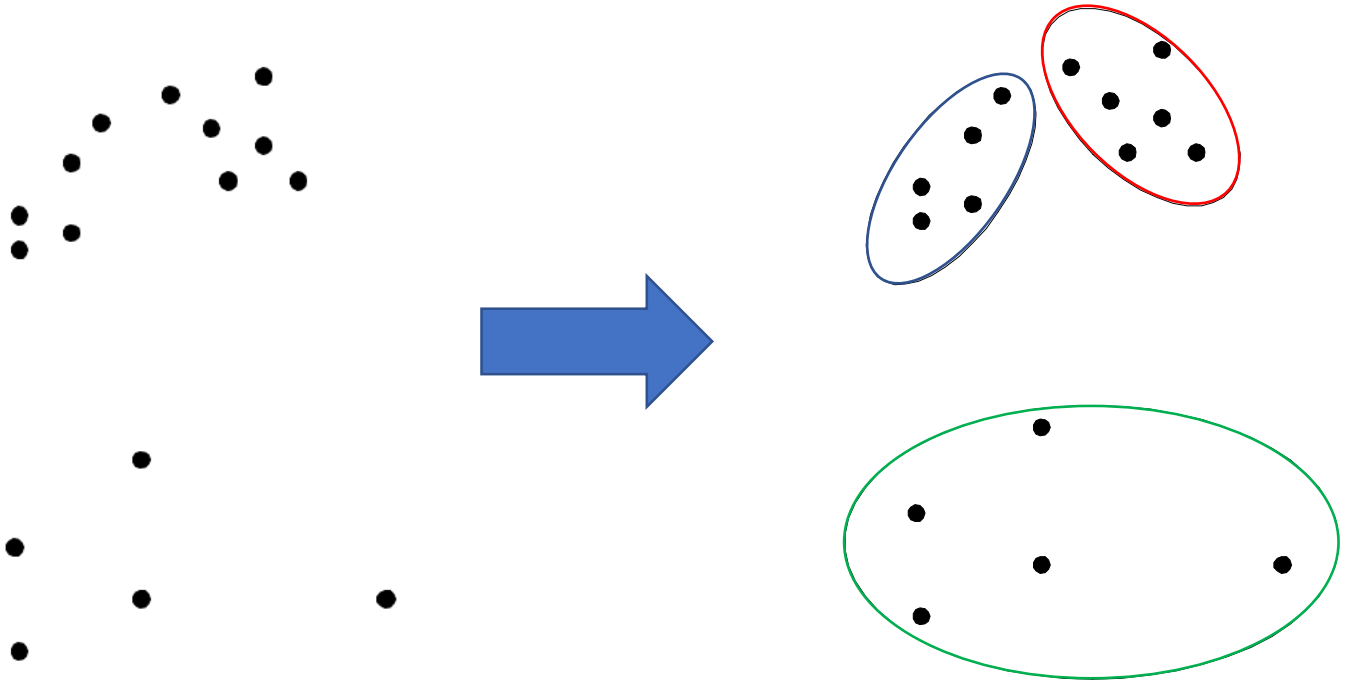
2. 聚类的不同类型

- 层次的(嵌套的)与划分的(非嵌套的)
- 互斥的、重叠的与模糊的
- 完全的与部分的

1. 概述

2. 聚类的不同类型 --- 层次的与划分的

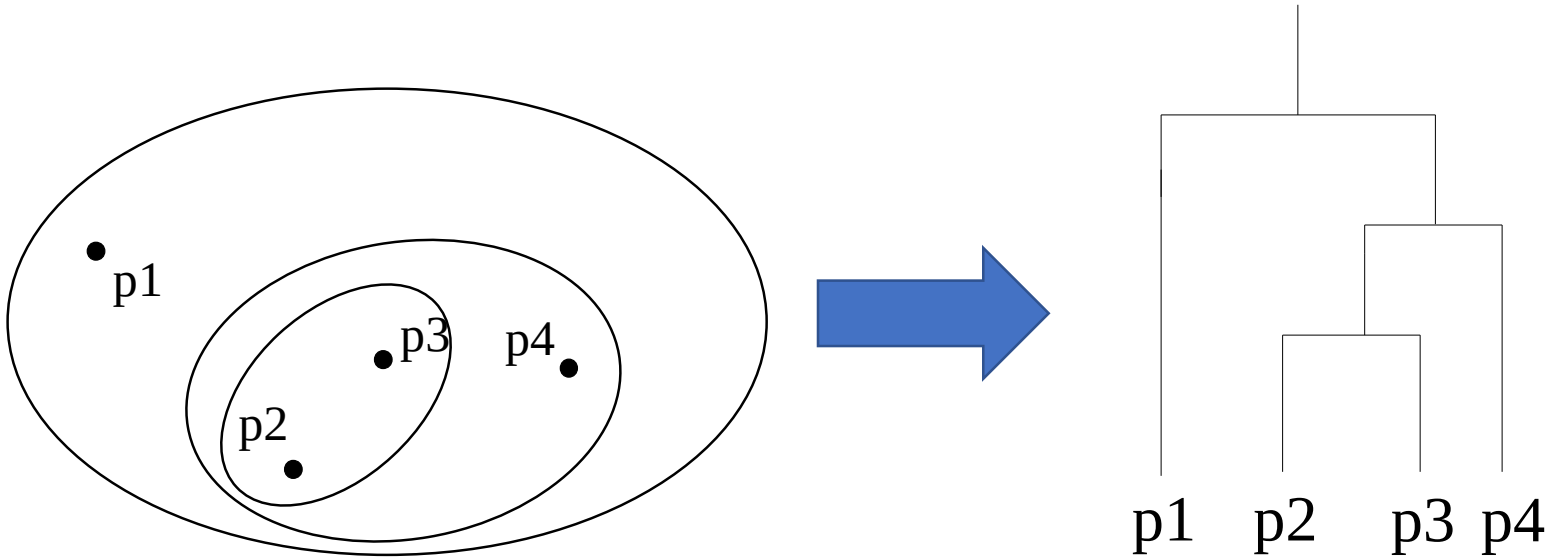
■划分聚类(partitional clustering)简单地将数据对象集划分成不重叠的子集(簇)，使得每个数据对象恰在一个子集中：



1. 概述

2. 聚类的不同类型 --- 层次的与划分的

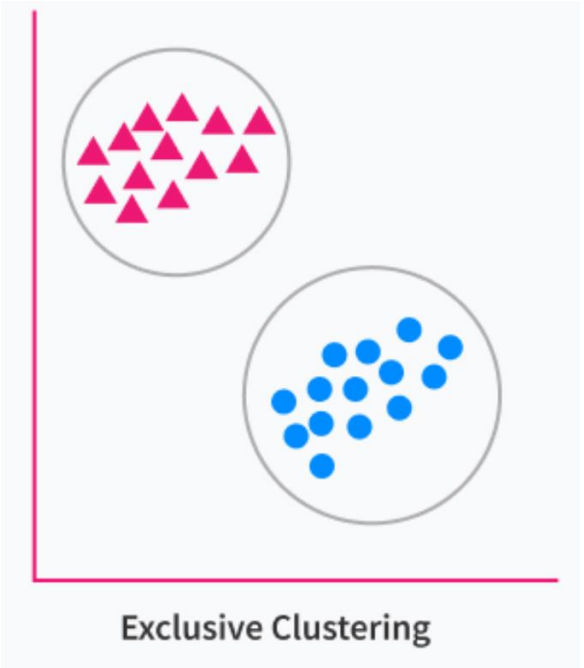
■层次聚类是嵌套簇的集合，组织成一棵树。除叶结点外，树中每一个结点(簇)都是其子女(子簇)的并集，而树根则是包含所有对象的簇。通常树叶是单个数据对象的单元素。

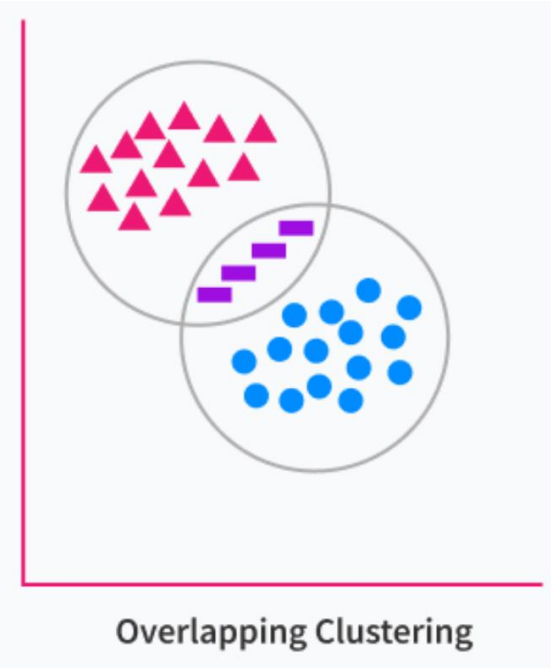


1. 概 述

2. 聚类的不同类型 --- 互斥的、重叠的与模糊的

- 互斥的(exclusive)：每个对象都被指派到单个簇。
- 重叠的(overlapping)或非互斥的(non-exclusive)：一个对象同时属于多个组(类)。





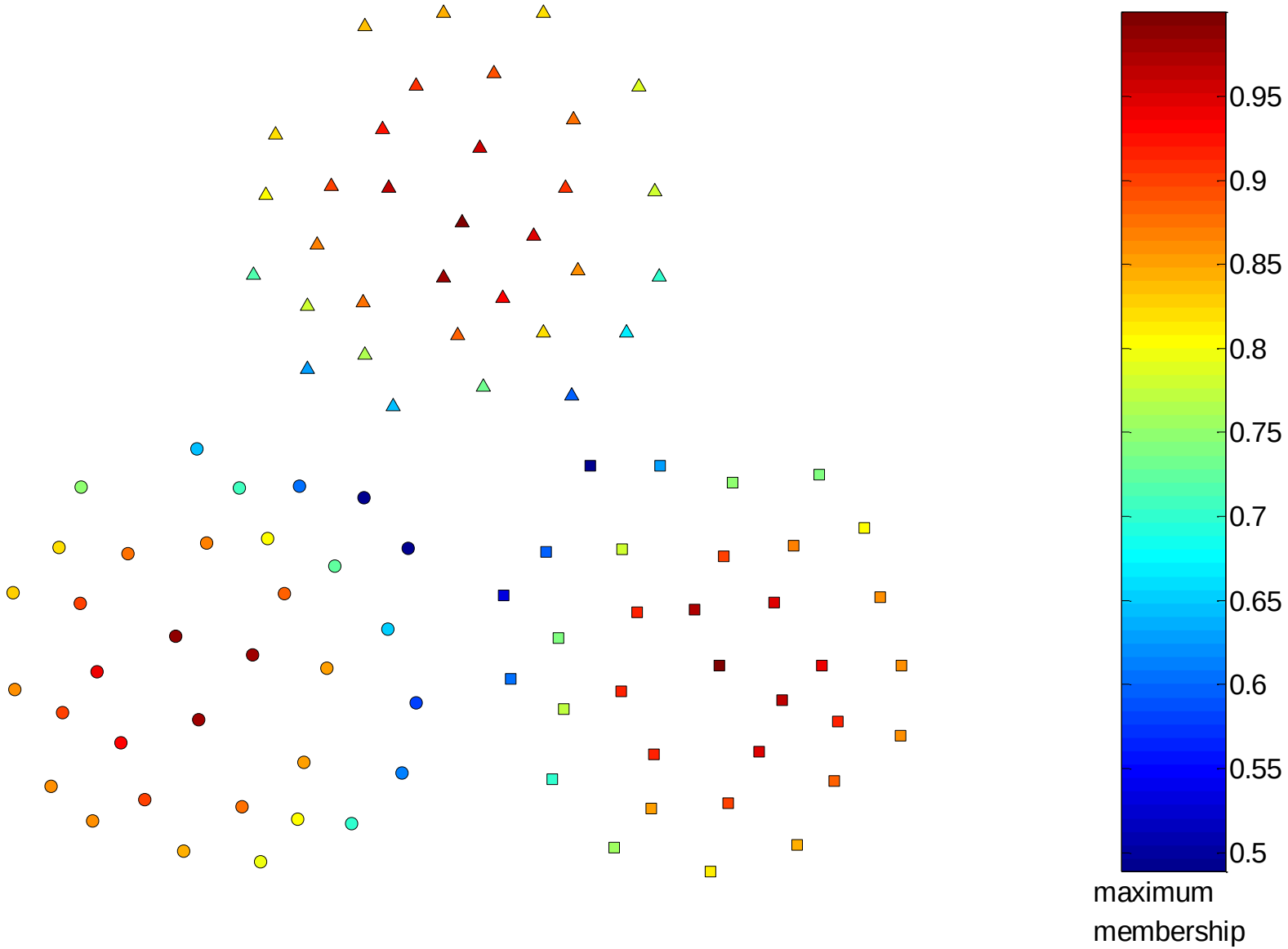
1. 概 述

2. 聚类的不同类型 --- 互斥的、重叠的与模糊的

- 模糊聚类(fuzzy clustering)：每个对象以0(绝对不属于)和1(绝对属于)之间的隶属权值归属于每个簇 。
- 每个对象的隶属于不同簇的权值之和必须等于1。
- 通常将对象指派到具有最大隶属权值或概率的簇，将模糊或概率聚类转换成互斥聚类 。

1. 概述

- 每个点指派到它具有最大隶属权值的簇。
- 属于各个簇的点用不同的标记显示，而点在簇中的隶属度用明暗程度表示。
- 点越黑，它在被指派的簇中隶属度越高。
- 靠近簇中心的点的隶属度最高，而簇间点的隶属度最低。



1. 概述

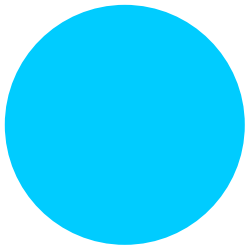
2. 聚类的不同类型 --- 完全的与部分的

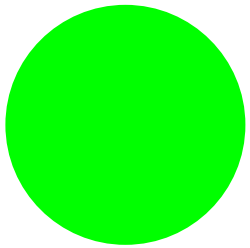
- 完全聚类(complete clustering)：将每个对象指派到一个簇。
- 部分聚类(partial clustering)：数据集中的某些对象可能不属于明确定义的组。
- 数据集中的一些对象代表噪声、离群点或“不感兴趣的背景”。

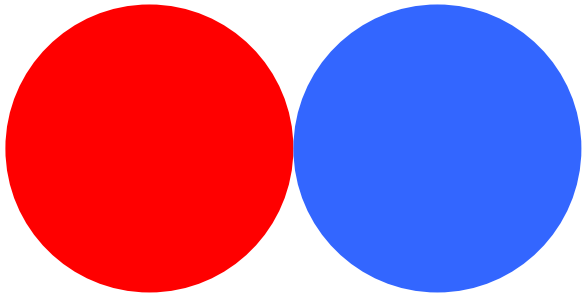
1. 概述

3. 簇的不同类型 --- 明显分离的

- 簇是对象的集合，其中每个对象到同簇中每个对象的距离比到不同簇中任意对象的距离都近(或更加相似)。
- 使用一个阈值来说明簇中的所有对象必须充分接近(或相似)。
- 不同组中任意两点之间的距离都大于组内任意两点之间的距离。
- 明显分离的簇不必是球形的，可以具有任意形状。







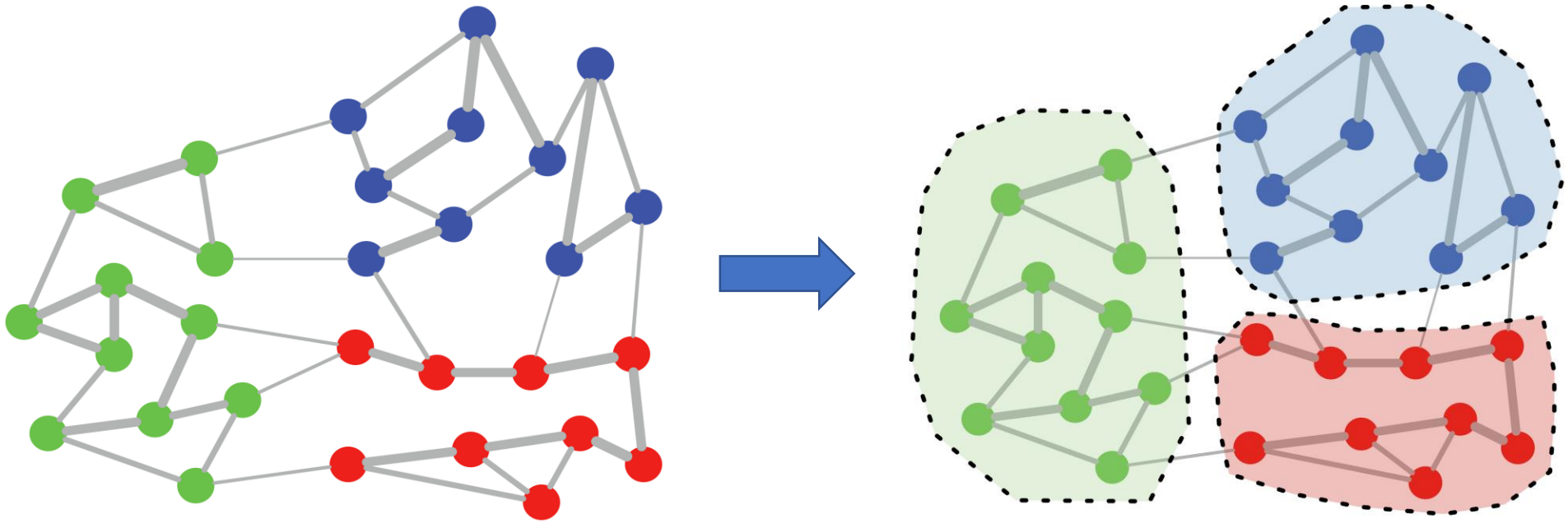
3. 簇的不同类型 --- 基于原型的

- 簇是对象的集合，其中每个对象到定义该簇的原型的距离比到其他簇的原型的距离更近(或更加相似)。
- 对于具有连续属性的数据，簇的原型通常是质心，即簇中所有点的平均值。
- 当质心没有意义时(例如当数据具有分类属性时)，原型通常是中心点，即簇中最有代表性的点。
- 对于许多数据类型，原型可以视为最靠近中心的点。在这种情况下，通常把基于原型的簇看作基于中心的簇(center-based cluster)。毫无疑问，这种簇趋于呈球形。

1. 概 述

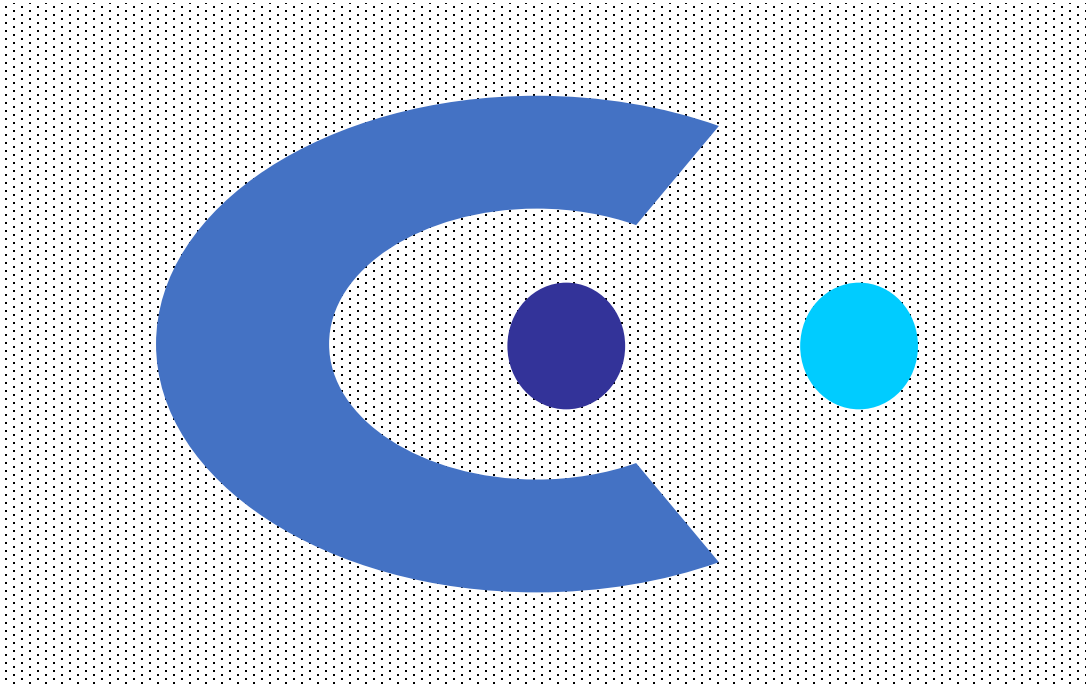
3. 簇的不同类型 --- 基于图的

- 如果数据用图表示，其中结点是对象，而边代表对象之间的联系，则簇可以定义为连通分支(connected component)，即互相连通但不与组外对象连通的对象组。



3. 簇的不同类型 --- 基于密度的

■簇是对象的稠密区域，被低密度的区域环绕。



目 录

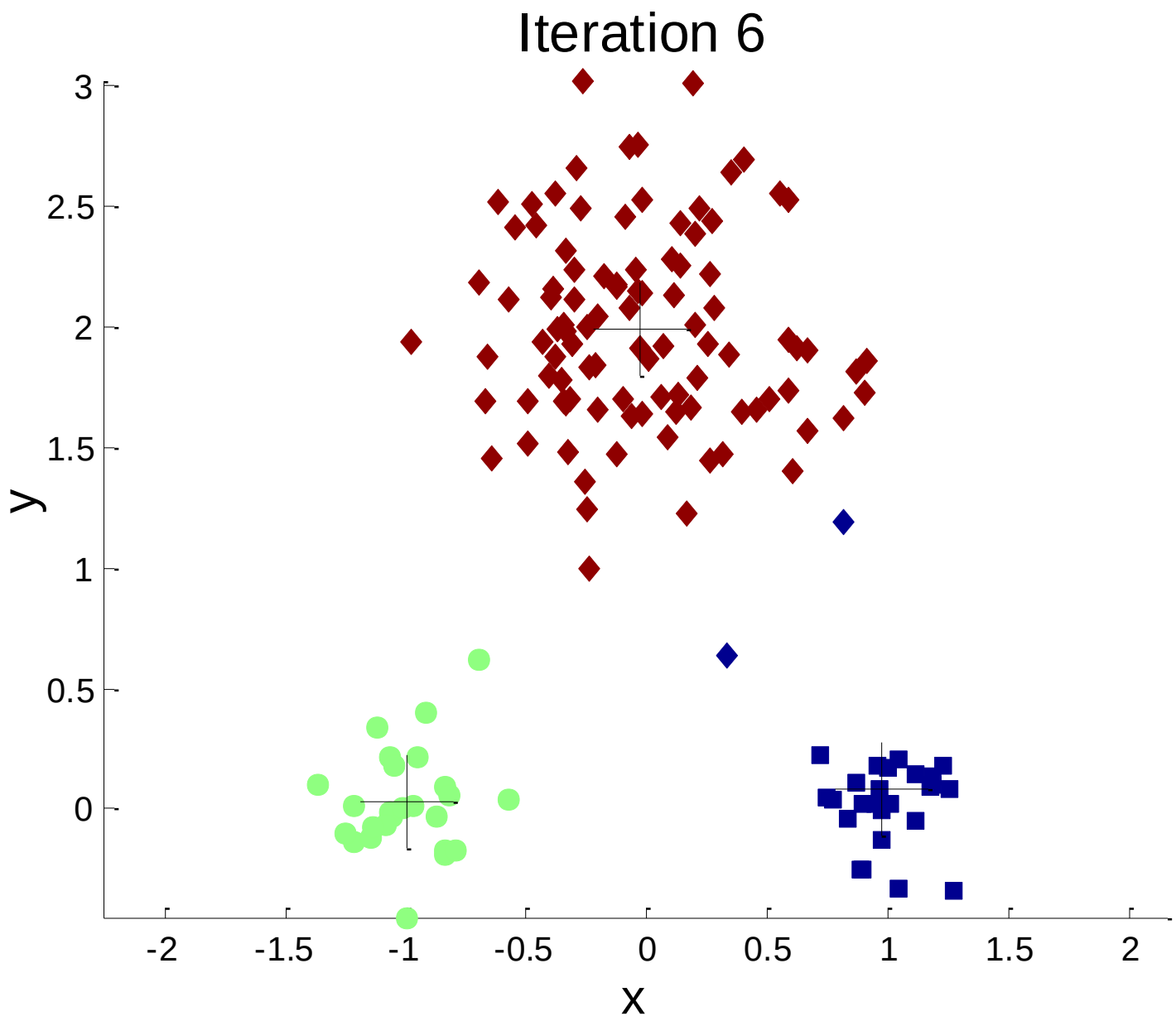
2.K均值

1. K均值算法

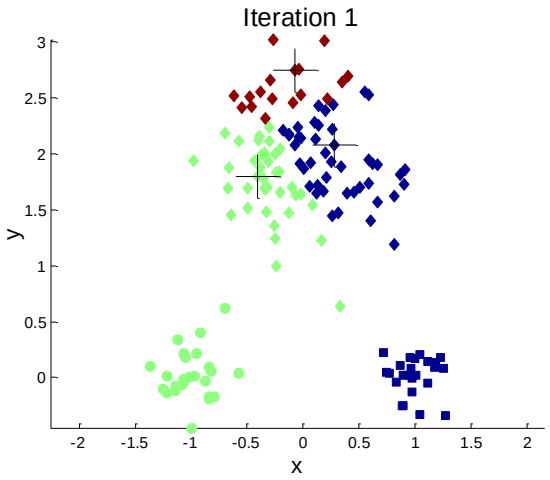
- 首先，选择K个初始质心，其中K是用户指定的参数，即所期望的簇的个数。
- 每个点被指派到最近的质心，而指派到一个质心的点集为一个簇。
- 然后，根据被指派到簇的点，更新每个簇的质心。
- 重复指派和更新步骤，直到簇不发生变化，或者直到质心不发生变化。

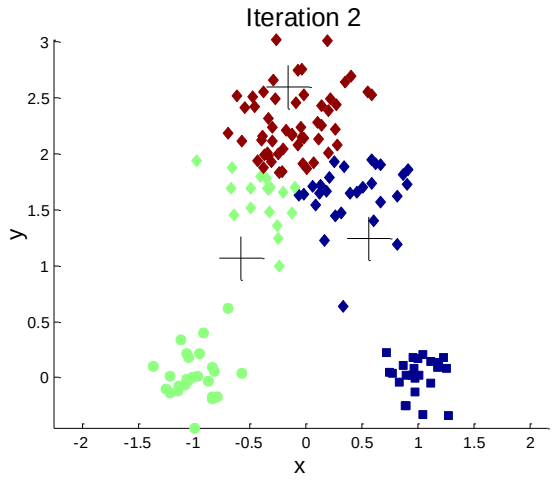
算法7.1 基本K均值算法	
1:	选择K个点作为初始质心
2:	**repeat**
3:	将每个点指派到最近的质心，形成K个簇
4:	重新计算每个簇的质心
5:	**until** 质心不发生变化

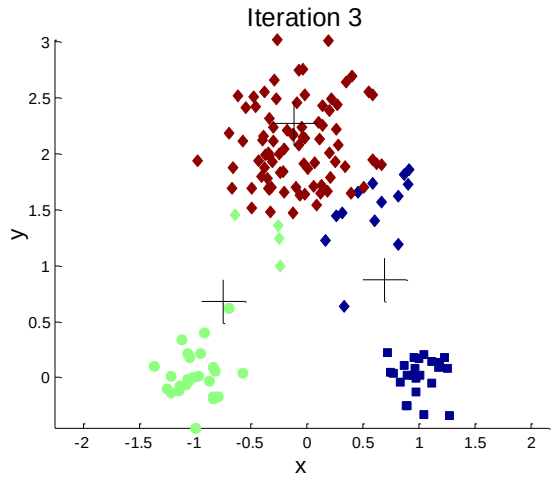
2.K均值

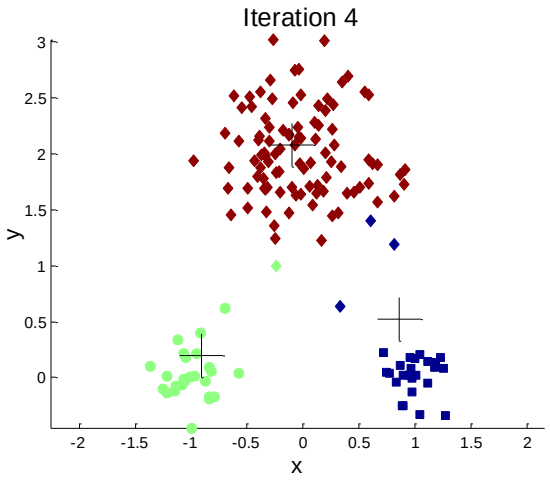


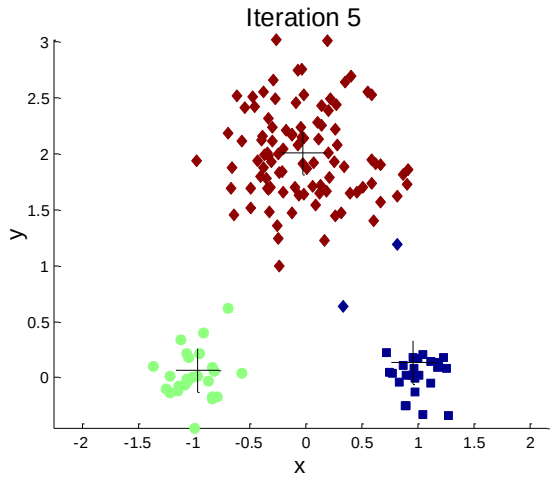
2.K均值

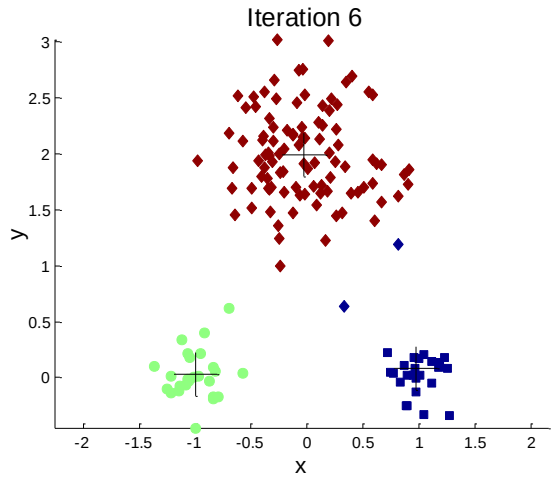

Iteration 1
x
y


Iteration 2
x
y

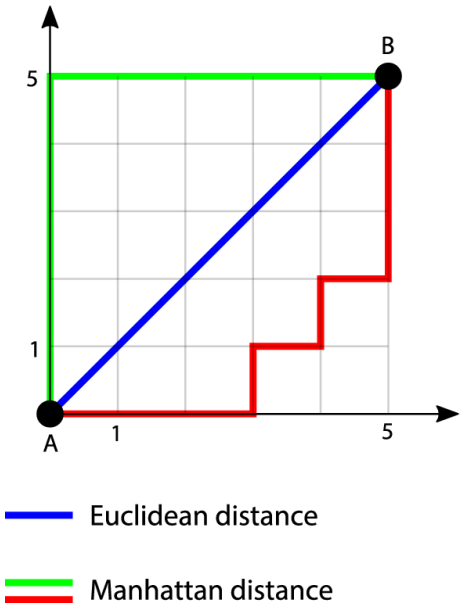

Iteration 3
x
y


Iteration 4
x
y


Iteration 5
x
y


Iteration 6
x
y

2.K均值



1. K均值算法 --- 指派点到最近的质心

- 为了将点指派到最近的质心，我们需要邻近度度量来量化所考虑的数据的“最近”概念。
- 对欧氏空间中的点使用欧几里得距离(L_2)，对文档使用余弦相似性。
- 对于给定的数据类型，可能存在多种适合的邻近度度量。
  - 曼哈顿距离(L_1)可用于欧几里得数据，而Jaccard度量常常用于文档。

2.K均值

1. K均值算法 --- 指派点到最近的质心

- 由于算法要重复计算每个点与每个质心的相似度，因此通常来说，K均值使用的相似性度量是相对简单的。
- 然而在某些情况下，如数据在低维欧几里得空间中时，许多相似度计算都是有可能避免的，因此能显著地加快K均值算法的速度。
- 二分K均值是另一种通过减少相似度计算量来加快K均值算法速度的方法。

2.K均值

1. K均值算法 --- 质心和目标函数

- K均值算法的步骤4为“重新计算每个簇的质心”，因为质心可能随数据邻近度度量和聚类目标的不同而改变。
- 聚类的目标通常用一个目标函数表示，该函数依赖于点之间或点到簇的质心的邻近度。
  - 例如，最小化每个点到最近质心的距离的平方。
- 一旦选定了邻近度度量和目标函数后，质心的选择就可以从数学上确定。

2.K均值

1. K均值算法 --- 质心和目标函数(欧氏空间中的数据)

- 误差的平方和(Sum of the Squared Error，SSE)作为度量聚类质量的目标函数。SSE 也称为散布(scatter)。
- 计算每个数据点的误差，即它到最近质心的欧几里得距离，然后计算误差的平方和。
- 给定由两次运行K均值算法产生的两个不同的簇集，我们更喜欢误差平方和最小的那个，因为这说明聚类的原型(质心)可以更好地代表簇中的点。

2.K均值

1. K均值算法 --- 质心和目标函数(欧氏空间中的数据)

■SSE形式地定义如下:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2$$

■其中，$dist$ 是欧几里得空间中两个对象之间的标准欧几里得距离(L_2)。

■第i个簇的质心(均值) $c_i = \frac{1}{m_i} \sum_{x \in C_i} x$

表7.1 符号表

符号	描述
x	对象
C_i	第i个簇
c_i	簇C_i的质心
c	所有点的质心
m_i	第i个簇中对象的个数
m	数据集中对象的个数
K	簇的个数

2.K均值

1. K均值算法 --- 质心和目标函数(欧氏空间中的数据)

- K均值算法的步骤3和步骤4试图直接最小化 SSE (目标函数)。
- 步骤 3 通过将点指派到最近的质心形成簇，最小化给定质心集的 SSE。
- 步骤4 重新计算质心，进一步最小化 SSE。
- K均值的步骤3和步骤4只能确保针对选定的质心和簇找到关于SSE 的局部最优，而不是对所有可能的选择来优化 SSE。

2.K均值

1. K均值算法 --- 质心和目标函数(文档数据)

- 文档数据可以用余弦相似度度量。

$$\cos(x,y)=\frac{x\cdot y}{\|x\|\|y\|}=\frac{\sum_{k=1}^{n}x_k y_k}{\sqrt{\sum_{k=1}^{n}x_k^2}\sqrt{\sum_{k=1}^{n}y_k^2}}$$

- 假定文档数据用文档-词矩阵表示。

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

2.K均值

1. K均值算法 --- 质心和目标函数(文档数据)

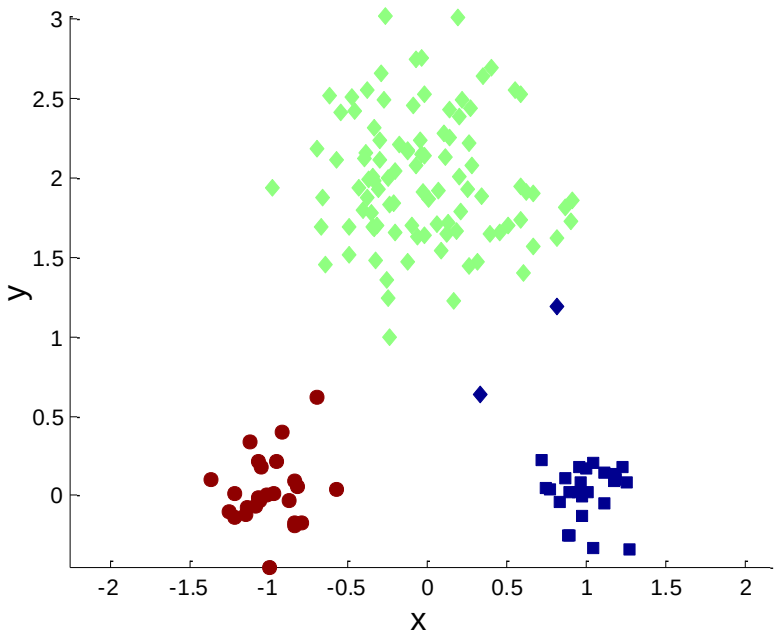
- 目标是最大化簇中文档与簇的质心的相似性，该量称作簇的凝聚度(cohesion)。
- 对于该目标，与欧几里得中的数据一样，簇的质心是均值。
- 总 SSE 的类似量是总凝聚度(total cohesion)：

$$\text{总凝聚度} = \sum_{i=1}^{K} \sum_{x \in C_i} cosine(c_i, x)$$

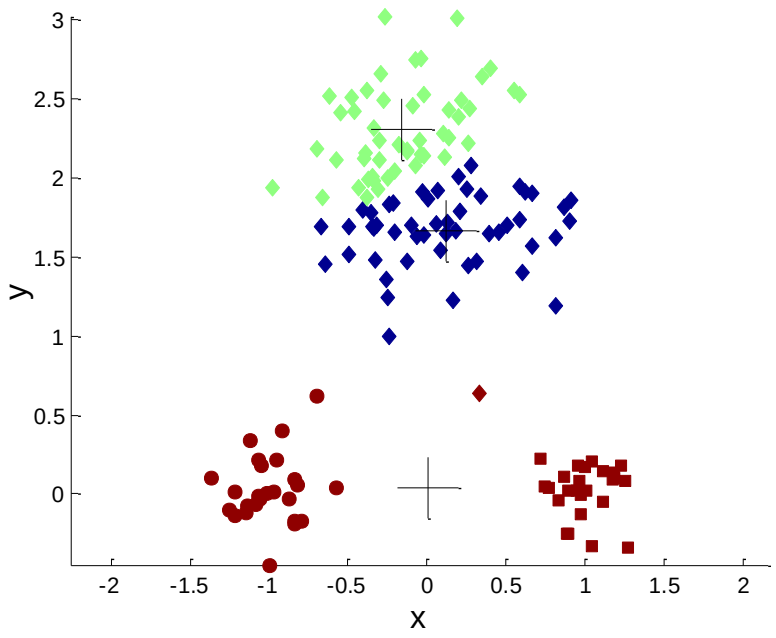
2.K均值

1. K均值算法 --- 选择初始质心

■当质心随机初始化时，K均值算法的不同执行将产生不同的总SSE。

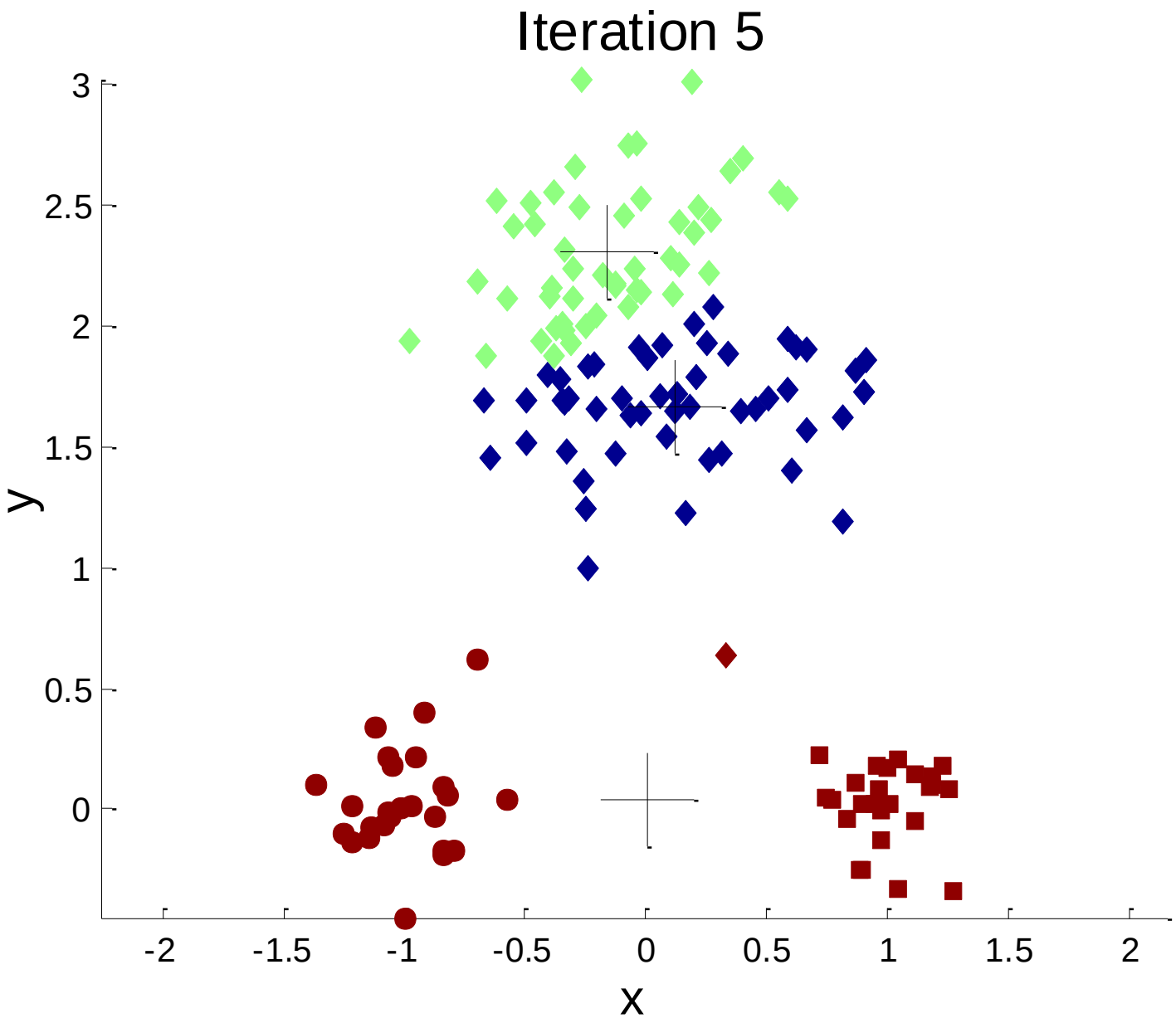


最优聚类

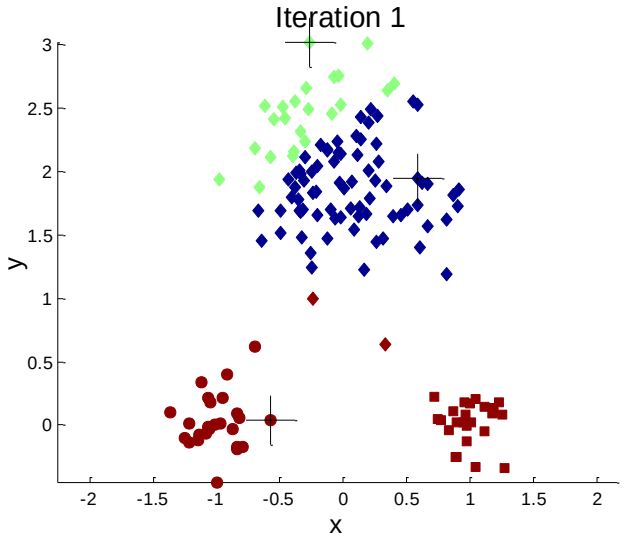


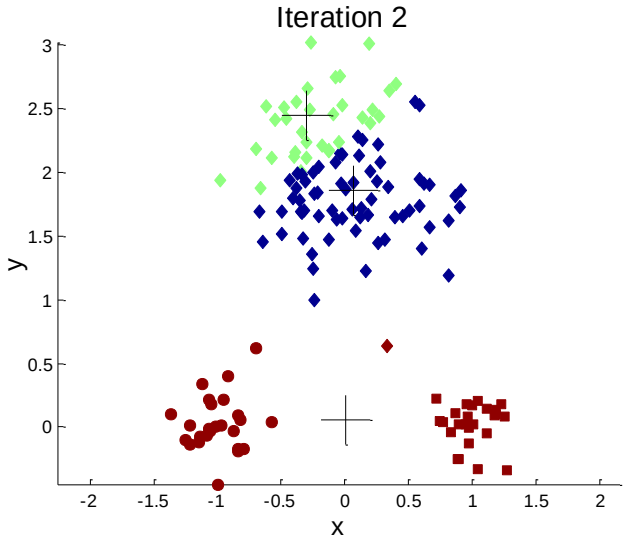
次优聚类

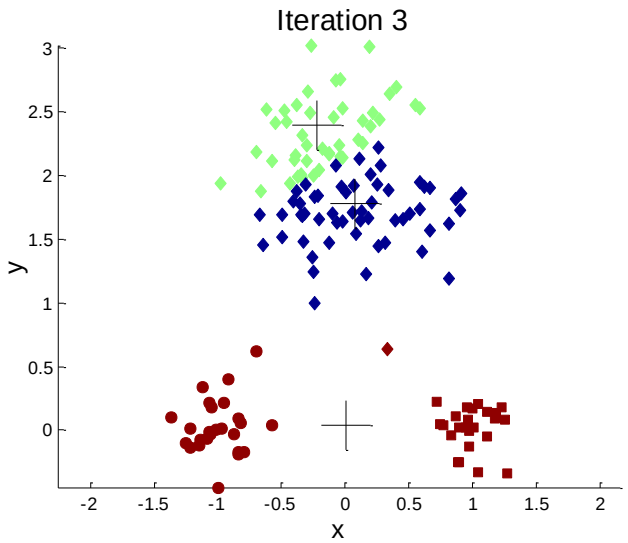
2.K均值

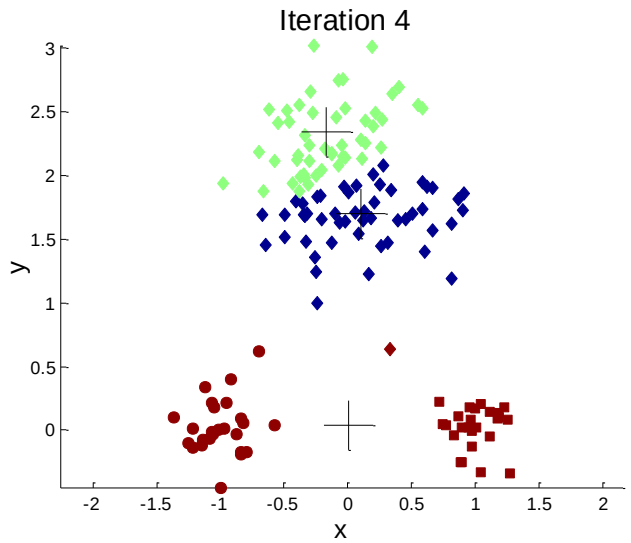


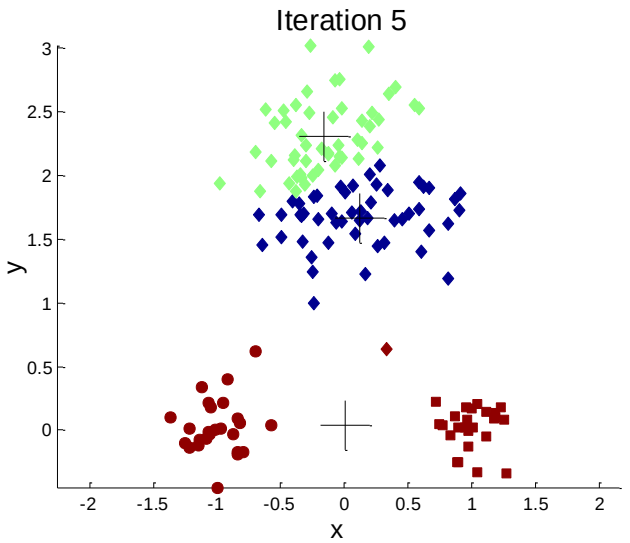
2.K均值


Iteration 1
x
y


Iteration 2
x
y


Iteration 3
x
y


Iteration 4
x
y


Iteration 5
x
y

2.K均值

1. K均值算法 --- 选择初始质心

- K均值++(K-means++)：一种初始化K均值的新方法。
- 该方法保证在 $Olog(k)$内可以找到一个最优K均值聚类的解决方案。
- 思路：
  - 随机挑选第一个质心，然后将余下的每个质心选取为尽可能远离其余质心。
  - 每一个点都有一定概率被选为新的质心，并且该概率和该点到其最近质心的距离平方成比例。

1. K均值算法 --- 选择初始质心

■K均值++(K-means++)

算法7.2 K均值++初始化算法	
1:	随机选择一个点作为初始质心
2:	**for** i=1 到试验次数 **do**
3:	计算每个点到其最近质心的距离 $d(x)$
4:	为每个点分配一个与该点的 $d(x)^2$成比例的概率
5:	使用加权概率从剩余点中选择新的质心
6:	**end for**

2. 二分K均值

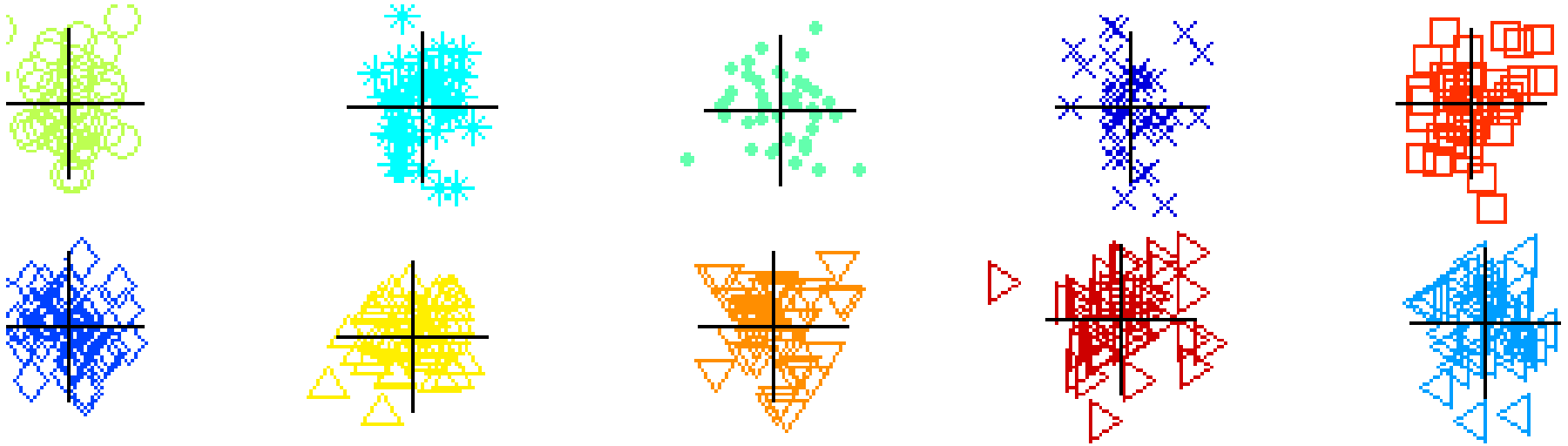
- 二分K均值算法是基本K均值算法的直接扩展，它基于一种简单想法。
- 为了得到K个簇，将所有点的集合分裂成两个簇，从这些簇中选取一个继续分裂，如此下去，直到产生K个簇。

2. 二分K均值

算法7.3　二分K均值算法	
1:	初始化簇表，使之包含由所有的点组成的簇
2:	**repeat**
3:	从簇表中取出一个簇　　/*有不同的选取方式*/
4:	{对选定的簇进行多次二分“试验”}
5:	**for** i=1 到试验次数 **do**
6:	使用基本K均值二分选定的簇
7:	**end for**
8:	从二分试验中选择具有最小总SSE的两个簇　　/*划分最好的两个簇*/
9:	将这两个簇添加到簇表中
10:	**until** 簇表中包含K个簇

2.K均值

2. 二分K均值



2. 二分K均值

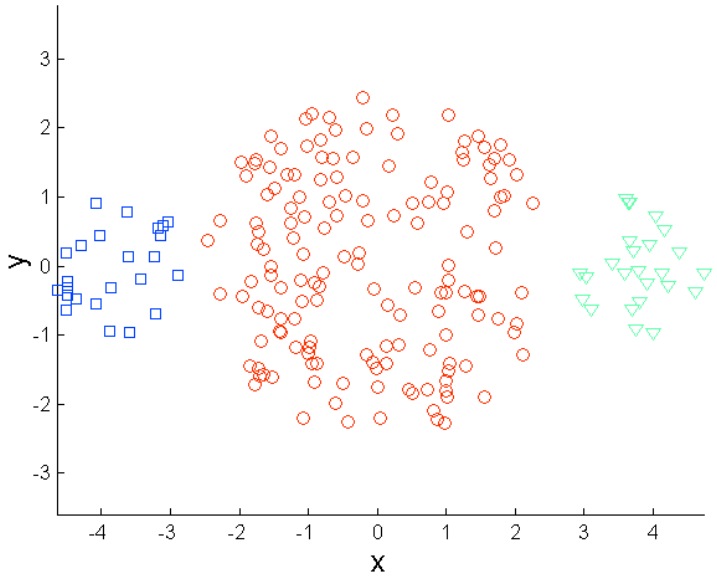
- 待分裂的簇有许多不同的选择方法：
  - 可以在每一步选择最大的簇
  - 选择具有最大SSE的簇
  - 使用一个基于大小和SSE 的标准进行选择
- “局部”使用K-means算法，即将单个聚类一分为二，所以并不代表最终的聚类集总SSE为最小。
- 因此，我们通常使用结果簇的质心作为标准K均值算法的初始质心，以此对结果簇逐步求精。

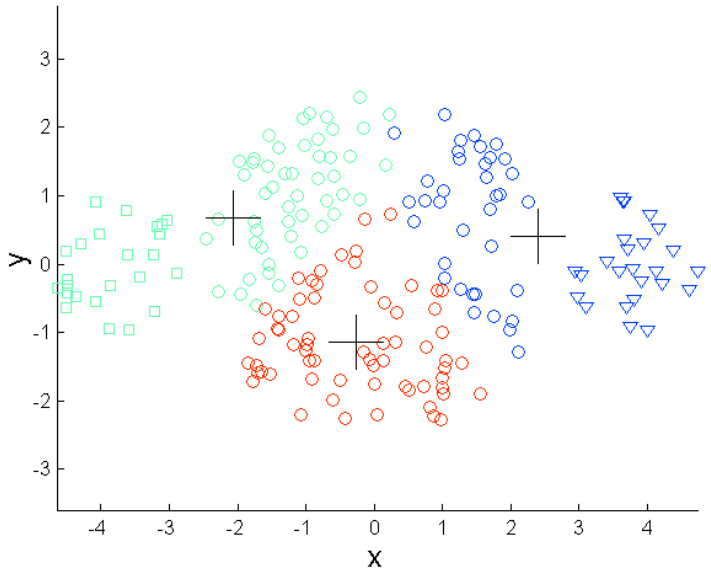
2.K均值

3. K均值和不同的簇类型

- 对于发现不同的簇类型，K均值及其变种都具有一些局限性。
- 当簇具有非球形状或具有不同尺寸或密度时， K 均值都很难检测到“自然”簇。

原始样本





3个簇

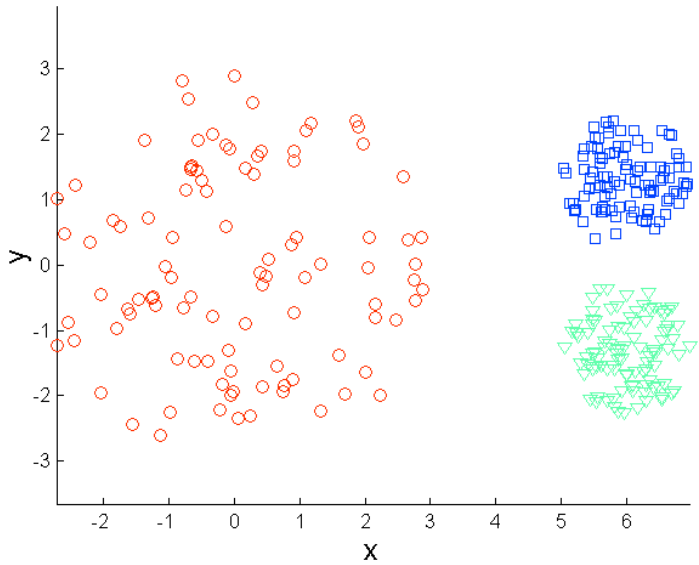
不同尺寸

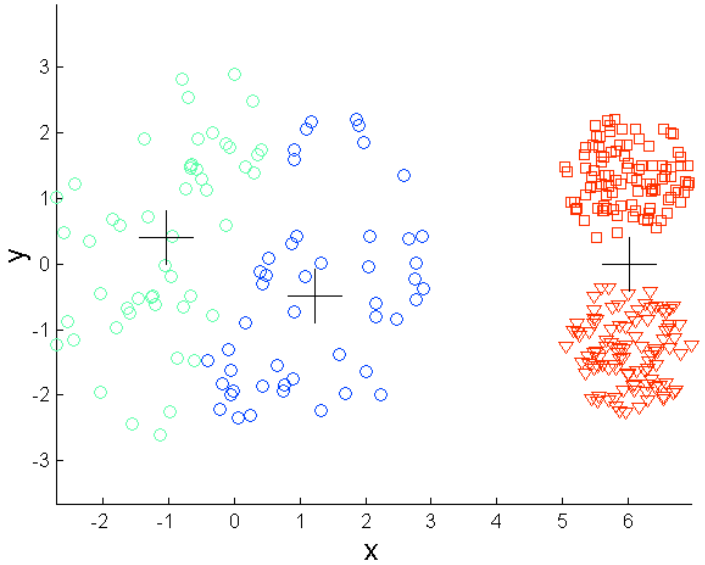
2.K均值

3. K均值和不同的簇类型

- 对于发现不同的簇类型，K均值及其变种都具有一些局限性。
- 当簇具有非球形状或具有不同尺寸或密度时，K均值都很难检测到“自然的”簇。

原始样本





3个簇

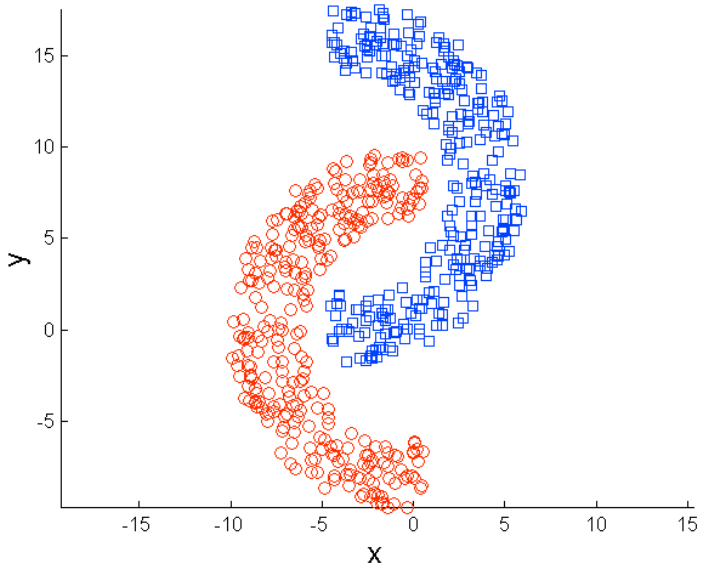
不同密度

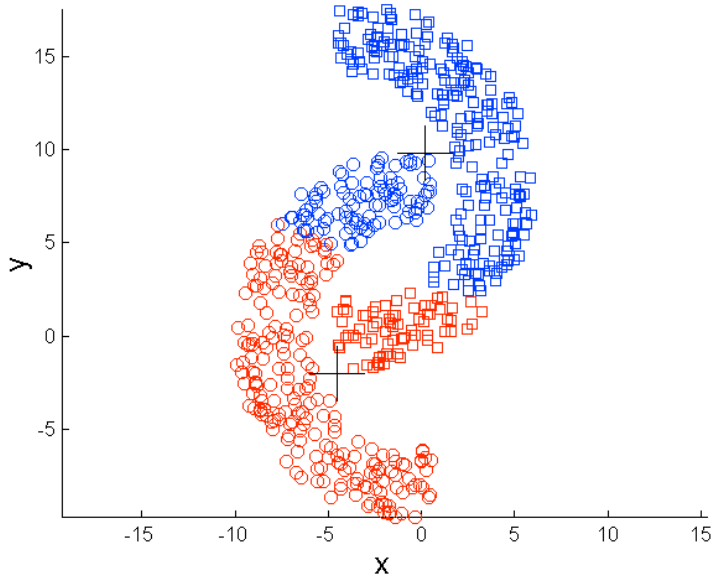
2.K均值

3. K均值和不同的簇类型

- 对于发现不同的簇类型，K均值及其变种都具有一些局限性。
- 当簇具有非球形状或具有不同尺寸或密度时，K均值都很难检测到“自然的”簇。

原始样本





2个簇

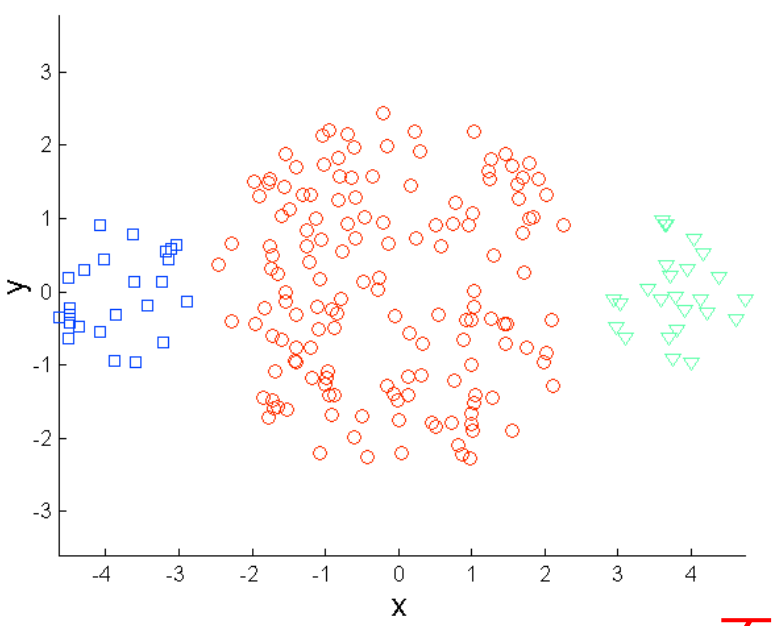
非球形状

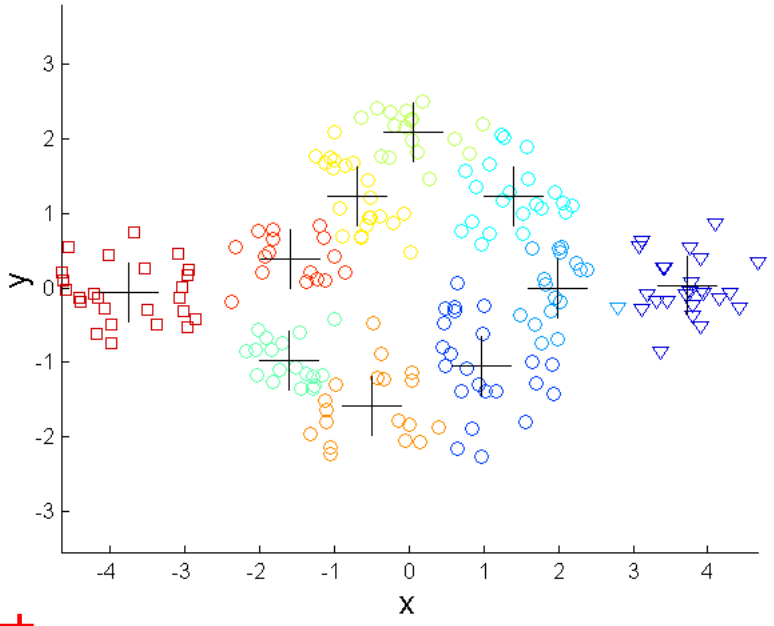
2.K均值

3. K均值和不同的簇类型

■ 一种解决方案是找到大量的簇，使它们中的每一个都代表“自然”簇的一部分。但这些小簇需要在后处理步骤中组合在一起。

原始样本





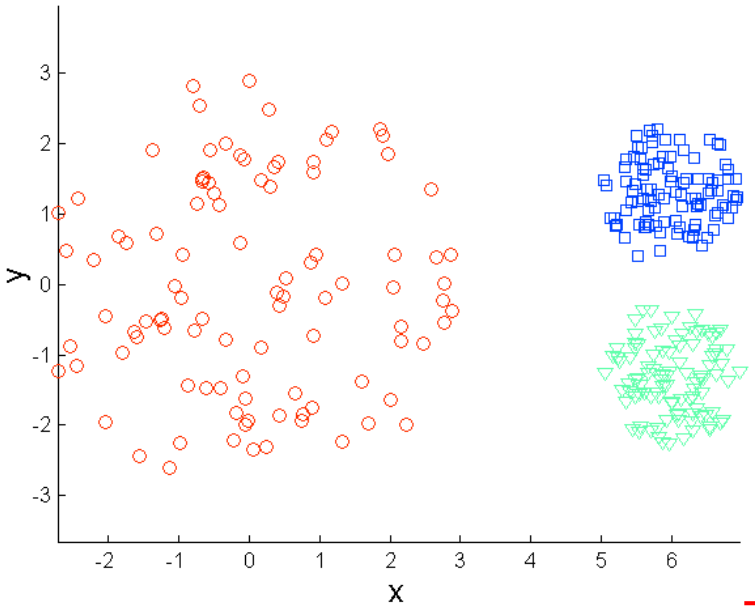
不同尺寸

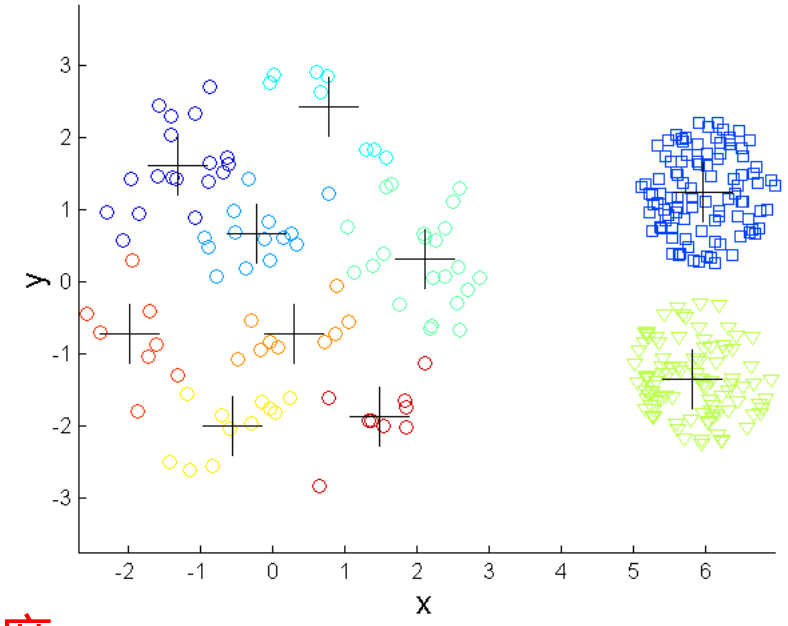
2.K均值

3. K均值和不同的簇类型

■ 一种解决方案是找到大量的簇，使它们中的每一个都代表“自然”簇的一部分。但这些小簇需要在后处理步骤中组合在一起。

原始样本



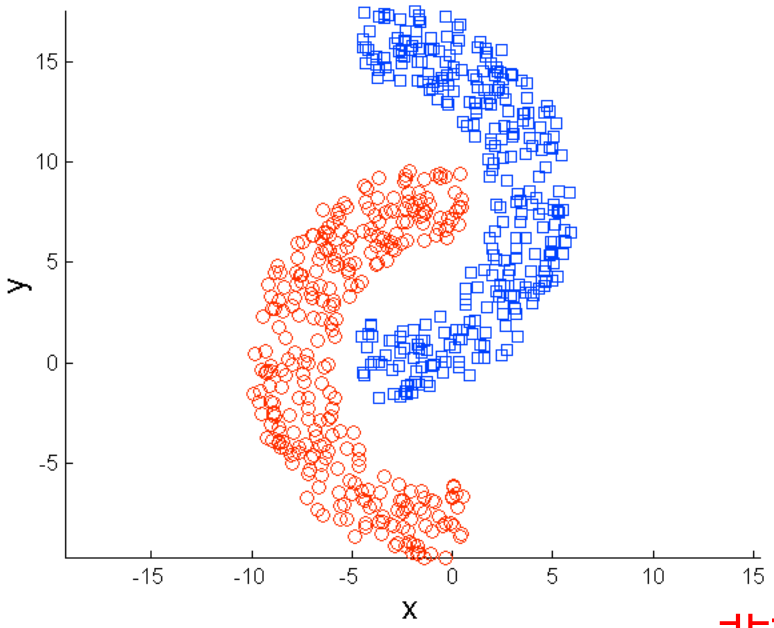


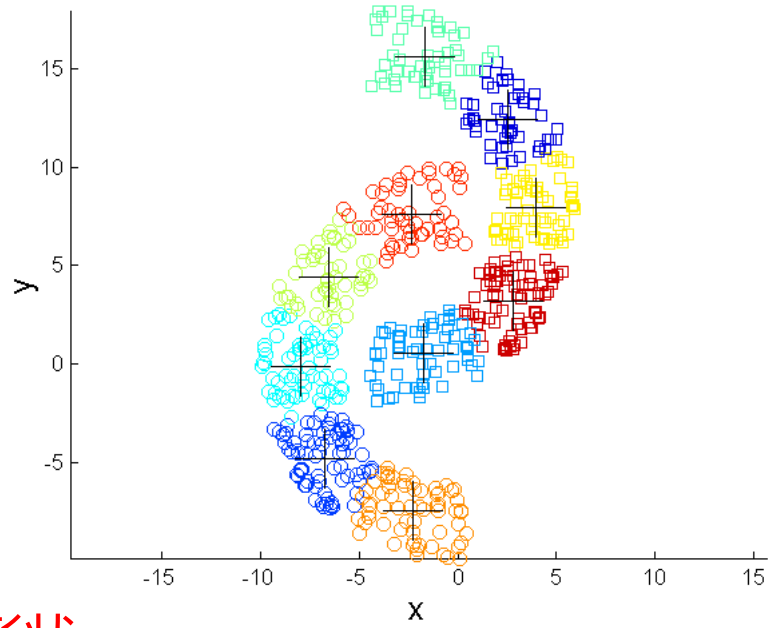
不同密度

3. K均值和不同的簇类型

■ 一种解决方案是找到大量的簇，使它们中的每一个都代表“自然”簇的一部分。但这些小簇需要在后处理步骤中组合在一起。

原始样本





非球形状

目 录

3.凝聚层次聚类

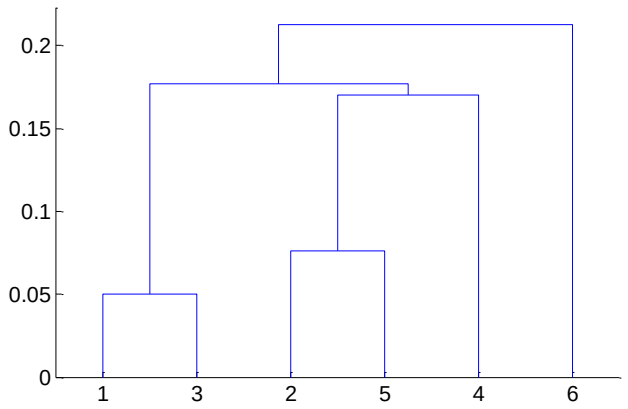
1. 基本概念

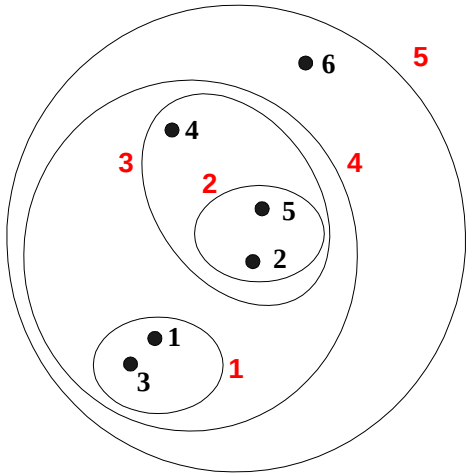
- 有两种产生层次聚类的基本方法。
- 凝聚的：从点作为个体簇开始，每一步合并两个最接近的簇。这需要定义簇的邻近度概念。
- 分裂的：从包含所有点的某个簇开始，每一步分裂一个簇，直到仅剩下单点簇。在这种情况下，我们需要确定每一步分裂哪个簇，以及如何分裂。
- 最常见的是凝聚层次聚类技术。

3.凝聚层次聚类

1. 基本概念

- 层次聚类常常使用称作树状图(dendrogram)的类似于树的图显示。该图显示簇-子簇联系和簇合并(凝聚)或分裂的次序。
- 对于二维点的集合，层次聚类也可以使用嵌套簇图(nested cluster diagram)表示 。





3.凝聚层次聚类

2. 基本凝聚层次聚类算法

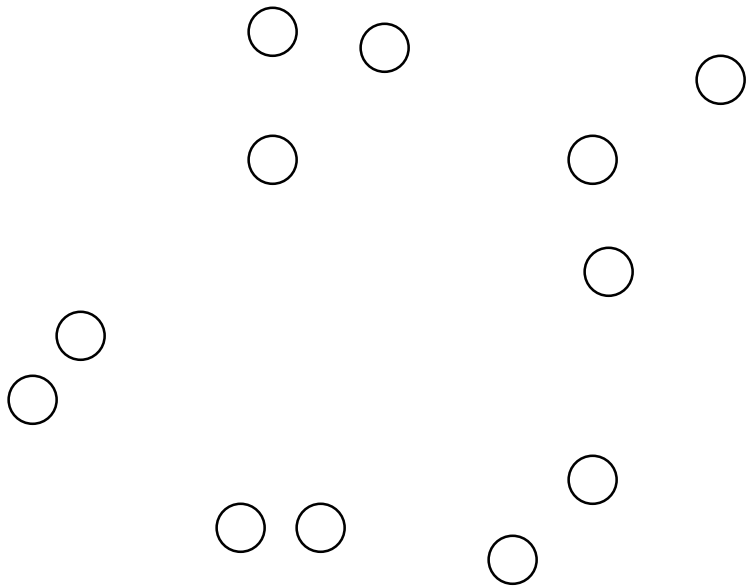
■ 从个体点作为簇开始，相继合并两个最接近的簇，直到只剩下一个簇。

算法7.4 基本凝聚层次聚类算法	
1:	如果需要，计算邻近度矩阵
2:	**repeat**
3:	合并最接近的两个簇
4:	更新邻近度矩阵，以反映新的簇与原来的簇之间的邻近性
5:	**until** 仅剩下一个簇

3.凝聚层次聚类

2. 基本凝聚层次聚类算法

■ 从个体点作为簇开始。



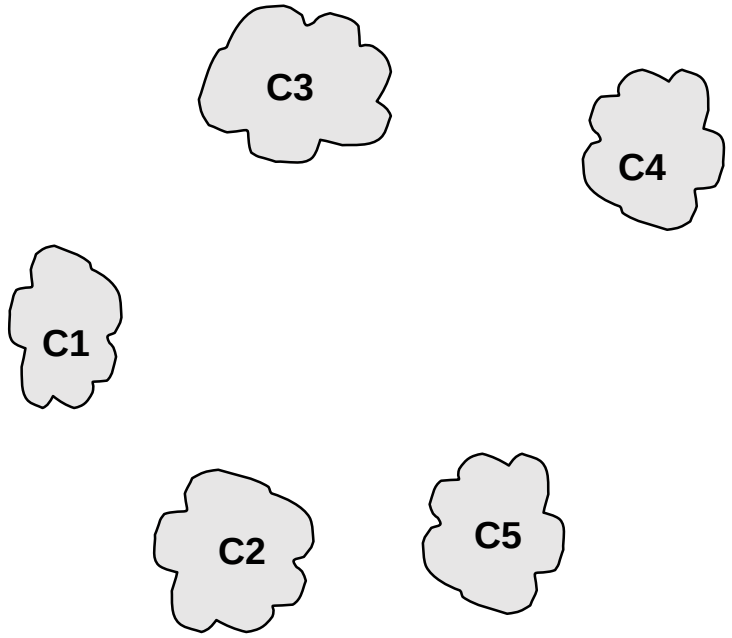
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

邻近度矩阵

3.凝聚层次聚类

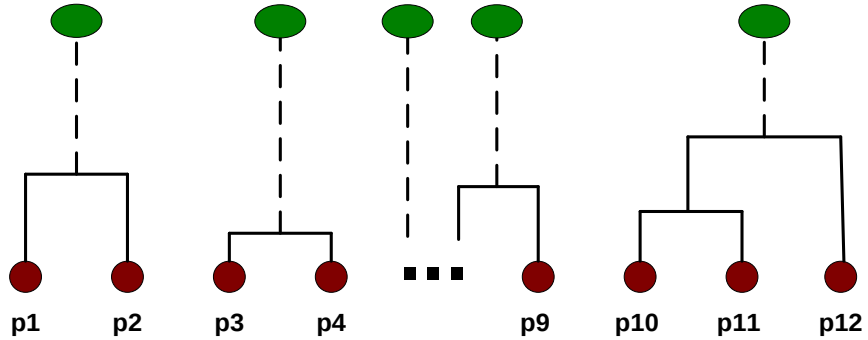
2. 基本凝聚层次聚类算法

■ 经过一些合并步骤后。



	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

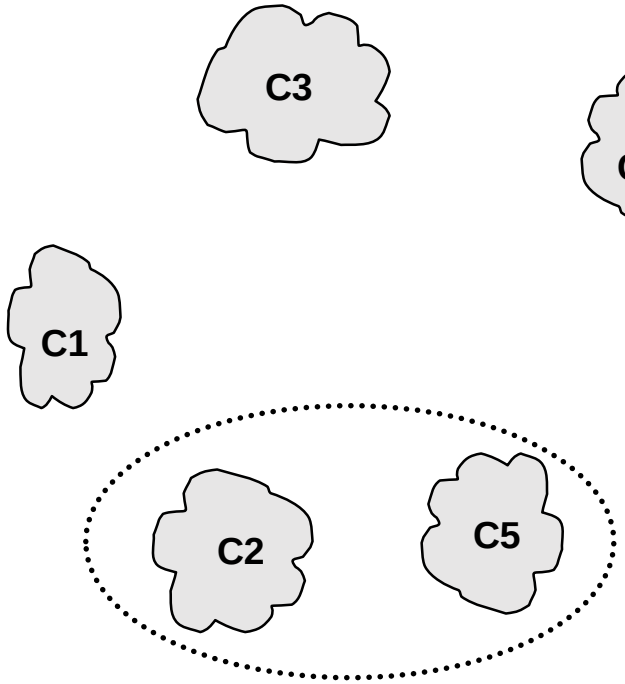
邻近度矩阵

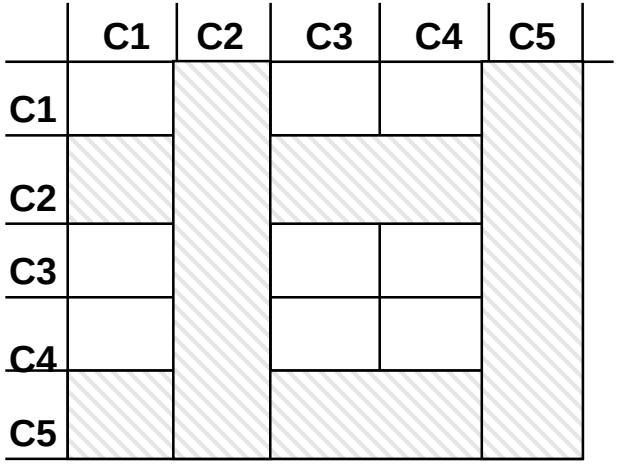


3.凝聚层次聚类

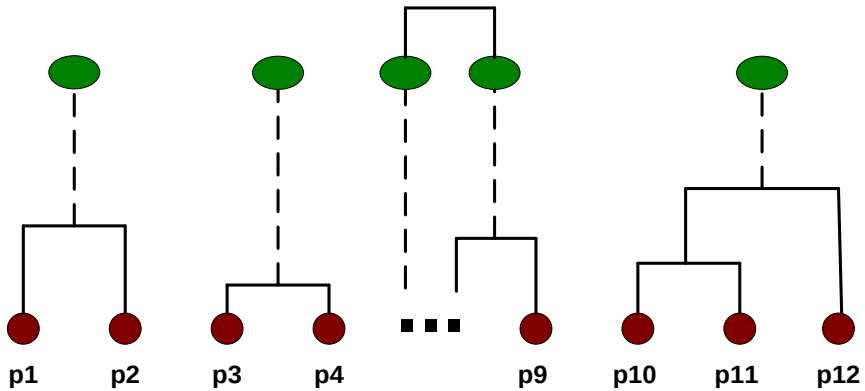
2. 基本凝聚层次聚类算法

■ 合并两个最近的簇（C2和C5）并更新邻近度矩阵。





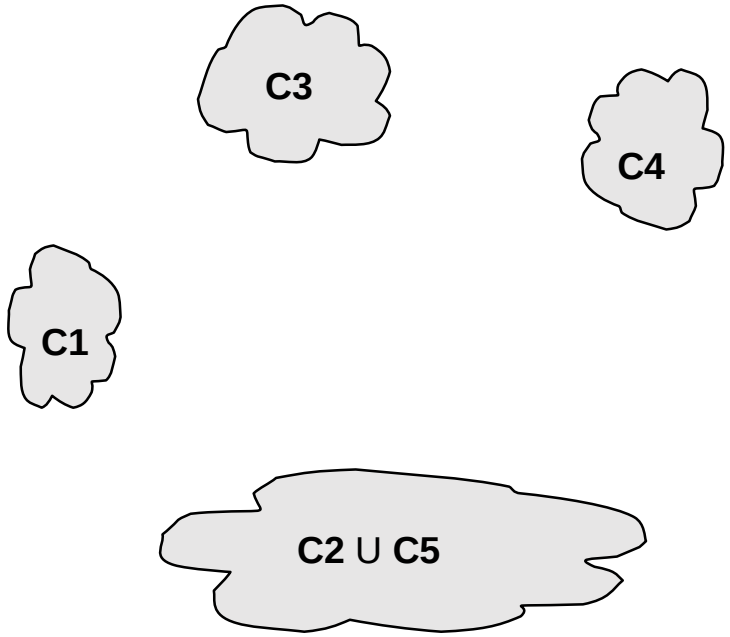
邻近度矩阵



3.凝聚层次聚类

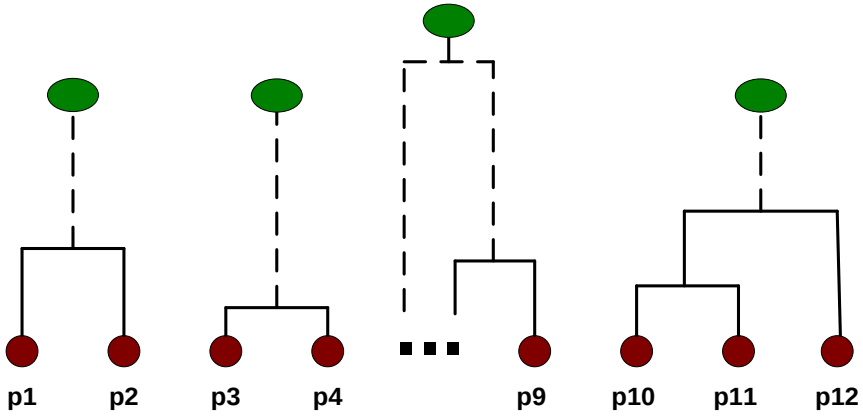
2. 基本凝聚层次聚类算法

■ 问题是“如何更新邻近度矩阵？”



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

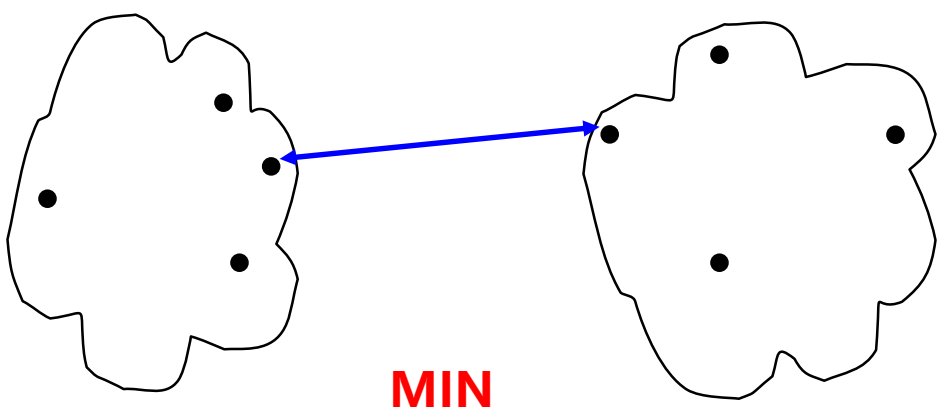
邻近度矩阵

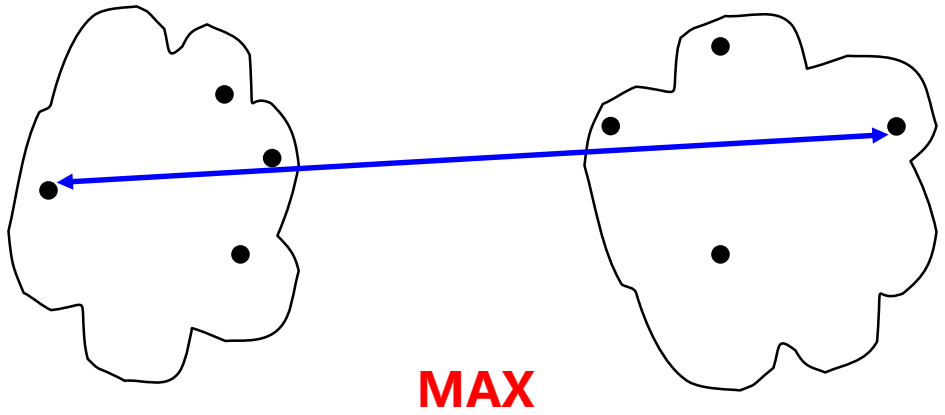


3.凝聚层次聚类

2. 基本凝聚层次聚类算法 --- 定义簇之间的邻近度

- MIN(单链) 定义簇的邻近度为不同簇的两个最近的点之间的邻近度;或者使用图的术语表达为不同的结点子集中两个结点之间的最短边。
- MAX(全链) 取不同簇中两个最远的点之间的邻近度作为簇的邻近度;或者使用图的术语表达为不同的结点子集中两个结点之间的最长边。

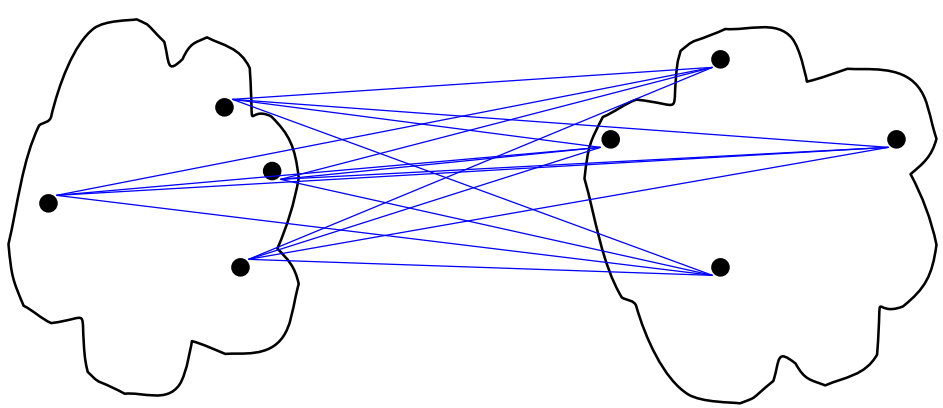




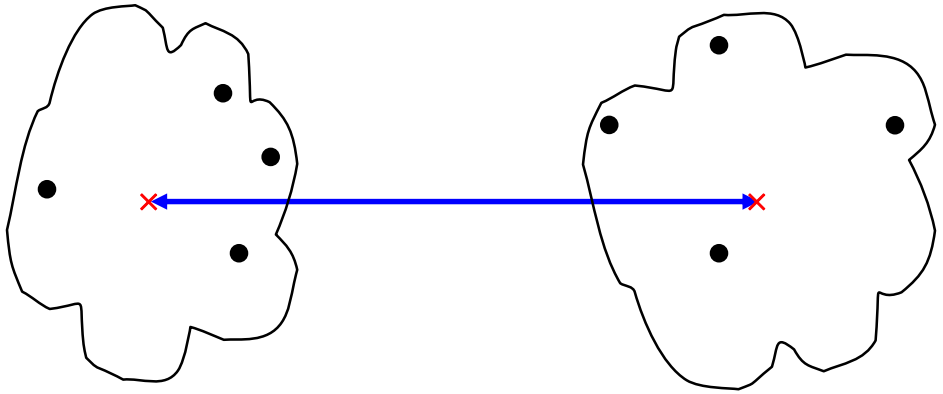
3.凝聚层次聚类

2. 基本凝聚层次聚类算法 --- 定义簇之间的邻近度

- 组平均(group average)技术。它将取自不同簇的所有点对邻近度的平均值(平均边长)定义为簇的邻近度。
- 质心间距离（Distance Between Centroids）：簇的邻近度定义为簇质心之间的邻近度。



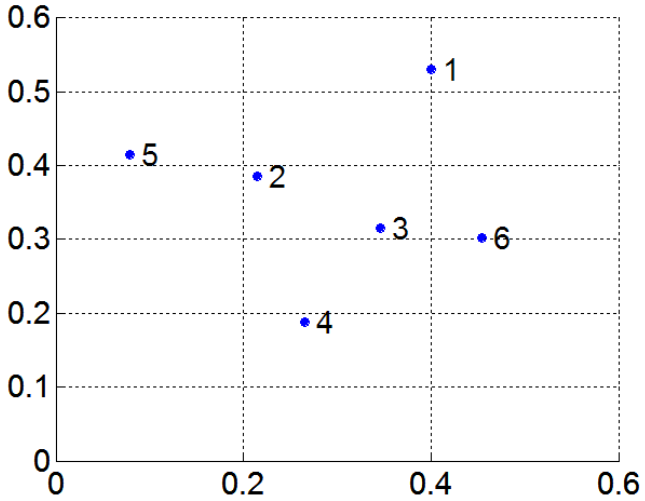
组平均



质心间距离

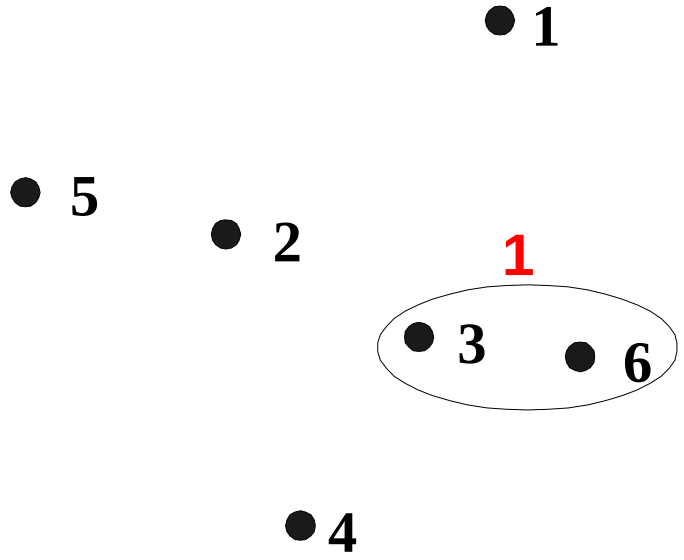
3.凝聚层次聚类

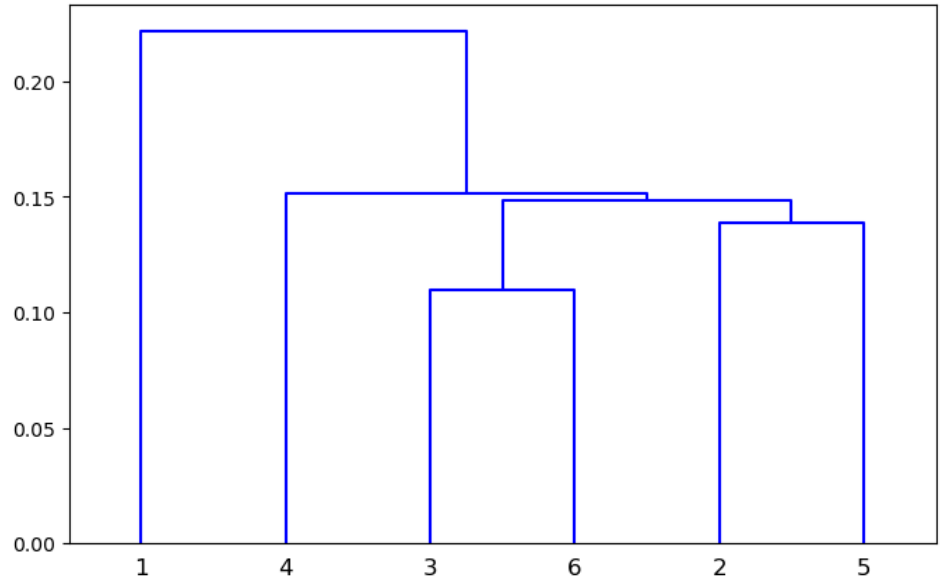
3. 具体例子 --- MIN



	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

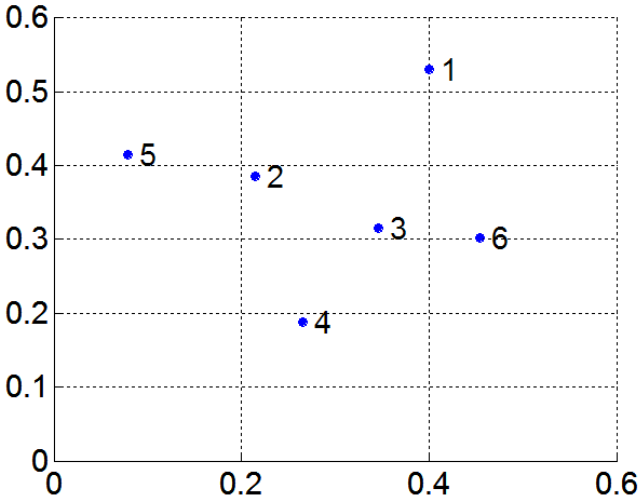
距离矩阵





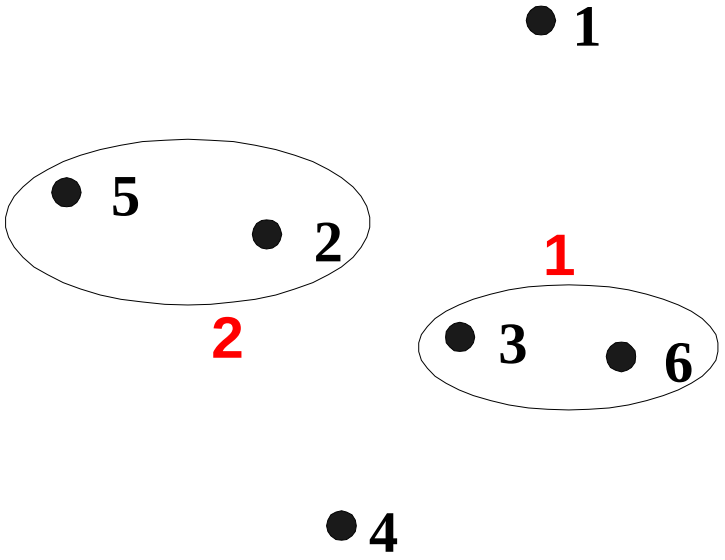
3.凝聚层次聚类

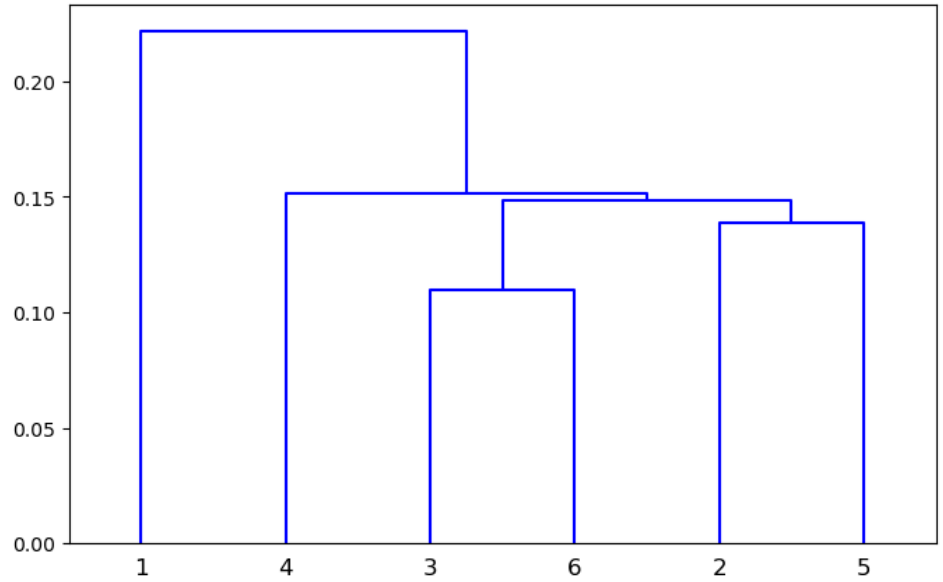
3. 具体例子 --- MIN



	p1	p2	p3p6	p4	p5
p1	0.00	0.24	0.22	0.37	0.34
p2	0.24	0.00	0.15	0.20	**0.14**
p3p6	0.22	0.15	0.00	0.15	0.28
p4	0.37	0.20	0.15	0.00	0.29
p5	0.34	**0.14**	0.28	0.29	0.00

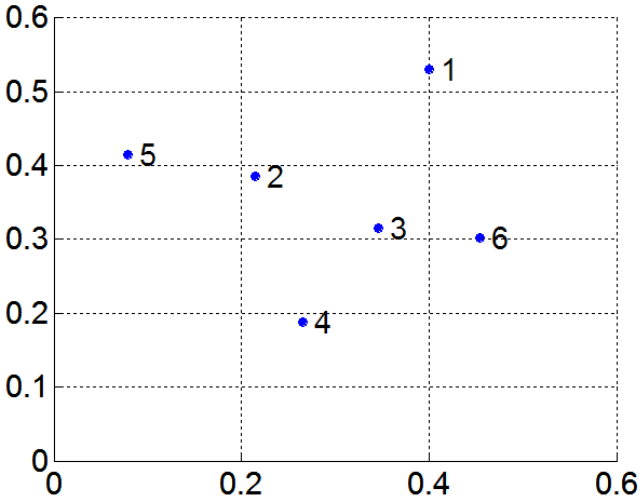
距离矩阵





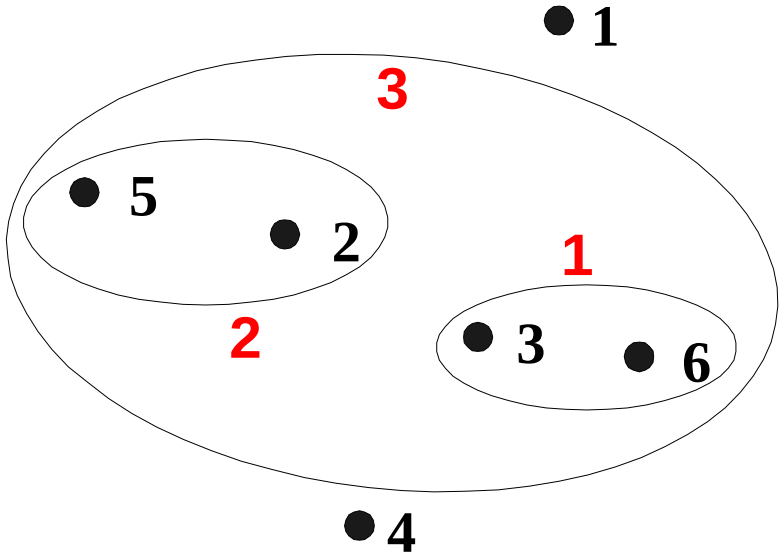
3.凝聚层次聚类

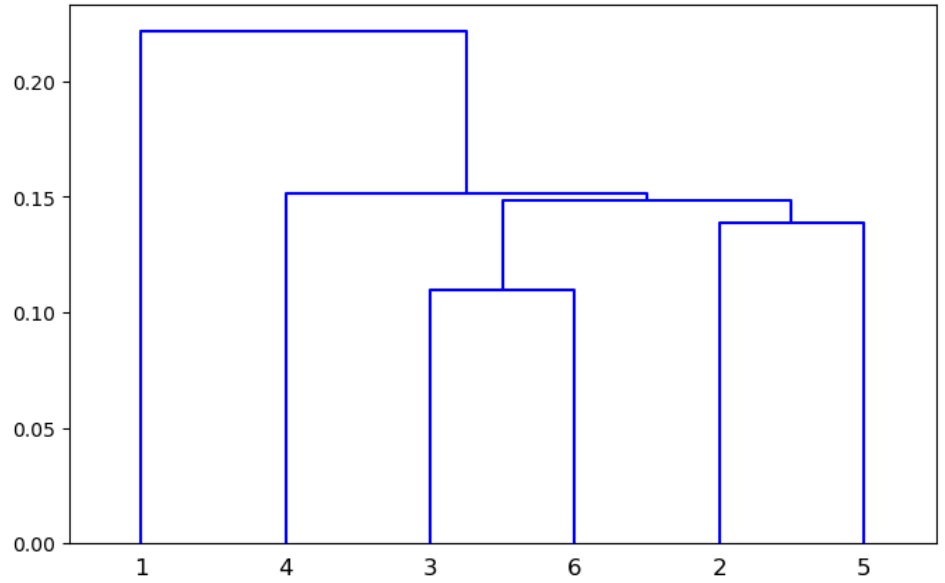
3. 具体例子 --- MIN



	p1	p2p5	p3p6	p4
p1	0.00	0.24	0.22	0.37
p2p5	0.24	0.00	**0.15**	0.20
p3p6	0.22	**0.15**	0.00	0.15
p4	0.37	0.20	0.15	0.00

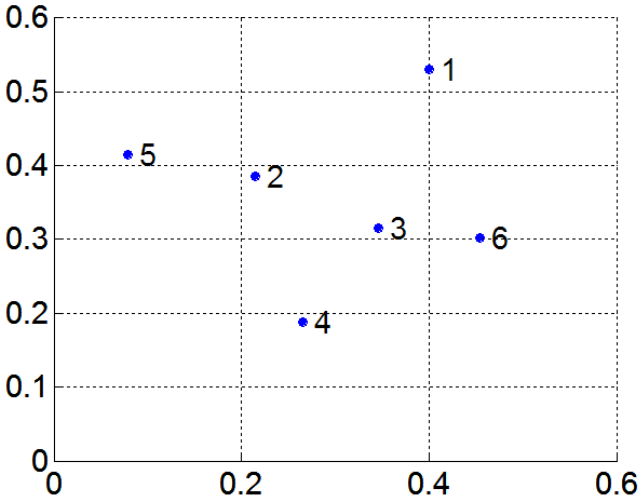
距离矩阵





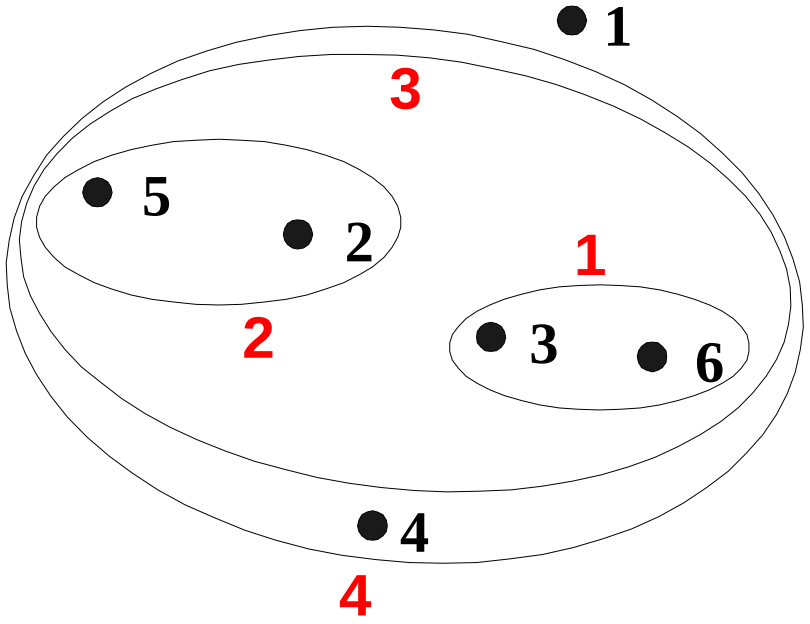
3.凝聚层次聚类

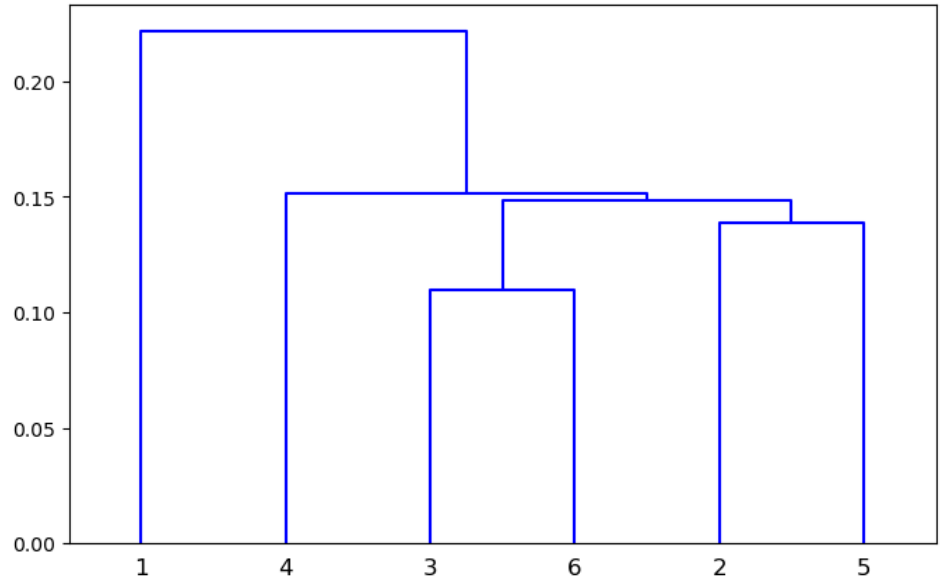
3. 具体例子 --- MIN



	p1	p2p3p5p6	p4
p1	0.00	0.22	0.37
p2p3p5p6	0.22	0.00	0.15
p4	0.37	0.15	0.00

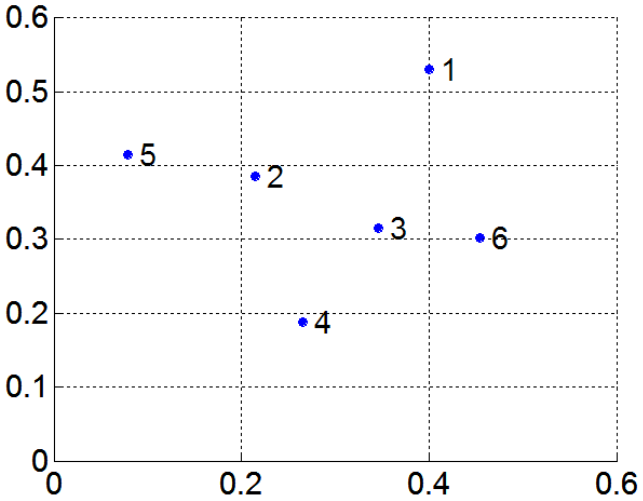
距离矩阵





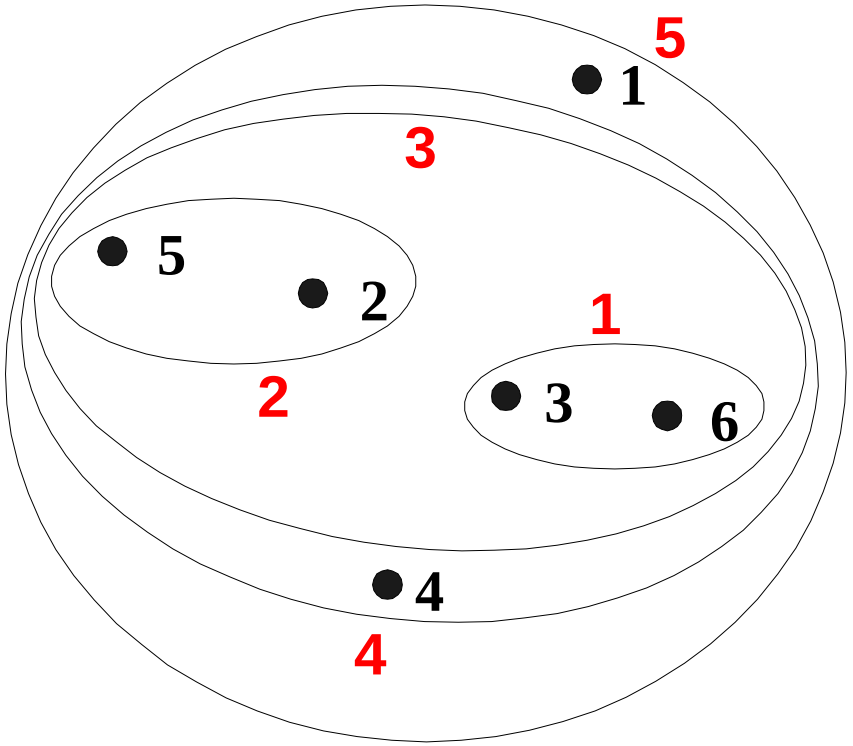
3.凝聚层次聚类

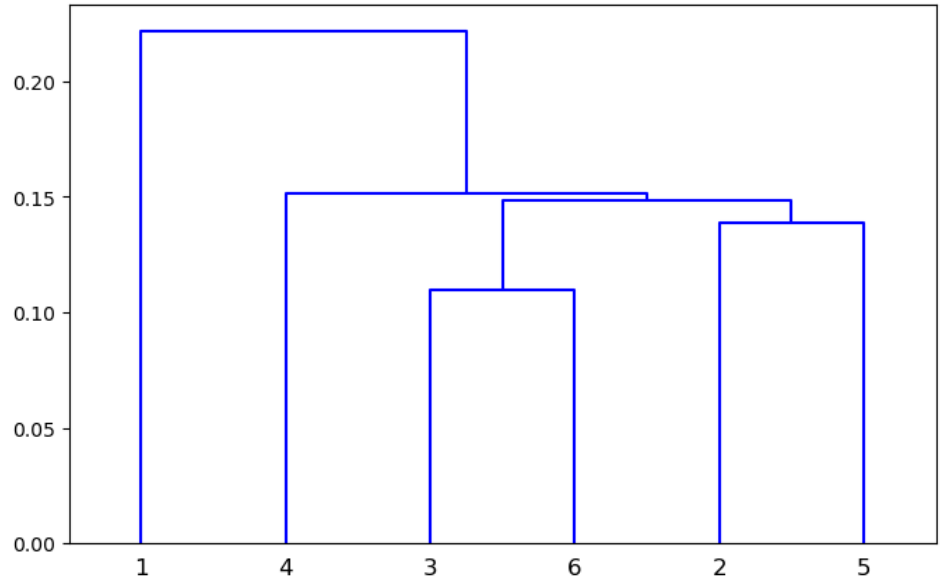
3. 具体例子 --- MIN



	p1	p2p3p4p5p6
p1	0.00	0.22
p2p3p4p5p6	0.22	0.00

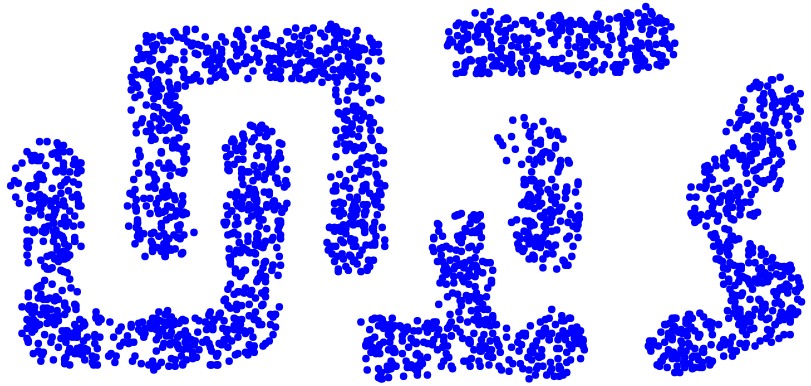
距离矩阵



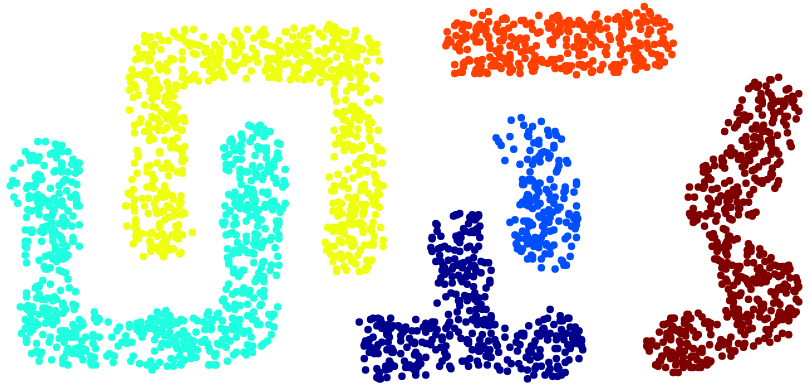


3.凝聚层次聚类

3. 具体例子 --- MIN



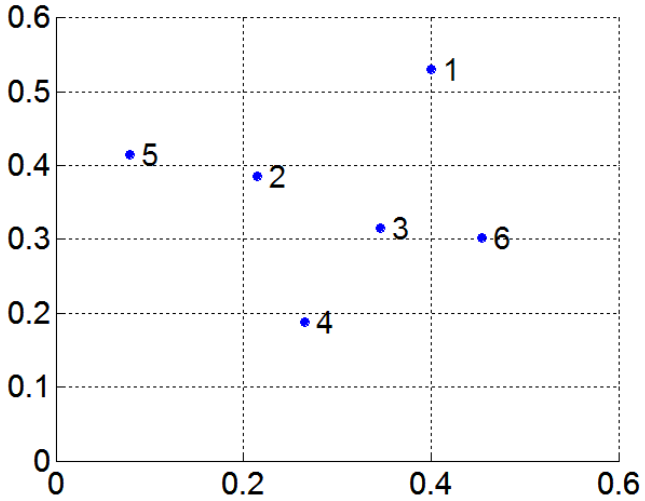
原始点



六个簇

3.凝聚层次聚类

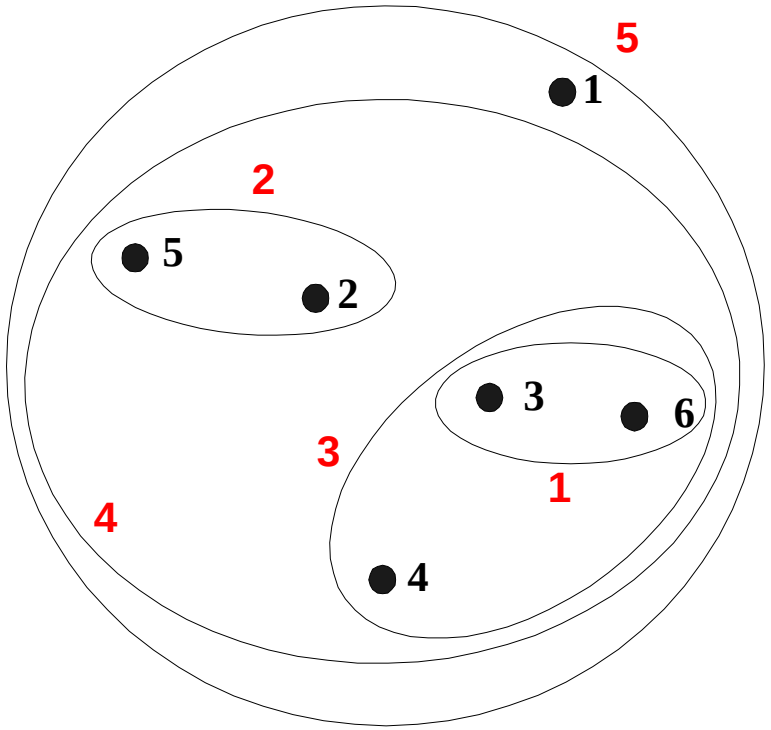
3. 具体例子 --- 组平均

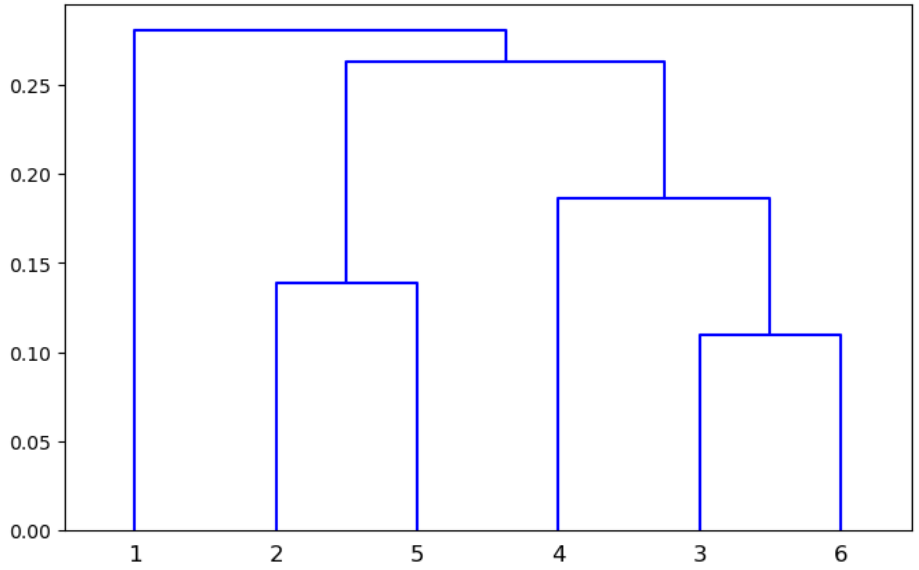


距离矩阵

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

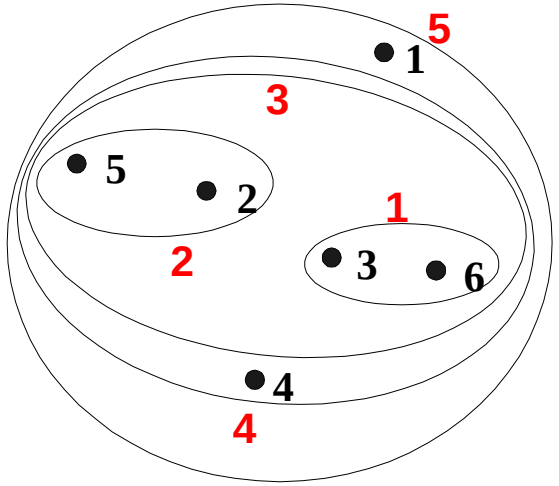
$$proximity(Cluster_i, Cluster_j) = \frac{\sum_{\substack{p_i \in Cluster_i \\ p_j \in Cluster_j}} proximity(p_i, p_j)}{|Cluster_i| \times |Cluster_j|}$$



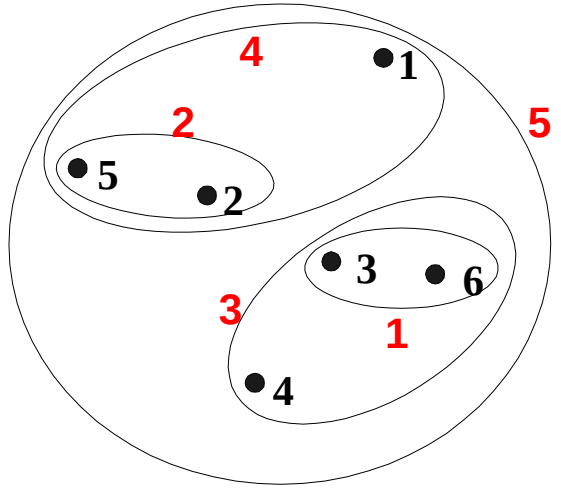


3.凝聚层次聚类

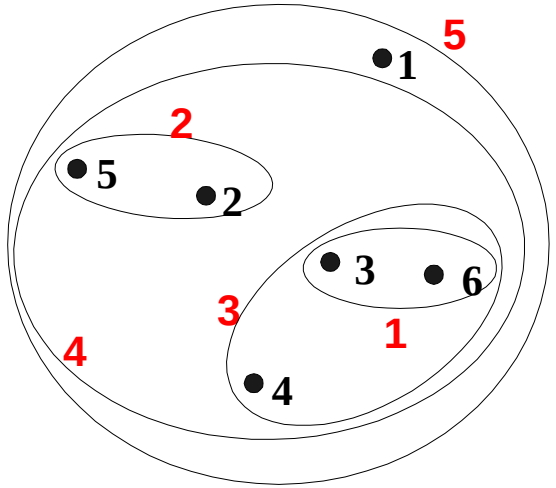
MIN



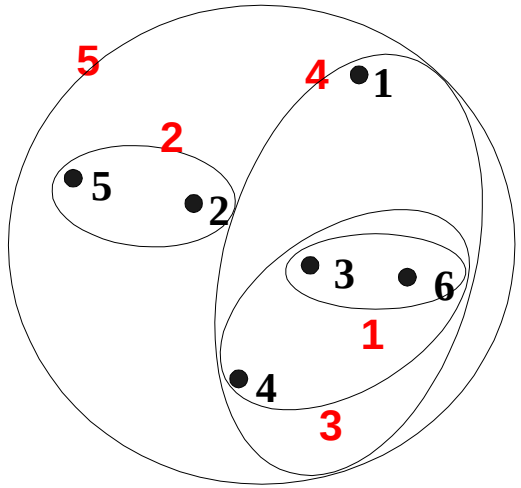
MAX



组平均



Ward



Ward方法：两个簇的邻近度定义为两个簇合并时导致的平方误差的增量。

目 录

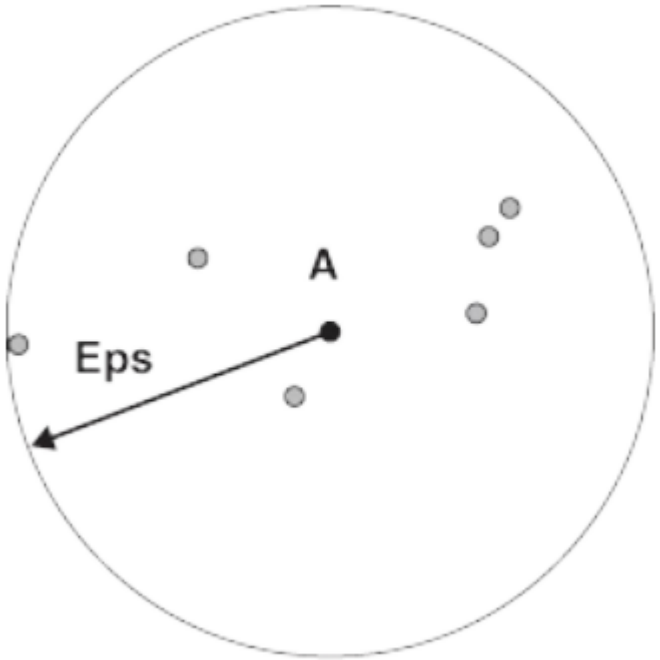
4.DBSCAN

1. 基本概念

■ 基于密度的聚类寻找被低密度区域分离的高密度区域。

2. 传统的密度:基于中心的方法

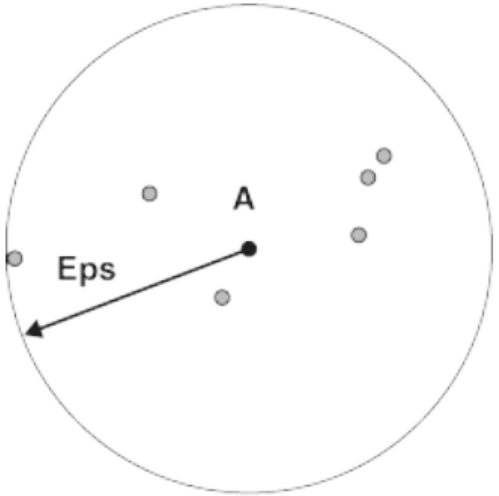
■ 在基于中心的方法中，通过计算数据集中特定点的指定半径（Eps）内的点的数量来估计该点的密度。



点A的 Eps 半径内的点个数为7，包括A本身。

2. 传统的密度:基于中心的方法

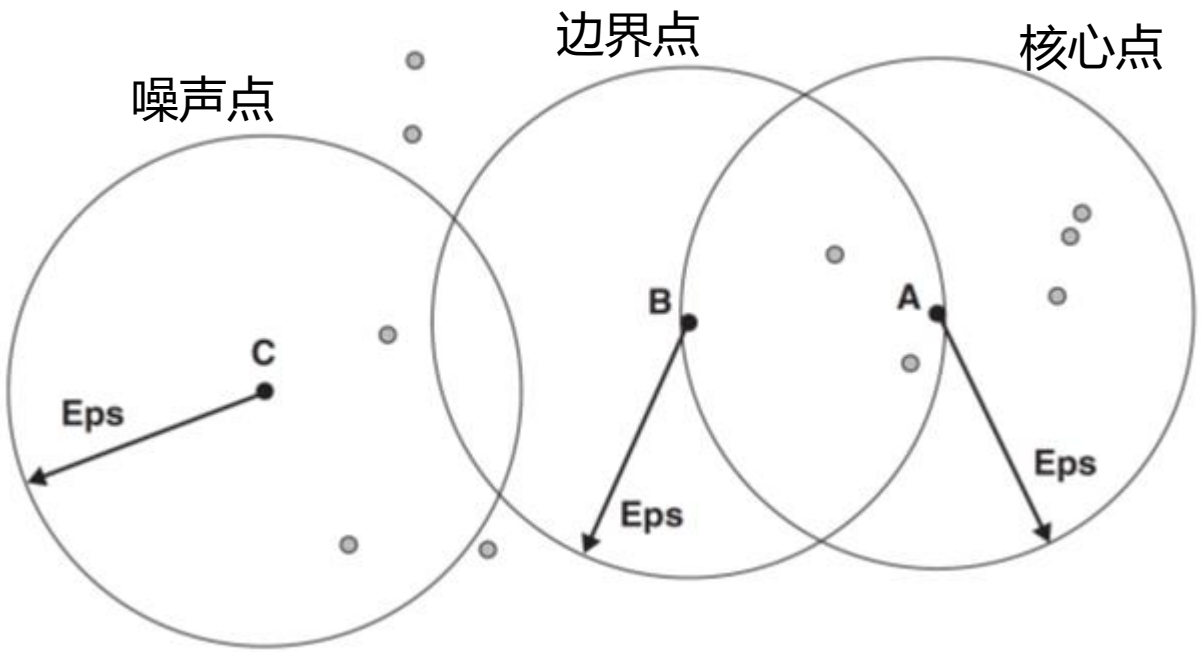
- 该方法实现简单，但是点的密度取决于指定的半径。
- 如果半径足够大，则所有点的密度都等于数据集中的点数 m。
- 如果半径太小，则所有点的密度都是 1。



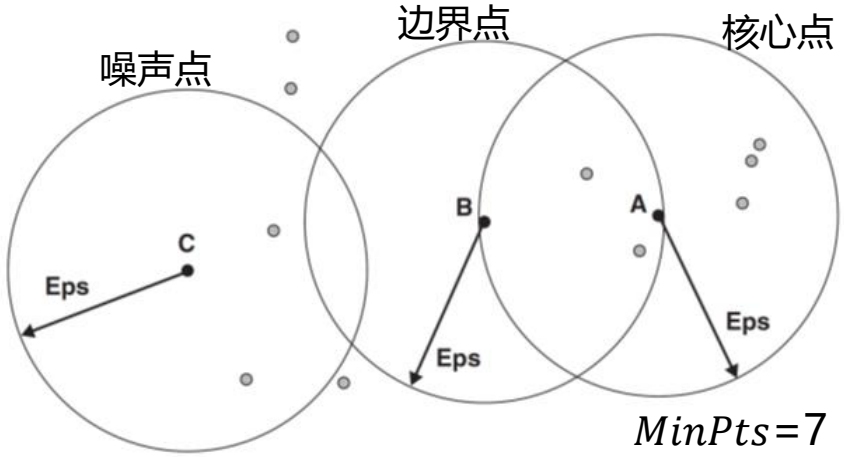
点A的 Eps 半径内的点个数为7，包括A本身。

2. 传统的密度:基于中心的方法

■ 基于中心的方法使得我们可以将点分类为：(1)稠密区域内部的点(核心点)；(2)稠密区域边缘上的点(边界点)；(3)稀疏区域中的点(噪声或背景点)。

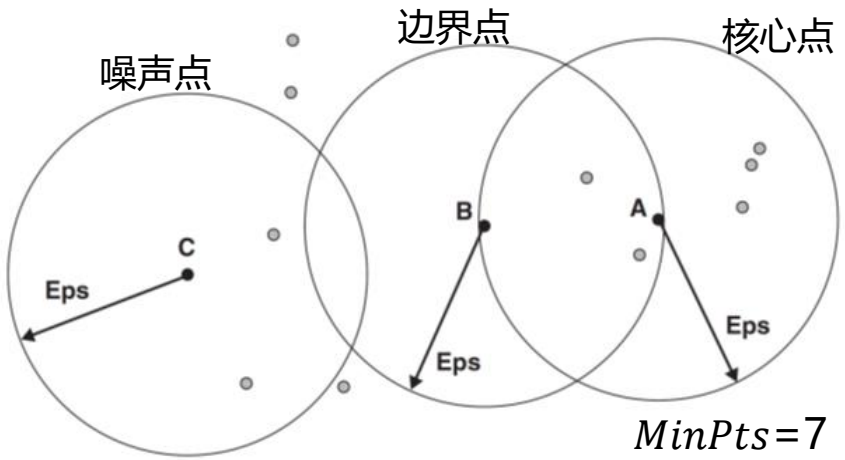


4.DBSCAN



2. 传统的密度:基于中心的方法

- 核心点(core point)：这些点在基于密度的簇内部。如果在距离该点 Eps 的范围内至少有 $MinPts$ 个点，则该点为核心点，其中 Eps、$MinPts$ 均为用户指定参数。例如：点A是核心点。
- 边界点(border point)：边界点不是核心点，但它落在某个核心点的邻域内。边界点可能落在多个核心点的邻域内。例如：点B是边界点。
- 噪声点(noise point)：噪声点是既非核心点也非边界点的任何点。例如：点C是噪声点。

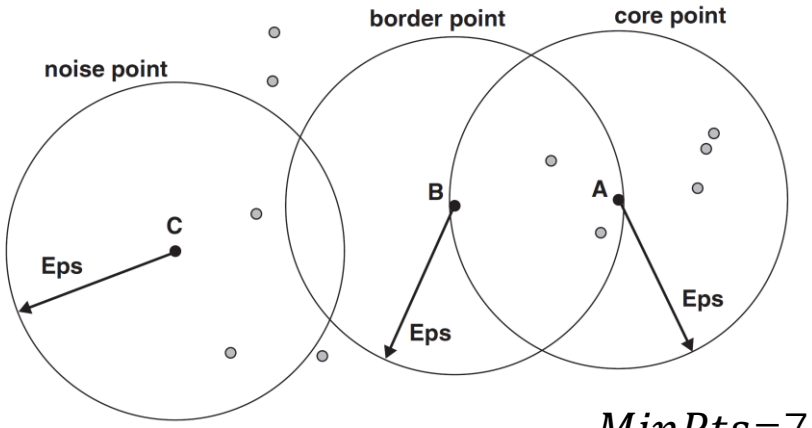


3. DBSCAN算法

■ 任意两个足够靠近(相互之间的距离在 Eps 之内)的核心点将放在同一个簇中。同样，任何与核心点足够靠近的边界点也放到与核心点相同的簇中。 如果一个边界点靠近不同簇的核心点，则可能需要解决平局问题。噪声点被丢弃。

算法7.5 DBSCAN算法	
1:	将所有点标记为核心点、边界点和噪声点
2:	删除噪声点
3:	为距离在 Eps 之内的所有核心点之间赋予一条边
4:	每组连通的核心点形成一个簇
5:	将每个边界点指派到一个与之关联的核心点的簇中

4.DBSCAN



$MinPts=7$

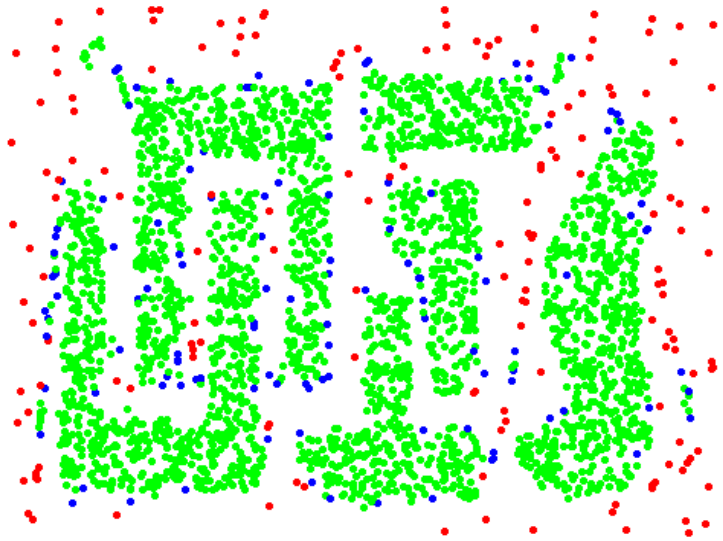
4. 时间和空间复杂度

- 基本时间复杂度是 $O(m \times$找出 Eps 邻域中的点所需要的时间)，其中m是点的个数。在最坏情况下，时间复杂度是 $O(m^2)$。
- 在低维空间(尤其是二维空间)，有一些数据结构，如 kd树，可以有效地检索特定点在给定距离内的所有点，因此平均时间复杂度可以降低到 $O(mlogm)$。
- 即便对于高维数据，DBSCAN的空间复杂度也是 $O(m)$，因为对每个点，它只需要维持少量数据，即簇标签和每个点是核心点、边界点还是噪声点的标识。

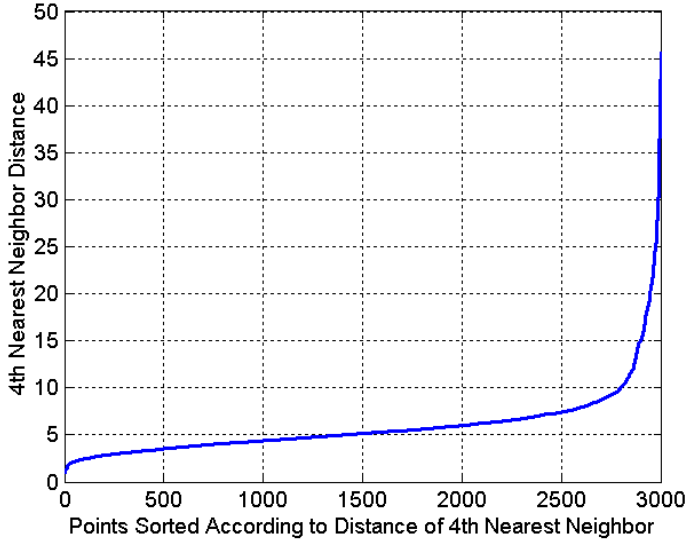
5. 例子

- 该数据集包含3000 个二维点。该数据的 Eps 值通过对每个点到其第4个最近邻的距离排序绘图并识别急剧变化处的值来确定。选取 Eps=10，对应于曲线的拐点。 $(Eps = 10, MinPts = 4)$

原始点



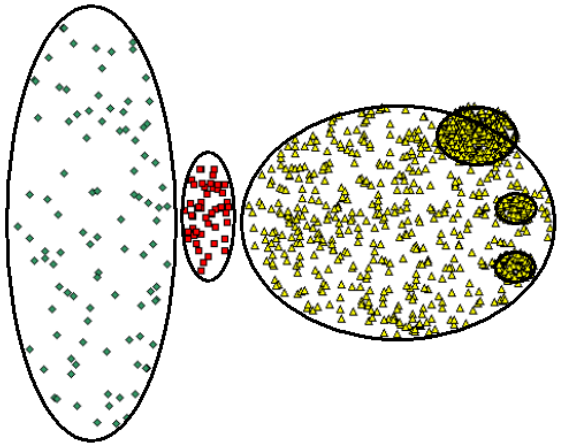
点类型: 核心, 边界, 噪声



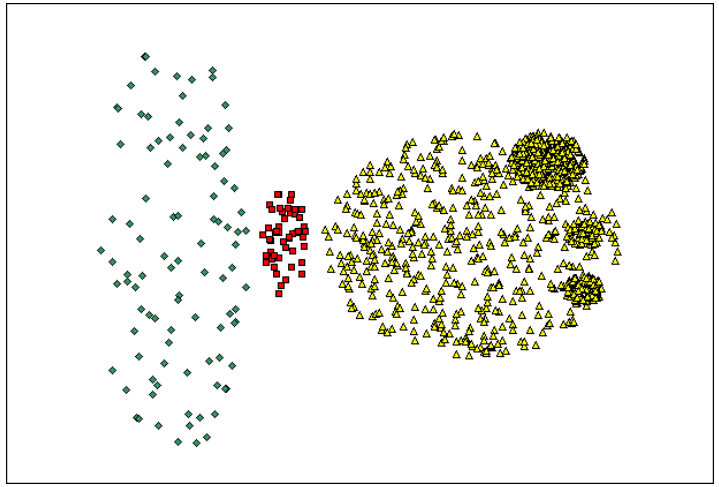
6. 优点与缺点

- 因为DBSCAN基于密度定义簇，因此它相对来说能抗噪声，并且能够处理任意形状和大小的簇。这样，DBSCAN可以发现许多使用K均值不能发现的簇，例如上页中的那些簇。
- 当簇的密度变化太大时，DBSCAN就会遇到麻烦。对于高维数据，它也会出现问题，因为对于这样的数据，更难定义密度。
- 当近邻计算需要计算所有的点对邻近度时(对于高维数据，常常如此)，DBSCAN的开销可能是很大的。

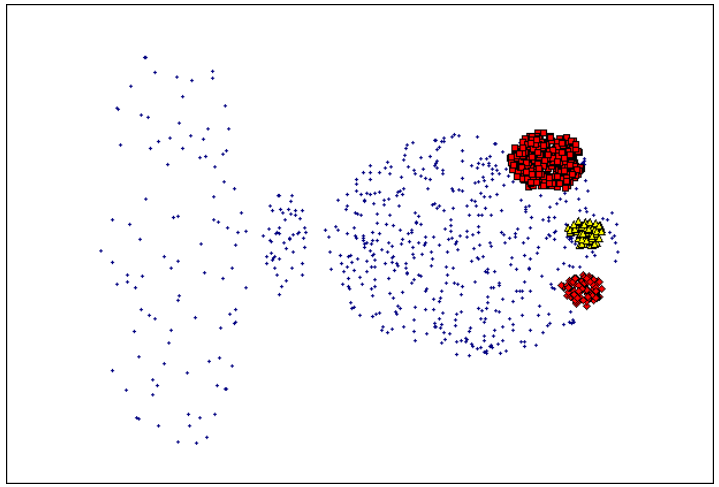
6. 优点与缺点



原始点



(MinPts=4, Eps=9.92)



(MinPts=4, Eps=9.75)

目 录

5.簇评估

1. 概述

- 能够识别数据中是否存在非随机结构正是簇验证的重要任务之一。下面列举了簇验证的一些重要问题：
- 1) 确定数据集的聚类趋势(clustering tendency)，即识别数据中是否实际存在非随机结构。
- 2) 确定正确的簇个数。
- 3) 不引用附加的信息，评估聚类分析结果对数据拟合的情况。
- 4) 将聚类分析结果与已知的客观结果(如，外部提供的类别标签)比较。
- 5) 比较两个簇集，确定哪个更好。

不使用外部信息

可以有监督或无监督

5.簇评估

1. 概述

- 用于评估簇的各方面的度量或指标一般分成如下3类。
- 无监督的。聚类结构的优良性度量，不考虑外部信息，例如，SSE。
- 簇的有效性的无监督度量常常可以进一步分成两类：
  - 簇的凝聚性(cluster cohesion)(紧凑性、紧致性)，该度量确定簇中对象如何密切相关;
  - 簇的分离性(cluster separation)(孤立性)，该度量确定某个簇不同于其他簇的地方。
- 无监督度量通常称为内部指标(internal index)，因为它们仅使用出现在数据集中的信息。

5.簇评估

1. 概述

- 用于评估簇的各方面的度量或指标一般分成如下3类。
- 有监督的。度量聚类算法发现的聚类结构与某种外部结构的匹配程度。
  - 例如，监督指标熵，它度量簇标签与外部提供的标签的匹配程度。
- 有监督度量通常称为外部指标(external index)，因为它们使用了不在数据集中出现的信息。

5.簇评估

1. 概述

- 用于评估簇的各方面的度量或指标一般分成如下3类。
- 相对的。比较不同的聚类或簇。相对簇评估度量是用于比较的监督或无监督评估度量。因而，相对度量实际上不是一种单独的簇评估度量类型，而是度量的一种具体使用。例如，两个K均值聚类可以使用 SSE 或熵进行比较。

5.簇评估

2. 无监督簇评估:使用凝聚度和分离度

- 簇凝聚度：衡量簇中对象的紧密关系。
  - 例如：SSE，越小表示聚类效果越好

$$SSE = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$$

- 簇分离度：测量簇与其他簇的区别或分离程度。
  - 例如：组平方和(SSB)，SSB越高，簇之间的分离性越好

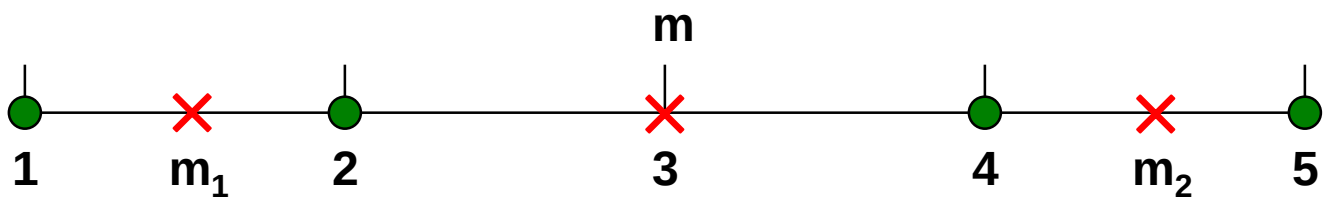
$$SSB = \sum_{i} |C_i|(m - m_i)^2$$

  - 簇质心m_i到所有数据点的总均值m的距离的平方和
  - $|C_i|$ 表示簇i的大小

5.簇评估

$$SSE = \sum_{i} \sum_{x \in C_i} (x - m_i)^2 \qquad SSB = \sum_{i} |C_i| (m - m_i)^2$$

2. 无监督簇评估:使用凝聚度和分离度



K=1 : $SSE = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$

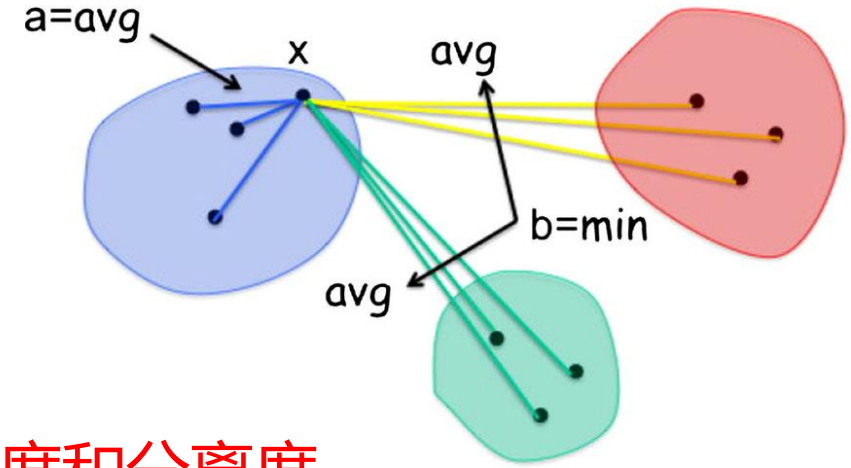
$SSB = 4 \times (3-3)^2 = 0$

K=2 : $SSE = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$

$SSB = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$

可以证明总SSE和总SSB之和是一个常数，最小化SSE(凝聚度)等价于最大化SSB(分离度)。

5.簇评估

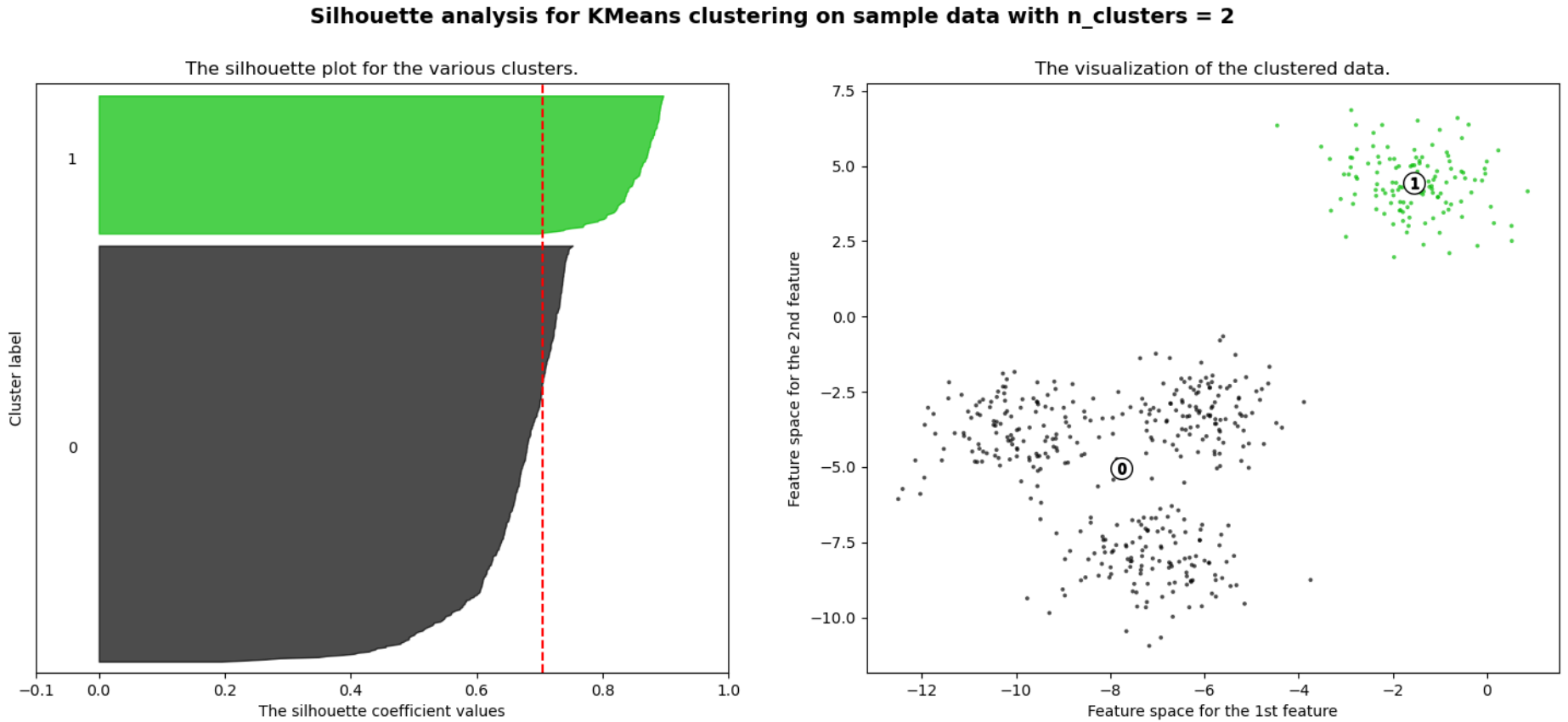


3. 轮廓系数

- 轮廓系数(silhouette coefficient)方法结合了凝聚度和分离度。
- 计算个体点的轮廓系数过程由如下3步组成。
- 1)对于第i个对象，计算它到簇中所有其他对象的平均距离。该值记作a_i。
- 2)对于第i个对象和不包含该对象的任意簇，计算该对象到给定簇中所有对象的平均距离。找出最小值。该值记作 b_i。
- 3)对于第i个对象，轮廓系数是$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$
- 轮廓系数的值在-1和1之间变化。越接近1，说明聚类效果越好。

5.簇评估

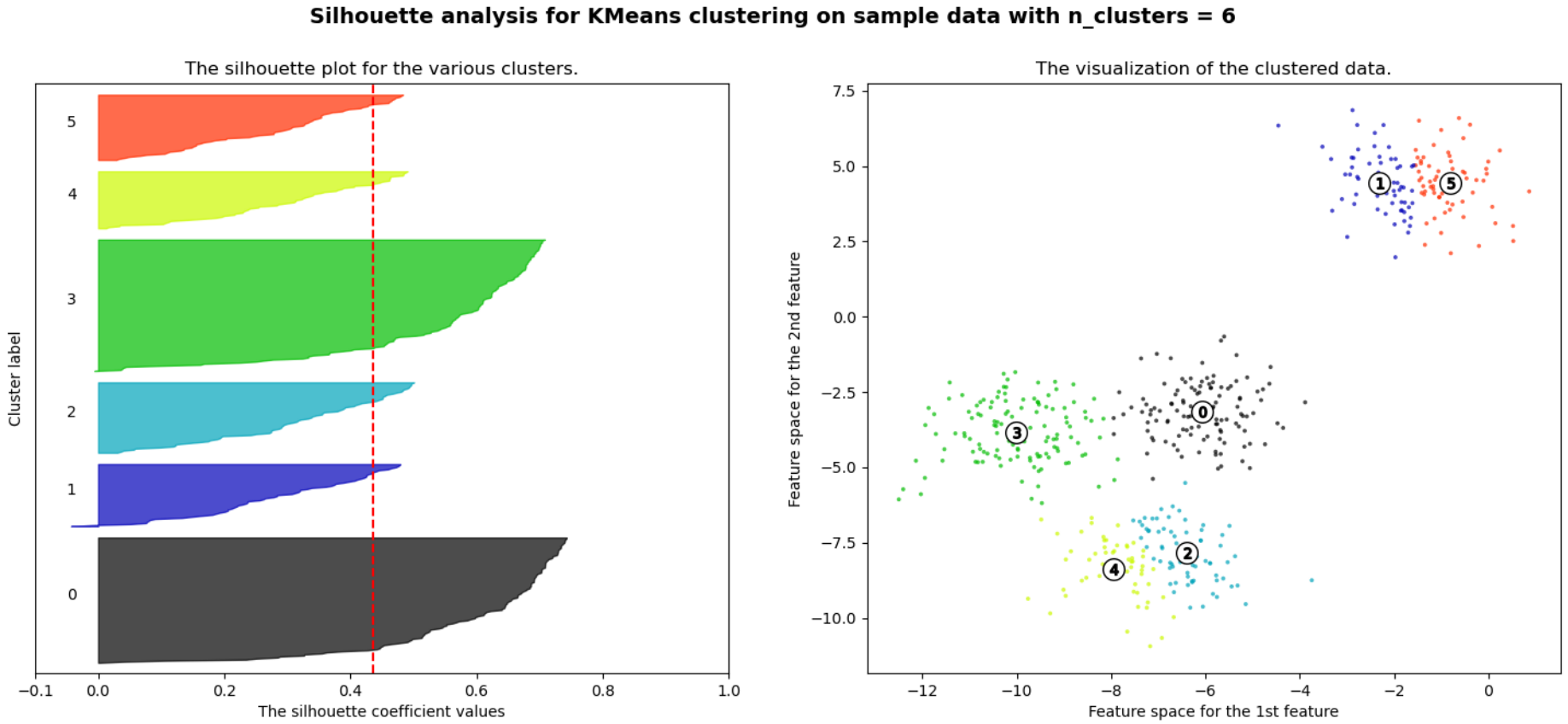
3. 轮廓系数



平均值=0.7049787496083262

5.簇评估

3. 轮廓系数



平均值=0.4358297989156284

5.簇评估

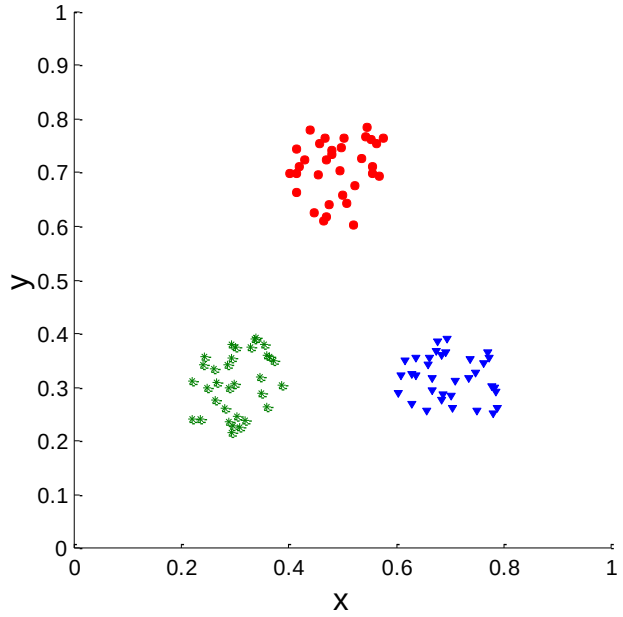
4. 无监督簇评估:使用邻近度矩阵

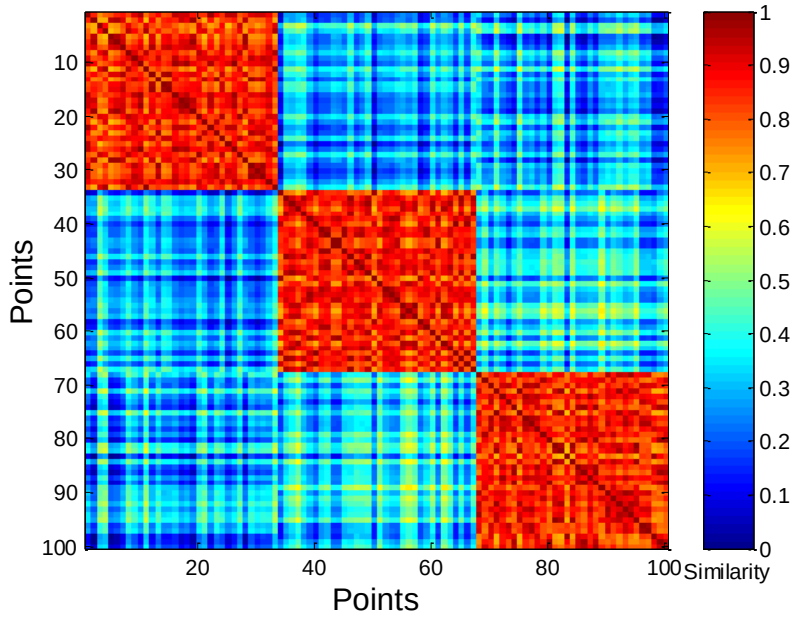
- 如果给定数据集的相似度矩阵和数据集聚类分析得到的簇标签，则可以通过考察相似度矩阵和基于簇标签的相似度矩阵的理想版本之间的相关性来评估聚类的“优良性”。
- 对于理想的簇，它包含的点与簇内所有点的相似度为 1，而与其他簇中的所有点的相似度为0。这样，如果将相似度矩阵的行和列排序，使得属于相同簇的对象在一起，则理想的相似度矩阵具有块对角(block diagonal)结构。
- 换言之，在相似度矩阵中代表簇内相似度的项的块内部相似度非零(为 1)，而其他为 0。
- 理想的相似度矩阵可以通过如下方法构造：创建一个矩阵，每个数据点一行一列(与实际的相似度矩阵类似)，如果它所关联的一对点属于同一个簇，矩阵的一个项为 1。其他项均为 0。

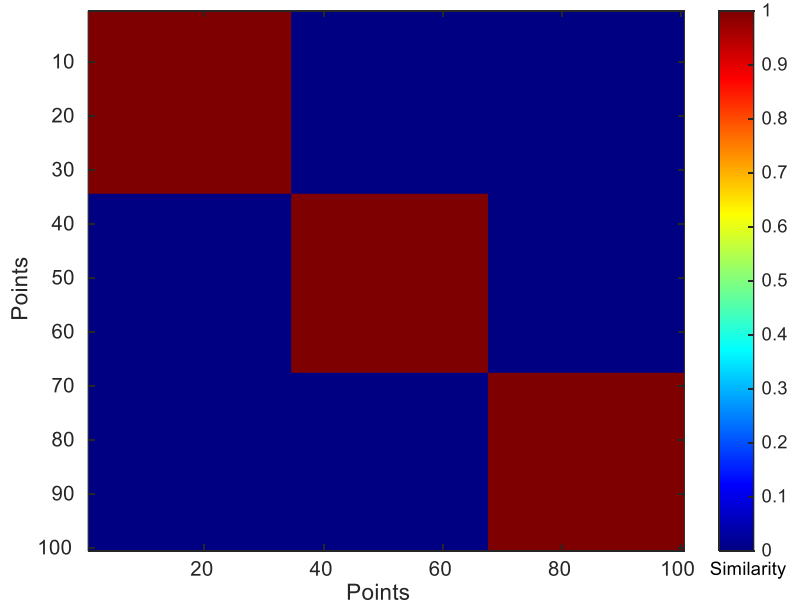
5.簇评估

4. 无监督簇评估:使用邻近度矩阵

- 理想的和实际的相似度矩阵之间的高度相关表明了属于同一个簇的点相互很接近，而二者间的低相关性表明相反的情况。





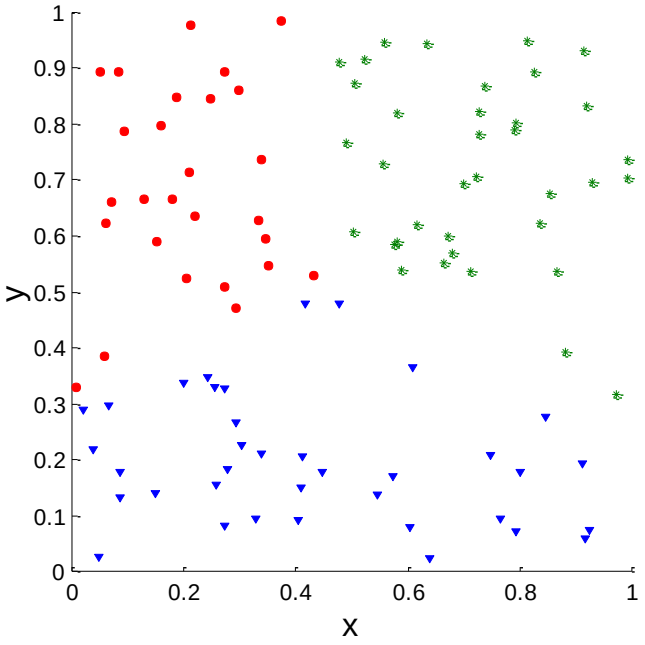


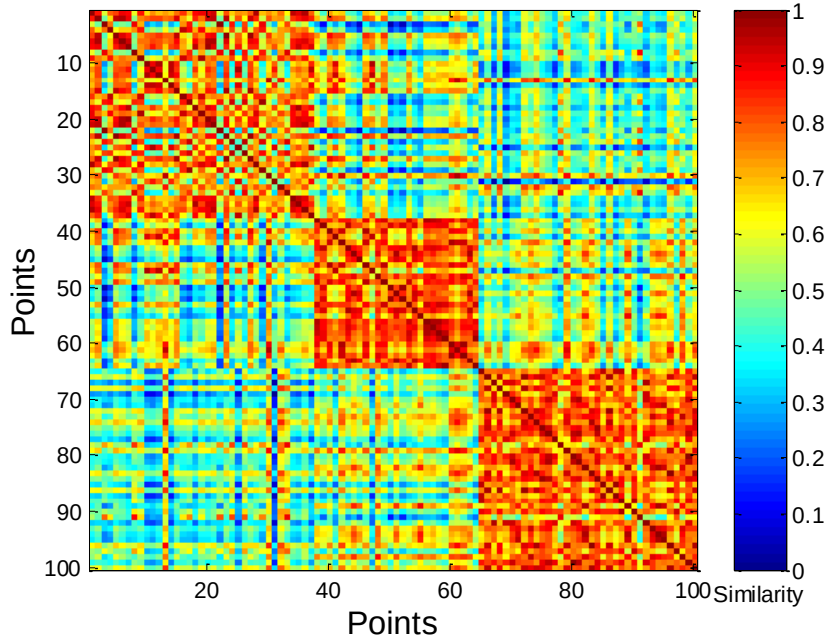
Corr = 0.9235

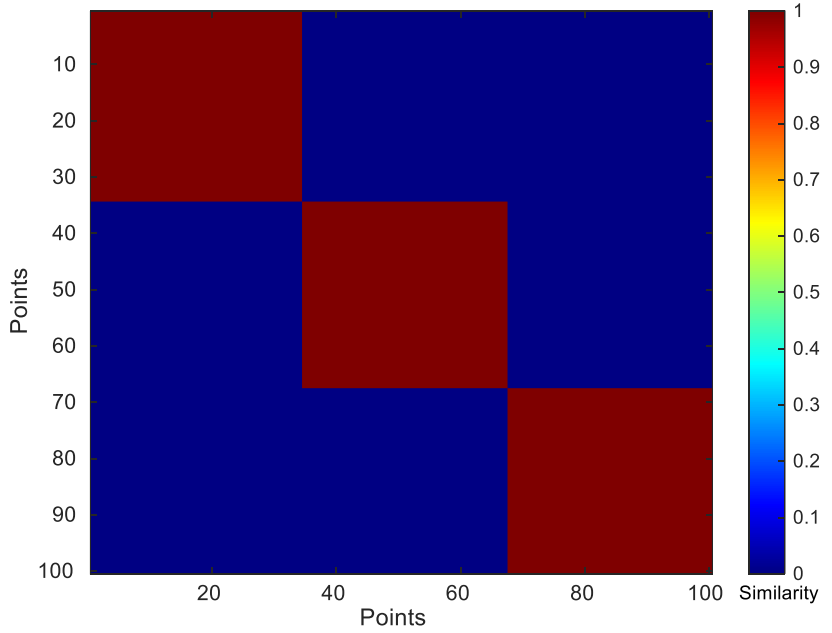
5.簇评估

4. 无监督簇评估:使用邻近度矩阵

- 理想的和实际的相似度矩阵之间的高度相关表明了属于同一个簇的点相互很接近，而二者间的低相关性表明相反的情况。







Corr = 0.5810