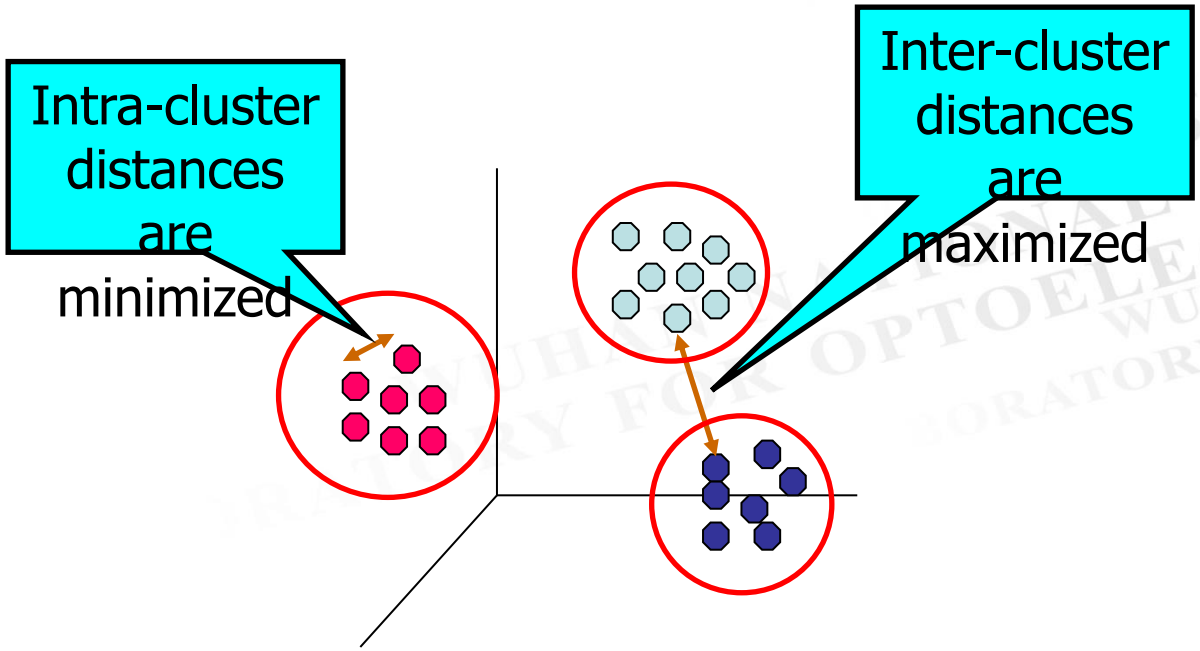
聚类分析：基本概念和算法

电子信息与通信学院 冯 缤
fengbin@hust.edu.cn

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
  - 组内对象相互之间是相似的（同质性），组间对象是不同的，同质性越大，组件差别越大，聚类就越好



什么是聚类分析

- 旨在理解的聚类
  - 为了理解和分析数据，将其划分成具有公共特性的对象组
  - 归类相关的文档方便浏览，归类具有相似功能的基因和蛋白质，归类具有相似价格波动的股票
- 旨在实用的聚类
  - 为了进一步的数据分析和数据处理技术的预处理
  - 数据压缩，数据汇总

什么不是聚类分析

- 监督式分类
  - 具有先验的类标号信息
- 简单的分割
  - 根据姓名的起始字母将学生分成不同的组

How many clusters?

Six Clusters

Two Clusters

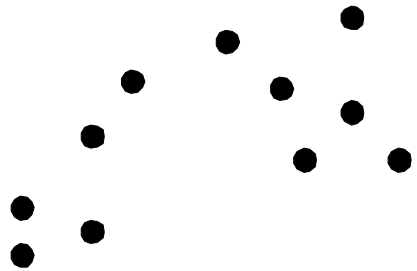
Four Clusters

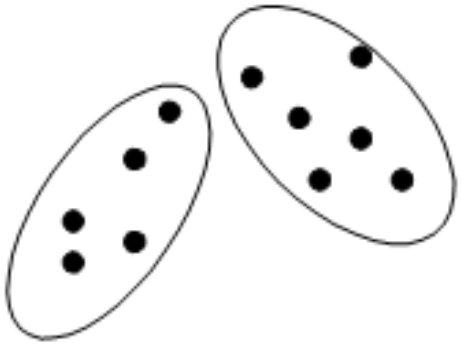
聚类类型

- 层次的与划分的
  - 嵌套的 vs. 非嵌套的
- 层次聚类是嵌套簇的集合，组织成树形，除叶结点外，树中每一个结点都是其子女的并
- 划分聚类简单的将数据对象集划分成不重叠的子集，使得每个数据对象恰在一个子集中

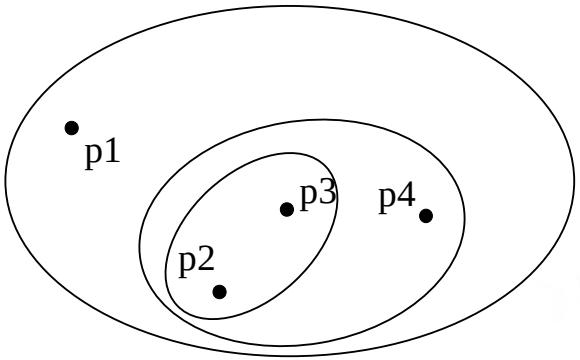
聚类类型

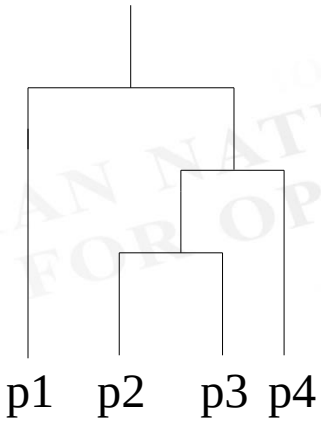
- 划分的





- 层次的





- 互斥的，重叠的和模糊的
- 互斥的(exclusive)：每个对象都被指派到单个簇
- 重叠的(overlapping)：将对象合理的同时指派到多个簇中
  - 模糊的(fuzzy clustering)：对象以一个0(绝对不属于)和1(绝对属于)之间的隶属权值属于每个簇
    - 模糊集

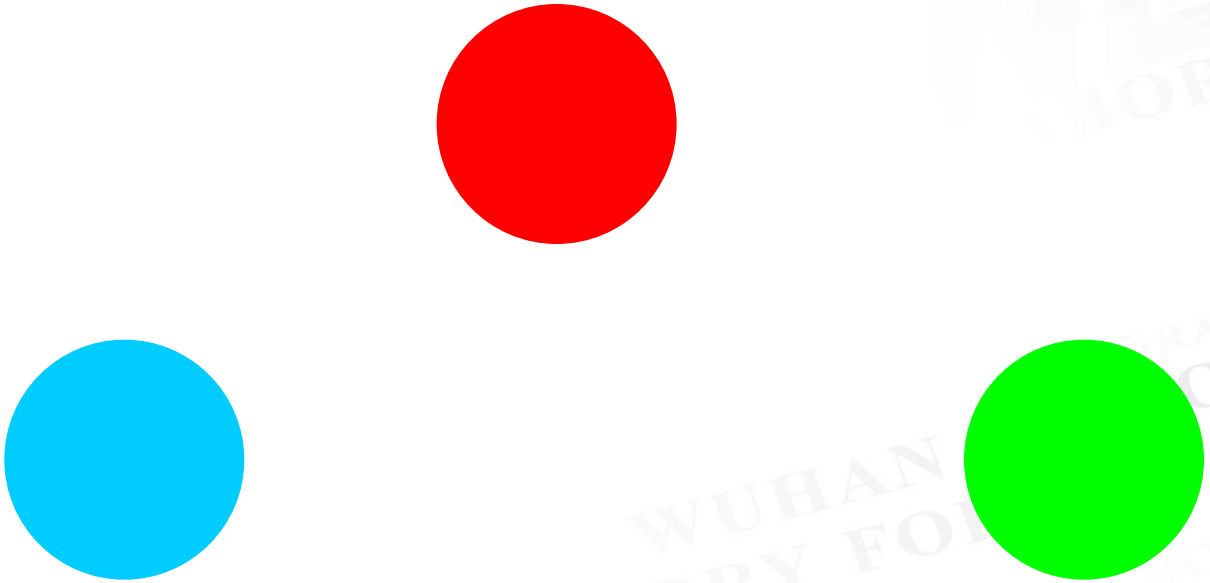
- 完全的与部分的
- 完全聚类(complete clustering)：将每个对象指派到一个簇
- 部分聚类不指派所有对象
  - 离群点，不感兴趣的事件

簇类型

- 明显分离的簇
- 基于原型的簇
- 基于图的簇
- 基于密度的簇
- 概念簇

明显分离的簇

- 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更近（或更相似）



3个明显分离的簇

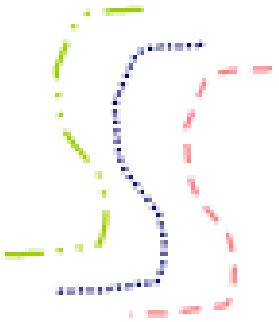
基于原型的簇

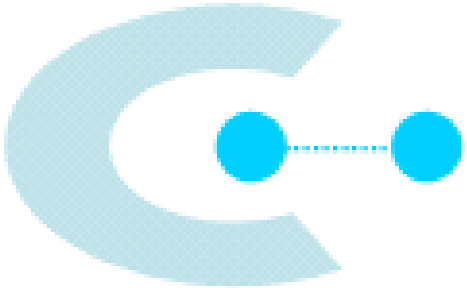
- 每个对象到定义该簇的原型的距离比到其他簇的原型的距离更近
  - 质心
  - 基于中心的簇

4个基于原型的簇

基于图的簇

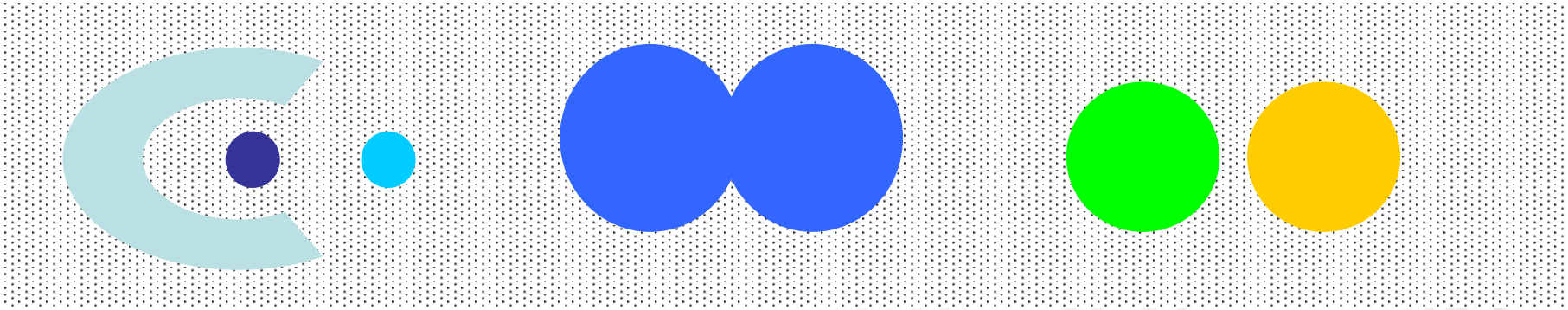
- 结点是对象，边代表对象之间的联系，簇定义为互相连通但不与组外对象连通的对象组
  - 每个对象到该簇某个对象的距离比到不同簇中任意点的距离更近
  - 基于邻近的簇





基于密度的簇

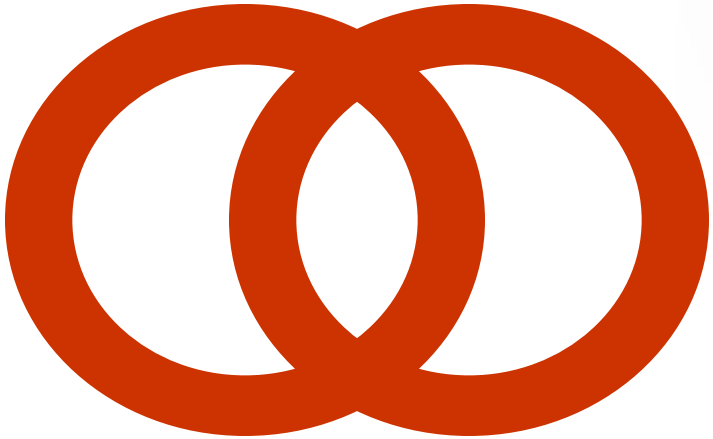
- 簇是对象的稠密区域，被低密度的区域环绕
  - 当具有噪声和离群点时，常常使用基于密度的簇定义



6个基于密度的簇

概念簇

- 簇定义为具有某种共同性质的对象的集合



2个重叠的环

聚类算法

- K均值
  - 基于原型的划分的聚类技术，试图发现用户指定个数的簇（由质心代表）
- 凝聚的层次聚类
  - 由多个单点簇重复合并，直到产生单个的包含所有点的簇
- DBSCAN
  - 基于密度的聚类算法，个数自动确定
  - 忽略噪声，不完全聚类

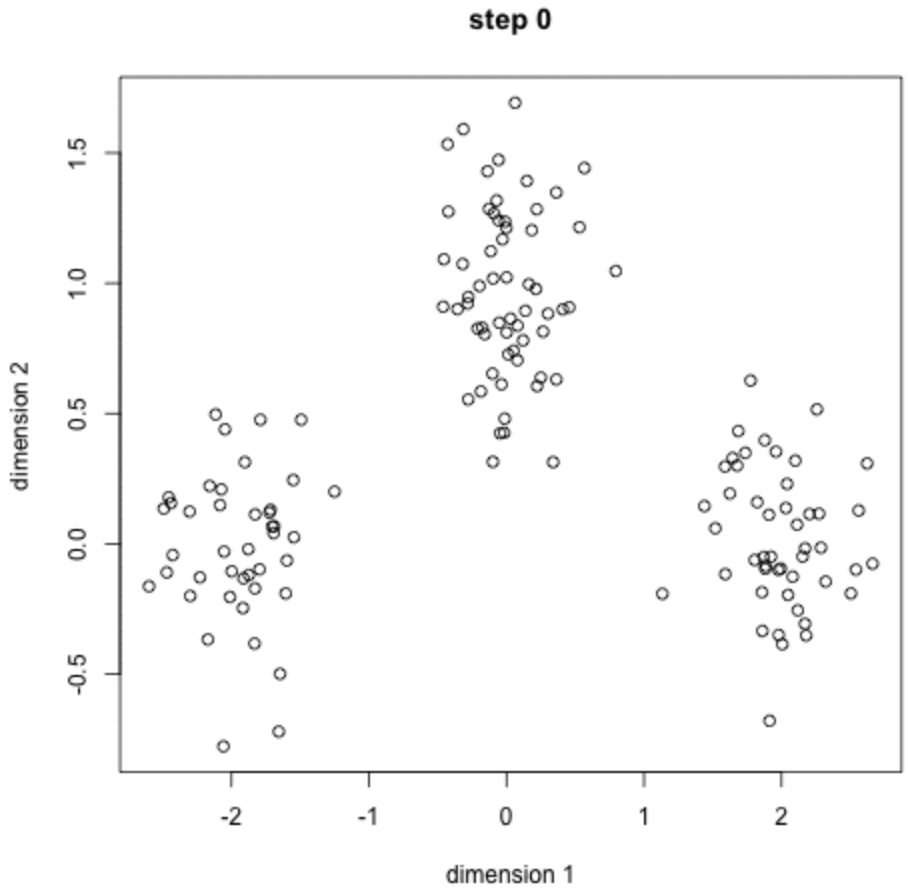
K均值聚类

- K均值用质心定义原型
  - 基于原型的单层划分
  - 每个聚类和一个质心点(中点)相关联
  - 每个点被指派到与之最接近的质心所属的类中
  - 聚类的数量(K)必须被指定

Algorithm 1 Basic K-means Algorithm.

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K均值聚类



K均值聚类

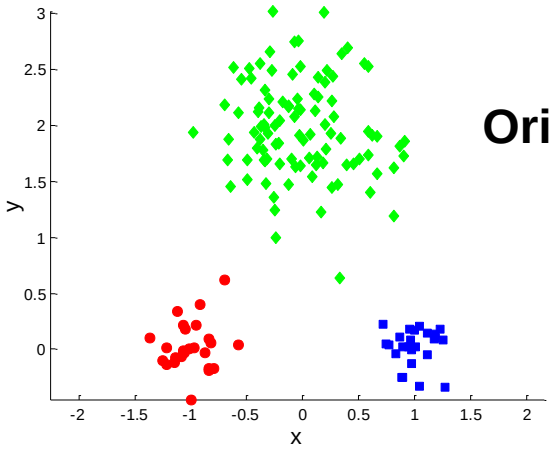
- 1. 指派点到最近的质心
  - 邻近度度量
  - 欧式距离，余弦相似性
- 2. 质心和目标函数

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

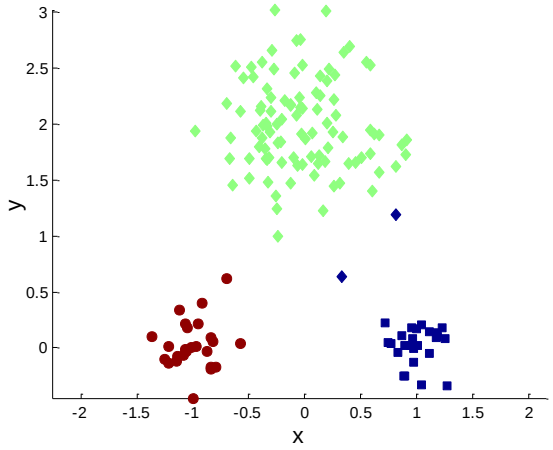
$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x$$

K均值聚类

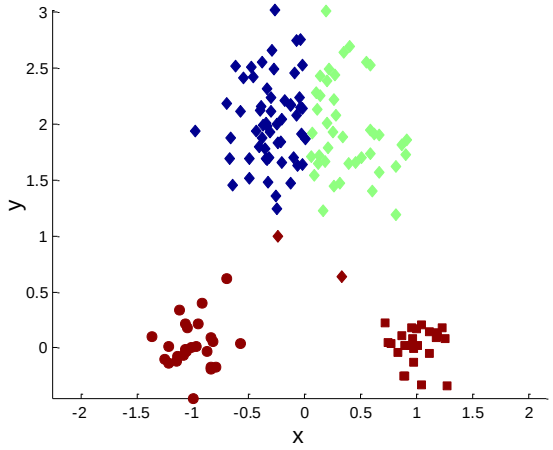
- 3. 初始选择质心
  - 通常随机选取初始质心
  - 常常产生不同的聚类结果



Original Points

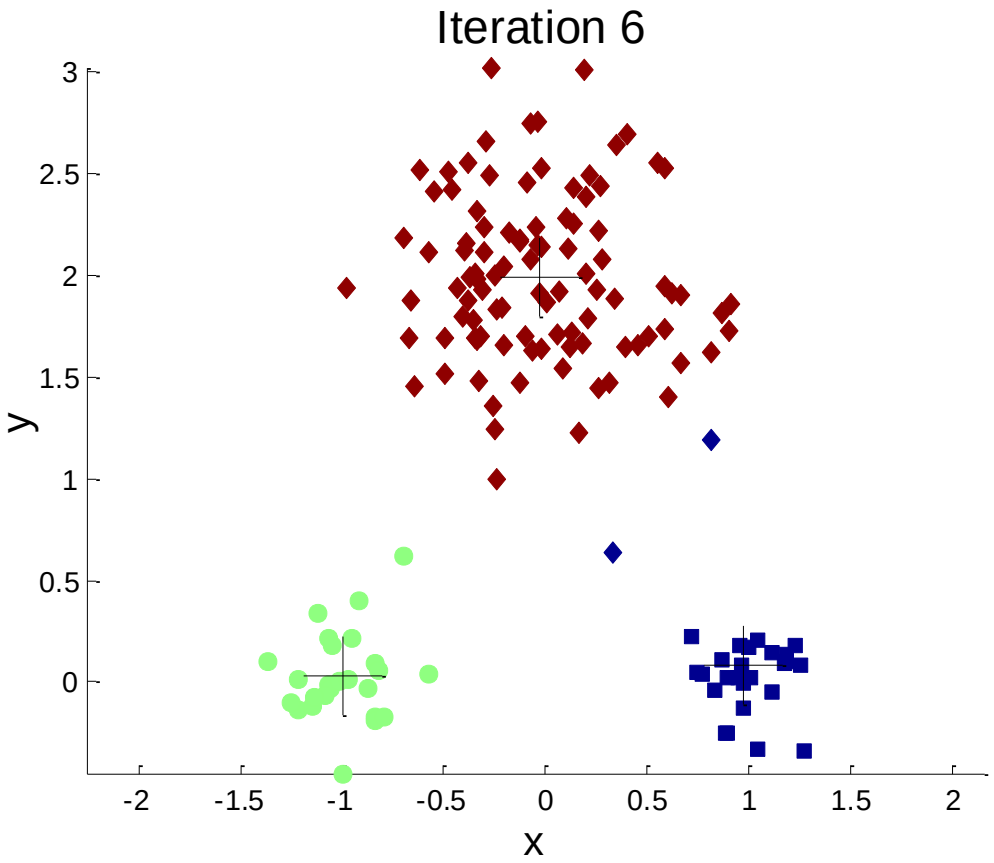


Optimal Clustering

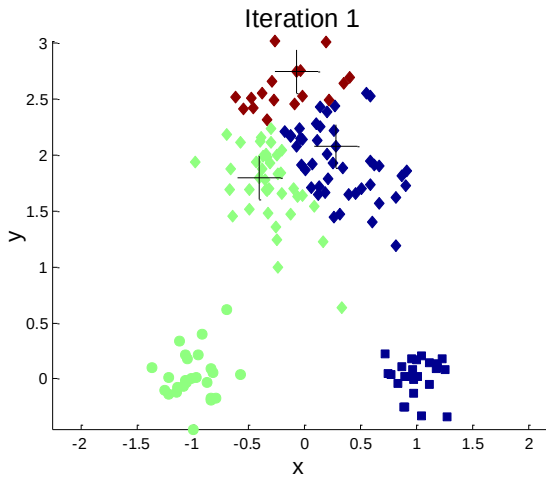


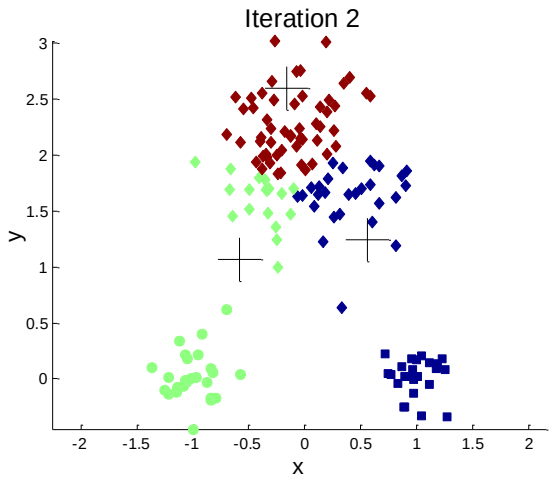
Sub-optimal Clustering

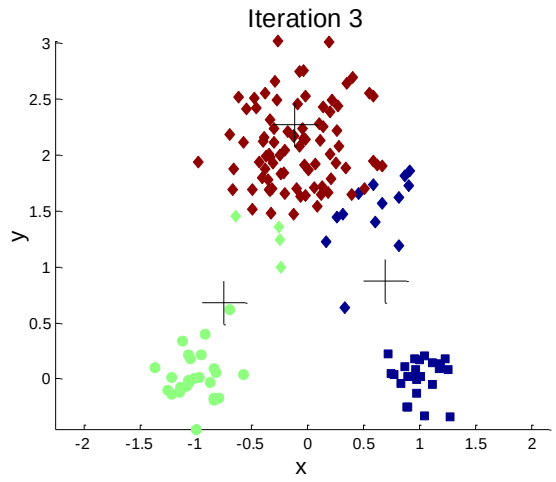
K均值聚类

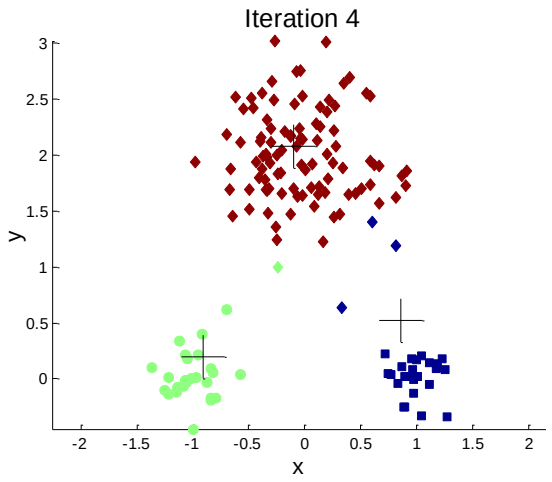


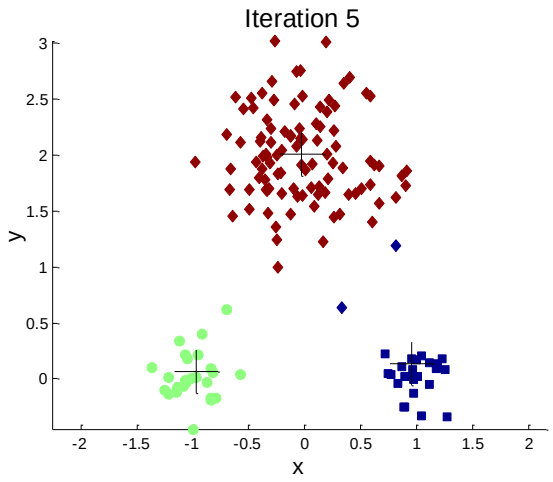
K均值聚类

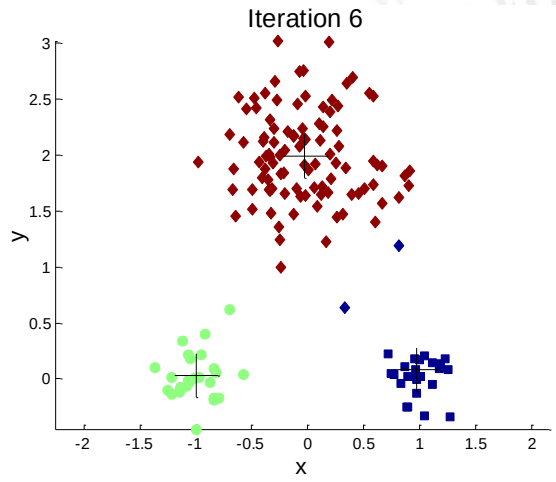

Iteration 1
x
y


Iteration 2
x
y

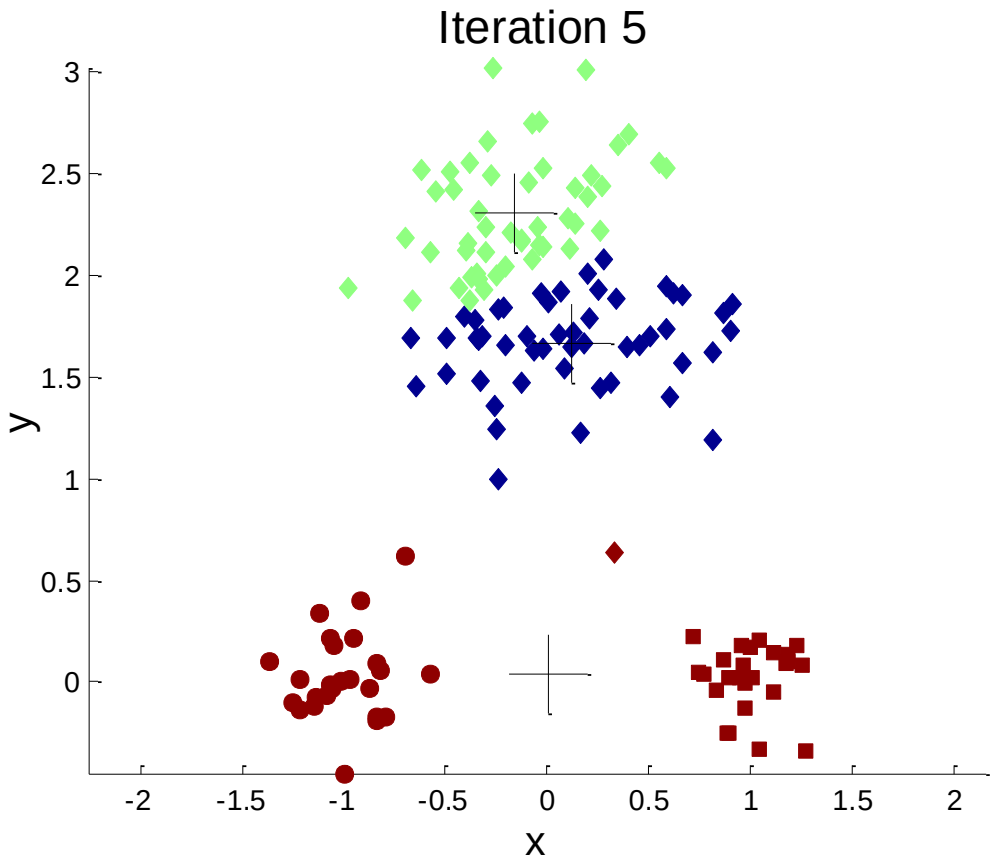

Iteration 3
x
y


Iteration 4
x
y

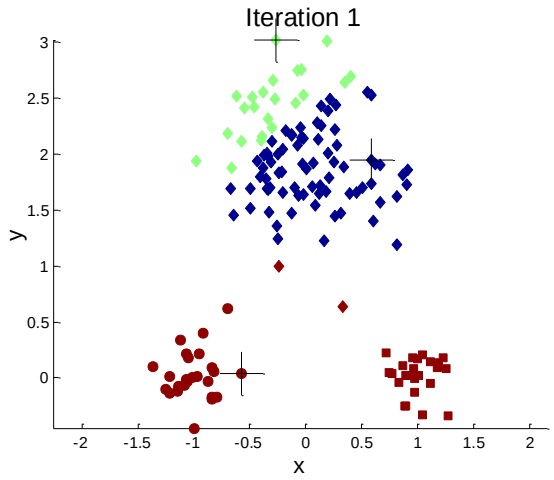

Iteration 5
x
y

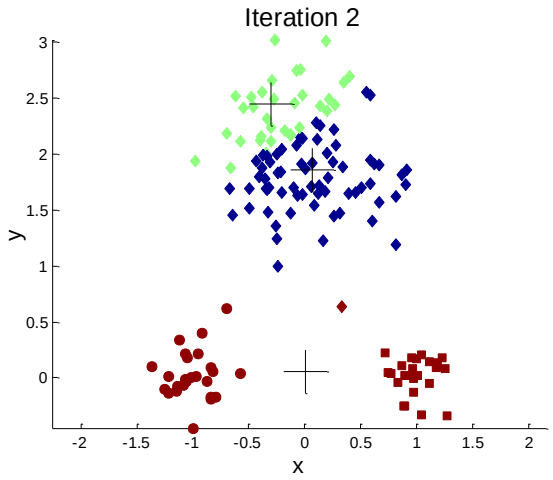

Iteration 6
x
y

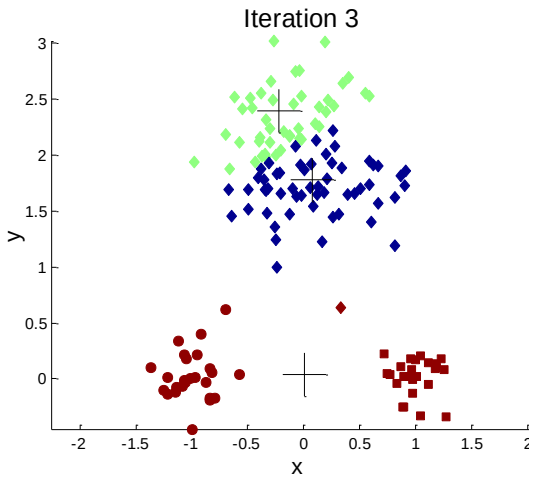
K均值聚类

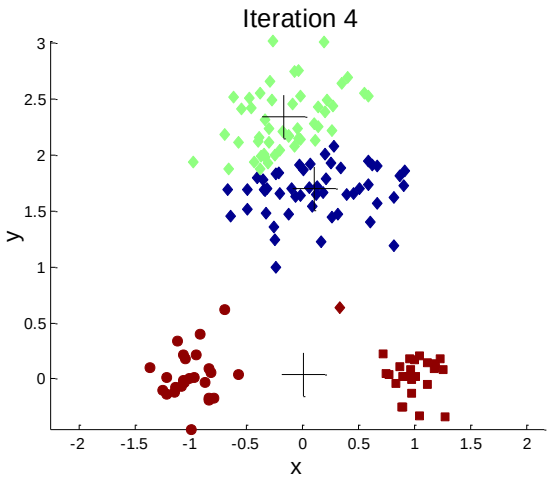


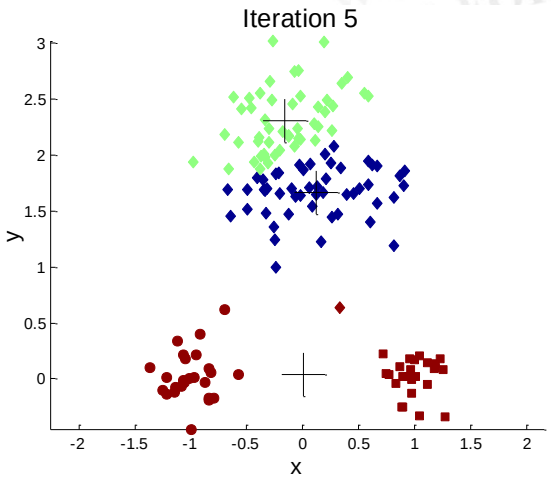
K均值聚类


Iteration 1
x
y

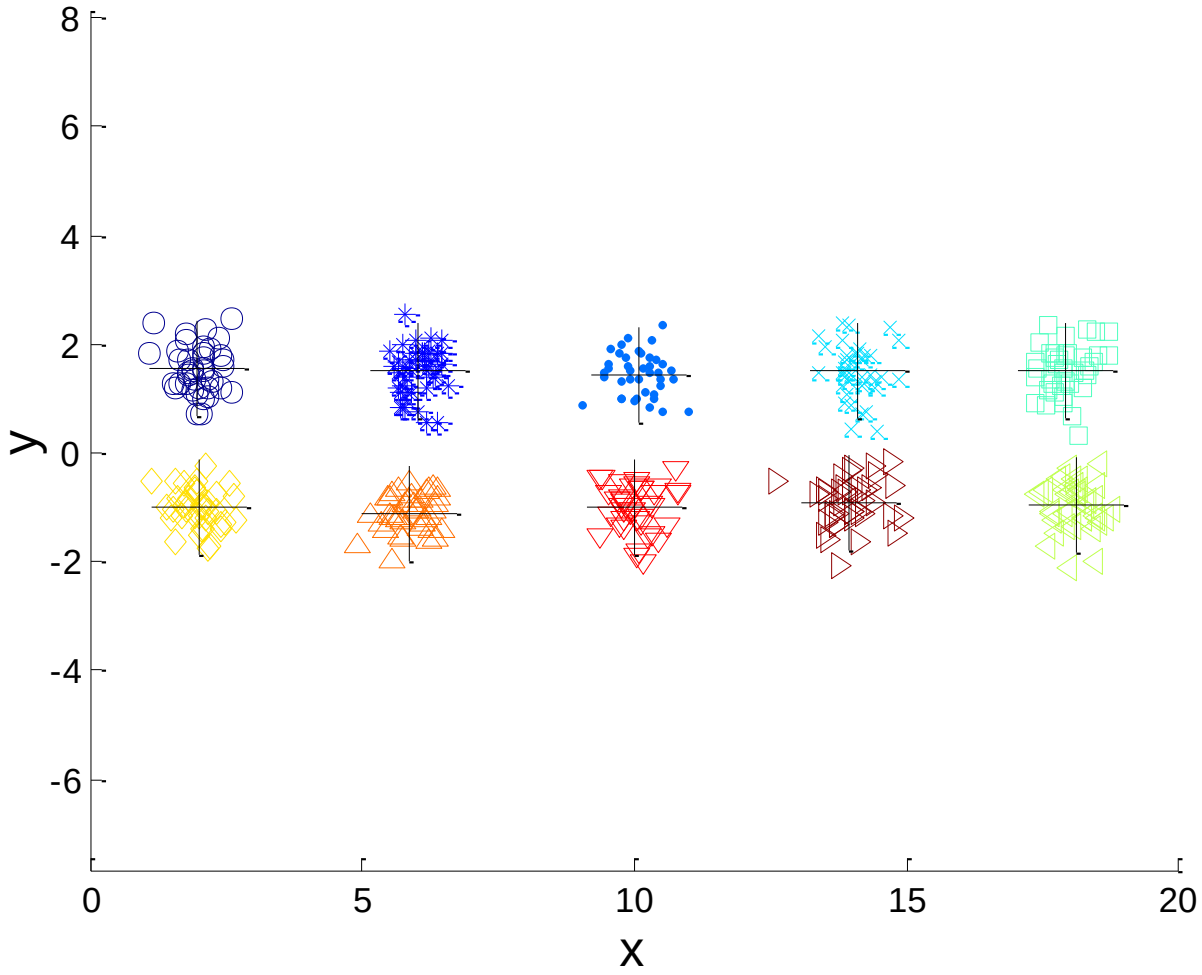

Iteration 2
x
y


Iteration 3
x
y

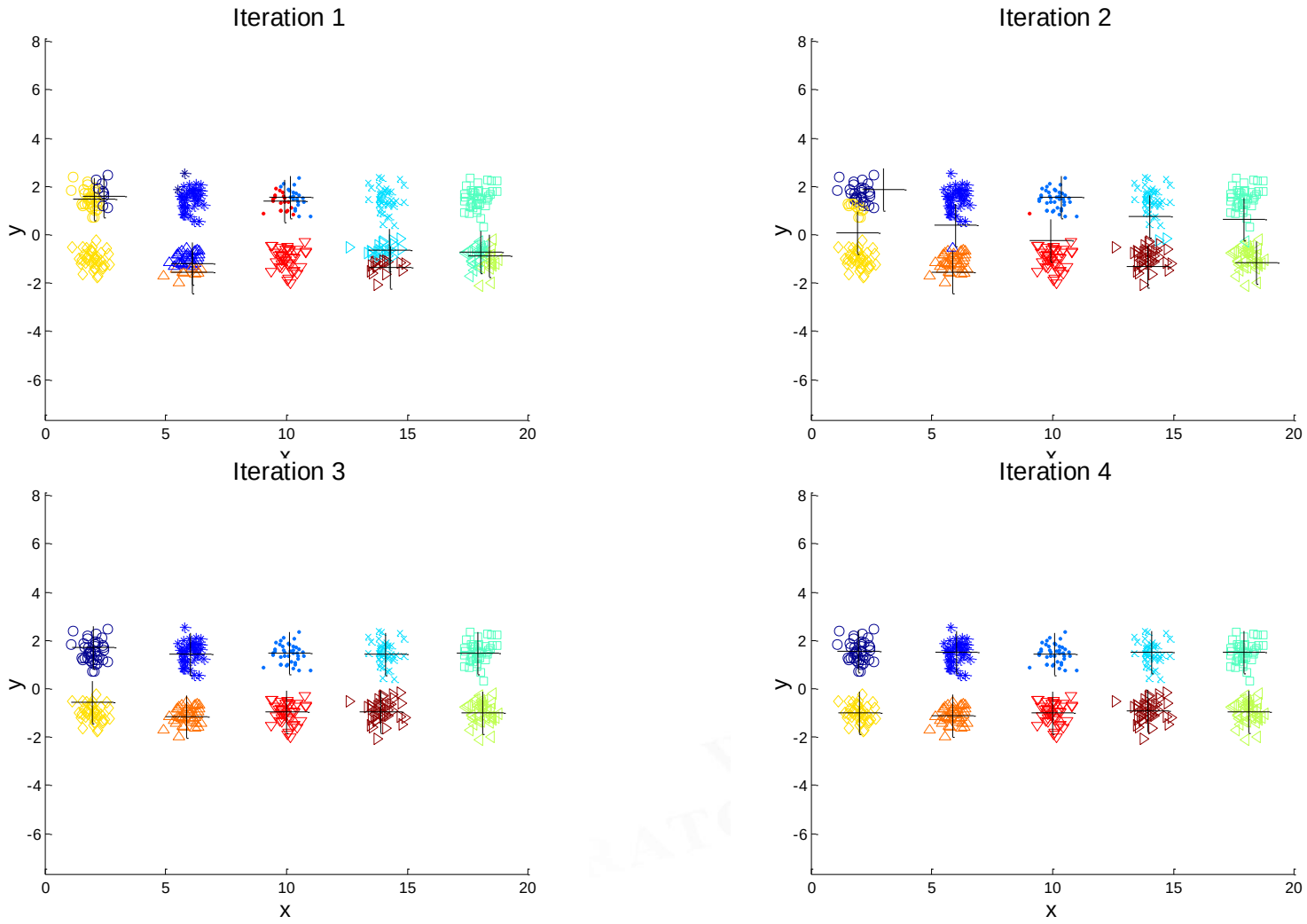

Iteration 4
x
y


Iteration 5
x
y

K均值聚类



K均值聚类



K均值聚类

- 随机初始化的局限
- 如果有K个自然簇，那么为每个簇都初始化一个质心的概率是很小的
  - 当K很大时尤其如此
  - 如果每个簇具有同样的大小n，则

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K! n^K}{(Kn)^K} = \frac{K!}{K^K}$$

  - 当K=10，则概率为$10!/10^{10} = 0.00036$

K均值聚类

- 对原数据抽样，使用层次聚类技术聚类，提取K个簇，并用这些簇的质心作为初始质心
  - 取样样本较小
  - K相对于样本大小较小
- 选择离已经选取过的初始质心最远的点
  - 得到随机的，散开的初始质心集合
  - 会选中离群点，且开销非常大
- 二分K均值
  - 对初始化问题不太敏感

K均值聚类

- 处理空簇
  - 当所有的点在指派步骤都未分配到某个簇时发生
  - 需要某种策略来选择替补质心
  - 1. 选择一个距离当前任何质心最远的点
    - 消除当前对总平方误差影响最大的点
  - 2. 从具有最大SSE的簇中选择一个替补质心
    - 将簇分裂并降低聚类的总SSE
  - 如果有多个空簇，则过程重复多次

K均值聚类

- 离群点
  - 使用平方误差时，离群点会过度影响所发现的簇
    - 簇的质心可能不如没有离群点时那样有代表性
  - 在可能的条件下，提前删除离群点
    - 某些情况下，离群点是感兴趣的事件，不能删除
  - 也可以在后处理时识别离群点

K均值聚类

- 后处理
- 通过增加簇个数来降低总的SSE
  - 分裂一个簇：选择具有最大SSE的簇
  - 引进一个新的质心：选择离所有簇质心最远的点
- 通过减少簇个数来最小化总的SSE增长
  - 合并两个簇：选择质心最接近的两个簇或使总SSE增加最少的簇

K均值聚类

- 增量的更新质心
  - 在点到簇的每次指派后，增量的更新质心，而不是在所有点都指派到簇中之后才更新质心
  - 每步需要零次或两次质心更新
    - 留在原簇或转移到新簇
  - 导致次序依赖性
  - 开销更大

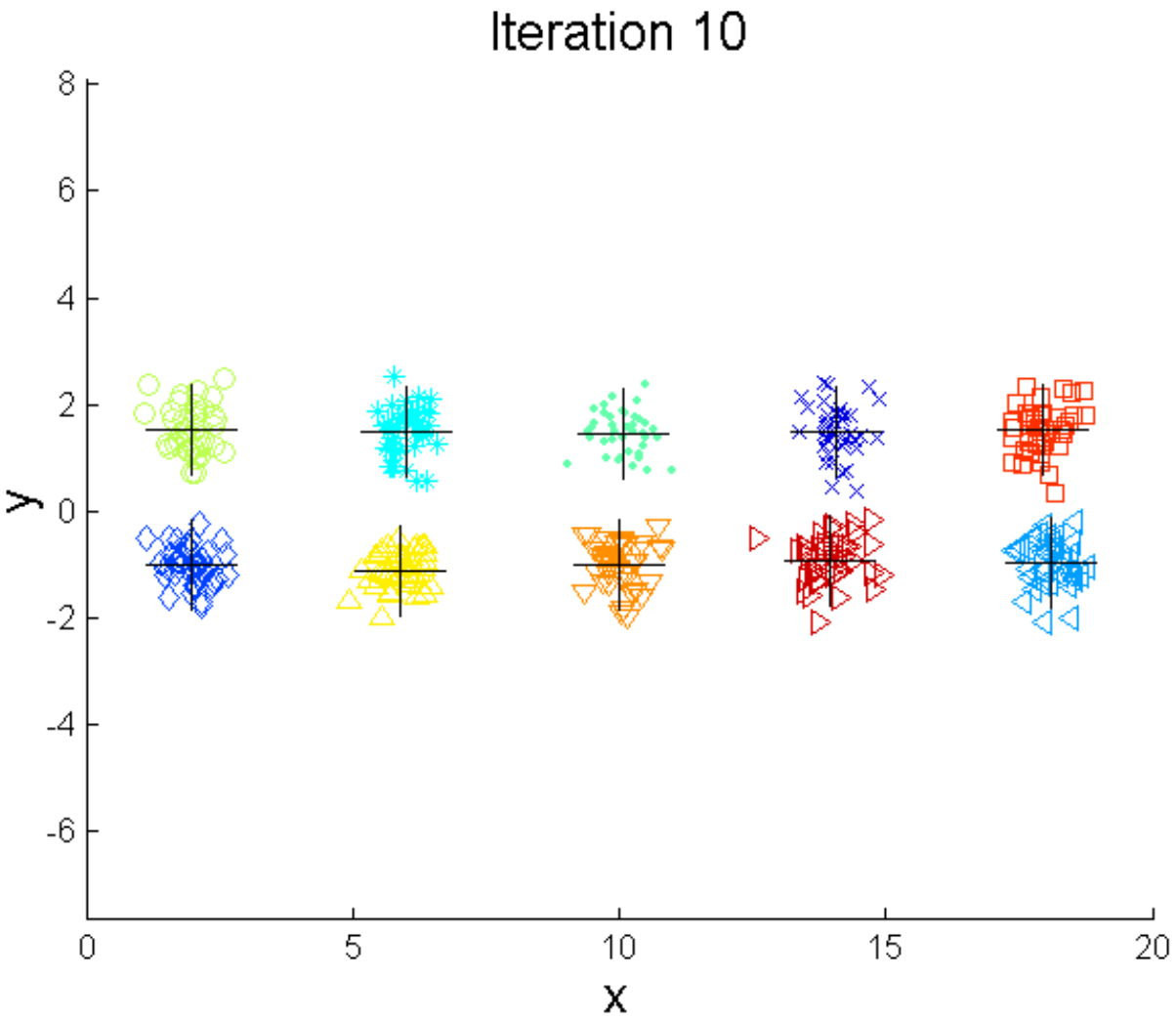
K均值聚类

- 二分K均值：基本K均值算法的直接扩充
  - 将所有的点集合分裂成两个簇，从中选择一个继续分裂，直到产生K个簇为止

```
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
4:        {从选定的簇进行多次二分“试验”}
5:        for i=1 to 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
9:        将这两个簇添加到簇表中
10: until 簇表中包含 K 个簇
```

待分裂的簇有许多不同的选择方法：最大的簇，最大SSE的簇等等，不同的选择导致不同的簇

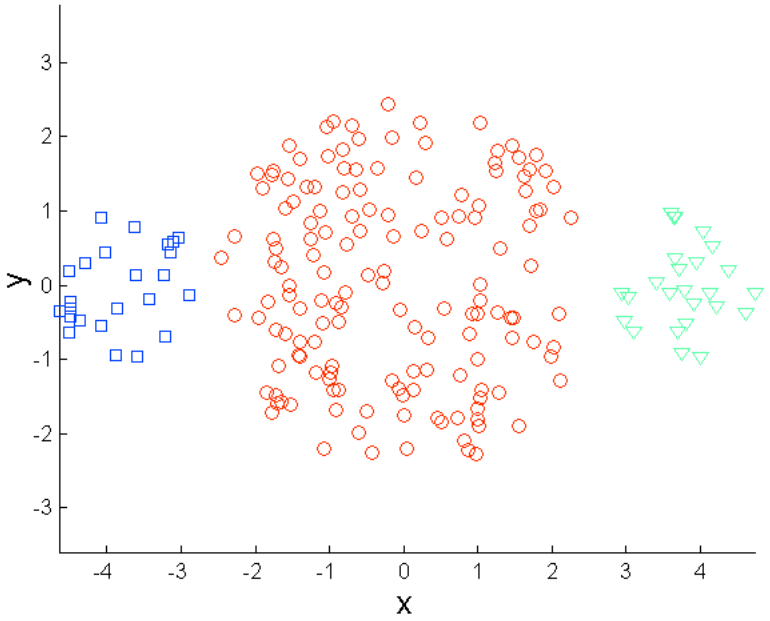
K均值聚类



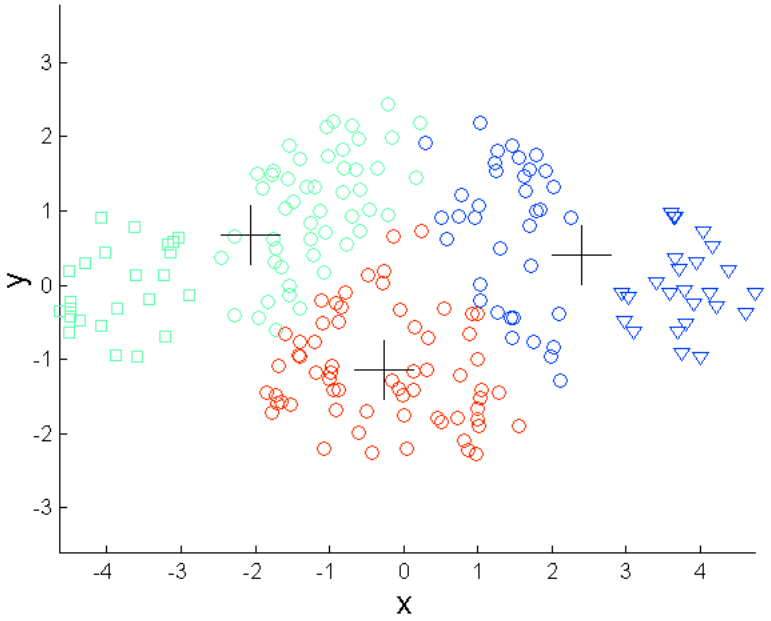
K均值聚类

- 在下列情况时，K均值算法很难检测到“自然的”簇
  - 具有不同尺寸
  - 具有不同密度
  - 具有非球形形状

K均值聚类



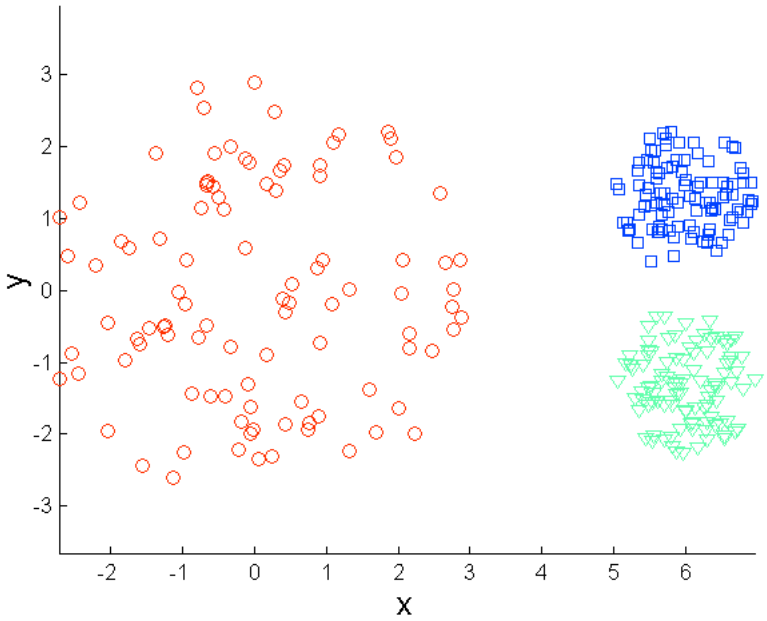
Original Points



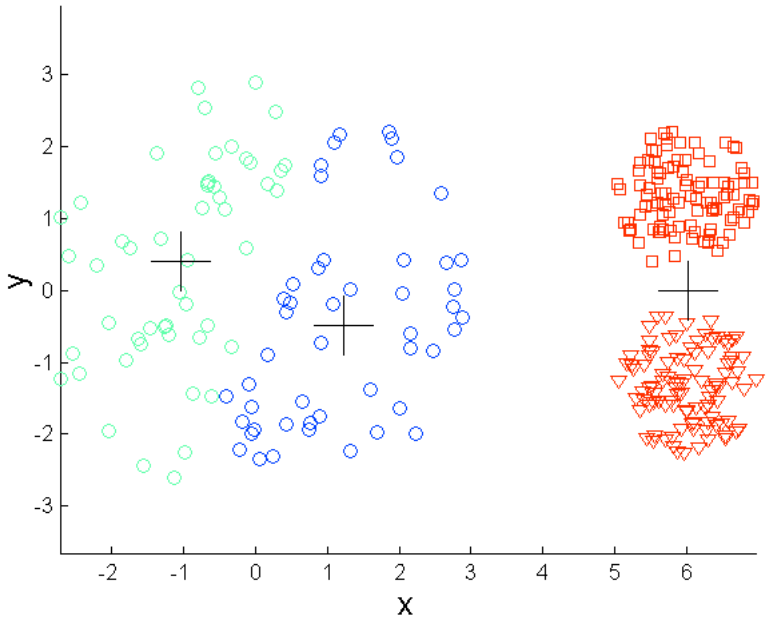
K-means (3 Clusters)

不同尺寸

K均值聚类



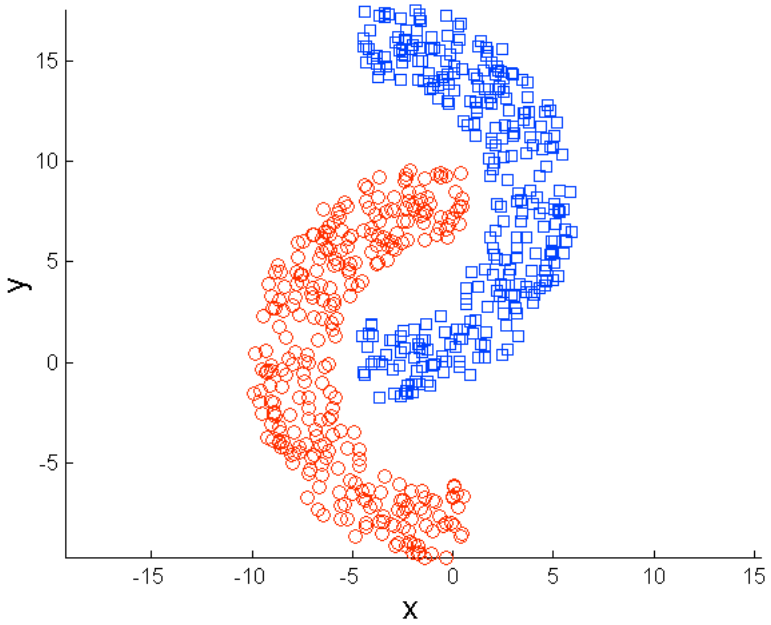
Original Points



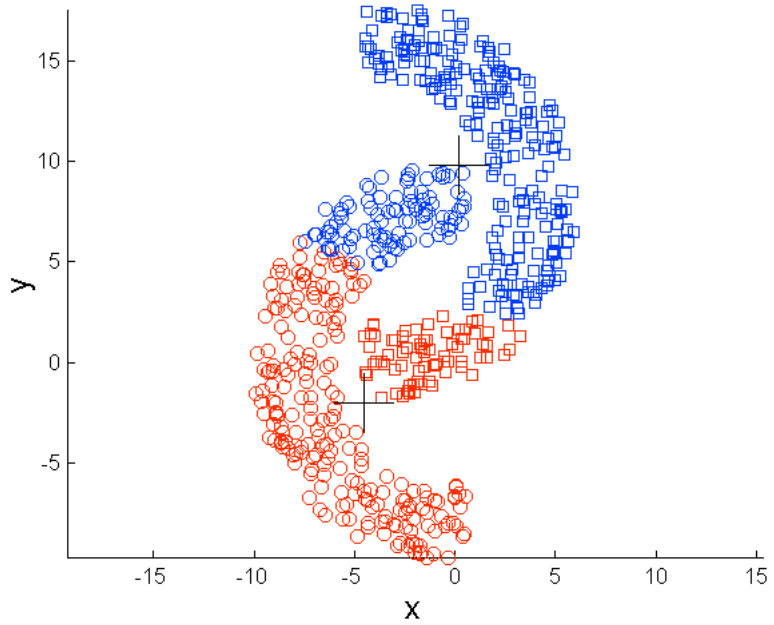
K-means (3 Clusters)

不同密度

K均值聚类



Original Points

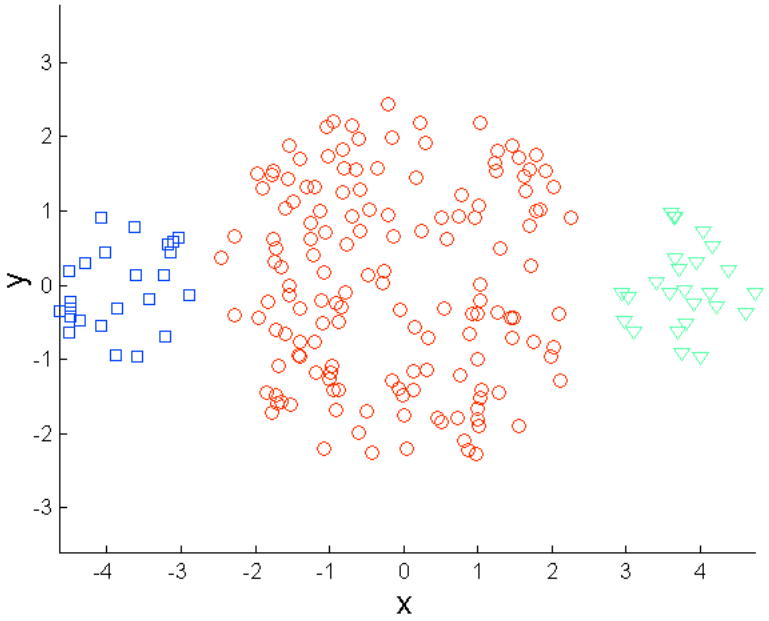


K-means (2 Clusters)

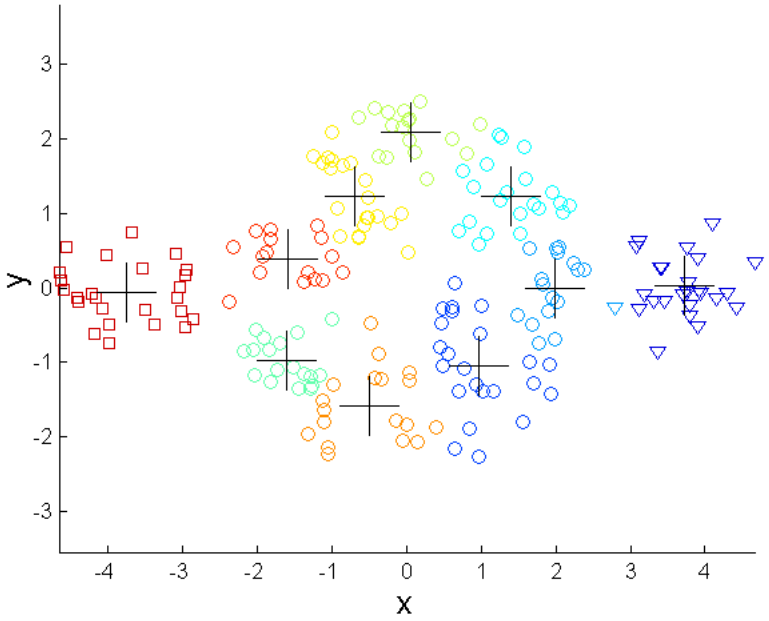
非球形

K均值聚类

- 一个解决方法是使用更多的簇

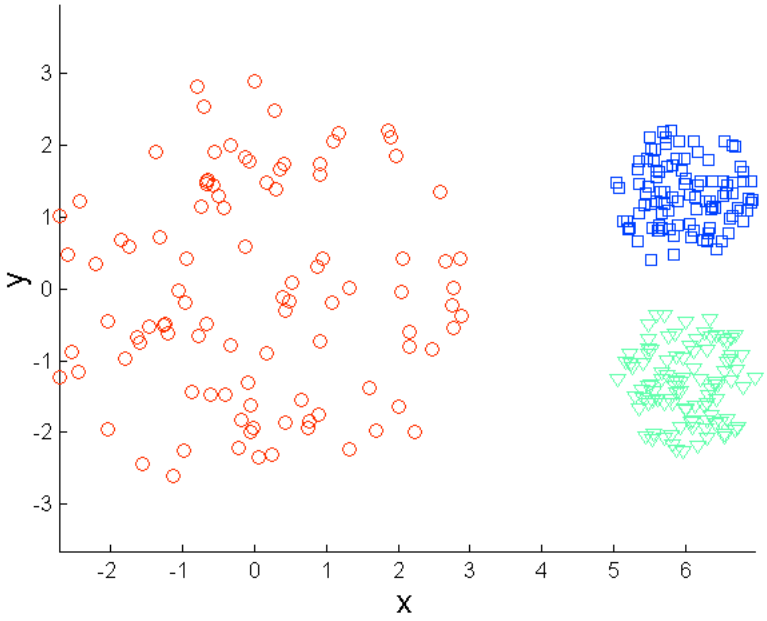


Original Points

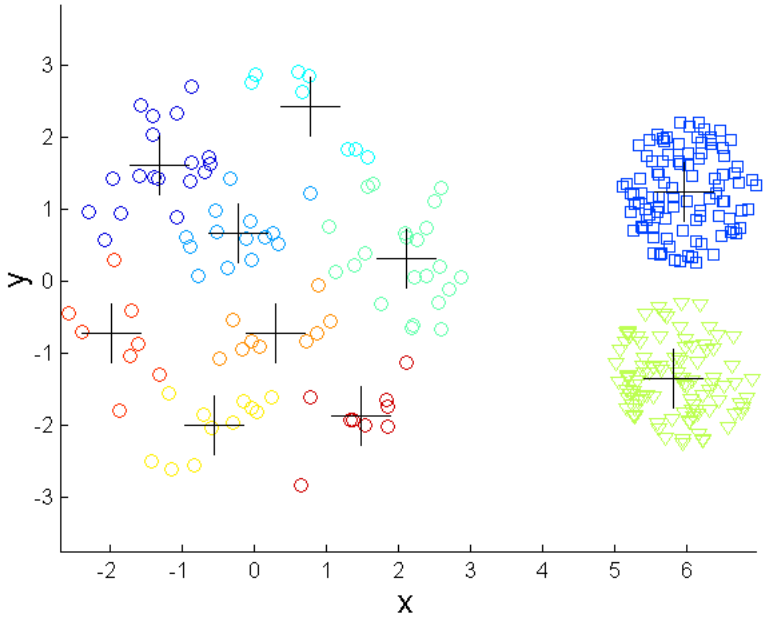


K-means Clusters

K均值聚类

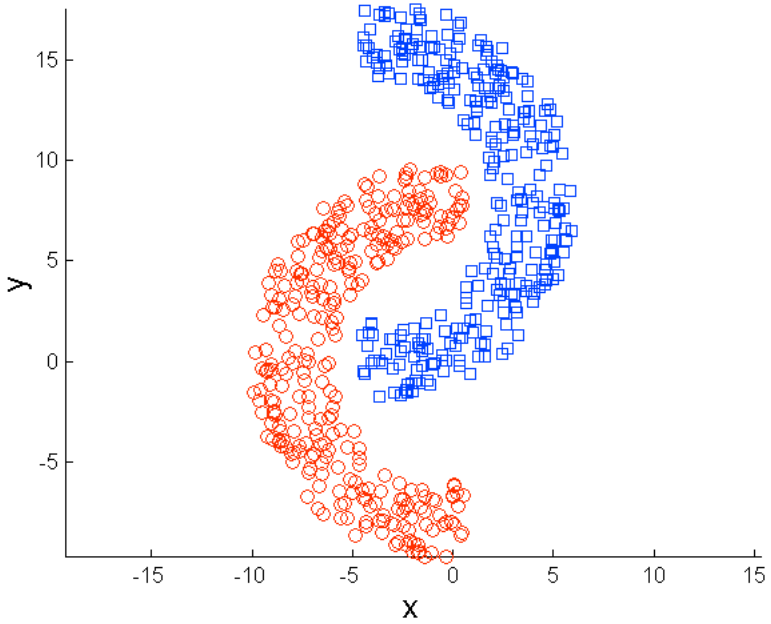


Original Points

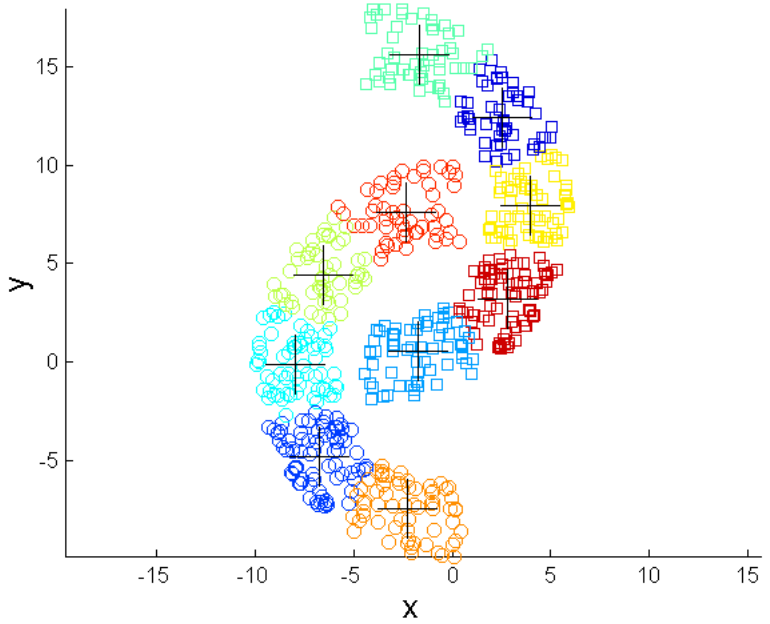


K-means Clusters

K均值聚类



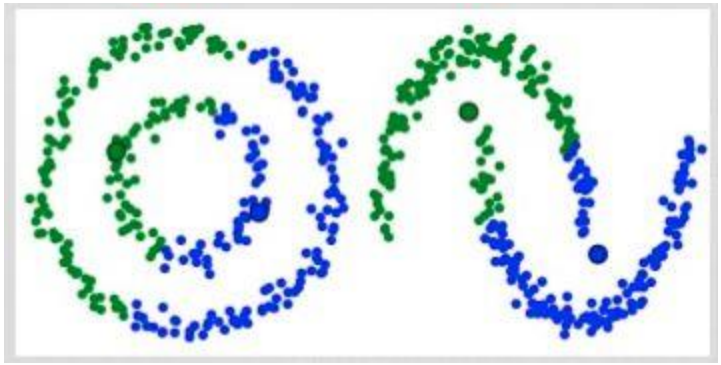
Original Points



K-means Clusters

高斯混合模型（GMM）聚类

- K 均值的一个主要缺点是它只简单使用了群集的平均值作为中心

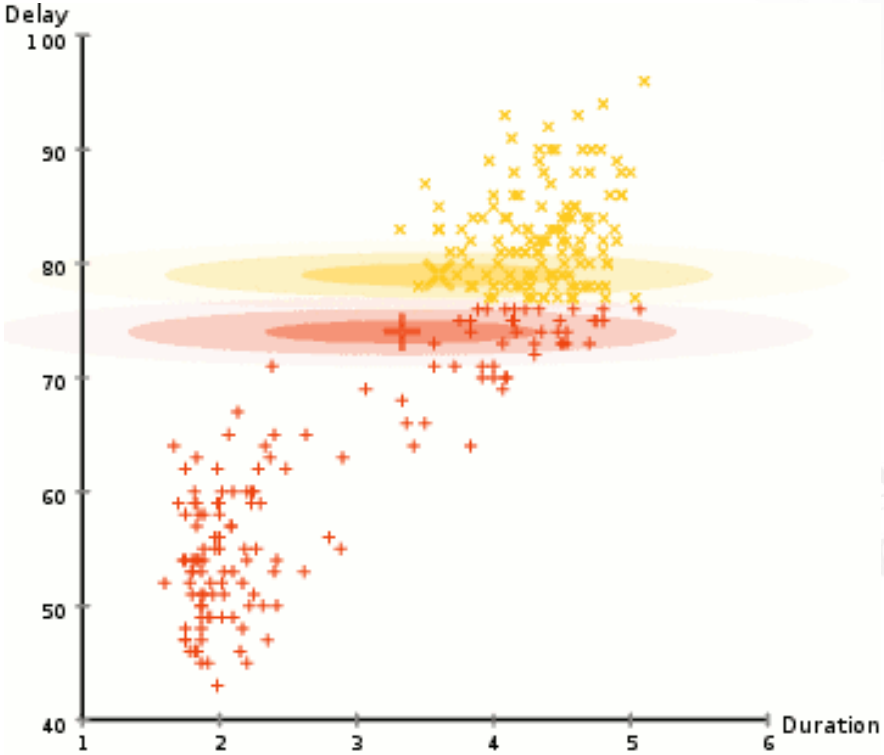


高斯混合模型（GMM）聚类

- GMM比 K 均值更具灵活性
- 假设数据点呈高斯分布，采用两个参数描述集簇的形状：平均值和标准偏差
  - 可以适应任何类型的椭圆形状
- 采用最大期望（EM）的优化算法求解每个集簇的高斯参数

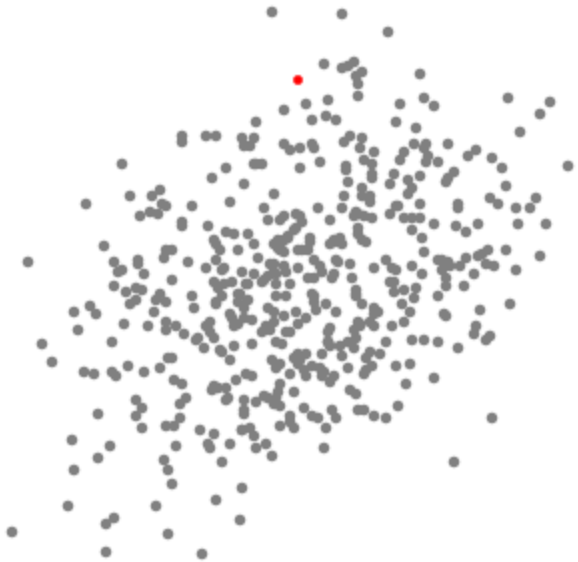
高斯混合模型（GMM）聚类

- K 均值是 GMM 的一种特殊形式



Mean Shift聚类

- 基于滑动窗口的聚类算法
  - 均值漂移
- 找到数据点的密集区域



Mean Shift聚类

- 均值漂移的基本形式
- 给定d维空间的n个数据点集X，那么对于空间中的任意点x的mean shift向量

$$M_h = \frac{1}{K}\sum_{x_i \epsilon S_k}(x_i - x)$$

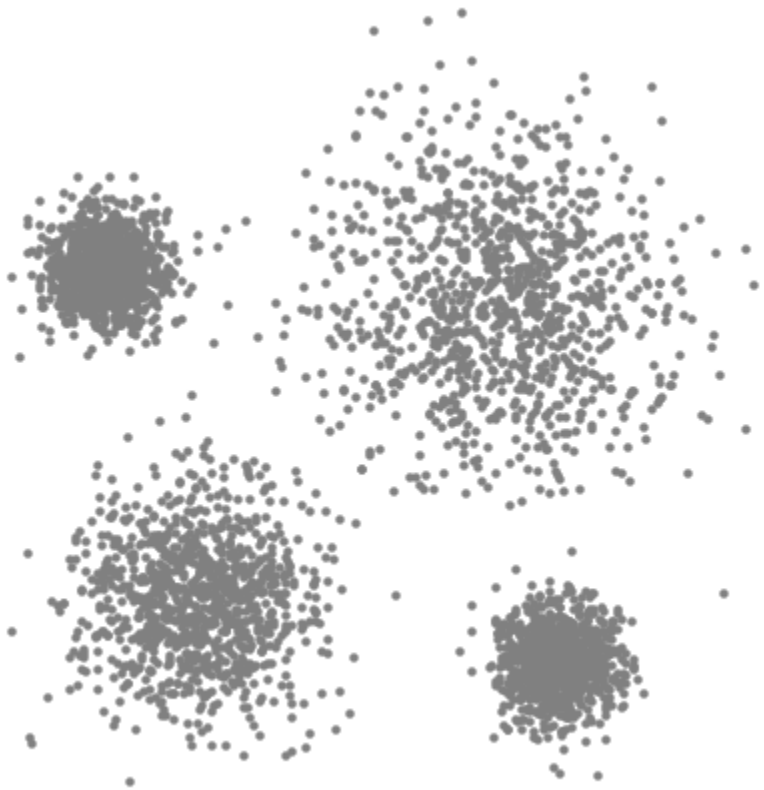
$x = x + M_h$

Mean Shift聚类

- 算法流程：
- 1、在未被标记的数据点中随机选择一个点作为中心center
- 2、找出离center距离在bandwidth之内的所有点，记做集合M，认为这些点属于簇c
- 3、以center为中心点，计算集合M中所有点到它的漂移向量
- 4、center = center+shift
- 5、重复步骤2、3、4，直到收敛，记住此时的center
- 6、如果收敛时当前簇c的center与已经存在的某个簇c2中心的距离小于阈值，则将c2和c合并。否则，把c作为新的聚类

Mean Shift聚类

- 下图给出了所有滑动窗口的收敛过程
- 每个黑点表示滑动窗口的质心，每个灰点是数据点

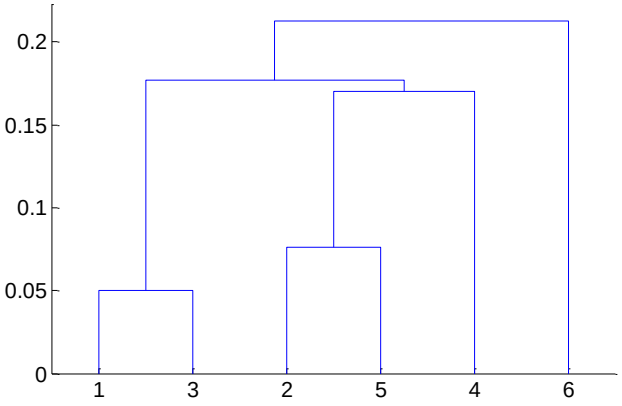


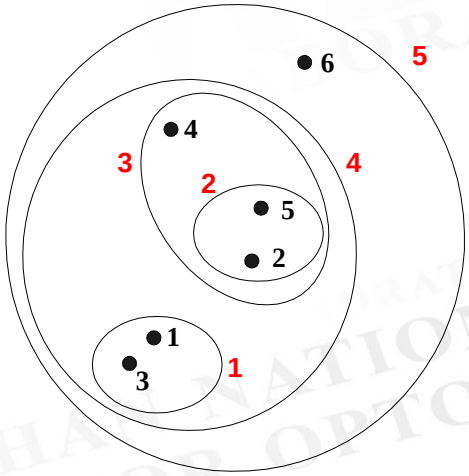
Mean Shift聚类

- 1. 不需要选择聚类数量
- 2. 聚类中心向最大密度点趋近比较直观，易于解释
- 3. 窗口大小（bandwidth）的选择非常重要，对结果影响较大

凝聚层次聚类

- 生成一系列嵌套的簇
- 可以用树状图表示





凝聚层次聚类

- 两种产生层次聚类的基本方法
  - 自下而上或自上而下
- 凝聚的
  - 从点作为个体簇开始，每一步合并两个最接近的簇
  - 需要定义簇的邻近性概念
- 分裂的
  - 从包含所有点的一个簇开始，每一步分裂一个簇，直到仅剩下单点簇
  - 分裂的原则

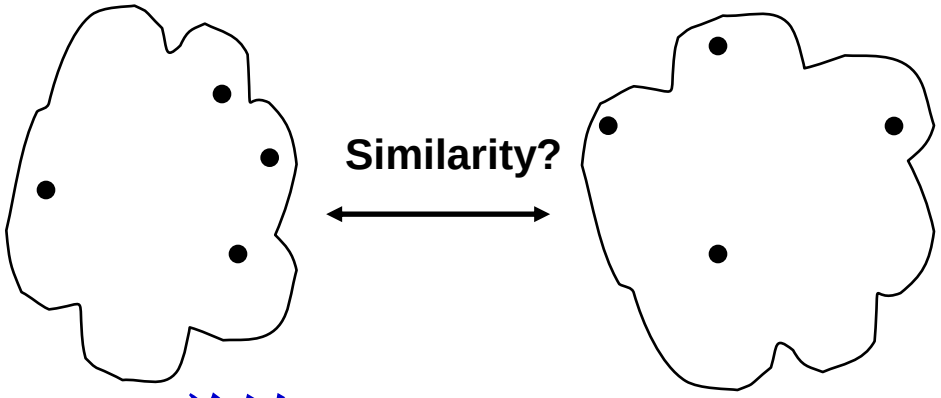
凝聚层次聚类

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: 更新邻近性矩阵，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

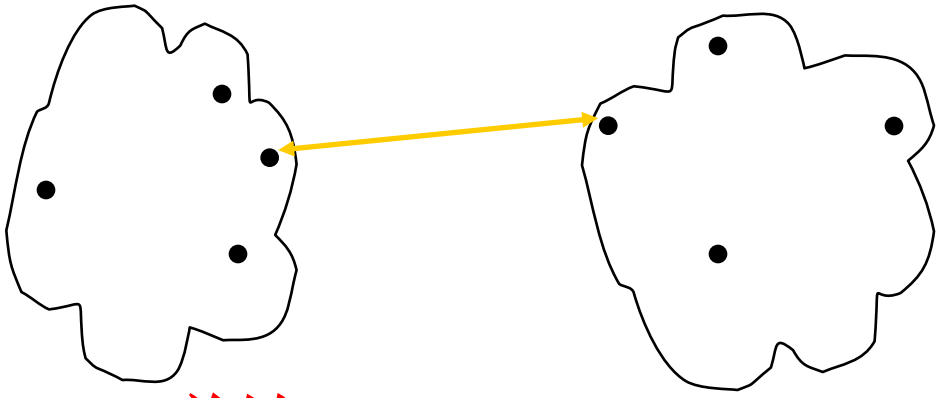
- 通过在合适的层次终止聚类的过程来获得希望的簇的数量
- 关键是两个簇之间的邻近度计算

凝聚层次聚类



- MIN(单链)
- MAX (全链)
- 组平均
- 质心距离
- Ward方法
  - 使用合并两个簇导致的SSE增加来度量两个簇之间的邻近性

凝聚层次聚类

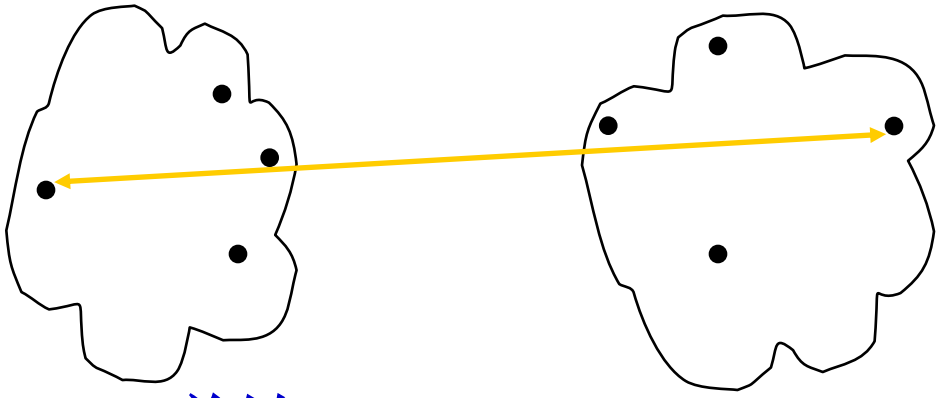


- MIN(单链)
- MAX (全链)
- 组平均
- 质心距离
- Ward方法
  - 使用合并两个簇导致的SSE增加来度量两个簇之间的邻近性

不同簇的两个最近的点之间的邻近度

不同的结点子集中两个结点之间的最短边

凝聚层次聚类

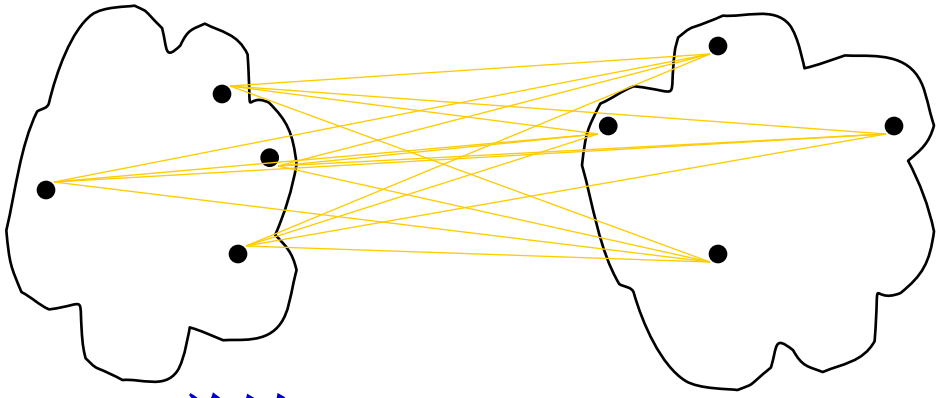


- MIN(单链)
- MAX (全链)
- 组平均
- 质心距离
- Ward方法
  - 使用合并两个簇导致的SSE增加来度量两个簇之间的邻近性

不同簇中两个最远的点之间的邻近度作为簇的邻近度

不同的结点子集中两个结点之间的最长边

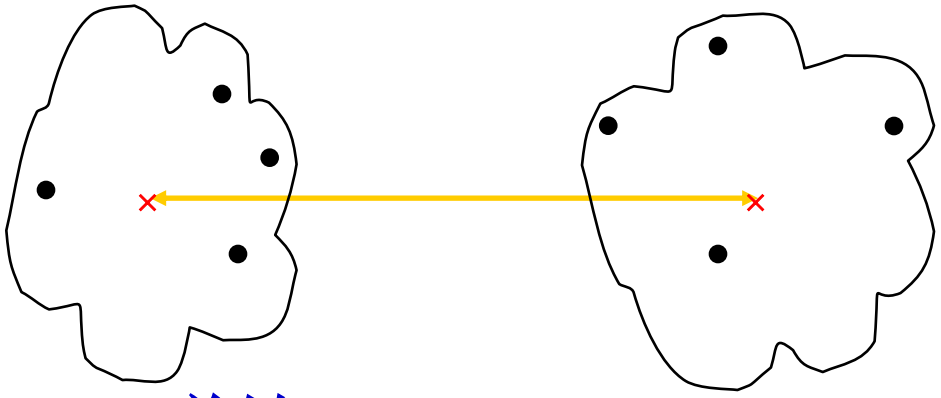
凝聚层次聚类



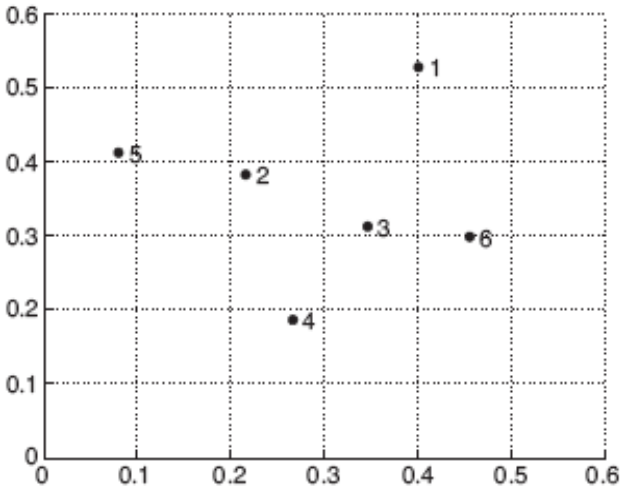
- MIN(单链)
- MAX (全链)
- 组平均
- 质心距离
- Ward方法
  - 使用合并两个簇导致的SSE增加来度量两个簇之间的邻近性

取自不同簇的所有点对的平均逐对邻近度

凝聚层次聚类



- MIN(单链)
- MAX (全链)
- 组平均
- 质心距离
- Ward方法
  - 使用合并两个簇导致的SSE增加来度量两个簇之间的邻近性



6个二维点的集合

Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

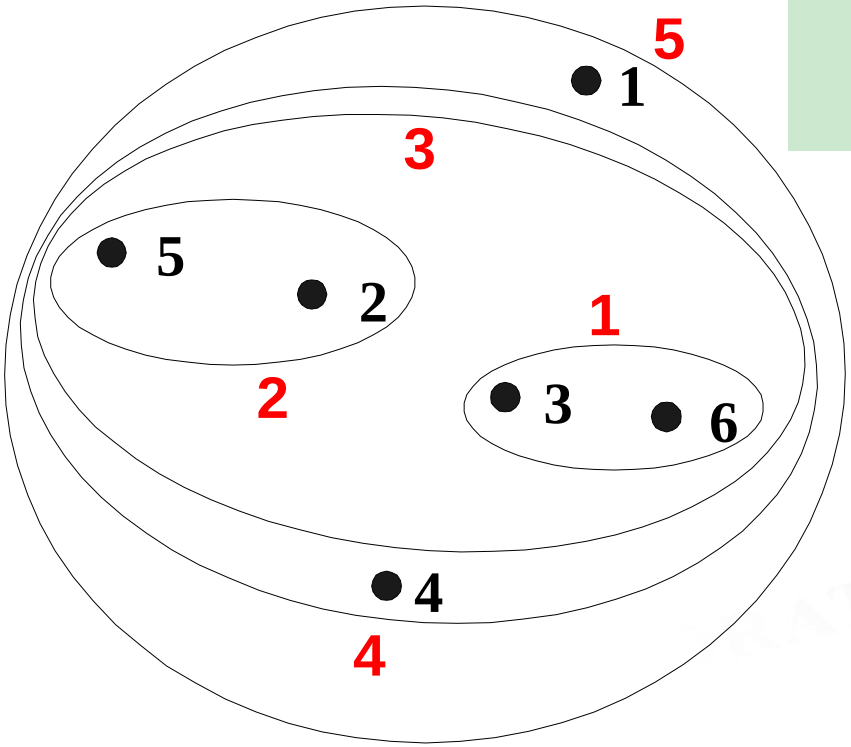
6个点的XY坐标

	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

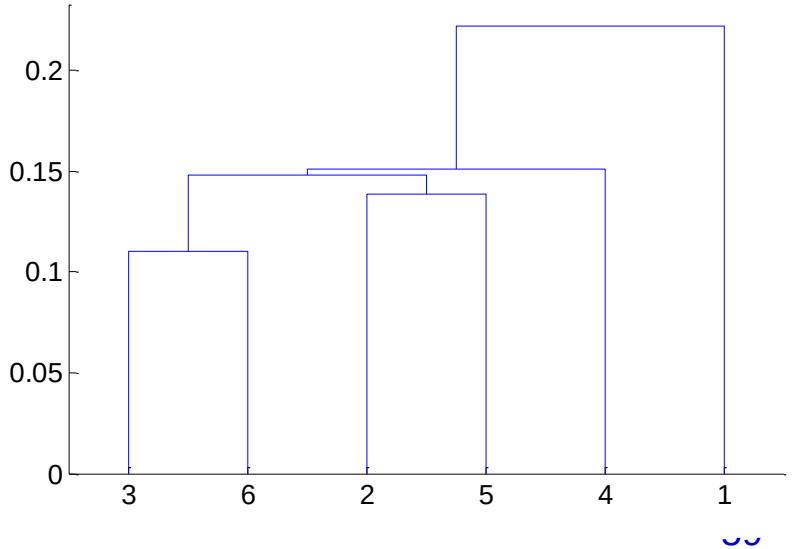
6个点的欧几里德距离矩阵

凝聚层次聚类

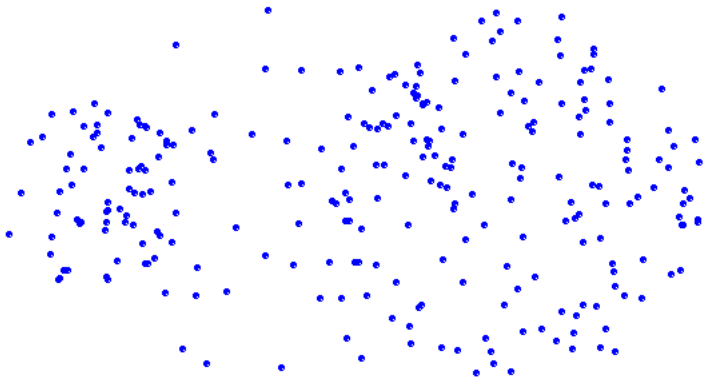
- 单链聚类
  - 对噪声和离群点很敏感



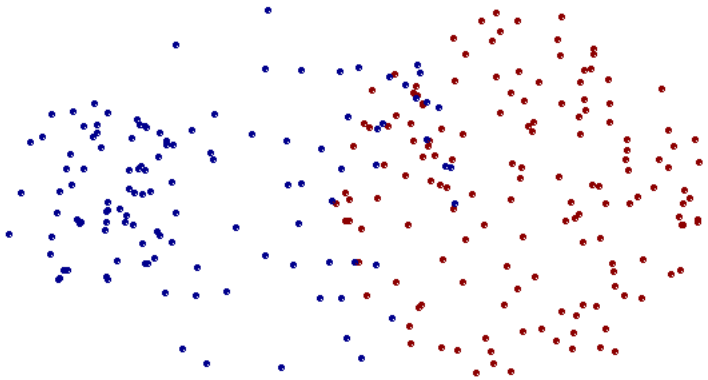
$$\begin{aligned}\text{Dist}(\{3,6\},\{2,5\}) &= \min(\text{dist}(3,2),\text{dist}(6,2),\text{dist}(3,5),\text{dist}(6,5))\\ &= \min(0.15,0.25,0.28,0.39)\\ &= 0.15\end{aligned}$$



凝聚层次聚类



Original Points



Two Clusters

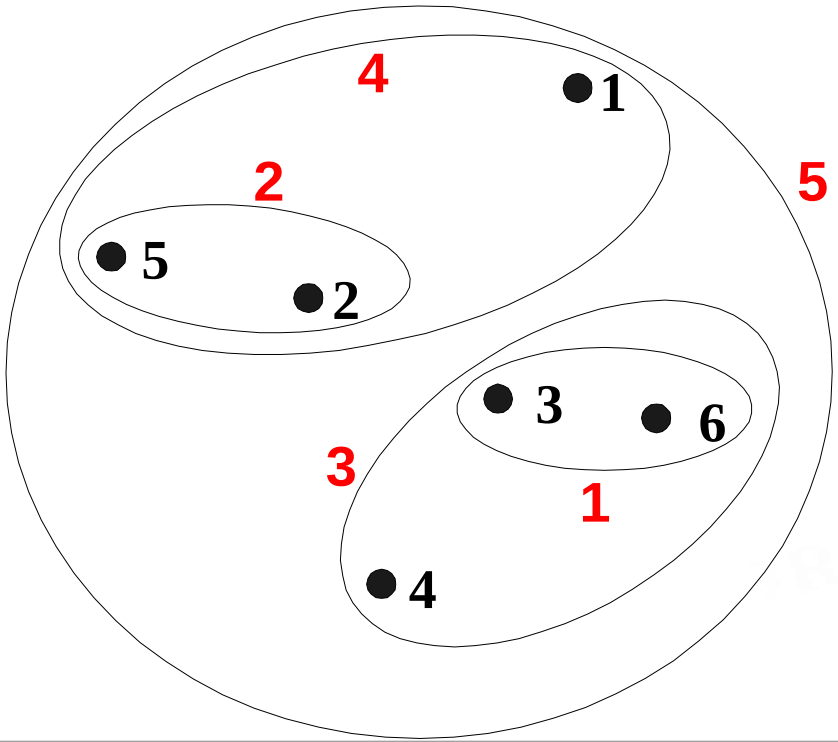
凝聚层次聚类

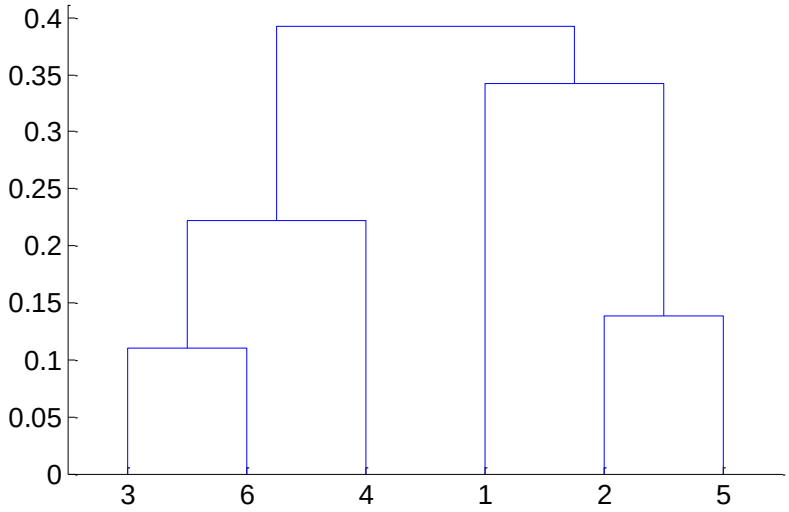
➢全链聚类

Dist({3, 6}, {4}) = max(dist(3, 4), dist(6, 4))
= max(0.15, 0.22)
= 0.22

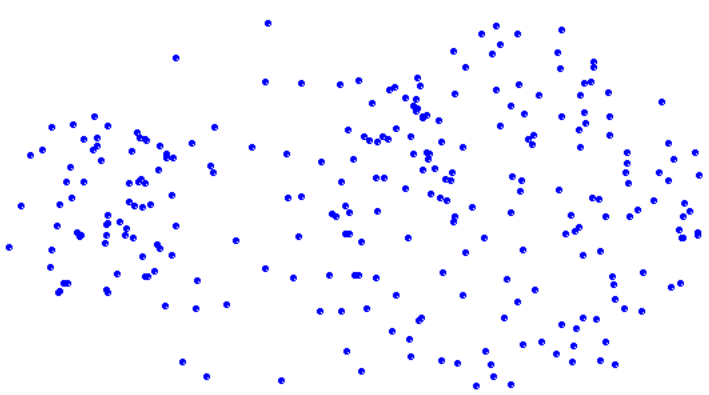
Dist({3, 6}, {2, 5}) = max(dist(3, 2), dist(6, 2), dist(3, 5), dist(6, 5))
= max(0.15, 0.25, 0.28, 0.39)
= 0.39

Dist({3, 6}, {1}) = max(dist(3, 1), dist(6, 1))
= max(0.22, 0.23)
= 0.23

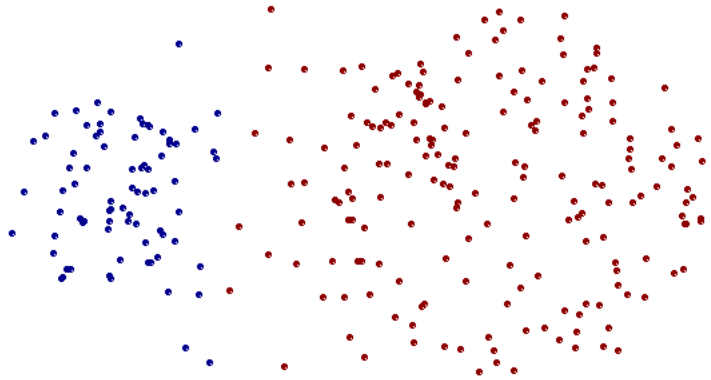




凝聚层次聚类



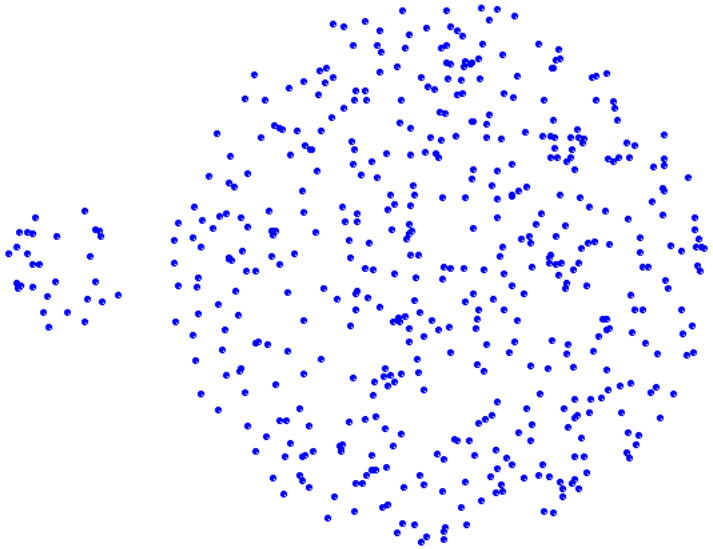
Original Points



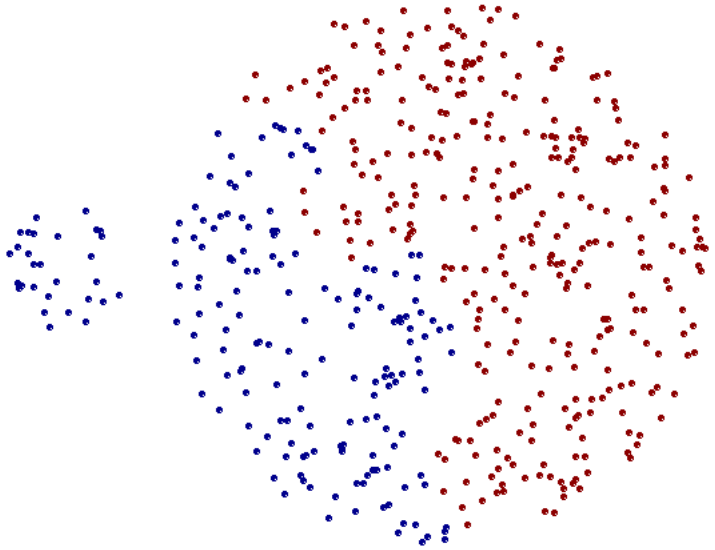
Two Clusters

- 对噪声和离群点不是很敏感

凝聚层次聚类



Original Points



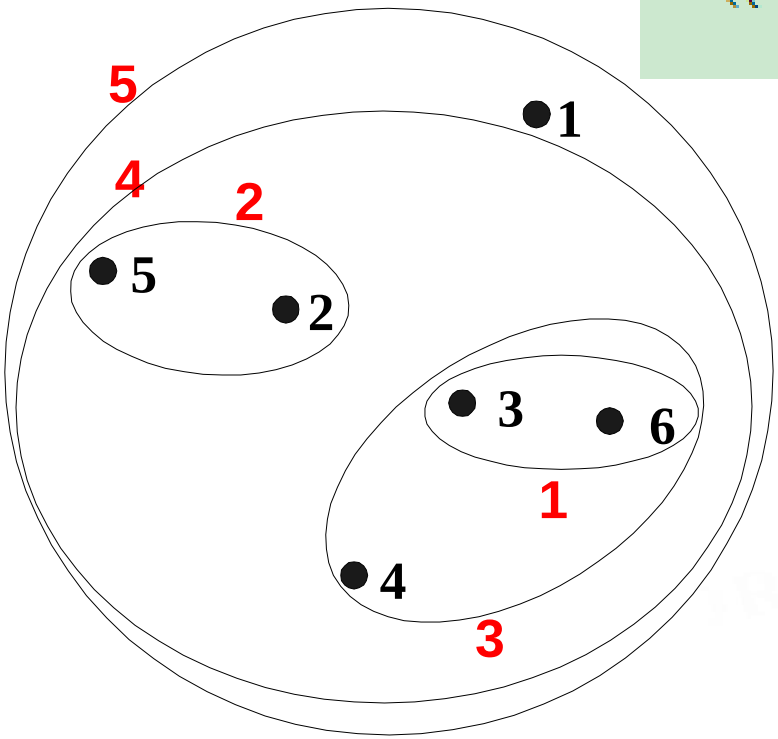
Two Clusters

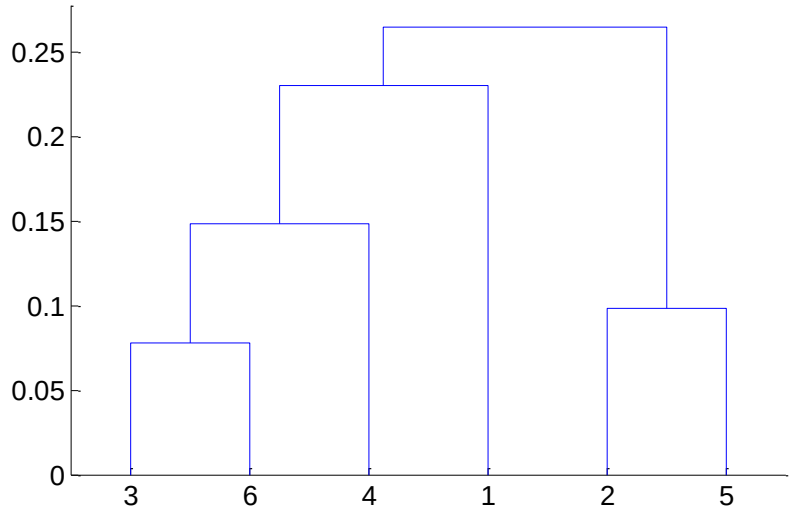
- 倾向于分裂较大的簇

凝聚层次聚类

$$proximity(C_i, C_j) = \frac{\sum_{\substack{x \in C_i \\ y \in C_j}} proximity(x, y)}{m_i * m_j}$$

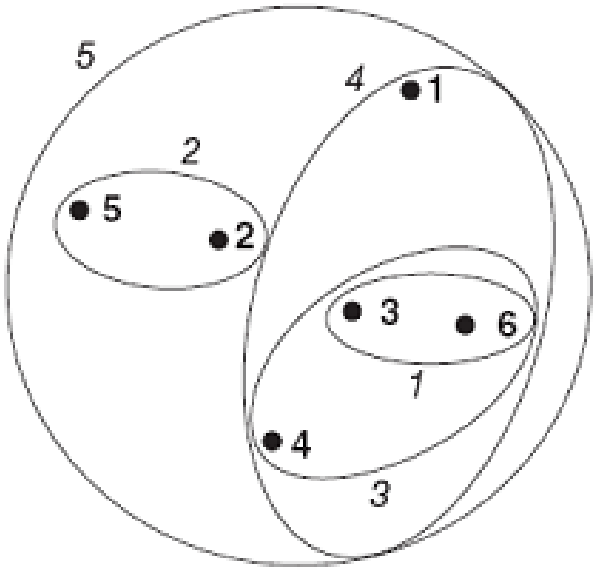
Dist({3, 6, 4}, {1}) = (0.22+0.37+0.23)/(3*1)
= 0.28
Dist({2, 5}, {1}) = (0.2357+0.3421)/(2*1)
= 0.2889
Dist({3, 6, 4}, {2, 5}) = (0.15+0.28+0.25+0.39+0.20+0.29)/(6*2)
= 0.26

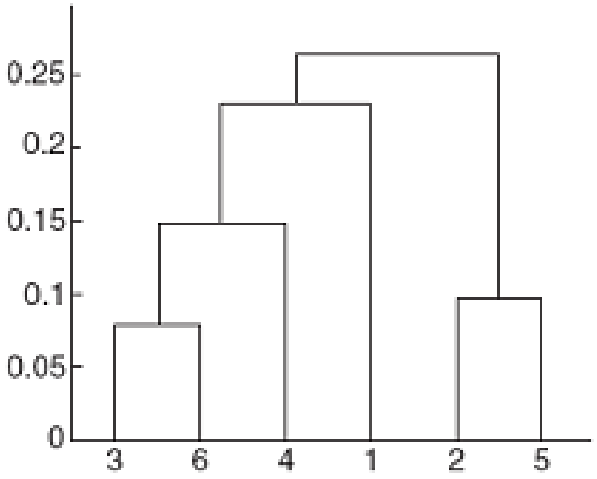




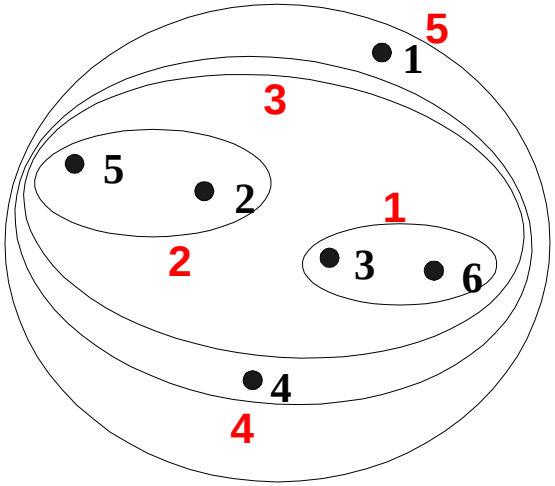
凝聚层次聚类

- **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



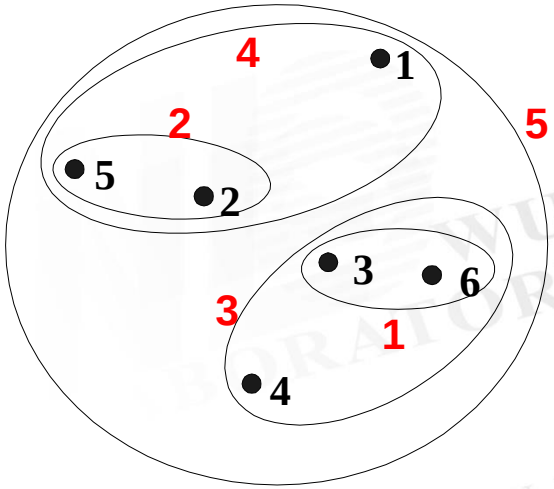


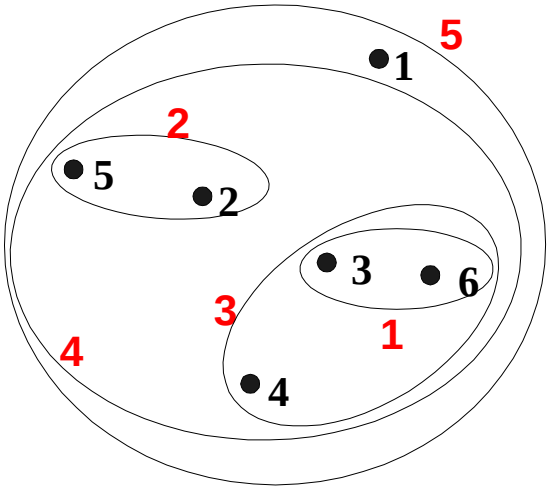
凝聚层次聚类



MIN

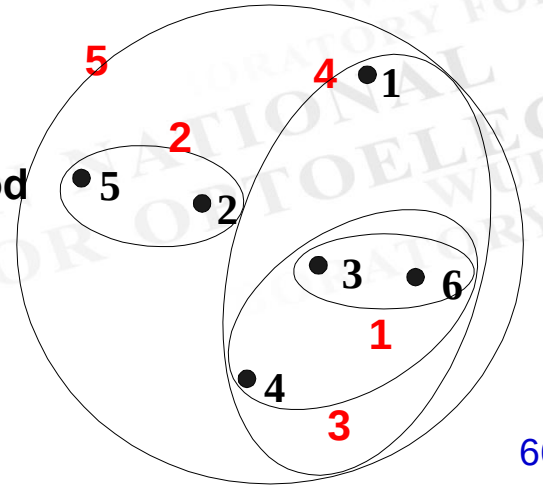
MAX





Group Average

Ward's Method

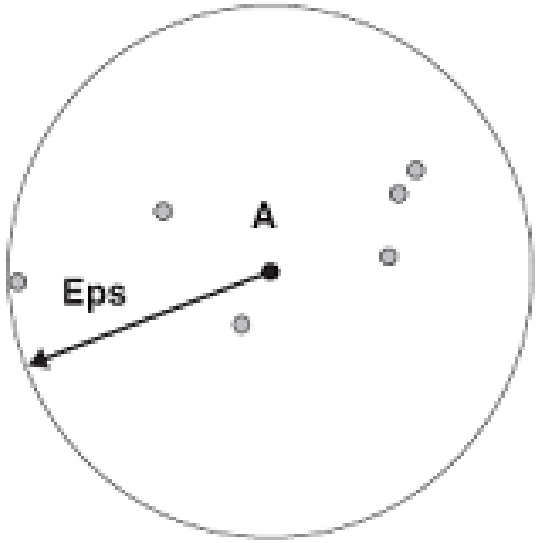


凝聚层次聚类

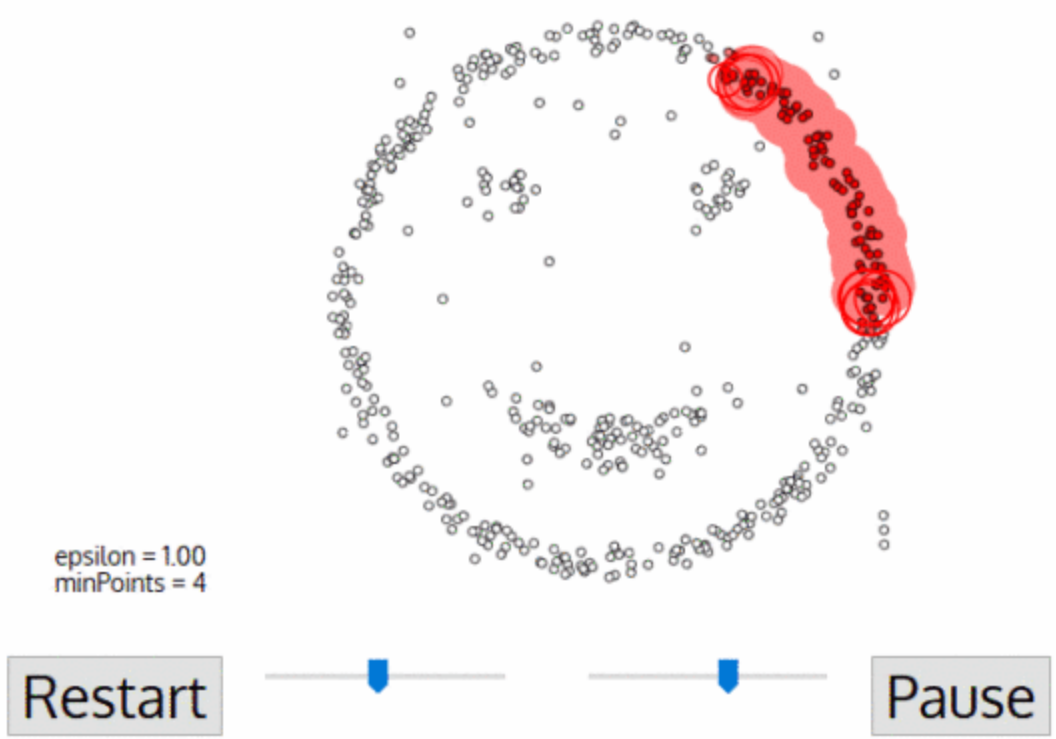
- 1. 缺乏全局目标函数
- 2. 合并决策是最终的
  - 决策不能撤销，阻碍局部最优变成全局最优
- 3. 不需要指定聚类的数量
- 4. 效率低，复杂度高

DBSCAN

- DBSCAN是基于密度的聚类算法
  - 寻找被低密度区域分离的高密度区域
  - 基于中心的密度定义
    - 某一区域范围内(Eps)点的个数



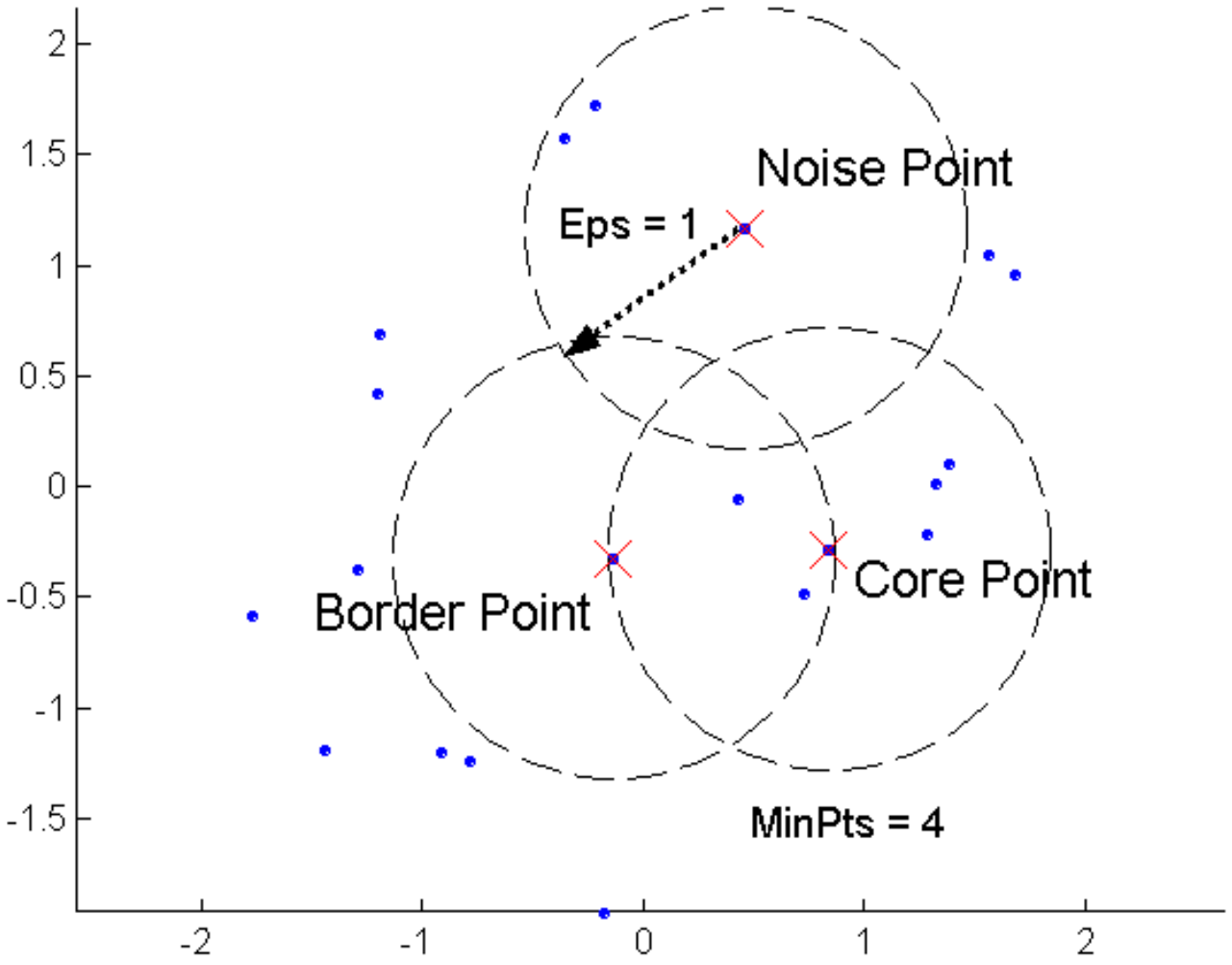
DBSCAN



DBSCAN

- 根据基于中心的密度，可以将点分类为
- 1. 核心点：该点的给定邻域内的点数超过给定的阈值MinPts
  - 在基于密度的簇内部
- 2. 边界点：给定邻域内的点数小于MinPts，但落在某个核心点的邻域内
  - 可能落在多个核心点的邻域内
- 3. 噪声点：非核心点或边界点

DBSCAN


Noise Point
Eps = 1
Border Point
Core Point
MinPts = 4
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1
0
1
2

DBSCAN

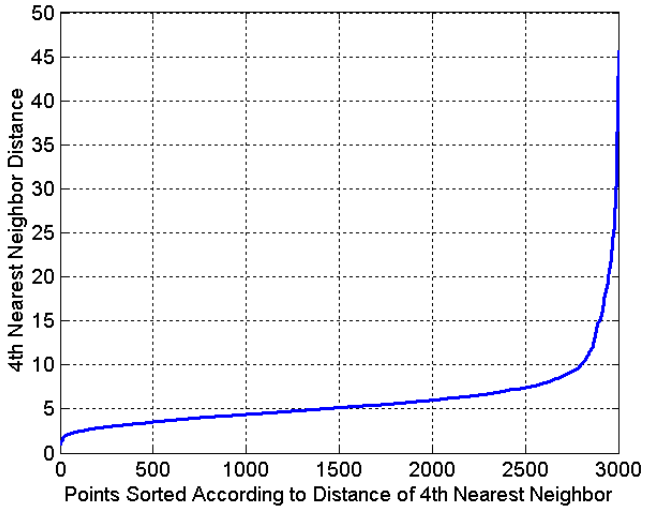
- 任意两个足够靠近的核心点放在同一个簇中
- 任何与核心点足够靠近的边界点也放在核心点相同的簇中
  - 平局问题
- 丢弃噪声点

DBSCAN 算法

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

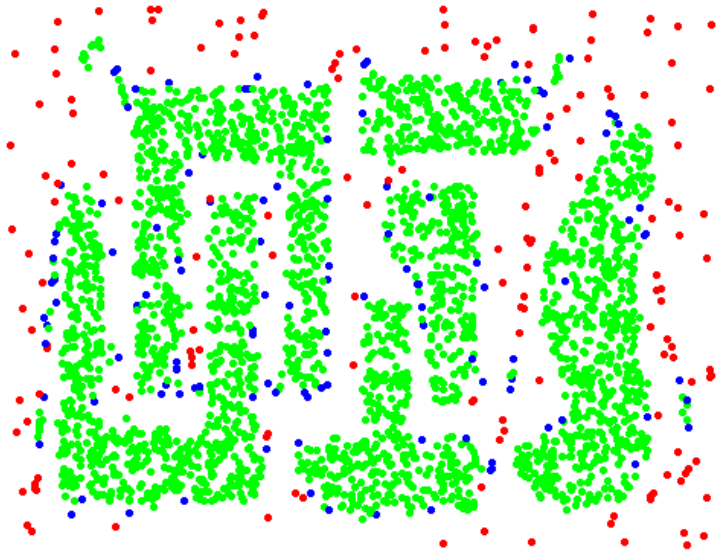
DBSCAN

- 参数选择问题
  - Eps和MinPts
- 观察点到它的k个最近邻的距离的特性
  - 如果簇密度差异不极端，平均情况下k的变化不会太大
  - k的值太小，则少量邻近点的噪声或离群点可能会不正确的标记为簇。如果k的值太大，则小簇可能标记为噪声



DBSCAN

Original Points



Point types: core, border and noise

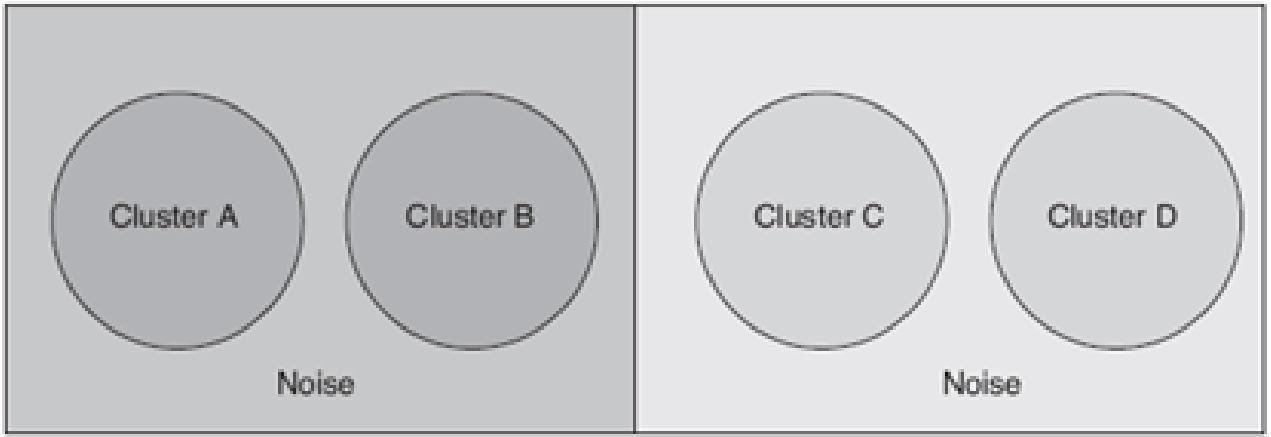
Eps = 10, MinPts = 4

DBSCAN

- 不需要预先设置集簇数量
- 能够很好地适应任意大小和任意形状的集簇
- 当集簇具有不同的密度时，性能下降明显
  - 当密度变化时，用于识别邻域点的距离阈值 **Eps** 和**MinPts**的无法自适应变化

DBSCAN

- 当簇的密度变化很大时
  - Eps足够低时，可以发现簇C和D，簇A和B及包围它们的点将变成单个簇
  - Eps足够高时，可以发现簇A和B，其余的点和簇将标记为噪声

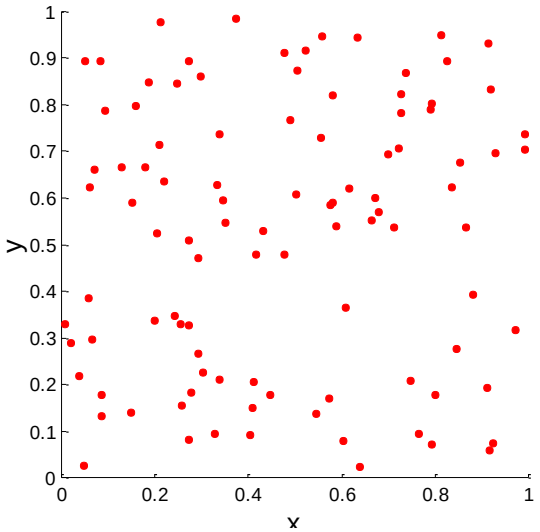


簇评估

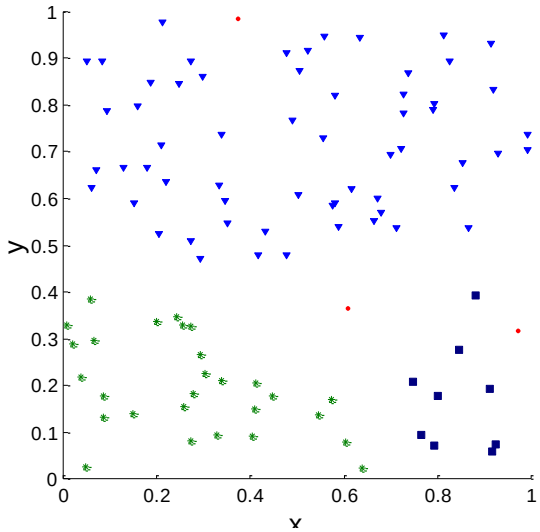
- 对于监督分类中模型的评估相对容易实现，且存在广泛接受的评估度量和过程
  - 准确度，精度
- 对于非监督分类（聚类），如何评估获得的簇的好坏？
  - 簇存在于观察者的眼中
- 为什么需要评估簇呢？
  - 几乎每种聚类算法都会在数据集中发现簇，即使数据集中根本没有自然的簇结构

簇评估

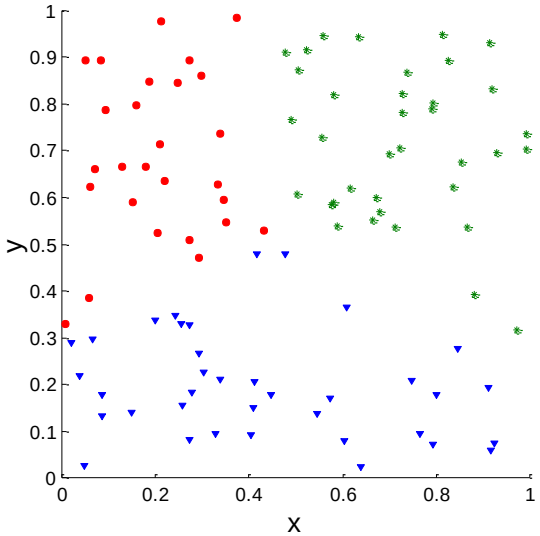
Random Points



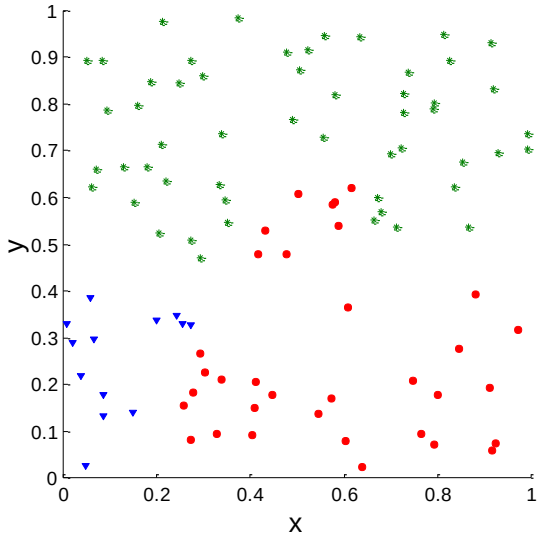
DBSCAN



K-means



Complete Link



簇评估

- 1. 确定数据集的聚类趋势
  - 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（如外部提供的类标号）比较
- 5. 比较两个簇集，确定哪个更好

簇评估

- 用于评估簇的各方面的度量或指标分成三类
- 非监督的（内部指标）
  - 聚类结构的优良性度量，不考虑外部信息
  - 凝聚性&分离性
- 监督的（外部指标）
  - 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
- 相对的
  - 比较不同的聚类或簇
  - 可采用外部或内部指标

非监督簇评估

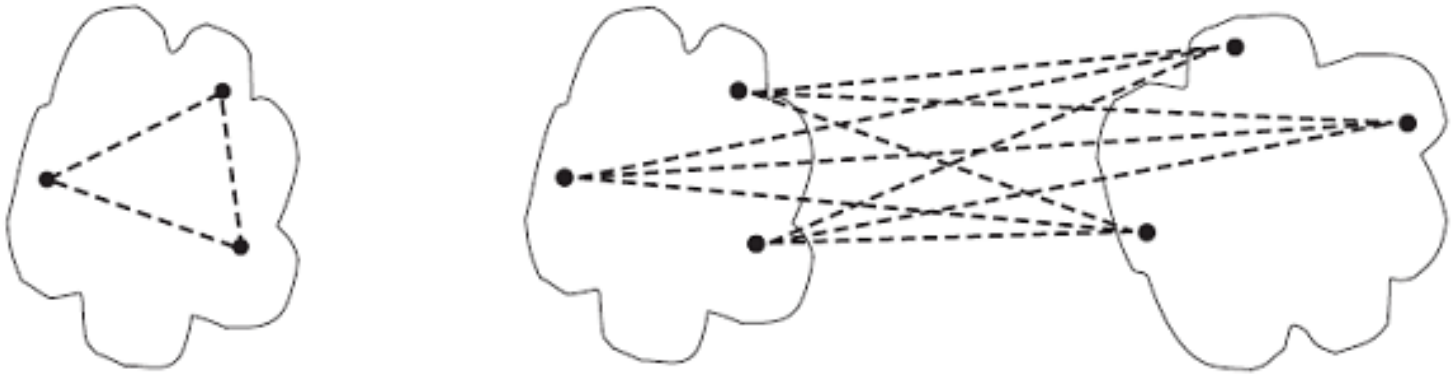
- 一般的，可以用个体簇有效性的加权和来表示K个簇的集合的总体簇有效性

$$overall\ validity = \sum_{i=1}^{K} w_i\ validity(C_i)$$

- Validity函数可以是凝聚度、分离度，或这些量的组合

非监督簇评估

- 基于图的凝聚度和分离度



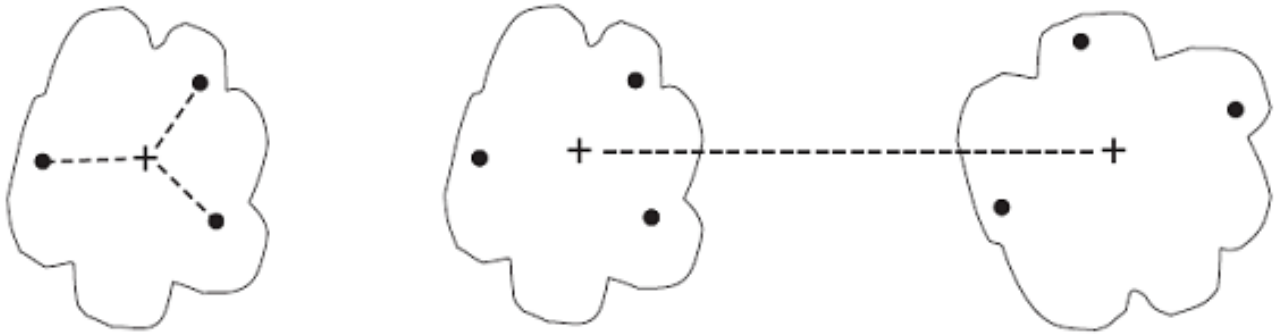
(a) Cohesion.　　(b) Separation.

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

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

非监督簇评估

- 基于原型的凝聚度和分离度



(a) Cohesion. (b) Separation.

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

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

$$separation(C_i) = proximity(c_i, c)$$

非监督簇评估

➢凝聚度和分离度的总度量

基于图的簇评估度量表

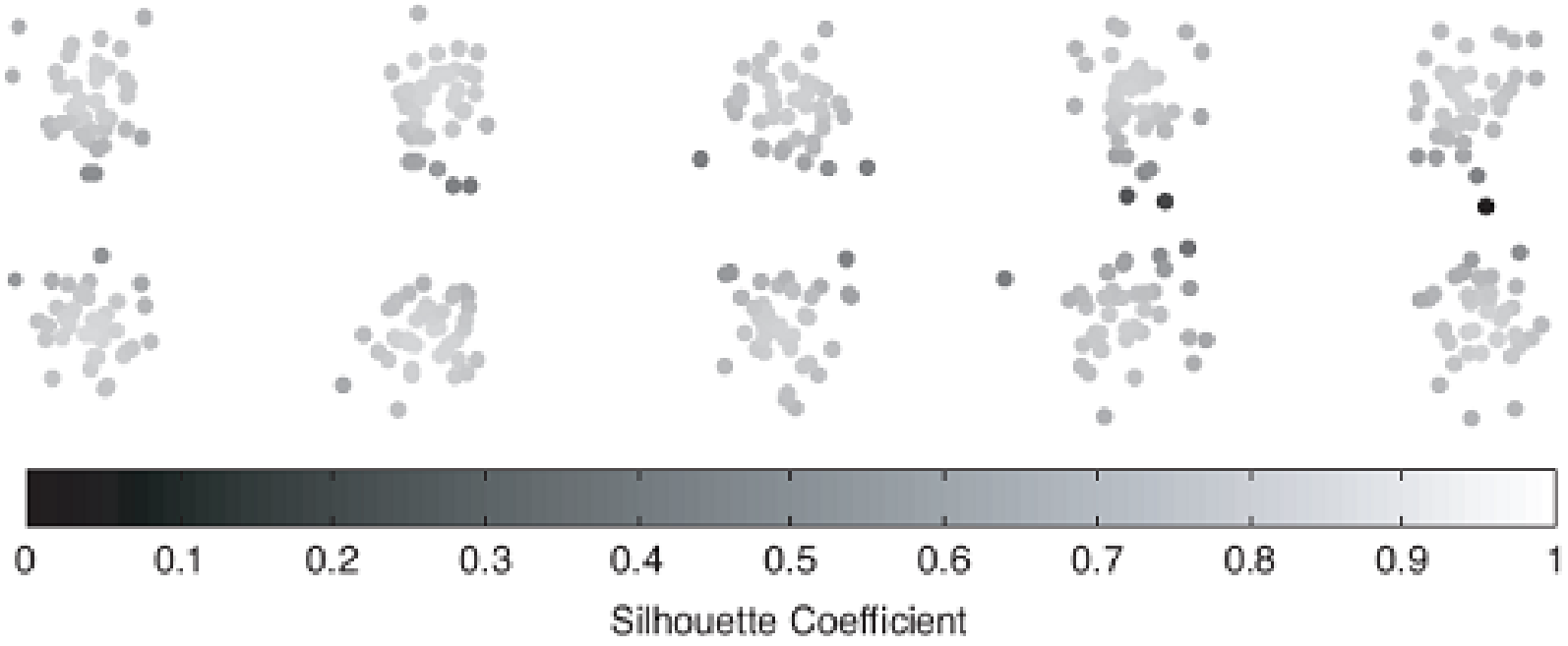
名称	簇度量	簇权值	类型
I_1	$\sum_{x \in C_i, y \in C_i} proximity(x, y)$	$\frac{1}{m_i}$	基于图的凝聚度
I_2	$\sum_{x \in C_i} proximity(x, c_i)$	1	基于原型的凝聚度
ε_1	proximity(c_i, c)	m_i	基于原型的分离度
G_1	$\sum_{j=i}^{k} \sum_{x \in C_i, y \in C_i} proximity(x, y)$	$\frac{1}{\sum_{x \in C_i, y \in C_i} proximity(x, y)}$	基于图的凝聚度和分离度

非监督簇评估

- 轮廓系数
- 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，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。对于所有的簇，找出最小值，记作b_i
- 3. 对于第i个对象，轮廓系数是
  - $s_i=(b_i-a_i)/max(a_i, b_i)$
  - 取值范围-1到1，越大越好

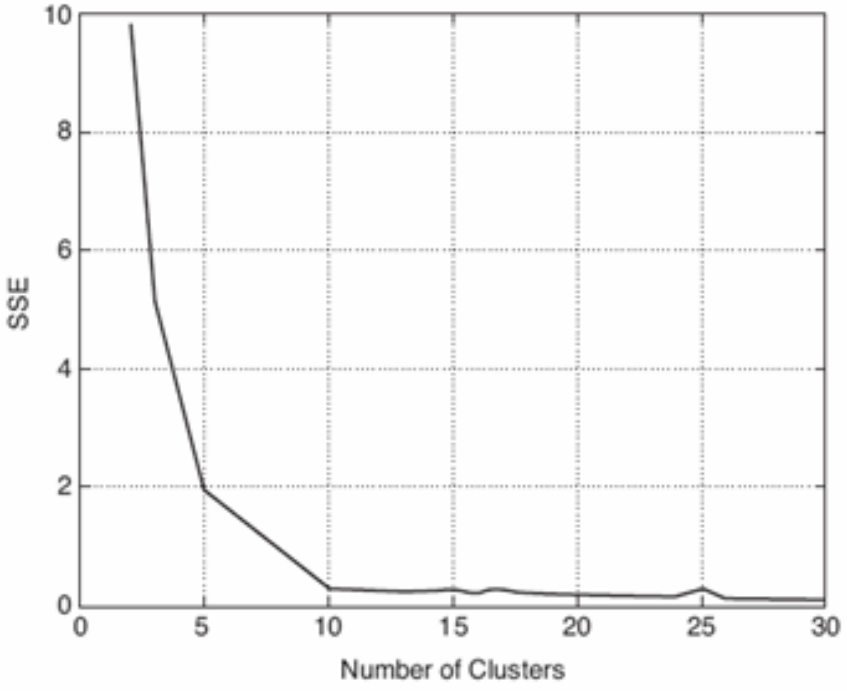
非监督簇评估

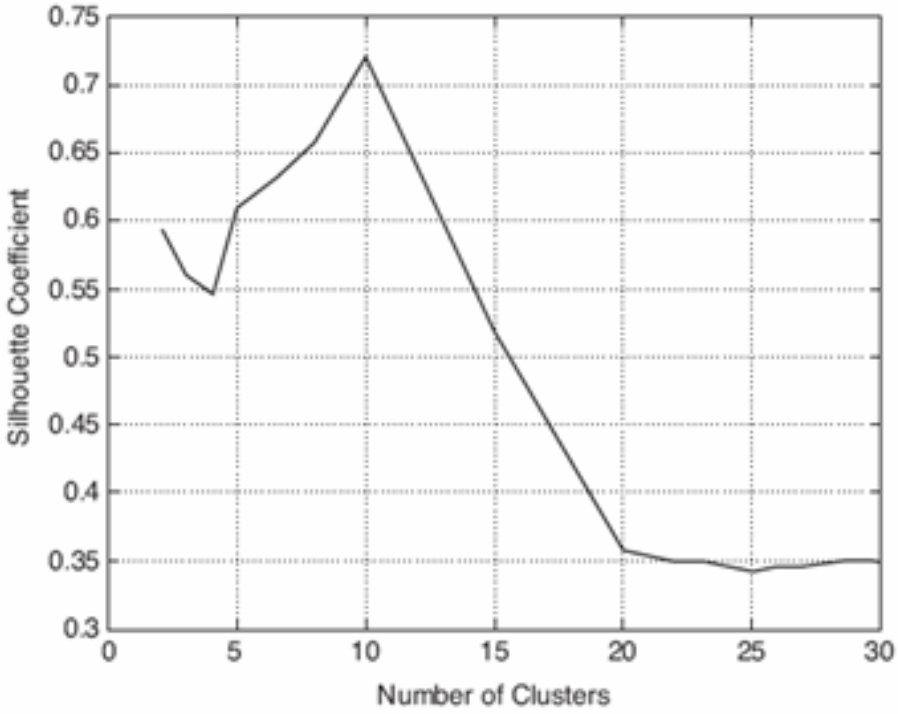
10个簇中点的轮廓系数



确定正确的簇个数

- 通过寻找簇个数的评估度量曲线图中的拐点，尖峰或下降点，试图发现簇的自然个数





聚类趋势

- 通过对聚类趋势的度量，试图评估数据集中是否包含簇
  - 不用进行聚类
- **Hopkins统计量**

$$H = \frac{\sum_{i=1}^{P} w_i}{\sum_{i=1}^{P} u_i + \sum_{i=1}^{P} w_i}$$

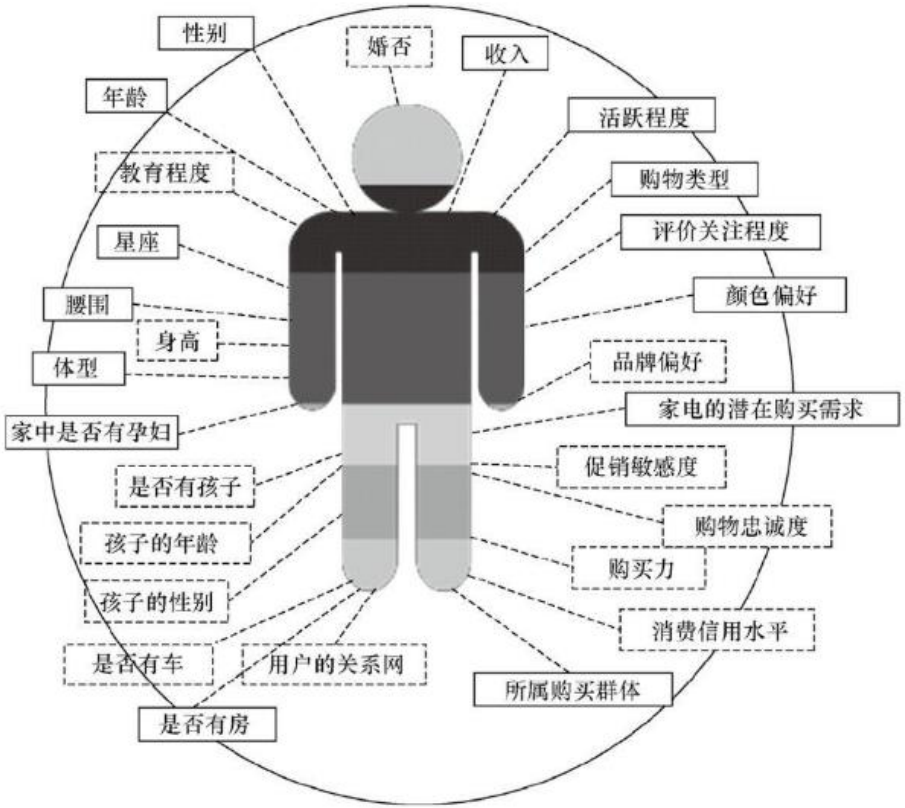
  - **H为0.5左右表明样本点近似随机分布**
  - **H接近0或1表明样本点是有规律分布的**

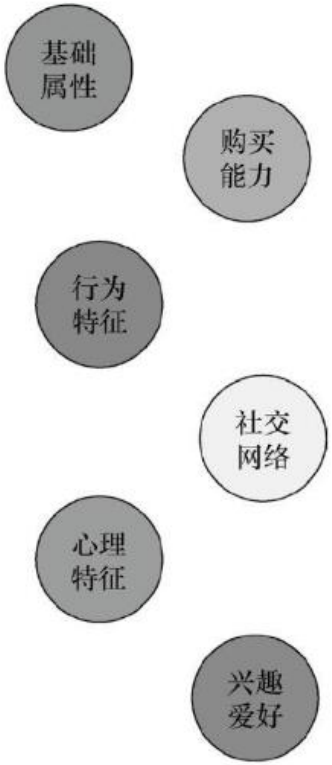
京东大数据应用

- 大数据集群服务器超过1.5万台
- 数据总容量突破200PB
- 每天20万个作业运行
- 拥有接近2亿活跃用户

京东大数据应用

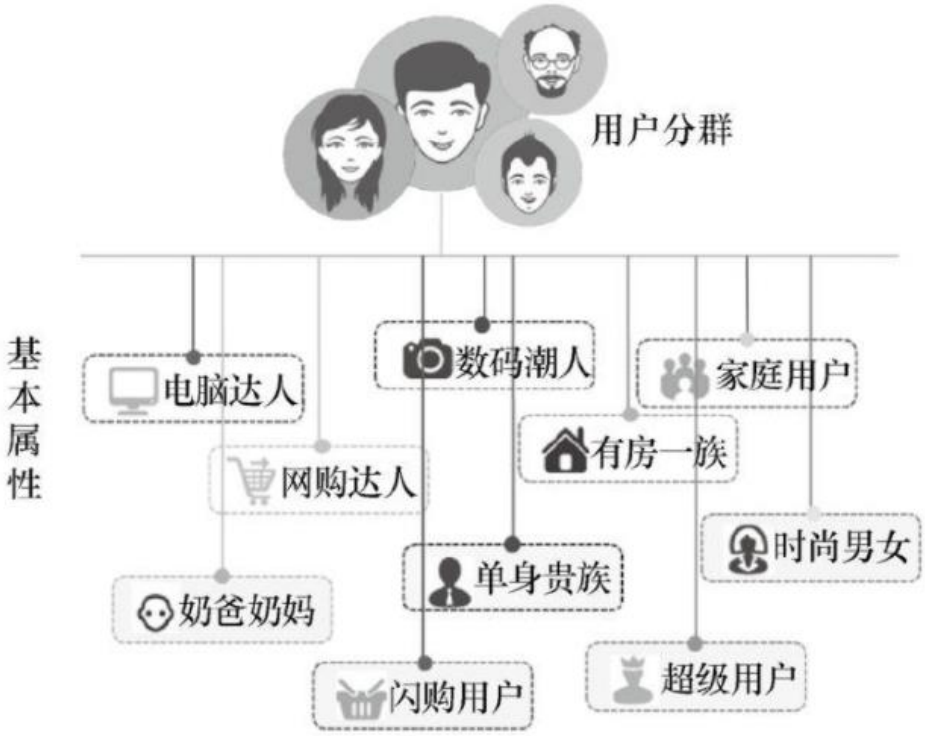
- 京东的用户画像分解
  - 六个维度
  - 27个量化指标

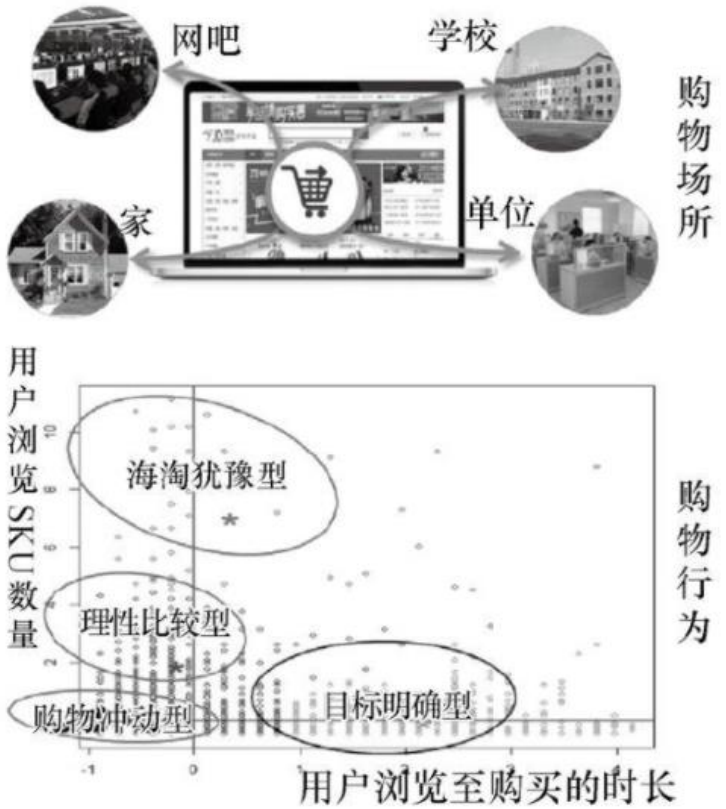




京东大数据应用

- 在用户画像基础上对用户分群





京东大数据应用

➢用户画像实例