聚类分析的定义是什么?

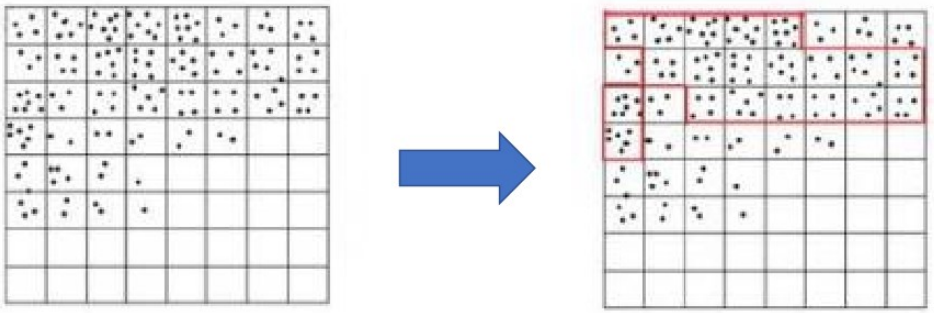
- 聚类分析：将物理或抽象对象的集合分成相似的对象类的过程称为聚类。
  - 簇：数据对象的集合；对象与同一个簇中的对象彼此相似；而与其他簇中的对象相异
  - 是一种无监督学习：没有事先定义好的类

传统的聚类算法有哪几种，它们总的优缺点是什么?

- 四种算法：基于密度的聚类、基于划分的聚类、基于层次的聚类、基于网格的聚类。
  - 还有其他类型的聚类。
- 优点：
  - 基于网格的聚类速度较快；
  - 基于密度的聚类可以获得任意形状的簇；
  - 一般来说，传统聚类算法对于**纬度较低的数据集**较有效。
- 缺点：传统聚类算法对于**纬度较高的数据集**可能不太适合。

四种传统聚类算法的原理是怎样的?

- **基于划分的聚类**
  - 给定一个由n个对象组成的数据集合，对此数据集合构建k个划分（$k \le n$），每个划分代表一个簇，即将数据集合分成多个簇的算法。
  - 要求：
    - ①每个簇至少有一个对象；
    - ②每个对象必须且仅属于一个簇
  - 典型算法：K-均值和K-中心点算法等。
- **基于层次的聚类**
  - 对给定的数据集合进行层层分解的聚类过程

    - 具体分为：凝聚法和分裂法
        - **凝聚法：**指起初每个对象被认为是一个簇，然后不断合并相似的簇，直到达到一个令人满意的终止条件；
        - **分裂法：**先把所有的数据归于一个簇，然后不断分裂彼此相似度最小的数据集，使簇被分裂成更小的簇，直到达到一个令人满意的终止条件。
    - 根据**簇间距离度量方法**的不同，层次法可分为不同的种类。常用的距离度量方法包括：最小距离、最大距离、平均值距离和平均距离等。
    - 典型算法：CURE、Chameleon和BIRCH等
- **基于密度的聚类**
    - 这类算法的思想是：只要某簇邻近区域的密度超过设定的某一阈值，则扩大簇的范围，继续聚类。
    - 这类算法可以获得**任意形状的簇**。
    - 典型算法：DBSCAN、OPTICS和 DENCLUE等。
    - 缺点：（对参数的取值敏感，R的取值需要合适）
- **基于网格的聚类**
    - 基于网格的聚类算法首先将问题空间量化为有限数目的单元，形成一个空间网格结构，随后聚类在这些网格之间进行。
    - 这类算法速度较快。
    - 典型算法：STING、WaveCluster和CLIQUE等。
    - 
    - 缺点：参数敏感、一般情况下只在低维空间有效（格子数量随维度指数级增长）

聚类分析的典型应用？

- 作为获得数据集中数据分布的工具

- 作为其他数据挖掘算法的预处理步骤

如何评价一个聚类方法的好坏?

- **好的聚类方法将产生具有以下优点的高质量聚类:**
  - 类内相似度高
  - 类间相似度低
  - 聚类结果的质量取决于该方法及其实现所使用的相似性度量
  - 聚类方法的质量也通过其发现部分或全部隐藏模式的能力来衡量
- **衡量聚类效果的方法:**
  - **不相似/相似度量**:相似性是根据距离函数表示的,通常是度量:d(i,j)
  - 有一个单独的"质量"功能可以衡量集群的"良好"。
  - 距离函数的定义对于区间比例,布尔值,分类,序数比和向量变量通常有很大的不同。
  - 权重应基于应用程序和数据语义与不同的变量关联。
  - 很难定义"足够相似"或"足够好"
  - 答案通常是高度主观的。
  - 附:二元变量相依表**(重要)**
    - 对称二元变量之间的距离度量:

$$d(i,j) = \frac{b+c}{a+b+c+d}$$

    - 非对称二元变量之间的距离度量:

$$d(i,j) = \frac{b+c}{a+b+c}$$

    - Jaccard系数(similarity measure for asymmetric binary variables):

$$sim_{Jaccard}(i,j) = \frac{a}{a+b+c}$$

    - 例子:

二元变量的相依表

		Object *j*		
		1	0	sum
Object *i*	1	a	b	a+b
	0	c	d	c+d
	sum	a+c	b+d	p

Example 7.2 **Dissimilarity between binary variables.** Suppose that a patient record table (Table 7.2) contains the attributes *name*, *gender*, *fever*, *cough*, *test-1*, *test-2*, *test-3*, and *test-4*, where *name* is an object identifier, *gender* is a symmetric attribute, and the remaining attributes are asymmetric binary.

For asymmetric attribute values, let the values *Y* (*yes*) and *P* (*positive*) be set to 1, and the value *N* (*no* or *negative*) be set to 0. Suppose that the distance between objects (patients) is computed based only on the asymmetric variables. According to Equation (7.10), the distance between each pair of the three patients, Jack, Mary, and Jim, is

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33$$
$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67$$
$$d(Mary, Jim) = \frac{1+2}{1+1+2} = 0.75$$

Table 7.2 A relational table where patients are described by binary attributes.

name	*gender*	*fever*	*cough*	*test-1*	*test-2*	*test-3*	*test-4*
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	Y	N	N	N	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- **评价聚类方法的标准：**
  - 聚类分析是一种无监督的学习，事先对给定数据集合的结构一无所知，没有利用任何先验知识。无论采用哪种聚类算法，其聚类结果的合理性和有效性都有待评价。 聚类有效性对聚类分析具有重要意义，被认为是聚类分析的一个瓶颈。

    对于相同的数据集合， 采用不同的聚类方法，可能得到不同的聚类结果。

    即便是采用同一种聚类方法，若选择不同的初始参数（如聚类数、聚类中心等）也可能会得到不同的聚类结果。
  - **可伸缩性**
    - 即算法中模式数发生变化的情况。有些算法在模式数小的条件下，算法的性能很好，但是模式数增大后，算法性能下降。如PAM算法是一种k-中心点算法，它对小的数据集合非常有效，但对大的数据集合则没有良好的可伸缩性。
  - **高维性**
    - 即算法中模式属性个数发生变化的情况。同样，有些算法只擅长处理低维数据。在高维空间中聚类是一个挑战，特别是数据有可能非常稀疏和偏斜。
  - **可解释性和可用性**
    - 知识发现过程中，聚类结果总是表现为一定的知识，这就要求聚类结果可解释、易理解。这与可视化密切相关，同时也与实际应用有关。如SOM（Self Organization Mapping）算法用于文本聚类可以产生知识地图，表现了良好的可视化性能。
  - **发现任意形状的簇**
    - 一个簇可能是任意形状的，但一般的聚类算法是基于欧氏距离和曼哈顿距离度量实现聚类，更趋于发现球状簇。在这方面，基于密度的聚类方法较好。
  - **处理噪声数据的能力**

        - 噪声数据可能是数据本身不完整，也可能是孤立点数据（Outlier）。有些算法不擅于处理孤立点数据，因此还专门出现了发现孤立点数据的算法。
    - **用于决定输入参数的领域只是最小化和输入记录顺序敏感性**
        - 一方面要求降低算法对输入参数的敏感程度，另一方面要求输入记录顺序对算法的结果影响小。如经典的k-均值算法，需要预先给出簇的数目。在一些知识发现应用中，这一参数非常影响聚类的质量。这常常是高效率算法的弱点。

K-Means算法怎么算的?

- K均值（K-means）算法是一种简便、实用的无监督聚类分析算法。
    - 这种算法在**已知簇的个数**时，可很好地实现数据的聚类分析。
- **过程:**
    - 随机选择k个数据点作为聚类中心
    - 计算其他点到这些聚类中心点的距离，通过对簇中距离平均值的计算，不断改变这些聚类中心的位置。
    - 知道这些聚类中心不再变化为止。
- **算法:**
    - 输入：n个数据的数据集合和已知的簇个数k
    - 输出：n个数据各属于k个簇中哪个簇的信息
    - 算法步骤：
        - 1）任意从n个数据中选择k个作为初始的簇中心；
        - 2）将剩余的n-k个数据按照一定的距离函数划分到最近的簇；
        - 3）repeat
        - 4）按一定的距离函数计算各个簇中数据的各属性平均值，作为新的簇中心；
        - 5）重新将n个数据按照一定的距离函数划分到最近的簇；
        - 6）until簇的中心不再变化。
- **K均值算法优势:**
    - 算法简单
    - 执行和收敛过程相对较快，是一种常见的聚类算法。
- **K均值算法局限性:**

- 必须事先知道聚类数
- 算法要求簇是密集的、簇和簇之间的差异比较大
- 数据集的平均值的计算必须有适当的定义
- 不能用于非凸面的聚类
- 对于某些孤立数据和“噪声” 点敏感等。

约束聚类怎么说?

- 有些时候，需要带约束的聚类以符合实际生活要求（自定义约束）
- **根据约束条件施加对象分类**：
  - **聚类对象的约束**
    - 这种约束条件限制参与聚类的对象，选择满足条件的对象进行聚类分析。
    - 提前到预处理阶段进行操作，将约束聚类转化为无约束聚类。
  - **障碍约束**
    - 例如：城市中存在河流、铁路、桥梁、湖泊、高山等，对城市中的对象按照空间距离聚类。
    - 通过对距离度量函进行重新定义来实现去约束化。
  - **参数约束**
    - 某些约束通过参数的形式出现在聚类算法中，如簇的数目约束
  - **簇约束**
    - 要求聚类结果中簇满足一定条件，如同一簇内相似度的偏差或簇内包含对象的总数在某个范围内等。
- **根据约束条件执行的遵守程度分类：**
  - **刚性约束条件：**聚类时必须满足的条件。
    - Must-link：约束指明哪些实例对必须聚到同一簇中；
    - Cannot-link：约束指明哪些实例对不能划分到同一簇中。
    - 处理刚性约束条件的一般策略是，在聚类的指派过程中严格遵守约束。
  - **柔性约束条件：**聚类时应最大可能满足的条件。

- 具有柔性约束的聚类是一个优化问题。
- 当聚类违反约束时，在聚类上施加一个惩罚。
- 此类聚类的最优化目标包含两部分：优化聚类质量和最小化违反约束的惩罚。
- **总体目标函数**是聚类质量得分和惩罚得分的组合。

- 例：带障碍约束的空间聚类

  带障碍约束的空间聚类是指在具有障碍物的空间数据集聚类，使得同一类中的数据相似性最大，不同类之间的数据相异性最大。

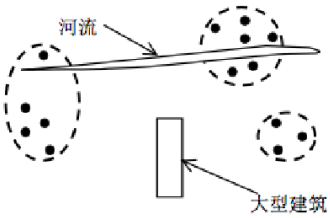
  

  图 1　不考虑障碍约束的空间聚类

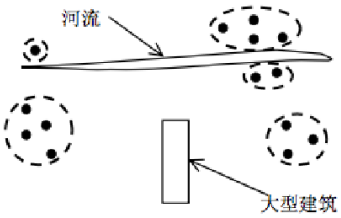
  

  图 2　考虑障碍约束的空间聚类

高维数据聚类怎么说?

- **聚类高维数据**
  - 许多应用：文本文档、DNA微阵列数据
  - 主要的挑战：
    - 许多不相关的尺寸可能会掩盖群集
    - 由于等距，距离度量变得无意义
    - 集群可能仅存在于某些子空间中（也就是**为什么要进行子空间聚类**）
  - 方法：
    - 特征转换：仅在大多数尺寸相关时才有效
      - PCA和SVD仅在要素高度相关/冗余时才有用
    - 特征选择：包装器或过滤器方法
      - 有助于找到数据具有良好簇的子空间
    - 子空间集群：在所有可能的子空间中查找集群
      - CLIQUE，ProClus和基于模式的频繁群集

- **子空间聚类相关**
  - 找到相应的子空间，进而找到在相应子空间上的簇。
  - 子空间聚类方法尽可能在数据集的不同子空间上进行聚类，每一个簇的子空间并不一定相同。
  - 子空间聚类算法也需要相应的策略用于数据集中不同簇的特征子空间的搜索，**不同的搜索策略得到的子空间往往不同，得到的聚类结果也往往不同。**
  - 根据搜索的方向的不同，可以将子空间聚类方法分成两大类：
    - **自顶向下的搜索策略**
    - **自底向上的搜索策略**
  - 特征空间搜索策略
    - 自底向上搜索策略
      - 利用了关联规则中的先验性质：频繁项集的所有非空子集也一定是频繁的，反之，所有非频繁项集的超级肯定是非频繁的。
      - 自底向上子空间聚类算法一般是基于网格密度，采用自底向上搜索策略进行的子空间聚类算法。
    - 自顶向下搜索策略
      - 首先把整个数据集划分为 k 个部分，并为各个部分赋予相同的权值；
      - 然后采用某种策略迭代的对各个部分进行不断地改进，并更新这些部分的权值；
      - 迭代直到达到某种平衡或达到某种条件停止下来
  - 子空间聚类算法
    - 自底向上算法
      - **Clique**：R Agrawal, et al. SIGMOD 1998
      - **ENCLUS**：CHCheng, et al. SIGKDD, 1999
      - **MAFIA**：S Goil, et al. 1999
      - **LTREE**：B Liu, et al. 2000
      - **CBF**：J W Chang, et al. 2000
      - **DOC**：C M Procopiuc,et a1. 2002
    - 自顶向下算法
      - Proclus：C Aggarwal，et al. SIGMOD 1999

聚类的应用范围、应用场景

- 用户市场细分（发现潜在顾客群）
- 金融领域
- 城市规划
- 模式识别
- 图像分割
- 文本聚类
- 异常检测
- 生物基因、商业选址、信息修复、语言处理、WWW……

和分类在应用上的区别

①目标不同，应用场景不同

②分类是有监督（开始时的数据集有标识）的学习过程，聚类是无监督的学习过程。

③数据集不一样，分类有训练集（带标识）和测试集（不带标识），聚类无这种区分，只有一个数据集。

④类别标识的含义不同，分类操作开始前每个类别都标清了，聚类开始前不知道

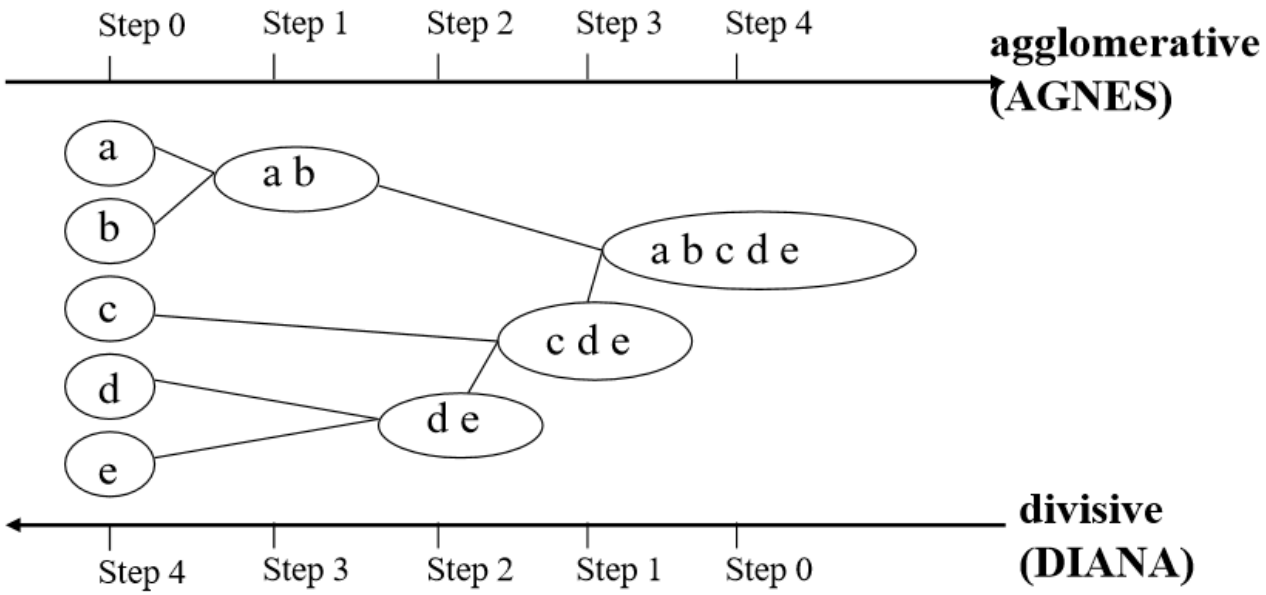
⑤方法的原理不一样：

分类：从训练集中提取隐含其中的对于类别的共同信息以分类器的模式展示出来，在数据集上完成操作；

聚类：同质分组，根相似性度量的方式对一组训练集进行聚类，要求组间尽量差距大，组内尽量差异小。

额外内容

层次方法相关



-
  - 使用距离矩阵作为聚类标准。
  - 此方法不需要输入簇数k，但需要终止条件

DBSCAN算法原理

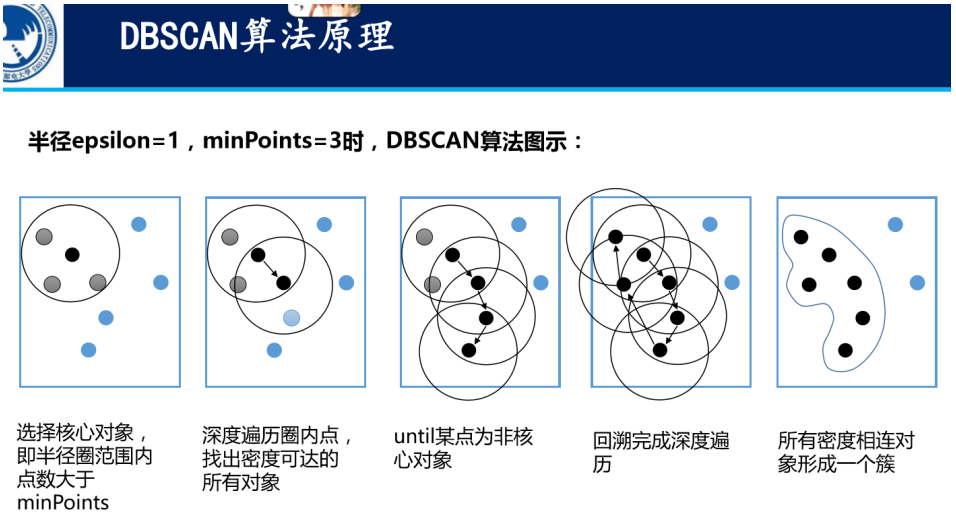


Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

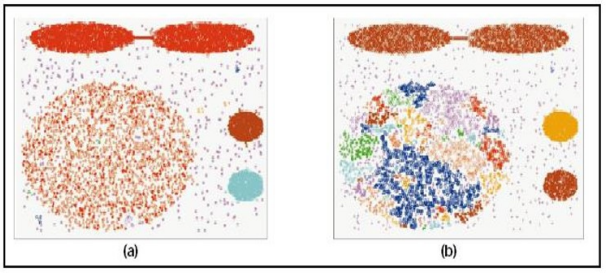
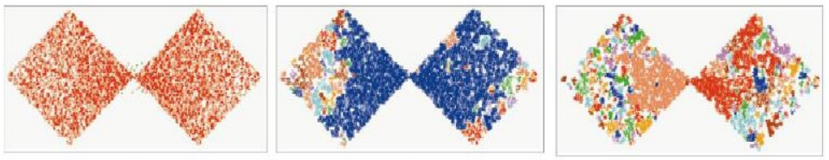


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



CLIQUE算法原理

CLIQUE（Clustering In QUEst)是一种基于网格的聚类方法，多用于**发现子空间中基于密度的簇**。

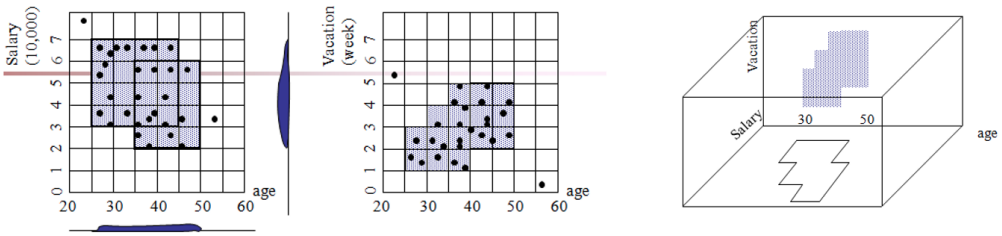
算法流程：

1. **单元划分**：把每个维划分成不重叠的区间，从而把数据对象的整个嵌入空间划分成单元。
2. **单元识别**：使用密度阈值识别稠密单元和稀疏单元。
3. **迭代连接子空间**：检查相连单元中的点数是否满足密度阈值。当没有候选产生或候选都不稠密时，迭代终止。
4. **识别簇**：使用每个子空间中的稠密单元来识别可能具有任意形状的簇。

   思想是利用最小描述长度（MDL)原理，使用最大区域来覆盖连接的稠密单元，其中最大区域是一个超矩形，落入该区域中的每个单元都是稠密的，并且该区域在该子空间的任何维上都不能再扩展。

CLIQUE识别候选搜索空间的主要策略是使用**稠密单元关于维度的单调性**。

在子空间聚类的背景下，单调性陈述如下：
一个k-维（>1）单元c至少有l个点，仅当c的每个（k-1）-维投影（它是（k-1）-维单元）至少有1个点。



分布式聚类

■ 分布式聚类算法的核心策略：

- 对分散在不同区域的局部数据进行局部聚类，构建局部模型；
- 对局部聚类的结果进行二次聚类，合并整合，以获得全局聚类的结果；
- 根据全局聚类的结果,对局部数据的归属进行调整，最终完成分布式聚类。