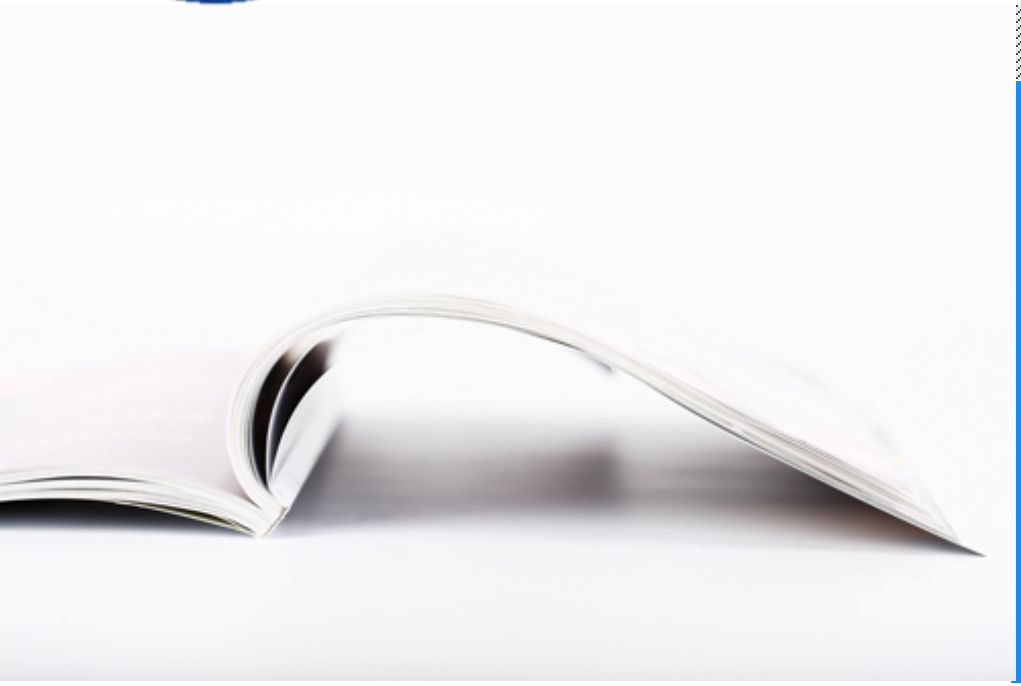


数据科学导论

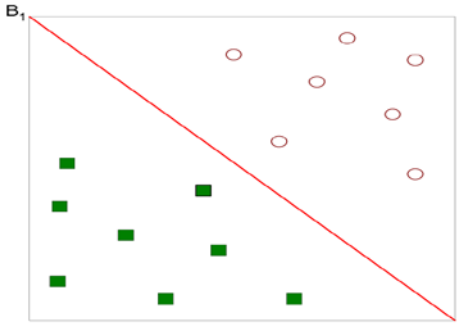
第四节

聚 类

徐童

2020.3.27

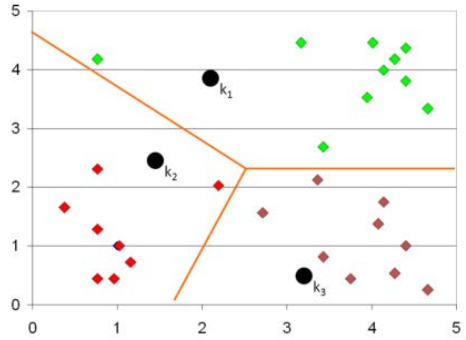
- 数据挖掘的基本方法



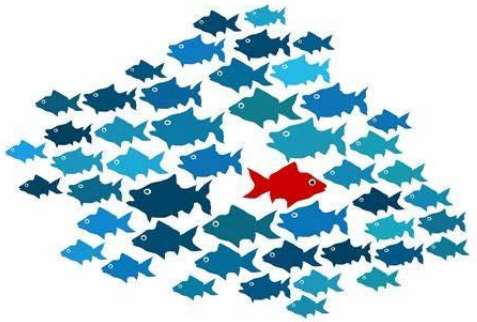
分类

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

关联规则



聚类



离群检测

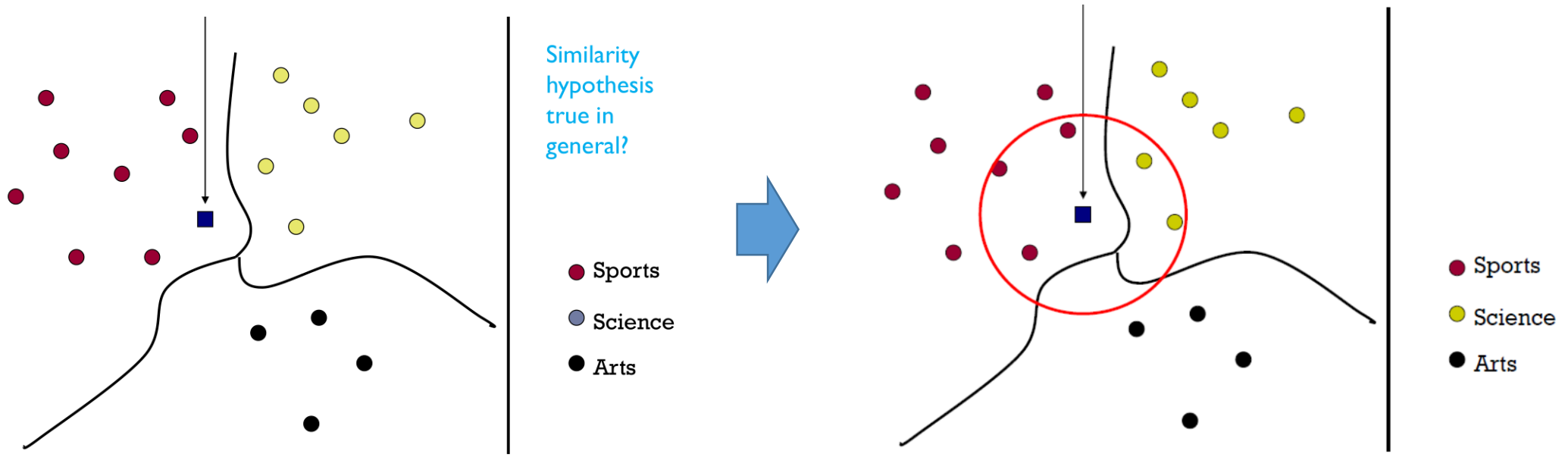
- **时常会面临的困扰**

- 数据常有，而标签不常有
  - 没有标签，无法支撑有监督学习
  - 然而，无标签数据廉价而易得

- *无标签数据可以提炼何种规律*?

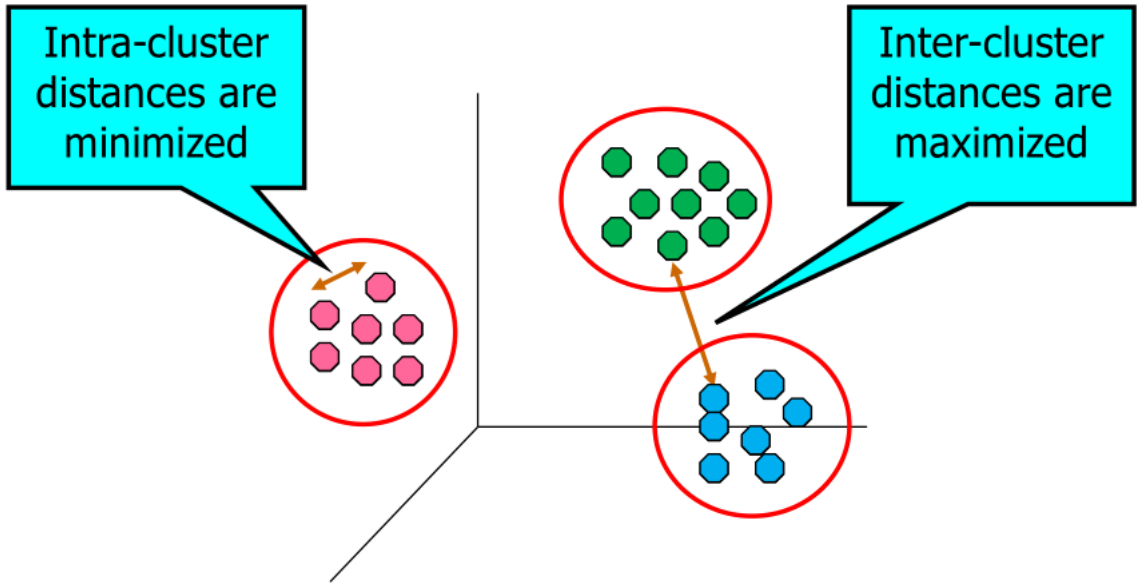
- **回顾：最近邻分类的假设**

- 先前的向量空间模型遵循一个思路：表征空间上相近的文档是相似的
- 同样的，可以提出合理假设：**表征空间上相近**的样本应该属于同一个类别



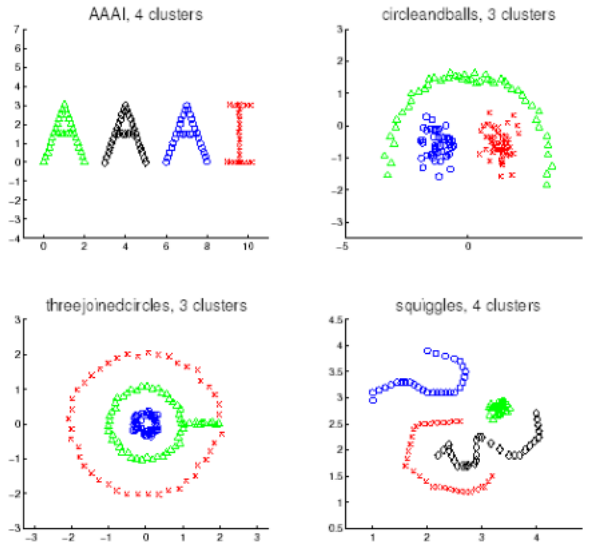
- **基本概念：簇（Cluster）**

- 通过将样本表征为高维向量，相似样本将自发地形成“簇”的结构
  - 簇的特点：簇内相似（距离较近），簇间相异（距离较远）



- **由簇衍生的聚类问题**

- 聚类（Clustering）方法的目的，就是将样本分为若干个簇（Clusters）
  - 其中，每个簇都由相似的样本所组成的结构
  - 聚类方法，是最常见的无监督学习（Unsupervised Learning）方法
  - 与分类依托样本标签，天然具有问题合理性的判别不同，聚类的合理性受问题与方法定义影响存在一定不确定性，需要专门约定。

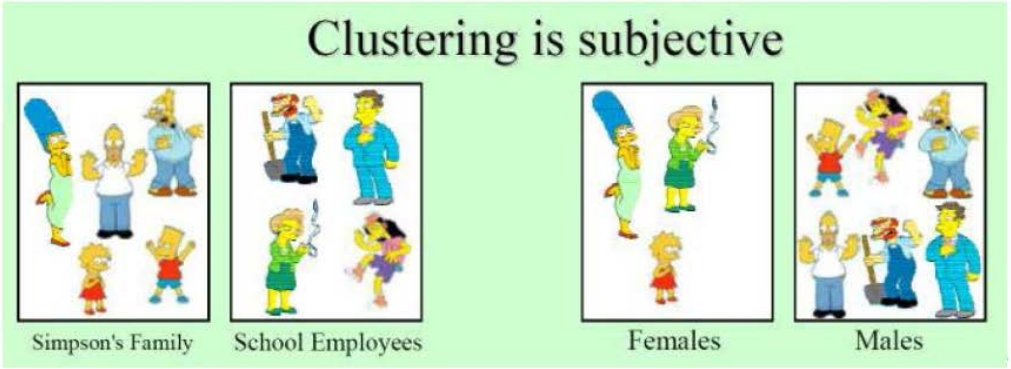


- **聚类的三个基本问题**

- 在聚类过程中，以下三个基本问题应得到回答
  - 在数据中，样本是如何形成簇/社团的?
    - 换言之，样本的“群体性”的依据是什么
  - 如何度量样本之间的相似性?
    - 不同的相似性度量可能导致截然不同的簇
  - 簇的数量如何确定?
    - 由于聚类没有天然标签，簇的数量往往是个开放性问题

- **基本问题（1）何以为簇?**

- 基于不同的“群体性”立场，可以得到不同的簇
  - 因此，聚类是具有一定主观性的，其主观性来自于聚类依据的选择
  - 在选定聚类依据时，应根据聚类的目的加以选择
    - 例如，实验分组应该考虑学生的专业还是性别?



- **基本问题（2）如何度量?**

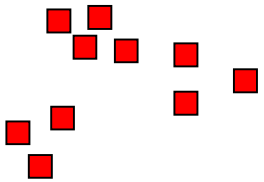
- 即使明确了聚类的目的，在相似性的度量上依然面临问题
  - 相似性度量往往存在一定局限性，未必反映聚类的真实意图
  - 相似性度量往往根据算法或表征加以选择，例如大多数时候采用距离衡量相似性，但有隐患
    - 例如，用向量表征人的爱好，好友之间是不是向量绝对相似?

- **基本问题（3）簇的数量?**

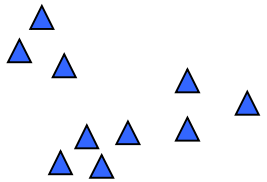
- 由于聚类没有天然标签，簇的数量往往是个开放性问题
  - 过大或过小的簇都应该避免，会导致失去代表性，但这未必可通过簇数调节

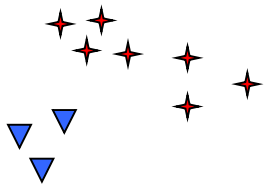
How many clusters?

Six Clusters

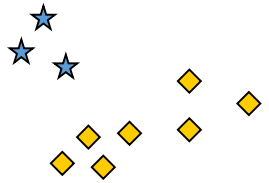


Two Clusters





Four Clusters



本节目录

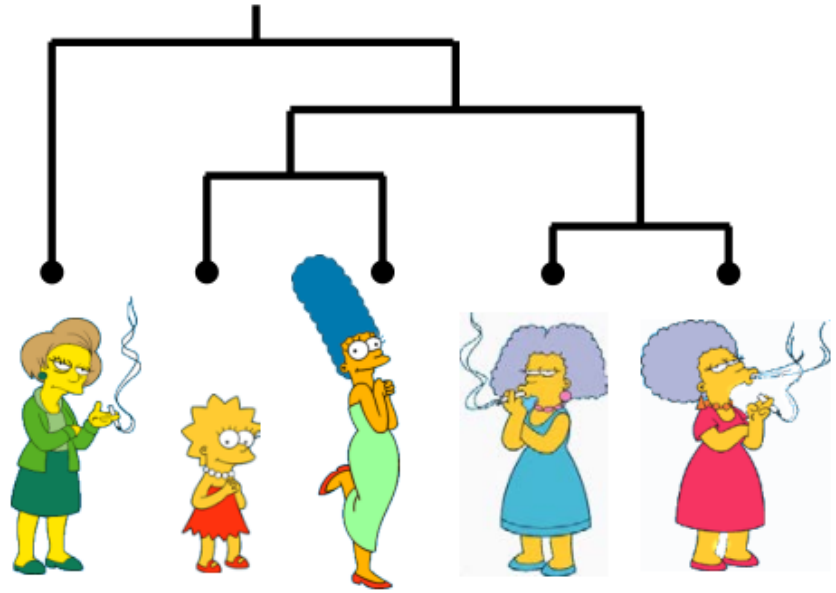
- **常见的聚类方法**
  - K均值聚类
  - 层次聚类
  - 基于密度聚类
  - 模糊聚类初阶
- 聚类问题的评估

- **常见的聚类方法**

- 通常将聚类方法分为层次的（Hierarchical）与划分的（Partitional）两种

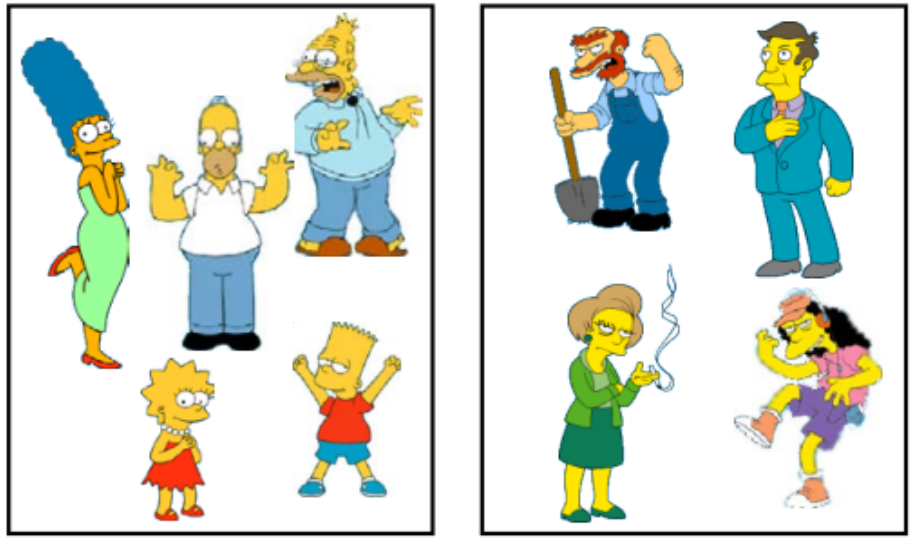
层次聚类

树状形式的嵌套簇集合



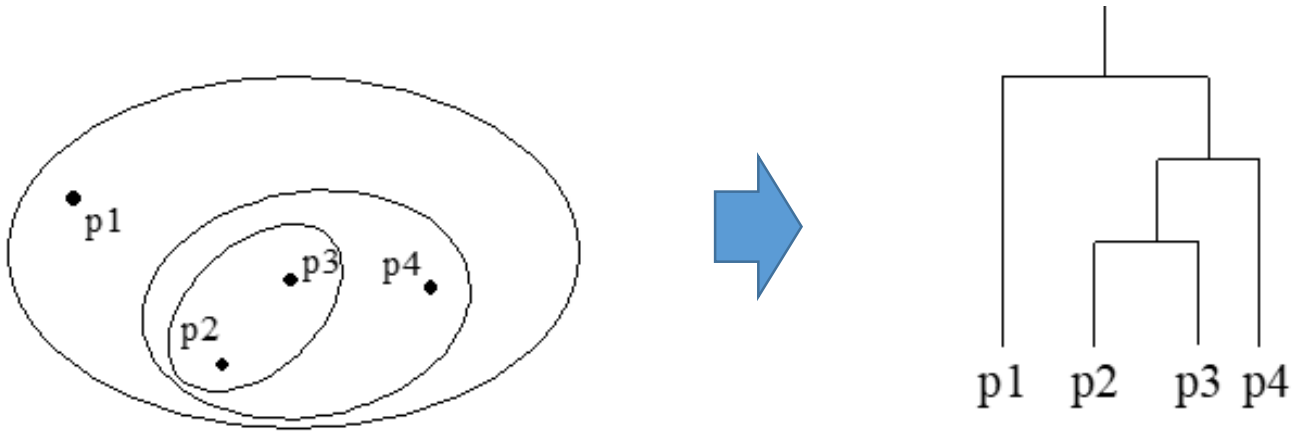
划分聚类

简单将样本划分为不重叠的簇



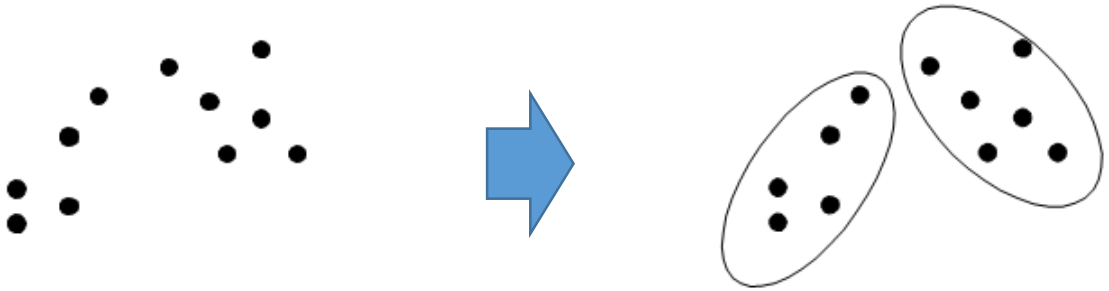
- **常见的聚类方法**

  - 层次聚类（Hierarchical Clustering）
    - 整体呈树状结构，除叶节点外，每个节点是子节点的并集
    - 叶节点一般为单个样本组成的单元簇



- **常见的聚类方法**

  - 划分聚类（Partitional Clustering）
    - 每个对象恰在一个子集（簇）中
    - 簇与簇之间相互不重叠



- **其他区分规则**

- 互斥（Exclusive） v.s. 重叠 （Non-exclusive） v.s. 模糊（Fuzzy）
  - 在非互斥（重叠）聚类中，每个样本可能属于多个簇
    - 某种意义上，与多标签分类相对应
  - 在模糊聚类中，每个样本以一定从属度隶属于不同的簇
    - 对于单个样本，隶属于所有簇的从属度之和应为1（归一化要求）
- 同构（Homogeneous） v.s. 异构（Heterogeneous）
  - 不同的簇可能具有不同的规模、形状、密度等属性

本节目录

- 常见的聚类方法
  - K均值聚类
  - 层次聚类
  - 基于密度聚类
  - 模糊聚类初阶
- 聚类问题的评估

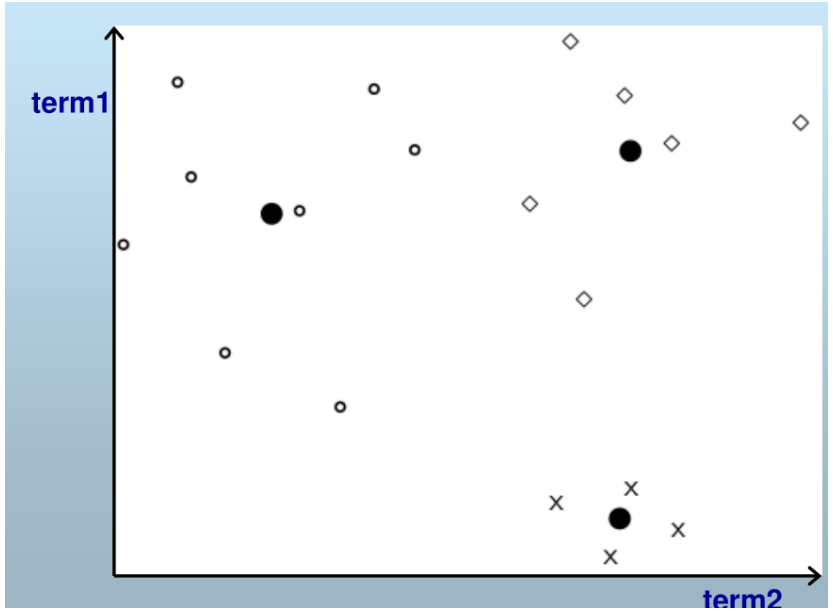
先前回顾

- **前提：向量空间模型中的质心概念**

- 某种意义上说，由向量表示的文档，可以视作高维空间中的一个点
- 由此，“质心”就是一系列点（文档）的重心
  - 我们可以用如下公式来计算一类文档的质心

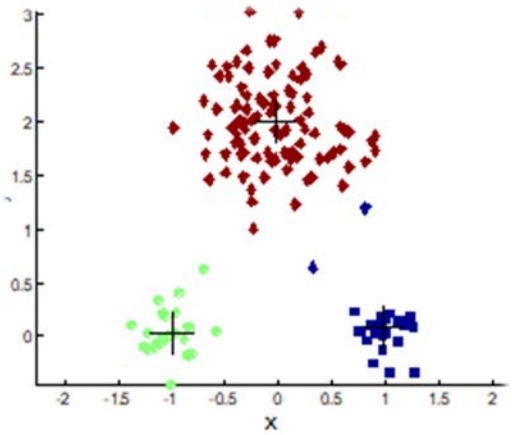
$$\vec{\mu}(C) = \frac{1}{|C|}\sum_{d \in C}\vec{d}$$

  - 其中，C是文档的集合



• K均值聚类的基本思想

- 已知文档由高维向量表征的前提下，簇的中心可以近似反应整个簇的属性
- K均值（K-means）聚类的思想，就是通过设定K个中心，来形成K个簇
  - 然后，通过不断更新簇中心的向量，来更新聚类的结果，直至收敛
  - 簇中心的更新依赖于对当前簇中样本的算术平均
  - 簇中心更新后，根据距离将样本重新分配至不同的簇
  - **收敛**：所有节点的聚类结果不再更新，停止迭代



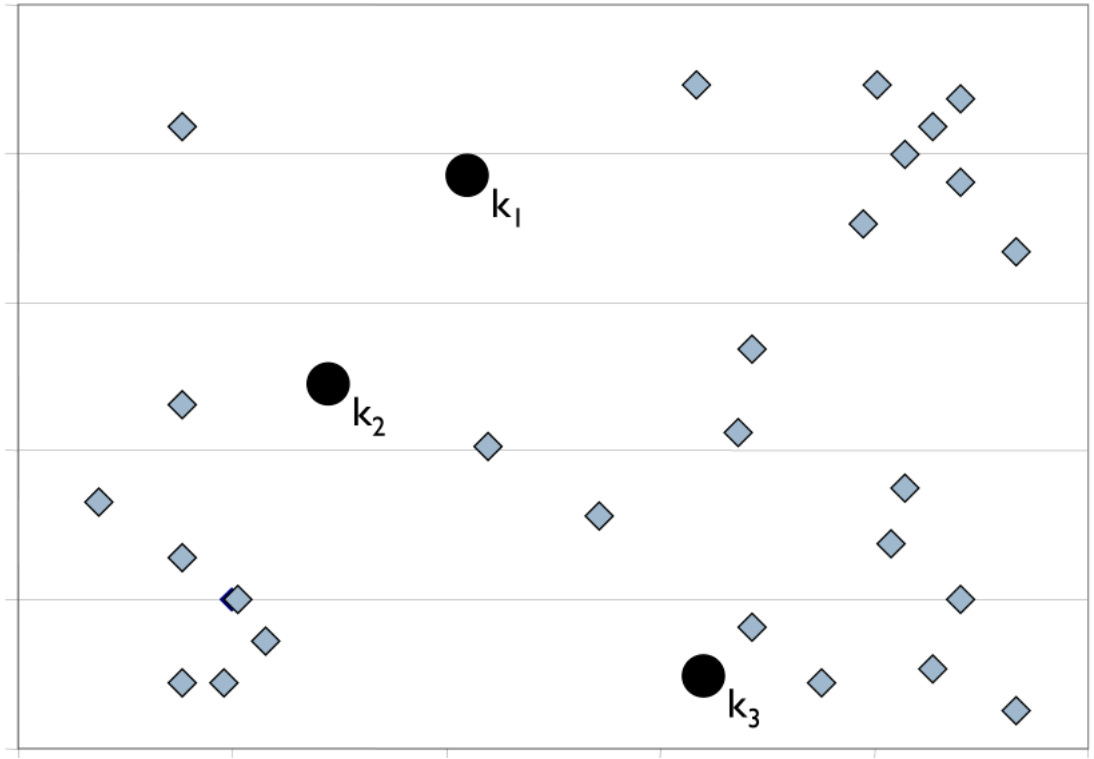
- **K均值聚类的基本思想**

- K均值（K-means）聚类的伪代码如下：

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

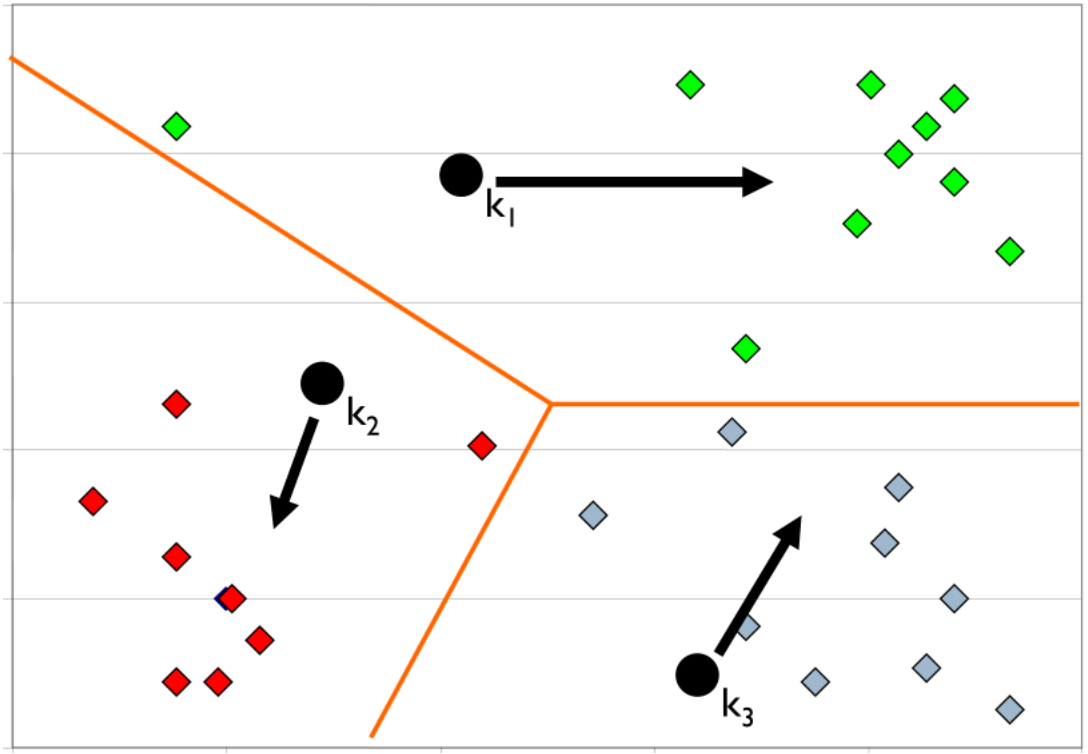
- **一个K均值聚类的实例**

  - 第一步：初始中心的选取（K = 3的情况下）



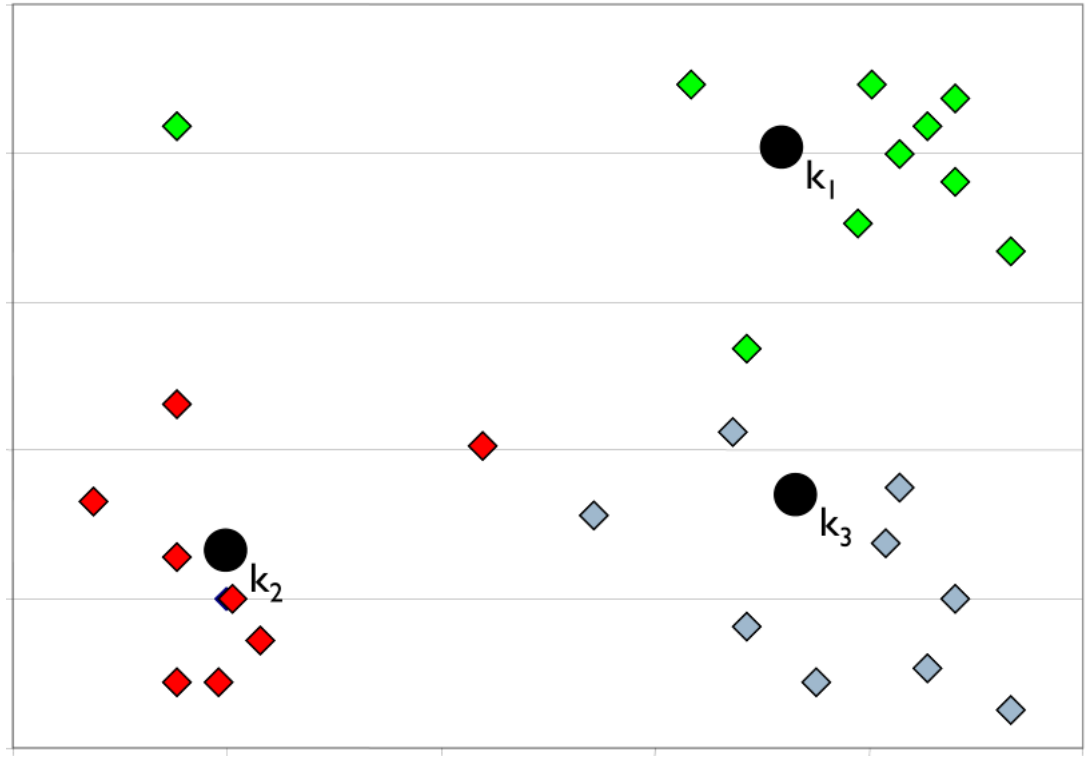
- **一个K均值聚类的实例**

  - 第二步：基于初始中心，进行第一次聚类



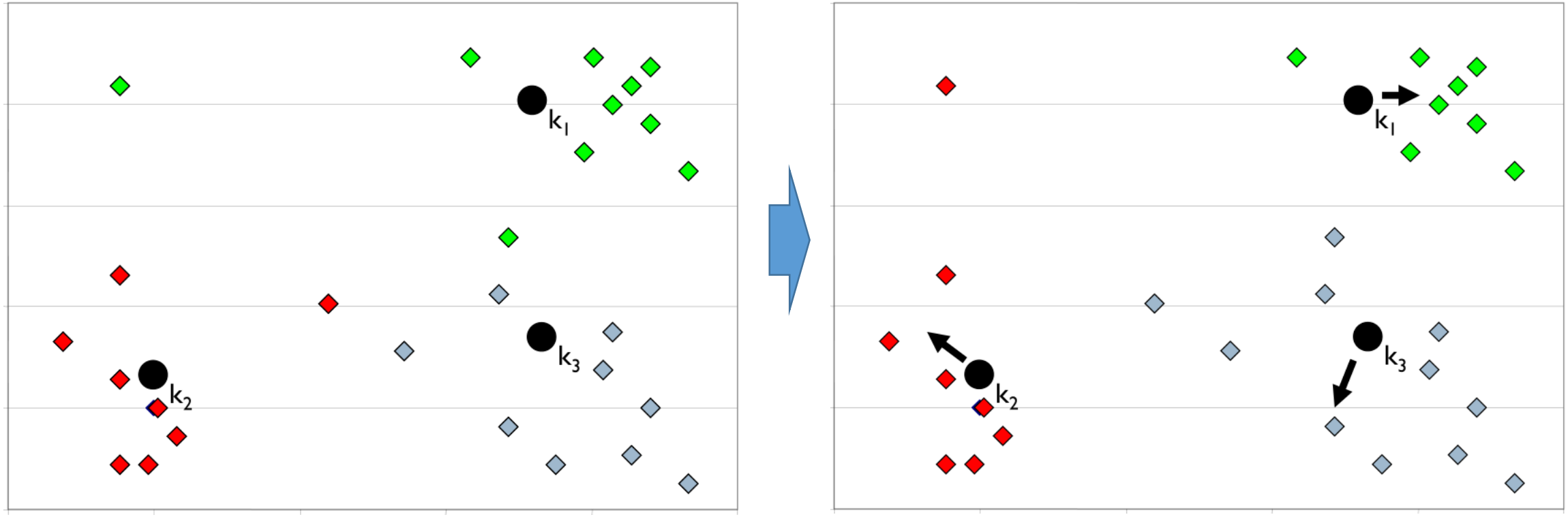
- **一个K均值聚类的实例**

  - 第三步：基于聚类结果，更新簇中心的向量



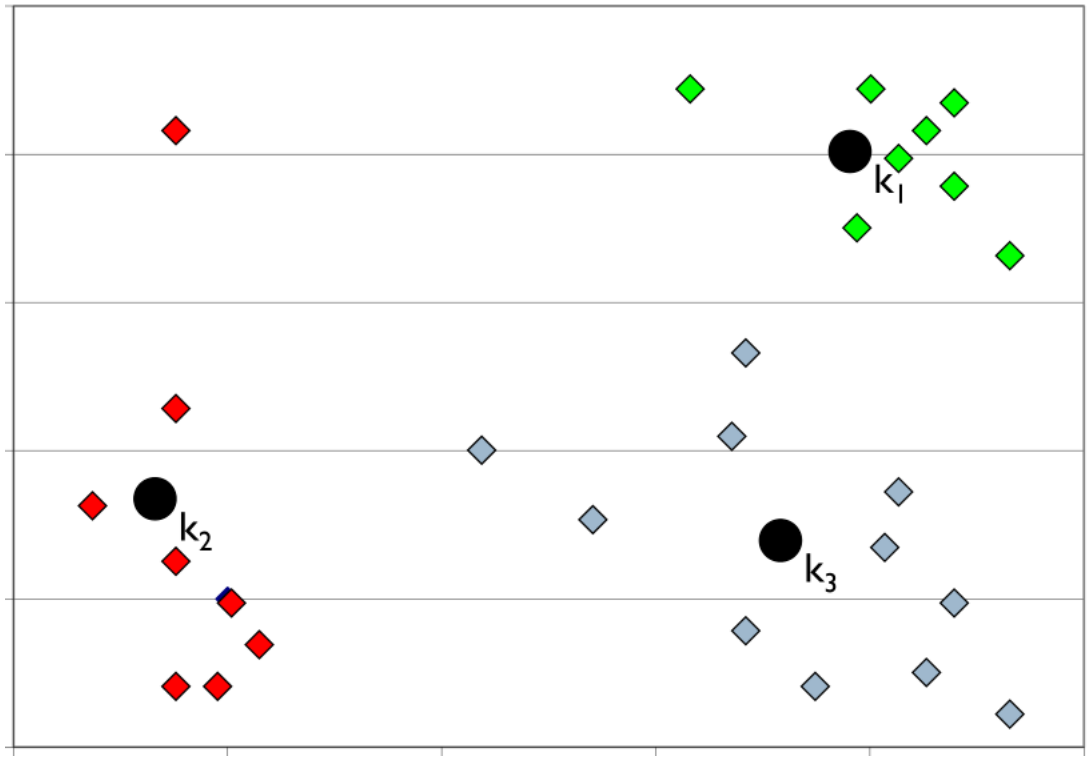
- **一个K均值聚类的实例**

  - 第四步：基于更新后的簇中心，再次进行聚类（该步将进行迭代）



- **一个K均值聚类的实例**

  - 第五步：基于更新聚类结果，再次更新簇中心（该步将进行迭代）



• K均值聚类的若干细节

- K均值的初始中心往往随机选定，因此不同中心可能导致不同聚类结果
- 一般而言，簇中心由样本向量的算术平均所决定
- 样本的归属通常由欧式距离决定，但也可由余弦相似度等指标替代
- 大多数情况下，聚类过程将在少数几次迭代后收敛（即不再更新）
  - 也可将停止条件修正为“低于一定数量的节点变更簇的归属”
- K均值聚类的复杂度为$O(n * K * I * d)$
  - 与样本数量（n），簇数量（K），迭代次数（I），向量维度（d）相关

K均值聚类的效果衡量

- 通常，可采用平方误差和（Sum of Squared Error, SSE）衡量聚类效果
  - 由于将样本分配给最近的簇，显然，我们希望优化样本到簇中心的距离
  - 为此，我们可以将相应的SSE定义为：$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$
    - 其中，x为样本，m_i为簇C_i的中心，x属于簇C_i
  - 显然，面对多个聚类结果时，我们倾向于选择较小的SSE对应的结果
  - 当簇数量K增加时，SSE一般趋于下降，因此尽量在同等K下比较SSE
    - 某个K和SSE都较小的聚类，显然优于K和SSE都较大的聚类

• K均值聚类的另类解读

- 从优化SSE的视角，重新审视K均值聚类的过程

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

  - 当我们沿着其中某个簇中心的方向进行优化时，可得

$$c_j = \text{argmin}_{i \in \{1,2,\cdots,k\}} dist(m_i, j)$$

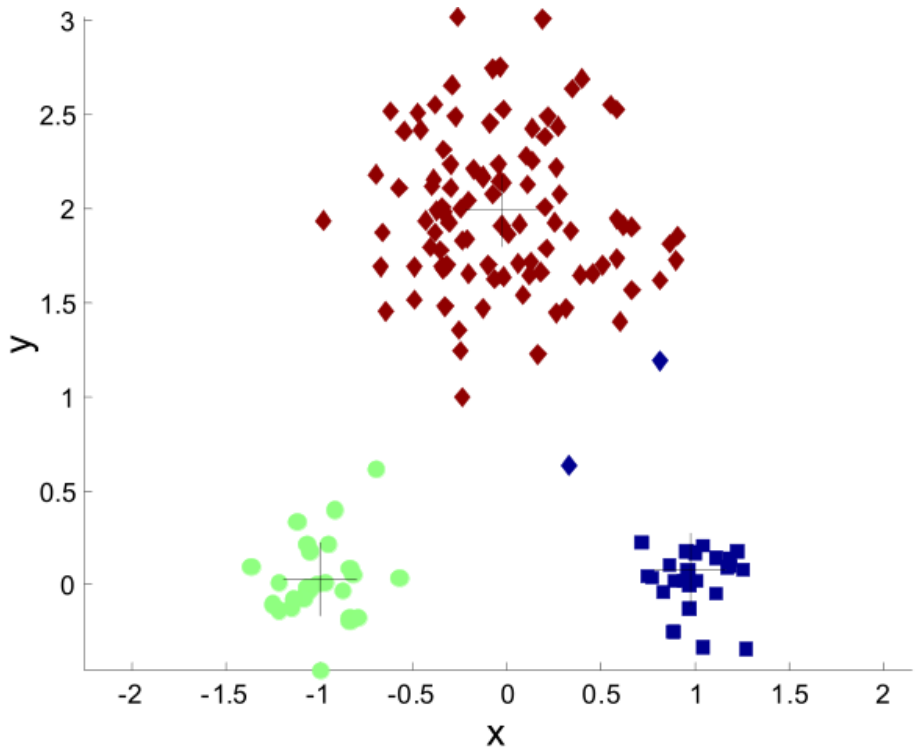
  - 此时，对SSE关于某个求偏导，显然可得

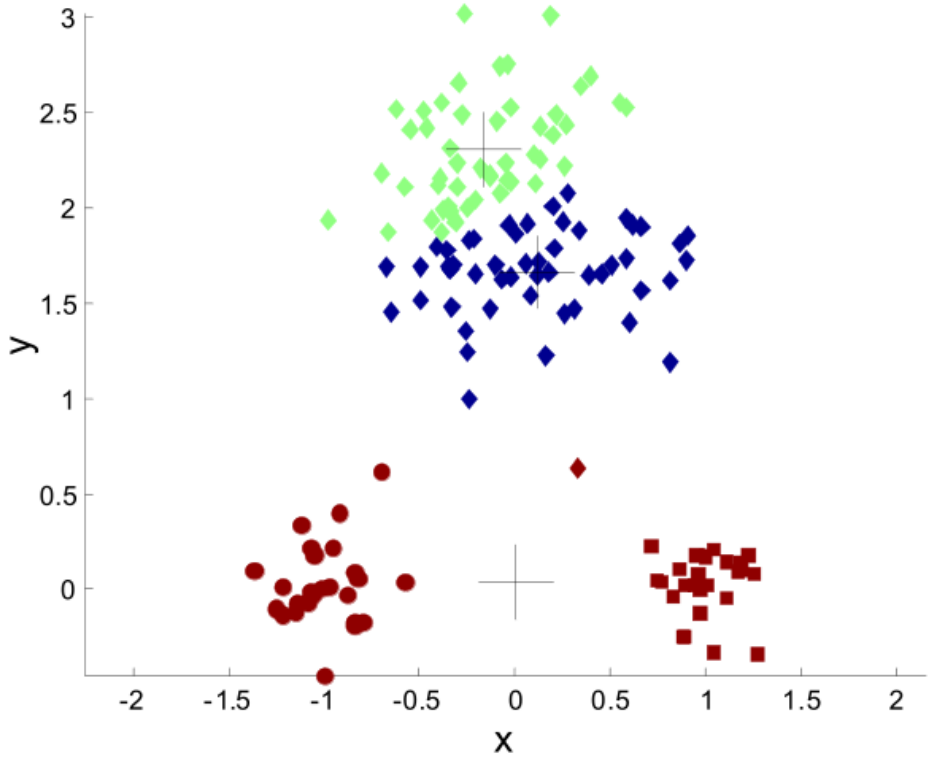
$$m_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

这个过程，也正是K均值聚类重新确定簇中心的过程

- **初始中心的选择**

  - 如前所述，不同的初始中心，可能导致截然不同的聚类结果





- **初始中心的选择**

- 如何解决初始中心选择的问题？
  - 最土的办法：多试几次。 但是试几次是个头呢？ （考验非欧的时候到了）
  - 采少数样本，借助其他聚类（如层次聚类）先确定出初始中心
    - 然而层次聚类开支较大，同时此方法仅适用于K较小的情况
  - 选择多于K个中心，然后从中挑选分隔较为明显的
  - 使用“后处理”来修补生成的簇
  - 采用二分K均值方法加以解决

- **解决方法（1）K均值的后处理**

  - 后处理的方法在于提升聚类的质量（如提升SSE）
  - 常见的后处理方法包括
    - 清除较小的、可能代表离群点的簇
    - 对较为“松弛”（如SSE较高）的簇进行拆分
    - 对较为“紧凑”（如SSE较低）的簇进行合并
    - 也可引进一个新的中心，或将一个簇完全打散（重新划分）
      - 通常，重新选择的簇中心是距离所有簇中心最远的点

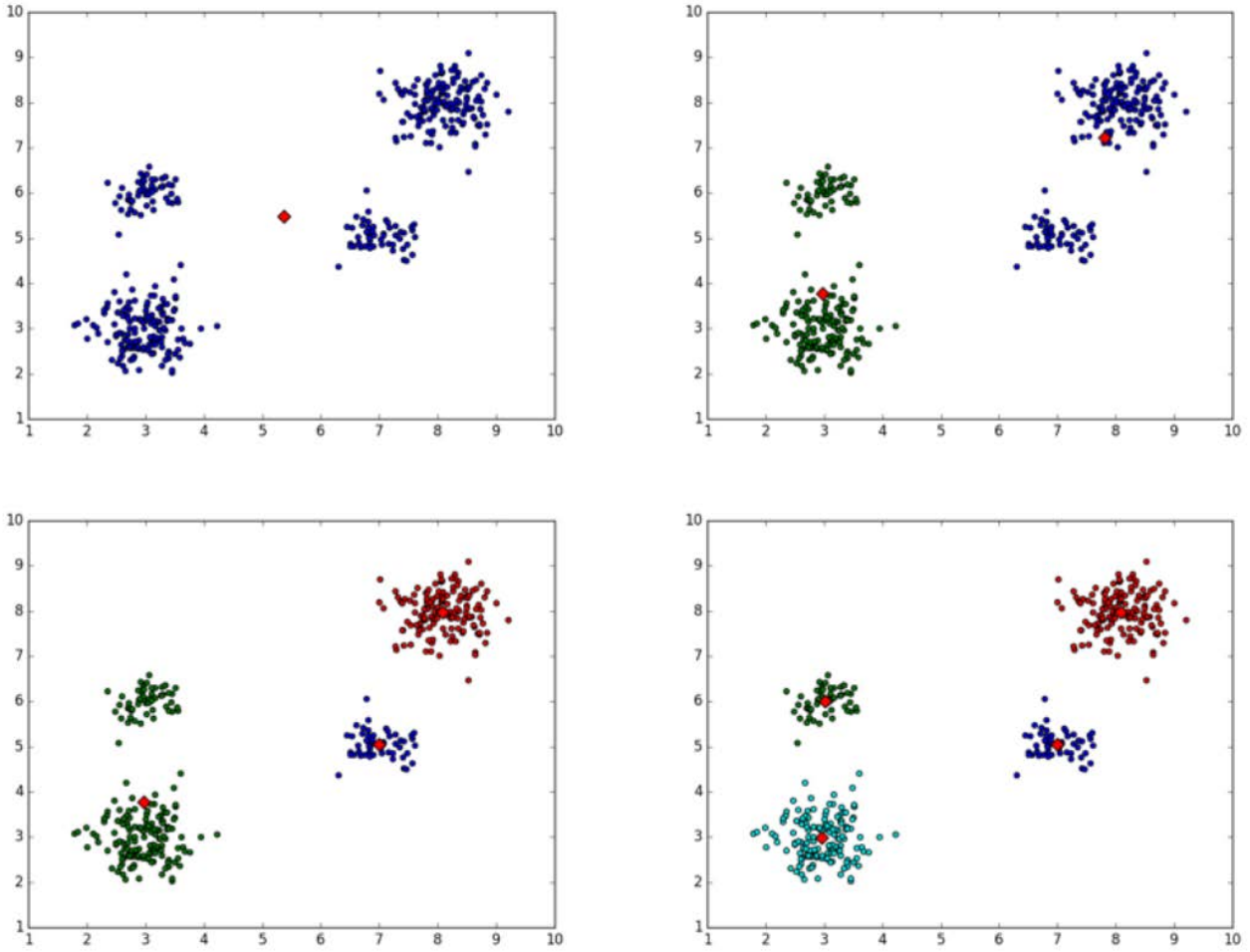
- **解决方法（2）二分K均值聚类**

  - 二分K均值聚类是K均值的一种变种，类似于一种层次聚类的思想
  - 基本思想：为了得到K个簇，先分为 2 个簇，然后不断选择其中一个分裂
    - 选择的标准可以是较大的簇，或者SSE较高的簇

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

- **解决方法（2）二分K均值聚类**

  - 一个二分K均值聚类的实例
    - 从这个实例可以看出，二分K均值受初始中心的影响不大
    - 究其原因，二分K均值可视作一个“逐步求精”的过程

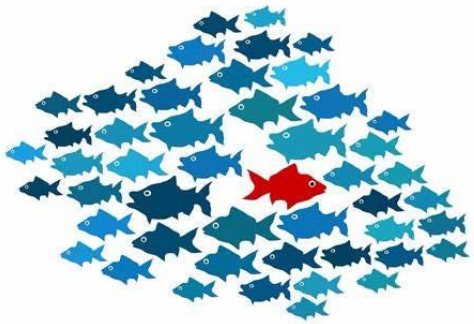


- **其他问题：空簇的处理**

- 基本K均值聚类有可能导致空簇的出现
  - 例如，所有的点在分配时都未被分配到某个簇
    - 如果以样本作为初始中心，则不会出现这种情况（簇内至少一个点）
  - 一般而言，通过新生成一个簇来替代这个空簇，思路类似后处理
    - 一种解决方法是：选择一个最远样本点新生成一个簇
    - 另一种解决方法是：将最大SSE的簇进行拆分

- **其他问题：离群点的处理**

- 在使用SSE衡量聚类结果时，离群点可能造成负面影响
  - 离群点可能影响簇的代表性，并且SSE也较高
    - 这种情况下，应该删除离群点以保障聚类质量
  - 但需要注意的是，离群点不是任何情况下都能删除
    - 有些时候，明显的离群点可能反而是我们研究的目标
      - 例如，交易中的异常或欺诈行为
      - 离群点的识别和处理可回顾离群分析部分

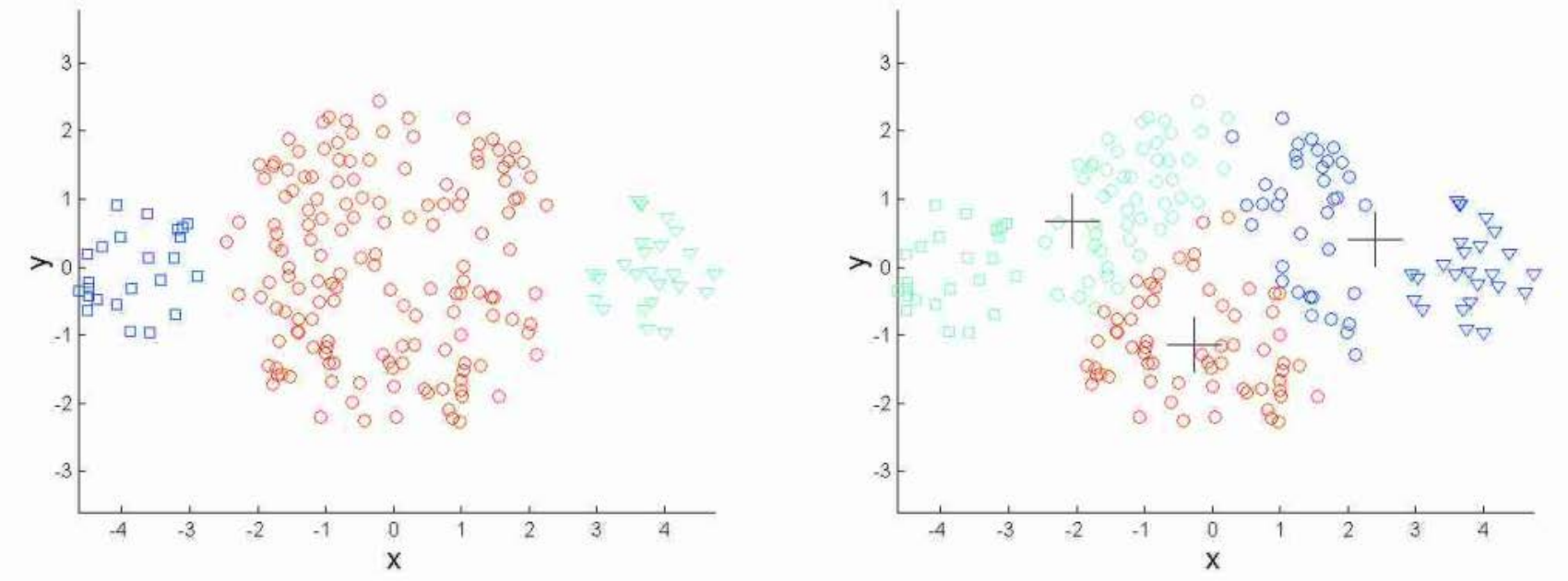


- **其他问题：中心的增量更新**

  - 在基本K均值聚类中，簇的中心在每次聚类完成后才进行
  - 事实上，可以增量地更新，例如某个样本进行重新聚类时更新
    - 此时，每步需要 0 次或 2 次中心更新（要么不更新，更新必然是一对）
    - 这种更新的好处在于不会有空簇的出现
      - 因为单点的簇必然维持原样
    - 增量更新的缺点在于可能导致次序依赖性，同时开支更大
      - 不同的处理次序可能导致不同聚类结果

- **K均值聚类的局限性**

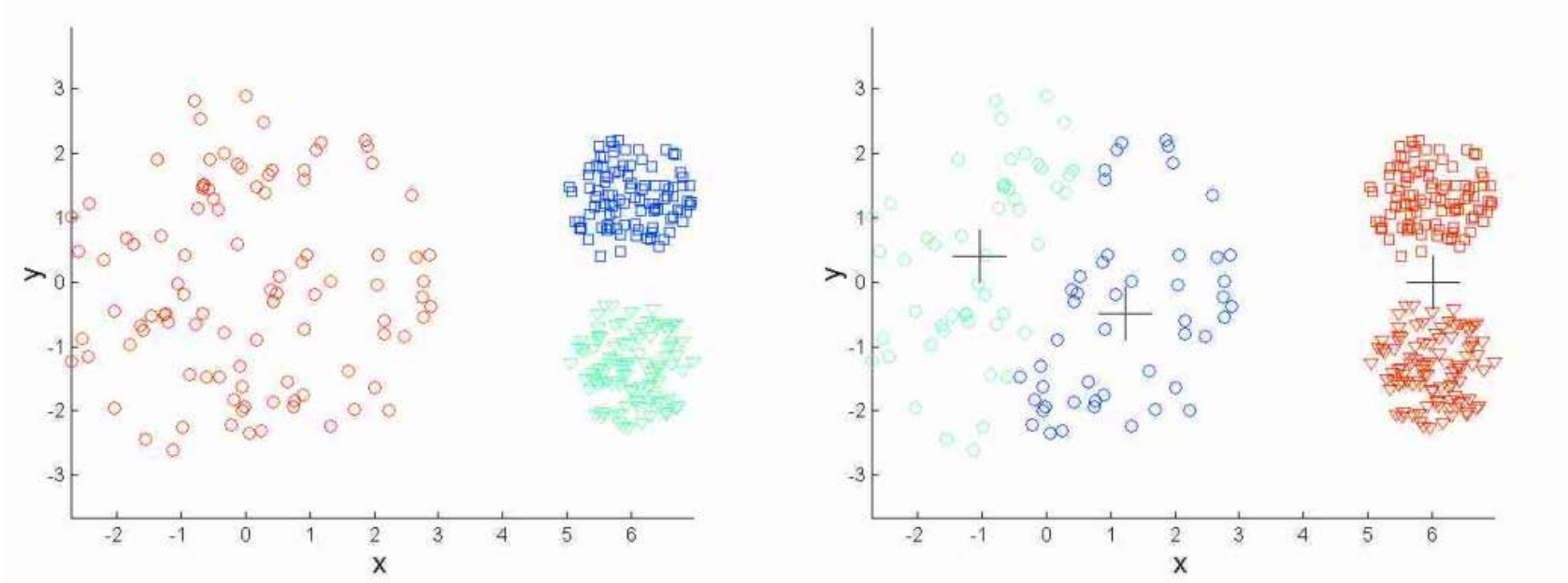
- 当簇存在不同规模、密度及不规则形状的情况下，K均值聚类效果较差



当出现规模不同的簇时，往往结果会受到一定干扰

- **K均值聚类的局限性**

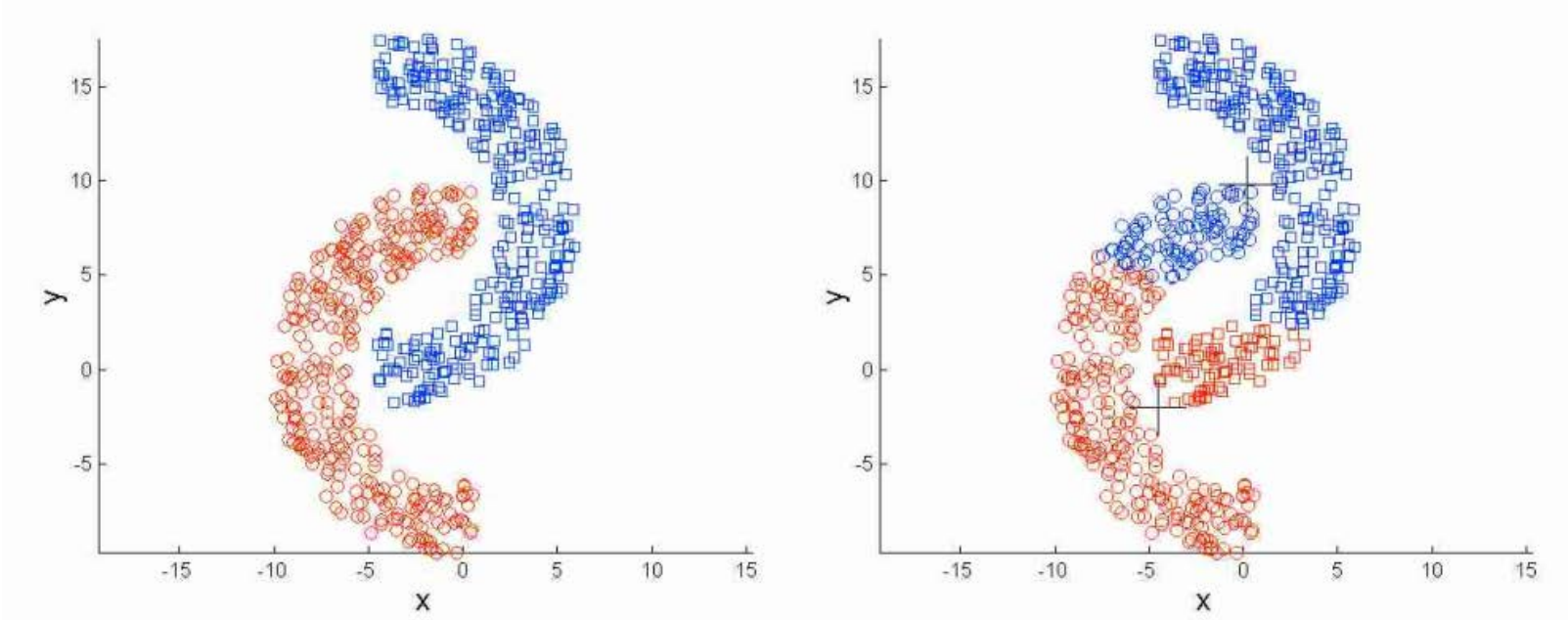
- 当簇存在不同规模、密度及不规则形状的情况下，K均值聚类效果较差



当出现密度不同的簇时，往往结果会受到一定干扰

- **K均值聚类的局限性**

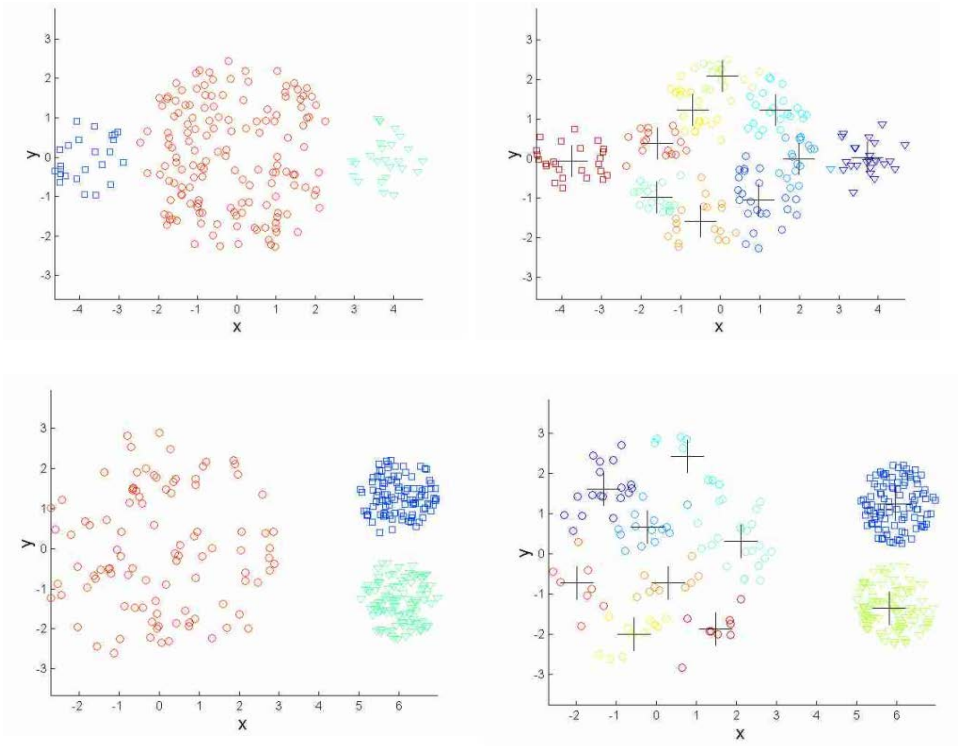
- 当簇存在不同规模、密度及不规则形状的情况下，K均值聚类效果较差

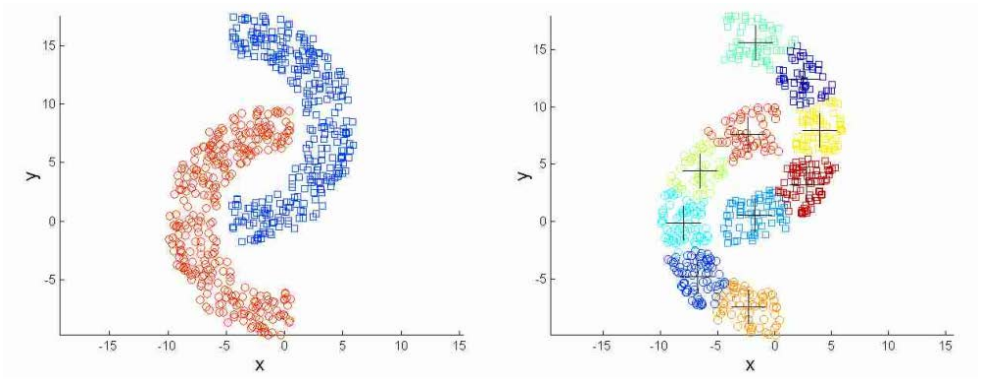


当出现不规则形状的簇时，往往很难有效聚类

- **K均值聚类的局限性**

  - 此外，K均值聚类容易受到离群点的干扰
  - 一种解决方法是：类似选择初始中心的思路，采用偏大的K，然后进行合并



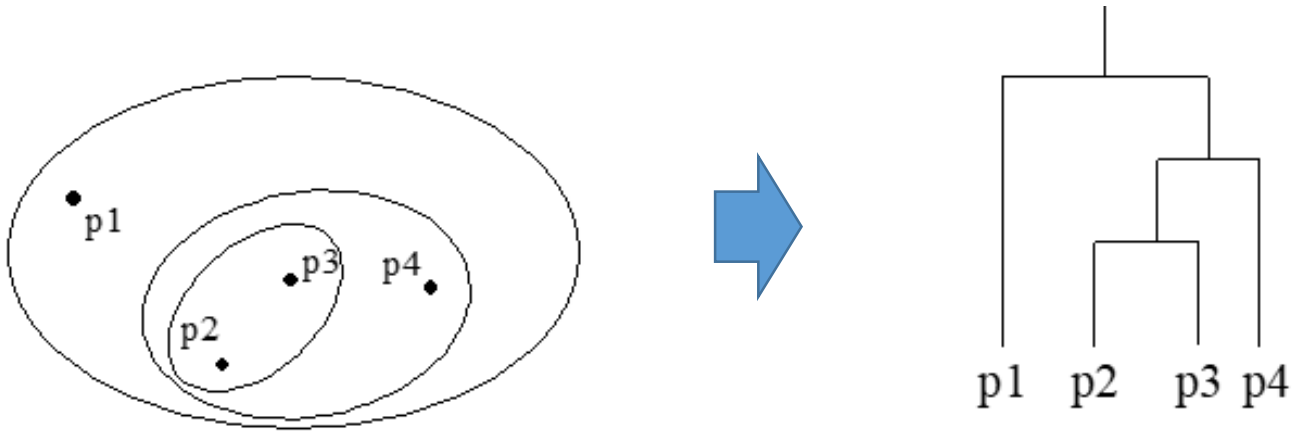


本节目录

- **常见的聚类方法**
  - K均值聚类
  - **层次聚类**
  - 基于密度聚类
  - 模糊聚类初阶
- 聚类问题的评估

- **层次聚类方法**

- 层次聚类（Hierarchical Clustering）
  - 整体呈树状结构，除叶节点外，每个节点是子节点的并集
  - 叶节点一般为单个样本组成的单元簇



- **层次聚类的优势**

  - 层次聚类不需要预设簇的数量
    - 可以根据需要随时调节，只需要在相应的层数切分树状结构即可
  - 层次聚类的结果往往可以对应到具有一定意义的分类学目录上
    - 例如，可以对应到WordNet的层次结构

```
dog, domestic dog, Canis familiaris
   => canine, canid
      => carnivore
        => placental, placental mammal, eutherian, eutherian mammal
          => mammal
            => vertebrate, craniate
              => chordate
                => animal, animate being, beast, brute, creature, fauna
                  => ...
```

- 犬 > 类犬动物 > 食肉动物 > 胎盘类哺乳动物 > 哺乳动物 > 脊椎动物…

- **两类基本的层次聚类**

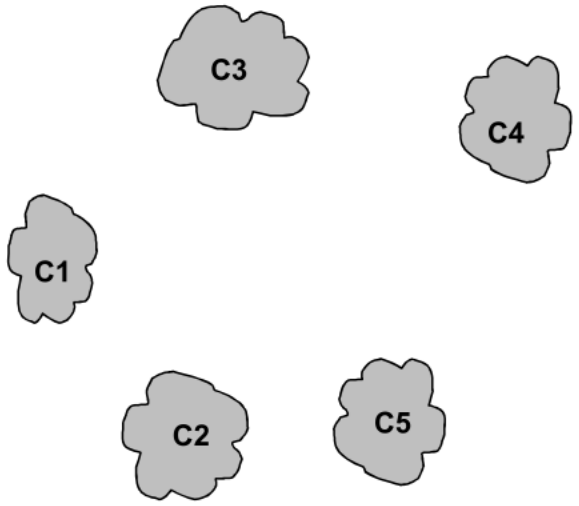
  - 通常而言，层次聚类具有以下两种基本形式
  - 凝聚式聚类（Agglomerative，自下而上）
    - 将所有样本视作个体簇，逐步合并最接近的两个簇
  - 分裂式聚类（Divisive，自上而下）
    - 从包含所有样本的完整簇开始，每一步分裂一个簇
  - 一般而言，凝聚式聚类更为常见

• 凝聚式聚类的基本流程

- 为实现凝聚式聚类，需要引入**邻近度矩阵**的概念
  - 用于存储两两簇之间的邻近度
- 凝聚式聚类的基本流程非常直观，主要迭代以下两步，直到仅剩一个簇
  1. 合并邻近度最高的两个簇
  2. 基于更新的簇重新计算邻近度，更新邻近度矩阵
- 不同的凝聚式聚类方法，区别主要在于不同的**邻近度定义**

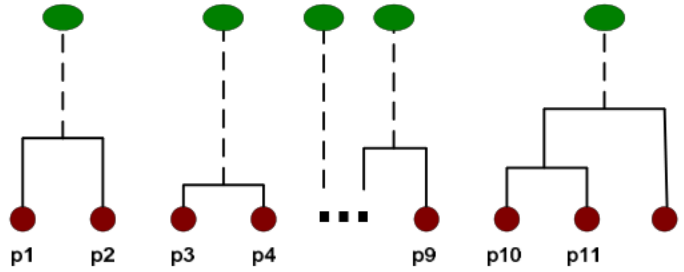
- **凝聚式聚类的实例**

  - 如下图所示，我们已经获得了五个簇，并得到了相应的邻近度矩阵



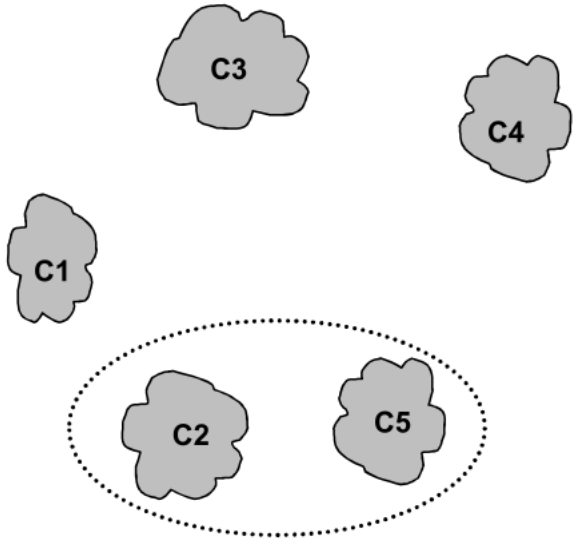
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

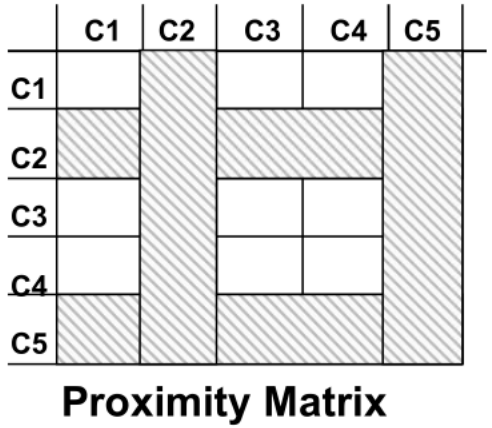
Proximity Matrix



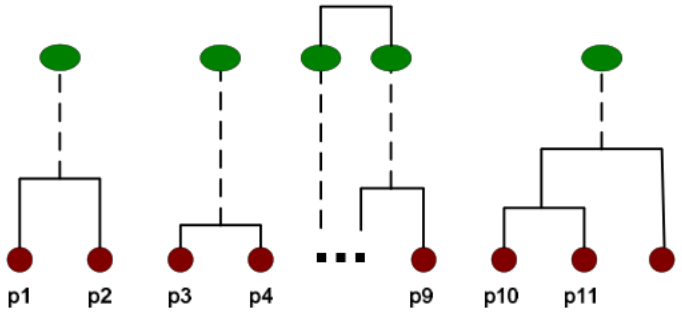
- **凝聚式聚类的实例**

- 通过邻近度矩阵，我们发现C2与C5之前的邻近度最高，可以进行合并



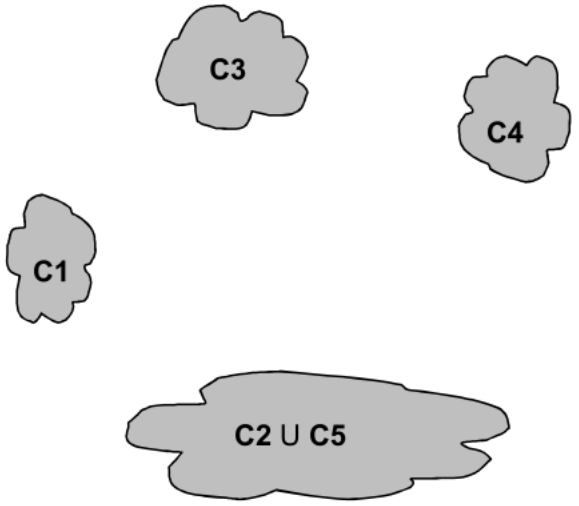


Proximity Matrix



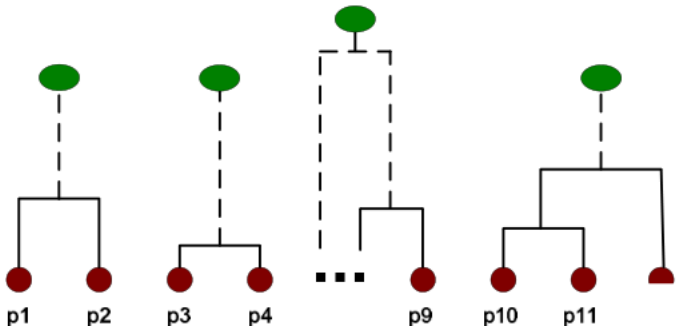
- **凝聚式聚类的实例**

  - 基于合并结果，重新计算两两簇之间的邻近度并更新邻近度矩阵



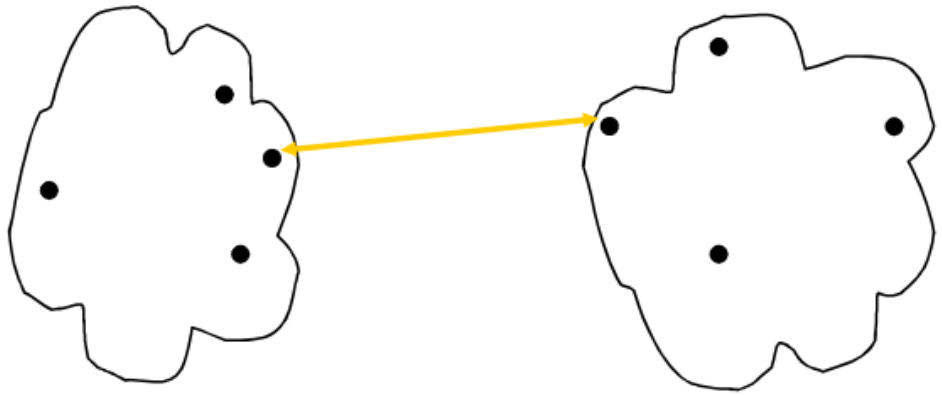
	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



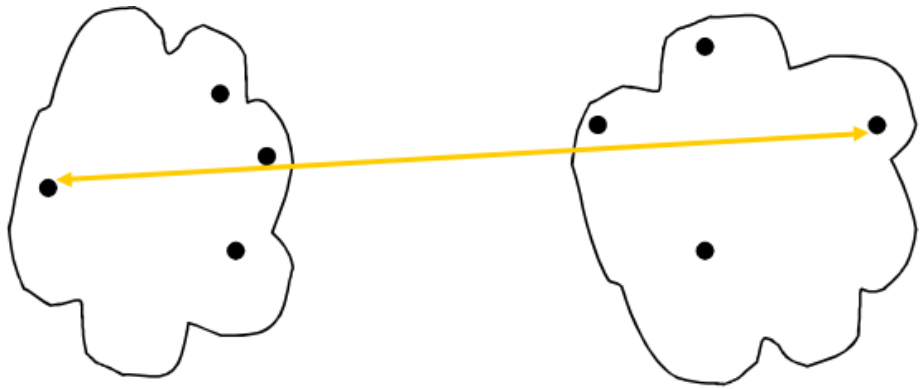
- **凝聚式聚类的邻近度定义**

- 上述过程的核心问题在于**邻近度**的计算，不同聚类方法计算方式不同
- 常见的邻近度定义包括：
  - 单链（Single Link），也可表示为MIN，指不同簇最近的点之间的邻近度
    - 单链技术擅长处理非椭圆形状的簇，但对噪声比较敏感



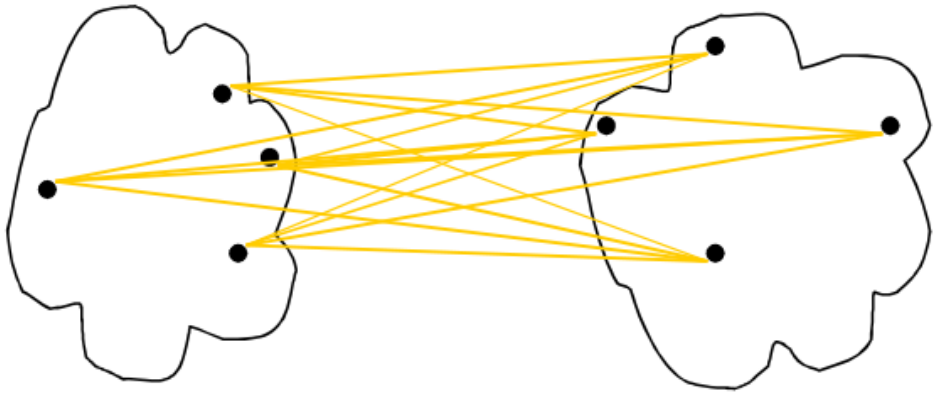
- **凝聚式聚类的邻近度定义**

- 上述过程的核心问题在于**邻近度**的计算，不同聚类方法计算方式不同
- 常见的邻近度定义包括：
  - 全链（Complete Link），也可表示为MAX，指不同簇最远的点之间的邻近度
    - 全脸对噪声不太敏感，但可能使得较大的簇变得支离破碎



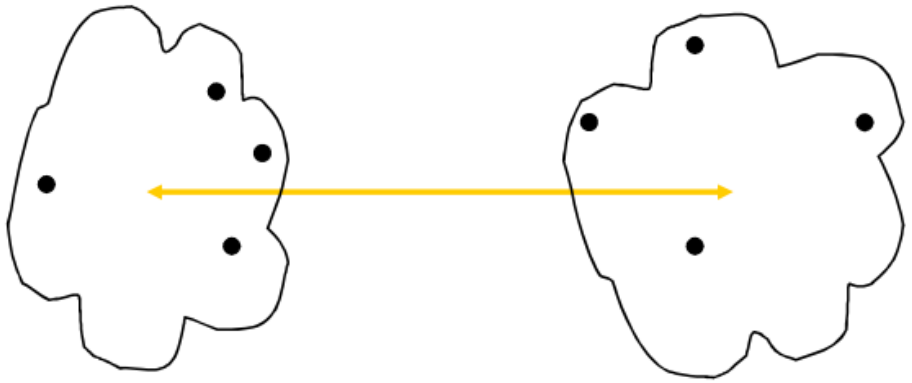
- **凝聚式聚类的邻近度定义**

- 上述过程的核心问题在于**邻近度**的计算，不同聚类方法计算方式不同
- 常见的邻近度定义包括：
  - 组平均（Group Average），指所有来自不同簇的两点之间的平均邻近度
    - 组平均方法是前面两种方法的折中产物



- **凝聚式聚类的邻近度定义**

- 上述过程的核心问题在于**邻近度**的计算，不同聚类方法计算方式不同
- 常见的邻近度定义包括：
  - 中心距离，指来自不同簇的两个簇中心之间的邻近度
    - 也可使用合并两个簇导致的SSE增加值等度量方式来衡量

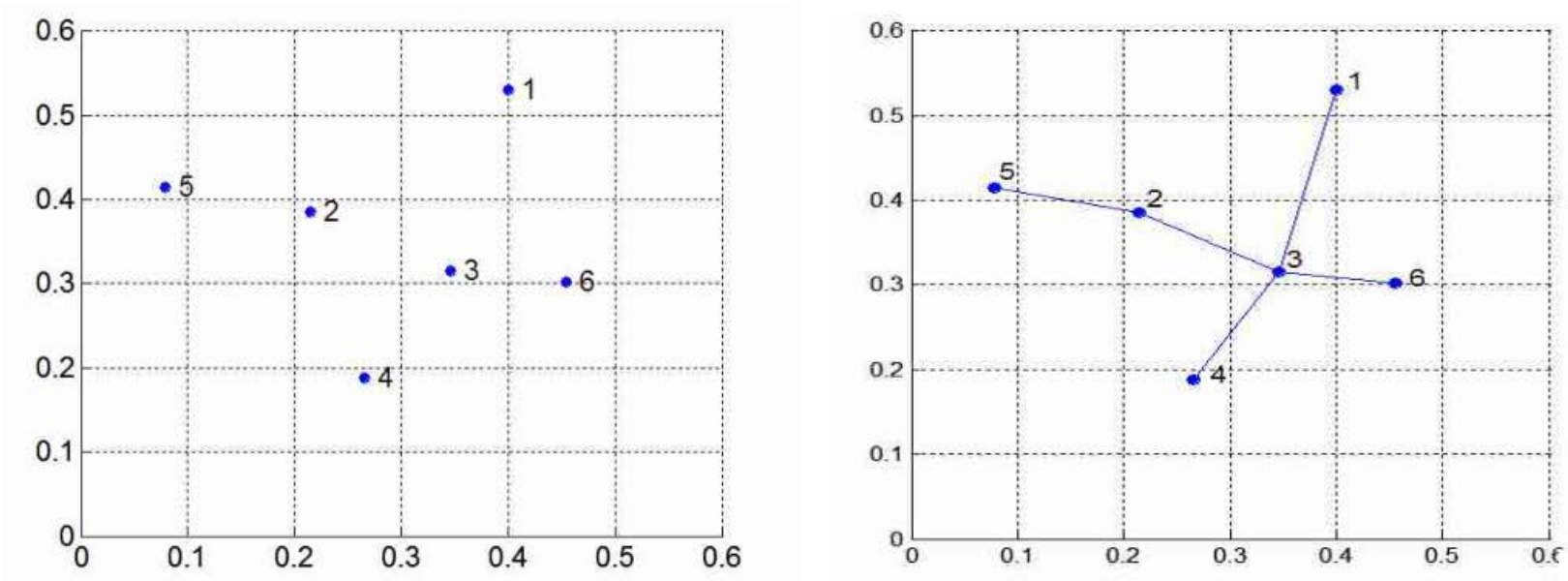


- **分裂式聚类**

  - 相比于凝聚式聚类，分裂式聚类较为罕见
    - 事实上，二分K均值聚类就是一个分裂式聚类的实例
  - 另一种分裂式聚类的代表方法是：最小生成树聚类（MST Clustering）
    1. 首先，基于差异度矩阵，生成一棵最小生成树（节点之间权值最小）
    2. 其次，每步断开差异度最大的一条边，从而创建一个新的簇
  - 一个有趣的现象：MST聚类的结果与单链凝聚聚类的结果相同，为什么？

- **最小生成树聚类的实例**

- 例如，下图左对应的样本点，可以生成下图右的最小生成树
  - 此时，每断开一条边，就可以将原来的簇一分为二



• 层次聚类的局限性

- 每一步的合并决策都是最终的
  - 一旦做出合并两个簇的决策，就无法撤销
- 没有全局的优化目标函数
  - 每一步都是一个局部最优的过程
- 不同的聚类方法（邻近度定义），或多或少都具有一些问题
  - 例如，对于噪声的敏感性，或者难以保留较大的簇等

本节目录

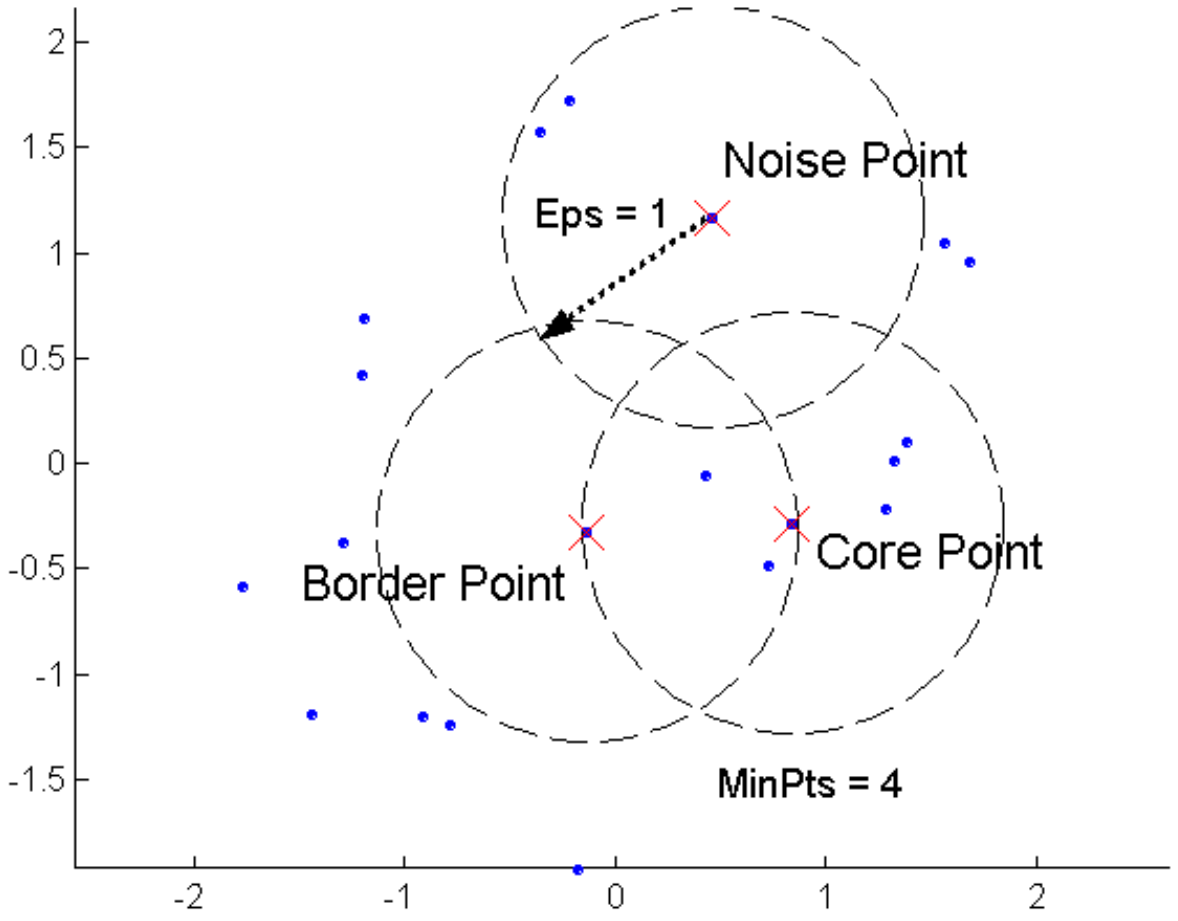
- **常见的聚类方法**
  - K均值聚类
  - 层次聚类
  - **基于密度聚类**
  - 模糊聚类初阶
- 聚类问题的评估

- **基于密度聚类的基本思想**

  - 密度聚类的基本假设在于：只有达到一定密度，才足以成为一个簇
    - 密度：指定样本一定半径的样本数量
  - DBSCAN算法，密度聚类的代表性算法
    - DBSCAN中的核心要素：三类不同的样本点
      - 核心点：稠密部分内部的点（即指定半径内样本数超过阈值的点）
      - 边界点：非核心点，但是处于稠密区域边界内/上的点
      - 噪声点：处于稀疏区域的点

- **基于密度聚类的基本思想**

  - 三类节点的示意图

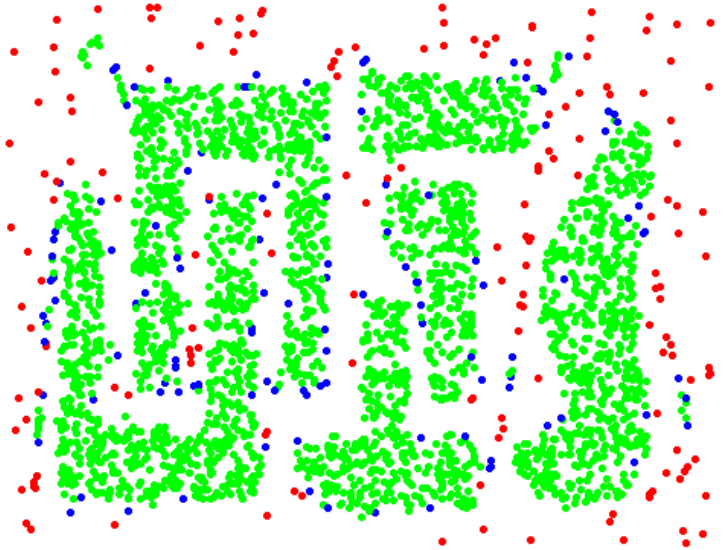


• DBSCAN的基本流程

- DBSCAN的基本流程可归纳如下

1. 将所有节点区分为核心点、边界点或噪声点
2. 删除噪声点
3. 将所有距离在预定半径内的核心点之间连一条边
4. 连通的核心点形成一个簇
5. 将所有的边界点指派到一个与之关联的核心点所在的簇中

- **DBSCAN的实例与优点**



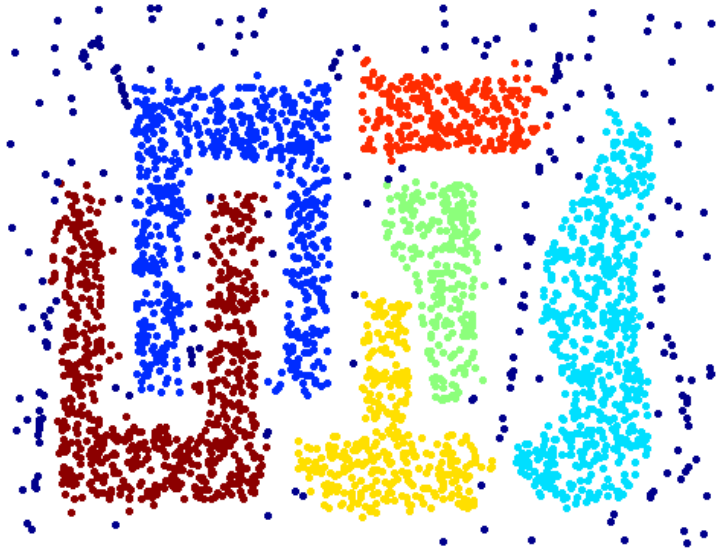
Point types: core, border and noise

Eps = 10, MinPts = 4

- **DBSCAN的实例与优点**

  - 可以看出，DBSCAN对噪声较为鲁棒，且可以处理不规则形状

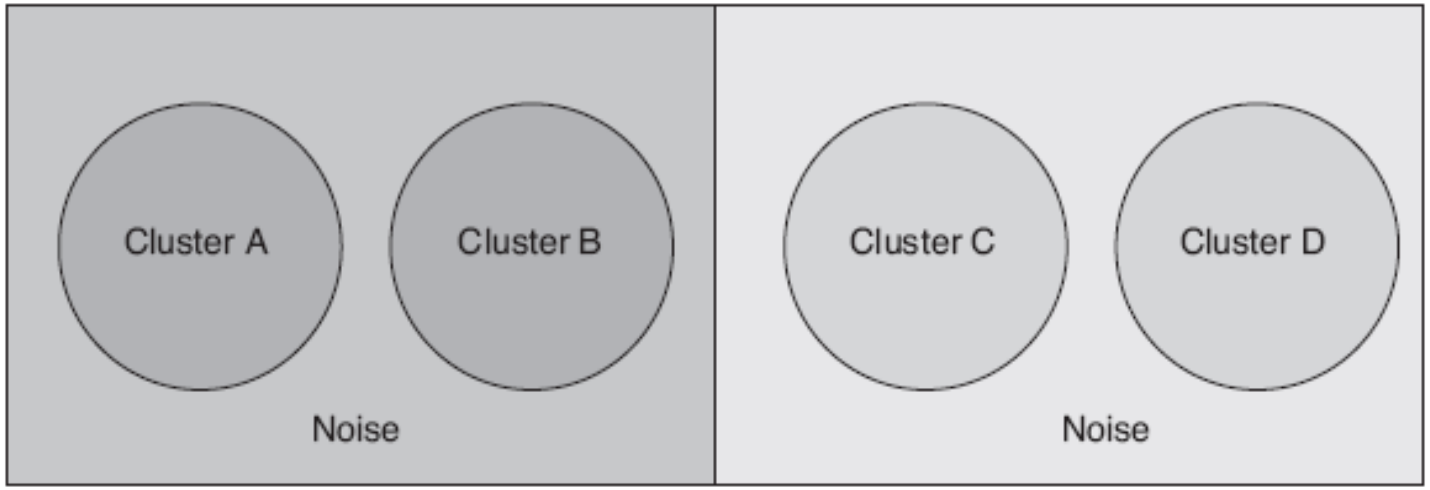
Original Points



Clusters

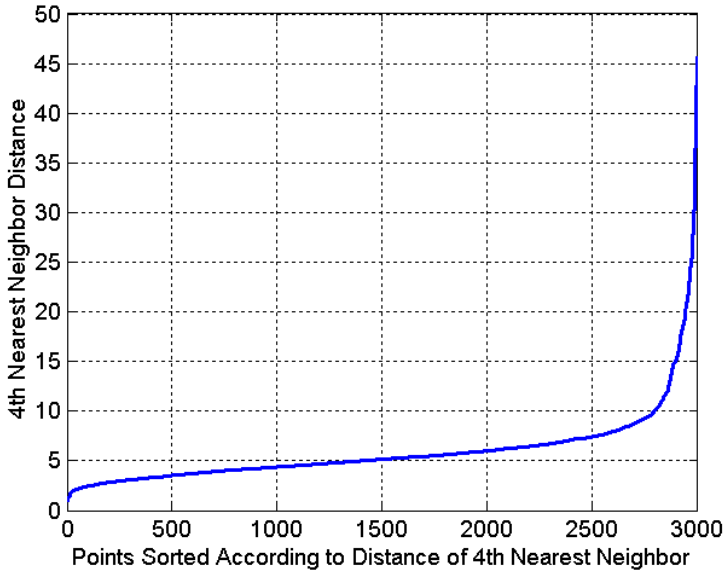
• DBSCAN的局限性

- 如果簇的密度变化很大，DBSCAN的效果可能会受到影响
  - 例如下图所示，其中左半边的噪声密度与C、D两个簇密度相同
  - 当半径阈值较低时，C、D可以被成功聚类，但A、B将与噪声合为一体
  - 反之，半径阈值较高，A、B可以被成功聚类，但C、D将被标为噪声（同左半边噪声）



• DBSCAN的参数选择

- 在DBSCAN中，核心的参数为预设半径Eps与半径内样本数阈值MinPts
  - 如果反过来看待这一问题，则Eps可近似视作MinPts-最近邻的大概距离
  - 因此，如果预先选定了样本数阈值MinPts，那么Eps可以近似根据MinPts-最近邻距离来确定
  - 例如，右图展示了当MinPts=4的情况，横轴为核心点的排序，纵轴为第4个最近邻的距离，按照从左到右的顺序展示。
  - 此时，选择出现明显拐点的位置对应的距离，作为Eps的大概取值。



本节目录

- **常见的聚类方法**
  - K均值聚类
  - 层次聚类
  - 基于密度聚类
  - **模糊聚类初阶**
- 聚类问题的评估

模糊聚类的模糊性

- 先前的聚类方法有一条基本假设：每个样本对簇的从属度非0即1
  - 即使在层次聚类中，各个类之间也仅是包含关系，而非重叠关系
- 然而在现实中，很难出现完全隶属的现象，更多是以模糊集的形式出现
  - 例如，一个人可能身兼多职，不同职位分配不同比例的时间
  - 这种情况下，该节点就是以一定从属度同时属于多个簇

- **从硬聚类到软聚类**

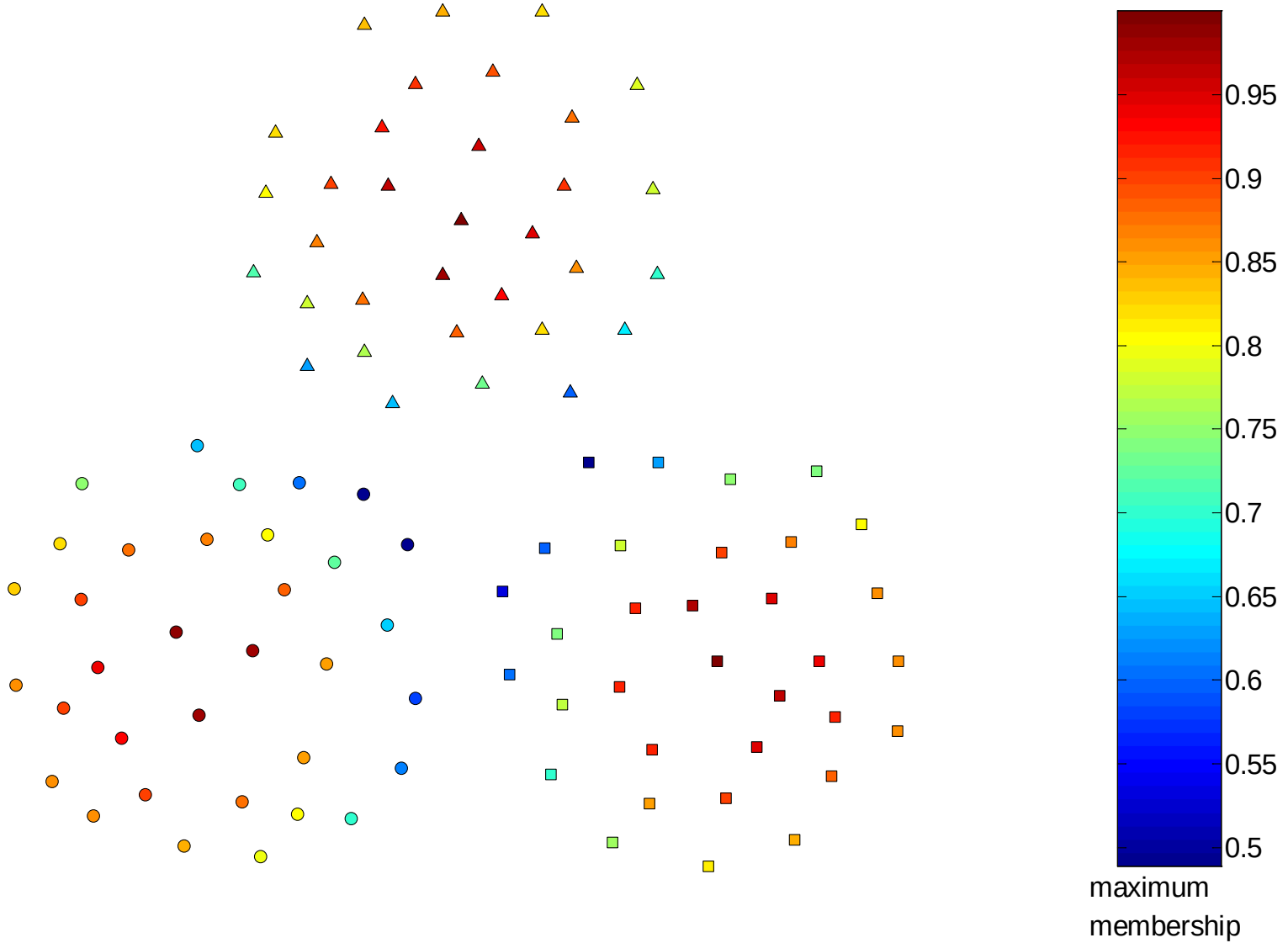
- 以K均值聚类为例，我们来观察硬聚类与软（模糊）聚类之间的区别
  - 一般的K均值聚类目标函数如右：$SSE = \sum_{j=1}^{k}\sum_{i=1}^{N} w_{ij}\left(x_i - c_j\right)^2, \qquad \sum_{j=1}^{k} w_{ij} = 1$
    - 其中，从属度 $w_{ij} \in \{0,1\}$
  - 当考虑模糊聚类时，上述目标函数可修正为：

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{N} w_{ij}^{p}\left(x_i - c_j\right)^2, \qquad \sum_{j=1}^{k} w_{ij} = 1$$

    - 其中，从属度$w_{ij} \in$ **[0,1]**

• 模糊聚类的实例

- 以模糊K均值聚类为例
  - 每次重新聚类时，不是更新样本的归属，而是更新从属度
  - 簇中心的更新，由样本向量的加权和决定



本节目录

- 常见的聚类方法
  - K均值聚类
  - 层次聚类
  - 基于密度聚类
  - 模糊聚类初阶
- **聚类问题的评估**

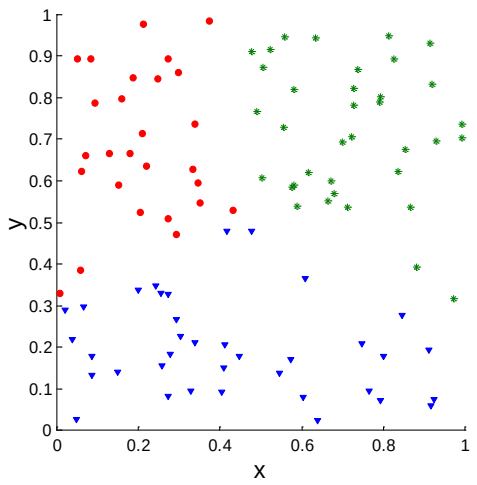
聚类问题的评估

- 如前所述，作为有监督学习，分类问题有着天然的标签可供评估使用
  - 相应的，也有各种指标用于衡量性能（可参见上节课的第一部分）
- 然而，作为无监督学习，聚类问题并没有天然标签，如何评估聚类结果？
- 首先，我们需要了解，为什么需要评估聚类结果的“好”与“坏”
  - 确定数据集的聚类趋势（Clustering Tendency），确定是否真的有群体性
  - 确定合理的簇的个数
  - 比较两个簇，或者比较基于两种方法的聚类结果，看哪种结果更合适
  - 将聚类的簇与已知的客观信息（例如，外部提供的标签、Query等）进行比较

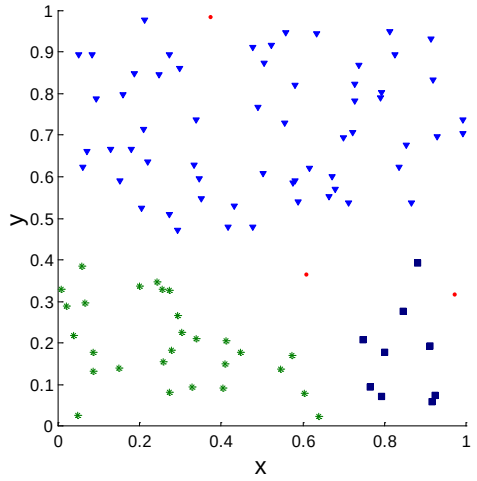
- **聚类问题的评估**

- 聚类评估的动机之一：各类聚类方法得到的结果往往很不相同

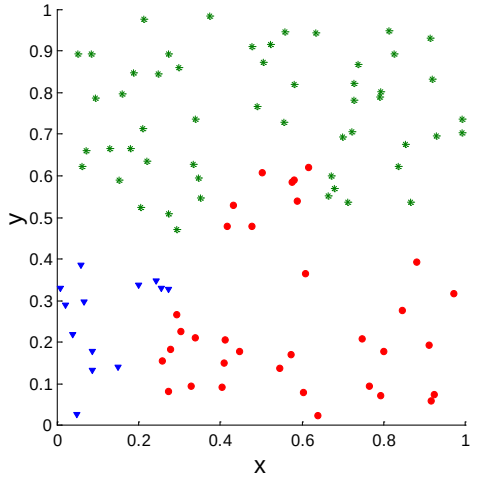
K-means



DBSCAN



Complete Link



- **聚类问题评估的类别**

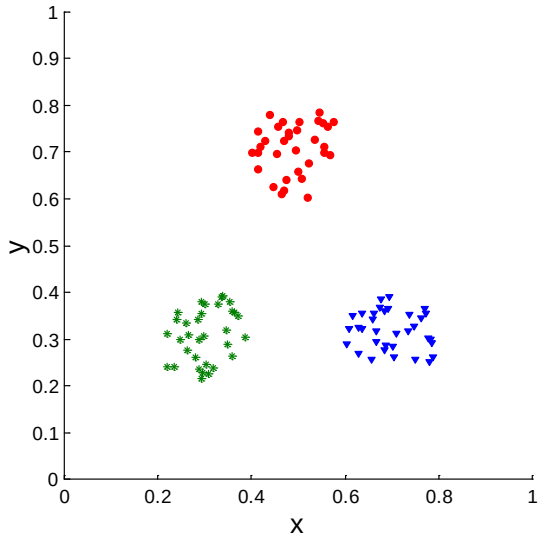
- 一般而言，聚类问题的评估标准可以分为以下三类
  - 非监督评估（或内部评估，Internal）
    - 仅使用数据本身的特性，而不考虑任何外部标签信息
  - 有监督评估（或外部评估，External）
    - 引入外部信息，衡量聚类结构与外部结果的匹配程度
  - 相对评估：主要用于比较两个簇或者两个聚类结果
    - 相对评估不是一种单独的度量类型，往往借助前两种的度量手段

• 非监督评估（1）基于邻近度矩阵

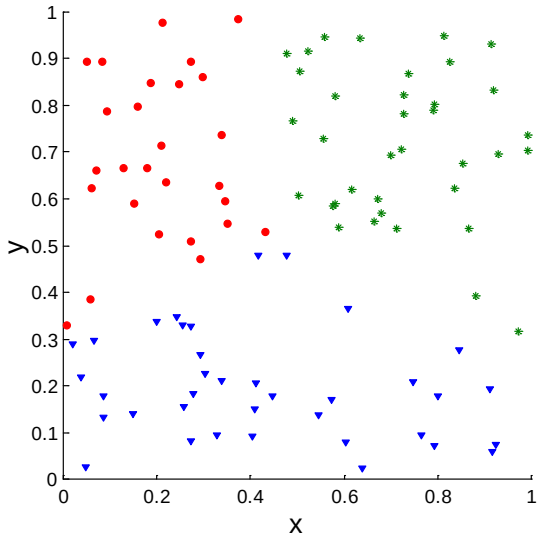
- 在层次聚类中，我们曾经借助邻近度矩阵，来判断两个簇是否需要合并
  - 邻近度越高，两个簇越应该合并。那么，最完美的簇应该是什么样呢?
    - 最理想的聚类结果应该是：簇内的点邻近度全为1，簇之间的邻近度全为0
- 由此，我们得到了所谓“理想的邻近度矩阵”定义
- 相应的，实际的邻近度矩阵与理想化矩阵的相似度，可以用于评估聚类结果
  - 然而，需要注意的是，这种方法对于部分基于密度的聚类方法不适用

• 非监督评估（1）基于邻近度矩阵

- 下图为两个基于邻近度矩阵计算相关性的例子
  - 直观上确实左图聚类效果更好，但也看出对于松散聚类不够友好



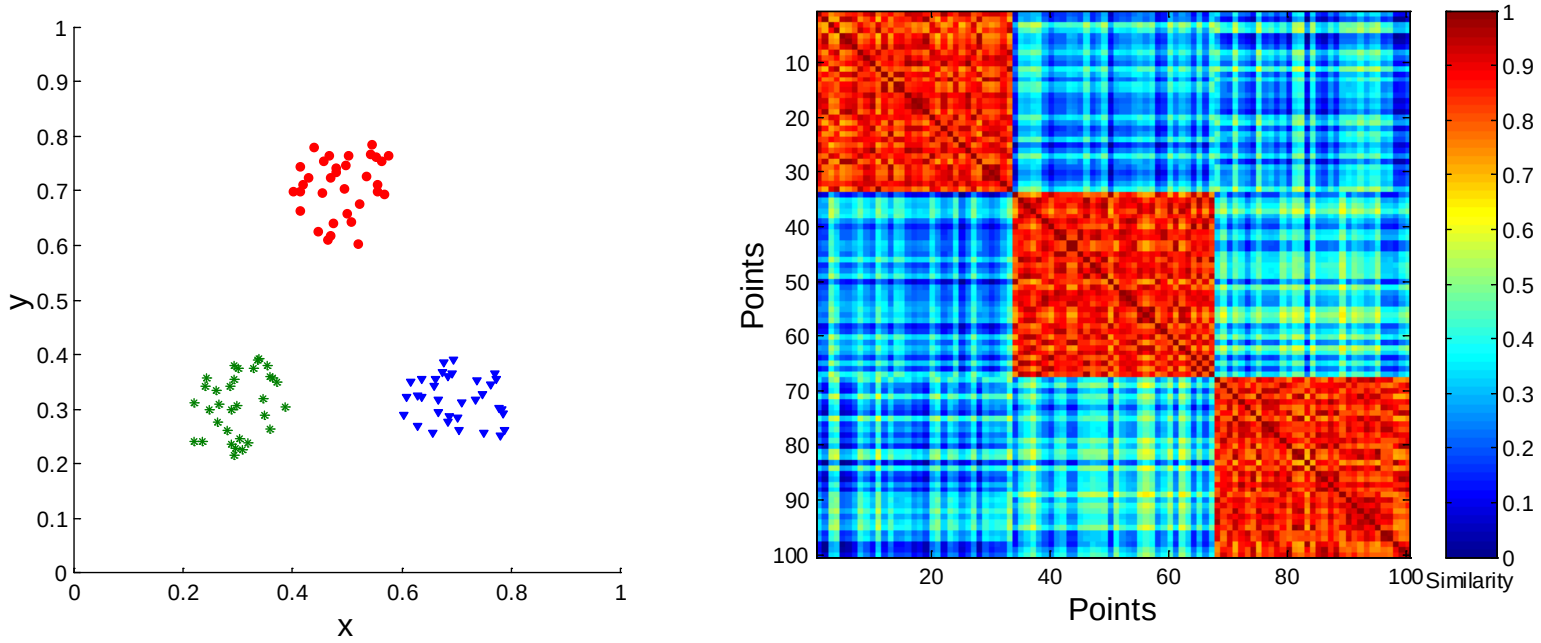
Corr = 0.9235



Corr = 0.5810

• 非监督评估（1）基于邻近度矩阵

- 通过邻近度矩阵，我们还可以可视化地评估聚类结果的好坏
  - 通过观察相似度矩阵是否体现出对角模式，可以大致判断结果好坏



• 非监督评估（2）凝聚度与分离度

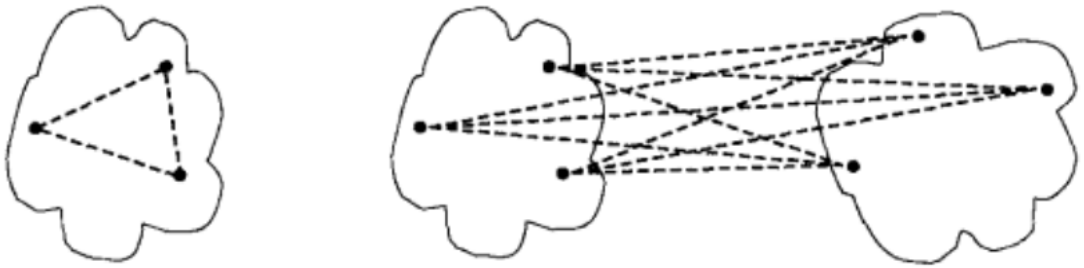
- 我们在介绍“簇”的概念时，提到过簇的基本假设
  - 簇的特点：簇内相似（距离较近），簇间相异（距离较远）
  - 可将簇内的邻近度定义为“凝聚度”（Cohesion），簇间的邻近度定义为“分离度”（Separation）。
    - 显然，凝聚度越高，分离度越低，聚类效果就越好

• 非监督评估（2）凝聚度与分离度

- 基于图和基于原型（中心）的两种度量方式
  - **基于图**，凝聚度由簇内各点邻近度之和定义，分离度由簇间各点的邻近度之和定义

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

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



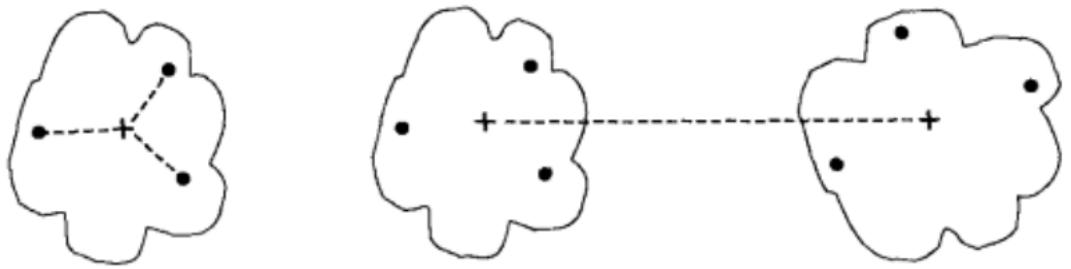
• 非监督评估（2）凝聚度与分离度

- 基于图和基于原型（中心）的两种度量方式
  - **基于原型**，凝聚度由簇内各点到中心的邻近度之和定义
  - 分离度由簇中心到其他簇内的中心（或点）的邻近度之和定义

$$cohesion(C_i) = \sum_{x \in C_i} proximity(\mathbf{x}, \mathbf{c}_i)$$

$$separation(C_i, C_j) = proximity(\mathbf{c}_i, \mathbf{c}_j)$$

$$separation(C_i) = proximity(\mathbf{c}_i, \mathbf{c})$$



• 有监督评估（1）面向分类的度量

- 在有外部标签的情况下，可以借助分类手段进行度量
  - 显然，我们希望聚类所得到的簇，能够完美对应到某个类别
    - 具体而言，主要衡量某个簇是否又单个类的成员组成
  - 因此，以下度量可用于聚类的有监督评估
    - 熵：衡量每个簇由单个类的样本所组成的程度
    - 纯度：簇在多大程度上包含单个类的对象，以最多类的比例计算
    - 此外，准确（Precision）、召回（Recall）、F值等也可使用

• 有监督评估（2）面向相似性的度量

- 在前面，我们提到过基于邻近度矩阵得到的“理想”邻近度矩阵
- 而相应的，基于分类标签，我们也可以获得分类对应的“理想”矩阵
  - 同一个类中的样本，对应的矩阵元素为1
  - 不同类中的样本，对应的矩阵元素为0
- 通过比较两个“理想”矩阵之间的相关性，可以近似估计聚类结果

- **有监督评估（2）面向相似性的度量**

  - 例如，可以定义以下元素：

f_{00} =具有不同的类和不同的簇的对象对的个数
f_{01} =具有不同的类和相同的簇的对象对的个数
f_{10} =具有相同的类和不同的簇的对象对的个数
f_{11} =具有相同的类和相同的簇的对象对的个数

  - 在这种情况下，利用Rand统计量和Jaccard系数（之前提过哦）进行评估

$$\text{Rand 统计量} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

$$\text{Jaccard 系数} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

本章小结

聚类

- 常见的聚类方法
  - K均值聚类：初始质心选择、K值选定、二分K均值
  - 层次聚类：自下而上与自上而下，邻近性度量
  - 基于密度聚类：DBSCAN方法，三类节点的区分
  - 模糊聚类初阶：模糊均值聚类方法
- 聚类问题的评估
  - 非监督/有监督评估方法

tongxu@ustc.edu.cn