数据挖掘导论
Introduction to Data Mining

第四章：聚类分析

王浩
Email：haowang@szu.edu.cn

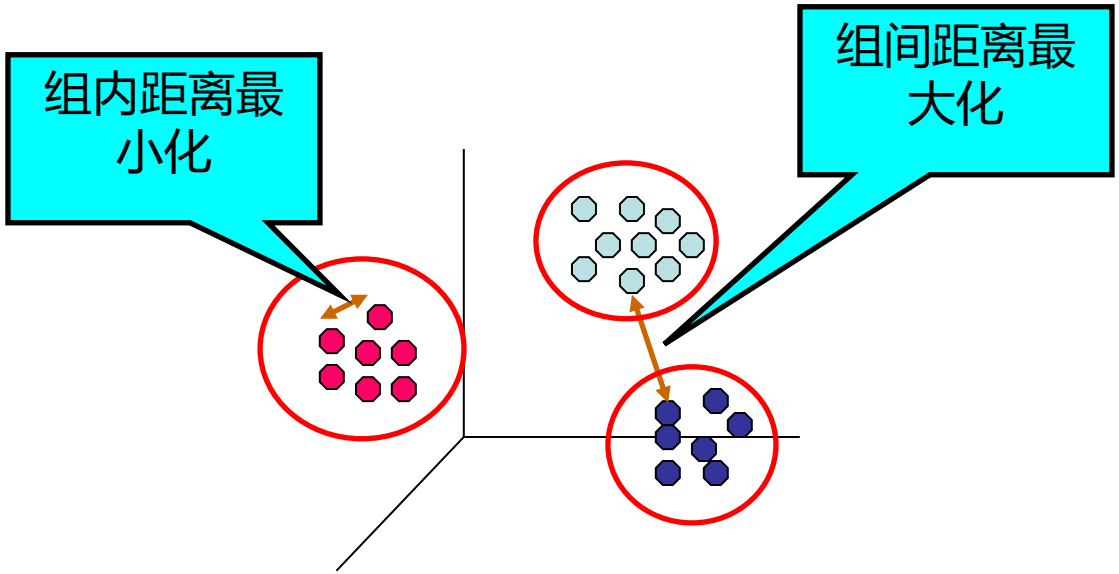
深圳大学 SHENZHEN UNIVERSITY
电子信息与工程学院

目标：

- 聚类分析的定义与应用
- 聚类的类型
- 簇的概念与类型
- 经典算法

什么是聚类分析

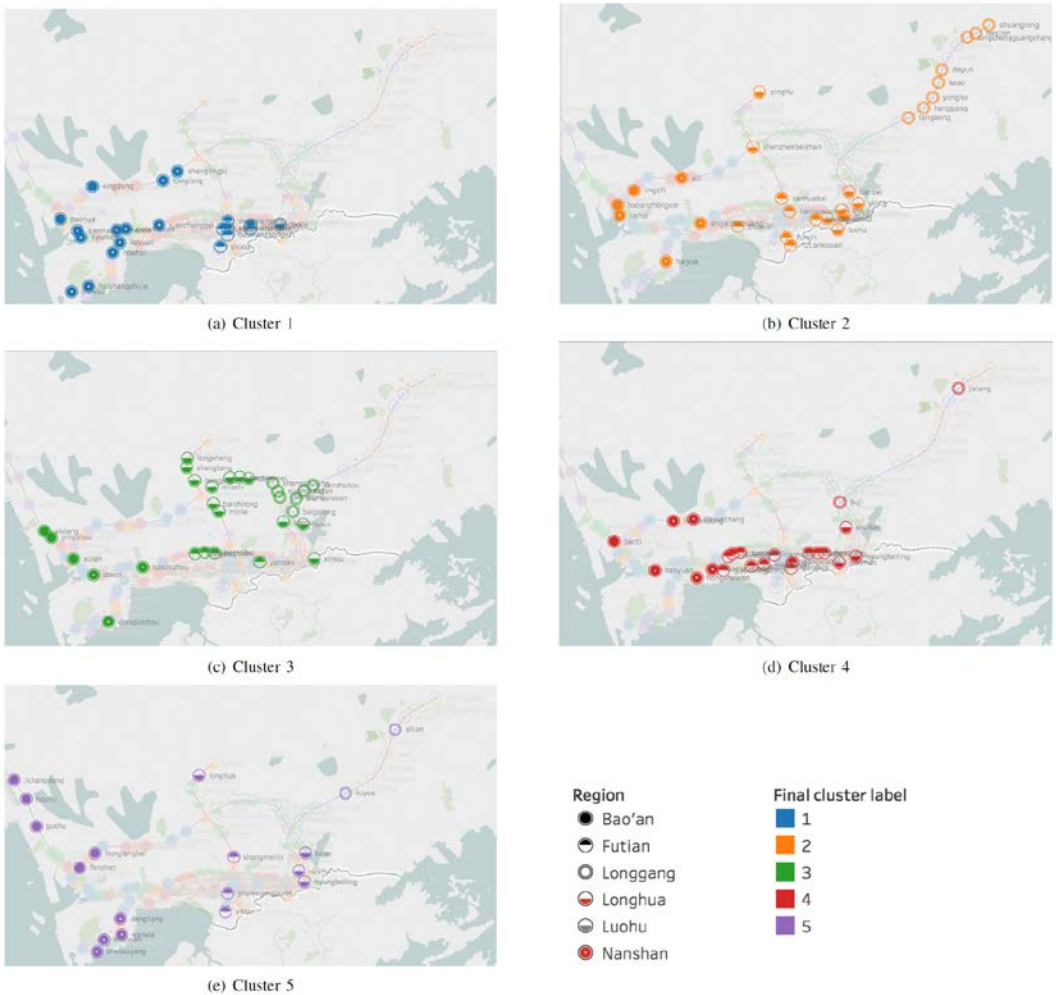
- 将数据分组，使得一个组（簇）内的对象彼此相似（或相关），而不同组中的对象不同（或不相关）。



聚类分析的应用

- **理解**
  - 将具有相似特征的地铁站聚类方便规划管理
  - 将具有相似出行特征的乘客聚类方便个性化服务
  - 将相关文档进行分组以便浏览
  - 将具有相似功能的基因和蛋白质分组
  - 将具有相似价格波动的股票分组

  ……
- **汇总**
  - 压缩大数据的规模



Source: Tang, L., Zhao, Y., Tsui, K.L., He, Y., Pan, L. A clustering refinement approach for revealing urban spatial structure from smart card data. Applied Sciences. Vol. 10, No.16, pp., 5606. 2020.

深圳地铁站的聚类

什么不是聚类分析

简单划分

- 按姓氏字母顺序将学生分成不同的注册组

查询结果

- 分组是外部规范的结果

有监督分类

- 具有类标签信息

关联分析

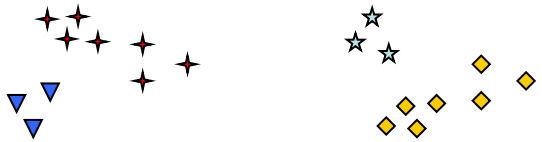
- 局部 vs. 全局连接

簇的概念是不精确的

有几类（簇）？

六个类（簇）

两个类（簇）



四个类（簇）

不同的聚类类型

聚类是整个簇的集合

聚类的差别：簇的集合是层次的还是划分的?

划分聚类

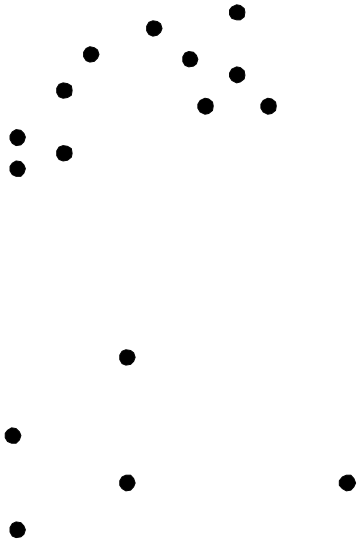
- 简单地将数据对象划分成不重叠的子集（簇），使得每个数据对象恰在一个子集中。

层次聚类

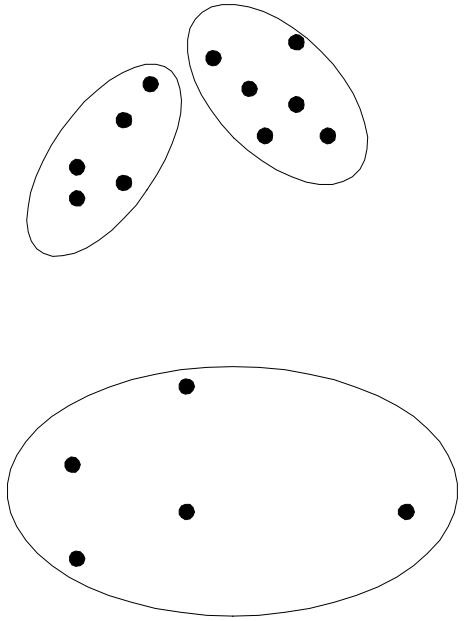
- 允许簇具有子簇，层次聚类是嵌套簇的集族*，成树状。

*集族：集合的集合

划分聚类

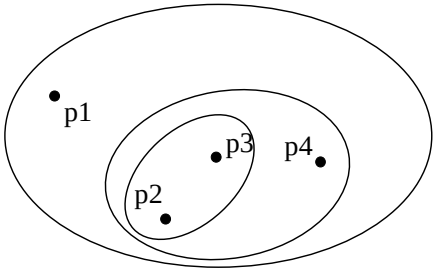


原始点

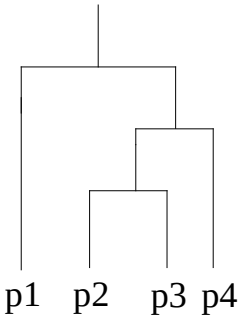


一种划分聚类

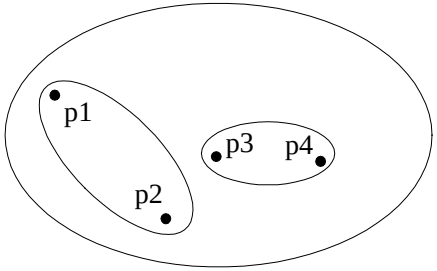
层次聚类



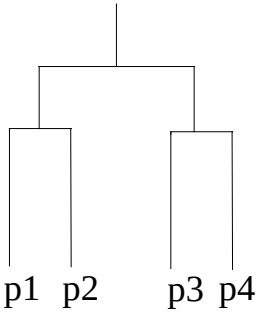
传统的层次聚类



传统树状图



非传统的层次聚类



非传统树状图

簇集合之间的其他区别

互斥的v.s.非互斥的

- 在非互斥的聚类中，点可能属于多个簇。
- 非互斥聚类中，对象可以表示多个类或“边界”点

模糊的v.s.非模糊的

- 在模糊聚类中，一个点以一个介于0和1之间的隶属权值属于每个簇
- 每个对象的权值总和必须为1
- 概率聚类具有相似的特征

完全的v.s.部分的

- 在某些情况下，我们只想对一些数据进行聚类

异质的v.s.同质的

- 大小、形状和密度大不相同的簇

簇的不同类型

明显分离的簇

基于中心的簇

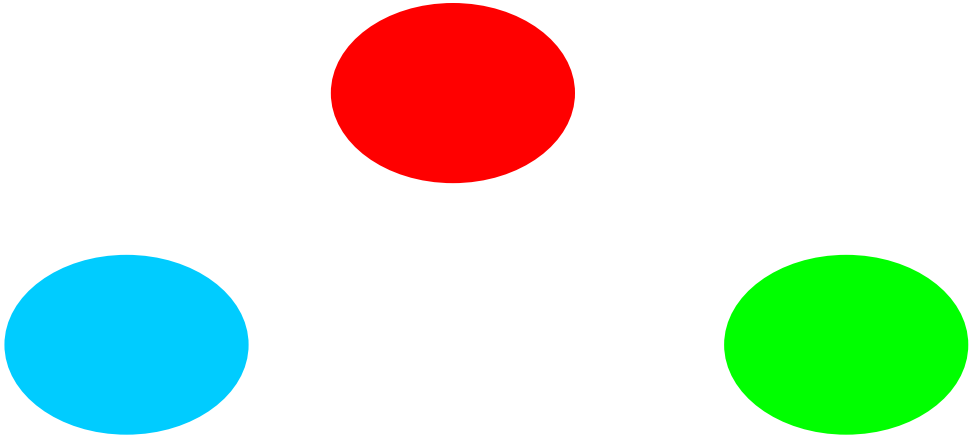
基于邻近的簇

基于密度的簇

概念簇

簇的不同类型：明显分离的

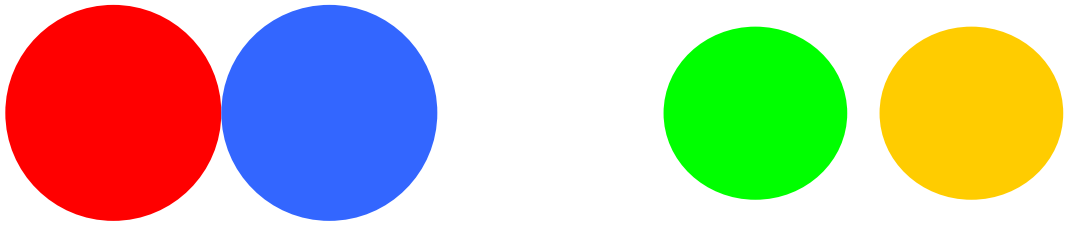
- 明显分离的簇:
  - 每个点到同簇中任意点的距离比到不同簇中任意点的距离更近。



3个明显分离的簇

簇的不同类型：基于中心的

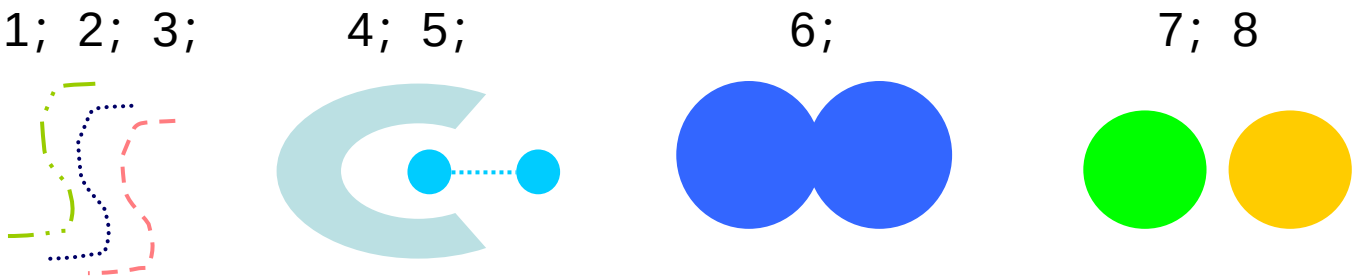
- 基于中心的簇
  - 每个点到该簇中心的距离比到任何其他簇中心的距离更近
  - 簇的中心通常是一个质心，即簇中所有点的平均值。当质心没有意义时（例如当数据具有标称属性时），簇中心是一个中心点（medoid），即簇中最具“代表性”的点。



4个基于中心的簇

簇的不同类型：基于邻近的

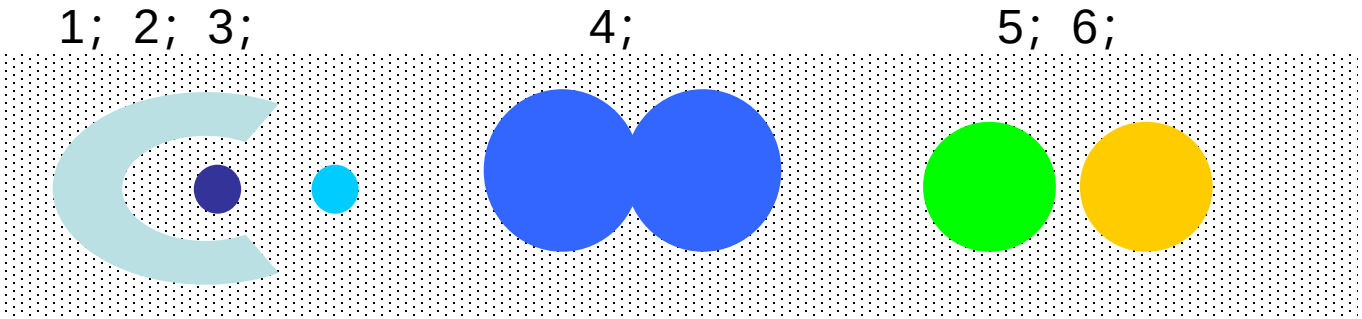
- 基于邻近的簇(最近邻)
  - 每个点到该簇中至少一个点的距离比到不同簇中任意点的距离更近。



8个基于邻近的簇

簇的不同类型：基于密度的

- 基于密度的簇：
  - 簇是点的稠密区域，由低密度区域与其他高密度区域分开。
  - 当簇不规则或者缠绕，并且有噪声和离群点时，常常使用基于密度的簇定义。

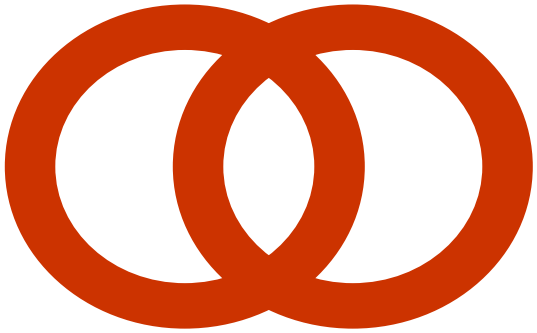


6 个基于密度的簇

簇的不同类型：概念簇

概念簇

- 寻找共享某些公同性质或表示特定概念的簇。
- 这个定义包含前面所有簇的定义。



2个相交的环

簇的不同类型：目标函数

用目标函数定义的簇

- 寻找使目标函数最小化或最大化的簇。
- 列举所有可能的聚类方法，并用给定的目标函数来评估每个潜在簇集合的“优度”. (NP 难问题)
- 可以有全局或局部目标：
  - 层次聚类算法通常具有局部目标。
  - 划分聚类算法通常具有全局目标。
- 全局目标函数方法的一种变体是参数模型。
  - 假设数据是若干统计分布的“混合物”；
  - 模型参数由数据确定。

将聚类问题映射到一个不同的问题

将聚类问题映射到另一个领域并解决该领域中的相关问题

- 邻接矩阵定义了一个加权图，其中节点是被聚类的点，加权边表示点之间的邻近度。
- 聚类相当于将图分解为连通分支，每个分支对应一个簇。
- 或使簇间的边权重最小，使簇内的边权重最大。

输入数据的特征很重要

邻近度或密度度量的簇

- 取决于数据和应用场景

影响邻近度或密度的数据特征如下：

- 维度
  - 稀疏性
- 属性类型
- 数据中的特殊关系
  - 例如，相关性
- 数据的分布

噪声和异常值会干扰聚类算法的运行

聚类算法

K 均值及其变种

层次聚类

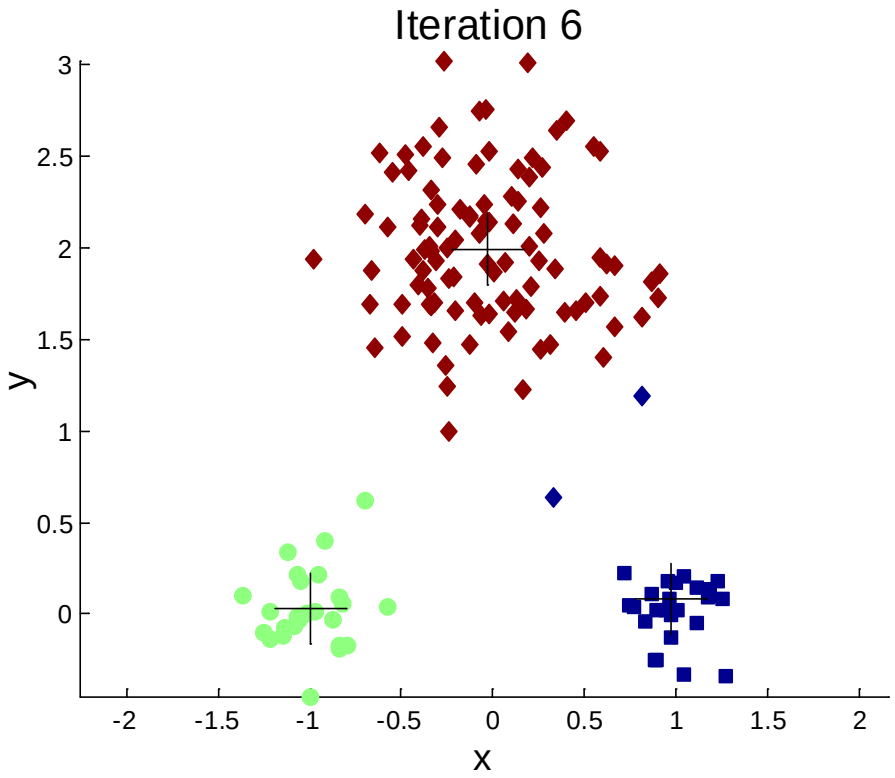
基于密度的聚类

K均值聚类

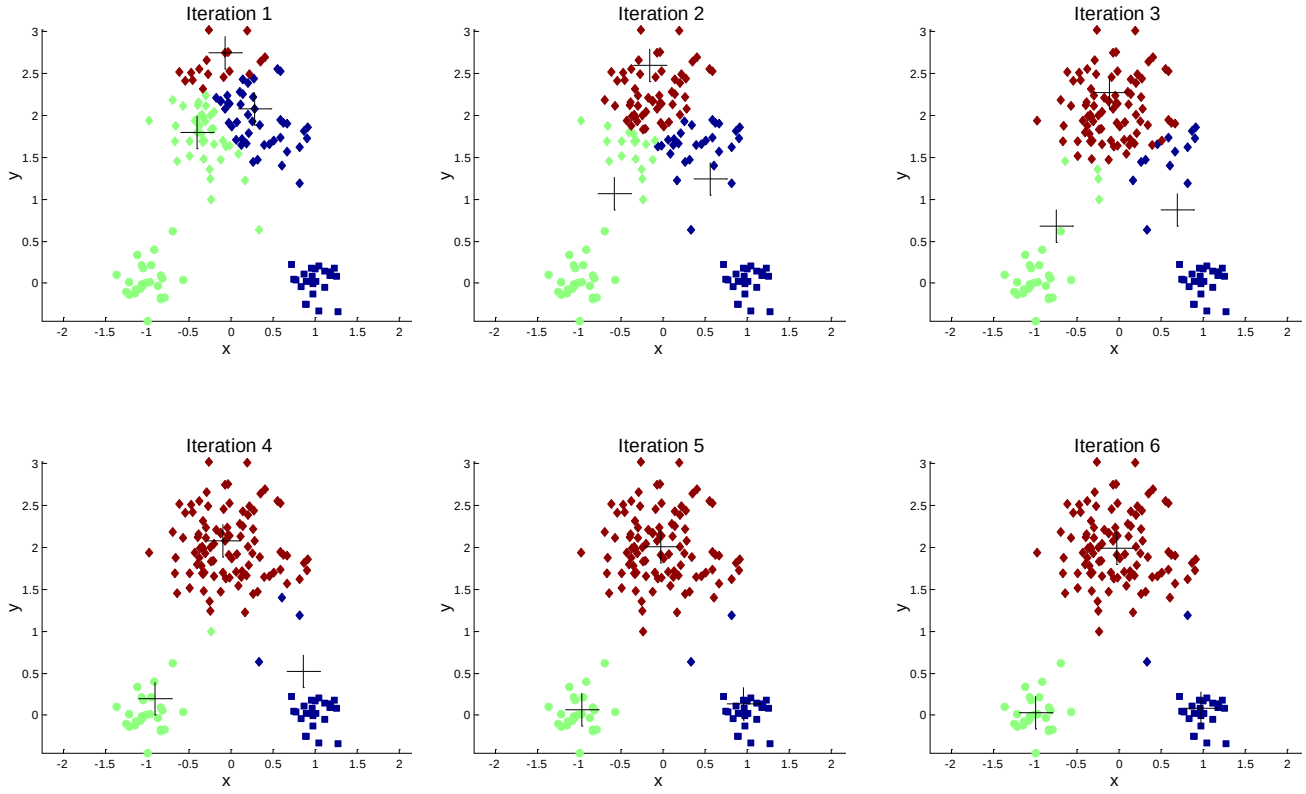
- ✓ K均值属于划分聚类法
- ✓ 必须提前指定簇的数量K
- ✓ 每个簇都与一个质心（中心点）相关联
- ✓ 将每个点指定给具有最近质心的簇
- ✓ 基本算法非常简单，一般步骤如下：

1：选择K个点作为初始质心。
2：Repeat
3：　　计算邻近度，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

K均值聚类的例子



K均值聚类的例子


Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
x
y

K均值聚类-细节

初始质心通常是随机选择的。

- 产生的簇会随着算法运行而变化。

质心通常是簇中的点的平均值。

“邻近度”是通过欧氏距离、余弦相似性、相关性等来衡量的。

对于以上常用的相似性度量，K均值都收敛。

大多数收敛发生在最开始的几次迭代中。

- 通常使用时将原算法中的停止条件更改为较弱的条件“直到相对较少的点改变簇”。

时间复杂度是$O(n \times K \times I \times d)$

- n = 点数, K = 簇的数量, I = 收敛所需的迭代次数, d = 属性数

评估聚类质量

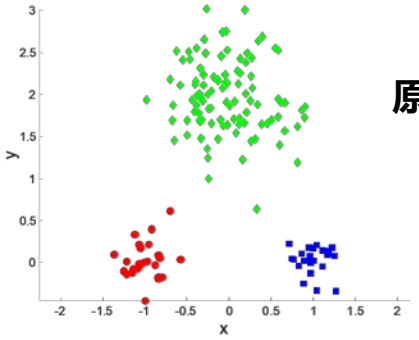
最常用的是误差平方和(Sum of Squared Error, SSE)

- 对于每个点,误差是到最近簇的距离
- 为了得到SSE，我们将这些误差平方并求和。

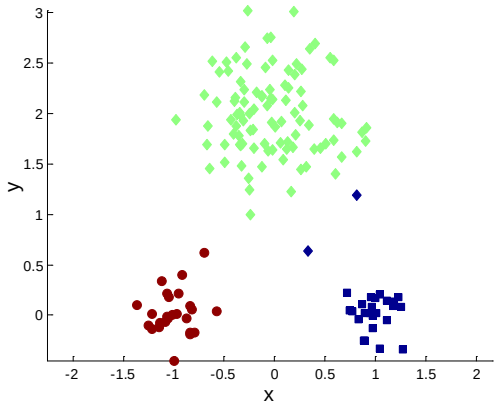
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

- x是簇C_i中的数据点， m_i是簇C_i的优化目标点。
  - 可以证明m_i对应于簇的中心（平均值）（使簇的SSE最小的质心是均值）
- 给定两组聚类，我们更喜欢误差最小的一组
- 减少SSE的一个简单方法是增加K，即簇的数量
  - 一般认为，SSE相同时K值较高的聚类比K值较小聚类差。

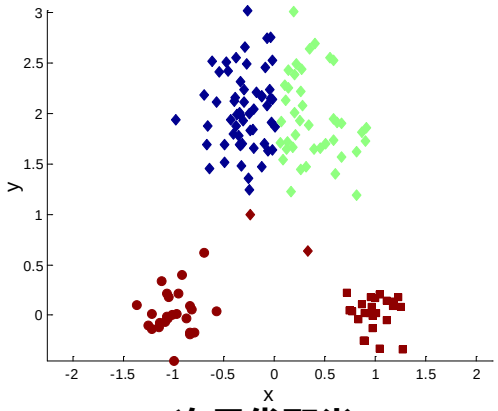
两种不同的K均值聚类



原始点



最优的聚类



次最优聚类

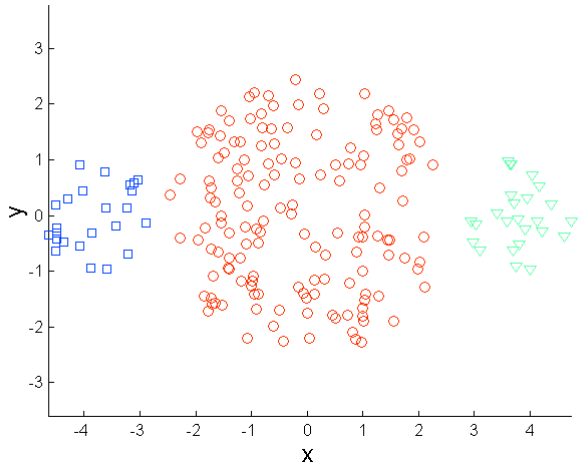
K均值的缺点

当簇的以下性质不同时，K均值就会存在问题

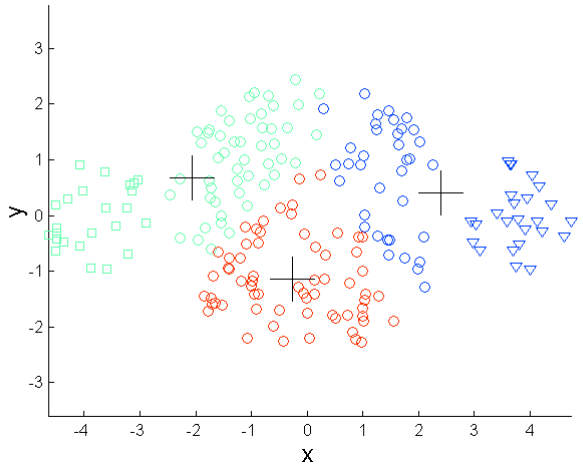
- 尺寸
- 密度
- 非球形形状（球形分布：方差一致的多变量高斯分布）

当数据包含离群点时K均值会存在问题。

K均值的缺点：尺寸不等

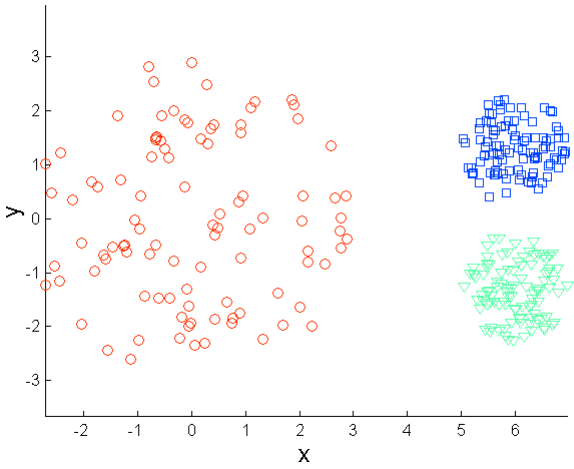


初始点

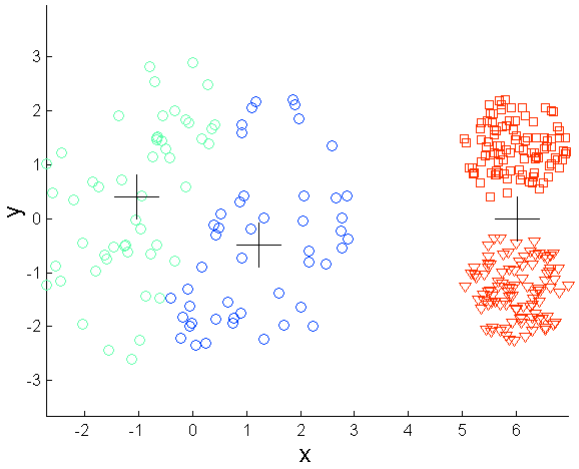


K均值 (3 个簇)

K均值的缺点：密度不等

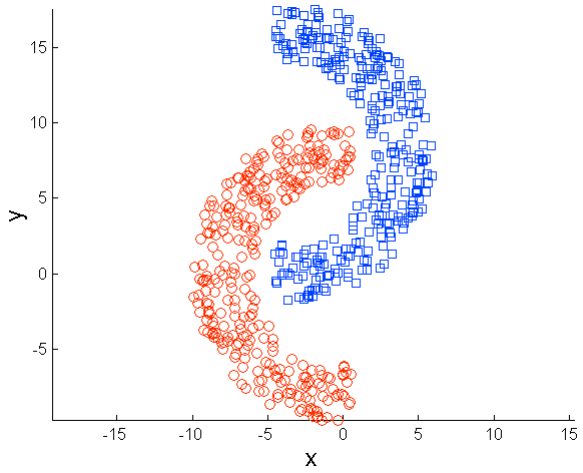


初始点

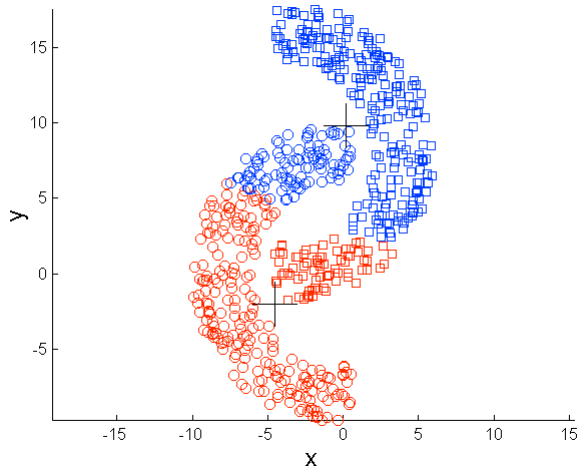


K均值(3个簇)

K均值的缺点：非球形形状

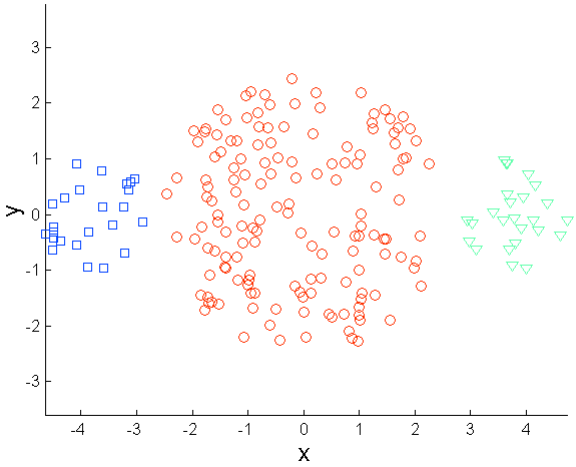


初始点

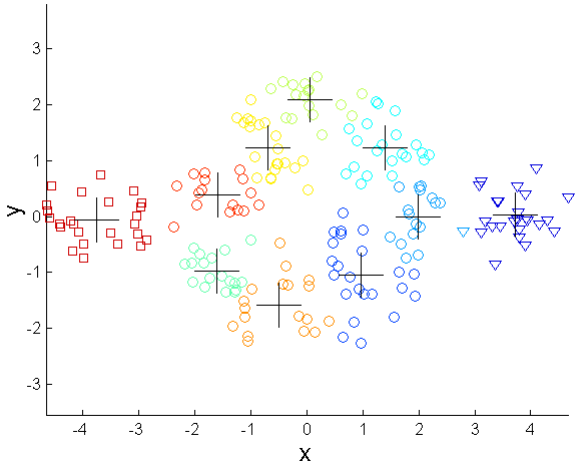


K均值 (2 个簇)

克服K均值的缺点



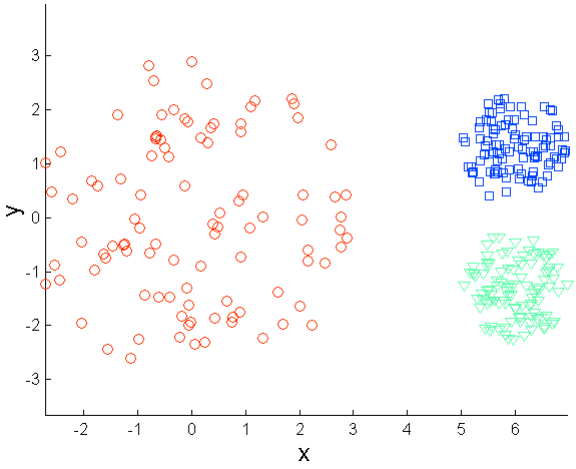
初始点



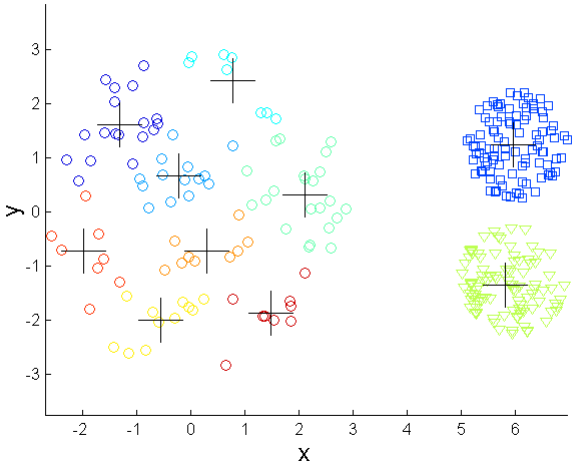
K均值簇

一种解决方案是使用多个簇，但需要将它们合并。

克服K均值的缺点

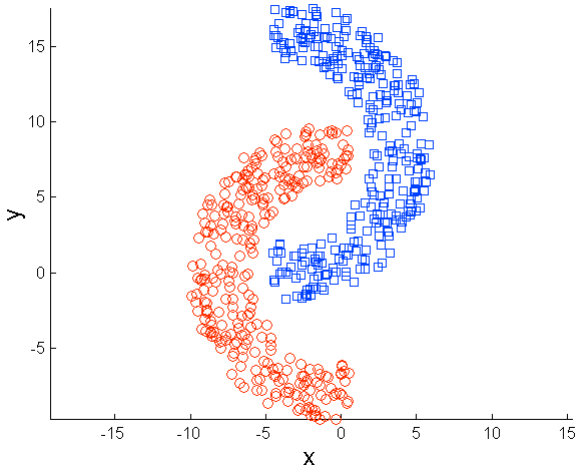


初始点

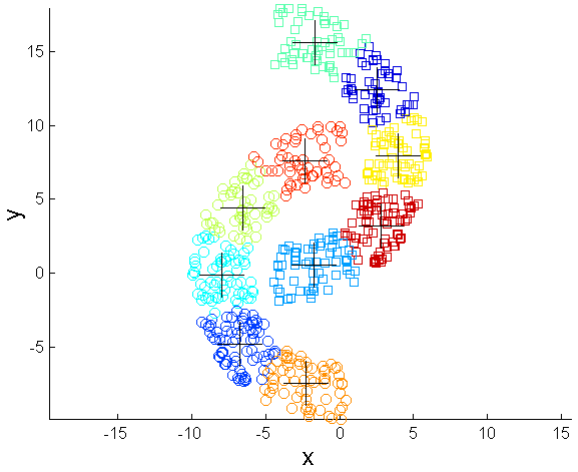


K均值簇

克服K均值的缺点

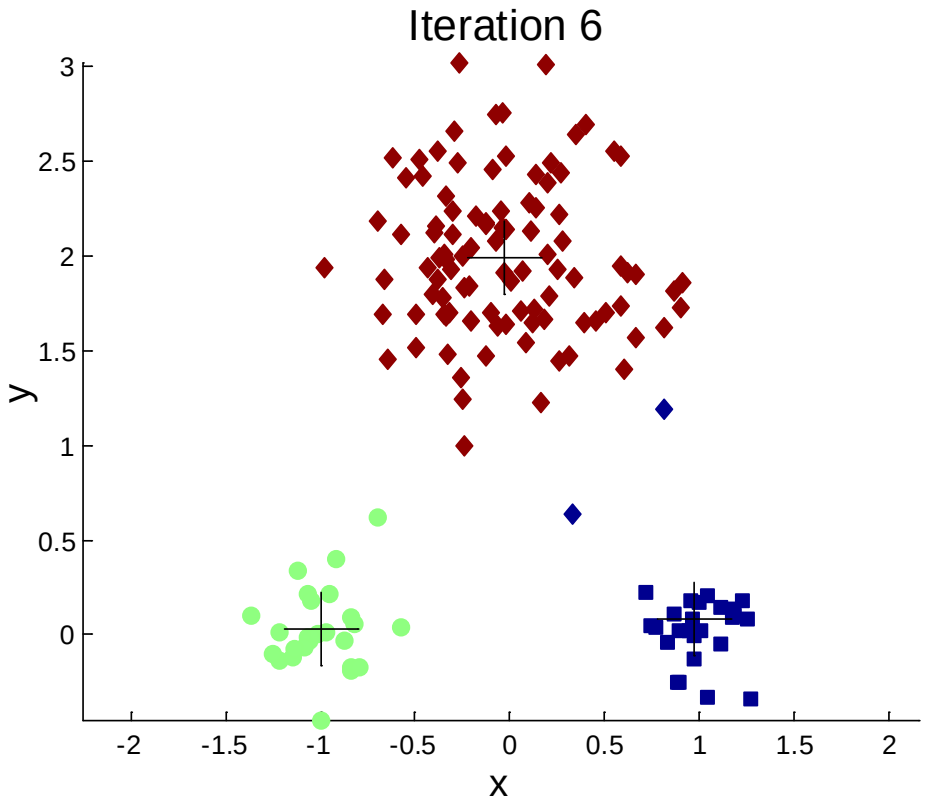


初始点

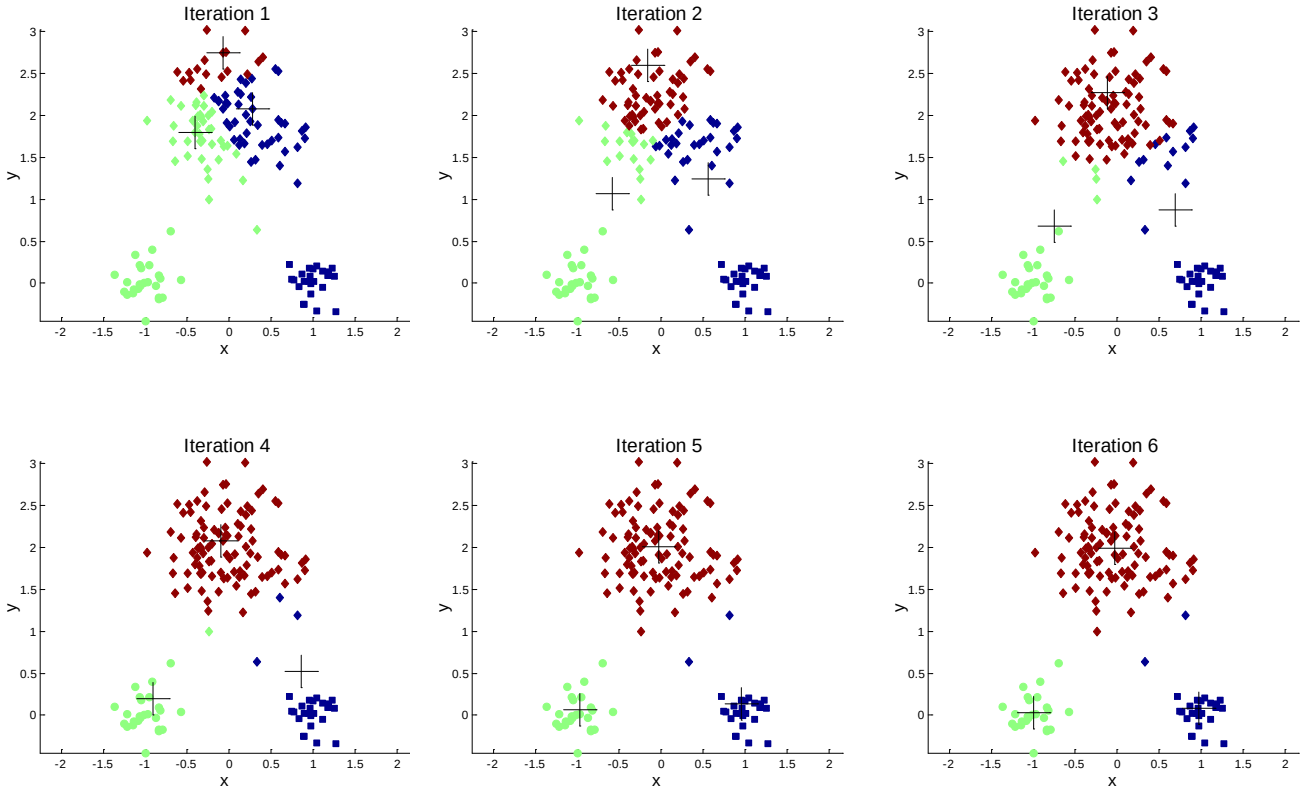


K均值簇

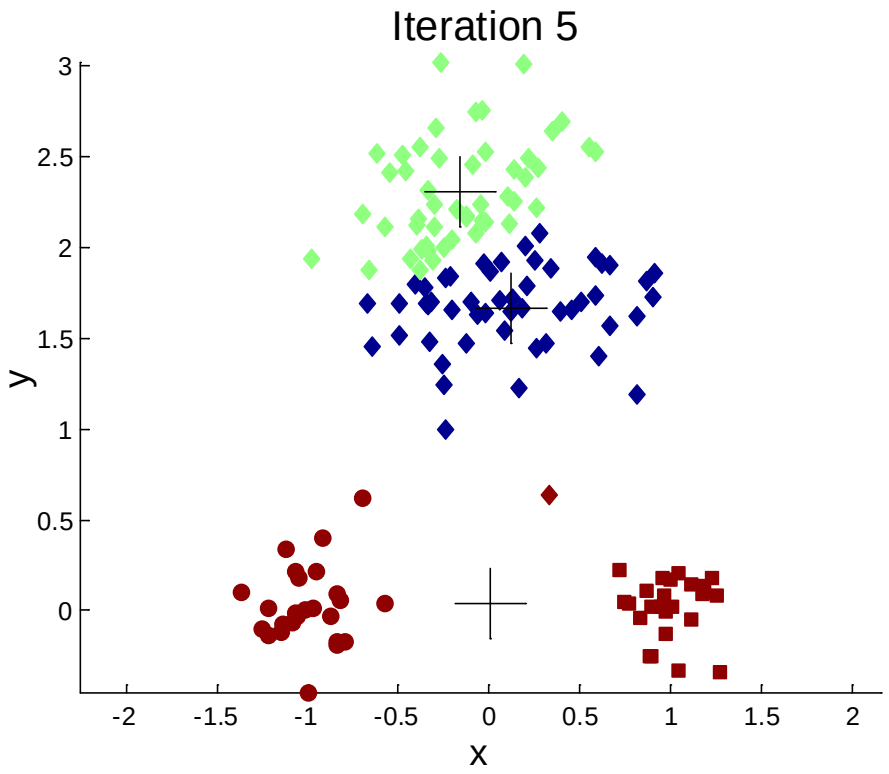
选择初始质心的重要性



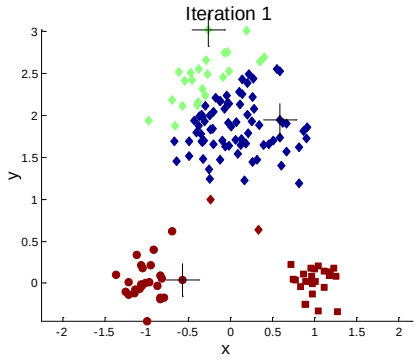
选择初始质心的重要性

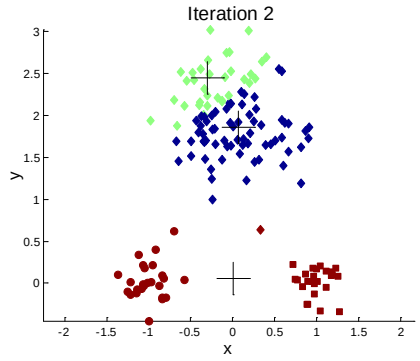

Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
x
y

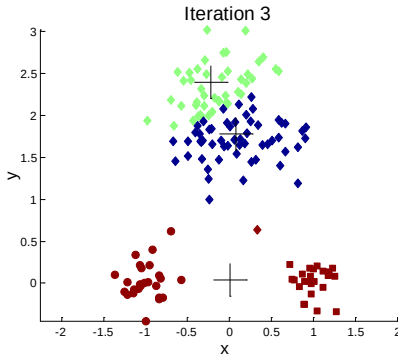
选择初始质心的重要性

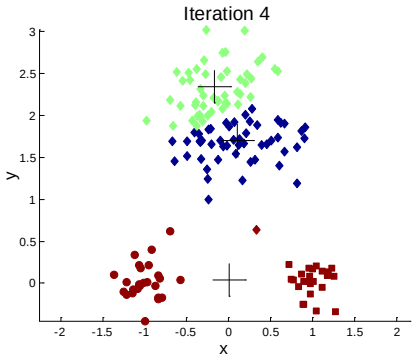


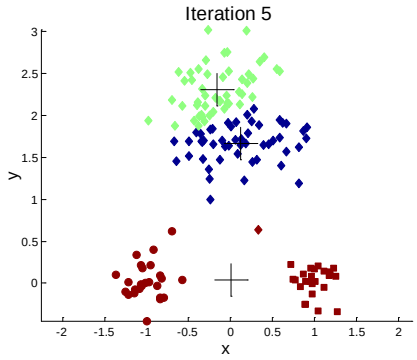
选择初始质心的重要性


Iteration 1
x
y


Iteration 2
x
y


Iteration 3
x
y


Iteration 4
x
y


Iteration 5
x
y

选择初始质心的问题

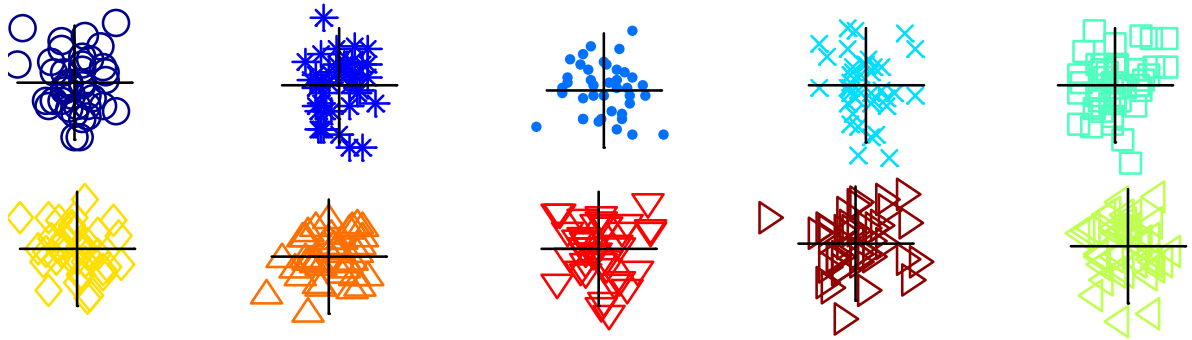
如果有K个“实”簇，那么从每个簇中选择一个质心的可能性很小。

- 当K较大时，几率相对较小
- 如果簇的大小相同都为 n, 那么

$$P = \frac{\text{从每个簇中选择一个质心的方式数量}}{\text{选择}K\text{个质心的方式数量}} = \frac{K!\,n^K}{(Kn)^K} = \frac{K!}{K^K}$$

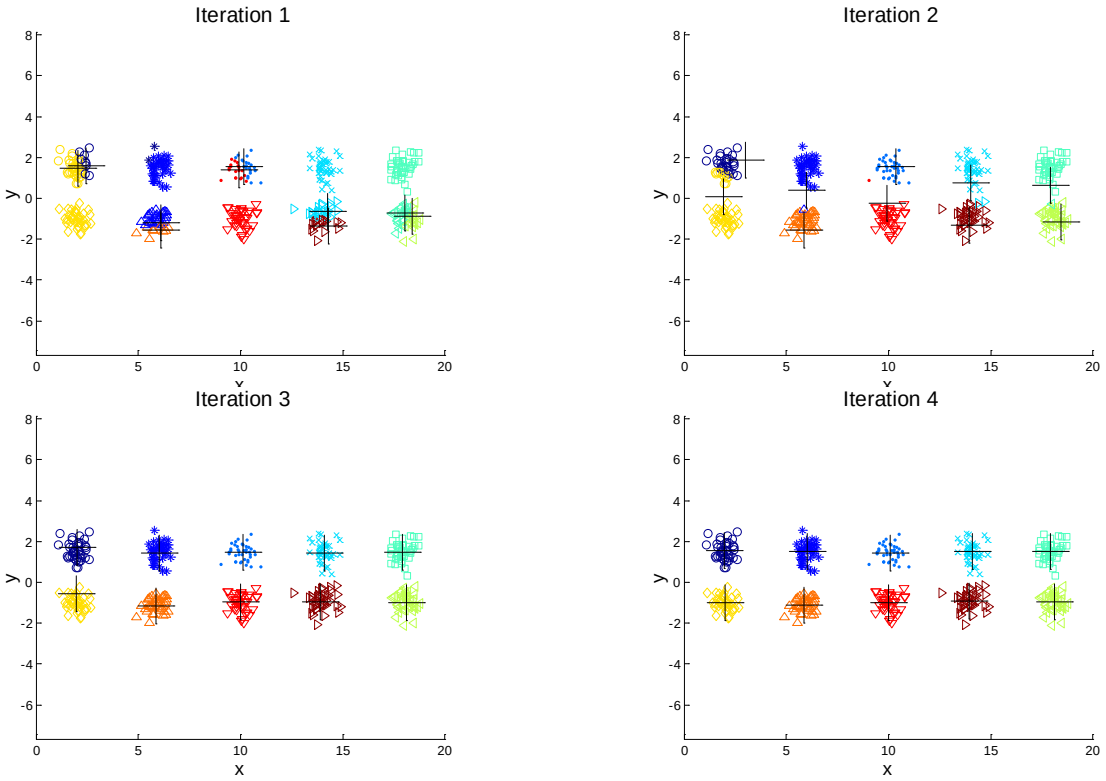
- 例如，如果K=10，那么概率 = $10!/10^{10} = 0.00036$
- 有时初始质心会以“正确”的方式重新调整自己，有时则不会.

10个簇的例子



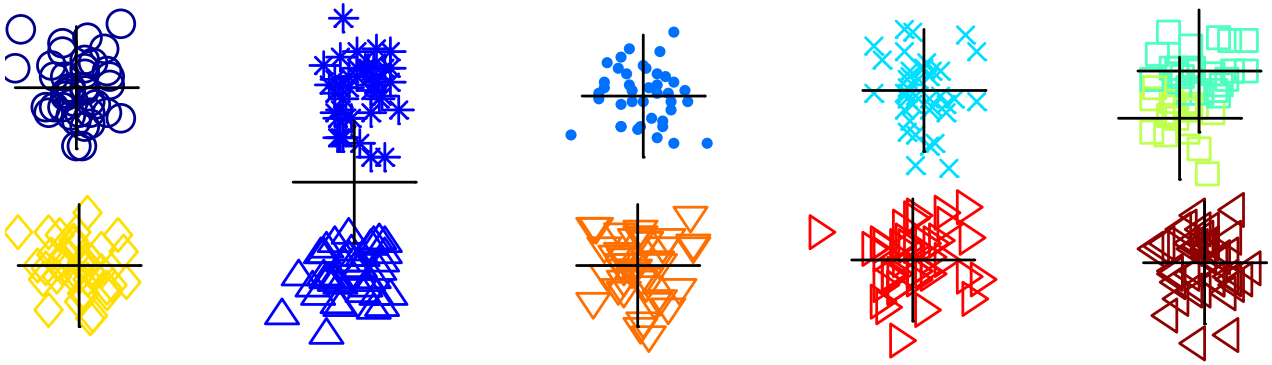
从每对簇的一个簇中的两个初始质心开始

10个簇的例子



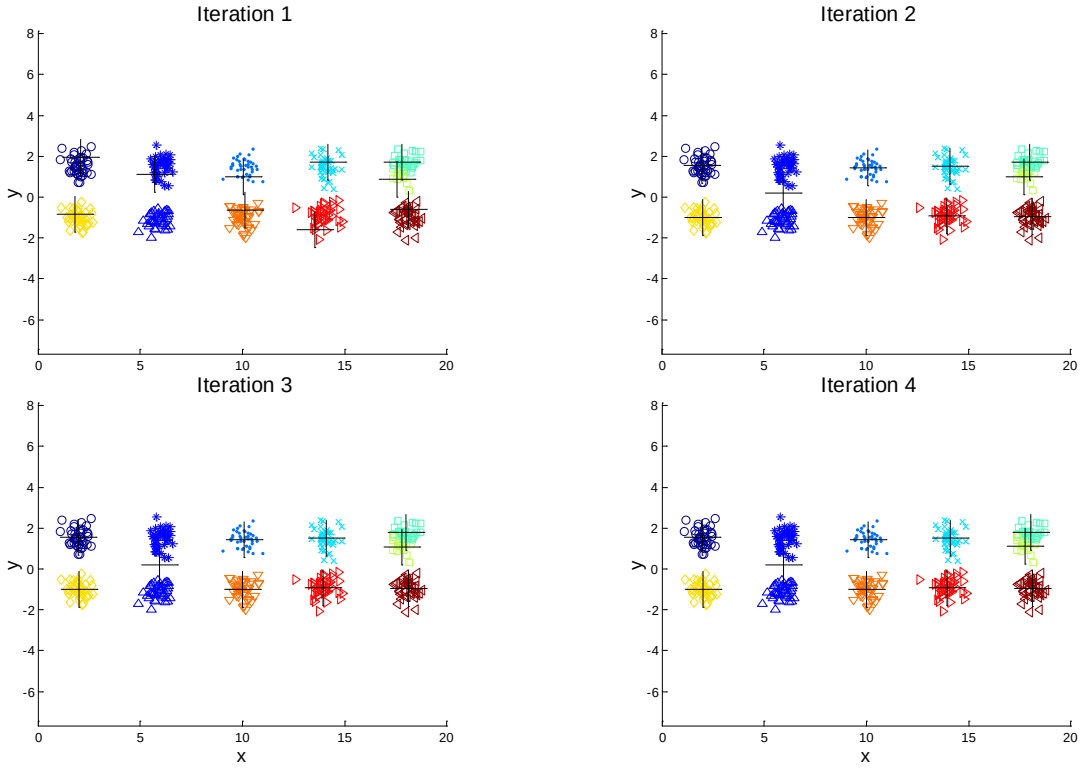
从每对簇的一个簇中的两个初始质心开始

10个簇的例子



开始时一些簇对具有三个初始质心，而有些簇对只有一个初始质心。

10个簇的例子



开始时一些簇对具有三个初始质心，而有些簇对只有一个初始质心。

关于初始质心问题的解决方法

重复运行多次

- 有帮助，但概论不由你决定。

采样并使用层次聚类提取K个簇来确定初始质心

选择>k个初始质心，然后在这些初始质心中进行选择

- 选择散开的点

二分K均值

- 不易受到初始化问题的影响

K均值++

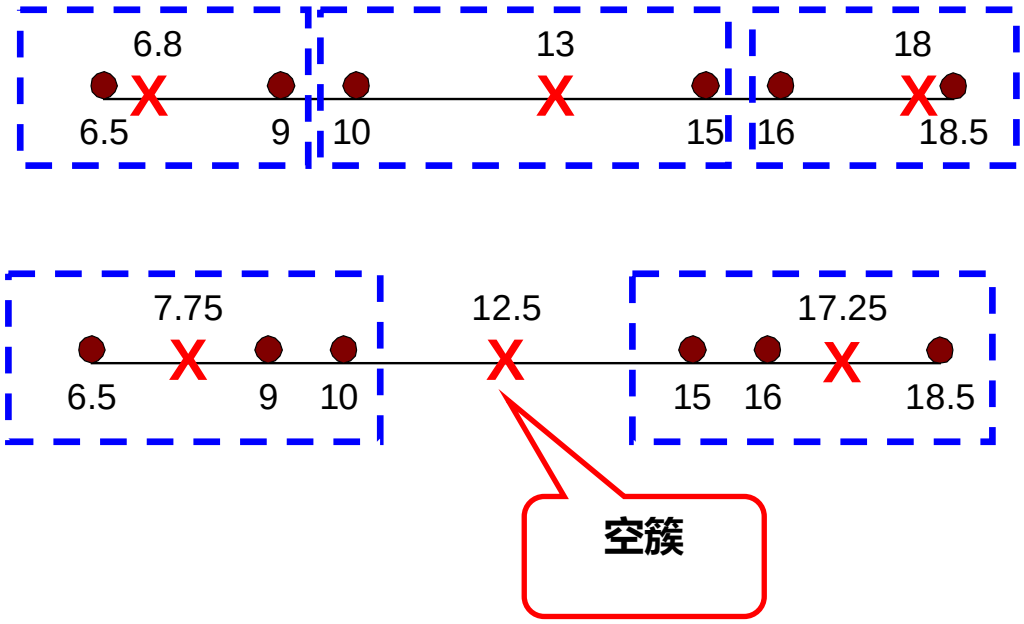
这种方法可能比随机初始化慢，但就SSE而言往往更好

K均值++的基本思路是假设已经选取了n个初始聚类中心(0<n<K)，则在选取第n+1个聚类中心时：距离当前n个聚类中心越远的点会有更高的概率被选为第n+1个聚类中心。

1. 随机选择一个点作为初始质心
2. Repeat k – 1 步
3. 计算每个点到最近质心的距离$d(x)$
4. 计算每个点成为下一个聚类中心的概率$\frac{d^2(x)}{\sum_{x \in X} d^2(x)}$
5. 基于概率从剩余点中选择新的质心
5. End

空簇

K均值可以产生空簇



处理空簇

基本的K均值算法会产生空簇

对付空簇的几种策略

- 选择对SSE贡献最大的点作为新的中心
- 或从SSE最高的簇中选择一个点作为新的中心
- 如果存在多个空簇，则可以重复上述操作多次。

增量地更新质心

在基本的K均值算法中，在所有的点都被分配给某个质心之后，质心被更新；

另一种方法是在每次指派后就更新质心（增量方法)

- 每次指派会更新零个或两个质心
- 缺点：计算成本高
- 缺点：导致次序依赖性
- 优点：保证不会产生空簇

预处理和后处理

预处理

- 归一化数据
- 消除离群值

后处理

- 消除可能代表离群值的小簇
- 拆分“松散”簇，即SSE相对较高的簇
- 合并“接近”且SSE相对较低的簇

二分K均值

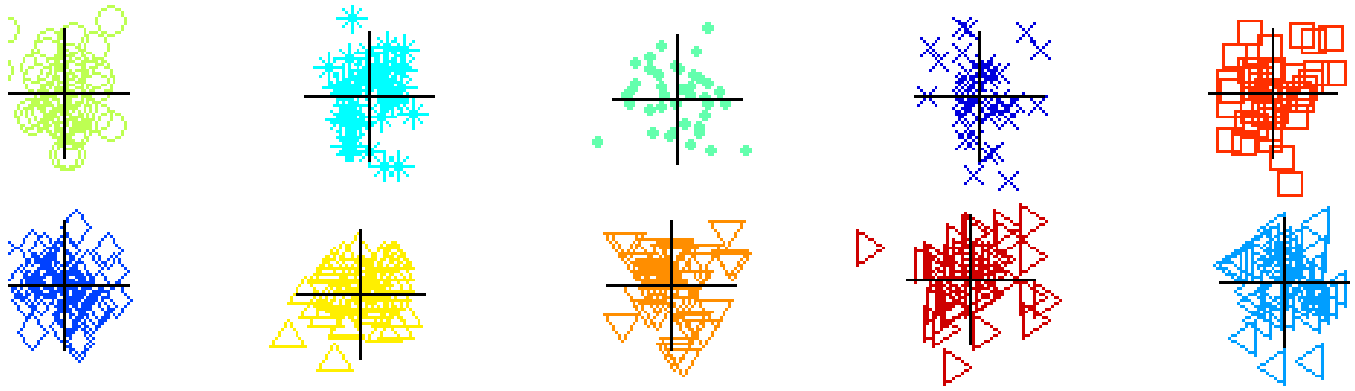
二分K均值算法：为了得到K个簇，将所有点的集合分裂成两个簇，从这些簇中选取一个继续分裂，如此下去，直到产生 K个簇。

- K均值的一个变种，它可以产生一个划分或层次聚类
- 具体步骤：

1. 将所有点看成一个簇；
2. 对每个簇，进行如下操作
   （1）计算总误差
   （2）在给定的簇上进行K-Means聚类（k=2）
   （3）计算将该簇一分为二之后的总误差
3. 选择使得误差SSE最大的那个簇进行划分操作
4. 重复2—3操作，直到达到用户指定的簇数目为止；

另一种做法是：选择SSE最小的簇进行划分，直到簇数目达到用户指定的数目为止。

二分K均值例子

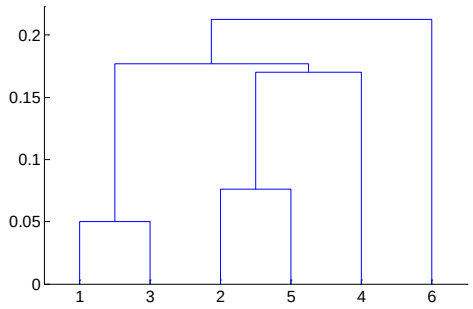


层次聚类

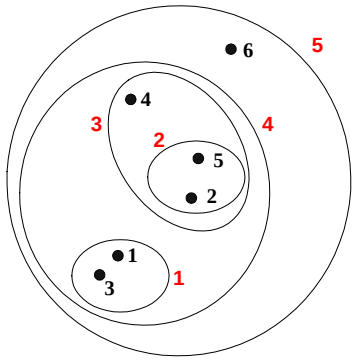
生成一组按层次树组织的嵌套簇

结果可以被可视化为树状图

- 记录合并或分裂次序的树状图



树状图



嵌套簇图

层次聚类的优势

不必假定任何特定数量的簇

- 通过在恰当水平上“切割”树状图，可以获得任何所需数量的簇

它们可能对应于某种有实际意义的分类法

- 比如生物科学中的物种划分、进化关系研究…

层次聚类

层次聚类的两种主要类型

- 凝聚型:
  - 从点作为个体簇开始
  - 每一步合并最近的一对簇，直到只剩下一个簇（或k个簇）
- 分裂型:
  - 从包含所有点的某个簇开始
  - 每一步分裂一个簇，直到每个簇包含一个单独的点（或有k个簇））

传统的层次聚类算法基于相似性或距离矩阵

- 一次合并或拆分一个簇

凝聚型层次聚类

思路：

先计算样本之间的距离；每次将距离最近的点合并到同一个类。然后，再计算类与类之间的距离，将距离最近的类合并为一个大类。不停的合并，直到合成了一个类。

基本算法步骤如下：

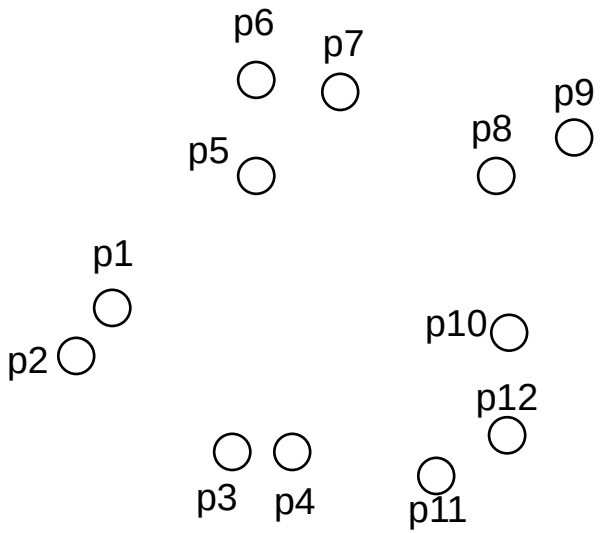
1. 计算邻近度矩阵
2. 从个体点作为一个簇开始
3. Repeat
4. 合并最接近的两个簇
5. 更新邻近度矩阵
6. Until 仅剩下一个簇

关键操作是计算两个簇的邻近度

– 不同的定义簇间距离的方法区分了不同的算法

起始阶段

从单个点的簇和邻近度矩阵开始

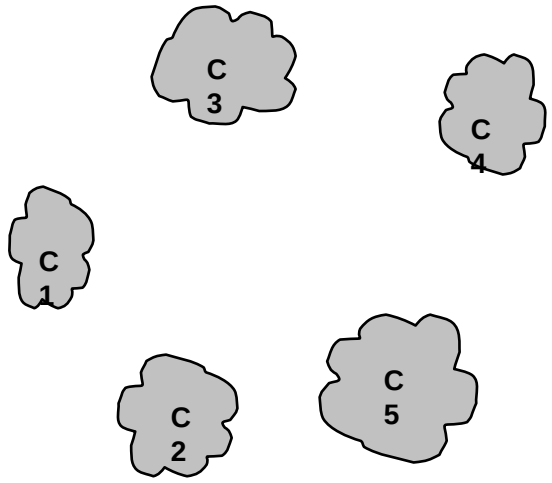


	p1	p2	p3	p4	p5	. .
p1						.
p2						
p3						
p4						
p5						
.						
.						
.						

邻近度矩阵

中间阶段

经过一些合并步骤，我们得到了一些簇

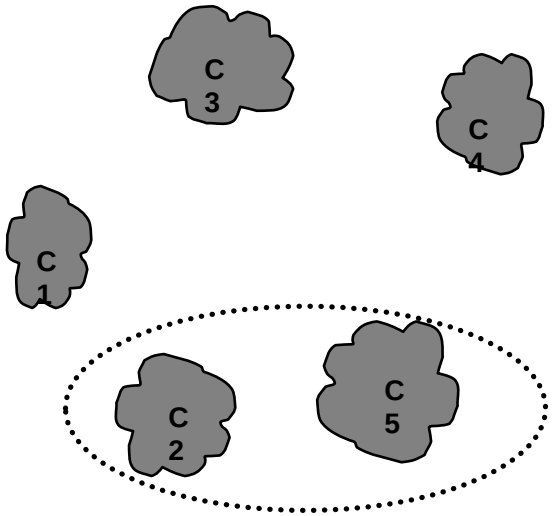


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

邻近度矩阵

中间阶段

- 我们希望合并两个最近的簇（C2和C5）并更新邻近度矩阵。

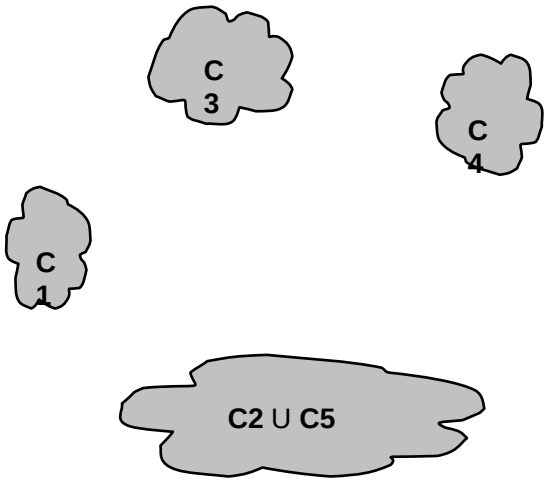


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

邻近度矩阵

合并之后

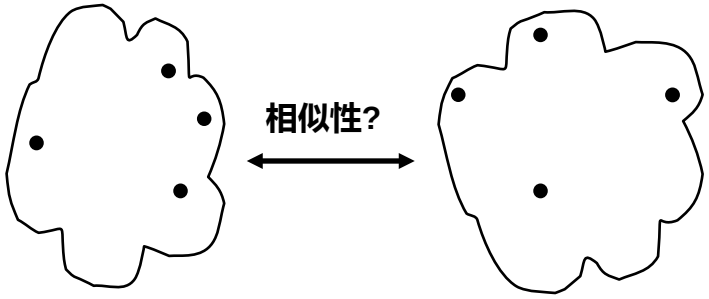
问题是”我们如何更新邻近矩阵？”



	C1	C2 ∪ C5	C3	C4
C1		?		
C2 ∪ C5	?	?	?	?
C3		?		
C4		?		

邻近度矩阵

定义簇间距离



MIN

MAX

组平均

质心间的距离

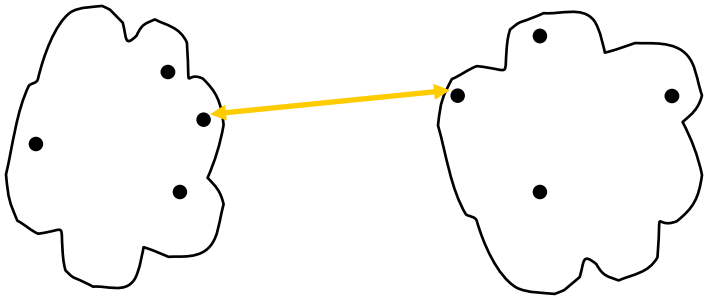
其他由目标函数驱动的方法

- 利用平方误差的Ward方法

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

邻近度矩阵

定义簇间距离



MIN

MAX

组平均

质心间的距离

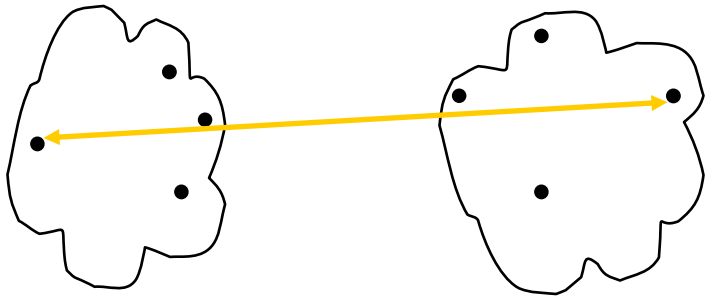
其他由目标函数驱动的方法

- 利用平方误差的Ward方法

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

邻近度矩阵

定义簇间距离



MIN

MAX

组平均

质心间的距离

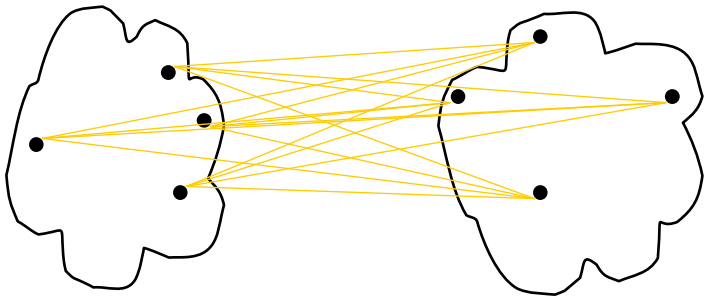
其他由目标函数驱动的方法

- 利用平方误差的Ward方法

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

邻近度矩阵

如何定义簇间距离



MIN

MAX

组平均

质心间的距离

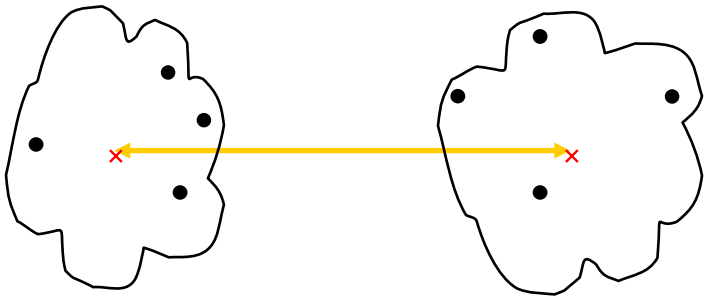
其他由目标函数驱动的方法

- 利用平方误差的Ward方法

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

邻近度矩阵

如何定义簇间距离



MIN

MAX

组平均

质心间的距离

其他由目标函数驱动的方法

- 利用平方误差的Ward方法

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

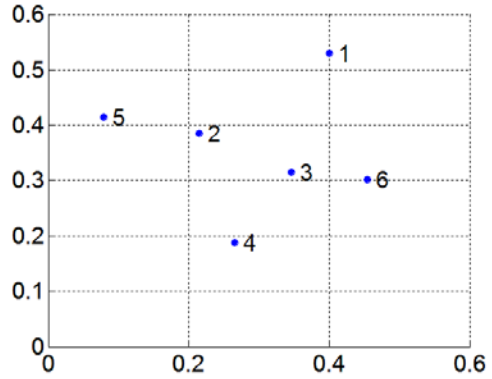
邻近度矩阵

MIN或单链

两个簇的邻近度定义为两个不同簇中任意两点之间的最短距离

- 由一对点决定, 即邻近图中的一条链

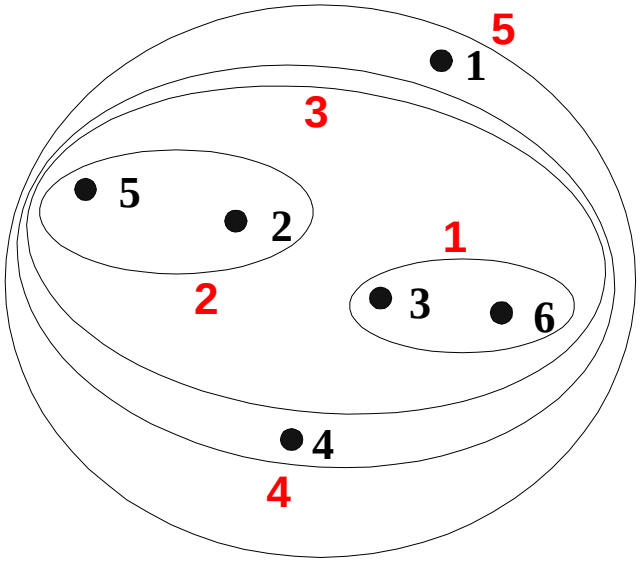
例子:



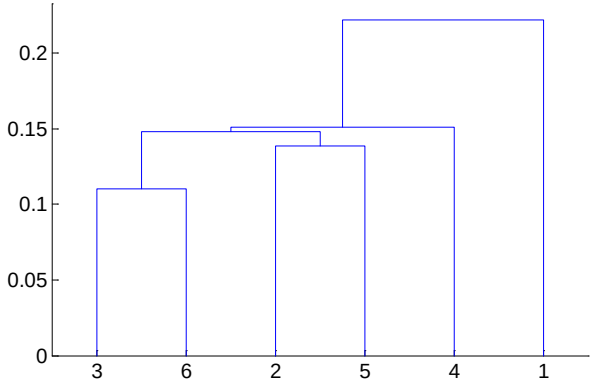
距离矩阵:

	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

层次聚类：MIN

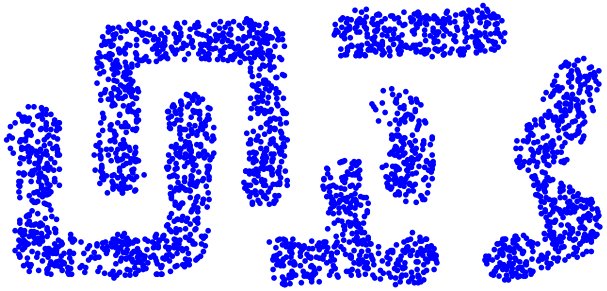


单链聚类

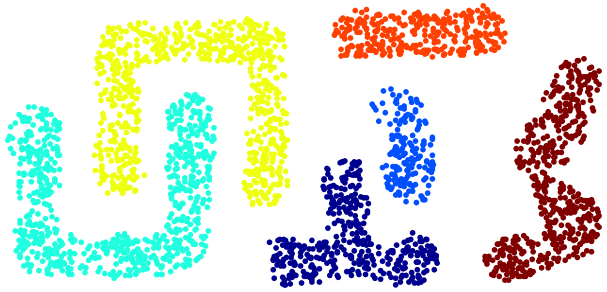


单链树状图

MIN的优势



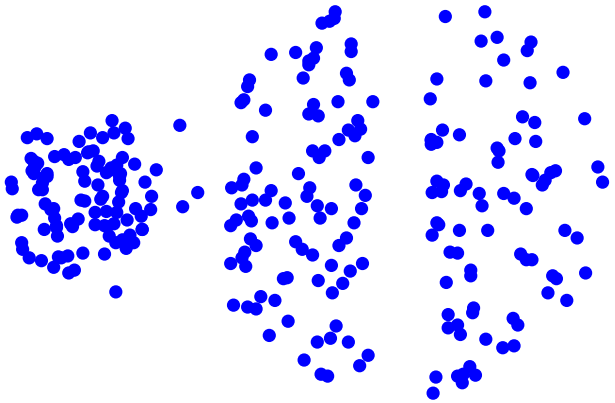
原始点



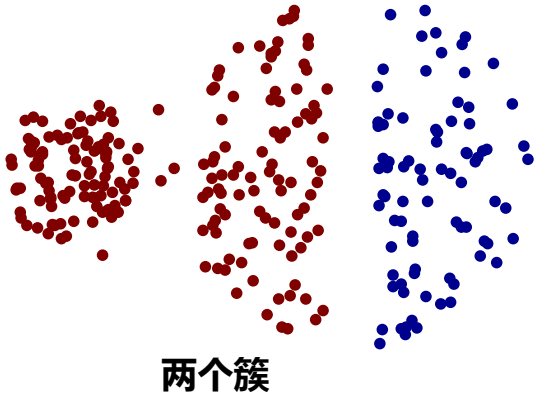
6个簇

- 可以处理非椭圆形状

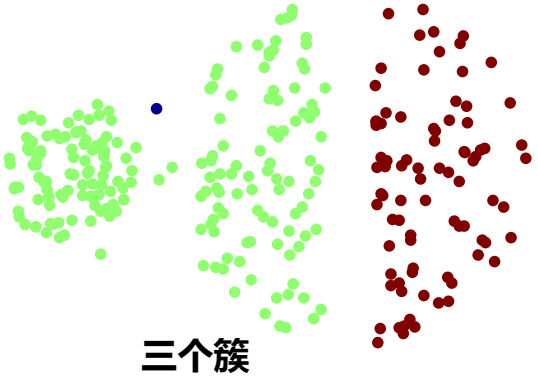
MIN的缺点



原始点



两个簇



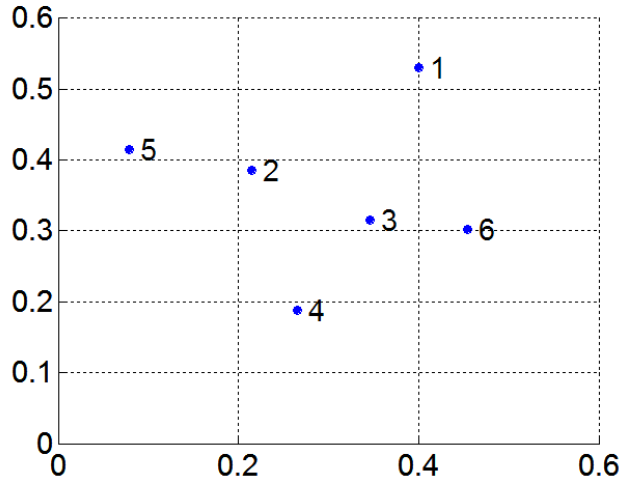
三个簇

- 对噪声和离群点比较敏感

MAX或全链

两个簇的邻近度定义为两个簇中任意两点之间的最长距离（最小相似度）

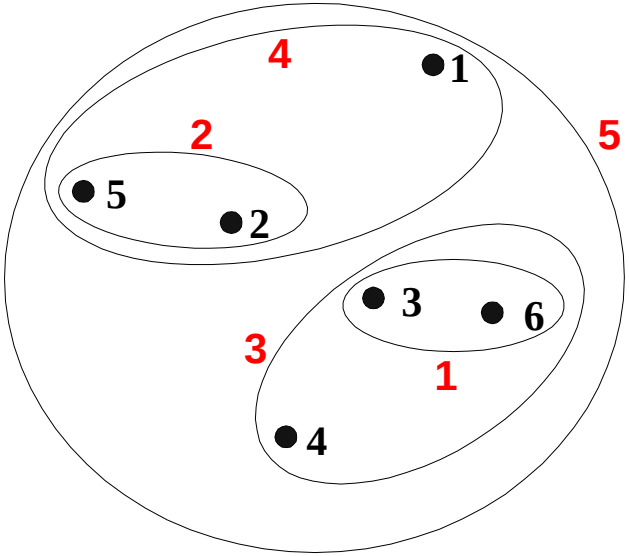
- 由两个簇中的所有点对决定



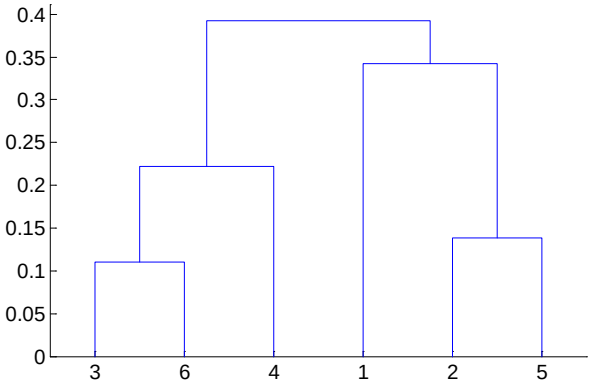
距离矩阵:

	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

层次聚类：MAX

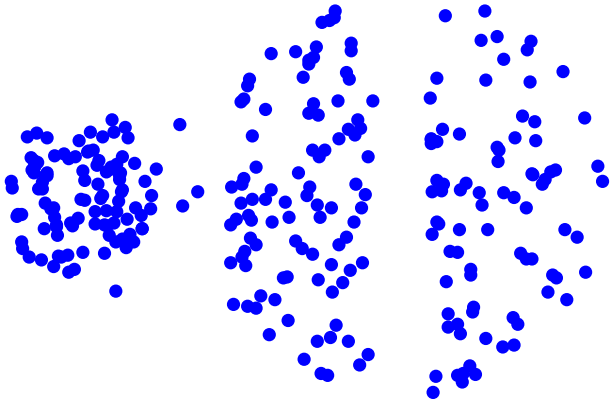


全链聚类

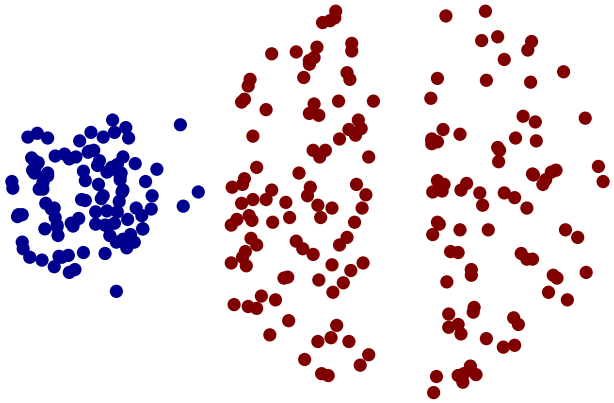


全链树状图

MAX的优势



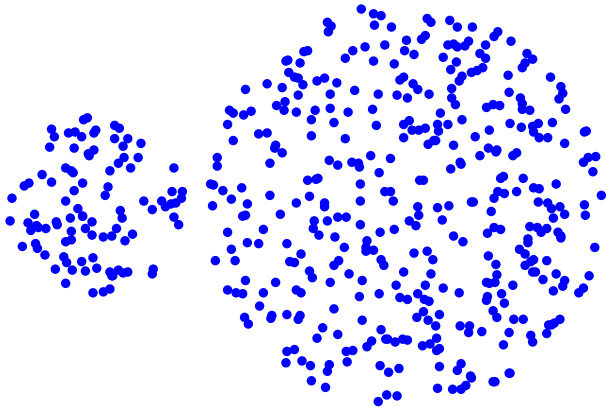
原始点



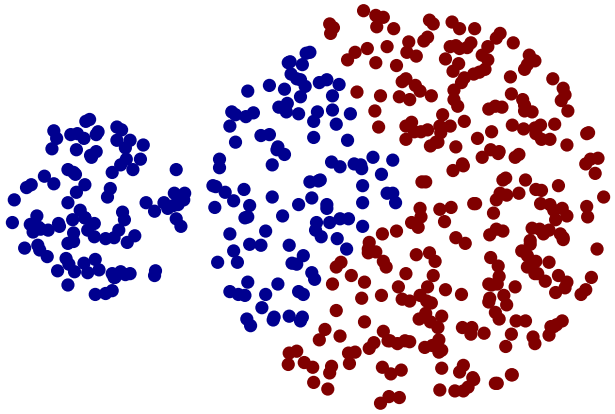
两个簇

- 对噪声和离群点不太敏感

MAX的缺点



原始点



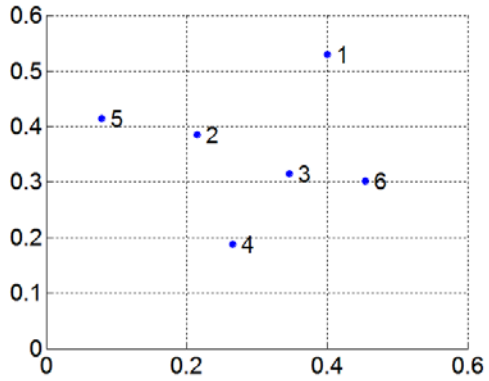
两个簇

- 倾向于分裂大簇
- 形成的簇偏向于球形簇

组平均

两个簇的邻近度定义为不同簇的所有点对邻近度的平均值。

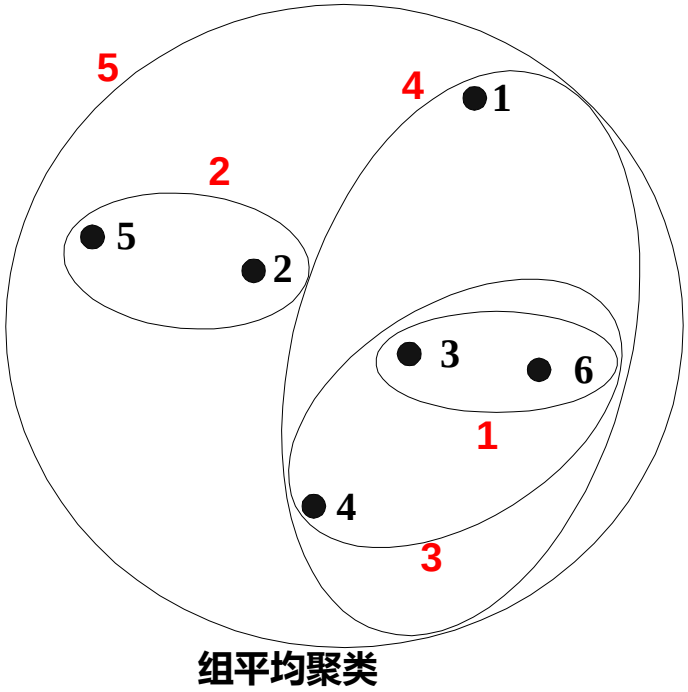
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{p_i \in \text{Cluster}_i} \text{proximity}(p_i, p_j)}{p_j \in \text{Cluster}_j} \\ |\text{Cluster}_i| \times |\text{Cluster}_j|$$



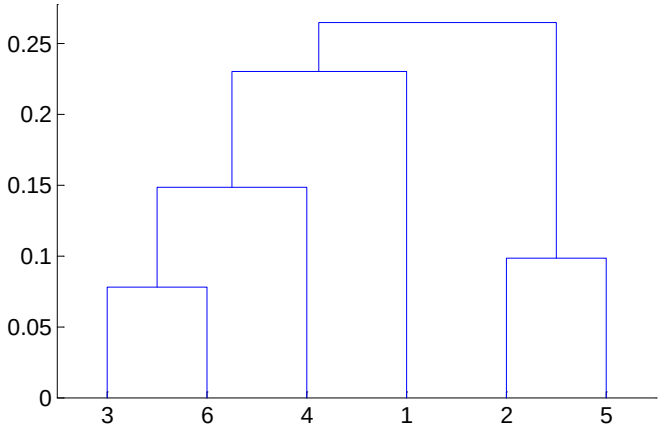
Distance Matrix:

	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

层次聚类：组平均



组平均聚类



组平均树状图

层次聚类：组平均

界于单链和全链之间的折中方法

优点

- 对噪声和离群点不太敏感

缺点

- 偏向于形成球形簇

簇的邻近度：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

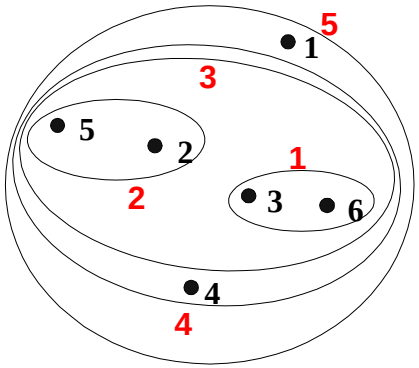
- 可以证明：当将两个点间距离的平方作为它们之间的邻近度时，Ward方法与组平均非常相似

不易受噪声和离群点的影响

偏向于形成球形簇

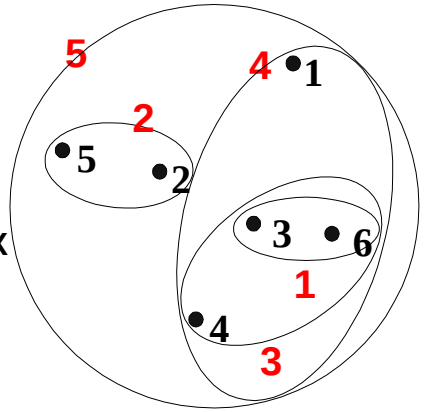
可以用做K均值的初始化

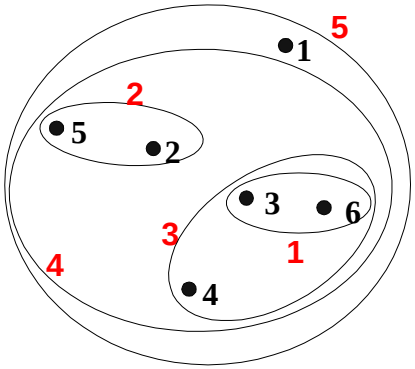
层次聚类：对比



MIN

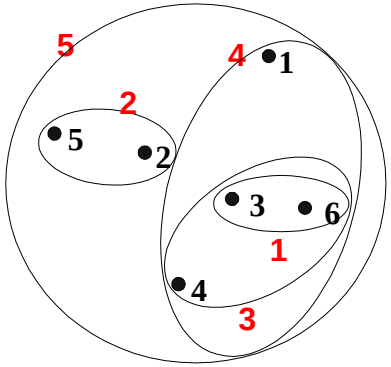
MAX





组平均

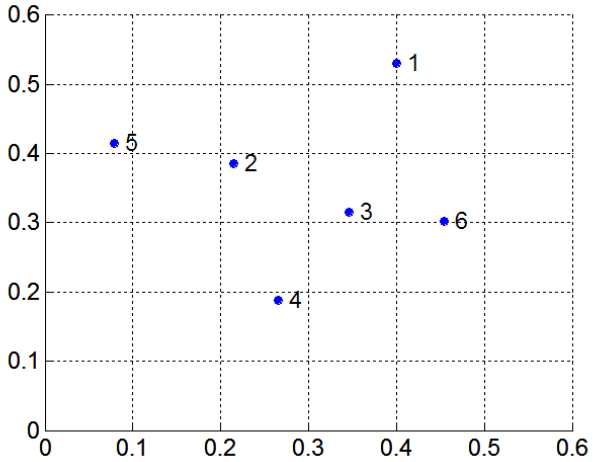
Ward方法

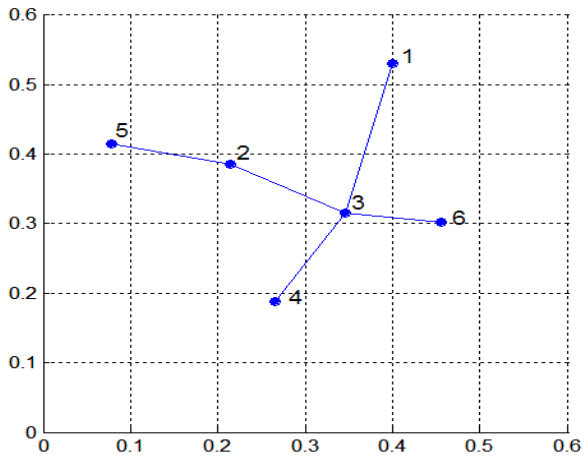


MST：分裂层次聚类

构建最小生成树 (Minimum Spanning Tree， MST)

- 在连续的步骤中，寻找最近的一对点（p，q），使得一个点（p）在当前树中，而另一个点（q）不在当前树中
- 把q加到树上，在p和q之间加一条边
- 直至有n-1条边，且此时边权重（一般为边的长度）的和最小

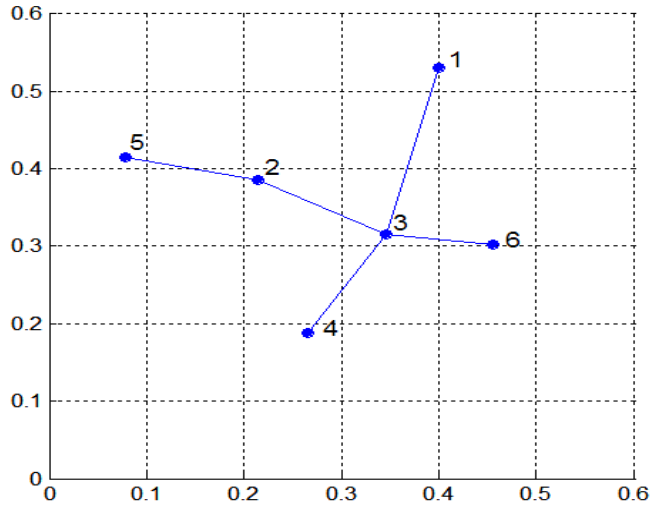




MST：分裂层次聚类

利用MST来构建层次聚类

1：计算相异度图的最小生成树。
2：**repeat**
3：　断开对应于最大相异度的边，创建一个新的簇。
4：**until** 只剩下单个簇。



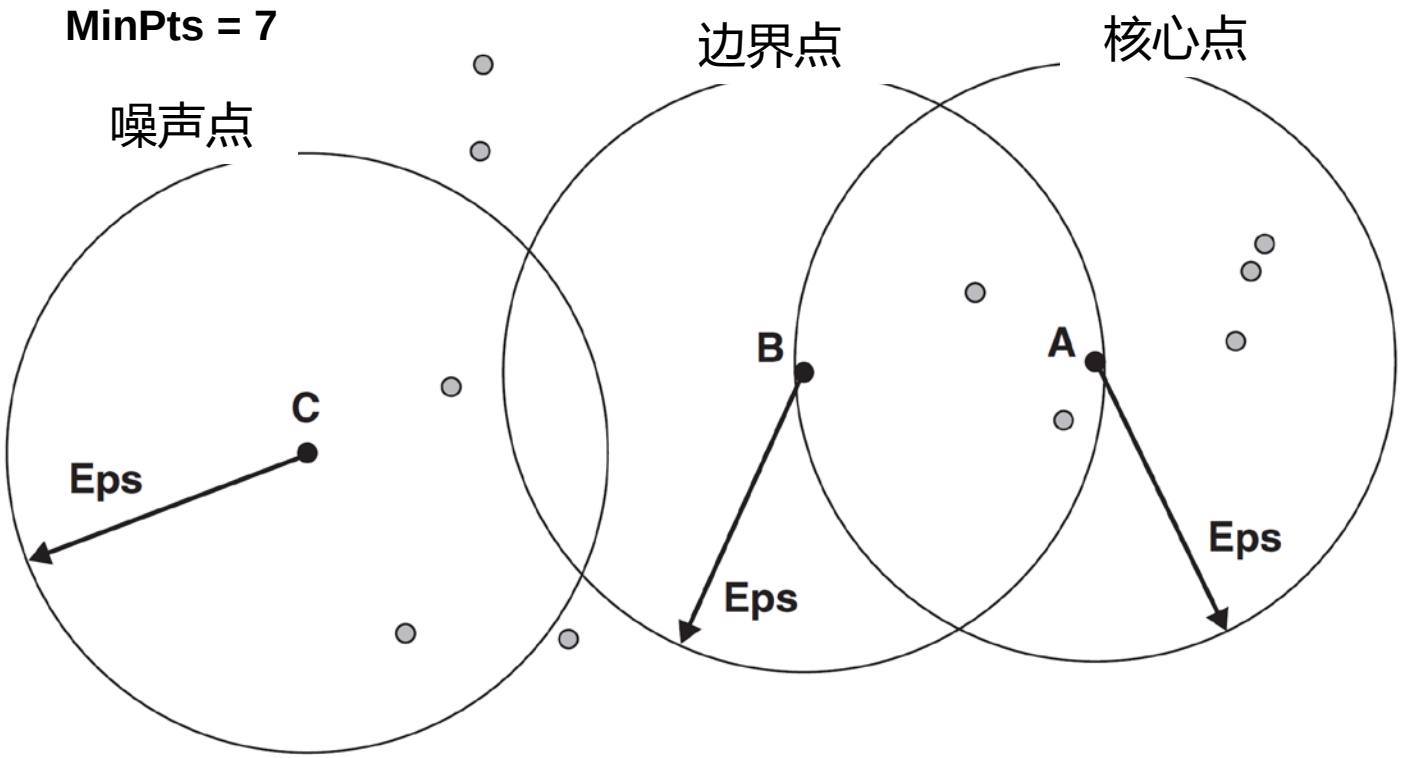
层次聚类：问题和局限性

- 一旦做出合并或分裂两个簇的决策，以后就不能再撤销;
- 缺乏直接被最小化的全局目标函数。
- 不同的方案存在以下一个或多个问题：
  - 对噪声和离群点敏感
  - 难以处理不同大小和非球形形状的簇
  - 会分裂大的簇

DBSCAN 是一种基于密度的聚类算法。

- 密度=指定半径（Eps）内的点数
- 如果一个点在Eps中有大于等于指定数量（MinPts）的样本点，那么它就是一个核心点
  - 计数需包含点本身
- 边界点不是核心点，但它落在某个核心点的邻域内。
- 噪声点是既非核心点也非边界点的任何点。

DBSCAN：核心，边界，和噪声点



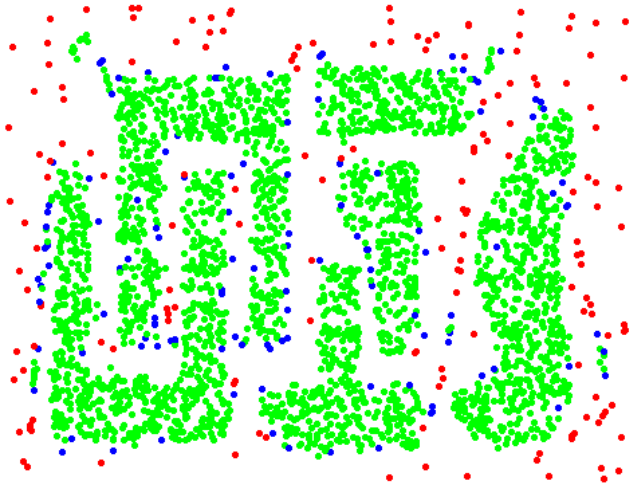
DBSCAN 算法

基本步骤：

1：将所有点标记为核心点、边界点或噪声点。
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。

DBSCAN:核心，边界和噪声点

原始点

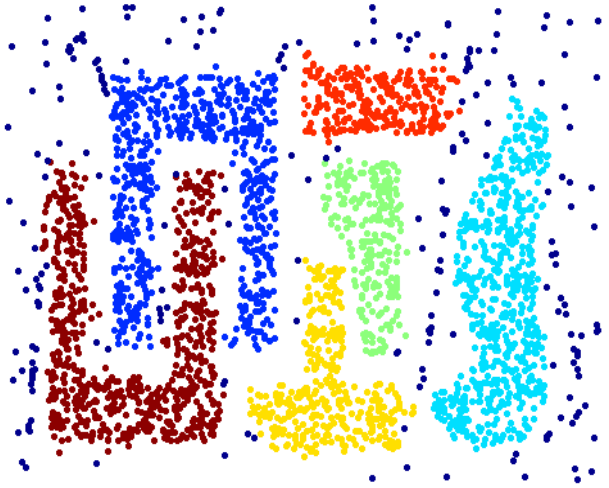


点的种类: 核心, 边界 和 噪声

Eps = 10, MinPts = 4

何时DBSCAN表现好

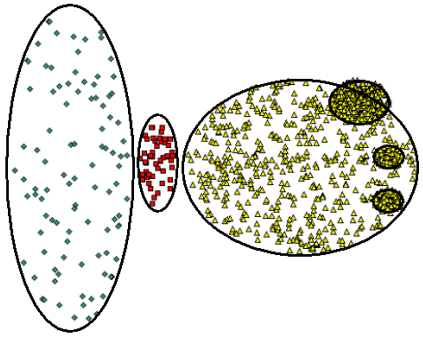
原始点



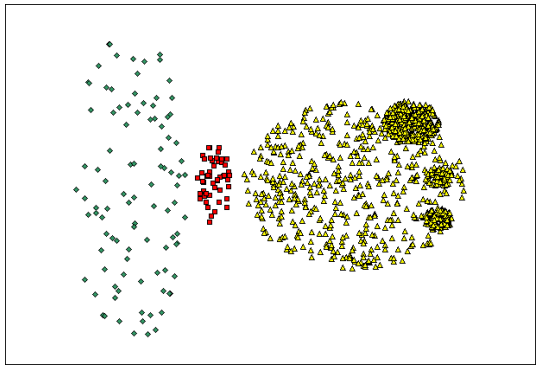
DBSCAN发现的簇

- 能抗噪声
- 能处理任意形状和大小的簇

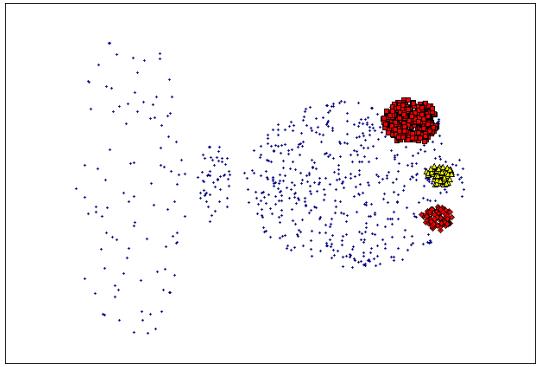
何时DBSCAN表现不好



原始点



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

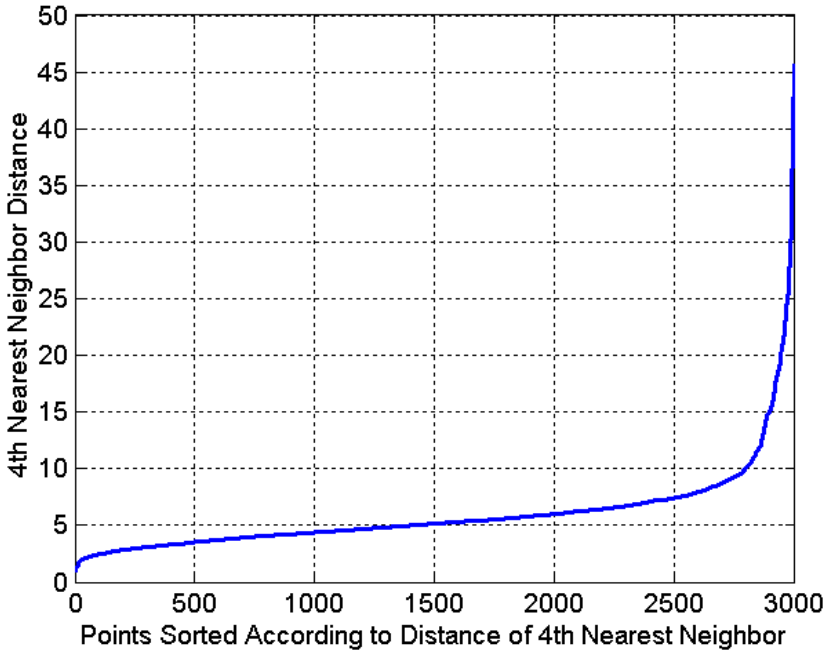
- 当簇的密度变化太大
- 当遇到高维数据

DBSCAN：Eps和MinPts的确定

思路：

对于某个簇中的点，它们到它的第k个最近邻的距离大致相同；而噪声点在较远的距离有第k个最近邻；

可以计算每个点到自身第k个最近邻的距离，并据此进行排序，如下图：



簇评估

对于监督分类，我们有多种方法来评估我们的模型的好坏

- 准确率(Accuracy), 精确率(precision), 召回率(recall)

对于聚类分析来说，类似的问题是如何评估所得到的簇的“优度”?

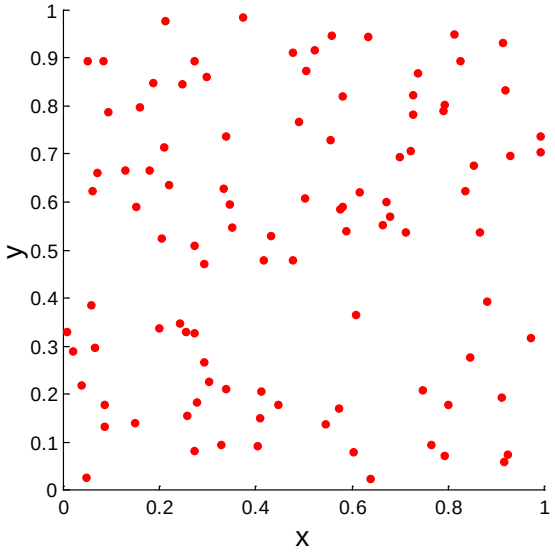
当然簇的好坏也是跟个人的主观看法有关!

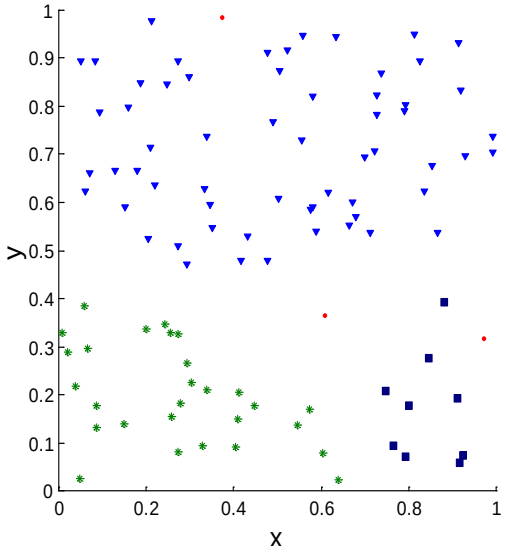
那我们为什么要评估它们呢?

- 比较聚类算法
- 比较两组簇
- 比较两个簇

在随机数据中发现的簇

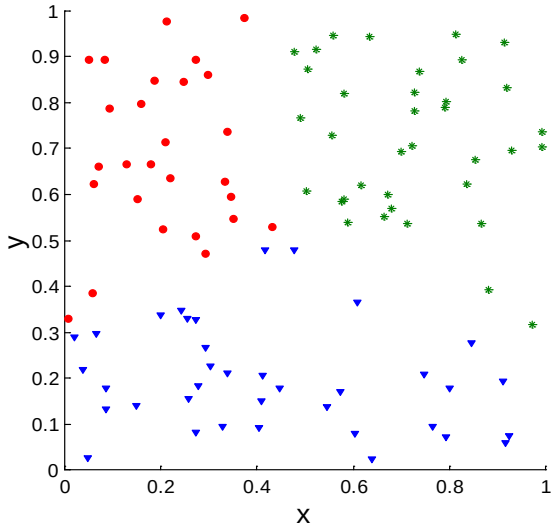
随机点

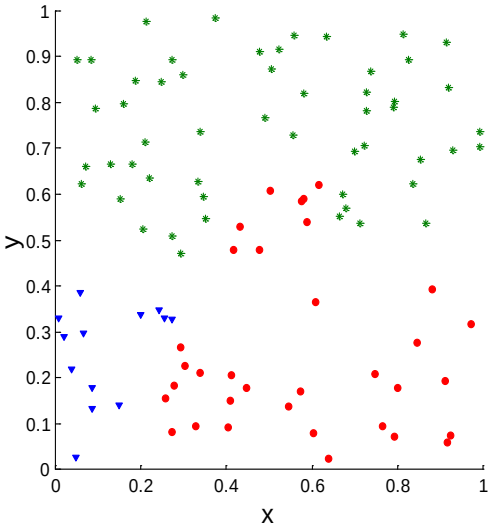




DBSCAN

K均值





全链（基于MAX的层次聚类）

评估簇要考虑的重要方面

1. 确定数据集的聚类趋势，即识别数据中是否实际存在非随机结构。
2. 将聚类分析的结果与外部已知的结果进行比较，例如，与外部提供的类别标签进行比较。
3. 在不参考外部信息的情况下，评估聚类分析结果与数据的拟合程度。
4. 比较两个簇集，以确定哪个更好。
5. 确定正确的簇个数。

对于2、3和4，我们可以进一步区分是要评估整体聚类结果还是只评估个别的簇。

簇评估的度量

用于判断聚类有效性各个方面的数值度量分为以下三种类型：

- 外部指标:用于度量簇标签与外部提供的类标签匹配的程度。
  - 熵
- 内部指标:用来衡量聚类结构的优劣，而不考虑外部信息。
  - 误差平方和Sum of Squared Error (SSE)
- 相对指标: 用来比较不同的聚类或簇。
  - 通常利用外部或内部指标，例如SSE或熵

有时这些被称为标准而不是指标

- 然而，有时标准是指一般策略，指标是实现标准的数值度量。

通过相关性度量簇的有效性

两个矩阵：

- 邻近度矩阵
- 理想的相似度矩阵
  - 行标与列标均为所有数据点
  - 如果关联的一对点属于同一簇，则矩阵的该项为1
  - 如果关联的一对点属于不同的簇，则矩阵的该项为0

计算两个矩阵之间的相关性

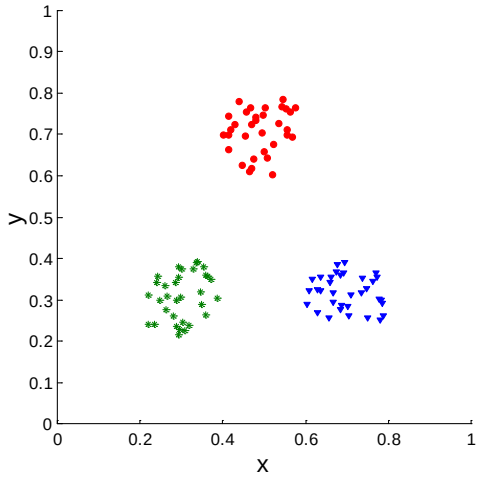
- 因为矩阵是对称的，所以只需要计算 n（n-1）/2个项的相关性。

高度相关表示属于同一簇的点彼此接近。

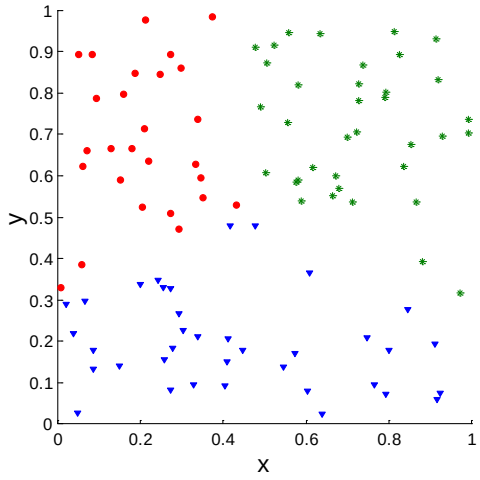
对于某些基于密度或基于近邻的簇不是一个好的度量。

通过相关性度量簇的有效性

两个数据集K均值聚类的理想相似度矩阵和邻近度矩阵的相关性。



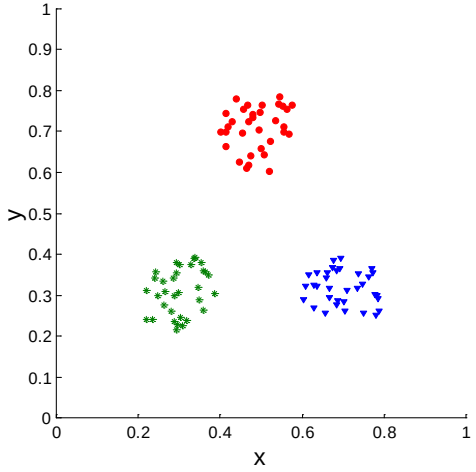
Corr = 0.9235

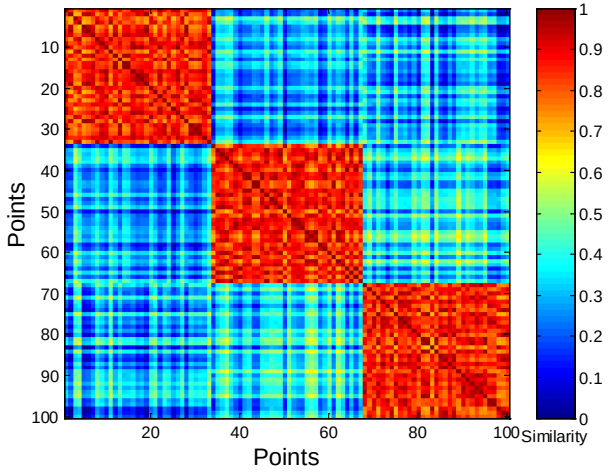


Corr = 0.5810

通过相似度矩阵可视地评价聚类

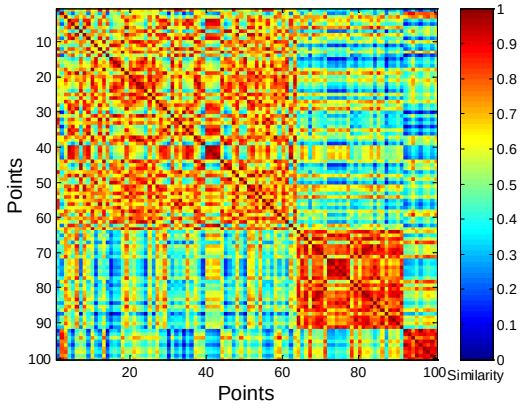
按照簇标签对相似度矩阵的行列排序，然后画出它。

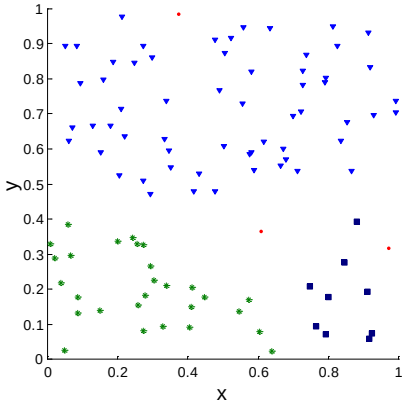




通过相似度矩阵可视地评价聚类

随机数据中的簇不是那么清晰

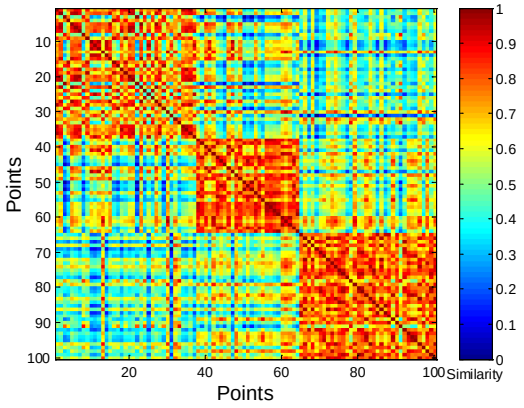


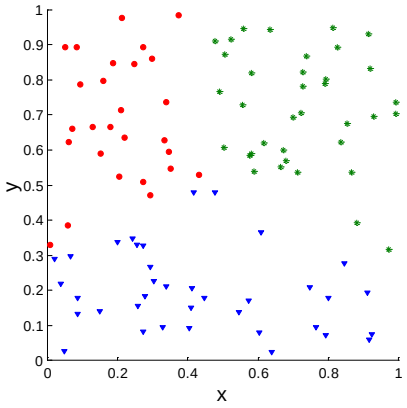


DBSCAN

通过相似度矩阵可视地评价聚类

随机数据中的簇不是那么清晰

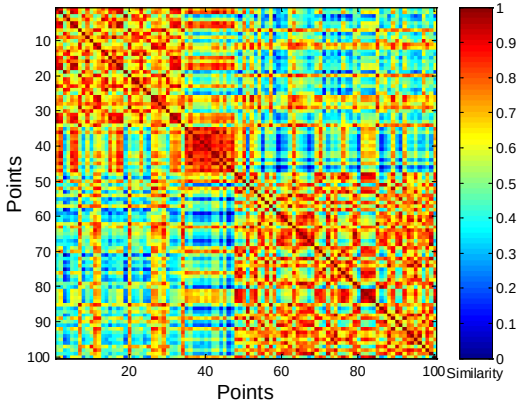


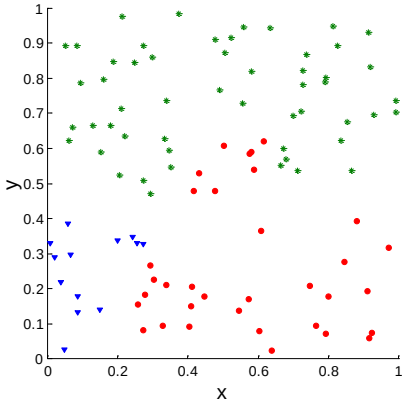


K均值

通过相似度矩阵可视地评价聚类

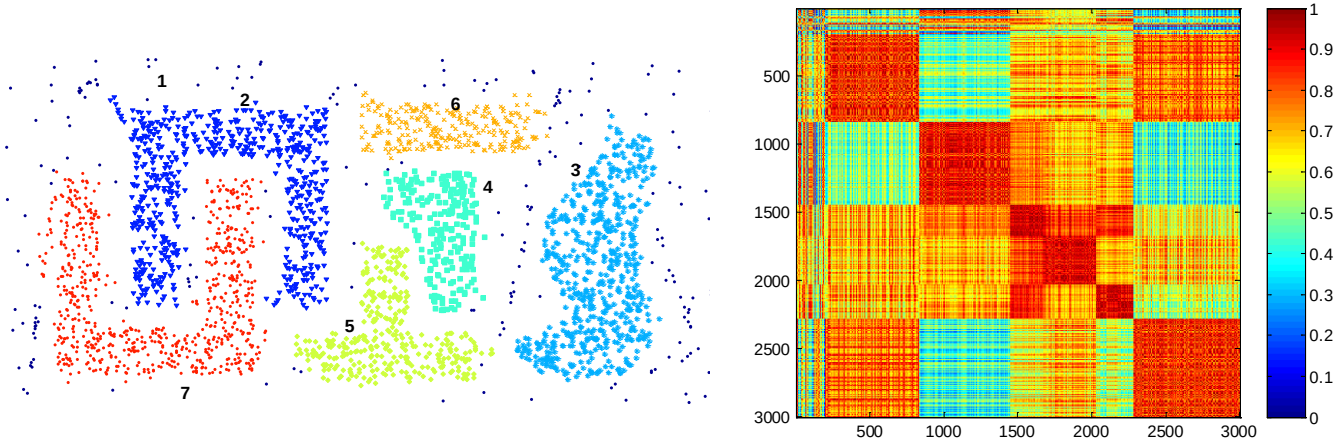
随机数据中的簇不是那么清晰





全链

通过相似度矩阵可视地评价聚类



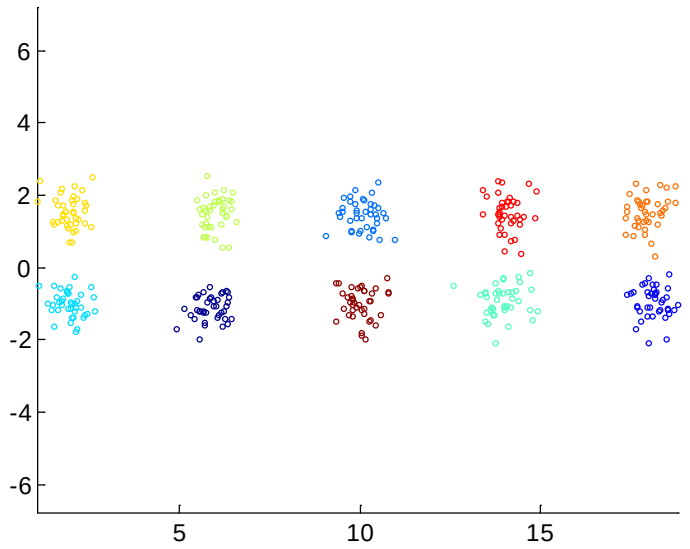
DBSCAN

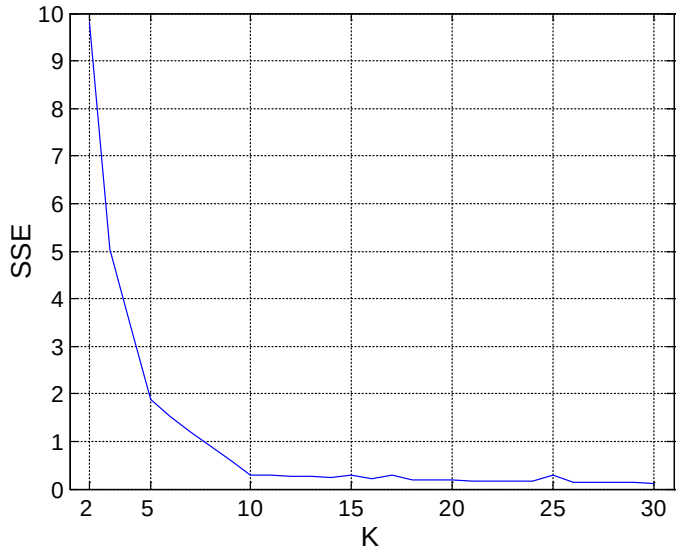
内部度量：SSE

内部指标：用来衡量聚类结构的优劣，而不考虑外部信息

--SSE

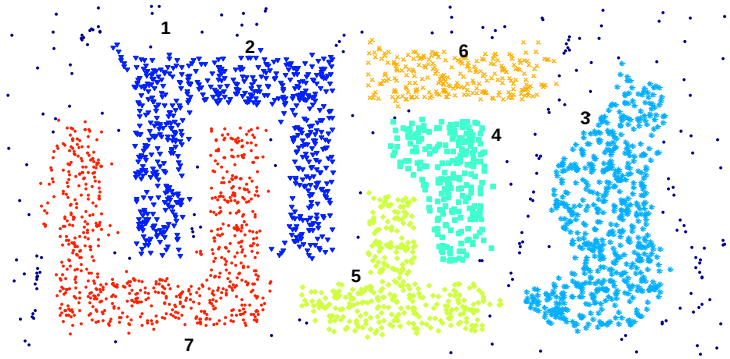
SSE也可以用来估计簇的数目。

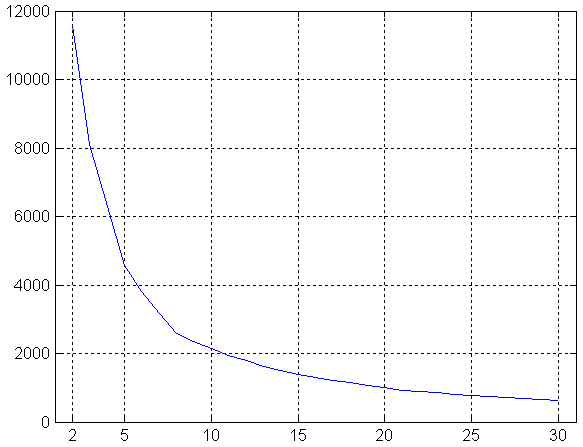




内部度量：SSE

对更复杂数据集的SSE曲线





利用K均值发现簇的SSE曲线

内部度量：凝聚度和分离度

簇的凝聚度: 度量簇中对象如何密切相关

- 例如:簇内平方和

簇的分离度: 度量某个簇不同于其他簇的地方

- 例如: 簇间误差平方和

凝聚度通过簇内平方和(within cluster sum of squares, WSS) (SSE)来度量

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

分离度通过簇间平方和(between cluster sum of squares, BSS)来度量

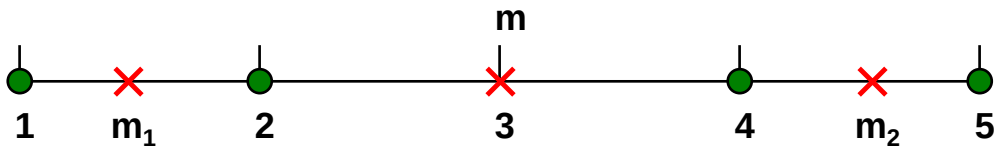
$$BSS = \sum_i |C_i|(m - m_i)^2$$

其中 $|C_i|$ 是簇 i 的大小，m_i是第i个簇的质心即第i个簇的均值，而m是所有数据点的总均值，x是簇i中的点。

内部度量：凝聚度和分离度

例子:

- BSS + WSS = 常数



K=1 个簇:

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

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

$$Total = 10 + 0 = 10$$

K=2 个簇:

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

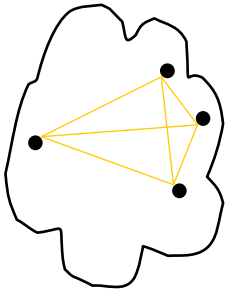
$$BSS = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$$

$$Total = 1 + 9 = 10$$

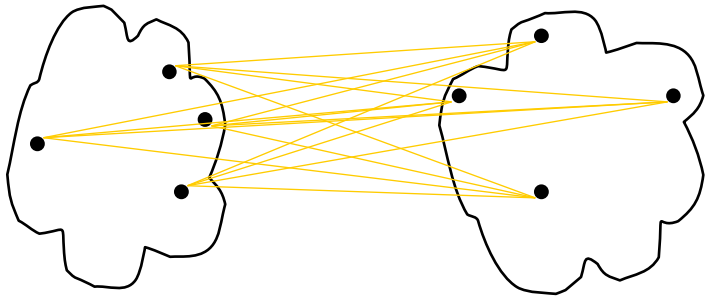
内部度量：凝聚度和分离度

基于邻近度图的方法也可以被用于凝聚度和分离度.

- 簇的凝聚度可以定义为连接簇内点的邻近度图中边的加权和。
- 两个簇间的分离度可以用从一个簇的点到另一个簇的点的边的加权和来度量。



cohesion



separation

簇有效性的外部度量：熵和纯度

表 8-9 《洛杉矶时报》文档数据集 K 均值聚类结果

簇	娱乐	财经	国外	都市	国内	体育	熵	纯度
1	3	5	40	506	96	27	1.227 0	0.747 4
2	4	7	280	29	39	2	1.147 2	0.775 6
3	1	1	1	7	4	671	0.181 3	0.979 6
4	10	162	3	119	73	2	1.748 7	0.439 0
5	331	22	5	70	13	23	1.397 6	0.713 4
6	5	358	12	212	48	13	1.552 3	0.552 5
合计	354	555	341	943	273	738	1.145 0	0.720 3

- **熵**：每个簇由单个类的对象组成的程度。对于每个簇，首先计算数据的类分布，即对于簇 i，计算簇 i 的成员属于类 j 的概率 $p_{ij} = m_{ij}/m_i$，其中 m_i 是簇 i 中对象的个数，而 m_{ij} 是簇 i 中类 j 的对象个数。使用类分布，用标准公式 $e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij}$ 计算每个簇 i 的熵，其中 L 是类的个数。簇集合的总熵用每个簇的熵的加权和计算，即 $e = \sum_{i=1}^{K} \frac{m_i}{m} e_i$，其中 K 是簇的个数，而 m 是数据点的总数。

- **纯度**：簇包含单个类的对象的另一种度量程序。使用前面的术语，簇 i 的纯度是 $p_i = \max_j p_{ij}$，而聚类的总纯度是 $purity = \sum_{i=1}^{K} \frac{m_i}{m} p_i$。

簇评估的总结

“聚类结构的验证是聚类分析中最困难和最令人沮丧的部分。

若没有在这个方向上的进一步努力，聚类分析将仍然是只有那些有经验和巨大勇气的真正信徒才能使用的魔法。”

Algorithms for Clustering Data, Jain and Dubes