



第五章 聚类分析

目录 CONTENTS

- 5.1 聚类的基本概念
- 5.2 基于划分的聚类方法
- 5.3 基于层次的聚类方法
- 5.4 基于密度的聚类方法
- 5.5 基于网格的聚类方法
- 5.6 聚类效果评估



Chapter 5.1

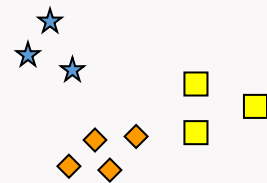
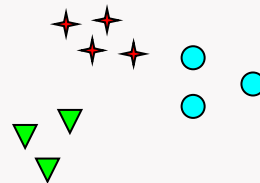
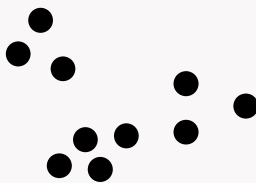
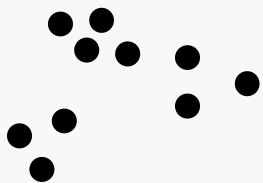
聚类的基本概念

□ 聚类分析

- **聚类分析** (Cluster analysis) 简称**聚类** (Clustering) , 是一个把一组数据对象划分到不同子集的过程。每个子集称作一个**簇** (Cluster) , 属于同一个簇中的数据对象彼此相似, 属于不同簇的数据对象彼此不相似。由聚类分析产生的簇的集合称作一个聚类。

□ 聚类分析的困难

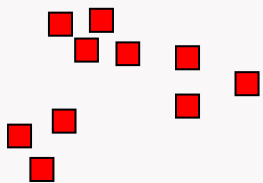
- 因为相似性的不确定性，所以簇的概念并没有客观的定义。



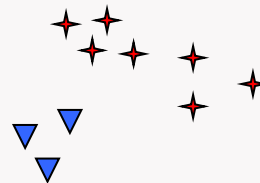
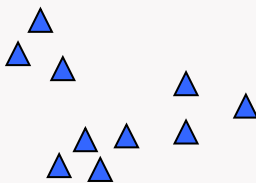
应该划分成几个簇？

6个簇

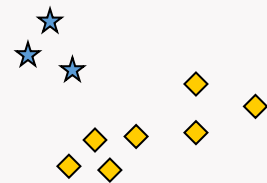
聚类的好坏依赖于数据的特性和期望的结果！



2个簇



4个簇



□ 聚类的类型

- **层次的和划分的**：“聚类是层次的还是划分的？”是最常讨论的问题，两者的区别是簇的集合是否嵌套：
 - 如果将数据集中的数据对象划分到不重叠的簇，使得每个数据对象都存在于一个簇中，则称之为**划分聚类**（Partitional clustering）；
 - 如果允许簇具有子簇，则称之为**层次聚类**（Hierarchical clustering），通常以树的形式表示数据对象之间的关系。

□聚类的类型

- **互斥的、重叠的和模糊的**：数据对象的隶属关系也是区分不同聚类的重要因素：
 - 如果每个数据对象都只属于唯一的簇，则称之为**互斥的**（Exclusive），比如按运动能力对球员进行聚类；
 - 如果某个数据对象可以同时属于多个簇，则称之为**重叠的**（Overlapping）或**非互斥的**（Non-exclusive），比如按兴趣爱好对球员进行聚类；
 - 如果每个数据对象以 $[0,1]$ 之间的隶属权值（隶属度）属于每个簇，则称之为**模糊聚类**（Fuzzy clustering），比如按绘画风格对图片进行聚类。

□ 聚类算法的要求

■ 数据挖掘对聚类算法的要求有以下几点：

- **具有可伸缩性**：许多聚类算法在小型数据集（几百或几千）上都可以运行良好，但是在大型数据集（百万级或十亿级）可能无法取得预期的结果。这种情况下，我们需要具有高度可伸缩性的聚类算法。
- **能够处理不同属性类型**：许多聚类算法都是针对数值属性设计的，然而现实中可能要面临混合着标称属性、序数属性的数据，甚至诸如图、序列、文档等更复杂的数据。这种情况下，我们需要能够以较小代价适应不同类型数据的聚类算法。

□ 聚类算法的要求

■ 数据挖掘对聚类算法的要求有以下几点：

➤ **能够发现任意形状的簇**：许多聚类算法基于欧式距离确定簇，导致得到的簇是球形的。然而，真实分布的簇可能是非球形的，甚至是非凸的。这种情况下，我们需要能够发现任意形状的簇的聚类算法。

□ 聚类算法的要求

■ 数据挖掘对聚类算法的要求有以下几点：

- 能够降低对于输入参数的领域知识的要求
- 能够处理噪声数据
- 对输入数据的顺序不敏感
- 能够进行增量聚类
- 能够处理高维数据
- 能够使聚类结果满足特定的约束
- 具有良好的解释性和可用性



Chapter 5.2

基于划分的聚类方法

□ 划分方法的定义

- 假设数据集 D 中包含 n 个欧式空间中的数据对象，划分方法把 D 中的数据对象分配到 k 个簇 C_1, C_2, \dots, C_k 中，使得对于 $1 \leq i, j \leq k$, $C_i \subset D$ 且 $C_i \cap C_j = \emptyset$ 。然后使用一个目标函数来评估划分的质量，使得簇内的数据对象相似，不同簇间的数据对象相异。

□ K-means算法

- K-means算法是最著名、最常用的基于划分的聚类方法。基本思想是首先创建一个初始 k 划分，然后不断地迭代计算每个簇的聚类中心，然后依据新的聚类中心调整划分情况，直至收敛。
- K-means算法是启发式算法，即每次聚类保证**局部最优**，利用局部最优聚类的上限来不断逼近全局最优。

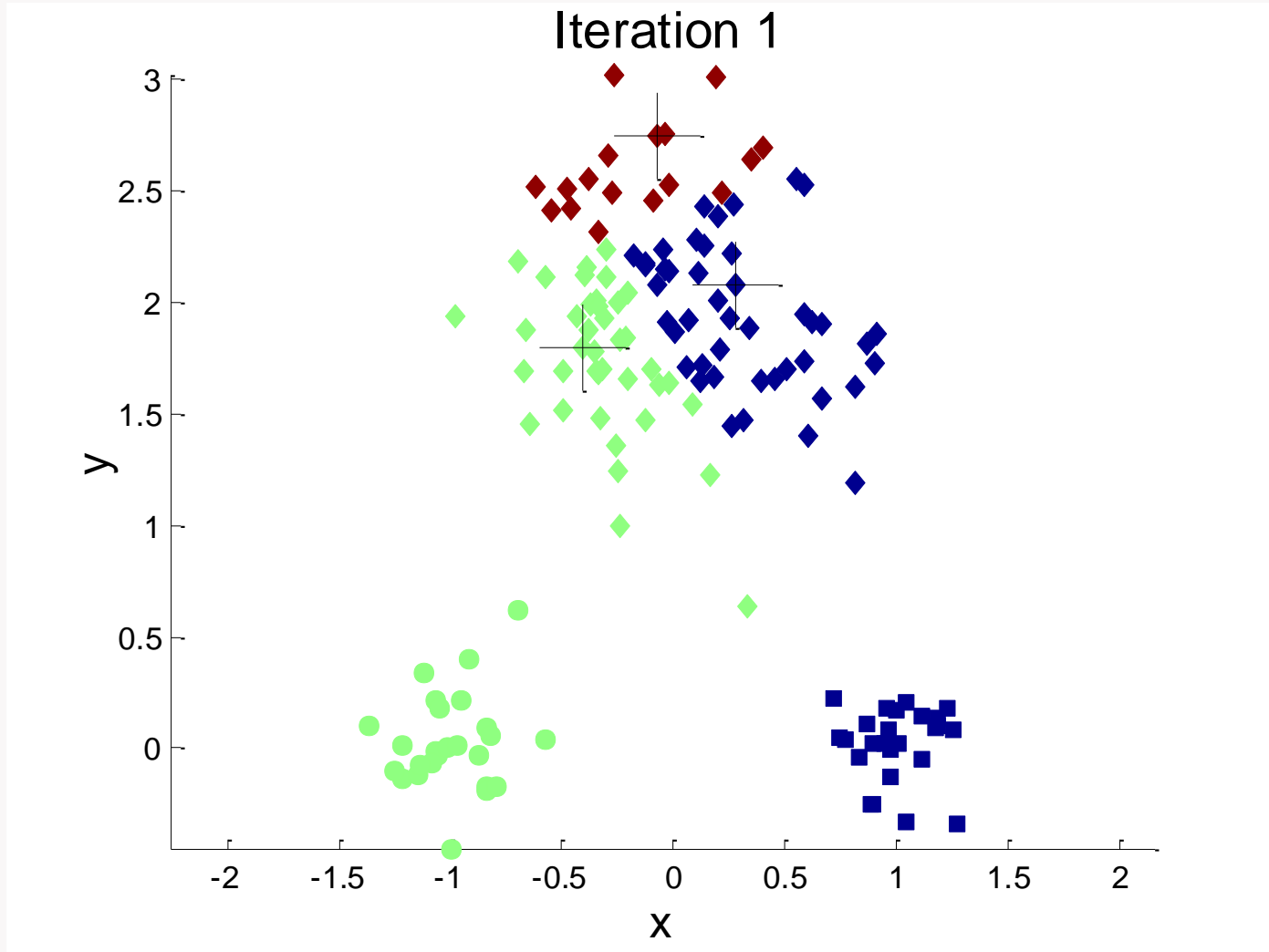
□ K-means算法

- 使用**误差平方和** (Sum of the Squared Error, SSE) 度量迭代中聚类的质量。使用欧式距离作为距离度量时, 目标函数表示为

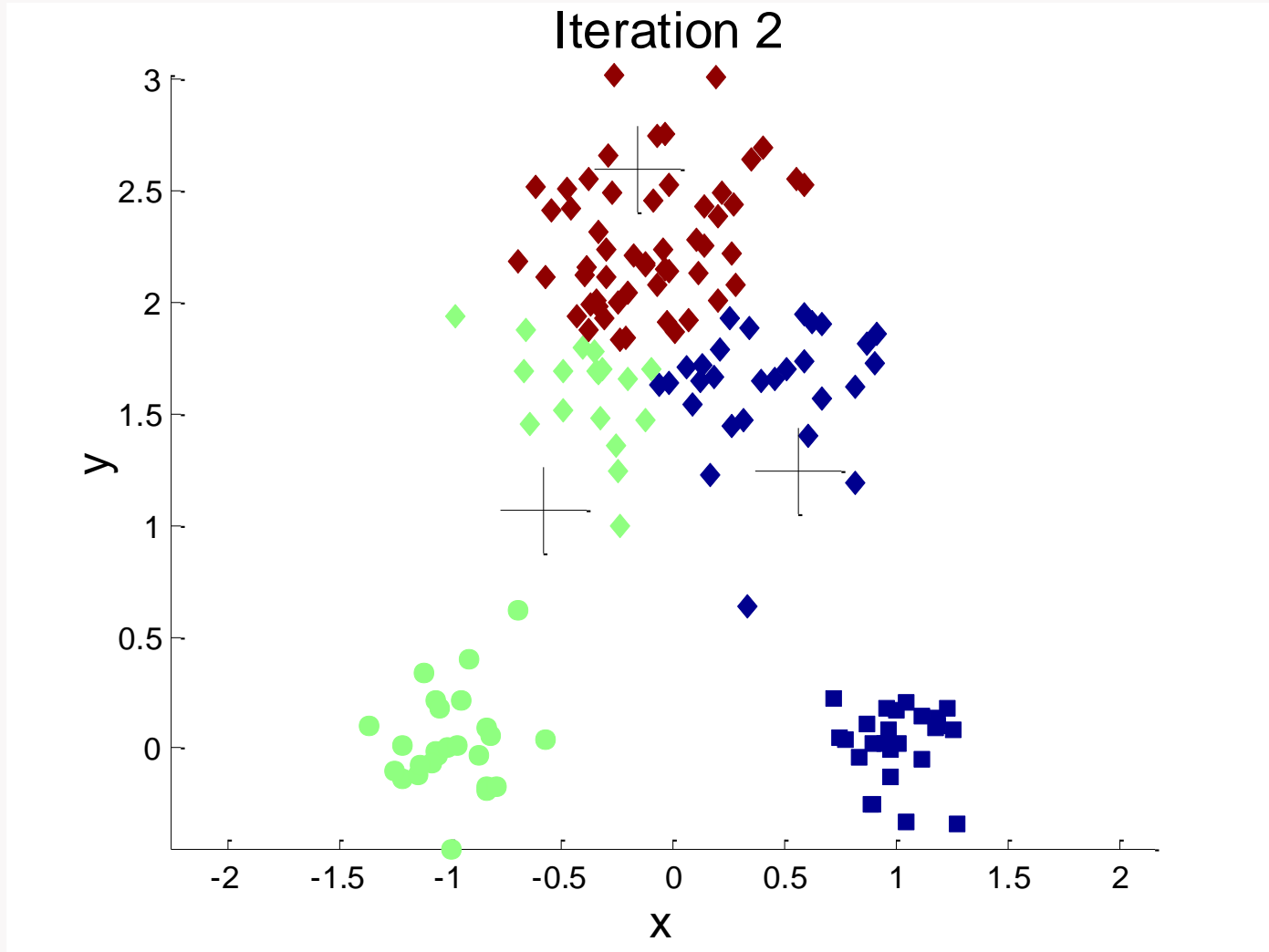
$$loss = \sum_i^K \sum_{x \in c_i} \|x - c_i\|_2^2$$

此时每个数据对象的损失表示为, 其与所属聚类中心的欧式距离。可以证明, 这种情况下使损失函数最小的质心是均值。

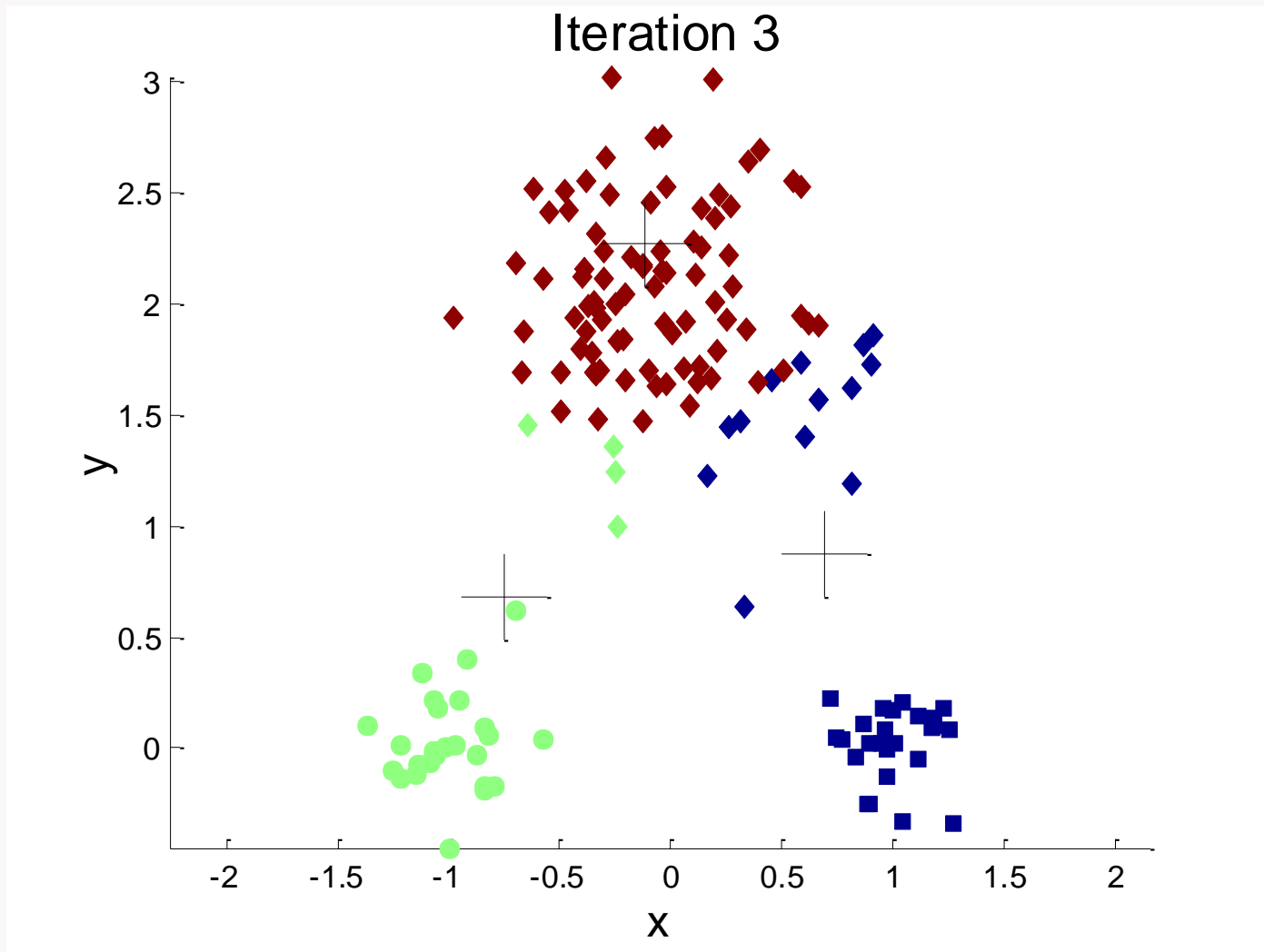
□ K-means算法



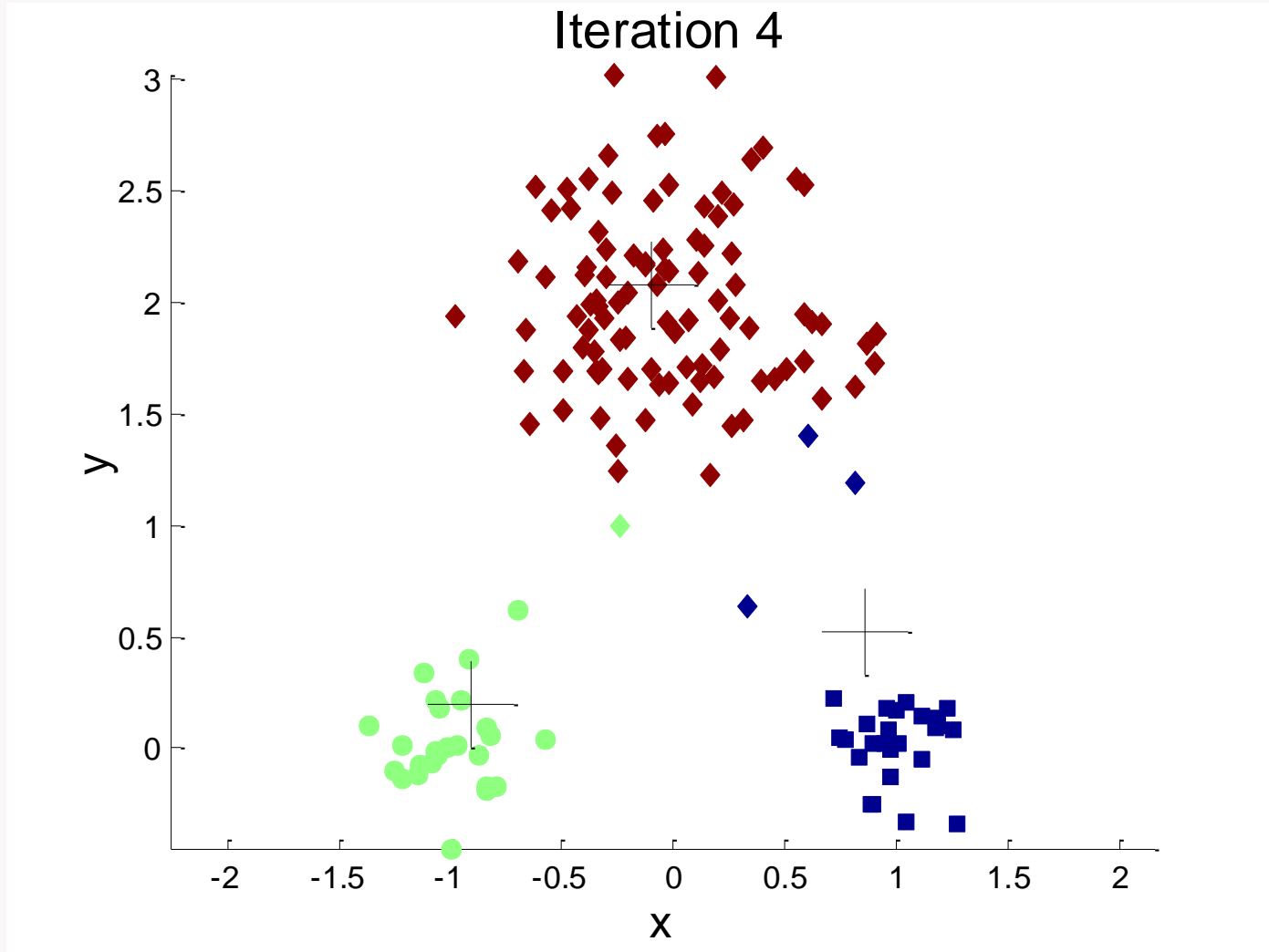
□ K-means算法



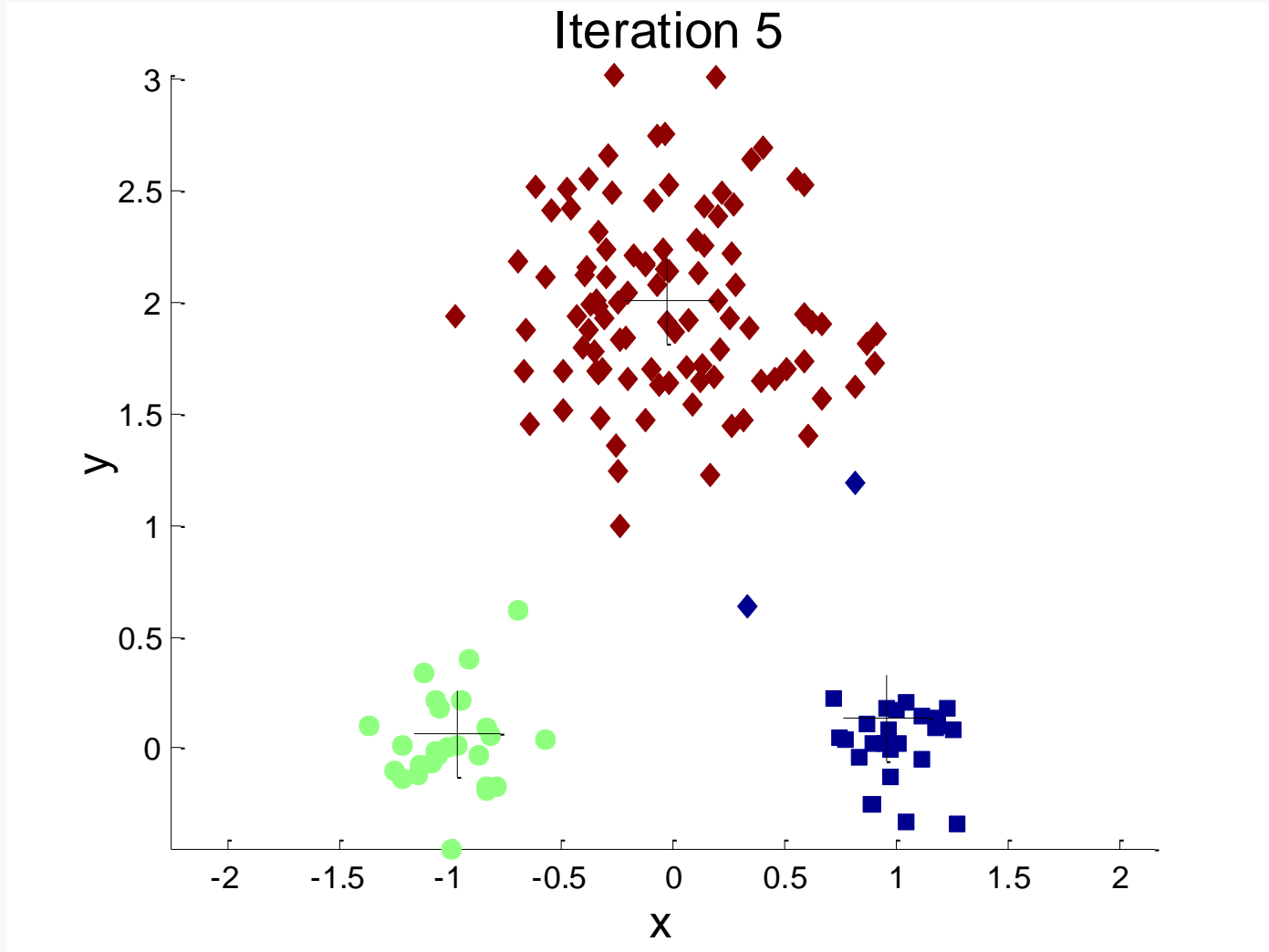
□ K-means算法



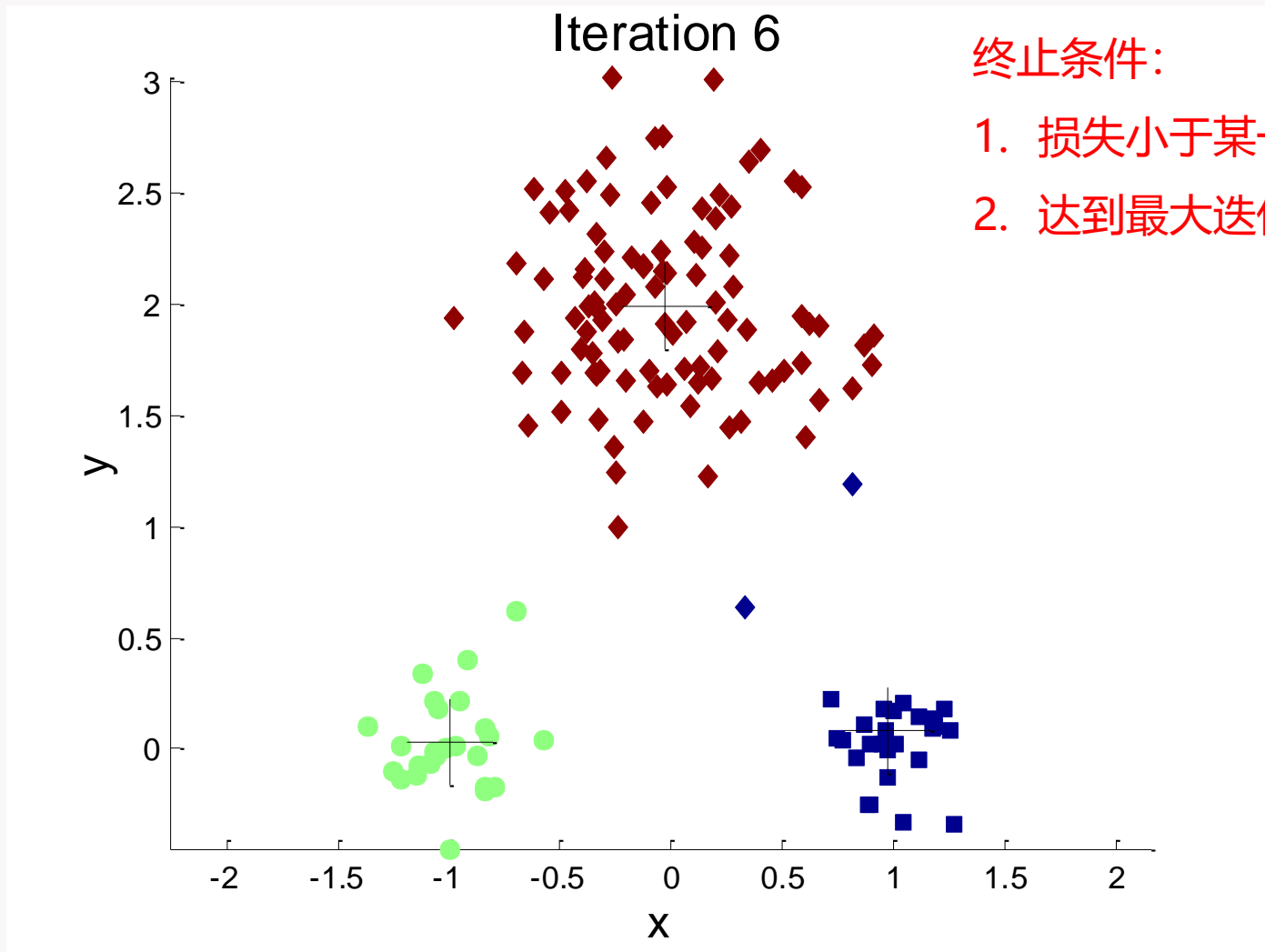
□ K-means算法



□ K-means算法



□ K-means算法



□ 余弦相似性

- 在高维空间中可能需要使用余弦相似性进行相似性度量。使用余弦相似性作为距离度量时，目标函数表示为

$$loss = \sum_i^K \sum_{x \in c_i} (1 - \text{Cos}(c_i, x))$$

可以证明，当每个数据对象单位化后 ($\|x\| = 1$)，使用余弦相似性等价于使用欧式距离。

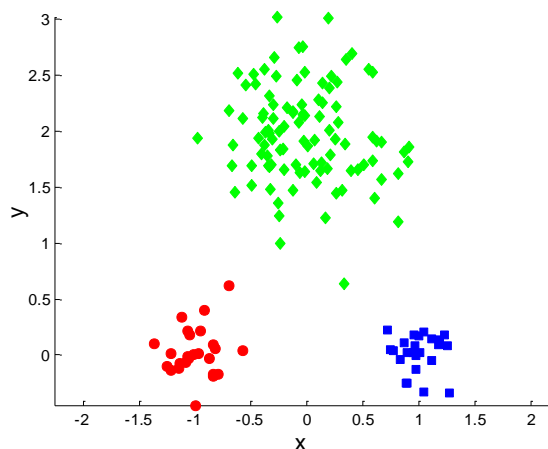
▣ 距离度量、聚类中心和目标函数的关系

■ 一些距离度量、目标函数的组合都可以用于K-means算法，并能确保收敛。

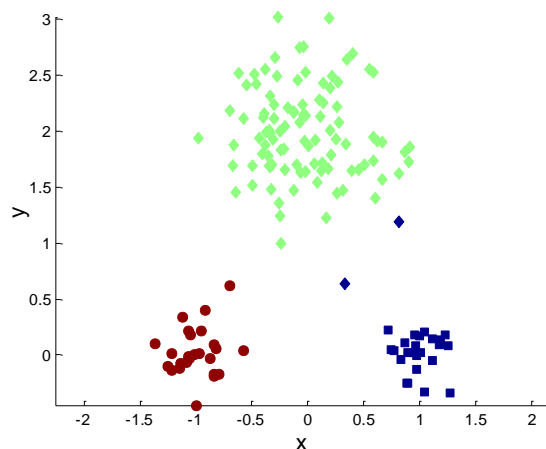
| 距离度量 | 聚类中心 | 目标函数 |
|--------|------|---------------------------|
| 平方欧氏距离 | 均值 | 最小化数据对象到簇质心的 L_2 距离的平方和 |
| 余弦距离 | 均值 | 最小化数据对象到簇质心的余弦距离和 |
| 曼哈顿距离 | 中位数 | 最小化数据对象到簇质心的 L_1 距离和 |

□ 初始质心的选择

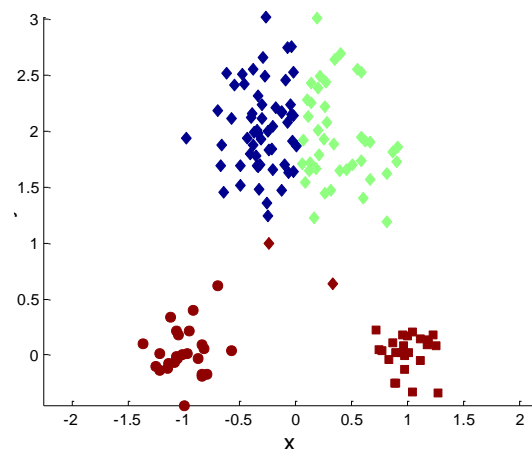
- 当质心初始化时，不同的初始质心导致不同的聚类结果，所以选择合适的初始质心是K-means算法的关键。常见的方法是随机选取质心，但是可能导致聚类质量很差。



数据分布



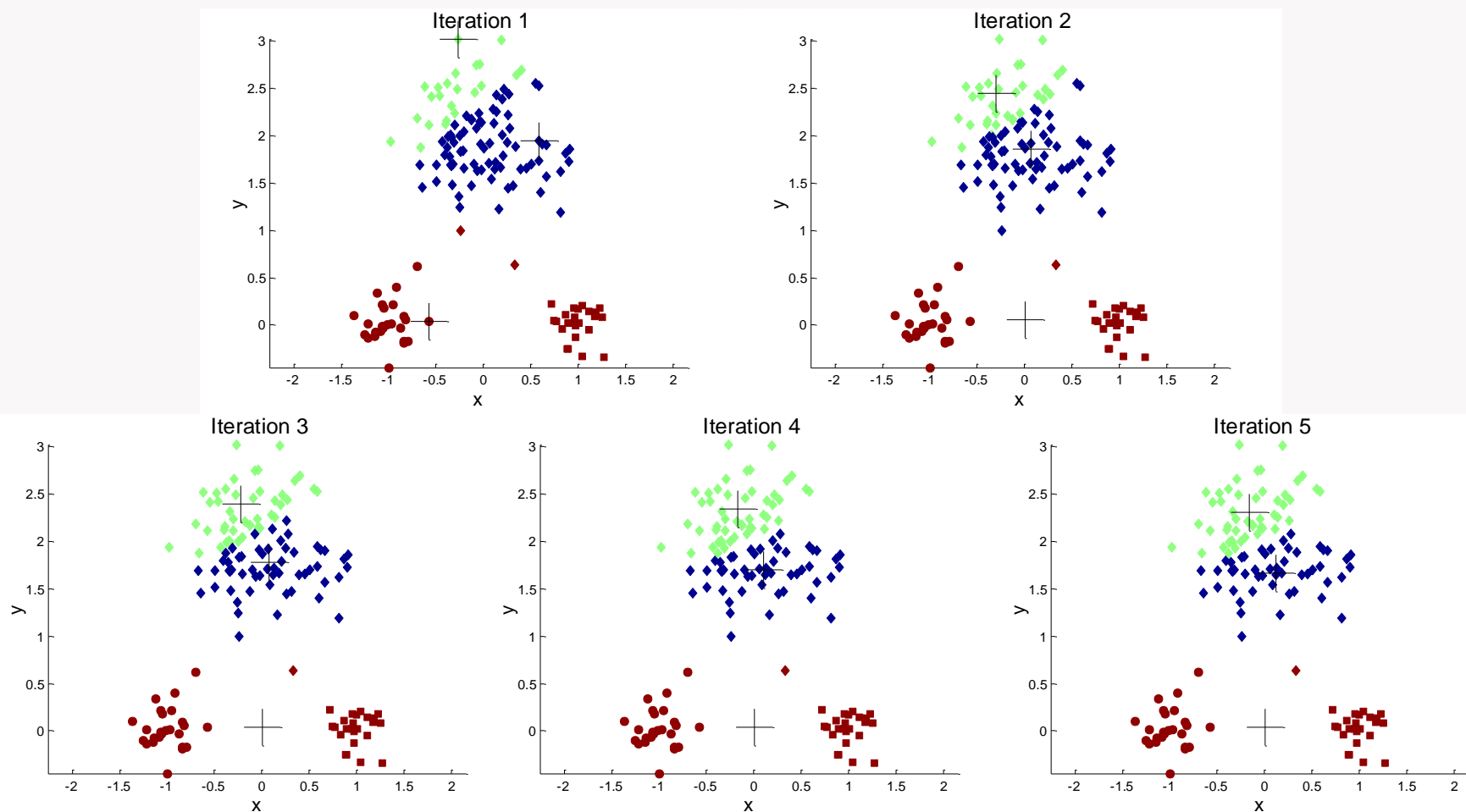
最优聚类



次优聚类

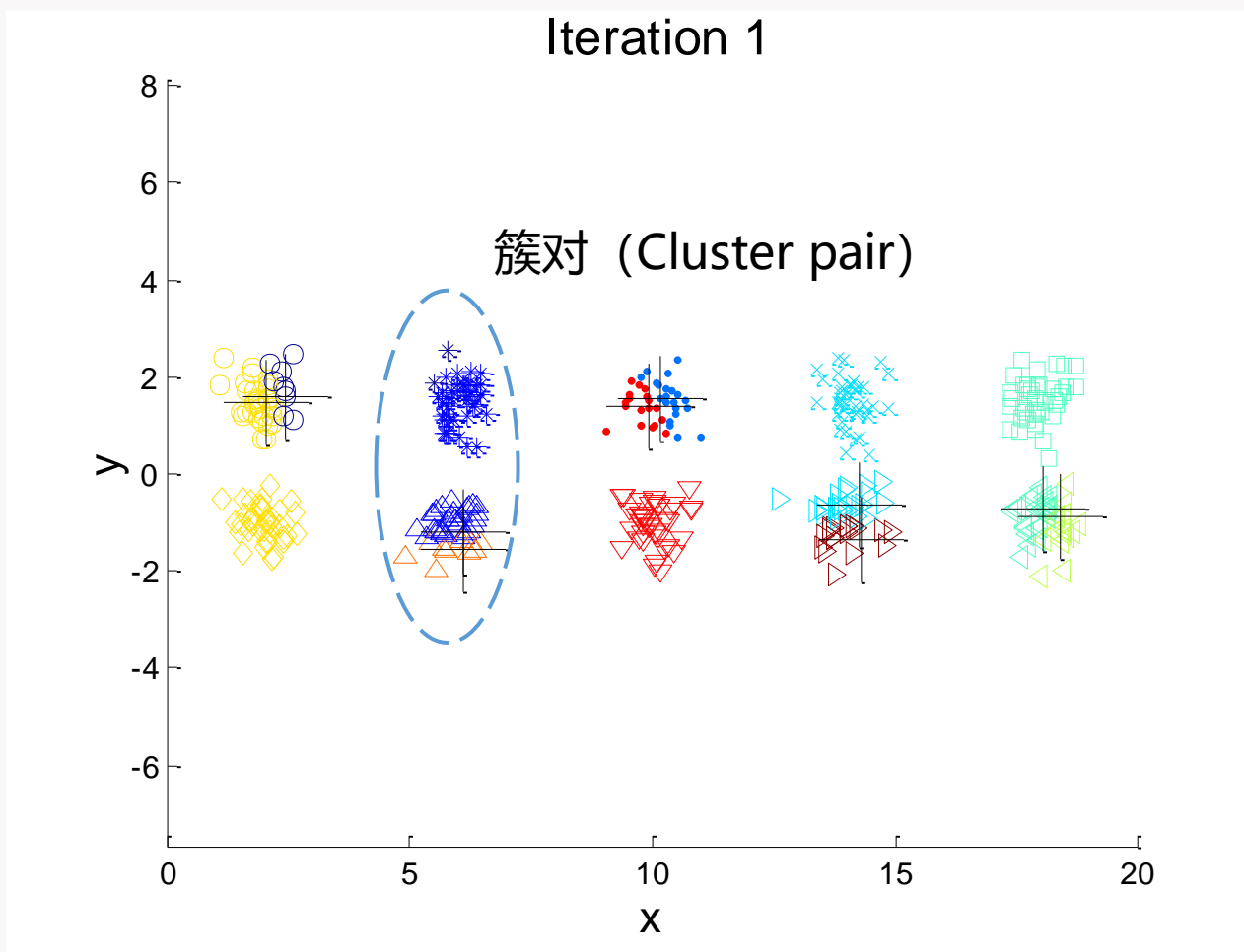
□ 初始质心的选择

■ 质心随机初始化的示例



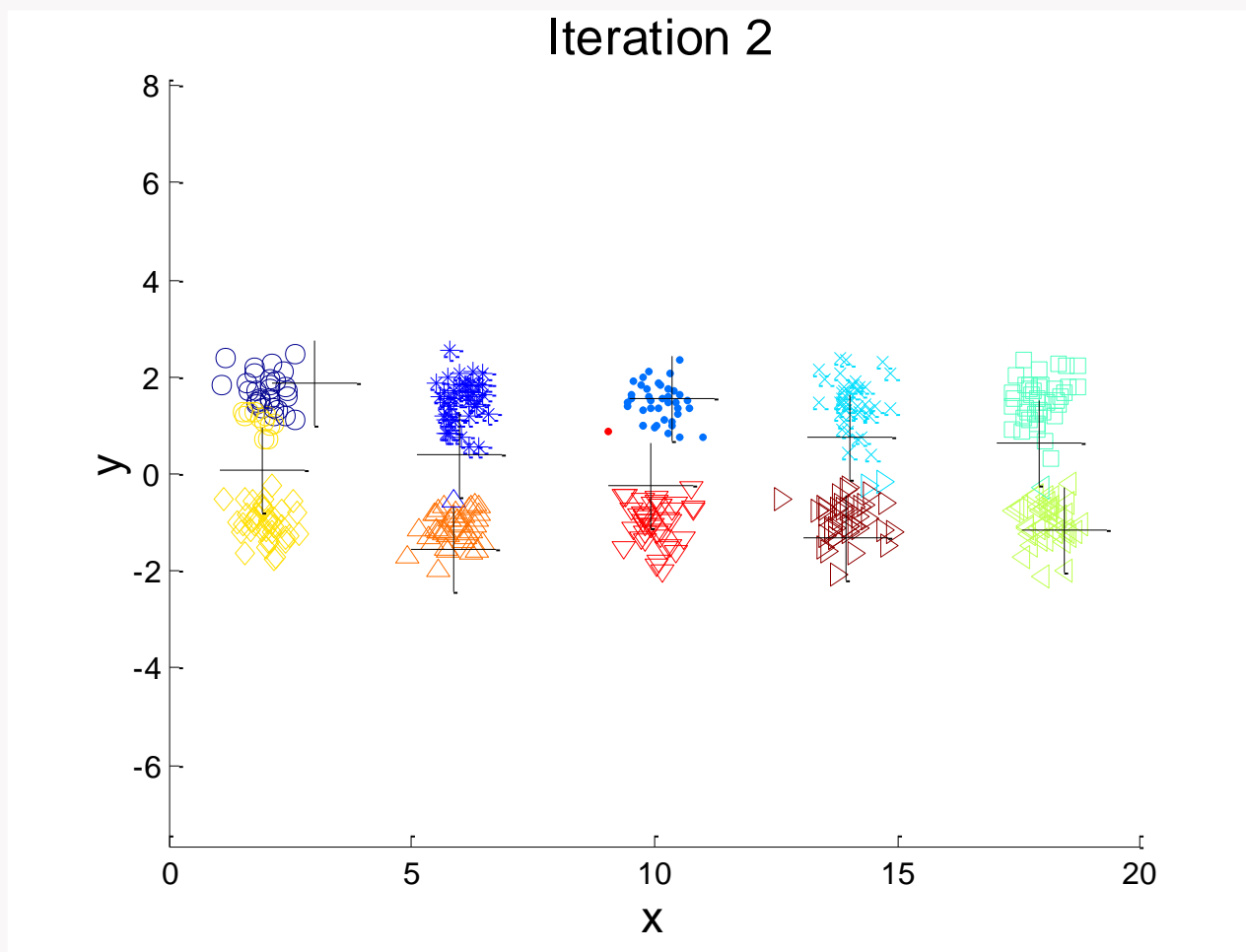
□ 初始质心的选择

■ 质心随机初始化的局限性



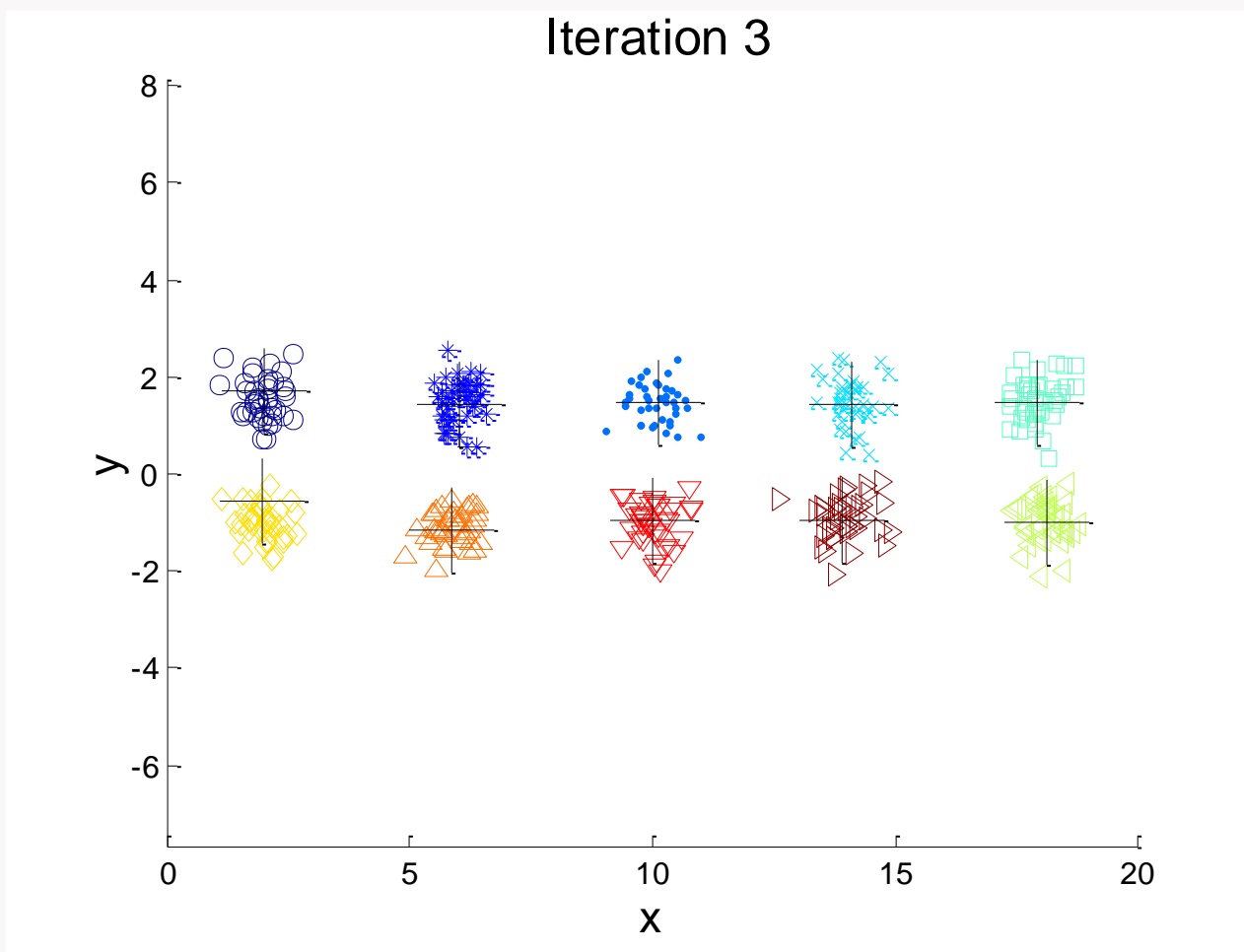
□ 初始质心的选择

■ 质心随机初始化的局限性



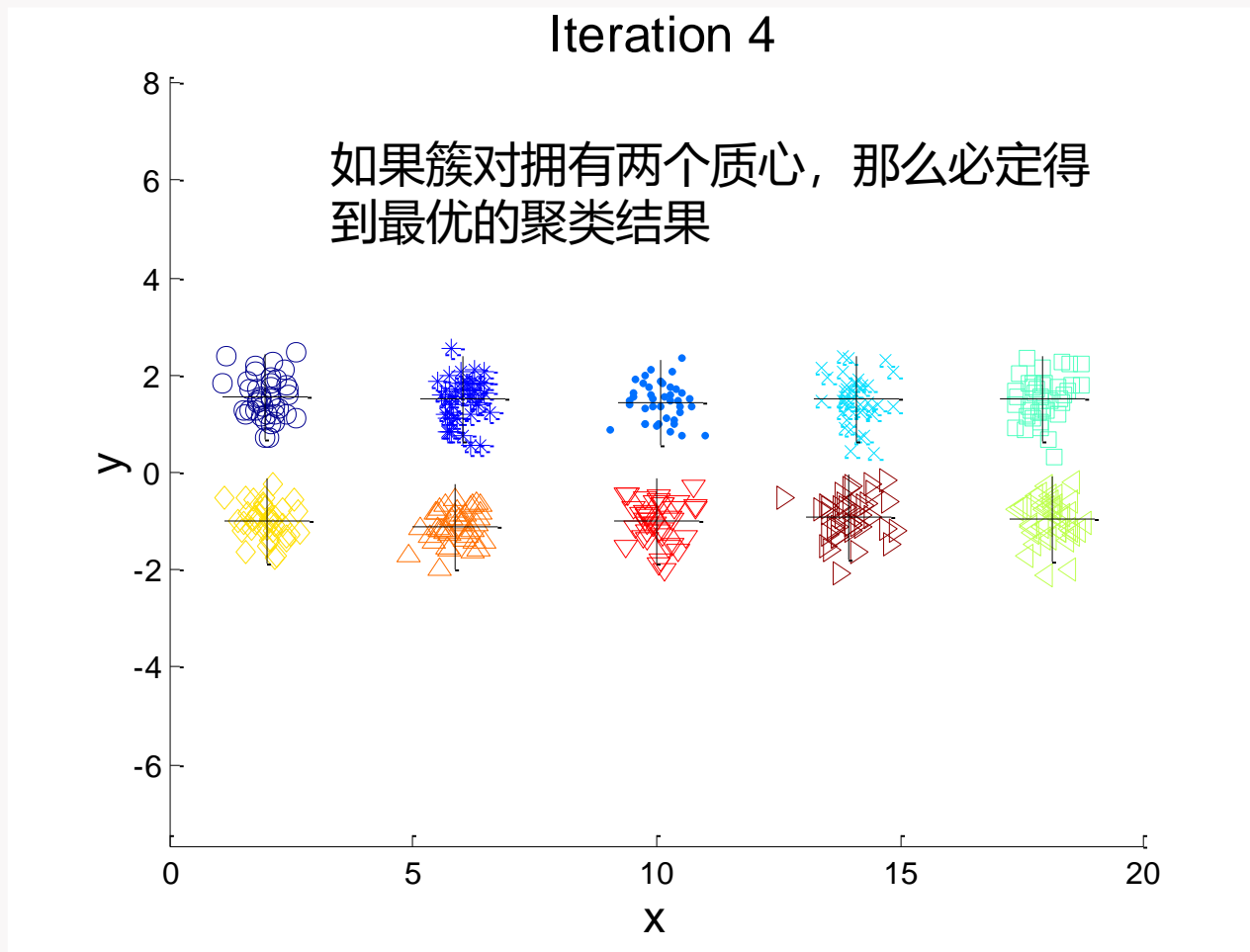
□ 初始质心的选择

■ 质心随机初始化的局限性



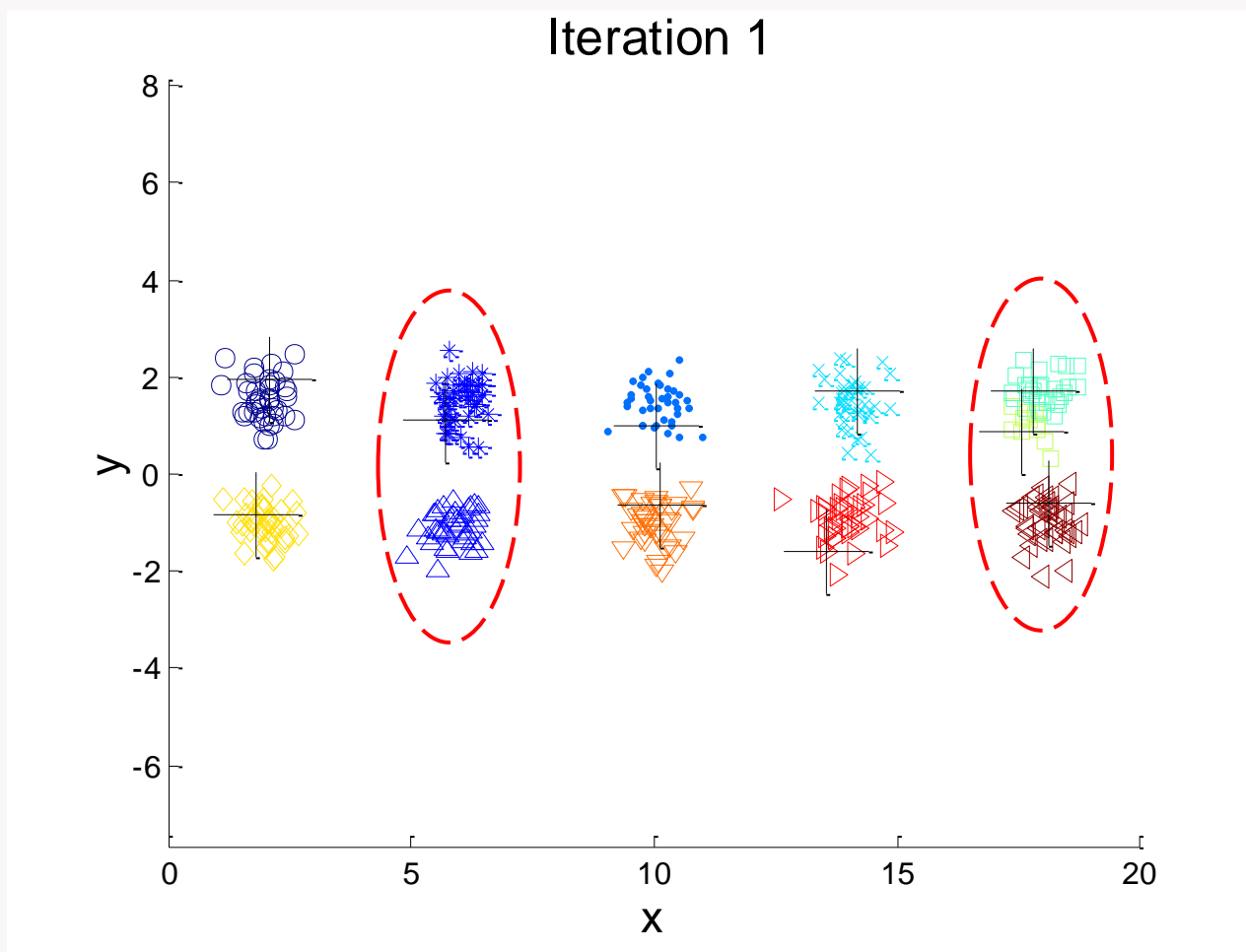
□ 初始质心的选择

■ 质心随机初始化的局限性



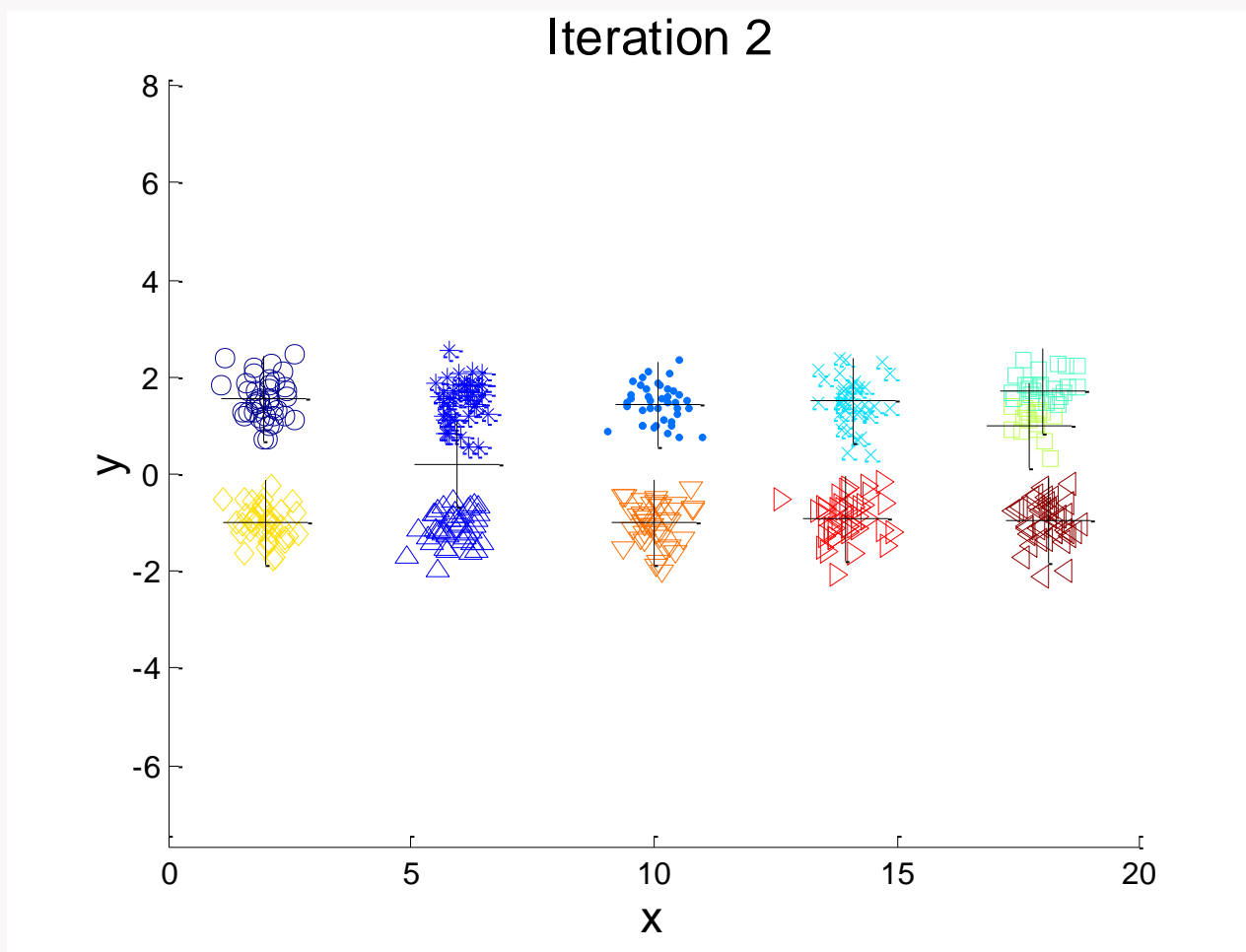
□ 初始质心的选择

■ 质心随机初始化的局限性



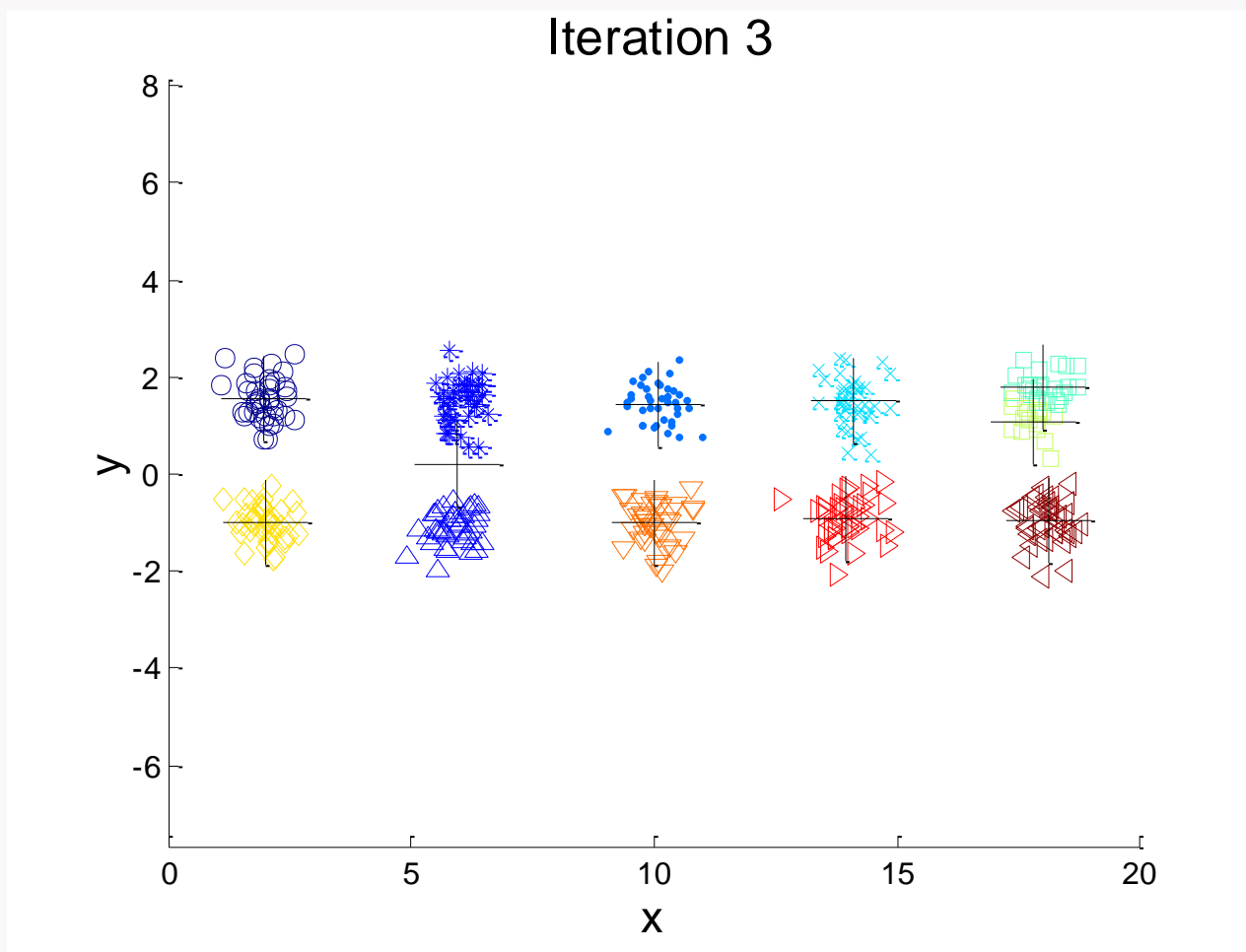
□ 初始质心的选择

■ 质心随机初始化的局限性



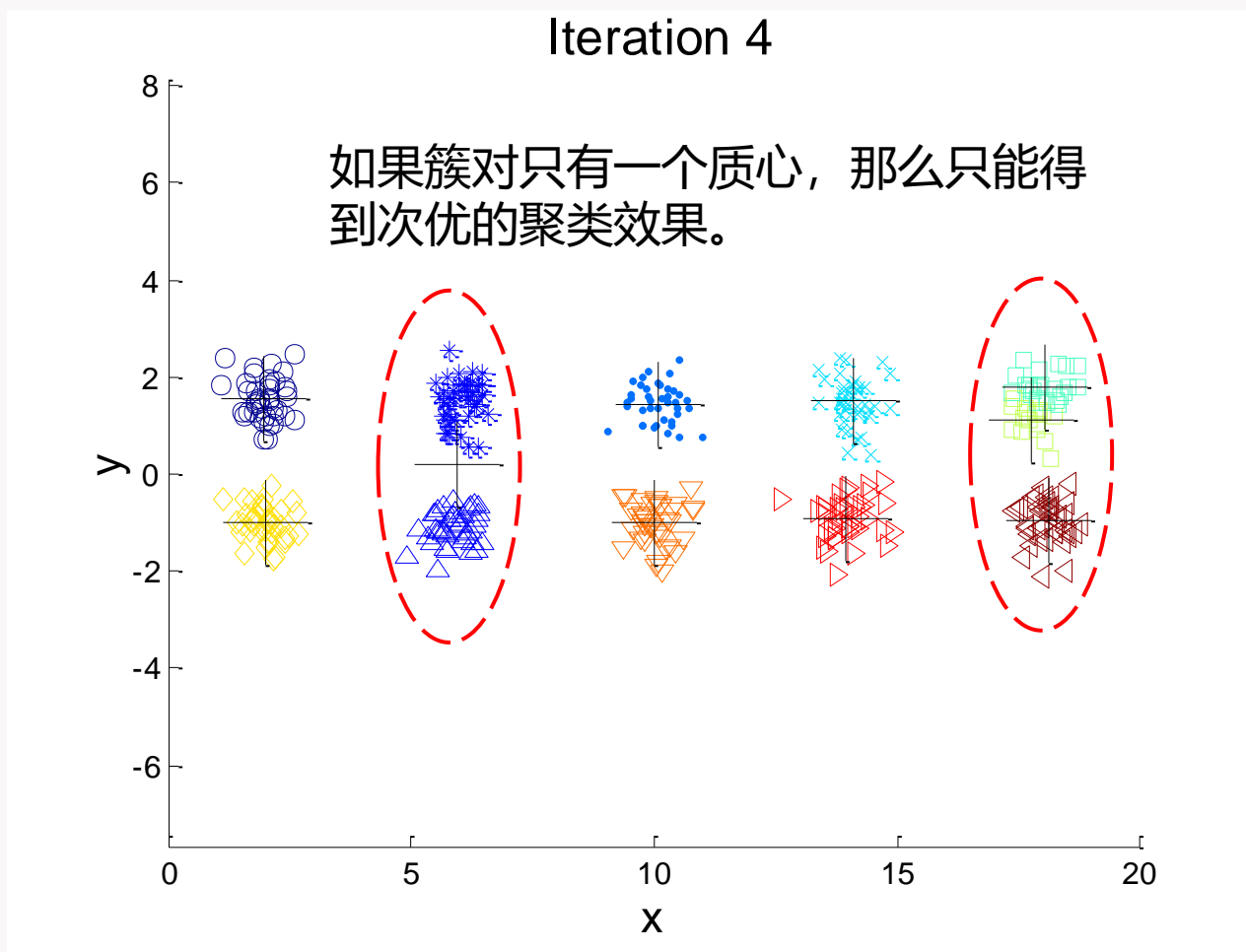
□ 初始质心的选择

■ 质心随机初始化的局限性



□ 初始质心的选择

■ 质心随机初始化的局限性



□ 初始质心的选择

■ 质心随机初始化的局限性

- 假设有K个簇，采用随机抽样的方式选择初始质心，则每个簇中拥有一个初始质心（得到最优的聚类结果）的概率是

$$p = \frac{K!}{K^K}$$

- 所以随着簇的增多，至少一个簇对拥有一个初始质心（得到次优的聚类结果）的概率逐渐增加，而且这个问题即使多次重复的无法解决。

| K | 2 | 4 | 6 | 8 | 10 | 20 | 40 |
|---|--------|-------|-------|-------|-------|-------------|--------------|
| p | 50.00% | 9.38% | 1.54% | 0.24% | 0.04% | $10^{-6}\%$ | $10^{-15}\%$ |

□ K-means++ 算法

■ K-means++ 算法在选择初始质心时采用如下策略：

- 第一个初始质心采用随机选择的方法；
- 假设已经选取了 n ($0 < n < K$) 个初始质心，那么在选择第 $n+1$ 个初始质心时，应该距离当前 n 个质心越远的数据对象有更高的概率被选为新的质心。

□ K-means++ 算法

Kmeans++(D, K)

D: 数据集, K: 簇的数量

1 c={}

2 c[0] = randomSelect(D)

随机选择第一个质心

3 **For** i=1,2,...,K-1 **do**

4 Dist = calDistance(D, c)

计算每个数据对象到每个质心的距离

5 P = calProbability(Dist)

根据距离计算每个数据对象被选中的概率

6 c[i] = selectSample(D, P)

根据概率选择数据对象作为新的质心，也可选择最远的数据对象作为质心。

7 **End For**

8 C = Kmeans(D, c)

9 **Return** C

▣ Bisecting K-means算法

- 二分K-means算法是K-means算法的简单拓展，采用二分法的方式分裂簇，从而降低了对初始质心的敏感程度。
- 它基于一种简单的思想：首先将所有数据对象看作一个簇，然后将它分裂成两个簇，再选择一个簇继续分裂成两个簇，直至得到 K 个簇。

□ Bisecting K-means算法

BKmeans(D, K, T)

D: 数据集, K: 簇的数量, T: 重复次数

1 cluster_index = init(D)

初始化数据对象的簇编号, 形成一个簇

2 count = 1

3 **Repeat**

4 $D_c = \text{selectCluster}(\text{cluster_index}, D)$

选择待分裂簇对应的数据对象, 选择方法有多种, 比如选择最大的簇或最大误差的簇

5 $C = \{\}$

6 **For** i in T **do**

7 $C[i] = \text{Kmeans}(D_c, 2)$

使用Kmeans算法将选定的数据对象分为2个簇, 重复此过程T次

8 **End For**

9 cluster_index = update(cluster_index, C)

选择最小误差的划分, 更新数据对象的簇编号

10 count += 1

11 **Until** count==K

12 **Return** cluster_index

局部二分并不能使得loss最小, 因此使用Bkmeans的结果作为Kmeans算法的初始质心, 并再次聚类是有必要的!

□ K-means算法对噪声更敏感

- 假设数据集中包含了7个标量数据对象： $\{1, 2, 3, 8, 9, 10, 25\}$ ，簇的数量设置为2，则
 - 对于聚类结果 $\{1, 2, 3\}$ 和 $\{8, 9, 10, 25\}$ ，第一个簇的聚类中心为2，第二个簇的聚类中心为13，目标函数值为196；
 - 对于聚类结果 $\{1, 2, 3, 8\}$ 和 $\{9, 10, 25\}$ ，第一个簇的聚类中心为3.5，第二个簇的聚类中心为14.67，目标函数值为189.67。
- 离群点 25 使得K-means将更为相似的 8 和 9 划分到不同的簇。此外，第二个簇的聚类中心为14.67，明显偏离簇中的数据对象！

□ K-中心点

- 采用实际的数据对象而非均值代表簇，其余的数据对象被分配到与其最为相似的代表对象所在的簇中。使用**绝对误差标准** (Absolute-Error Criterion) 定义目标函数：

$$loss = \sum_i^K \sum_{x \in o_i} |x - o_i|$$

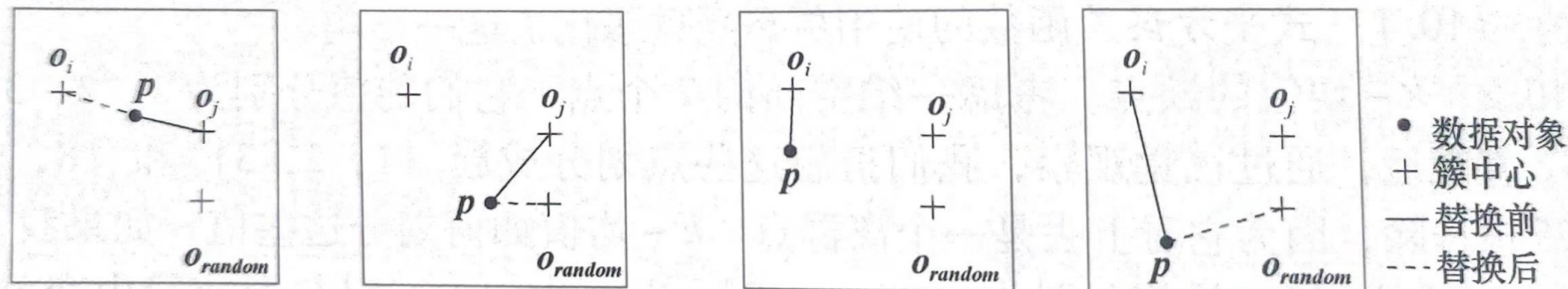
- 通过最小化该目标函数，把 n 个数据对象划分到 k 个簇中，这就是 k-中心点方法的基础。
- 当 k=1 时，可以快速的找到中心点，但当 k 为其他常数时，k-中心点问题是一个 NP-hard 问题。

□ 围绕中心点划分算法

- **围绕中心点划分算法** (Partitioning Around Medoids, 简称PAM) 是K-中心点问题的常见解决方式, 使用迭代和贪心的方法求解该问题。PAM算法的流程如下:
 - 随机选择 K 个数据对象作为初始的代表对象;
 - 将其余的数据对象划分到与其最相似的代表对象所属的簇;
 - 为了判断 o_j 是否是一个较好的代表对象, 随机地选择一个非代表对象 o_{random} 代替 o_j 并更新目标函数值;
 - 根据目标函数的变化决定是否使用 o_{random} 代替 o_j 。

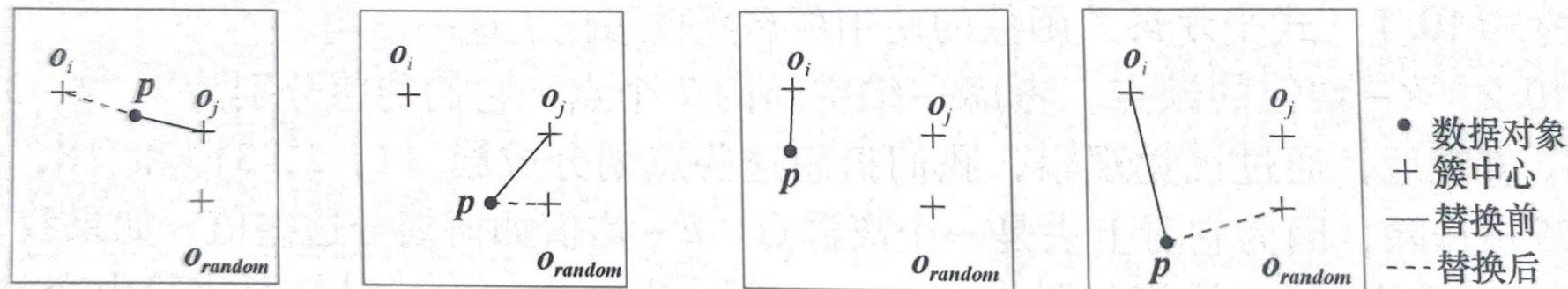
围绕中心点划分算法

- o_i 为一个簇的代表对象, o_j 为待考察的代表对象, o_{random} 为随机选择的非代表对象, 对于数据对象 p , 使用 o_{random} 替换 o_j 的结果有以下四种:
 - 替换前 p 属于 o_j , 替换后 p 属于 o_i , 则目标函数变化为 $d(p, o_i) - d(p, o_j)$;
 - 替换前 p 属于 o_j , 替换后 p 属于 o_j , 则目标函数变化为 $d(p, o_{random}) - d(p, o_j)$;



围绕中心点划分算法

- o_i 为一个簇的代表对象, o_j 为待考察的代表对象, o_{random} 为随机选择的非代表对象, 对于数据对象 p , 使用 o_{random} 替换 o_j 的结果有以下四种:
 - 替换前 p 属于 o_i , 替换后 p 属于 o_i , 则目标函数变化为 0;
 - 替换前 p 属于 o_i , 替换后 p 属于 o_j , 则目标函数变化为 $d(p, o_{random}) - d(p, o_i)$;



□ 围绕中心点划分算法

■ 计算使用 o_{random} 替换 o_j 的总变化值:

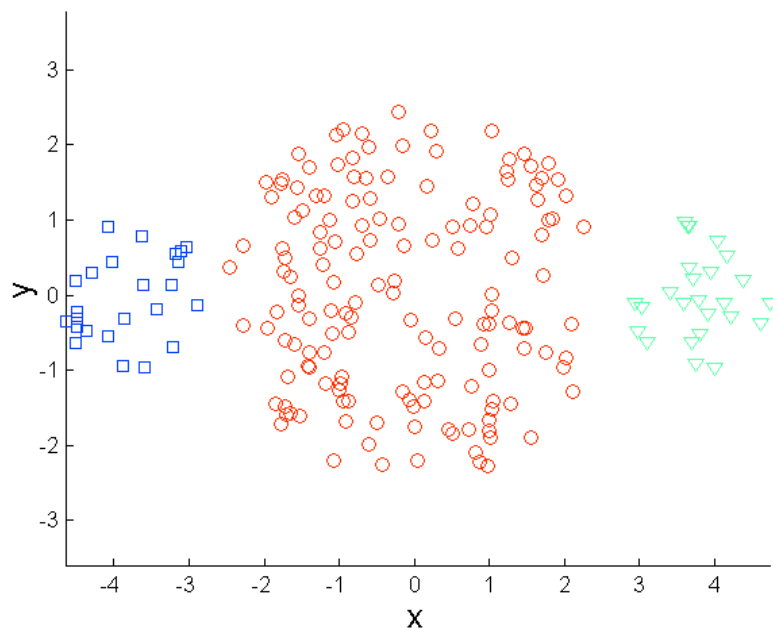
- 如果小于零, 说明使用 o_{random} 替换 o_j 会取得更好的聚类效果;
- 如果大于等于零, 说明当前的 o_j 是可以接受的, 在本次迭代不改变任何代表对象。

□ Kmeans算法和数据的分布

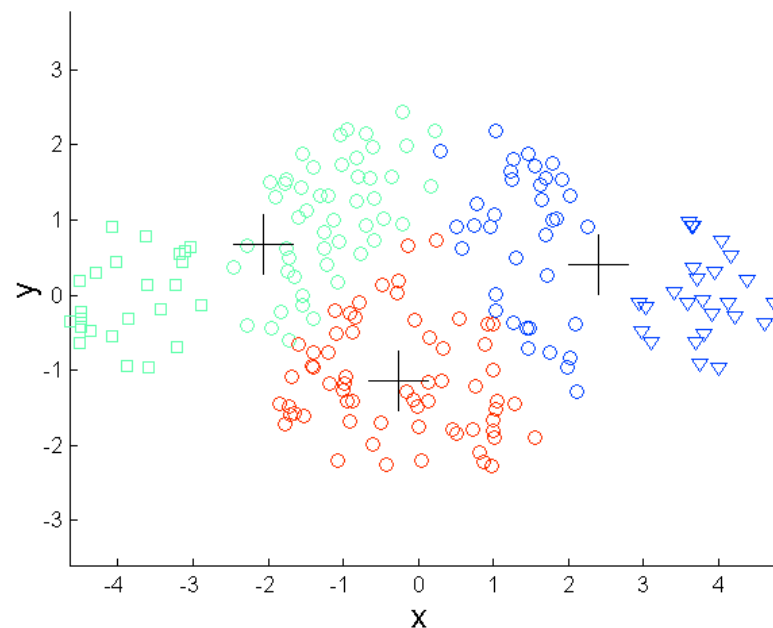
- 对于发现不同类型的簇，Kmeans算法和相关改进算法都具有一定的局限性。更具体地说，当簇具有非球形的形状或具有不同尺寸或密度时，Kmeans算法很难发现“真实簇”的形状。
- 可采用的解决方案是将数据划分成更多的簇，至少保证每个更小的簇是纯的。

□ Kmeans算法和数据的分布

■ 不同尺寸的簇的结果



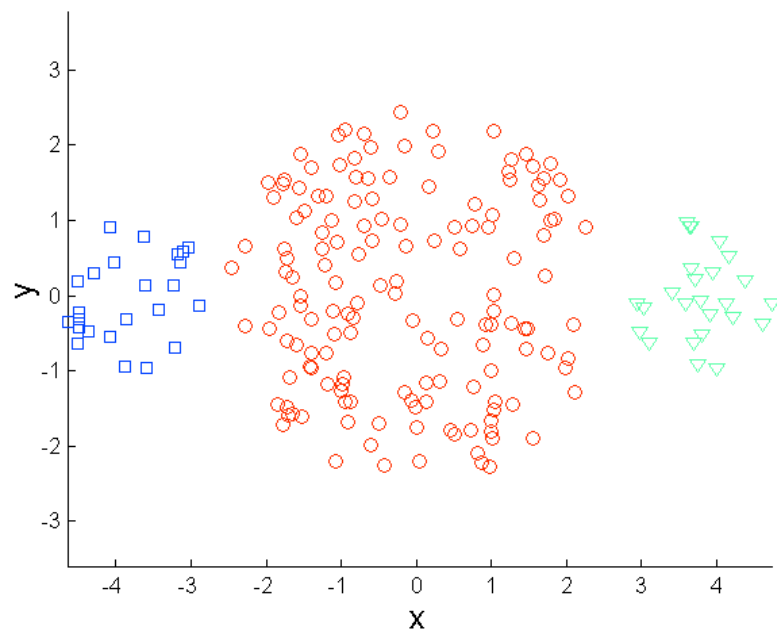
数据分布



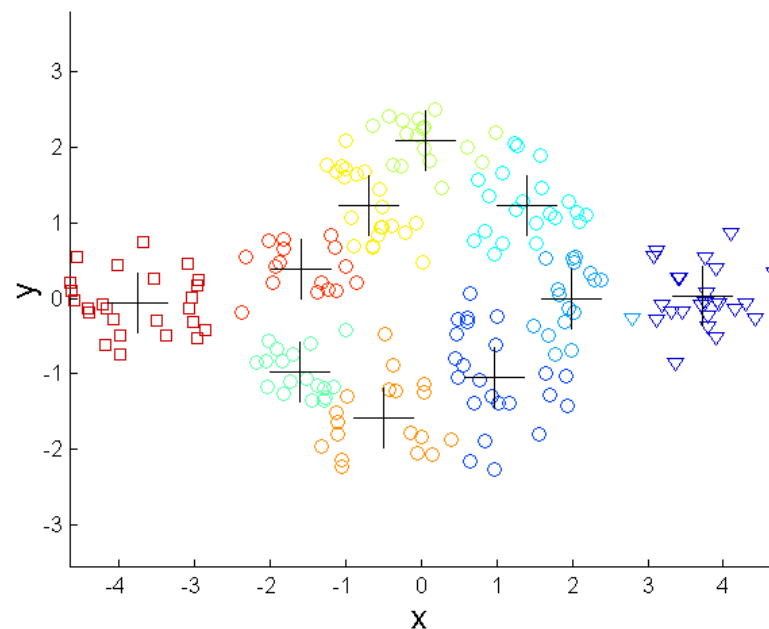
Kmeans聚类结果

□ Kmeans算法和数据的分布

■ 更大的K值



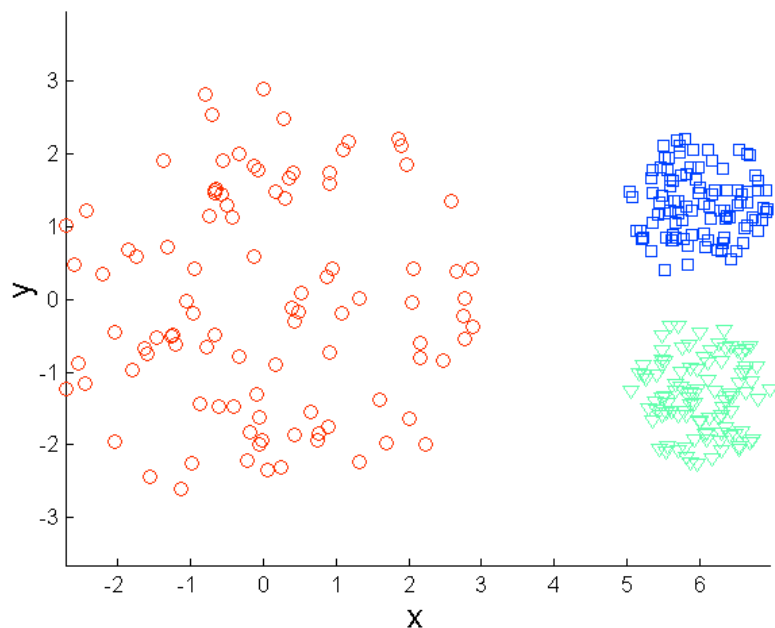
数据分布



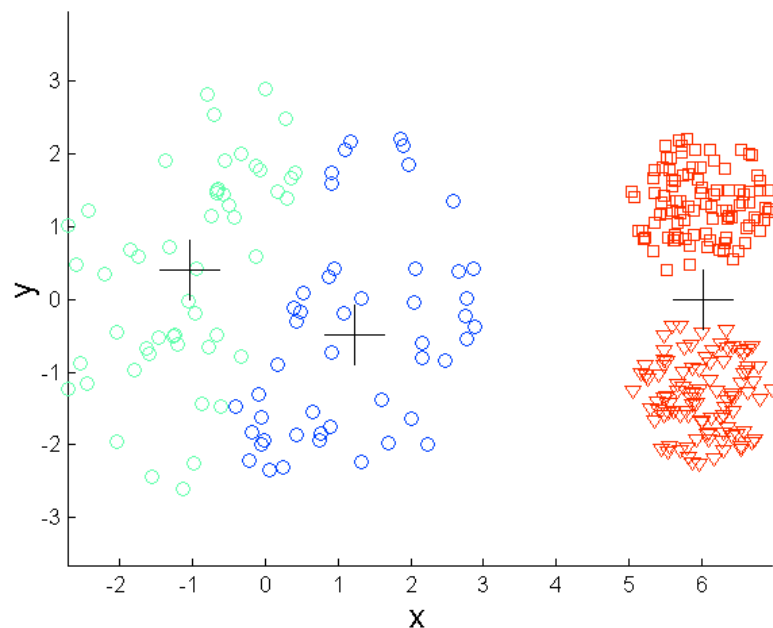
Kmeans聚类结果

□ Kmeans算法和数据的分布

■ 不同密度的簇的结果



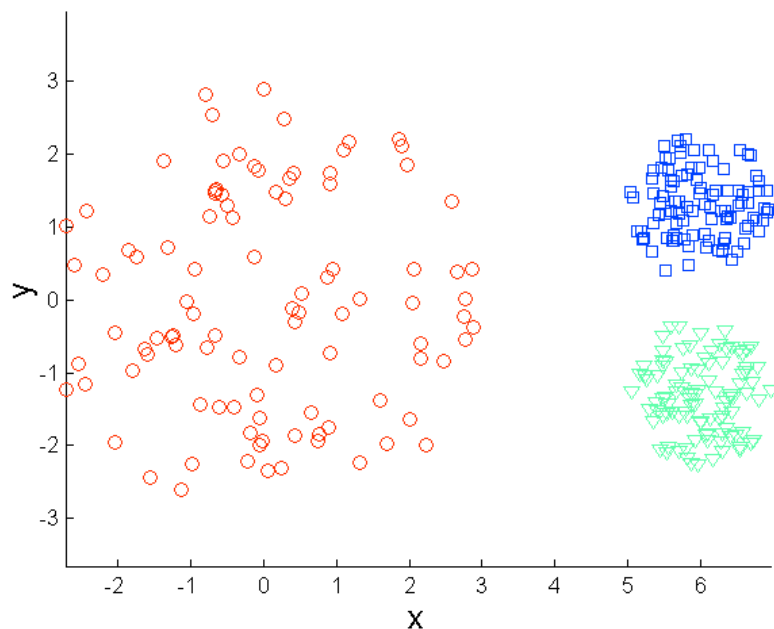
数据分布



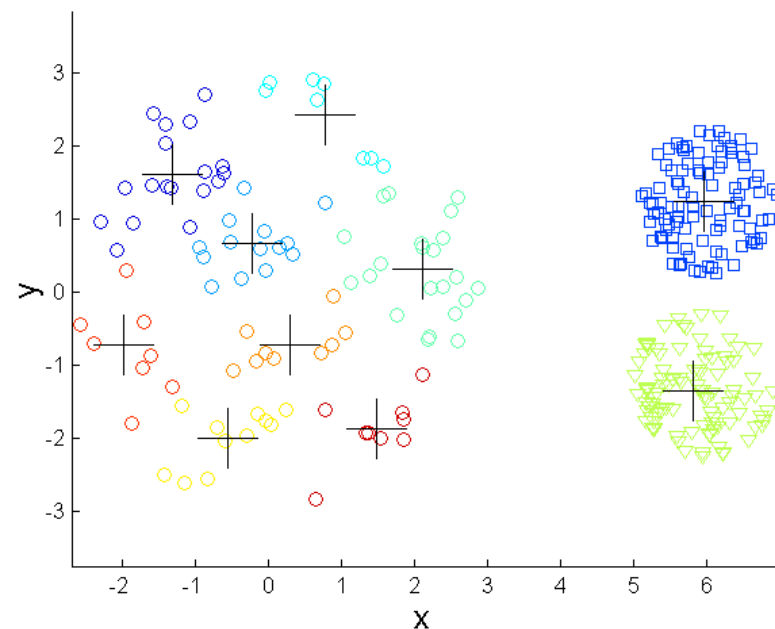
Kmeans聚类结果

□ Kmeans算法和数据的分布

■ 更大的K值



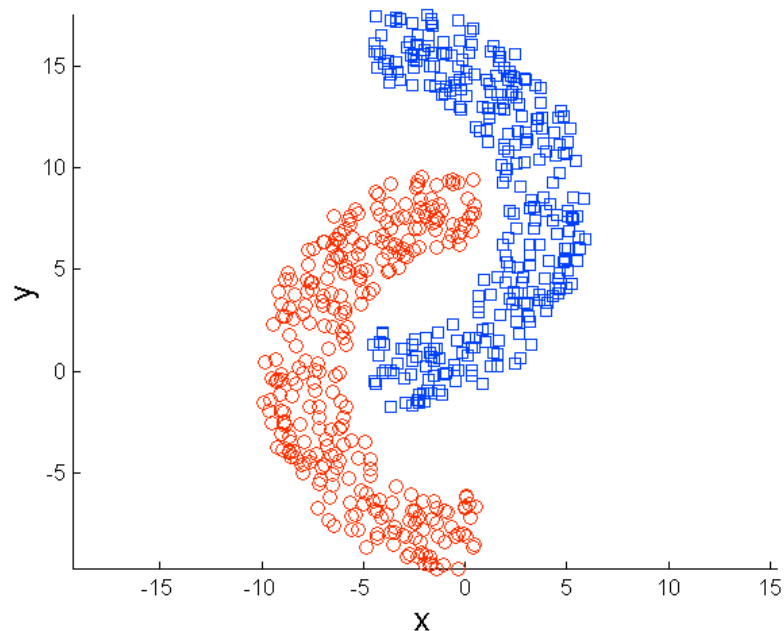
数据分布



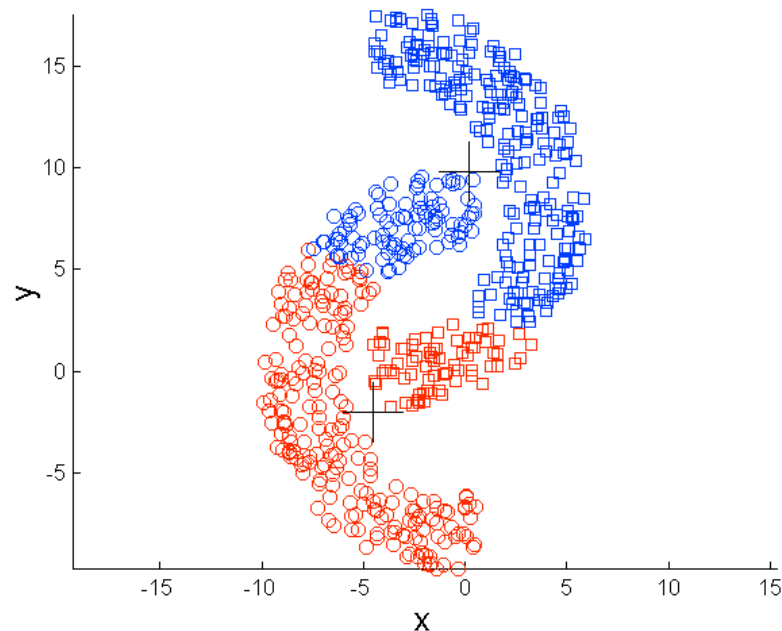
Kmeans聚类结果

□ Kmeans算法和数据的分布

■ 非球形的簇的结果



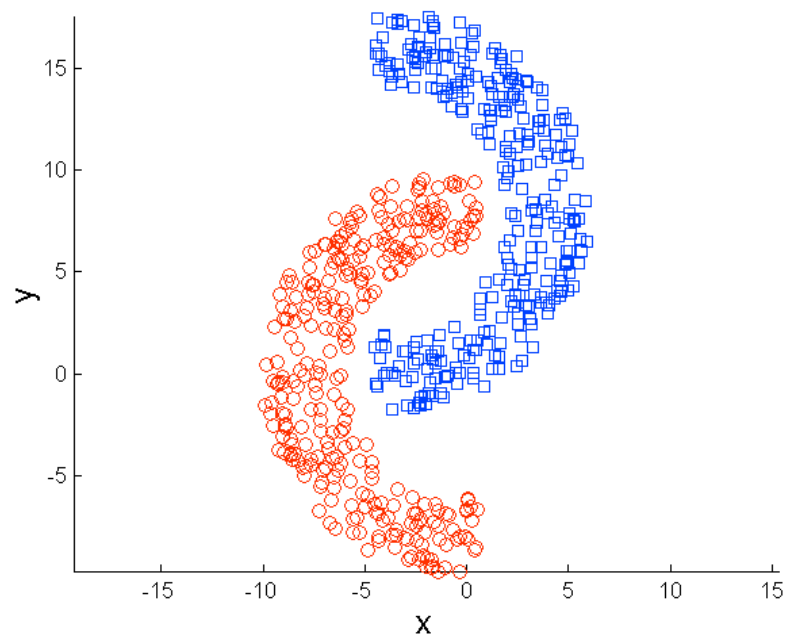
数据分布



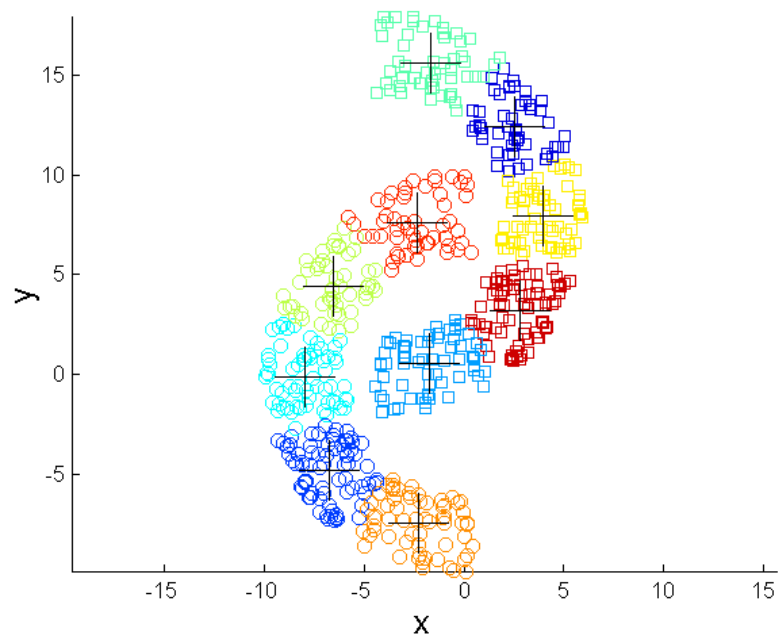
Kmeans聚类结果

□ Kmeans算法和数据的分布

■ 更大的K值



数据分布



Kmeans聚类结果

□ K-means算法的时间复杂度和空间复杂度

- K-means算法在运行时只需要存储数据集和质心，所以它的空间复杂度为 $O((N + K)m)$ ，其中 N 表示数据对象的数量， K 表示聚类中心的数量， m 表示属性的数量。所以K-means算法的空间复杂度是线性的。
- K-means算法运行的时间复杂度是 $O(Iter \times KN)$ ，其中 $Iter$ 表示迭代次数，所以K-means算法的时间复杂度也是线性的。

□ Kmeans算法的特点

■ 优点:

- 简单有效，可伸缩性强；
- 当真实簇是密集的，且簇间差异性较大，则会取得较好的效果。

■ 缺点:

- 需要人为指定 k 值；
- 对初始聚类中心敏感（K-means++和BK-means）；
- 通常取得局部最优解；
- 对噪声和异常值敏感（PAM）；
- 只适用于数值属性；
- 不适合所有类型的数据分布，难以处理不同尺寸的、不同密度的、非球形的簇。

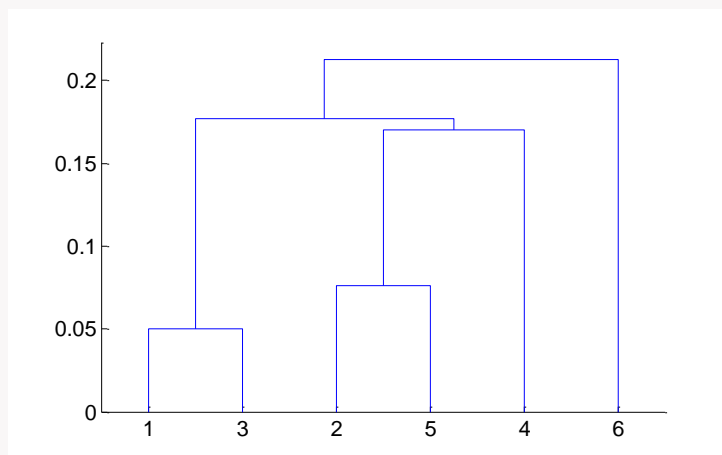


Chapter 5.3

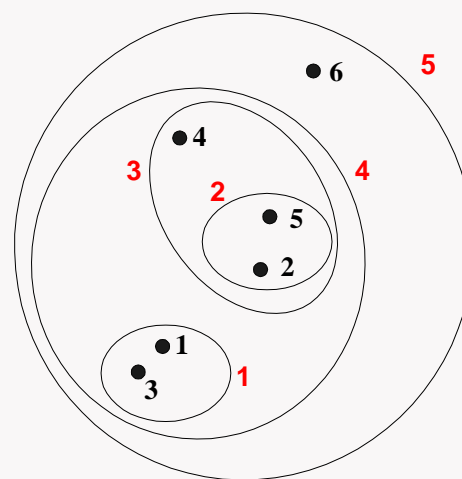
基于层次的聚类方法

□ 基于层次的聚类方法

- 与基于划分的聚类方法相比，层次聚类同样是被广泛使用的、非常重要的聚类技术。层次聚类按照层次是否被细化分为以下两类方法：
 - **凝聚的**：将每个数据对象看作一个簇，每一步合并最近的簇；
 - **分裂的**：将全部数据对象看作一个簇，每一步分裂一个簇。



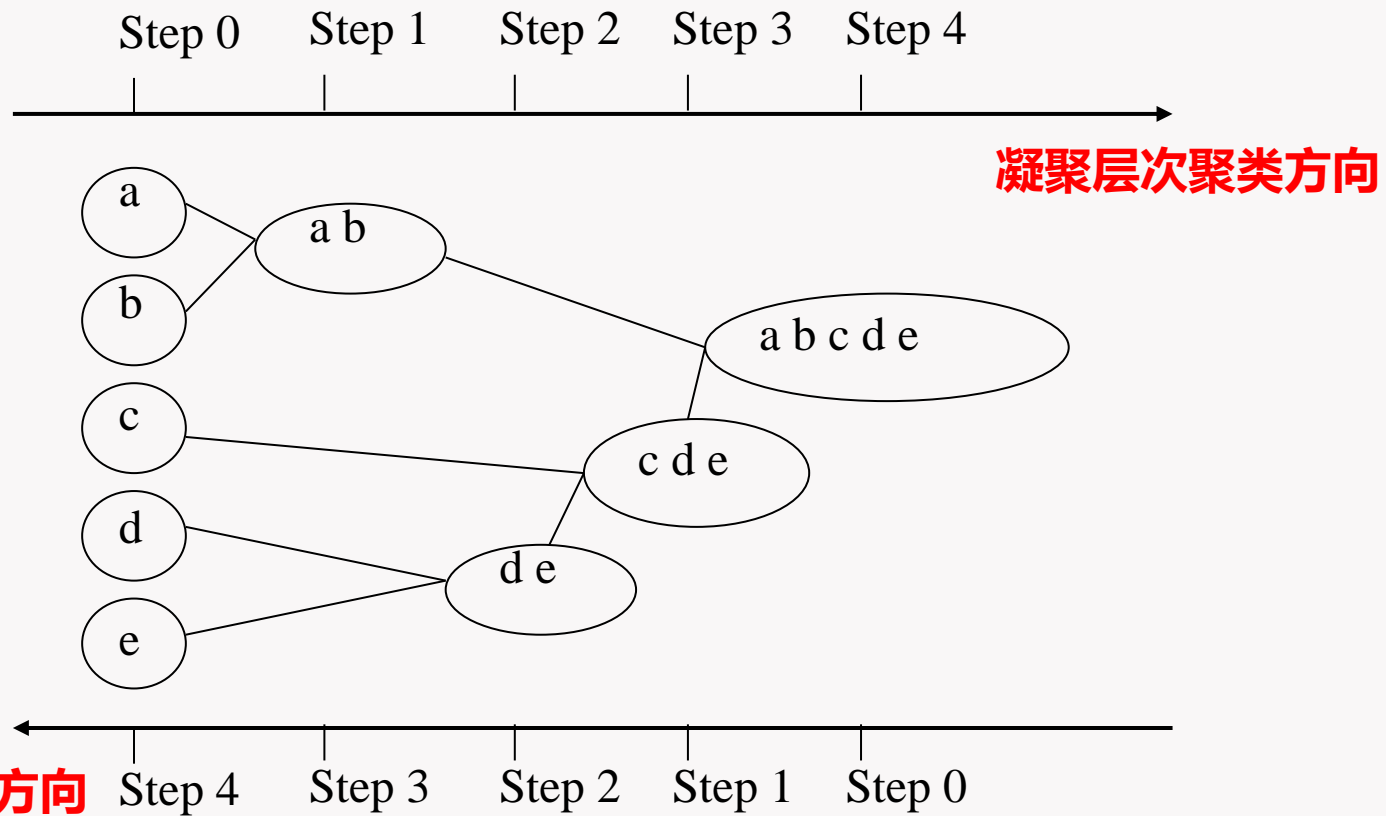
树状图 (Dendrogram)



嵌套簇图 (Nested cluster diagram)

□ 基于层次的聚类方法

■ 凝聚层次聚类和分裂层次聚类的示意图



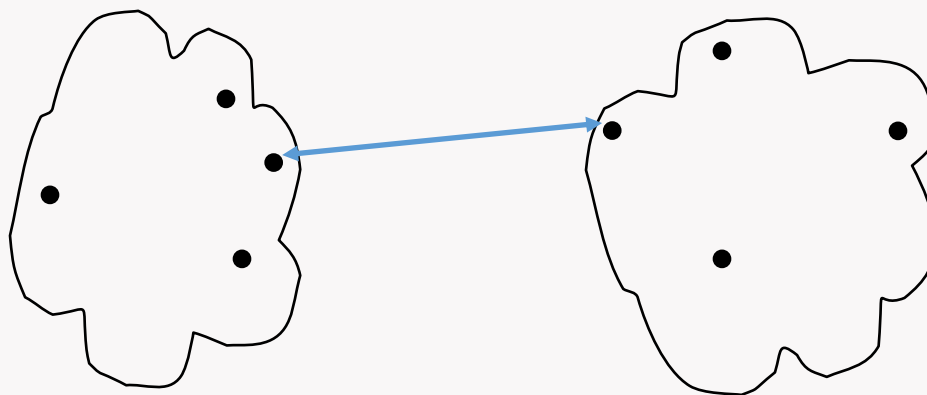
□ 凝聚层次聚类

- 凝聚层次聚类的算法都可以用一个框架描述：从每个数据对象作为簇开始，相继合并两个最接近的簇，直至剩下一个簇或满足用户指定的簇。交替迭代的执行以下两个步骤：
 - 利用簇的邻近度矩阵合并最接近的两个簇；
 - 更新簇的邻近度矩阵，重新计算合并的簇和未改变的簇之间的邻近度信息。

□ 簇之间的邻近度度量

- 凝聚层次聚类计算簇之间的邻近度度量时可以采取不同的计算方式，这些方式常从图的角度进行定义和区分：
- **单链** (Single-linkage)：簇之间的邻近度定义为不同簇中两个**最近**的数据对象的距离。

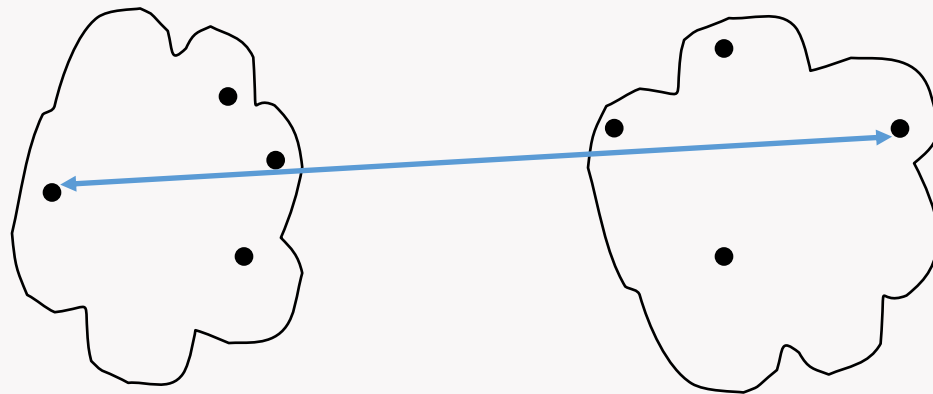
$$d_{sl}(C_i, C_j) = \min_{x \in C_i, y \in C_j} dist(x, y)$$



□ 簇之间的邻近度度量

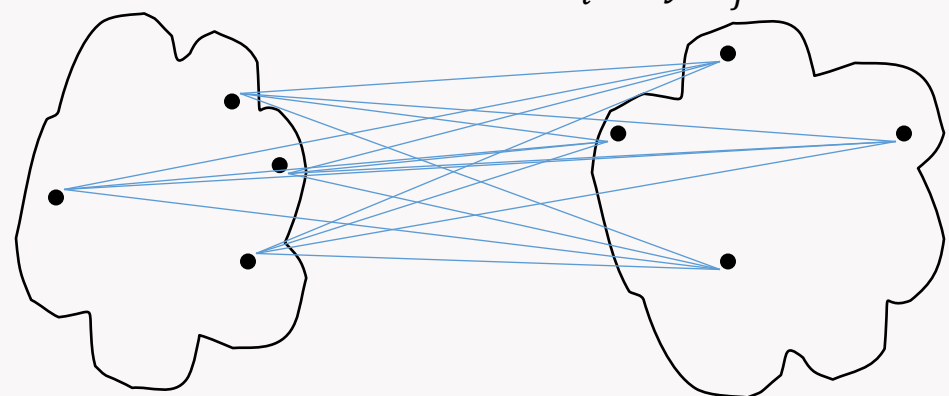
- 凝聚层次聚类计算簇之间的邻近度度量时可以采取不同的计算方式，这些方式常从图的角度进行定义和区分：
- **全链** (Complete-linkage)：簇之间的邻近度定义为不同簇中两个**最远**的数据对象的距离。

$$d_{cl}(C_i, C_j) = \max_{x \in C_i, y \in C_j} dist(x, y)$$



□ 簇之间的邻近度度量

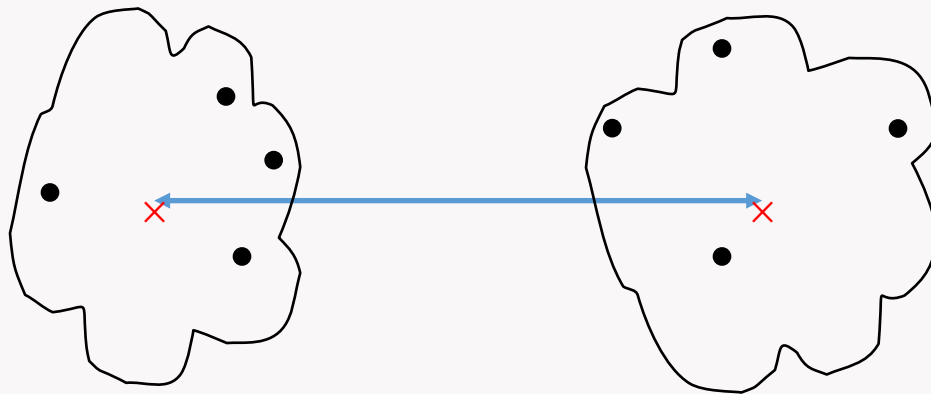
- 凝聚层次聚类计算簇之间的邻近度度量时可以采取不同的计算方式，这些方式常从图的角度进行定义和区分：
- **均链** (Average-linkage)：簇之间的邻近度定义为不同簇中**所有**数据对象之间距离的**均值**。

$$d_{al}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} \text{dist}(x, y)$$


□ 簇之间的邻近度度量

- 凝聚层次聚类计算簇之间的邻近度度量时可以采取不同的计算方式，这些方式常从图的角度进行定义和区分：
- **质心距离** (Distance between centroids)：簇之间的邻近度定义为不同簇的**质心**之间的距离。

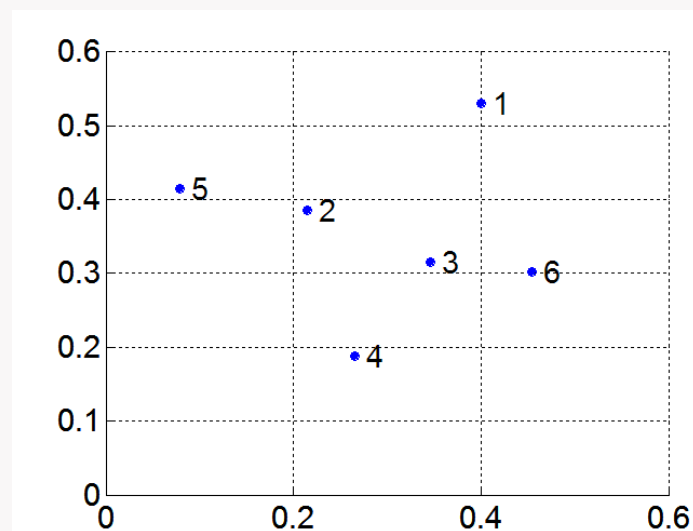
$$d_c(C_i, C_j) = \text{dist}(c_i, c_j)$$



□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

| 数据对象 | X坐标 | Y坐标 |
|------|--------|--------|
| p1 | 0.4005 | 0.5306 |
| p2 | 0.2148 | 0.3854 |
| p3 | 0.3457 | 0.3156 |
| p4 | 0.2652 | 0.1875 |
| p5 | 0.0789 | 0.4139 |
| p6 | 0.4548 | 0.3022 |

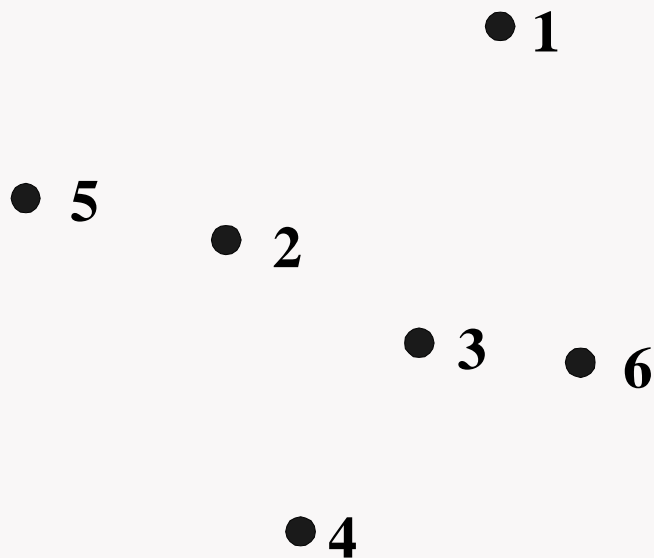


| | p1 | p2 | p3 | p4 | p5 | p6 |
|----|------|------|------|------|------|------|
| p1 | 0.00 | 0.24 | 0.22 | 0.37 | 0.34 | 0.23 |
| p2 | 0.24 | 0.00 | 0.15 | 0.20 | 0.14 | 0.25 |
| p3 | 0.22 | 0.15 | 0.00 | 0.15 | 0.28 | 0.11 |
| p4 | 0.37 | 0.20 | 0.15 | 0.00 | 0.29 | 0.22 |
| p5 | 0.34 | 0.14 | 0.28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

邻近度矩阵

□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

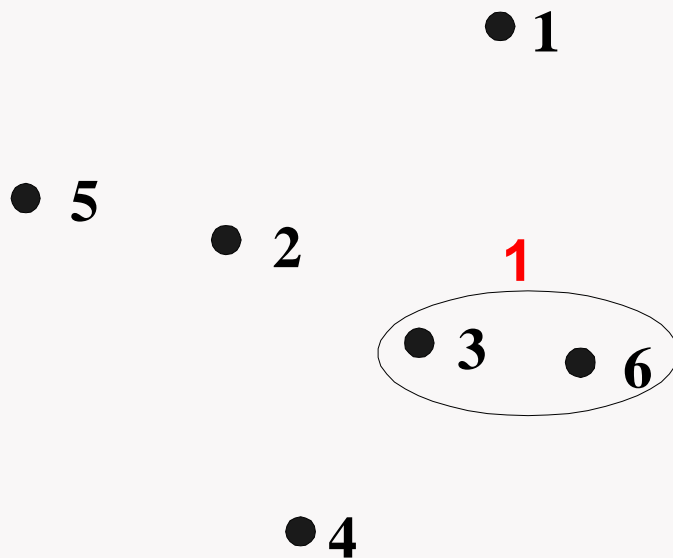


| 聚类 | 数据对象 |
|----|------|
| C1 | {p1} |
| C2 | {p2} |
| C3 | {p3} |
| C4 | {p4} |
| C5 | {p5} |
| C6 | {p6} |

| | 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 |

□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

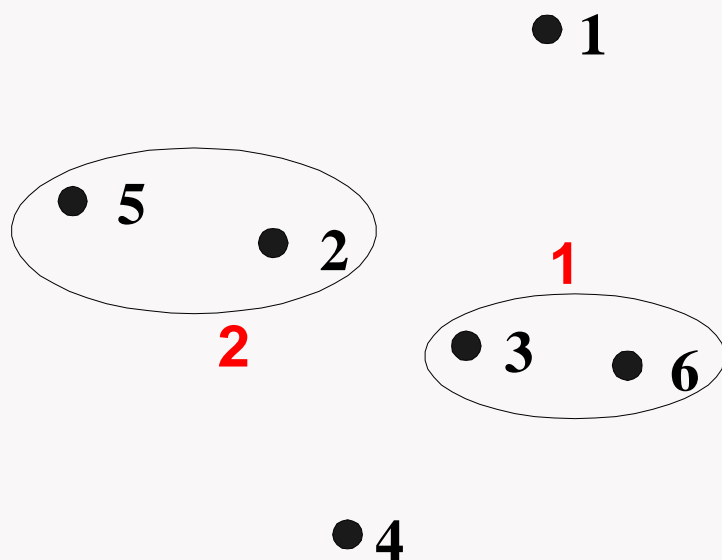


| 聚类 | 数据对象 |
|----|---------|
| C1 | {p1} |
| C2 | {p2} |
| C3 | {p3,p6} |
| C4 | {p4} |
| C5 | {p5} |

| | 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 |

□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

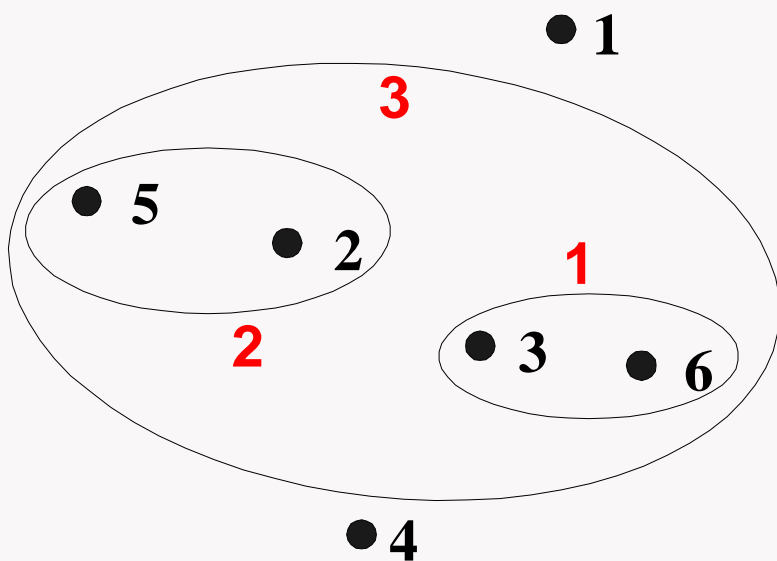


| 聚类 | 数据对象 |
|----|---------|
| C1 | {p1} |
| C2 | {p2,p5} |
| C3 | {p3,p6} |
| C4 | {p4} |

| | 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 |

□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

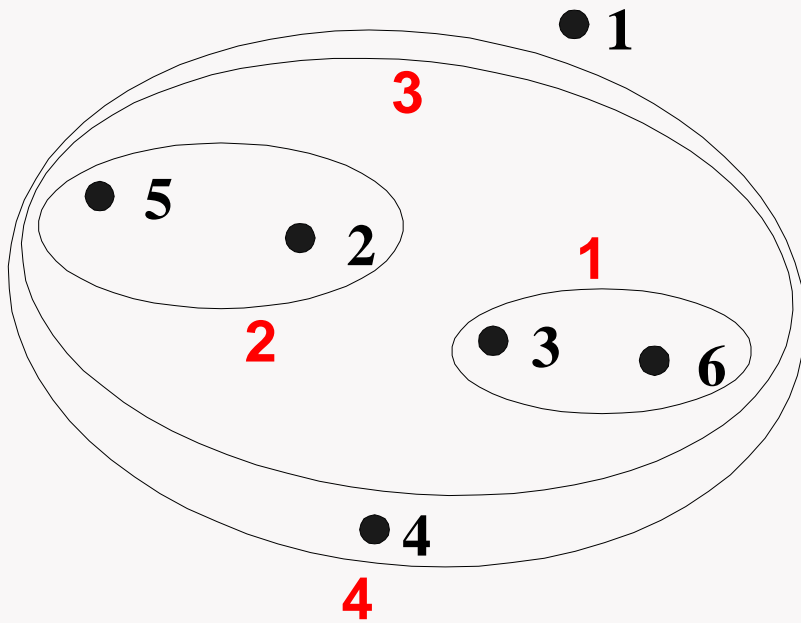


| 聚类 | 数据对象 |
|----|---------------|
| C1 | {p1} |
| C2 | {p2,p3,p5,p6} |
| C4 | {p4} |

| | 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 |

□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

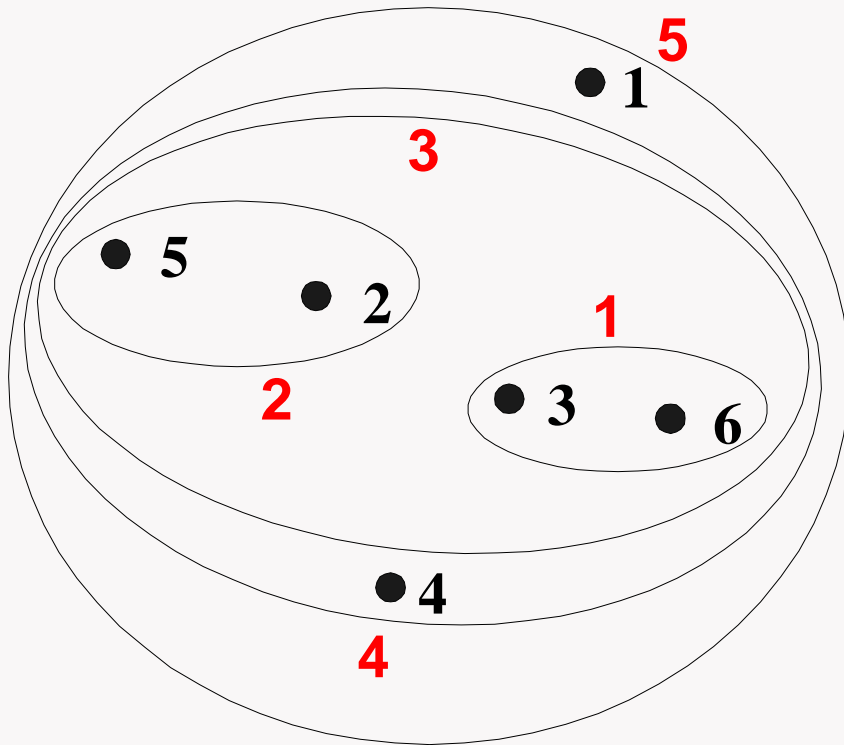


| 聚类 | 数据对象 |
|----|------------------|
| C1 | {p1} |
| C2 | {p2,p3,p4,p5,p6} |

| | 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 |

□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

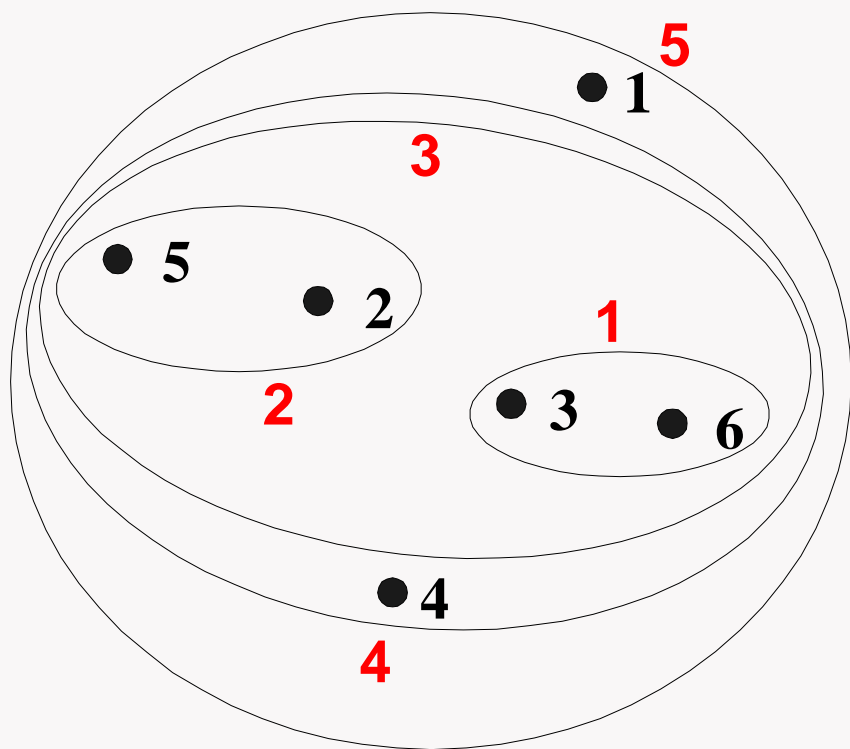


| 聚类 | 数据对象 |
|----|----------------------|
| C1 | {p1, p2,p3,p4,p5,p6} |

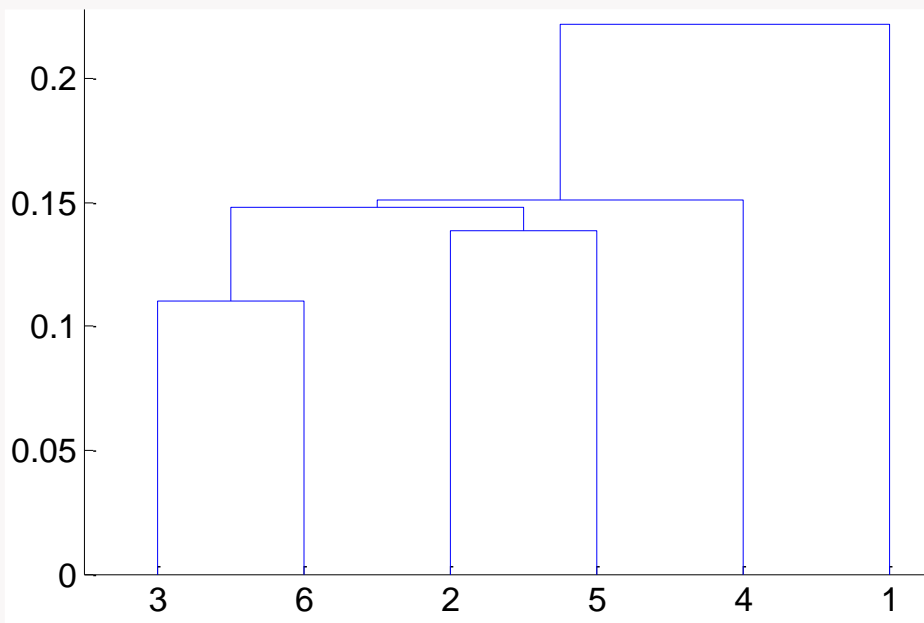
| | p1 | p2 | p3 | p4 | p5 | p6 |
|----|------|------|------|------|------|------|
| p1 | 0.00 | 0.24 | 0.22 | 0.37 | 0.34 | 0.23 |
| p2 | 0.24 | 0.00 | 0.15 | 0.20 | 0.14 | 0.25 |
| p3 | 0.22 | 0.15 | 0.00 | 0.15 | 0.28 | 0.11 |
| p4 | 0.37 | 0.20 | 0.15 | 0.00 | 0.29 | 0.22 |
| p5 | 0.34 | 0.14 | 0.28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

□ 凝聚层次聚类

■ 单链凝聚层次聚类示例

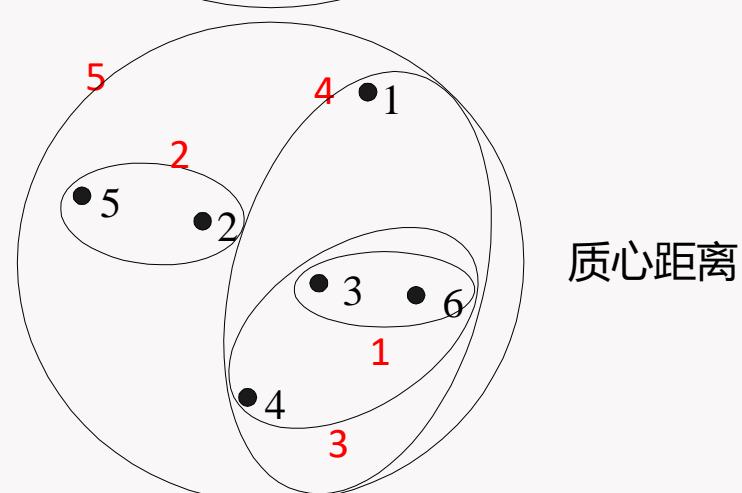
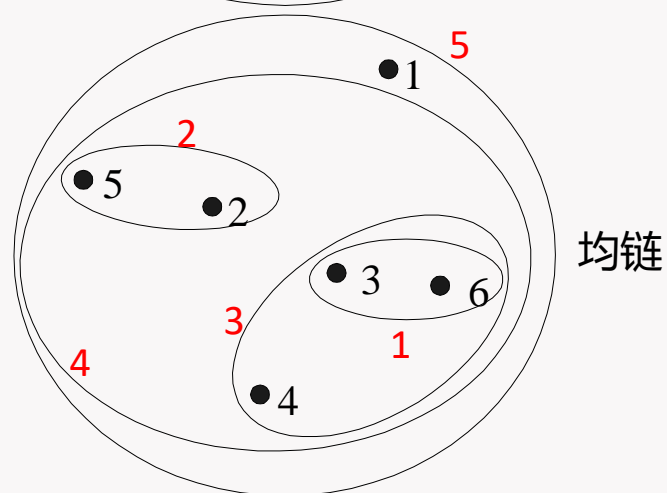
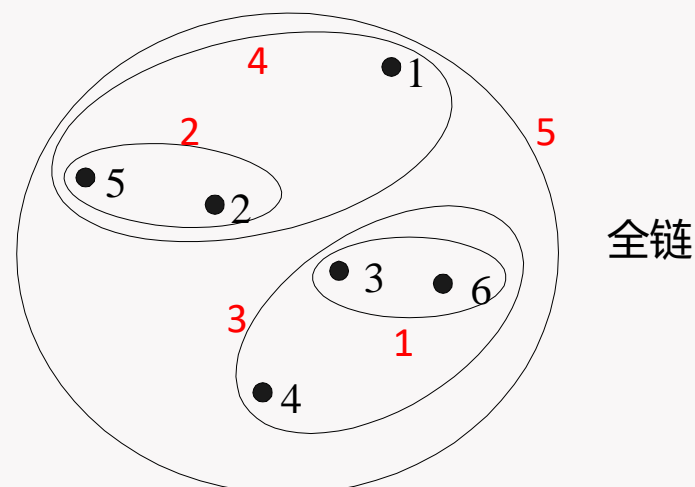
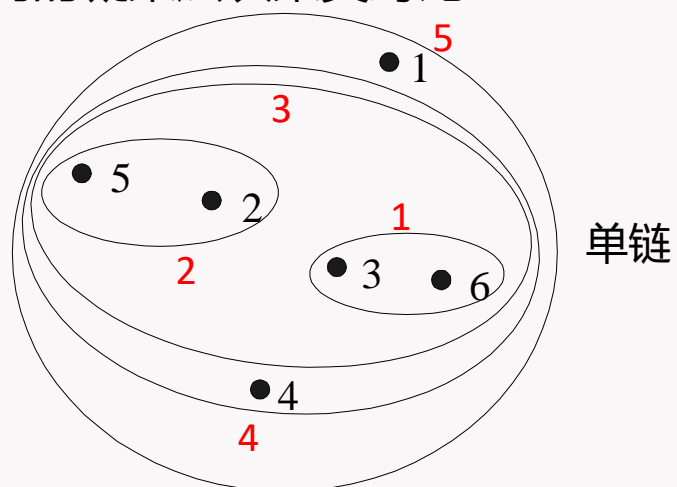


| 聚类 | 数据对象 |
|----|--------------------------|
| C1 | {p1, p2, p3, p4, p5, p6} |



凝聚层次聚类

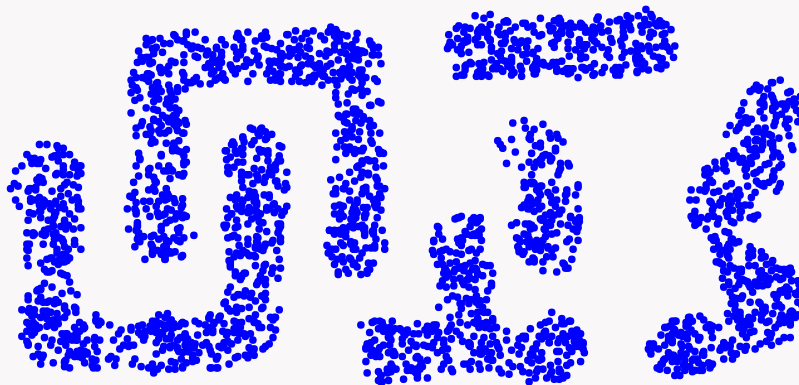
不同的凝聚层次聚类对比



□ 凝聚层次聚类

■ 不同的凝聚层次聚类对比

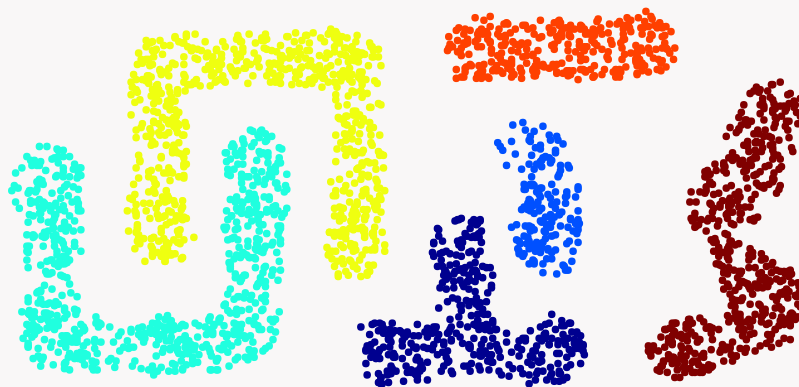
| | 单链 | 全链 | 均链 | 质心距离 |
|----|---------------|-----------|--------------------|--------------------|
| 优点 | 可以处理非椭圆形的簇 | | 簇间的距离不易受到噪声或异常值的影响 | 簇间的距离不易受到噪声或异常值的影响 |
| 缺点 | 聚类结果对噪声或异常值敏感 | 倾向于分裂较大的簇 | 倾向于形成球形的簇 | 倾向于形成球形的簇 |



□ 凝聚层次聚类

■ 不同的凝聚层次聚类对比

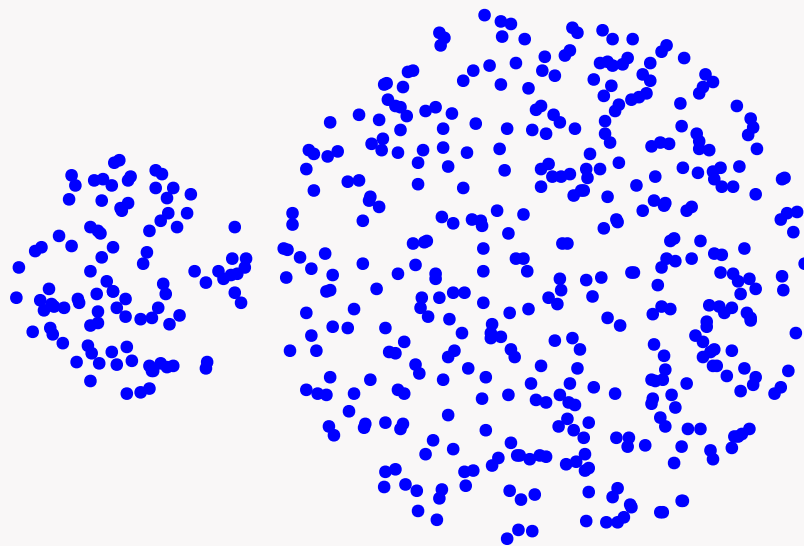
| | 单链 | 全链 | 均链 | 质心距离 |
|----|---------------|-----------|--------------------|--------------------|
| 优点 | 可以处理非椭圆形的簇 | | 簇间的距离不易受到噪声或异常值的影响 | 簇间的距离不易受到噪声或异常值的影响 |
| 缺点 | 聚类结果对噪声或异常值敏感 | 倾向于分裂较大的簇 | 倾向于形成球形的簇 | 倾向于形成球形的簇 |



□ 凝聚层次聚类

■ 不同的凝聚层次聚类对比

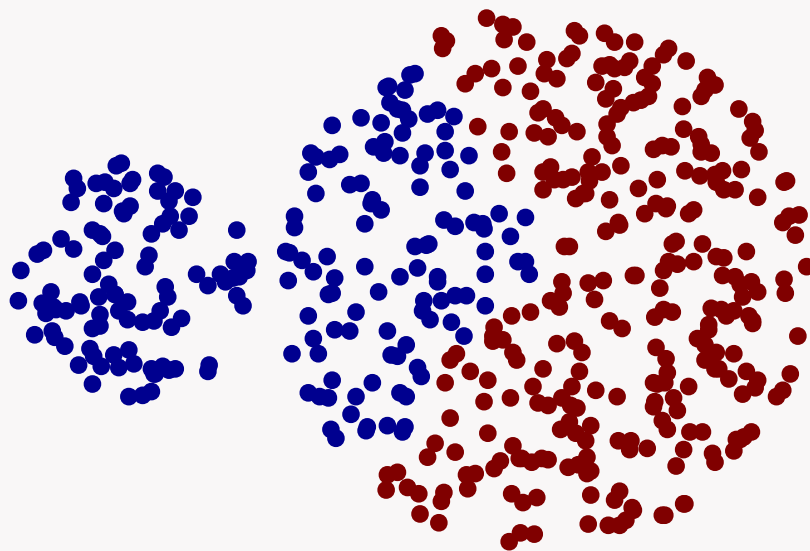
| | 单链 | 全链 | 均链 | 质心距离 |
|----|---------------|-----------|--------------------|--------------------|
| 优点 | 可以处理非椭圆形的簇 | | 簇间的距离不易受到噪声或异常值的影响 | 簇间的距离不易受到噪声或异常值的影响 |
| 缺点 | 聚类结果对噪声或异常值敏感 | 倾向于分裂较大的簇 | 倾向于形成球形的簇 | 倾向于形成球形的簇 |



□ 凝聚层次聚类

■ 不同的凝聚层次聚类对比

| | 单链 | 全链 | 均链 | 质心距离 |
|----|---------------|-----------|--------------------|--------------------|
| 优点 | 可以处理非椭圆形的簇 | | 簇间的距离不易受到噪声或异常值的影响 | 簇间的距离不易受到噪声或异常值的影响 |
| 缺点 | 聚类结果对噪声或异常值敏感 | 倾向于分裂较大的簇 | 倾向于形成球形的簇 | 倾向于形成球形的簇 |



□ 凝聚层次聚类方法的时间复杂度和空间复杂度

- 凝聚层次聚类方法需要存储包含 N^2 个元素的距离矩阵，其中 N 表示数据对象的数量。记录每个数据对象的簇信息需要约 N 个存储空间。所以总的空间复杂度为 $O(N^2 + N)$ 。
- 凝聚层次聚类方法需要 $O(N^2)$ 的时间计算距离矩阵，迭代过程的时间复杂度大致为 $O(N^2 \log(N))$ ，所以总的时间复杂度为 $O(N^2(\log N + 1))$ 。

□ 凝聚层次聚类方法的特点

■ 优点：

- 独有的层次结构，特别适合某些应用；
- 更易产生高质量的聚类；
- 对数据类型的适应性较好，易于拓展；
- 不需要指定簇的数量。

■ 缺点：

- 可伸缩性较差；
- 合并的操作无法撤销，导致局部最优聚类无法转换为全局最优聚类。



Chapter 5.4

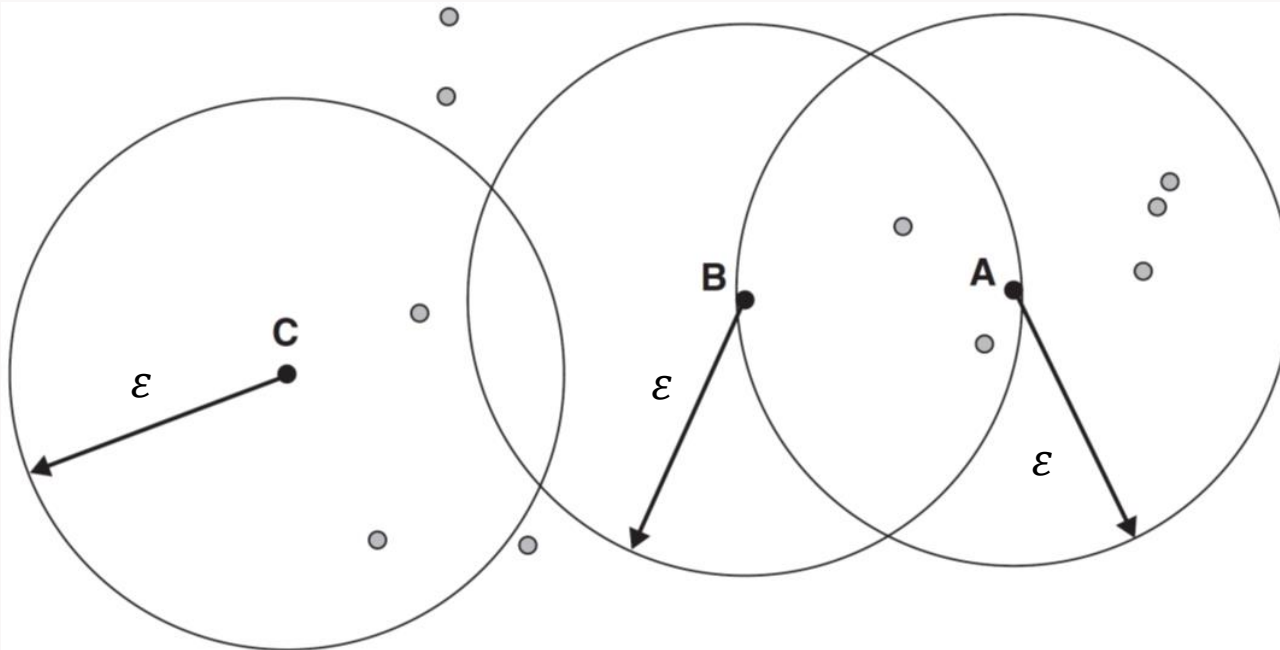
基于密度的聚类方法

□ 基于密度的聚类方法

- 基于密度的聚类方法通过寻找被低密度区域（区域内包含的数据对象少于阈值）分离的高密度区域的方式实现聚类。其中，**DBSCAN** (Density-Based Spatial Clustering of Application with Noise) 是一种简单有效、被广泛使用的基于密度的聚类方法。

DBSCAN

- DBSCAN通过用户指定的参数 ε (**epsilon**, $\varepsilon > 0$) 确定每个数据对象的邻域半径。所以数据对象 x 的 ε -邻域是以 x 为中心, ε 为半径的空间。
- **核心对象** (Core object) : 如果一个数据对象的 ε -邻域至少包含 n_{eps} 个数据对象 (包含自身), 则表示该数据对象处于高密度区域, 并称之为核心对象。



假设 $n_{eps} = 7$

$$N_{eps}(A) = 7$$

$$N_{eps}(B) = 4$$

$$N_{eps}(C) = 3$$

所以A是核心对象

□ DBSCAN

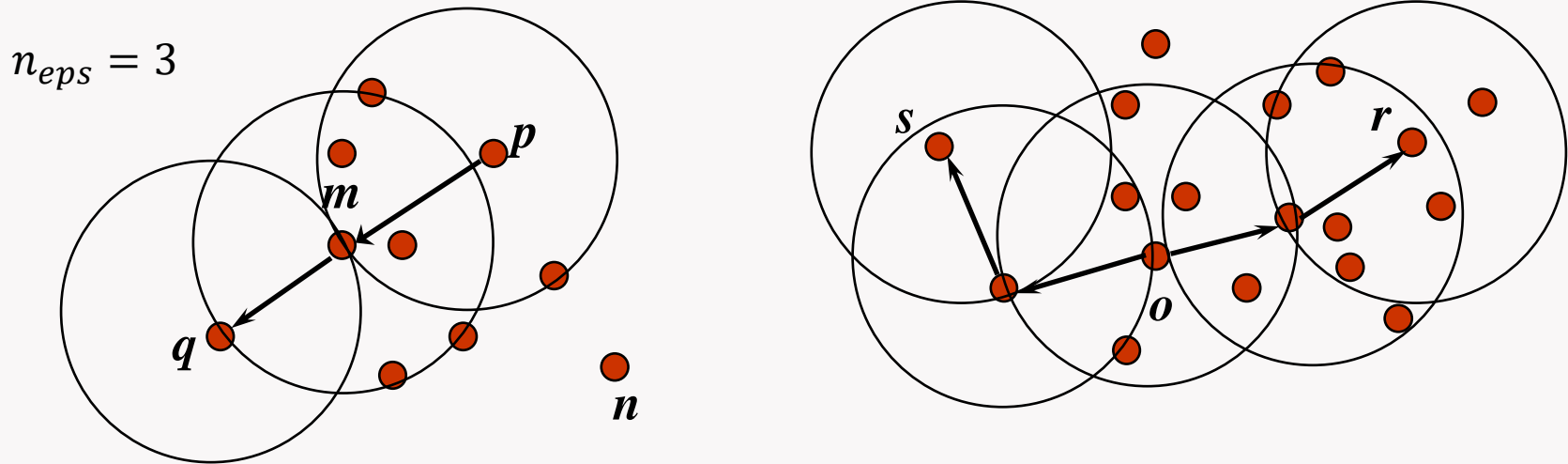
- 给定参数 ε 和 n_{eps} , DBSCAN的目的是找出数据集中所有的核心对象, 并利用核心对象和它们的邻域形成高密度区域, 每个高密度区域都对应一个簇。利用核心对象形成高密度区域时使用了如下概念:
 - **直接密度可达的** (Directly density-reachable) : 对于**对象** p 和**核心对象** q , 如果 p 在 q 的 ε -邻域, 则称 p 是从核心对象 q 直接密度可达的。如果 p 不是核心对象, 则直接密度可达关系是非对称的;
 - **密度可达的** (Density-reachable) : 对于**对象** p 和**核心对象** q , 如果存在一个**核心对象链** o_1, o_2, \dots, o_m , 使得 q 到 o_1 、 o_i 到 o_{i+1} 、 o_m 到 p 是直接密度可达的, 则称 p 是从核心对象 q 是密度可达的。如果 p 不是核心对象, 则直接密度可达关系是非对称的。

□ DBSCAN

- 给定参数 ε 和 n_{eps} , DBSCAN的目的是找出数据集中所有的核心对象, 并利用核心对象和它们的邻域形成高密度区域, 每个高密度区域都对应一个簇。利用核心对象形成高密度区域时使用了如下概念:
- **密度相连的** (Density-connected): 对于**对象** p 和**对象** q , 如果存在一个核心对象 o , 使得 p 和 q 都是从 o 密度可达的, 则称 p 和 q 是密度相连的。密度相连关系是对称的。

DBSCAN

三种关系的示例



- m , p , o 和 r 是核心对象;
- q 是从 p 密度可达的, 但 p 不是从 q 密度可达的;
- 对象 s 是从核心对象 r 密度可达的, 对象 s 和对象 r 是密度相连的。

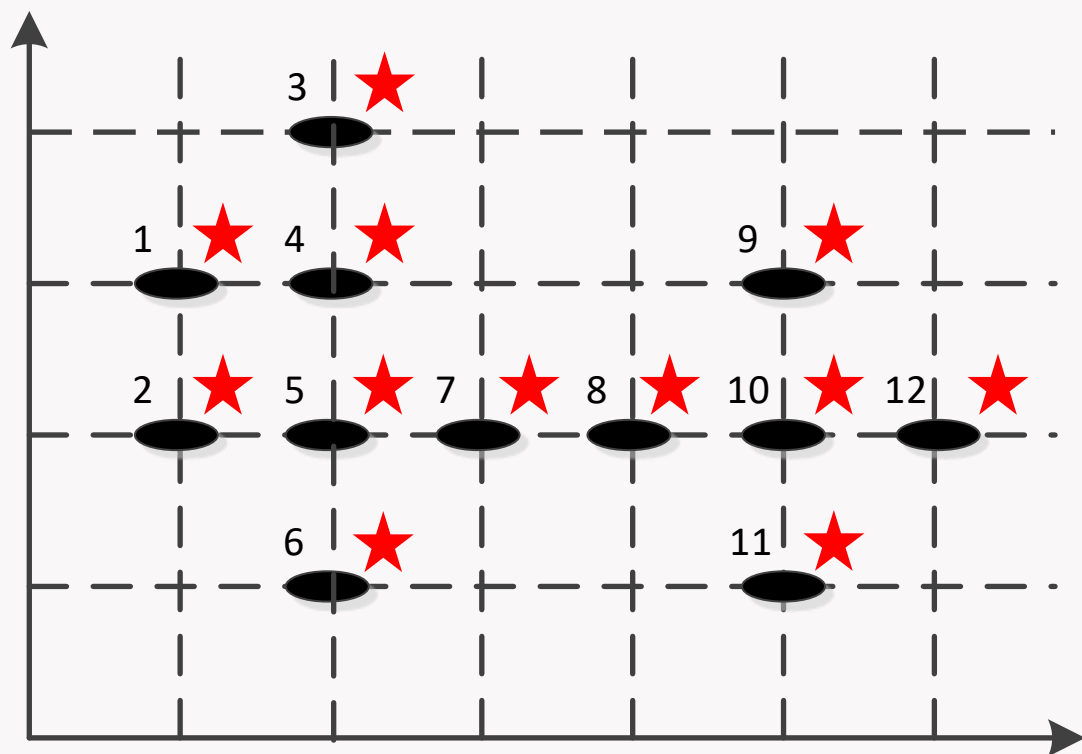
DBSCAN

DBSCAN算法主要包含以下几个步骤：

- 初始时将数据集中所有数据对象标记为 “unvisited” ,表示未计算邻域信息；
- 随机选择一个 “unvisited” 对象 p 并将标记更改为 “visited” , 如果判断为非核心对象, 则标记为**噪声点**。否则为 p 创建一个簇 C , 将 p 的 ε - 邻域的数据对象都放入 C 的候选集合 S_C 中；
- 逐个将 S_C 中**不属于其他簇的数据对象**放入 C 中, 并将标记更改为 “visited” 。特别的, 如果是该数据对象是核心对象, 则将其 ε - 邻域内的数据对象放入 S_C 中。此过程重复执行直到 C 不再增长 (S_C 为空) ；
- 重复第二步和第三步, 直到所有数据对象的标记都是 “visited” 。

DBSCAN

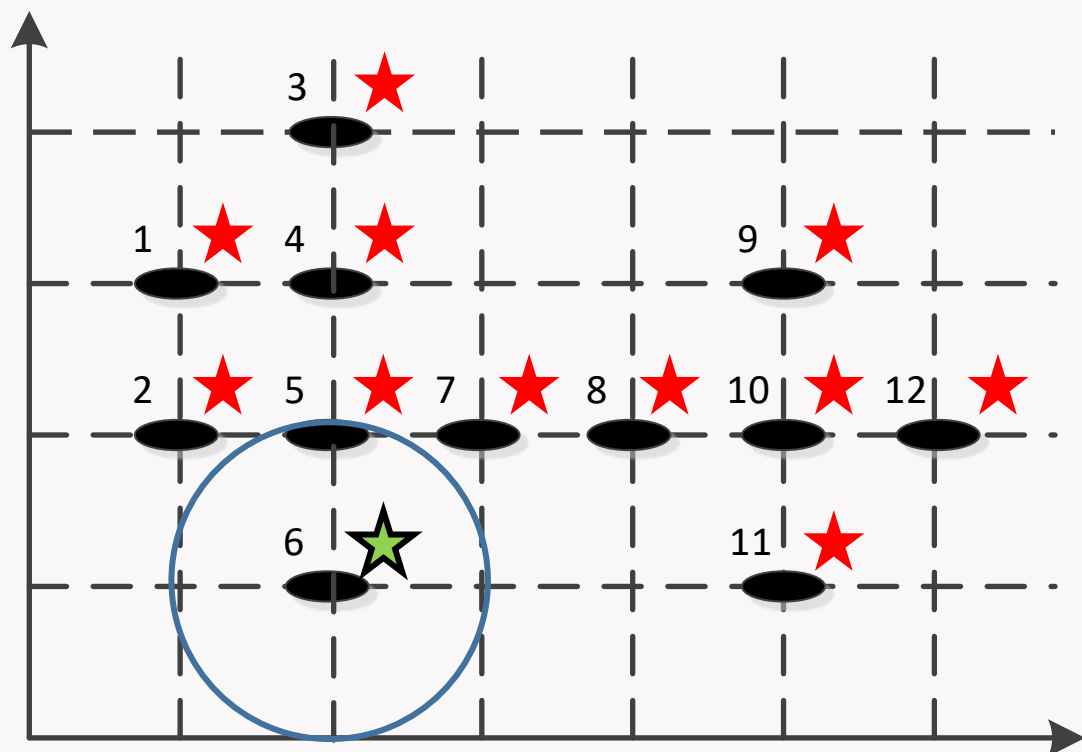
- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。



标记所有的数据对象为unvisited

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。



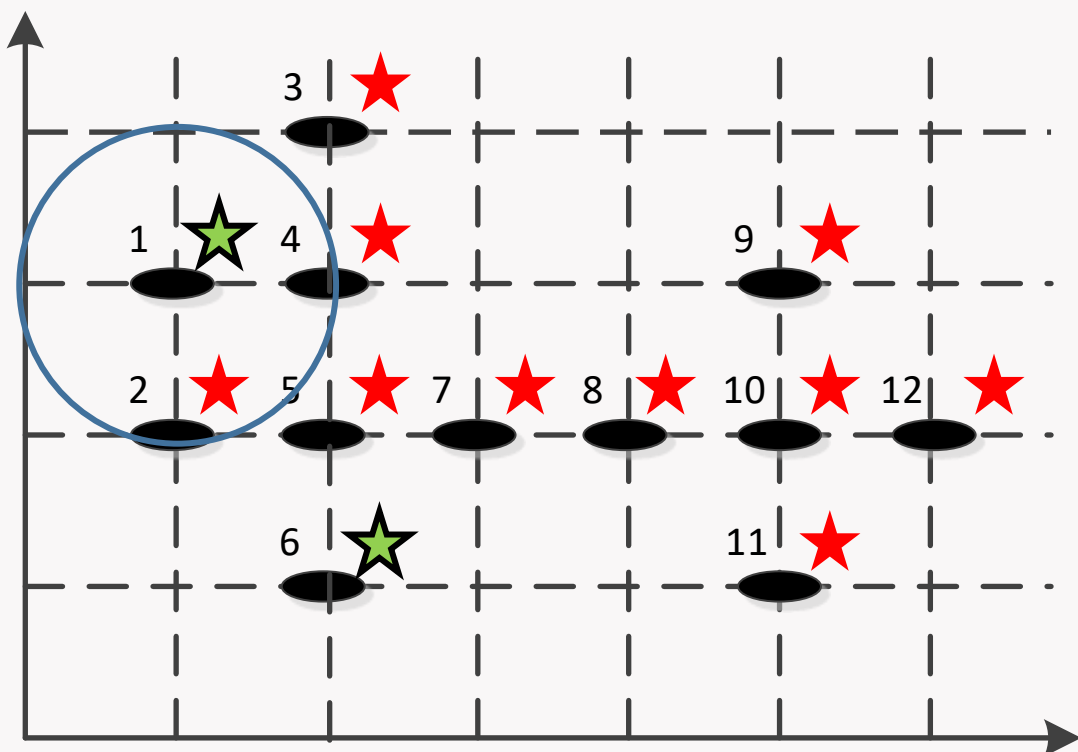
随机选择 unvisited 的数据对象6，将其标记更改为visited，根据距离矩阵有

$$N_{eps}(x_6) = 2$$

所以数据对象6是噪声对象。

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。



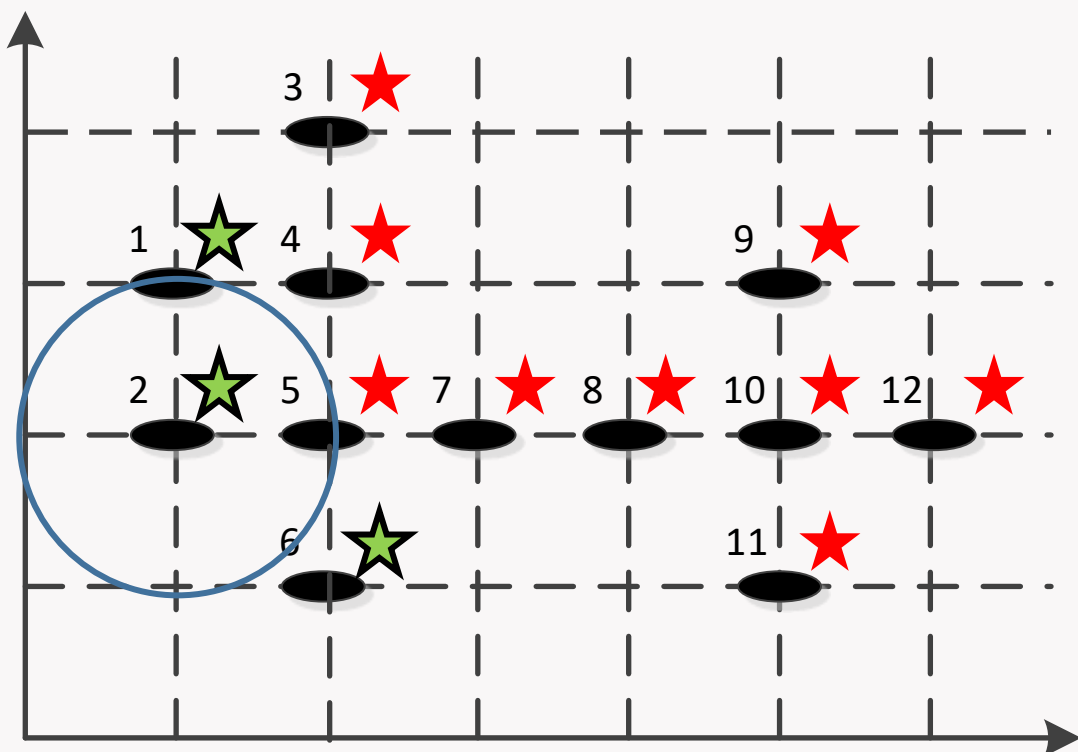
随机选择 unvisited 的数据对象1，将其标记更改为visited，根据距离矩阵有

$$N_{eps}(x_1) = 3$$

所以数据对象1是噪声对象。

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。



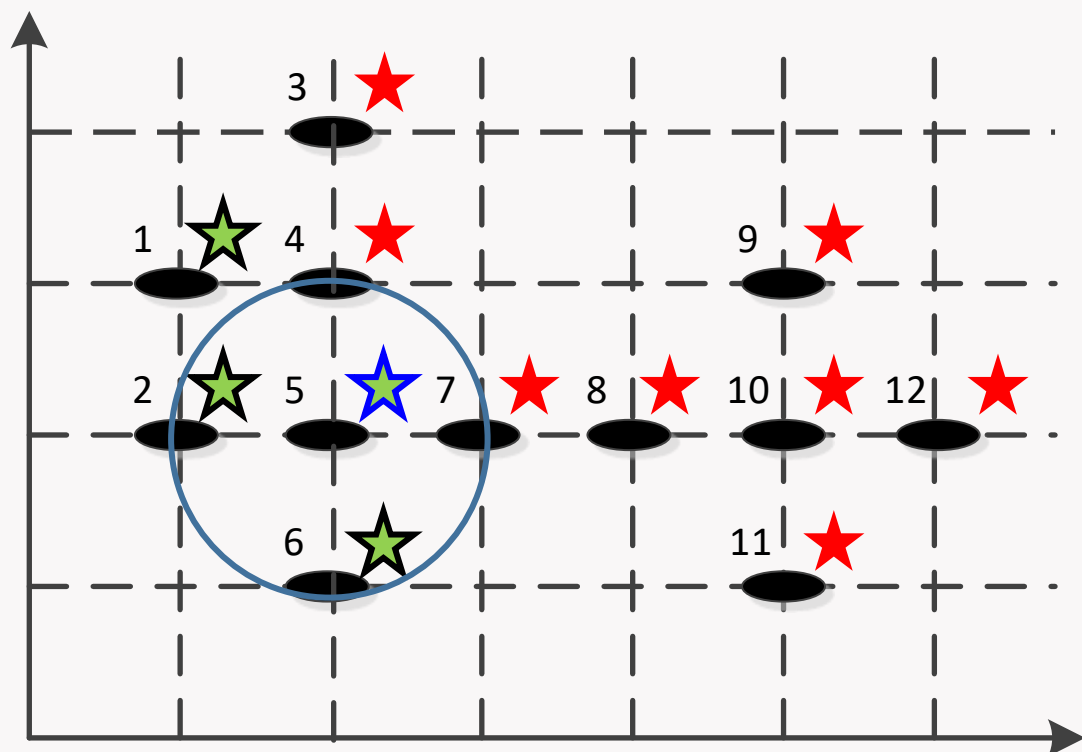
随机选择 unvisited 的数据对象2，将其标记更改为visited，根据距离矩阵有

$$N_{eps}(x_2) = 3$$

所以数据对象2是噪声对象。

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。



随机选择 unvisited 的数据对象5，将其标记更改为visited，根据距离矩阵有

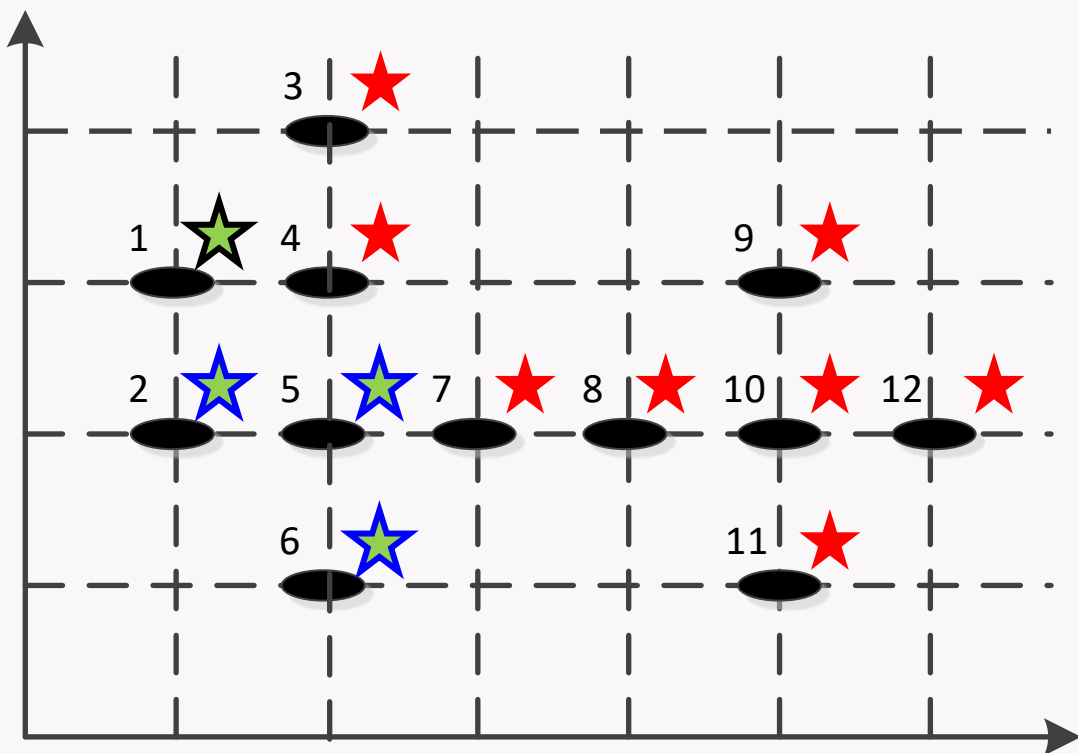
$$N_{eps}(x_5) = 5$$

所以数据对象5是核心对象。

| S_c | c |
|--------------------------|-----------|
| $\{x_2, x_4, x_6, x_7\}$ | $\{x_5\}$ |

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。

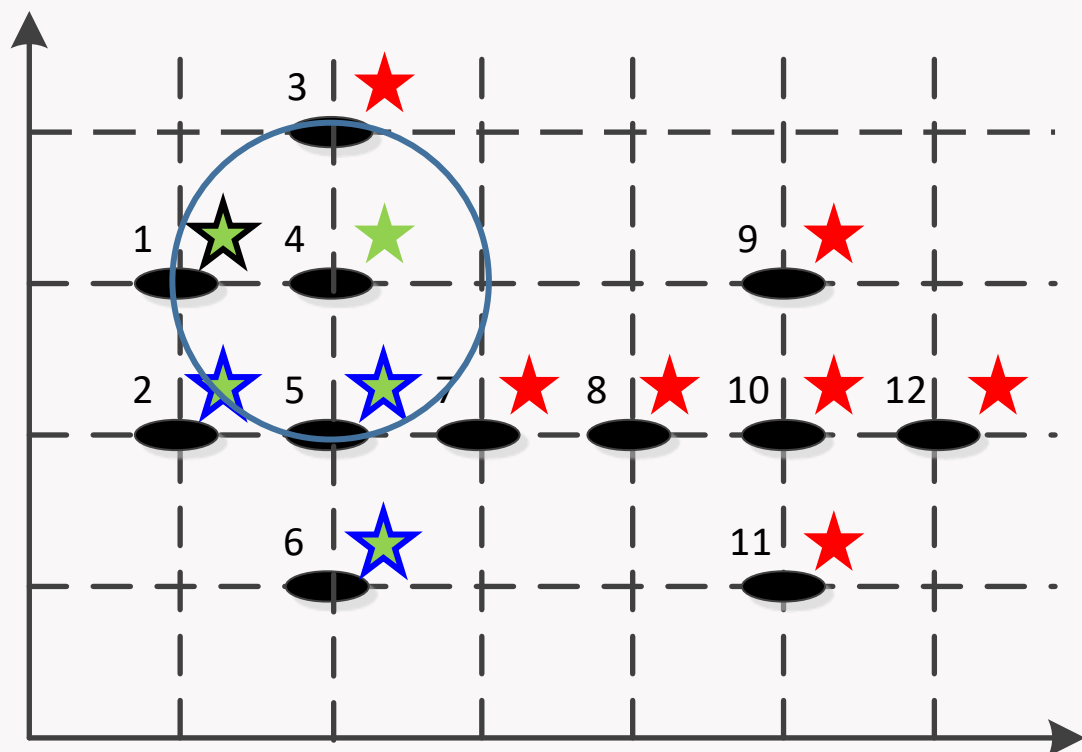


因为 x_2, x_6 的标识是 visited 而且不属于任何簇，所以直接放入 C 中。

| S_C | C |
|----------------|---------------------|
| $\{x_4, x_7\}$ | $\{x_5, x_2, x_6\}$ |

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。



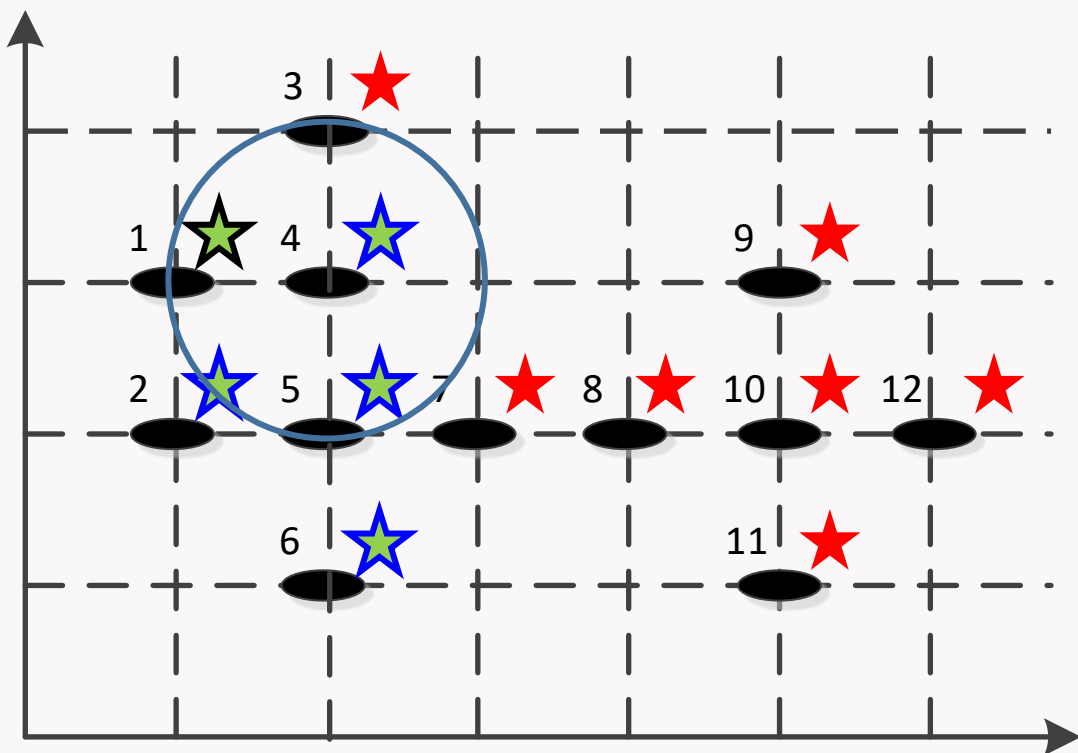
因为 x_4 的标识是 unvisited, 将其标记更改为 visited, 根据距离矩阵有

$$N_{eps}(x_4) = 4$$

所以数据对象4是核心对象。

| S_C | C |
|----------------|---------------------|
| $\{x_4, x_7\}$ | $\{x_5, x_2, x_6\}$ |

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。

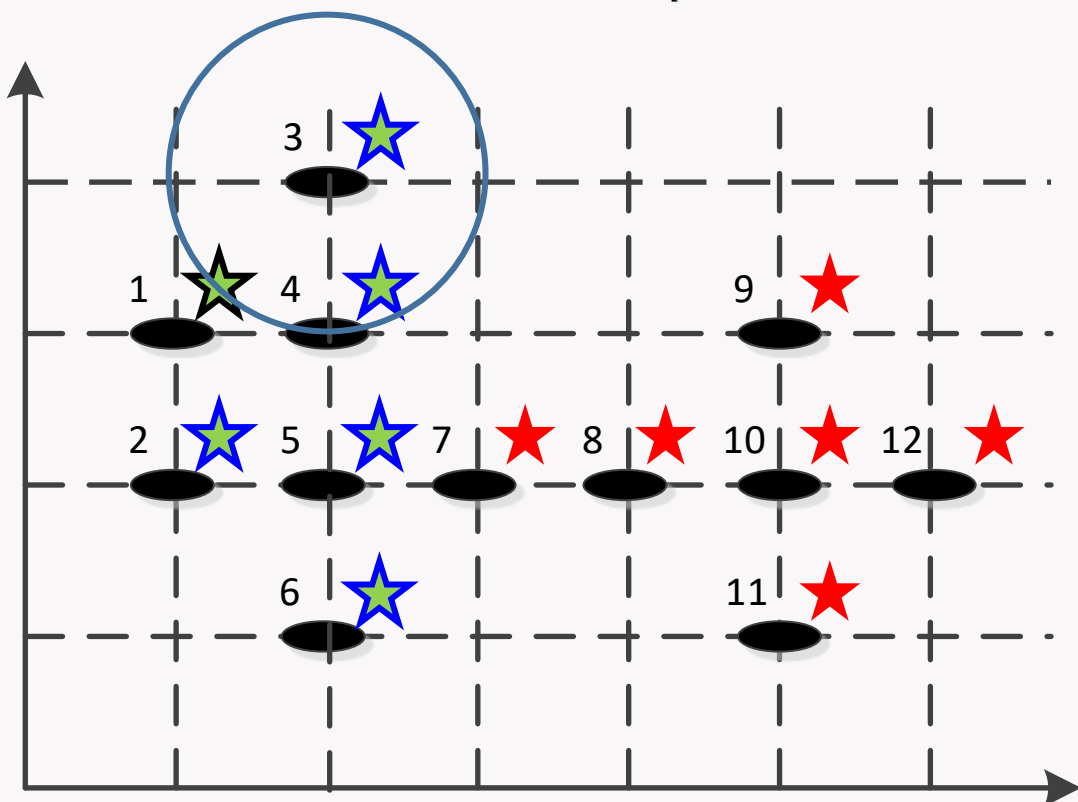


将 x_4 放入 C 中, 并将其 ε -邻域的不属于任何簇的数据对象放入 x_5 的 S_C 中。

| S_C | C |
|---------------------|--------------------------|
| $\{x_1, x_3, x_7\}$ | $\{x_5, x_2, x_6, x_4\}$ |

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。

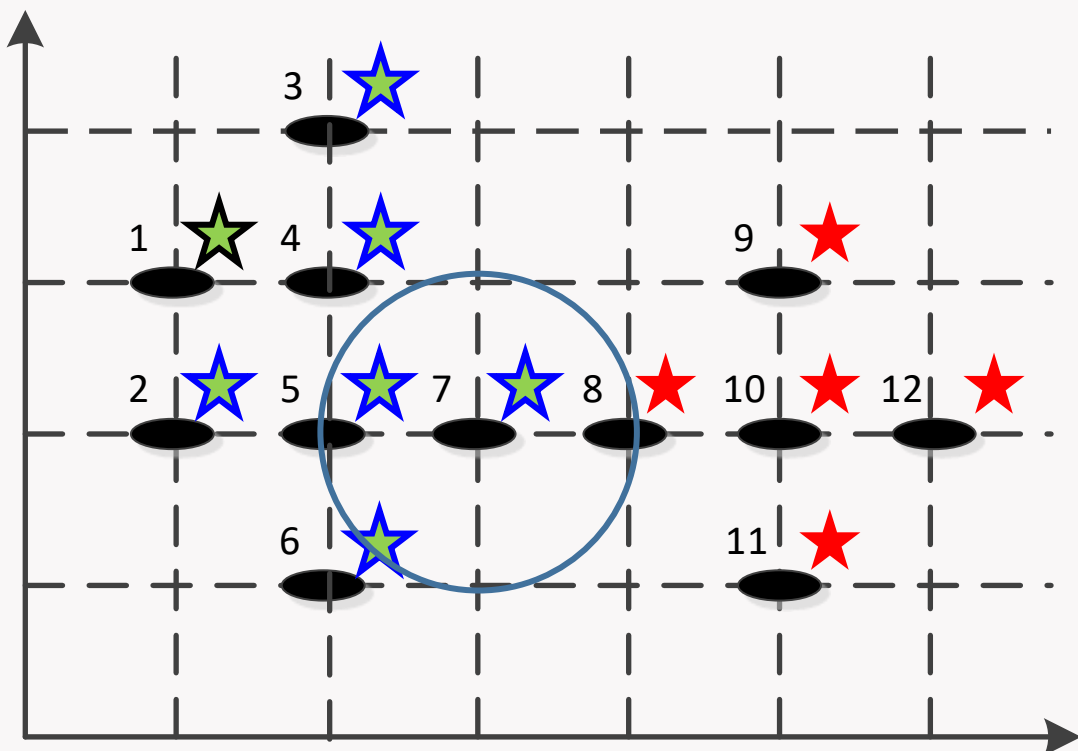


因为 x_3 的标识是 unvisited, 将其标记更改为 visited, 根据距离矩阵判断 x_3 是非核心对象, 所以直接放入 C 中。

| S_C | C |
|----------------|-------------------------------|
| $\{x_1, x_7\}$ | $\{x_5, x_2, x_3, x_6, x_4\}$ |

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。

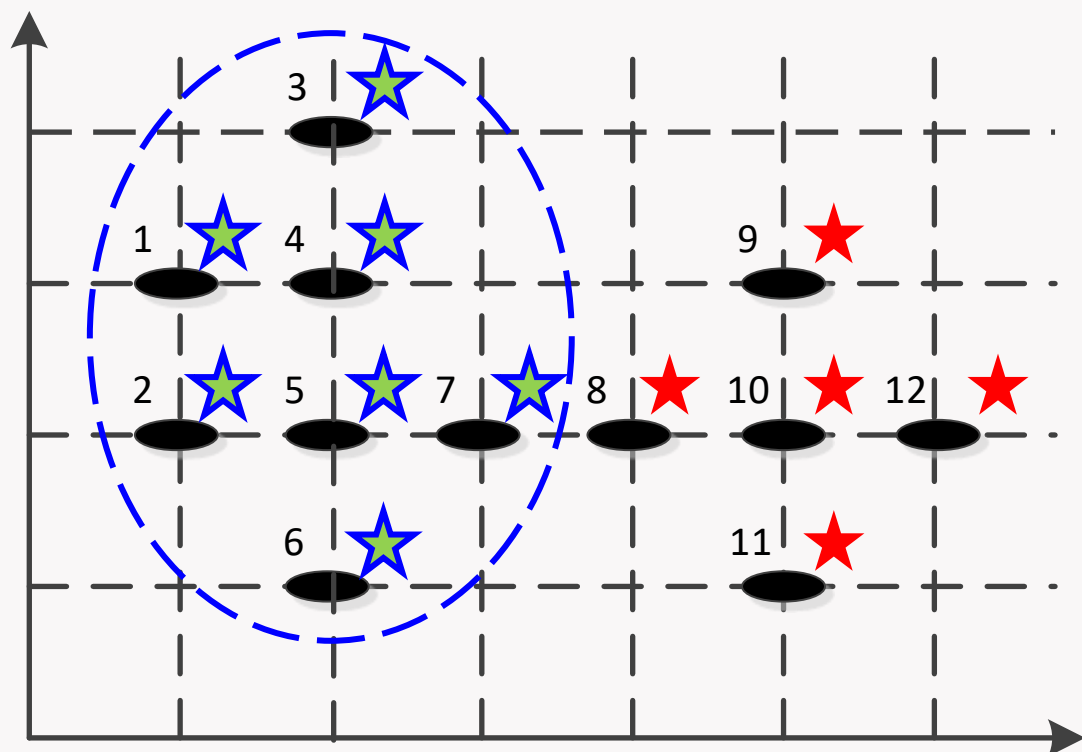


因为 x_7 的标识是 unvisited, 将其标记更改为visited, 根据距离矩阵判断 x_7 是非核心对象, 所以直接放入 C 中。

| S_C | C |
|-----------|------------------------------------|
| $\{x_1\}$ | $\{x_5, x_2, x_3, x_6, x_4, x_7\}$ |

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。

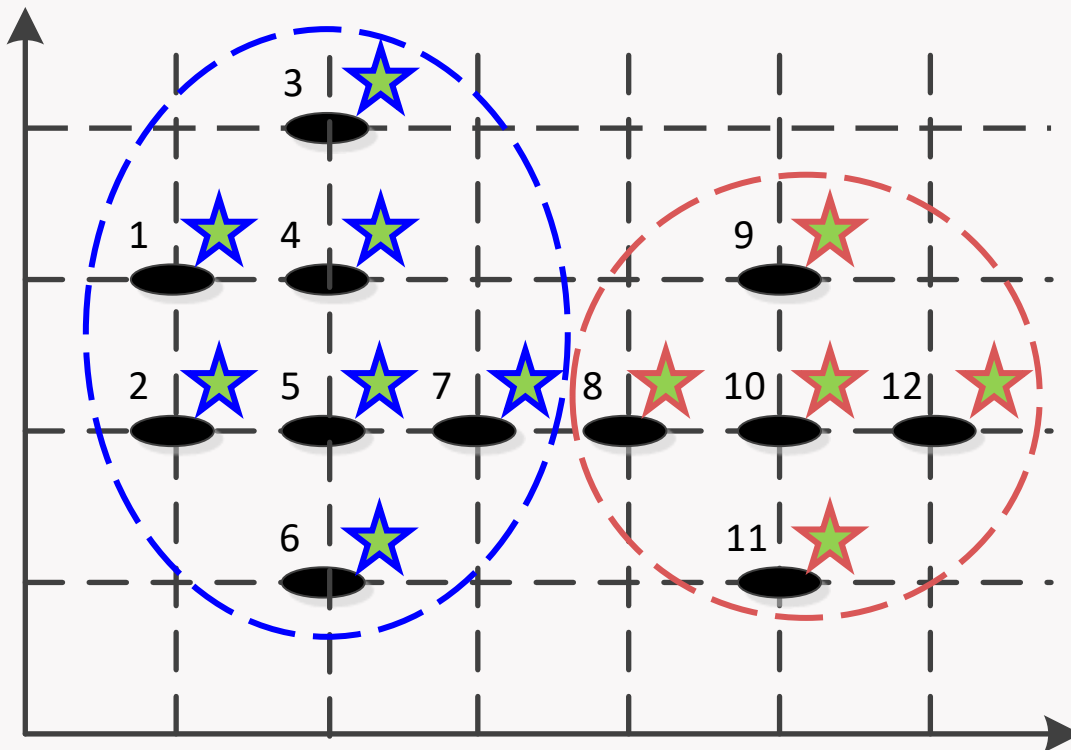


因为 x_1 的标识是 visited 而且不属于任何簇，所以直接放入 C 中。

| S_C | C |
|-------|-----------------------------------------|
| {} | $\{x_5, x_2, x_3, x_6, x_4, x_7, x_1\}$ |

DBSCAN

- DBSCAN示例：假设数据集中数据对象仅包含两个属性，在二维平面上的分布如图所示。其中， $\varepsilon = 1$ 且 $n_{eps} = 4$ 。

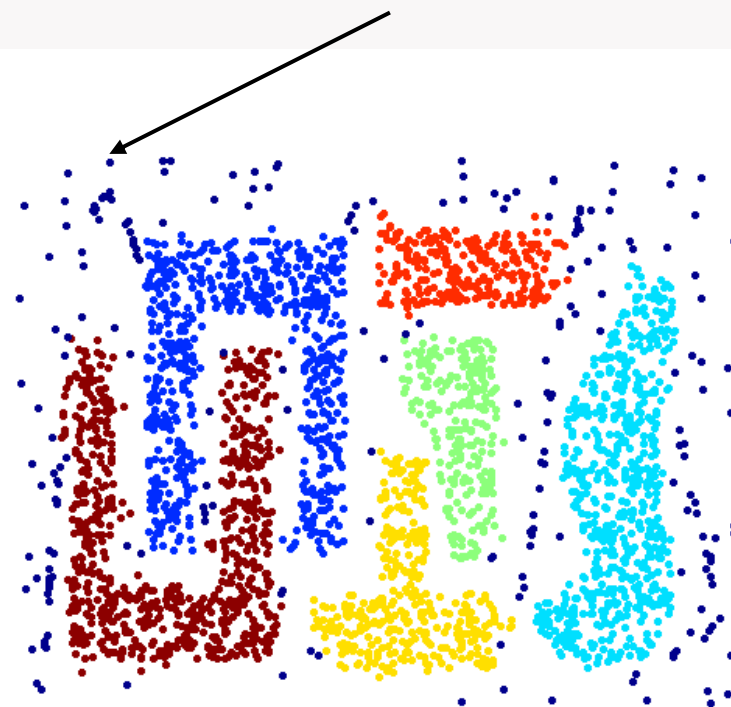


重复以上步骤，得到最终的聚类结果。

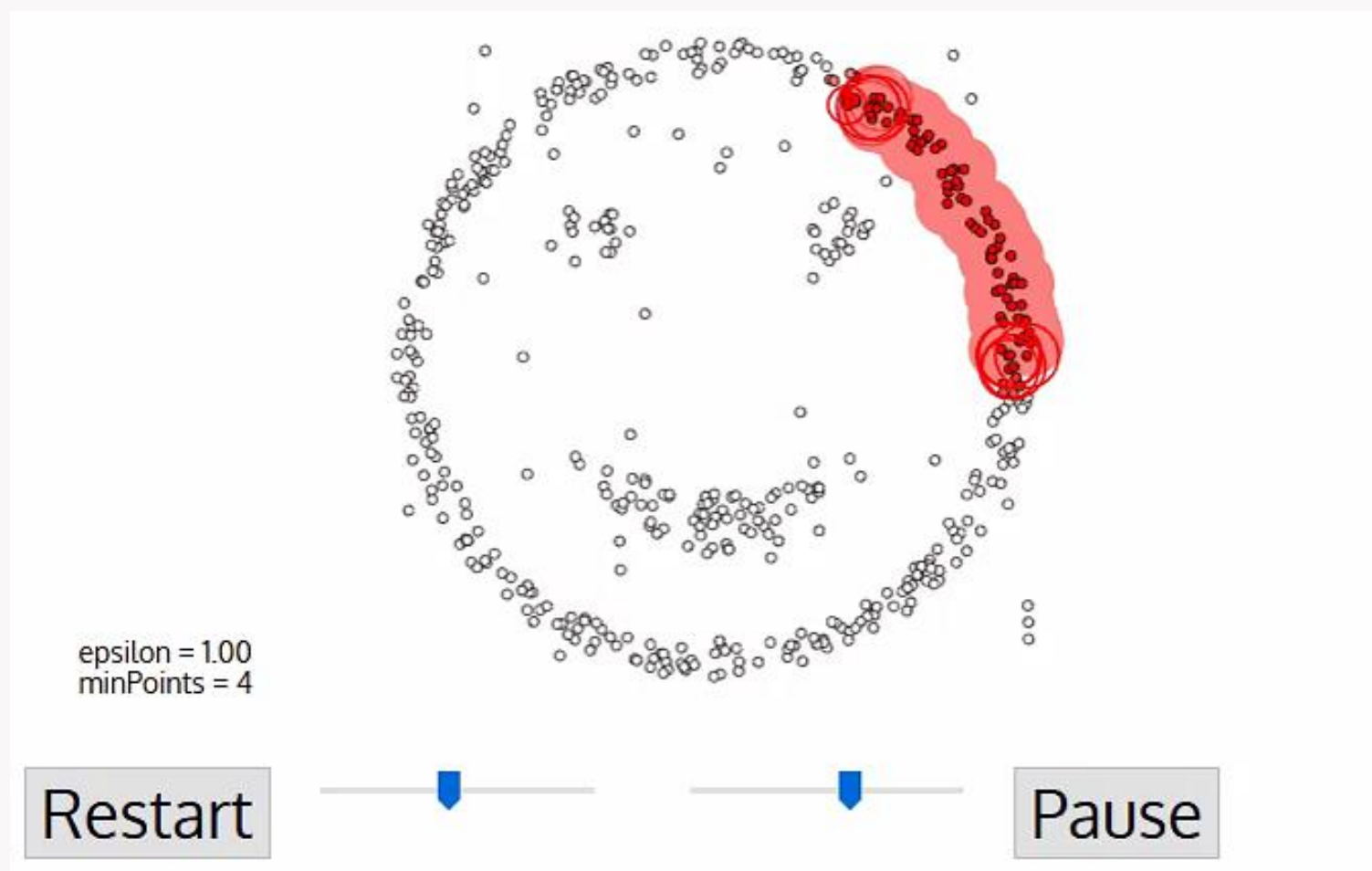
■ DBSCAN

■ 带噪声点的聚类结果

深蓝色的点表示被标记为噪声



DBSCAN演示



■ DBSCAN的空间复杂度和时间复杂度

- 对于每个数据对象，DBSCAN只需要记录簇编号和是否被 visited 等信息，所以空间复杂度是 $O(N)$ 。
- DBSCAN的时间复杂度为 $O(TN)$ ，其中 T 表示找出 ε -邻域中的数据对象所花费的时间。在最坏的情况下，DBSCAN的时间复杂度为 $O(N^2)$ 。使用某些快速检索的算法，可以使时间复杂度下降到 $O(N \log N)$ 。

▣ DBSCAN的特点

■ 优点:

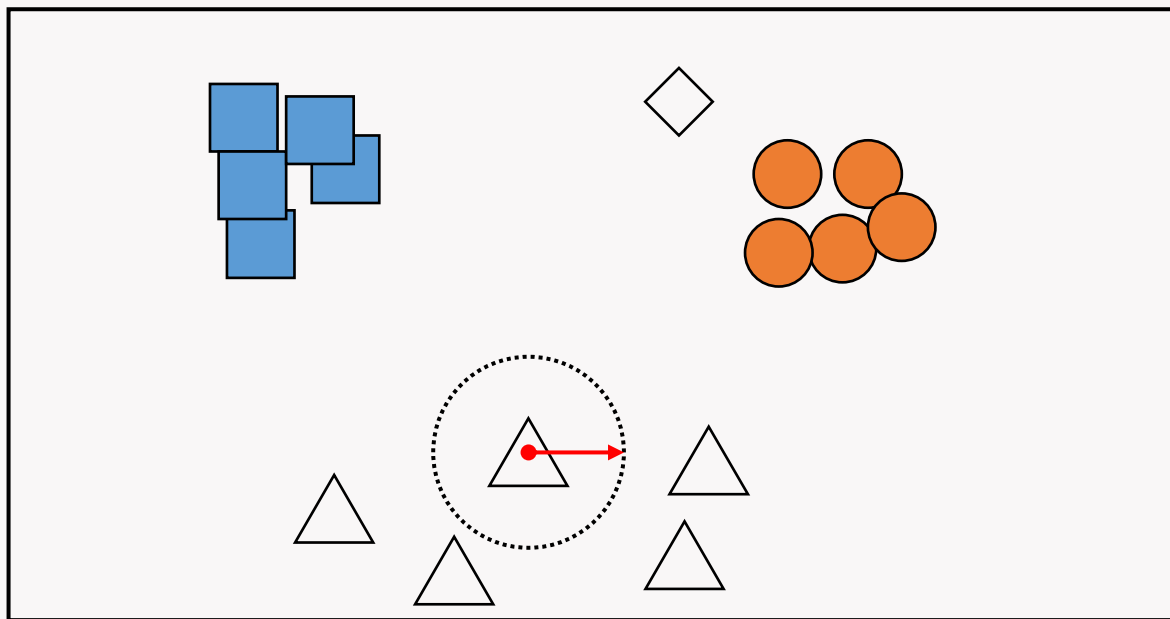
- 基于密度的定义使得DBSCAN且能处理任意形状和大小的簇;
- 可以在聚类同时发现异常点或噪声, 所以对数据集中的异常点或噪声不敏感;
- 不需要指定簇的数量, 不需要设置初始值。

■ 缺点:

- 可伸缩性较差;
- 高维空间中数据分布稀疏, 难以确定高密度的定义;
- 当空间中不同区域的密度差距很大时, 固定的 n_{eps} 可能使DBSCAN失效;
- 调试参数复杂, 需要对 ε 和 n_{eps} 进行联合调参。

▣ DBSCAN的改进

- 尽管DBSCAN能够根据给定的输入参数 ε 和 n_{eps} 聚类对象，但是DBSCAN对这些参数的设置非常敏感，且依赖于用户的经验。
- 现实的高维数据集通常具有非常倾斜的分布，**全局性**的密度参数不能很好地刻画其内在的聚类结构。



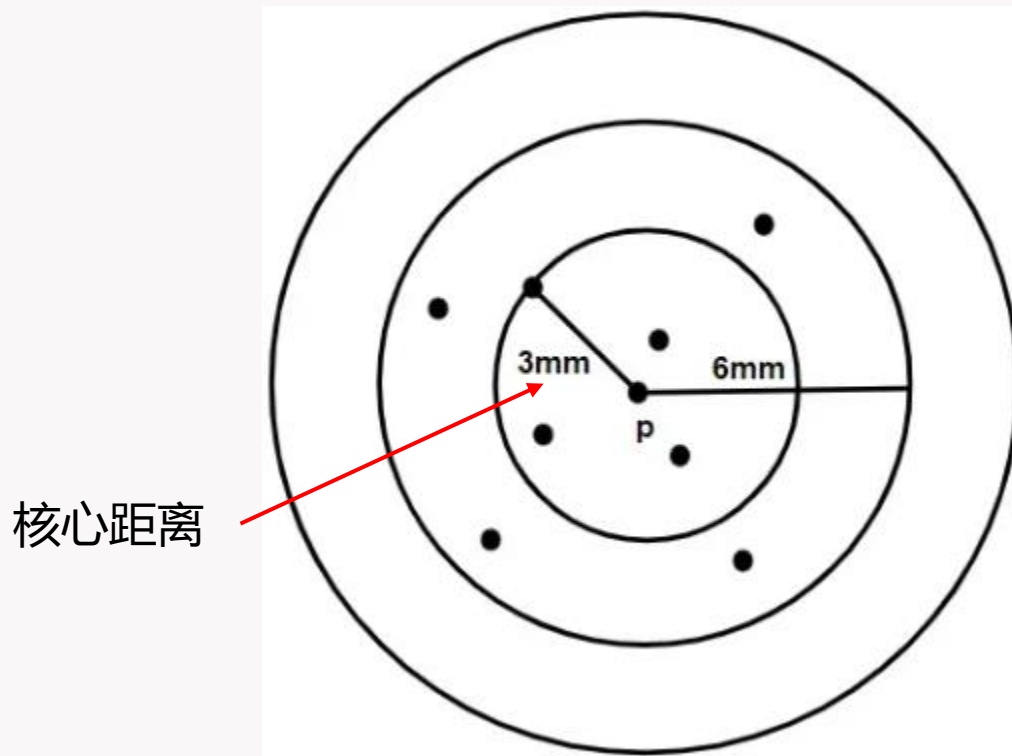
□ OPTICS

- 为了克服在聚类分析中使用一组全局参数的缺点，DBSCAN的研究团队又提出了 **OPTICS** (**O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure) 聚类分析算法。
- OPTICS算法不显式地产生数据的聚类，而是输出簇排序 (cluster ordering) 。这个排序是所有数据对象的线性表，描述了数据的基于密度的聚类结构。

□ OPTICS

■ 对于每个对象，OPTICS定义了两个概念：

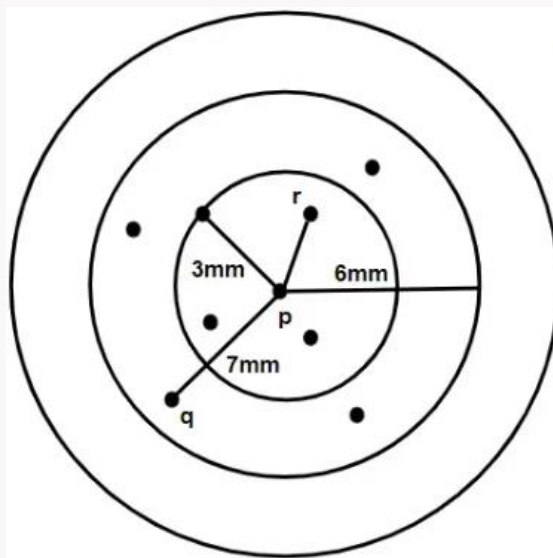
- 对象 p 的**核心距离** (core-distance)：使得 p 的 ε -邻域内至少有 n_{eps} 的最小 ε 值。或者说， p 的核心距离是使得 p 成为核心对象的最小半径阈值。



$$n_{eps} = 5$$

□ OPTICS

- 从对象 q 到对象 p 的**可达距离** (reachability-distance) : 使 p 从 q 密度可达的最小半径。根据密度可达的定义, q 必须是核心对象, 并且 p 必须在 q 的邻域内。所以, 可达距离为 $\max\{core_distance(q), dist(p, q)\}$ 。
- 对象 p 可能直接由多个核心对象可达, 使得 p 可能有多个可达距离。我们更关注于 p 的最小可达距离, 因为它给出了 p 连接到一个稠密簇的最短路径。



可达距离(p, r) = $core_distance(p)$

可达距离(p, q) = $dist(p, q)$

□ OPTICS示例

- 假设数据集中包含7个二维数据对象：

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

- 设置 $\varepsilon = Inf$, $n_{eps} = 2$

□ OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | 5.00 | 3.61 | 0 |

距离矩阵

核心距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

可达距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----|-----|-----|-----|-----|-----|-----|
| Inf | Inf | Inf | Inf | Inf | Inf | Inf |

OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | 5.00 | 3.61 | 0 |

距离矩阵

核心距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

p1的可达距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----|------|------|------|------|------|------|
| inf | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |

OPTICS示例

$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | | | |

距离矩阵

核心距离

| | | | | | | |
|------|------|------|------|------|------|------|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

排序后p1的可达距离

| | | | | | | |
|-----|------|------|------|------|------|------|
| 1 | 2 | 7 | 4 | 6 | 3 | 5 |
| Inf | 3.16 | 4.24 | 4.47 | 6.08 | 8.60 | 9.22 |

| Step | Center | OrderSeeds | Output |
|------|--------|---------------|--------|
| 1 | p1 | {2,7,4,6,3,5} | {p1} |

OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | 5.00 | 3.61 | 0 |

距离矩阵

核心距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

p2的可达距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----|-----------------|------|------|------|------|------|
| inf | 3.16 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |

OPTICS示例

$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | | | |

距离矩阵

核心距离

| | | | | | | |
|------|------|------|------|------|------|------|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

排序后p2的可达距离

| | | | | | | |
|-----|------|------|------|------|------|------|
| 1 | 2 | 4 | 7 | 6 | 3 | 5 |
| Inf | 3.16 | 1.41 | 2.00 | 5.39 | 6.32 | 6.71 |

| Step | Center | OrderSeeds | Output |
|------|--------|---------------|---------|
| 1 | p1 | {2,7,4,6,3,5} | {p1} |
| 2 | p2 | {4,7,6,3,5} | {p1,p2} |

□ OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | 5.00 | 3.61 | 0 |

距离矩阵

核心距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

p4的可达距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----|-----------------|------|-----------------|------|------|------|
| Inf | 3.16 | 5.10 | 1.41 | 5.39 | 5.00 | 1.41 |

OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | | | |
| 4.24 | 2.00 | 4.47 | 1.41 | | | |

距离矩阵

核心距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

排序后p4的可达距离

| 1 | 2 | 4 | 7 | 6 | 3 | 5 |
|-----|-----------------|-----------------|------|------|------|------|
| Inf | 3.16 | 1.41 | 1.41 | 5.00 | 5.10 | 5.39 |

| Step | Center | OrderSeeds | Output |
|------|--------|---------------|------------|
| 1 | p1 | {2,7,4,6,3,5} | {p1} |
| 2 | P2 | {4,7,6,3,5} | {p1,p2} |
| 3 | p4 | {7,6,3,5} | {p1,p2,p4} |

□ OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | 5.00 | 3.61 | 0 |

距离矩阵

核心距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

p7的可达距离

| 1 | 2 | 4 | 7 | 6 | 3 | 5 |
|-----|-----------------|-----------------|-----------------|------|------|------|
| Inf | 3.16 | 1.41 | 1.41 | 3.61 | 4.47 | 5.00 |

OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | | | |
| 4.24 | 2.00 | 4.47 | 1.41 | | | |

距离矩阵

核心距离

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|
| 3.16 | 1.41 | 1.00 | 1.41 | 1.00 | 3.61 | 1.41 |

排序后p7的可达距离

| 1 | 2 | 4 | 7 | 6 | 3 | 5 |
|-----|-----------------|-----------------|-----------------|------|------|------|
| Inf | 3.16 | 1.41 | 1.41 | 3.61 | 4.47 | 5.00 |

| Step | Center | OrderSeeds | Output |
|------|--------|---------------|---------------|
| 1 | p1 | {2,7,4,6,3,5} | {p1} |
| 2 | p2 | {4,7,6,3,5} | {p1,p2} |
| 3 | p4 | {7,6,3,5} | {p1,p2,p4} |
| 4 | p7 | {6,3,5} | {p1,p2,p4,p7} |

□ OPTICS示例

$$D = \{(1,2), (2,5), (8,7), (3,6), (8,8), (7,3), (4,5)\}$$

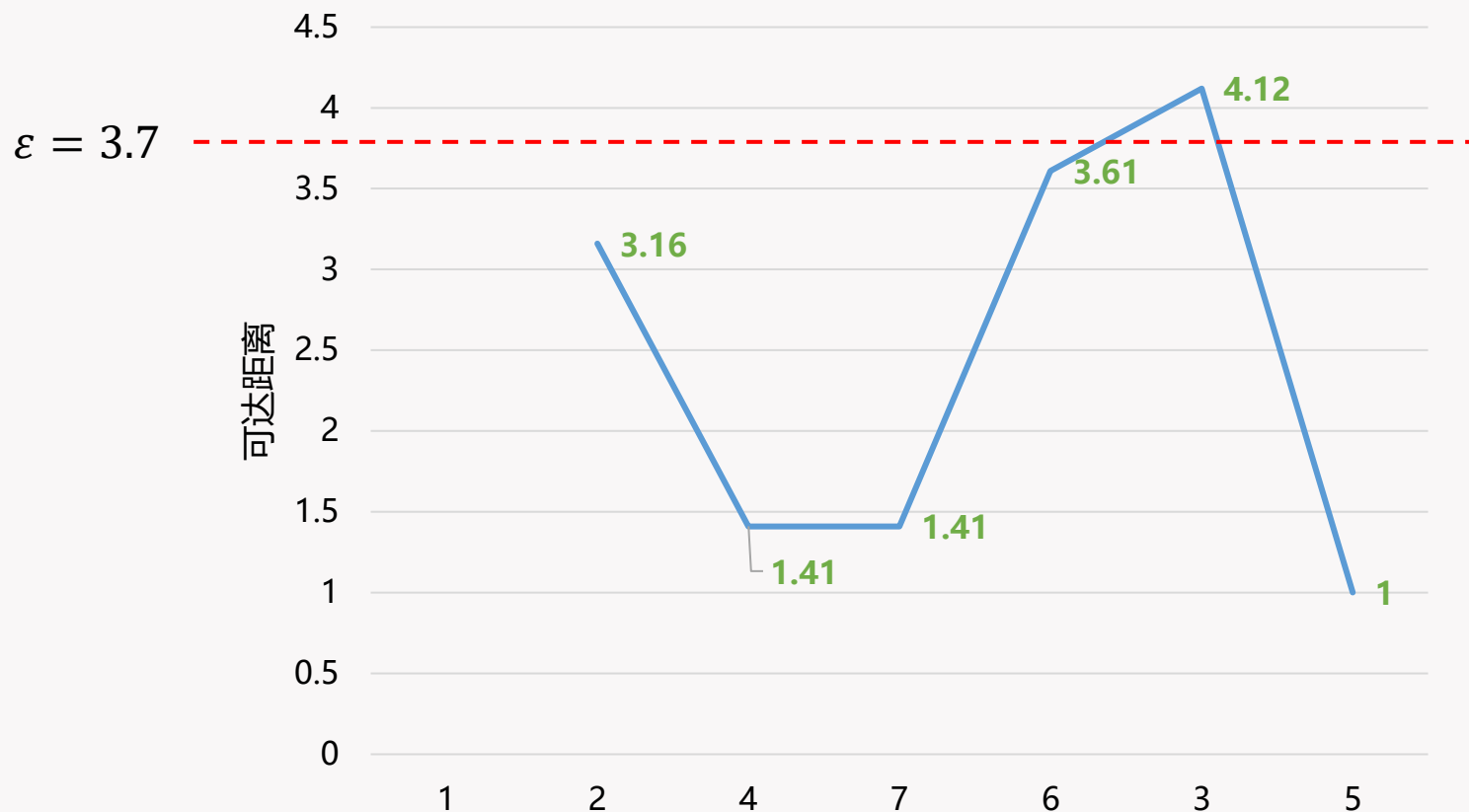
| | | | | | | |
|------|------|------|------|------|------|------|
| 0 | 3.16 | 8.60 | 4.47 | 9.22 | 6.08 | 4.24 |
| 3.16 | 0 | 6.32 | 1.41 | 6.71 | 5.39 | 2.00 |
| 8.60 | 6.32 | 0 | 5.10 | 1.00 | 4.12 | 4.47 |
| 4.47 | 1.41 | 5.10 | 0 | 5.39 | 5.00 | 1.41 |
| 9.22 | 6.71 | 1.00 | 5.29 | 0 | 5.10 | 5.00 |
| 6.08 | 5.39 | 4.12 | 5.00 | 5.10 | 0 | 3.61 |
| 4.24 | 2.00 | 4.47 | 1.41 | 5.00 | 3.61 | 0 |

距离矩阵

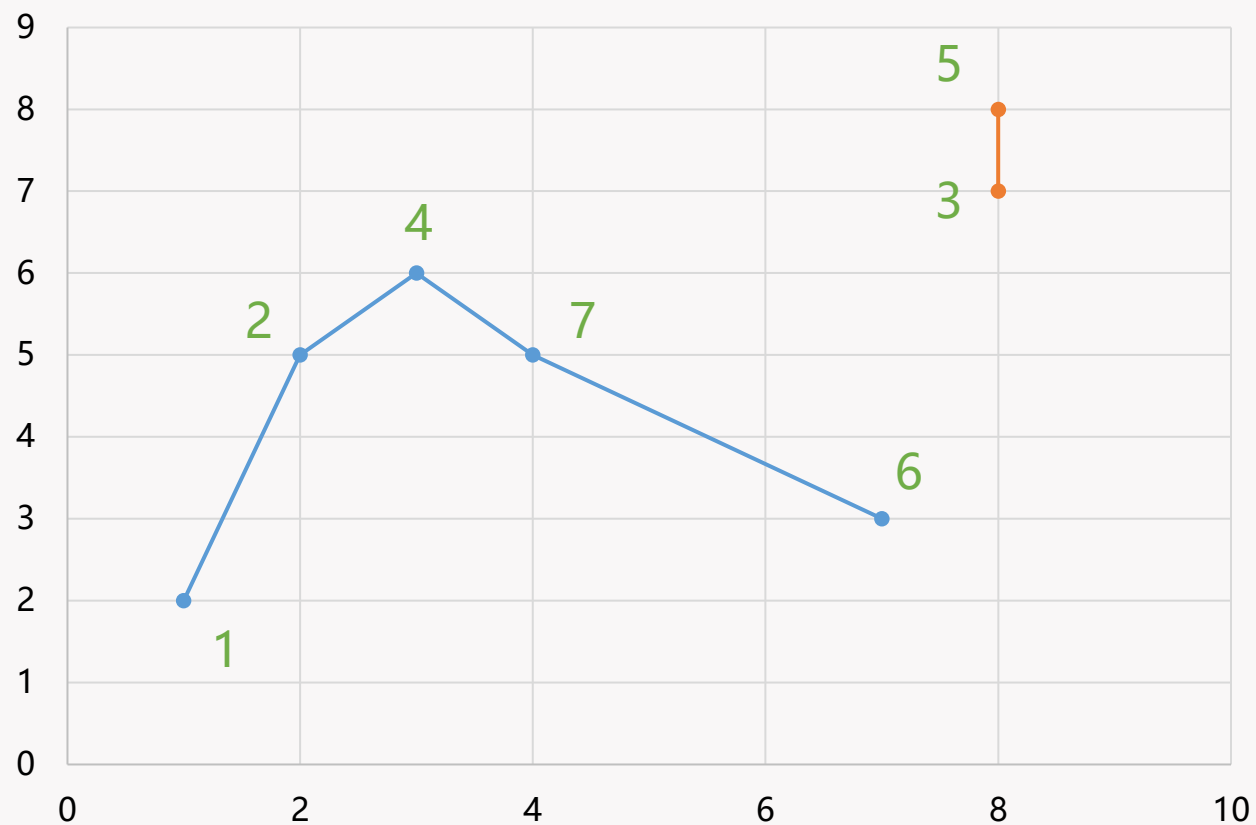
可达距离

| 1 | 2 | 4 | 7 | 6 | 3 | 5 |
|-----|------|------|------|------|------|------|
| Inf | 3.16 | 1.41 | 1.41 | 3.61 | 4.12 | 1.00 |

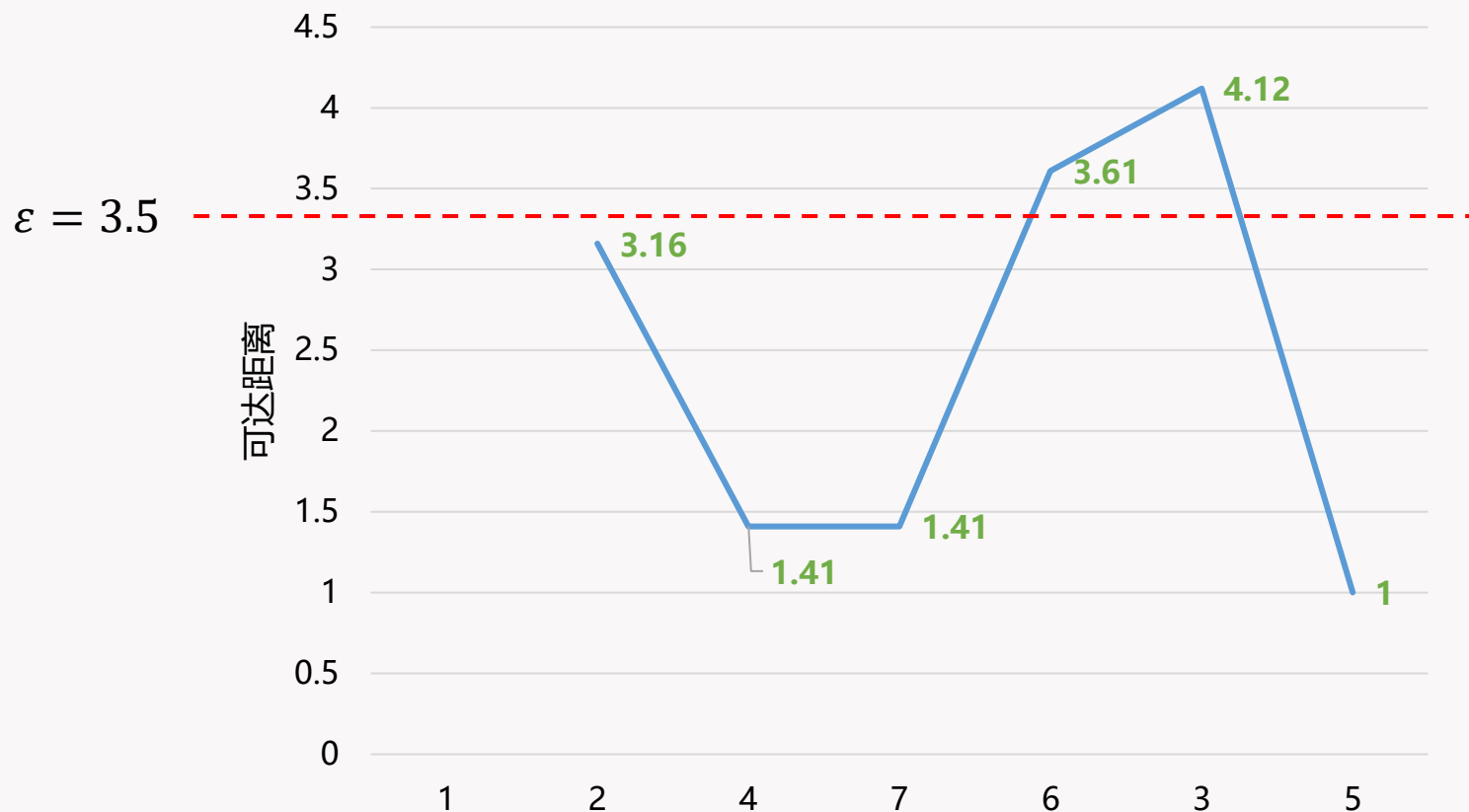
□ OPTICS示例



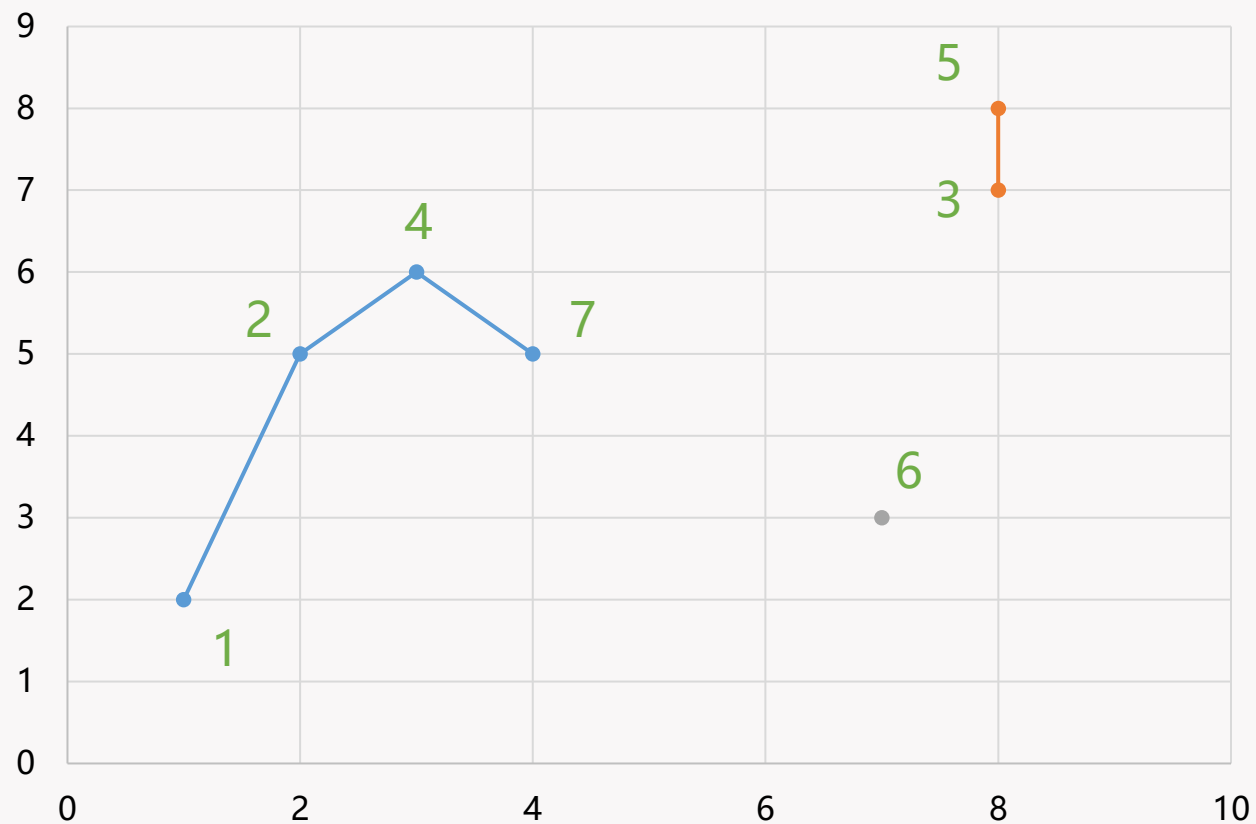
□ OPTICS示例



□ OPTICS示例



□ OPTICS示例



□ OPTICS

- 假设用户设置了 ε ，从结果序列按顺序取出数据对象，依次进行以下判断直到序列为空：
 - 如果该点的可达距离 $\leq \varepsilon$ ，则该点属于当前簇；
 - 如果该点的可达距离 $> \varepsilon$ ，则进行以下判断：若该点的核心距离 $> \varepsilon$ ，则该点为离群点；若该点的核心距离 $< \varepsilon$ ，则该点为新的簇。

- OPTICS的空间复杂度和时间复杂度
- OPTICS算法的结构和DBSCAN非常相似，因此具有相同的时间复杂度和空间复杂度。

□ OPTICS的特点

■ 优点:

- 相较于DBSCAN, OPTICS对 ε 的敏感度大大降低;
- OPTICS可以同时发现不同密度的簇结构。

■ 缺点:

- 可伸缩性较差。

□ OPTICS的算法流程

- 从数据集中选择任意的数据对象作为当前对象 p ;
- 检索 p 的 ε -邻域, 确定核心距离并设置可达距离为未定义, 输出当前对象 p ;
- 如果 p 不是核心对象, 从OrderSeeds表选择下一个对象, 如果OrderSeeds表为空, 则从数据集中任选一个未被处理的对象;
- 如果 p 是核心对象, 则对于 p 的 ε -邻域中的每个对象 q , 更新从 p 到 q 的可达距离, 并且如果 q 未被处理, 则把 q 插入OrderSeeds表中。
- 重复以上过程, 直数据集中所有数据对象都被处理且OrderSeeds表为空。



Chapter 5.5

基于网格的聚类方法

□ 基于网格的聚类方法

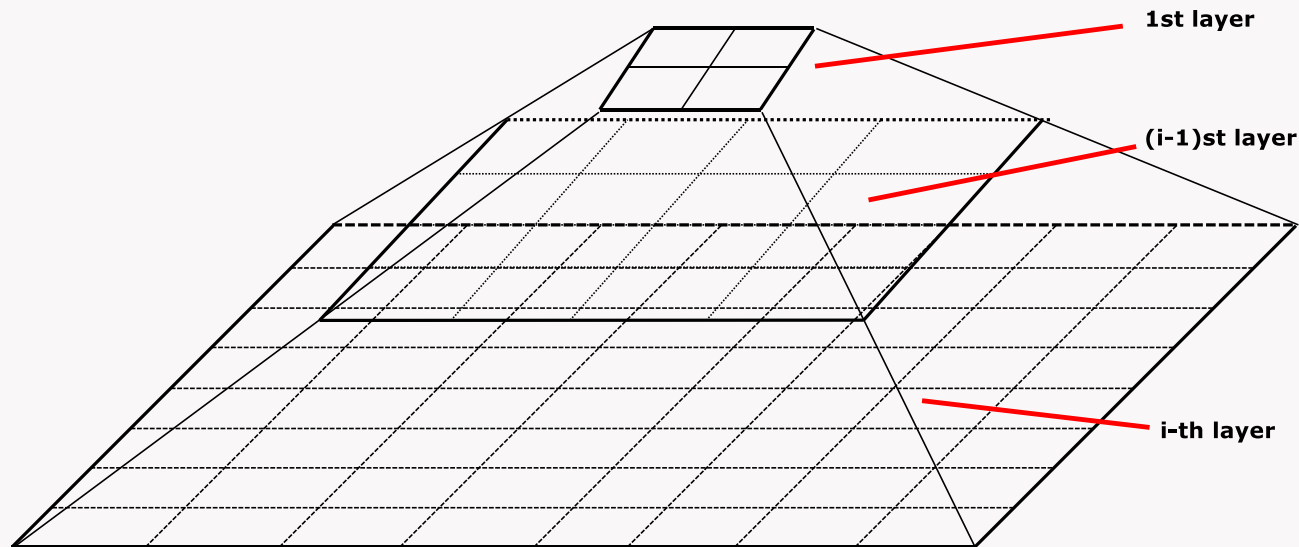
- 基于划分、层次、密度的聚类方法都是**数据驱动的**——划分数据集并且自动适应空间中数据的分布。而基于网格的聚类方法则是**空间驱动的**，把空间划分成独立于数据的单元。

□ 基于网格的聚类方法

- 基于网格的聚类方法使用一种多分辨率的网格数据结构，将空间量化成有限数目的单元（Cell），这些单元形成了网格结构，所有聚类操作都在该结构上进行。

▣ STING

- **STING** (STatistical INformation Grid) 是一种基于网格的多分辨率的聚类方法, 它将输入对象的空间区域划分为矩形单元。
- 空间使用分层和递归的方法进行划分并形成层次结构, 每层的矩形单元对应了不同级别的分辨率, 每个单元的属性的统计信息称作**统计参数**, 用于查询和数据分析。



▣ STING

■ 网格中常用的参数:

- Count: 网格中对象数目;
- Mean: 网格中所有值的平均值;
- Stdev: 网格中属性值的标准偏差;
- Min: 网格中属性值的最小值;
- max : 网格中属性值的最大值;
- Distribution: 网格中属性值符合的分布类型, 如正态分布, 均匀分布, 未知分布。

▣ STING

■ STING聚类的过程:

- 选定一层作为查询处理的开始点;
 - 对当前层次的每个单元计算与给定查询的相关程度的置信度, 标记为相关或不相关的单元;
 - 在更低的层次上重复步骤 2 直至达到最底层, 但只处理相关的单元;
 - 到达底层后如果满足有满足查询条件的单元格, 则由邻近的单元形成簇。如果没有满足查询条件的单元格, 则重新考虑不相关的单元。
- 如果最底层的网格数量非常多, 则STING的效果趋向于DBSCAN的聚类结果。因此, STING也可以看作基于密度的聚类方法。

▣ STING

■ 优点

- 网格结构有利于并行处理和增量更新;
- 效率高, 仅需扫描一次数据集来计算单元的统计信息, 时间复杂度为 $O(n)$ 。建立层次结构后查询时间为 $O(g)$, 其中 g 是最底层网格单元的数目。

■ 缺点

- 聚类质量取决于网格结构最底层的粒度;
- 不太适用于高维数据。



Chapter 5.6

聚类效果评估

□ 簇评估

- 对于模型结果的评估在数据挖掘中是必不可少且非常重要的。对于分类任务，存在一些客观的指标被广泛的使用和接受，比如准确率、召回率等。但是由于聚类问题的特点，不存在类似准确率的客观指标来评价聚类结果的好坏。
- 尽管如此，簇评估，或**簇确认**（Cluster validation）也是非常有必要的，目的是确认数据中是否存在非随机结构，或者说确认数据集中是否有自然的簇结构。

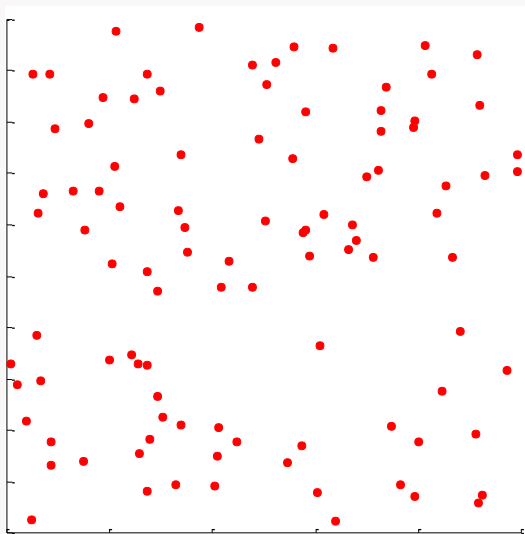
□ 簇评估

■ 簇评估主要确认以下几个重要的问题：

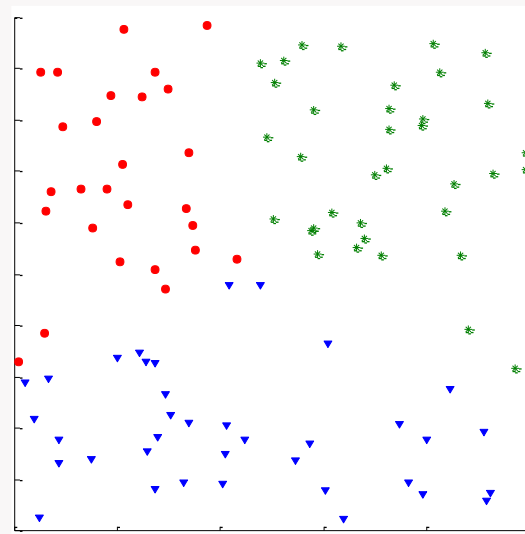
- 确认数据集的**聚类趋势**（Clustering tendency），即识别数据中是否实际存在非随机结构；
- 确认正确的簇个数；
- 测定聚类的质量。

□ 聚类趋势

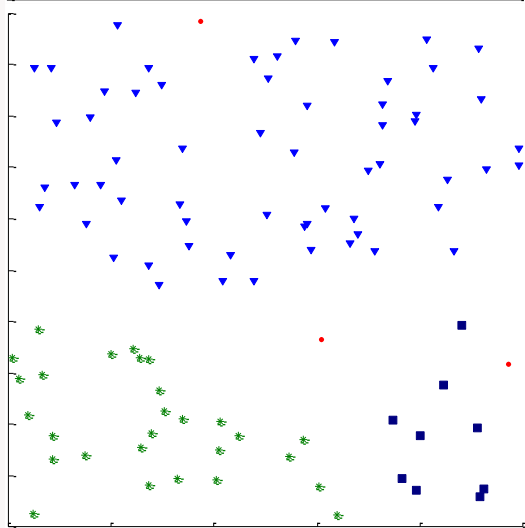
随机生成的数据



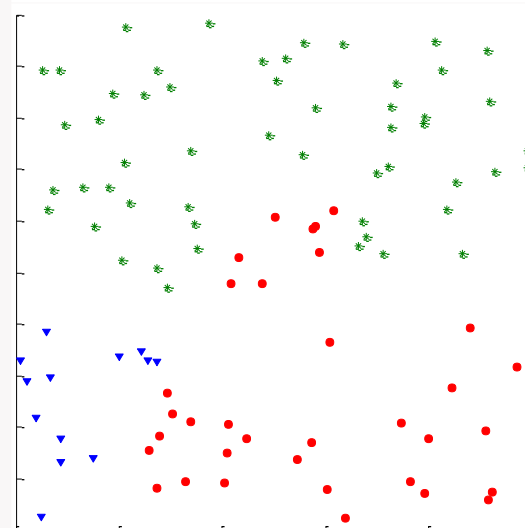
DBSCAN的结果



Kmeans的结果



全链的结果



□ 聚类趋势

- 通过计算数据集被均匀分布产生的概率，可以评估数据集的聚类趋势。比如利用空间随机性的统计检验—霍普金斯统计量进行描述。

□ 霍普金斯统计量

- 均匀地从空间中抽取 n 个点 p_1, p_2, \dots, p_n , 对于每个点 p_i , 找出 p_i 在数据集中的最近邻:

$$x_i = \min_{v \in D} \text{dist}(p_i, v)$$

- 均匀地从数据集中抽取 n 个点 q_1, q_2, \dots, q_n , 对于每个点 q_i , 找出 q_i 在 $D - \{q_i\}$ 中的最近邻:

$$y_i = \min_{v \in D, v \neq q_i} \text{dist}(q_i, v)$$

- 计算霍普金斯统计量 H

$$H = \frac{\sum y_i}{\sum y_i + \sum x_i}$$

如果数据集是均匀分布的, 则 H 趋向于0.5, 如果是高度倾斜的, 则 H 趋向于 0。

□ 确定簇数

- 确定数据集中“正确的”簇数是重要的，合适的簇数可以看作在聚类分析中寻找**可压缩性**和**准确性**之间好的平衡点。
- 确定簇数并非易事，即依赖于数据集分布的形状和尺度，也依赖于用户的需求。常见的方法有以下两种：
 - 经验法：对于 n 个点的数据集，设置簇数 k 为 $\sqrt{n/2}$ 。在期望的情况下，每个簇大约 $\sqrt{2n}$ 个数据对象；
 - 肘方法：增加簇数有助于降低每个簇的簇内方差之和，所以寻找簇内方差和簇数的曲线的拐点。

□ 测定聚类质量

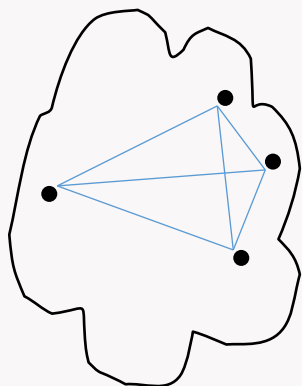
- 非监督的簇评估指标一般称作**内部指标** (Internal index) , 因为它们仅使用出现在数据集中的信息。簇的有效性的内部指标主要基于以下两个概念:
 - **凝聚度** (Cohesion) : 用来评估簇中数据对象之间的密切相关程度, 或者相似程度;
 - **分离度** (Separation) : 用来评估簇之间的差异性或者不相似性。

■ 基于图的凝聚度和分离度定义

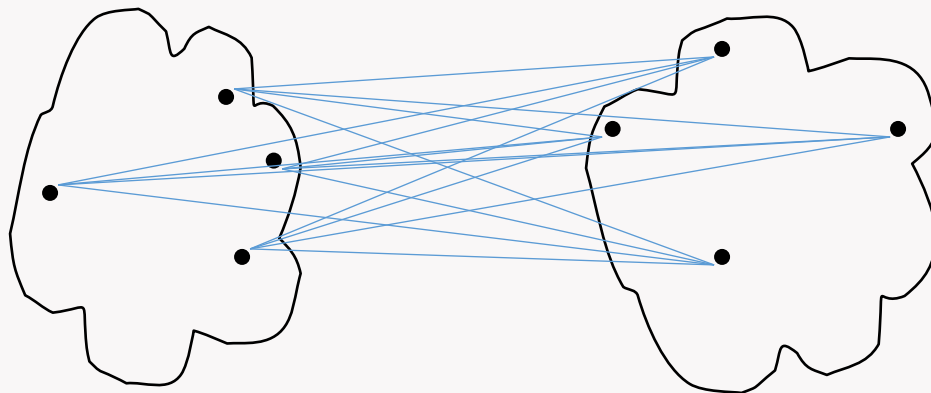
- 凝聚度表示为簇内数据对象之间的邻近度之和，分离度表示为不同簇的数据对象之间的邻近度之和：

$$cohesion(C_i) = \sum_{x,y \in C_i} dist(x,y)$$

$$separation(C_i, C_j) = \sum_{x \in C_i, y \in C_j} dist(x,y)$$



凝聚度



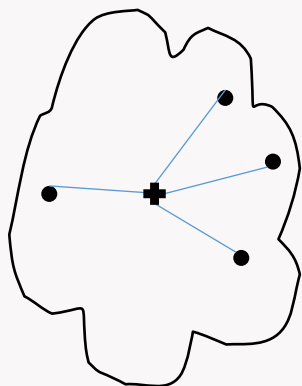
分离度

■ 基于原型的凝聚度和分离度定义

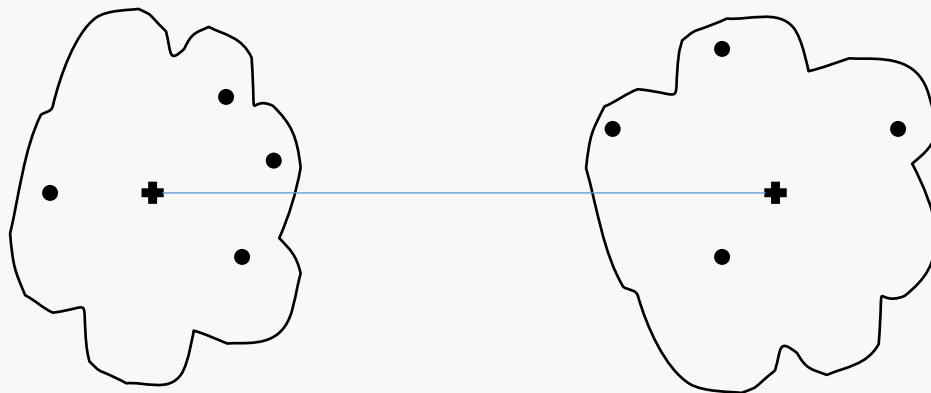
- 凝聚度表示为簇内数据对象与质心的邻近度之和，分离度表示为不同簇的质心之间的邻近度之和：

$$cohesion(C_i) = \sum_{x \in C_i} dist(x, c_i)$$

$$separation(C_i, C_j) = dist(c_i, c_j)$$



凝聚度



分离度

□ 非监督簇评估：轮廓系数

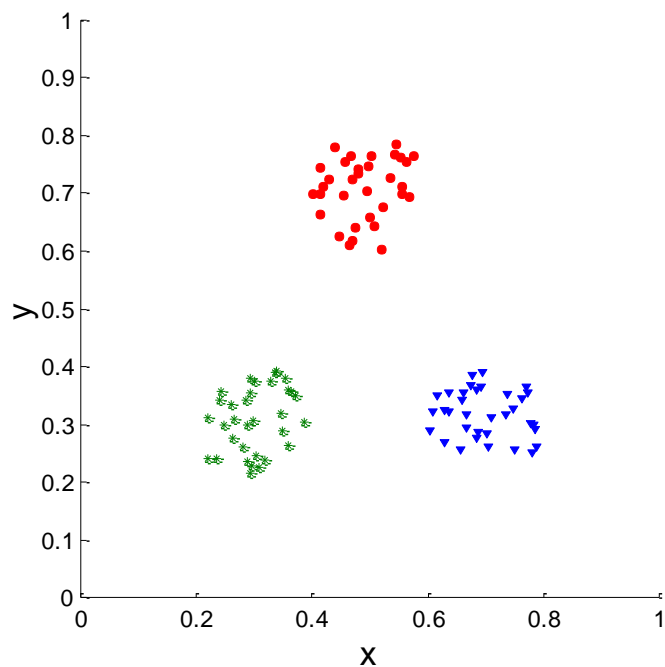
- **轮廓系数** (Silhouette coefficient) 是另一种流行的簇评估方法，可以对聚类结果做出更全面的评估。计算轮廓系数包含三个步骤：对于第 i 个数据对象，
 - 计算它到簇中所有其他数据对象的平均距离，记为 a_i ；
 - 分别计算它到其他各个簇中所有数据对象的平均距离，得到 $C-1$ 个值，取其中的最小值，记为 b_i ；
 - 它的轮廓系数表示为 $s_i = (b_i - a_i) / \max(a_i, b_i)$ 。
- 当所有数据对象的轮廓系数计算完以后，取簇中所有数据对象的轮廓系数的平均值作为该簇的平均轮廓系数。

□ 非监督簇评估：相似度矩阵

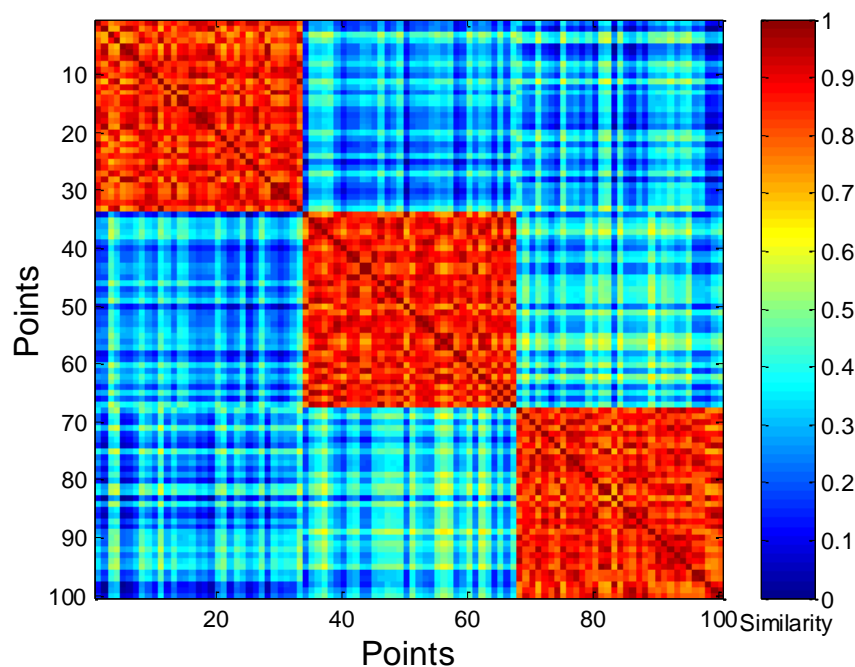
- 借助相似度矩阵可以直观评估簇的有效性，评估的方法就是相似度矩阵的可视化。主要做法是，将数据对象按照簇编号进行排序，然后计算数据对象之间的相似度矩阵并归一化。对于第 i 个数据对象和第 j 个数据对象：
 - 相似度越趋向于1，则表示越相似；
 - 相似度越趋向于0，则表示越不相似。
- 因此，理想的相似度矩阵应该是**块对角**（Block diagonal）结构，即簇内的数据对象是相似的，簇间的数据对象是不相似的。

□ 非监督簇评估：相似度矩阵

■ 相似度矩阵的示例



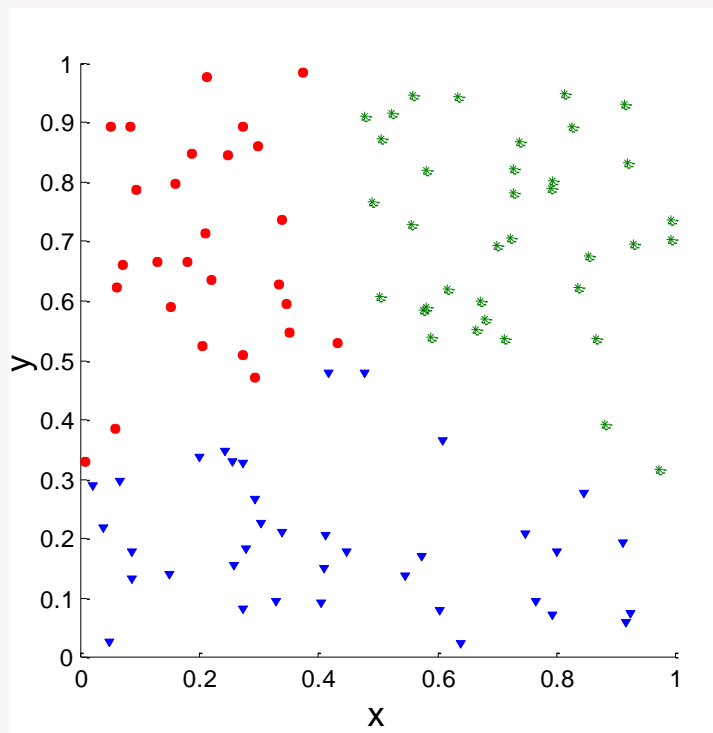
数据分布



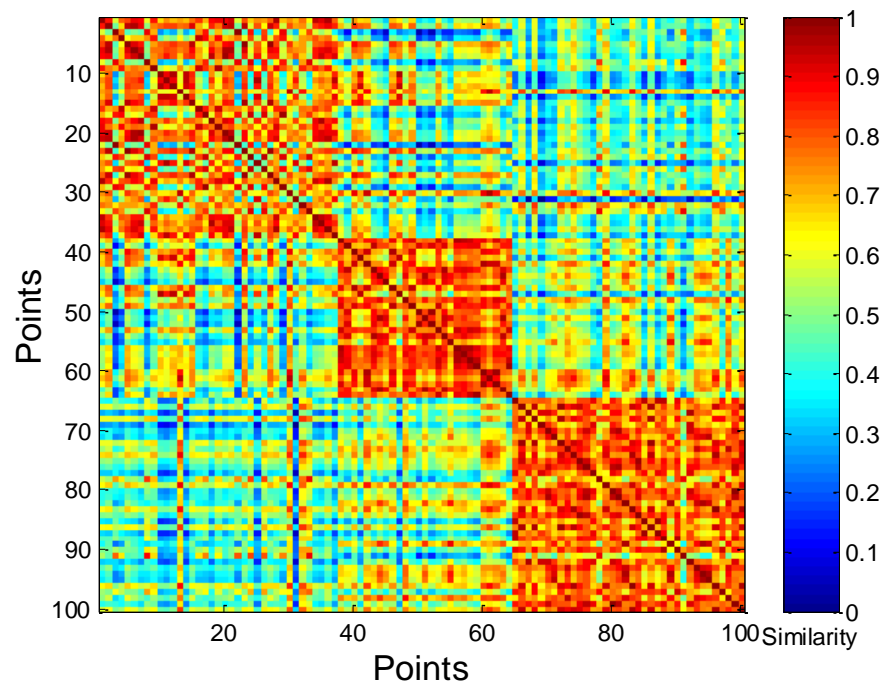
相似度矩阵可视化结果

□ 非监督簇评估：相似度矩阵

■ 相似度矩阵的示例



数据分布



相似度矩阵可视化结果



Chapter 5

本章小结

- 在数据挖掘中对聚类算法的要求
- 基于划分的聚类方法：K-means, K-means++, Bisecting K-means
- 基于划分的聚类方法的优缺点
- 基于层次的聚类方法：凝聚层次聚类
- 基于层次的聚类方法的优缺点
- 基于密度的聚类方法：DBSCAN, OPTICS
- 基于密度的聚类方法的优缺点
- 基于网格的聚类方法：STING
- 基于网格的聚类方法的优缺点
- 聚类效果评估：凝聚度、分离度、轮廓系数、相似度矩阵



第五章 完结

