机器学习初步

——聚类算法

李爽

计算机学院助理教授，特别副研究员

数据科学与知识工程研究所

E-mail: shuangli@bit.edu.cn

Homepage: shuangli.xyz

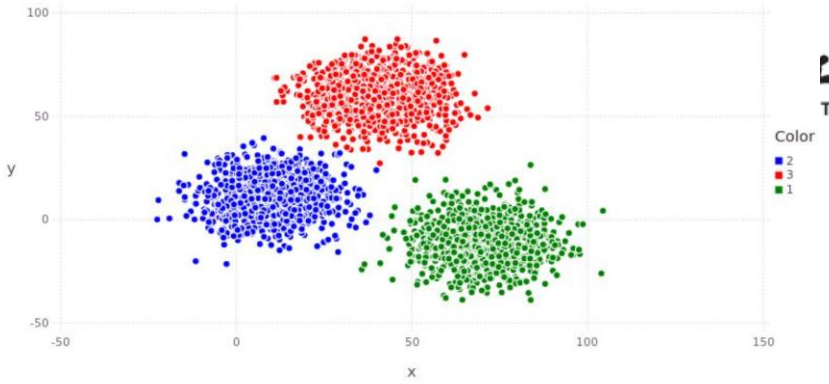
北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

一、无监督：聚类概述

- “物以类聚，人以群分”，聚类（Clustering）是人类认识世界的一种重要方法。所谓聚类就是按照事物的某些属性，把事物聚集成簇，使簇内的对象之间具有较高的相似性，而不同簇的对象之间的相似程度较差。

- 应用一：基于用户位置信息的商业选址
  - ✓ 随着信息技术的快速发展，移动设备和移动互联网已经普及到千家万户。在用户使用移动网络时，会自然的留下用户的位置信息。随着近年来GIS地理信息技术的不断完善普及，结合用户位置和GIS地理信息将带来创新应用。如百度与万达进行合作，通过定位用户的位置，结合万达的商户信息，向用户推送位置营销服务，提升商户效益。
  - ✓ 通过大量移动设备用户的位置信息，为某连锁餐饮机构提供新店选址。

无监督：聚类



一、无监督：聚类概述

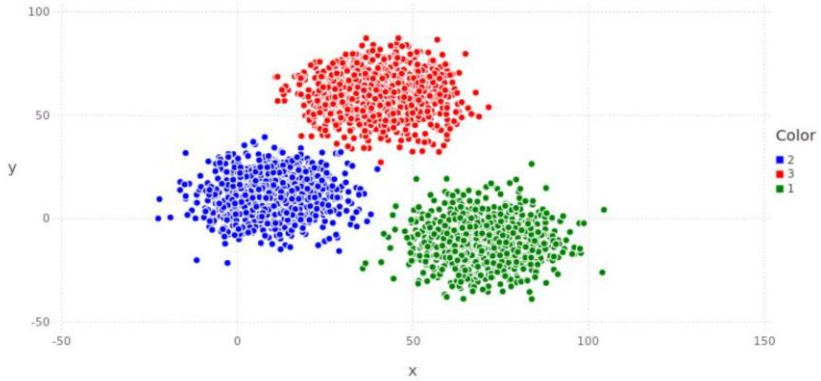
- 应用二：非人恶意流量识别

✓ 随着2016年第一季度Facebook发文称，其Atlas DSP平台半年的流量质量测试结果显示，由机器人模拟和黑IP等手段导致的非人恶意流量高达75% 。仅2016上半年，AdMaster反作弊解决方案认定平均每天能有高达 28% 的作弊流量。低质量虚假流量的问题一直存在，这也是过去十年间数字营销行业一直在博弈的问题。基于AdMaster海量监测数据，50%以上的项目均存在作弊嫌疑；不同项目中，作弊流量占广告投放5%到95%不等；其中垂直类和网盟类媒体的作弊流量占比最高；PC端作弊流量比例显著高于移动端和智能电视平台。

✓ 广告监测行为数据被越来越多地用于建模和决策，例如绘制用户画像，跨设备识别对应用户等。作弊行为，恶意曝光，网络爬虫，误导点击，甚至是在用户完全无感知的情况下被控制访问等产生的不由用户主观发出的行为给数据带来了巨大的噪声，给模型训练造成了很大影响。

✓ 希望基于给定的数据，建立一个模型来识别和标记作弊流量，去除数据的噪声，从而更好的使用数据，使得广告主的利益最大化。

无监督：聚类



一、无监督：聚类概述

- 应用三：商业数据挖掘
  - ✓ 用户分割：将用户划分到不同的组别中，并根据簇的特性而推送不同的广告；
  - ✓ 欺诈检测：发现正常与异常的用户数据，识别其中的欺诈行为。

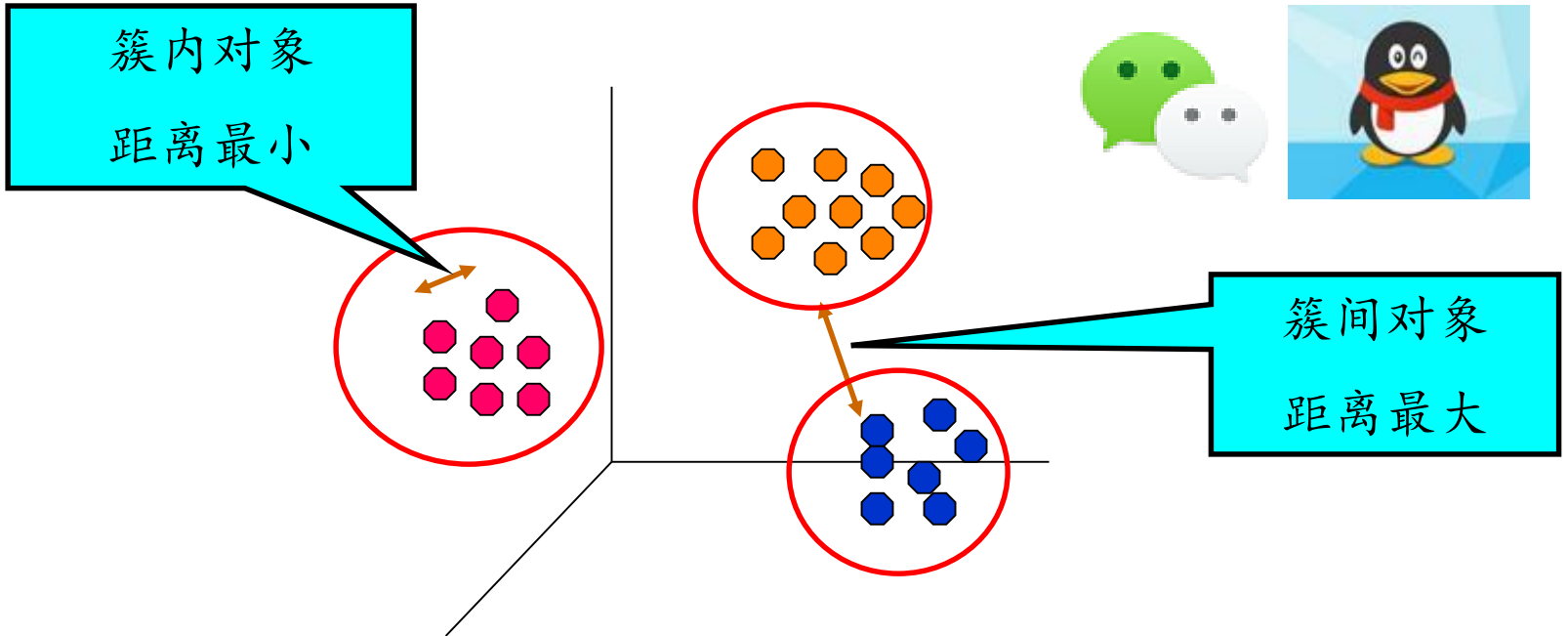
聚类问题简介：

聚类属于无监督学习的范畴，目标是将样本集划分成多个类，保证同一类的样本之间尽量相似，不同类的样本之间尽量不同，这些类称为簇（cluster）。

与分类算法的区别——聚类本质上也属于分类问题，与有监督的分类问题不同，聚类没有学习过程，直接完成对一组样本集的划分。

一、无监督：聚类概述

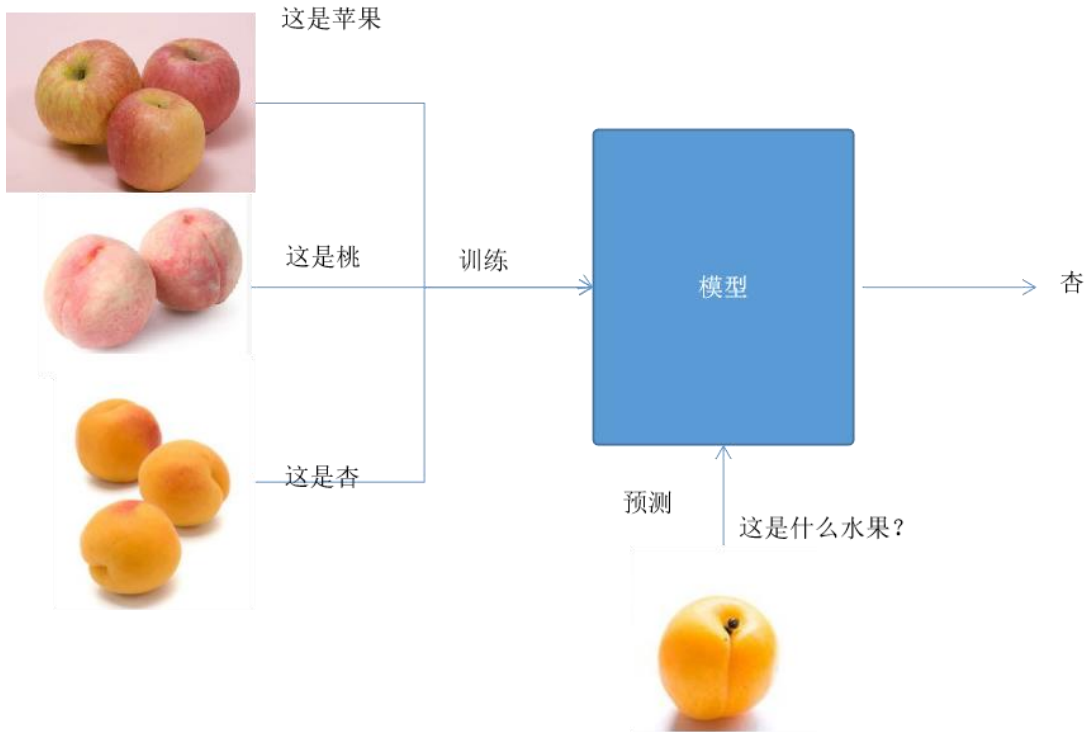
- 聚类(Clustering)：就是将一组物理的或抽象的对象，根据它们之间的相似程度，分为若干簇，相似的对象构成一组。
- 聚类可帮助用户理解数据集中的自然簇和结构。
- 簇(cluster)：聚在一起的一群对象，这些对象彼此相似。



北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

一、无监督：聚类概述

- 有监督的分类：



- 有监督的分类——先学习，再用学习得到的模型进行分类。

- 无监督的聚类：

- 无监督的聚类——直接完成对一堆水果的分类，类别事先未知，也没有学习过程。

一、无监督：聚类概述

- 聚类问题可以抽象成数学中的集合划分问题：

假设一个样本集$C = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$，聚类算法把这个样本集划分成若干个不相交的子集即簇。这些子集的并集是整个样本集：

$$C_1 \cup C_2 \ldots \cup C_m = C$$

- 每个样本只能属于这些子集中的一个，即任意两个子集之间没有交集：

$$C_i \cap C_j = \phi, \forall i, j, i \neq j$$

- 同一个子集内部的各个样本之间要很相似，不同子集的样本之间要尽量不同；
- 簇的数量m可由人工设定，也可以由算法自动确定。

一、无监督：聚类概述

- 聚类算法的核心是如何定义簇，或者说如何衡量两个样本是否相似，这里并没有统一的答案。考虑下面的整数样本集：

$$\{1,2,3,4,5,6,7,8,9\}$$

a. 可以划分成下面的两个子集：

$$\{1,3,5,7,9\}\{2,4,6,8\}$$

划分的依据是第一个子集的元素都是奇数，第二个都是偶数。

b. 也可以划分成这样：

$$\{1,4,7\}\{2,5,8\}\{3,6,9\}$$

这是按照每个数除以3之后的余数进行划分。

一、无监督：聚类概述

聚类的性能度量

- 聚类性能度量，亦称为聚类“有效性指标”（validity index）。
- 直观来讲：我们希望“物以类聚”，即同一簇的样本尽可能彼此相似，不同簇的样本尽可能不同。换言之，聚类结果的“簇内相似度”（intra-cluster similarity）高，且“簇间相似度”（inter-cluster similarity）低，这样的聚类效果较好。
- 按照这样的定义，可以将聚类的性能度量大致划分为了以下两类：

一、无监督：聚类概述

聚类的性能度量

- 外部指标 (external index)：

  将聚类结果与某个“参考模型” (reference model)进行比较。

- 内部指标 (internal index)：

  直接考察聚类结果而不用任何参考模型。

一、无监督：聚类概述

外部指标

- 外部指标的性能度量是将聚类结果与某个“参考模型”（reference model）进行比较，比如与领域专家的划分结果进行比较（其实这已经算是某种程度上对数据进行标注了），称为“外部指标”（external index）。
- 基于对参考模型权威的信任，我们可以认为参考模型对样本的划分是满足簇内相似度高且簇间相似度低的。所以对于“外部指标”，我们的度量目的就是要使得我们的聚类结果与参考模型尽可能相近，通常通过将聚类结果与参考模型结果对应的簇标记向量进行两两比对，来生成具体的性能度量。

一、无监督：聚类概述

外部指标

- 其度量的中心思想是：聚类结果中被划分到同一簇中的样本在参考模型中也被划分到同一簇的概率越高代表聚类结果越好。
- 常用的性能指标有：Jaccard系数、FM指数、Rand指数。
- 对数据集$D = \{x_1, x_2, \ldots, x_m\}$，假定通过聚类得到的簇划分为$C = \{C_1, C_2, \ldots, C_k\}$，参考模型给出的簇划分为$C^* = \{C_1^*, C_2^*, \ldots, C_s^*\}$。相应地，令$\lambda$与$\lambda^*$分别表示与$C$和$C^*$对应的簇标记向量。

一、无监督：聚类概述

我们将样本两两配对考虑，定义：

$$a = |SS|, SS = \left\{ (x_i, x_j) \mid \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j \right\}$$

SS包含了在C中隶属于相同簇，且在C^*中也隶属于相同簇的样本对

$$b = |SD|, SD = \left\{ (x_i, x_j) \mid \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j \right\}$$

SD包含了在C中隶属于相同簇，但在C^*中不隶属于相同簇的样本对

$$c = |DS|, DS = \left\{ (x_i, x_j) \mid \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j \right\}$$

DS包含了在C中隶不属于相同簇，但在C^*中也属于相同簇的样本对

$$d = |DD|, DD = \left\{ (x_i, x_j) \mid \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j \right\}$$

DD包含了在C中不隶属于相同簇且在C^*中也不隶属于相同簇的样本对

一、无监督：聚类概述

$$JC = \frac{a}{a+b+c}$$ Jaccard系数（Jaccard Coefficient, JC）

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$$ FM指数（Fowlkes and Mallows Index, FMI）

$$RI = \frac{2(a+b)}{m(m-1)}$$ Rand指数（Rand Index, RI）

指标数值在[0, 1]区间内，且数值越大越好。

一、无监督：聚类概述

内部指标：

考虑聚类结果的簇划分$C = \{C_1, C_2, \ldots, C_k\}$，定义簇$C$内样本间的平均距离：

$$\text{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \le i \le j \le |C|} \text{dist}(x_i, x_j)$$

簇C内样本间的最远距离：

$$\text{diam}(C) = \max_{1 \le i \le j \le |C|} \text{dist}(x_i, x_j)$$

簇C_i与簇C_j最近样本间的距离：

$$d_{min}(C) = \min_{x_i \in C_i, x_j \in C_j} \text{dist}(x_i, x_j)$$

簇C_i与簇C_j中心点间的距离：

$$d_{cen}(C) = \text{dist}(\mu_i, \mu_j)$$

一、无监督：聚类概述

DB指数（Davies-Bouldin Index, DBI）：

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{j \neq i}\left(\frac{\text{avg}(C_i)+\text{avg}(C_j)}{d_{\text{cen}}(\mu_i,\mu_j)}\right)$$

越小越好

Dunn指数（Dunn Index, DI）：

$$DI = \min_{1\le i\le k}\left\{\min_{j\neq i}\left(\frac{d_{min}(C_i,C_j)}{\max_{1\le l\le k}\text{diam}(C_l)}\right)\right\}$$

越大越好

一、无监督：聚类概述

聚类的距离度量

- 聚类分析中如何度量两个对象之间的相似性呢？
- 一般有两种方法，一种是对所有对象作特征投影，另一种则是距离计算。
- 前者主要从直观的图像上反应对象之间的相似度关系，而后者则是通过衡量对象之间的差异度来反应对象之间的相似度关系。

一、无监督：聚类概述

聚类的距离度量

距离度量的性质：

非负性： $\mathrm{dist}(x_i, x_j) \geq 0$

同一性： $\mathrm{dist}(x_i, x_j) = 0$ 当且仅当 $x_i = x_j$

对称性： $\mathrm{dist}(x_i, x_j) = \mathrm{dist}(x_j, x_i)$

直递性： $\mathrm{dist}(x_i, x_j) \leq \mathrm{dist}(x_i, x_k) + \mathrm{dist}(x_k, x_j)$

常用距离：

闵可夫斯基距离（Minkowski distance）：

$$\mathrm{dist}(x_i, x_j) = \left(\sum_{u=1}^{n} |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}}$$

p=2：欧氏距离（Euclidean distance）

p=1：曼哈顿距离（Manhattan distance）

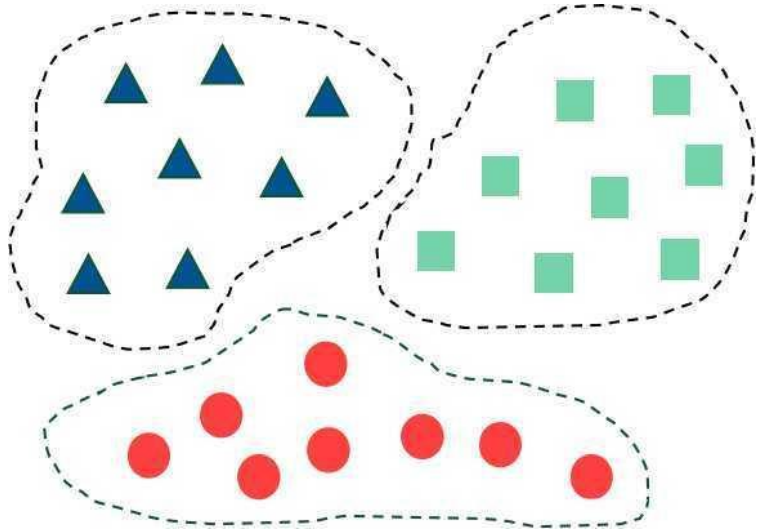
一、无监督：聚类概述

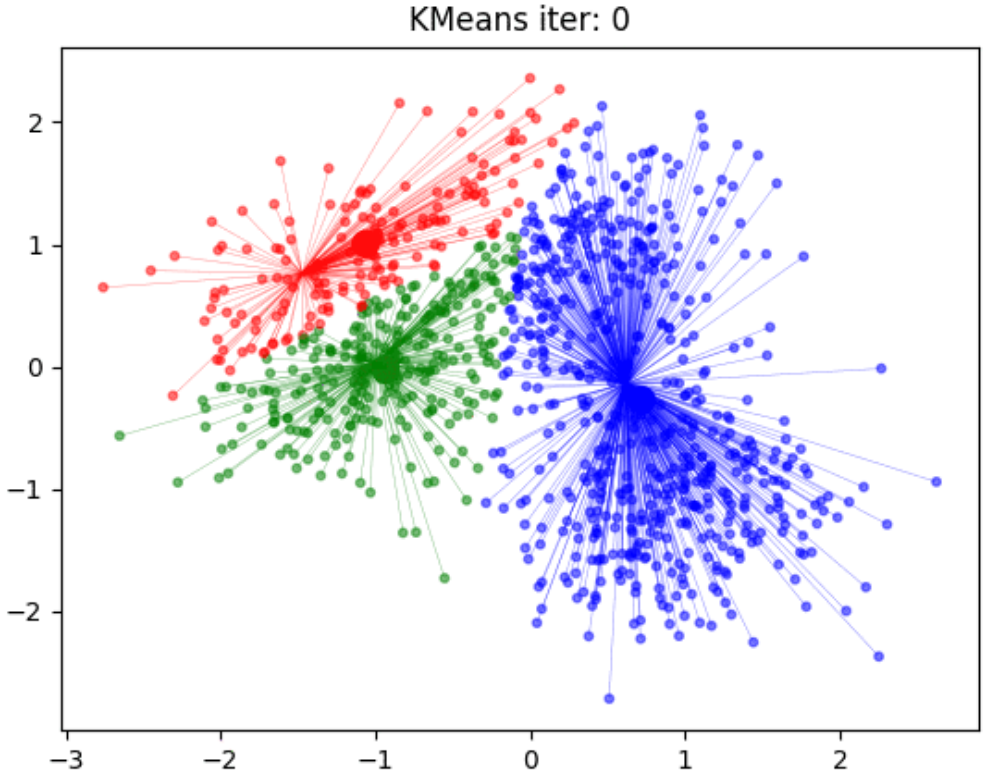
- 按照聚类分析算法主要思路的不同，聚类算法可以分为：
- 划分聚类方法
- 层次聚类方法
- 基于密度的聚类算法
- 基于网格的聚类算法
- 基于模型的聚类算法。

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

一、无监督：聚类概述

(1)划分聚类方法。对于给定的数据集，划分聚类方法通过选择适当的初始代表点将数据样本进行初始聚类，之后通过迭代过程对聚类的结果进行不断的调整，直到使评价聚类性能的准则函数的值达到最优为止。

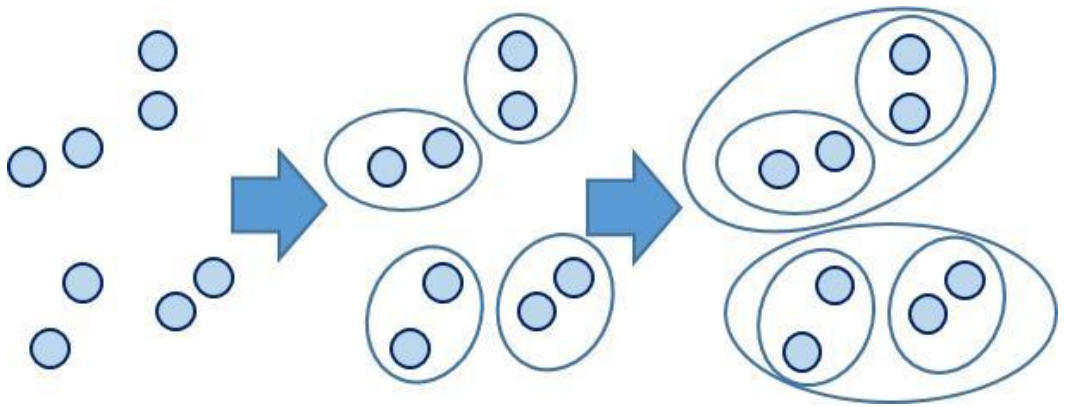


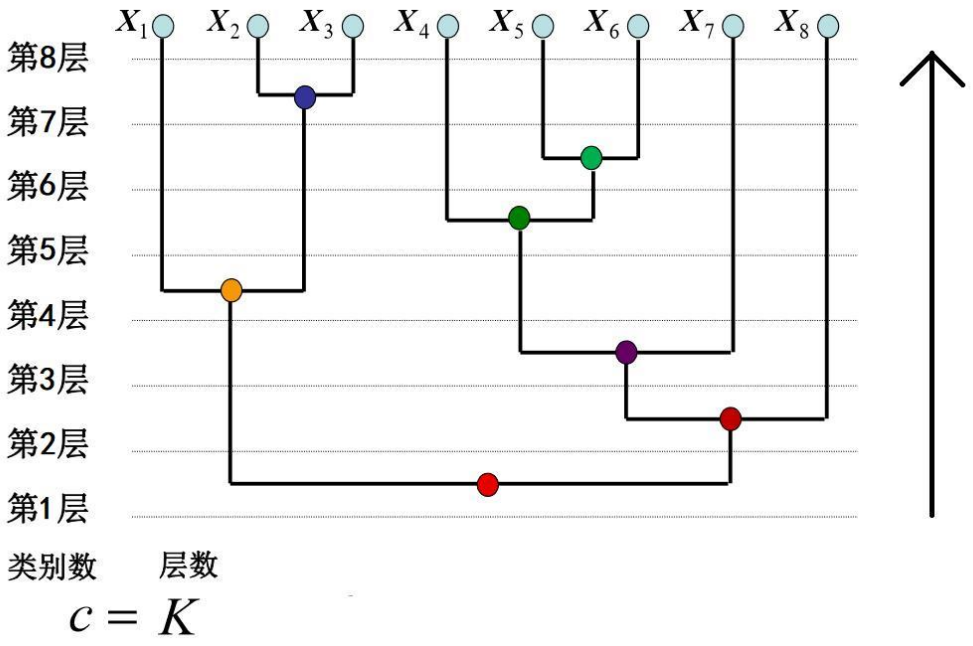


北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

一、无监督：聚类概述

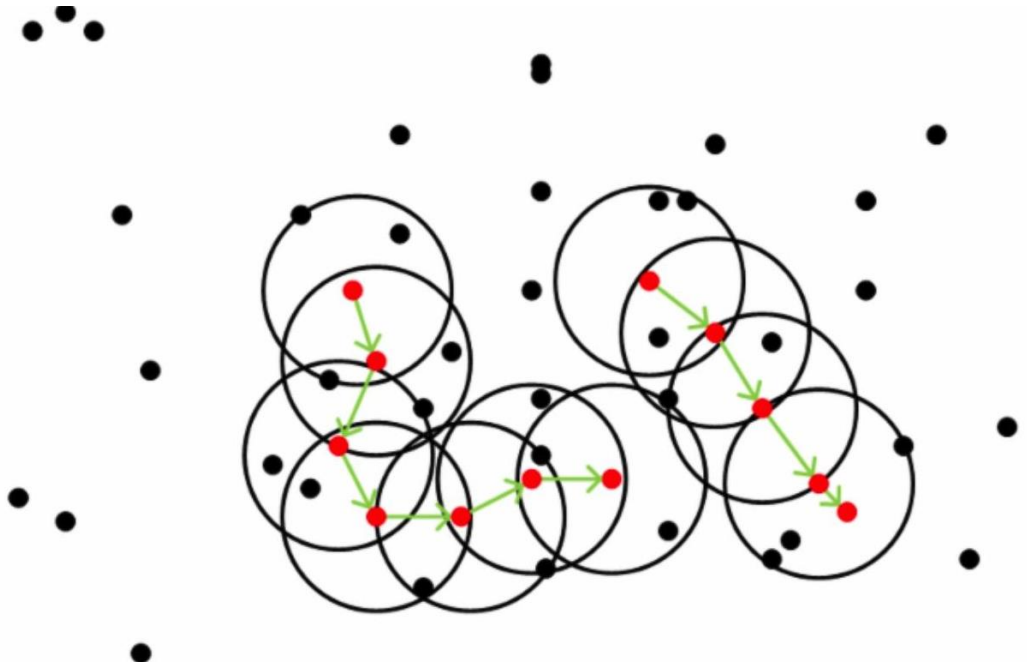
(2)层次聚类方法。层次聚类方法将给定数据集分层进行划分，形成一个以各个聚类为结点的树型结构。层次聚类方法分为自底向上（凝聚型层次聚类）和自顶向下（分解型层次聚类）两种方式。

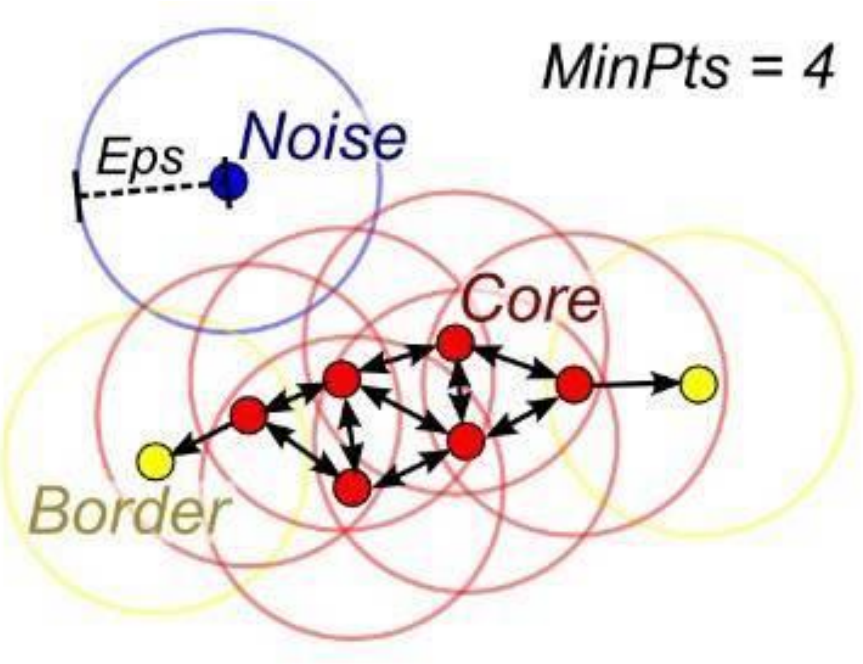




一、无监督：聚类概述

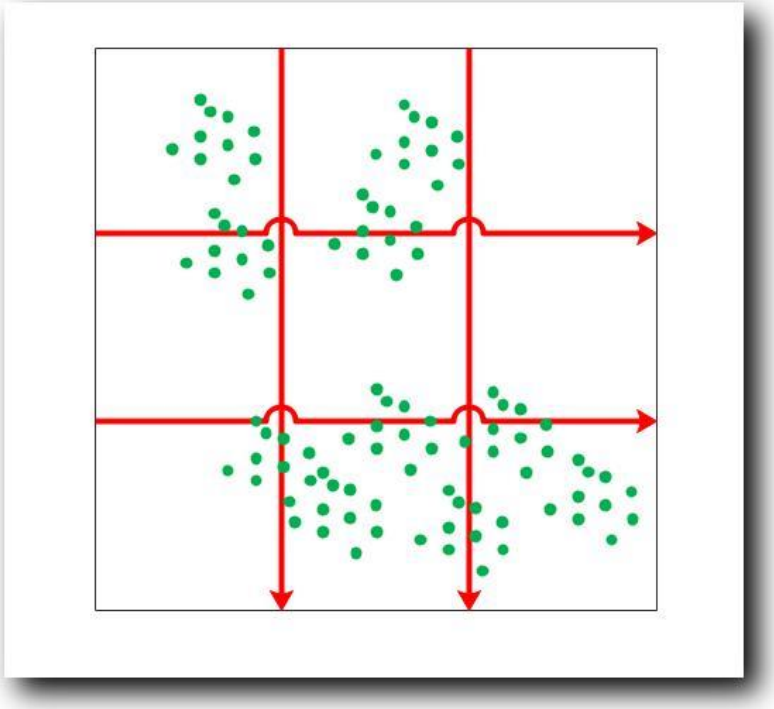
(3)基于密度的聚类算法。只要临近区域的密度（对象或数据点的数目）超过某个阈值就继续聚类。也就是说，对给定类中的每个数据点，在一个给定范围的区域中必须至少包含某个数目的点。这样的方法可以用来过滤噪声和孤立点数据，发现任意形状的类。

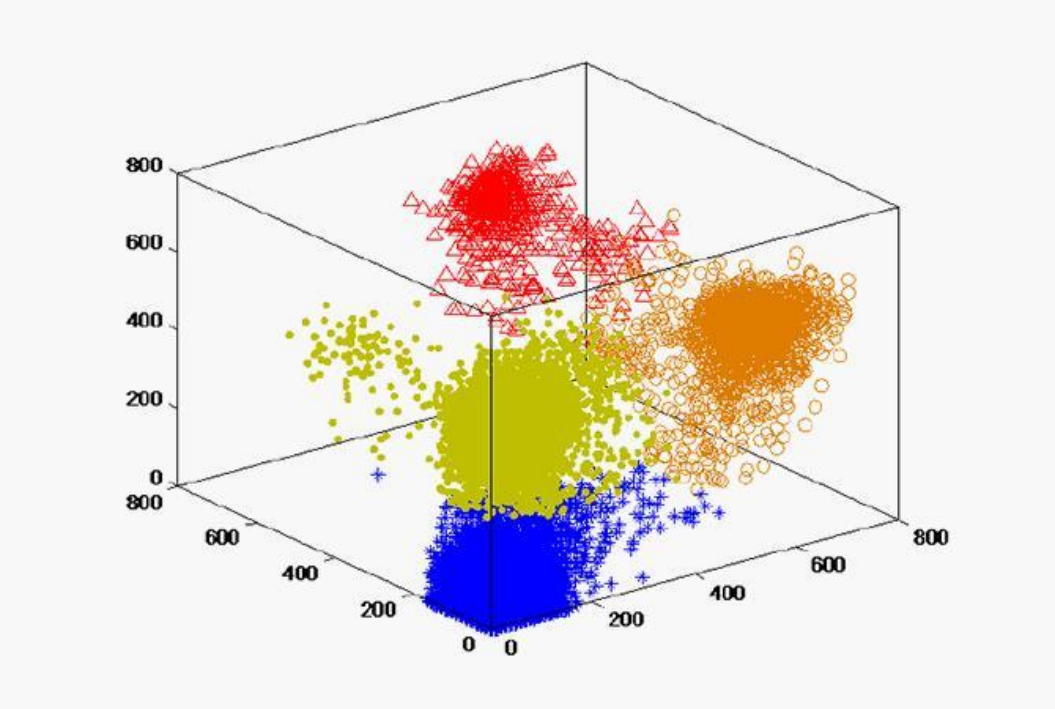




一、无监督：聚类概述

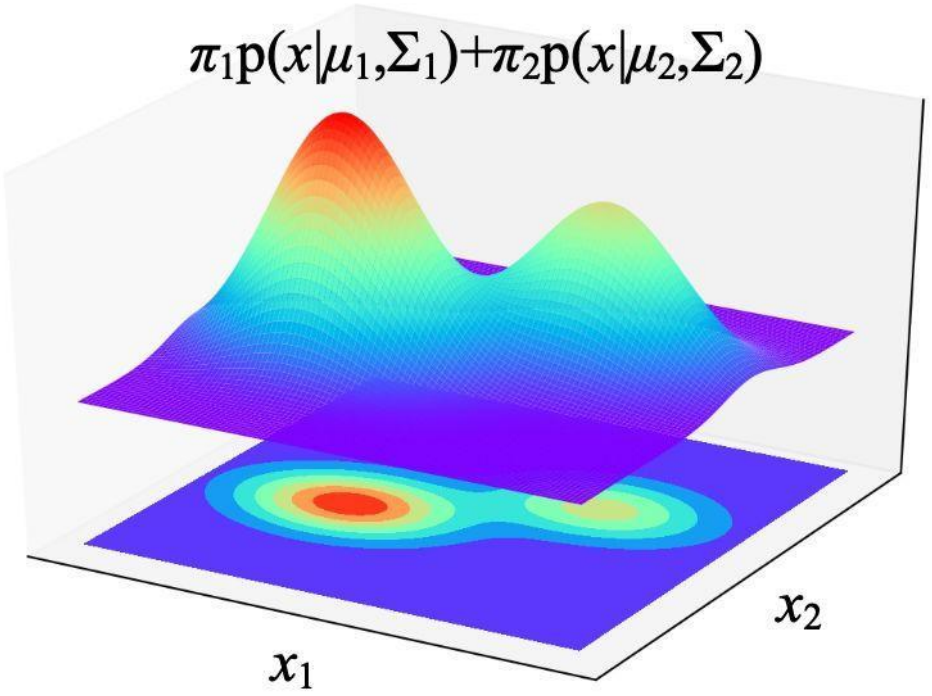
(4) 基于网格的聚类算法。首先把对象空间划分成有限个单元的网状结构，所有的处理都是以单个单元为对象的。这种方法的主要优点是处理速度快，其处理时间独立于数据对象的数目，只与划分数据空间的单元数有关。

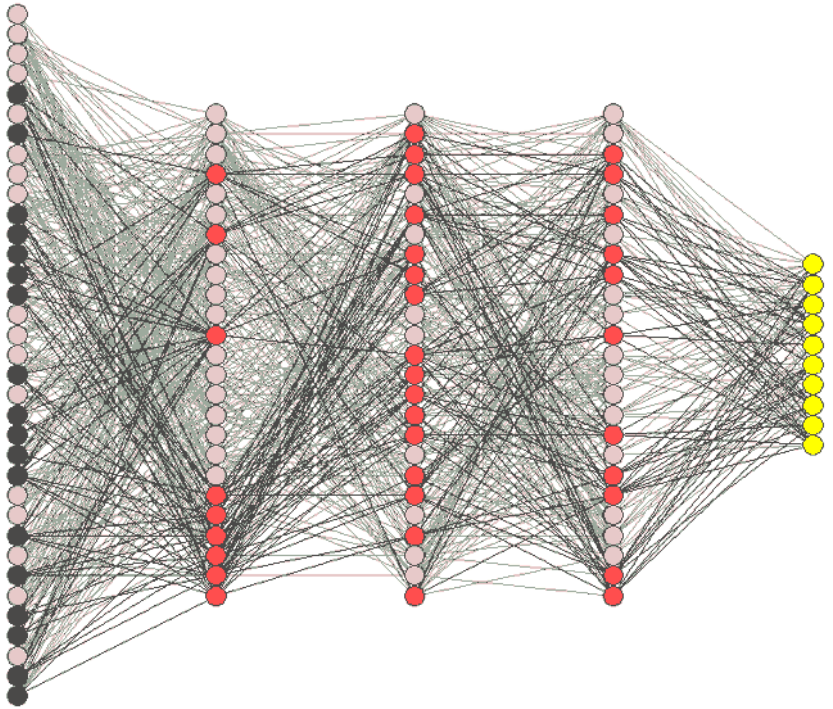




一、无监督：聚类概述

(5) 基于模型的算法。为每个簇假定一个模型，然后去寻找能够很好地满足这个模型的数据集。这样的算法经常基于这样的假定：数据集是由一系列的概率分布所决定的。





二、划分聚类方法

- 对于给定的数据集，划分方法通过选择适当的初始代表点将数据样本进行初始聚类，通过迭代过程对聚类的结果进行不断地调整，直到使评价聚类性能准则函数的值达到最优为止。
- 属于这样的聚类方法有：k均值（k-means）、k中心点（k-medoids）等。
- 划分方法的主要思想：
  - ✓ 给定一个包含n个数据对象的数据集，进行k个划分，每个划分表示一个簇，并且$k \leq n$。
  - ✓ 簇要满足下列条件：①每个簇至少包含一个对象；②每个对象属于且仅属于一个簇。
  - ✓ 对于给定的要构建的划分的数目k，划分方法首先给出一个初始的划分，然后采用一种迭代技术，通过对象在划分间移动来改进划分，使得每一次改进后的划分方案都较前一次更好。
  - ✓ 好的划分是指同一簇中的对象之间尽可能“接近”，不同簇中的对象之间尽可能“远离”。

二、划分聚类方法

- k-means聚类算法是划分聚类方法中最常用、最流行的经典算法，许多其他的方法都是k-means聚类算法的变种。
- K-means聚类算法将各个聚类子集内的所有数据样本的均值作为该聚类的代表点。
- 算法的主要思想是通过迭代过程把数据集划分为不同的类别，使得评价聚类性能的准则函数达到最优，从而使生成的每个聚类类内紧凑，类间独立。
- k-means聚类算法不适合处理离散型属性，但是对于连续型属性具有较好的聚类效果。

二、划分聚类方法

k均值算法的算法思想：

- 从包含n个数据对象的数据集中随机的选择k个对象，每个对象初始的代表一个簇的平均值或质心或中心；
- 对剩余的每个数据对象点根据其与各个簇中心的距离，将它指派到最近的簇；
- 根据指派到簇的数据对象点，更新每个簇的质心；
- 重复指派和更新步骤，直到簇不发生变化。k均值算法的目标函数E定义为：

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} [d(x, \bar{x}_i)]^2$$

二、划分聚类方法

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} [d(x, \bar{x}_i)]^2$$

- 其中x是空间中的点，表示给定的数据对象，$\bar{x}_i$是簇C_i的数据对象的平均值。
- 例如3个二维点(1, 3)，(2, 1)和(6, 2)的质心是((1+2+6)/3, (3+1+2)/3)=(3, 2)。
- $d(x, \bar{x}_i)$表示x与$\bar{x}_i$之间的距离。
- 这个目标函数可以保证生成的簇尽可能的紧凑和独立。

二、划分聚类方法

k均值算法

输入：所期望的簇的数目k，包含n个对象的数据集D

输出：k个簇的集合

1：从D中任意选择k个对象作为初始簇中心；

2：repeat

3：将每个点指派到最近的中心，形成k个簇；

4：重新计算每个簇的中心；

5：计算目标函数E；

6：until目标函数E不再发生变化或中心不再发生变化；

- 算法分析：k均值算法的步骤3和步骤4试图直接最小化目标函数E；
- 步骤3通过将每个点指派到最近的中心形成簇，最小化关于给定中心的目标函数E；
- 而步骤4重新计算每个簇的中心，进一步最小化E。

二、划分聚类方法

k均值算法举例：

- 假设要进行聚类的元组为$\{2,4,10,12,3,20,30,11,25\}$，假设要求的簇的数量为$k=2$。
  - 第一步：初始时用前两个数值作为簇的质心，这两个簇的质心记做：$m1=2$，$m2=4$；
  - 第二步：对剩余的每个对象，根据其与各个簇中心的距离，将它指派给最近的簇中，可得：$C1=\{2,3\}$，$C2=\{4,10,12,20,30,11,25\}$；
  - 第三步：计算簇的新质心：$m1=(2+3)/2=2.5$，$m2=(4+10+12+20+30+11+25)/7=16$;
  - 重新对簇中的成员进行分配可得$C1=\{2,3,4\}$和$C2=\{10,12,20,30,11,25\}$，不断重复这个过程，均值不再变化时最终可得到两个簇：$C1=\{2,3,4,10,11,12\}$和$C2=\{20,30,25\}$。

二、划分聚类方法

k均值算法举例：西瓜数据集聚类

请使用K均值聚类算法对所列西瓜数据集进行聚类，假设要求的簇的数量为$k=3$。

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0. 697	0. 46	11	0. 245	0. 057	21	0. 748	0. 232
2	0. 774	0. 376	12	0. 343	0. 099	22	0. 714	0. 346
3	0. 634	0. 264	13	0. 639	0. 161	23	0. 483	0. 312
4	0. 608	0. 318	14	0. 657	0. 198	24	0. 478	0. 437
5	0. 556	0. 215	15	0. 36	0. 37	25	0. 525	0. 369
6	0. 403	0. 237	16	0. 593	0. 042	26	0. 751	0. 489
7	0. 481	0. 149	17	0. 719	0. 103	27	0. 532	0. 472
8	0. 437	0. 211	18	0. 359	0. 188	28	0. 473	0. 376
9	0. 666	0. 091	19	0. 339	0. 241	29	0. 725	0. 445
10	0. 243	0. 267	20	0. 282	0. 257	30	0. 446	0. 459

二、划分聚类方法

k均值算法举例：西瓜数据集聚类

- 第一步：初始时随机选取三个样本 x_6, x_{12}, x_{27} 作为簇的质心
- 第二步：对剩余的每个对象，根据其与各个簇中心的距离，将它指派给最近的簇中，可得当前簇划分：

$$C_1 = \{x_5, x_6, x_7, x_8, x_9, x_{10}, x_{13}, x_{14}, x_{15}, x_{17}, x_{18}, x_{19}, x_{20}, x_{23}\}$$

$$C_2 = \{x_{11}, x_{12}, x_{16}\}$$

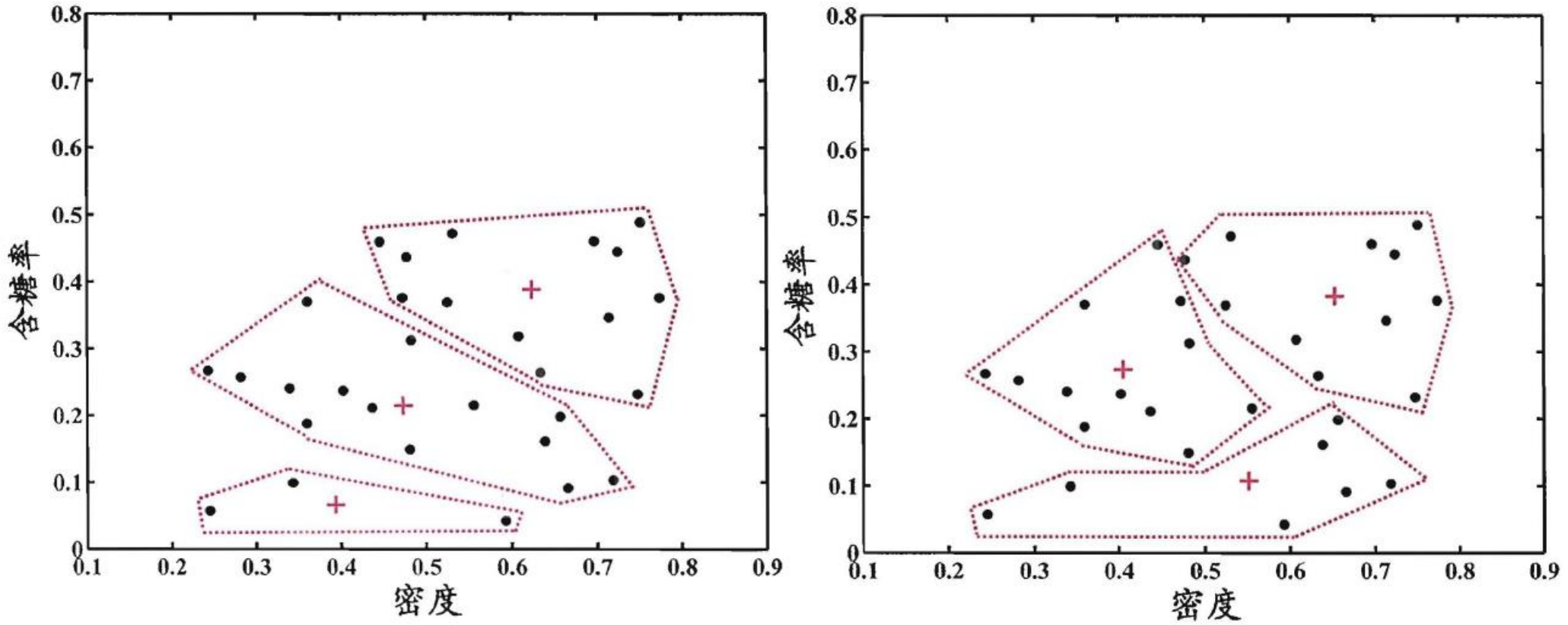
$$C_3 = \{x_1, x_2, x_3, x_4, x_{21}, x_{22}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}\}$$

- 第三步：根据C_1, C_2, C_3求出新的簇中心的均值向量：
- $\mu_1' = (0.473; 0.214)$ $\mu_2' = (0.394; 0.066)$ $\mu_3' = (0.623; 0.388)$
- 更新当前均值向量后，不断重复上述过程，如下图所示，第五轮迭代产生的结果与第四轮迭代相同，于是算法停止，得到最终的簇划分.

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

二、划分聚类方法

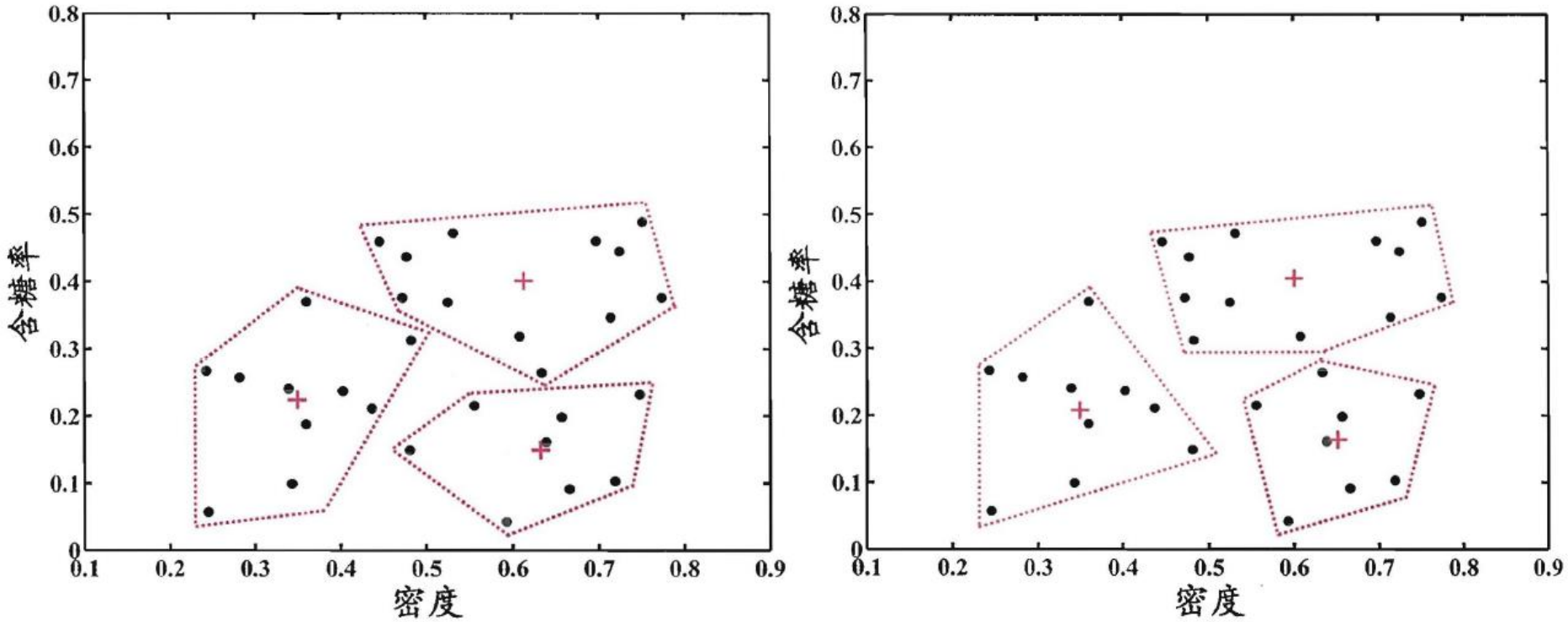
k均值算法举例：西瓜数据集聚类



(a) 第一轮迭代后　　(b) 第二轮迭代后

二、划分聚类方法

k均值算法举例：西瓜数据集聚类



(c) 第三轮迭代后　　(d) 第四轮迭代后

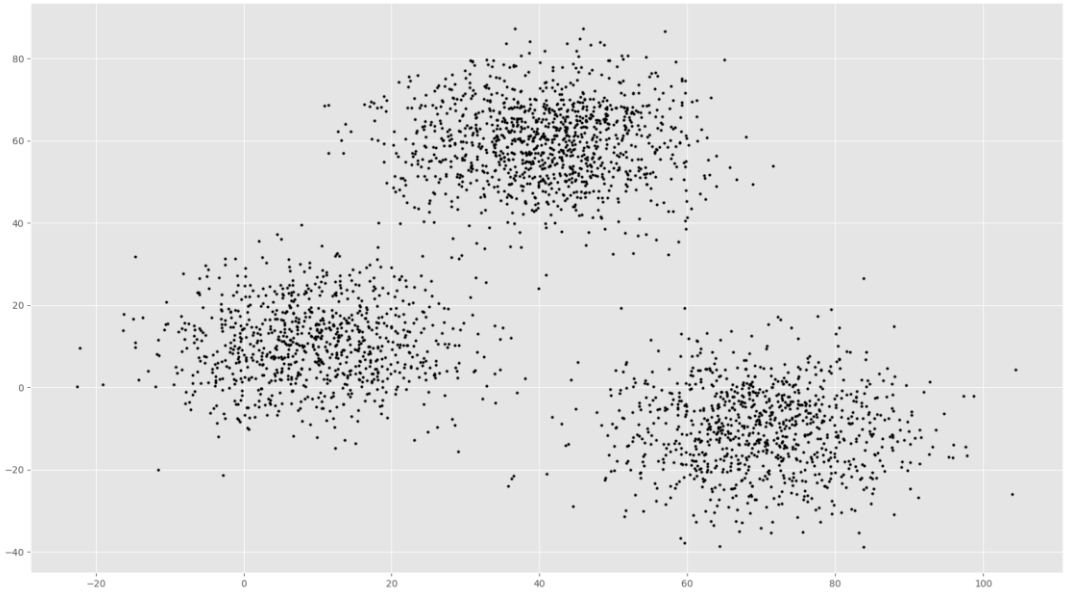
第五轮迭代产生的结果与第四轮迭代相同，于是算法停止，得到最终的簇划分.

二、划分聚类方法

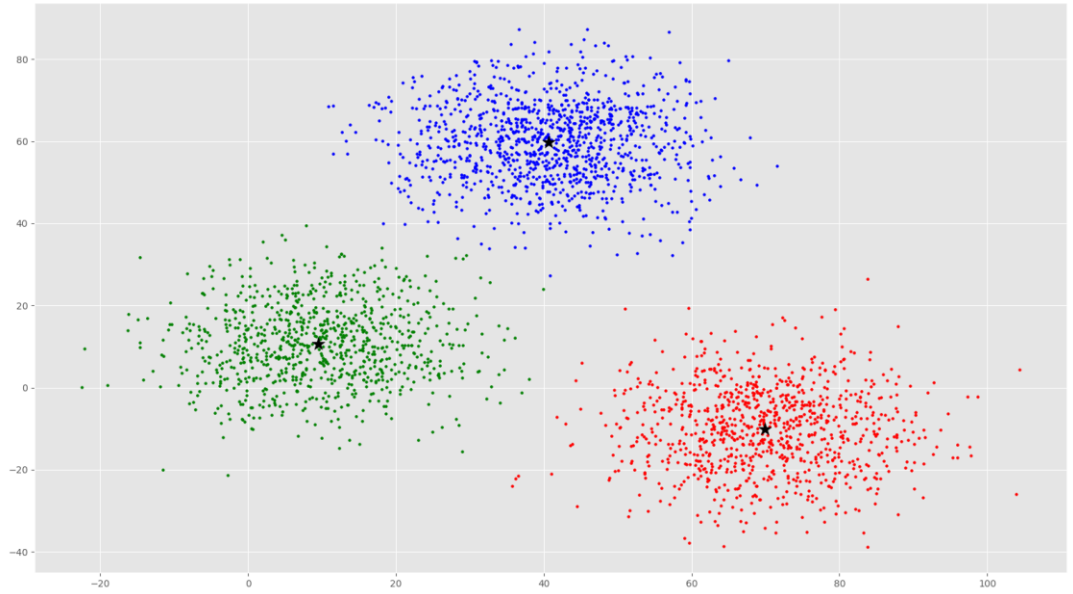
K-means算法聚类效果展示：

- 不同颜色代表不同的簇，黑色五角星代表不同簇的聚类中心

原始数据：



聚类之后：



二、划分聚类方法

- k均值算法对离群数据对象点是敏感的，一个极大值的对象可能在相当大的程度上扭曲数据的分布。目标函数E的使用更是进一步恶化了这一影响。
- k-中心点算法可以消除k-平均算法对孤立点的敏感性。
- k中心点算法：
  - ✓ 在每个簇中选出一个最靠近均值的实际的对象来代表该簇，其余的每个对象指派到与其距离最近的代表对象所在的簇中。
  - ✓ 每次迭代后的簇的代表对象点都是从簇的样本点中选取，选取的标准就是当该样本点成为新的代表对象点后能提高簇的聚类质量，使得簇更紧凑。

二、划分聚类方法

- k中心点算法使用绝对误差标准作为度量聚类质量的目标函数，其定义如下：

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, o_i)$$

- 其中，E是数据集中所有数据对象的绝对误差之和，x是空间中的点，代表簇C_i中一个给定的数据对象，o_i是簇C_i中的代表对象。
- 如果某样本点成为代表对象点后，绝对误差能小于原代表对象点所造成的绝对误差，那么k中心算法认为该样本点是可以取代原代表对象点的。
- 通常，该算法重复迭代，直到每个代表对象都成为它的簇的实际中心点，或最靠中心的对象。

二、划分聚类方法

- PAM(围绕中心点的划分)是最早提出的k-中心算法之一，它尝试将n个对象划分出k类。
- PAM算法的主要思想：首先为每个簇任意选择一个代表对象(即中心点)，计算其余的数据对象与代表对象之间的距离，将其加入到最近的簇，接着反复尝试用更好的非代表对象点来替代代表数据对象点，以改进聚类的质量 。
- 在PAM算法中，可以把过程分为两个步骤：
  - ✓ （1）建立：随机选择k个对象点作为初始的簇中心点；
  - ✓ （2）交换：对所有可能的对象对进行分析，找到交换后可以使误差减少的对象，代替原中心点。

二、划分聚类方法

算法PAM(k-中心点算法)

输入：簇的数目k，包含n个对象的数据集D

输出：k个簇，使得所有对象与其最近代表对象点的距离总和最小

1：任意选择k个对象作为初始的簇中心点；

2：将每个剩余对象指派给离它最近的中心点所代表的簇；

3：任意选择一个非中心对象o；

4：计算用o代替中心对象的总代价S；

5：如果S为负，则可以用o代替 以构成新聚类的k个中心对象；

6：重复(2)(3)(4)(5)，直到每个簇不再发生变化为止。

- 算法分析：k-中心点算法消除了k-平均算法对孤立点的敏感性；比k-平均算法更健壮。

二、划分聚类方法

- 例：给定含有5个数据对象的数据集D，D中的对象为A,B,C,D,E，各对象之间的距离如下表所示，根据所给的数据对其运行k-中心点算法实现划分聚类(设$k=2$)。

对象之间的距离

样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

算法按下面步骤执行：

- 步骤1：假如从5个对象中随机选取A、C作为初始聚类中心。
- 步骤2：计算其它对象与中心对象之间的距离，将每个剩余对象指派给离它最近的中心点所代表的簇，通过查询表5可知：可得到2个划分为：$\{A, B, E\}$和$\{C, D\}$。
- 步骤3：任选非中心对象B、D、E分别与中心对象A、C交换，计算样本点的代价。

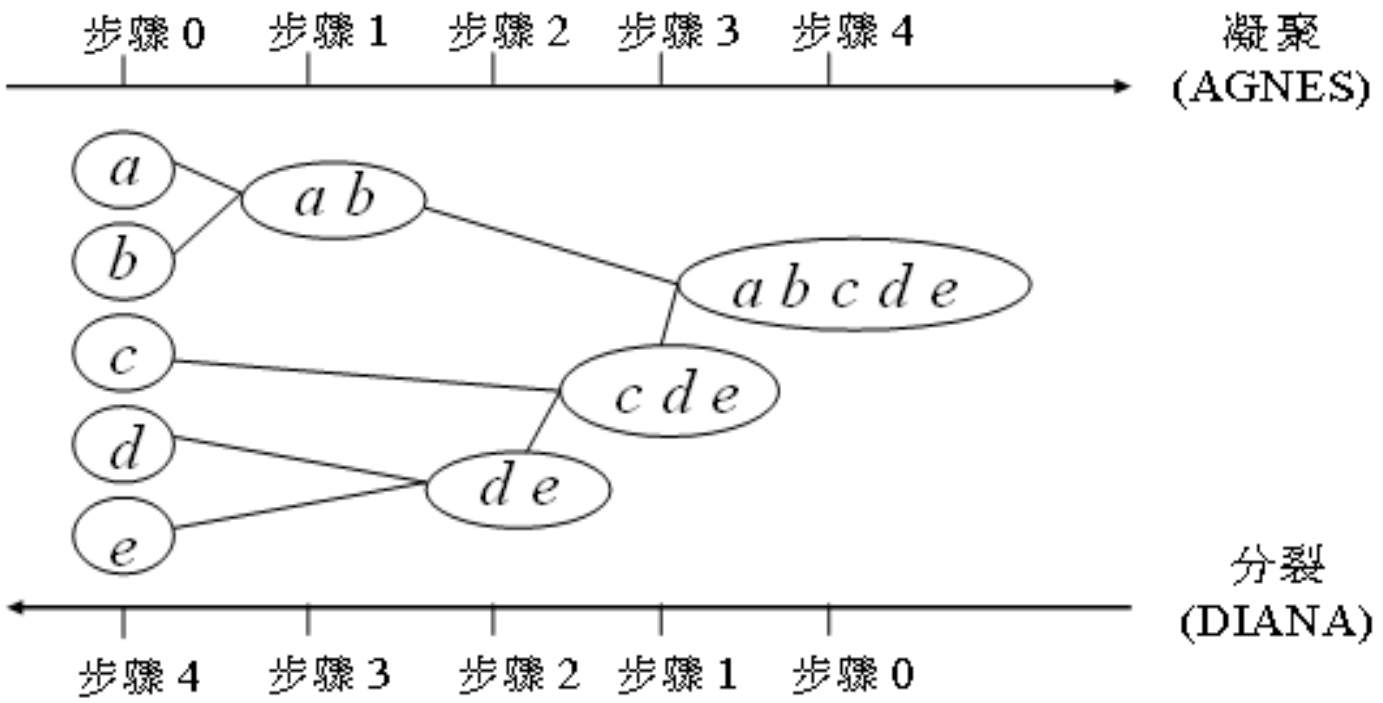
三、层次聚类方法

层次聚类方法有两种：

- **自底向上**的凝聚层次聚类方法。
  - ✓首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到所有的对象都在一个簇中，或者达到了某个终止条件。绝大多数的层次聚类方法都属于这一类，只是在簇间相似度的定义上有所不同。凝聚层次聚类算法的代表是AGNES算法。
- **自顶向下**的分裂层次聚类方法。
  - ✓它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到每个对象自成一簇，或者达到了某个终止条件，例如达到了某个希望的簇数目，或者两个最近的簇之间的距离超过了某个阈值。分裂层次聚类算法的代表是DIANA算法。

三、层次聚类方法

- 下图描述了一种凝聚层次聚类算法AGNES和一种分裂层次聚类算法DIANA对一个包含五个对象的数据集合$\{a, b, c, d, e\}$的处理过程。



三、层次聚类方法

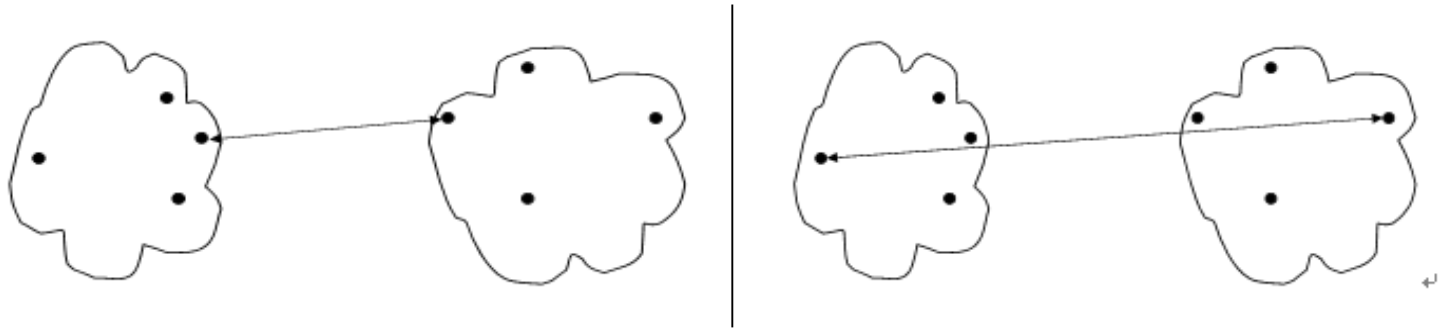
- AGNES将每个对象自为一簇，然后这些簇根据某种准则逐步合并，直到所有的对象最终合并形成一个簇。
  - ✓例如，如果簇C_1中的一个对象和簇C_2中的一个对象之间的距离是所有属于不同簇的对象间欧氏距离中最小的，则C_1和C_2合并。
- 在DIANA中，所有的对象用于形成一个初始簇。根据某种原则（如簇中最近的相邻对象的最大欧氏距离），将该簇分裂。簇的分裂过程反复进行，直到最终每个新簇只包含一个对象。
- 在凝聚或者分裂层次聚类方法中，用户可以定义希望得到的簇数目作为一个终止条件。

三、层次聚类方法

- 四个广泛采用的簇间距离度量方法如下：

（1）簇间最小距离：是指用两个簇中所有数据点的最近距离代表两个簇的距离，下图左。

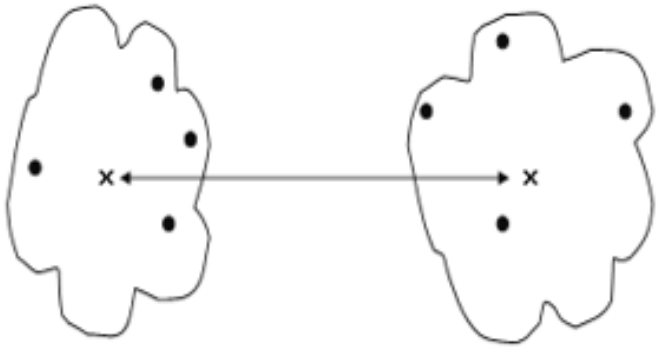
（2）簇间最大距离：是指用两个簇所有数据点的最远距离代表两个簇的距离，下图右。



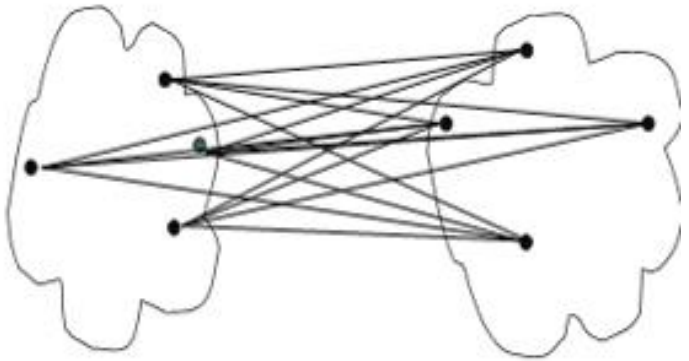
三、层次聚类方法

- 四个广泛采用的簇间距离度量方法如下：

（3）簇间均值距离：是指用两个簇各自中心点之间的距离代表两个簇的距离，如下图。



（4）簇间平均距离：是指用两个簇所有数据点间的距离的平均值代表两个簇的距离，如下图。



三、层次聚类方法

算法 AGNES(自底向上凝聚层次聚类)

输入：包含n个对象的数据集D，终止条件簇的数目k

输出：k个簇

1：将每个对象当成一个初始簇；

2：repeat

3：根据两个簇中最近的数据点找到最近的两个簇；

4：合并两个簇，生成新的簇的集合；

5：until达到定义的簇的数目

- 算法分析：

（1）简单，但遇到合并点选择困难的情况；

（2）一旦一组对象被合并，不能撤销；

（3）算法的复杂度为$O(n^2)$，不适合大数据集。

三、层次聚类方法

算法 DIANA (自顶向下的分裂层次聚类)

输入：包含n个对象的数据集D，终止条件簇的数目k

输出：k个簇

1：将包含n个对象的数据集D当成一个初始簇；

2：repeat

3：在同类簇中找到距离最远的样本点对；

4：以该样本点为代表，将原类簇中的样本点重新分属到这两个新簇中；

5：until达到定义的簇的数目

- 算法分析：

（1）缺点是已做的分裂操作不能撤销，类之间不能交换对象；

（2）如果在某步没有选择好分裂点，可能会导致低质量的聚类结果；

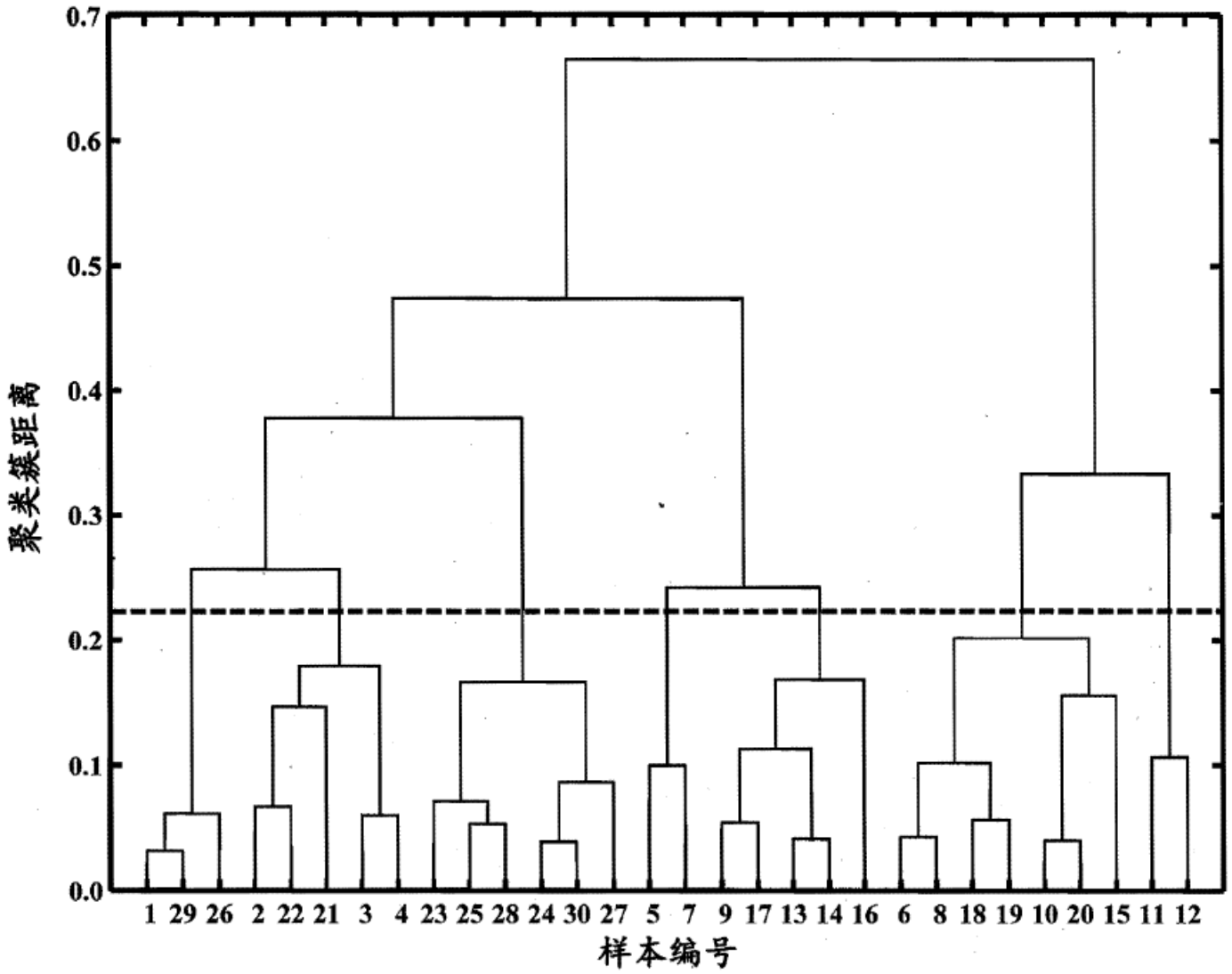
（3）算法的复杂度为$O(tn^2)$，t为迭代次数，不适合大数据集。

三、层次聚类方法

请使用AGNES算法对所列西瓜数据集进行聚类，假设要求的簇的数量为$k = 4$。

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0. 697	0. 46	11	0. 245	0. 057	21	0. 748	0. 232
2	0. 774	0. 376	12	0. 343	0. 099	22	0. 714	0. 346
3	0. 634	0. 264	13	0. 639	0. 161	23	0. 483	0. 312
4	0. 608	0. 318	14	0. 657	0. 198	24	0. 478	0. 437
5	0. 556	0. 215	15	0. 36	0. 37	25	0. 525	0. 369
6	0. 403	0. 237	16	0. 593	0. 042	26	0. 751	0. 489
7	0. 481	0. 149	17	0. 719	0. 103	27	0. 532	0. 472
8	0. 437	0. 211	18	0. 359	0. 188	28	0. 473	0. 376
9	0. 666	0. 091	19	0. 339	0. 241	29	0. 725	0. 445
10	0. 243	0. 267	20	0. 282	0. 257	30	0. 446	0. 459

三、层次聚类方法



令AGNES 算法一直执行到所有样本出现在同一个簇中，即k= 1， 则可得到左图所示的“树状图”，其中每层链接一组聚类簇.

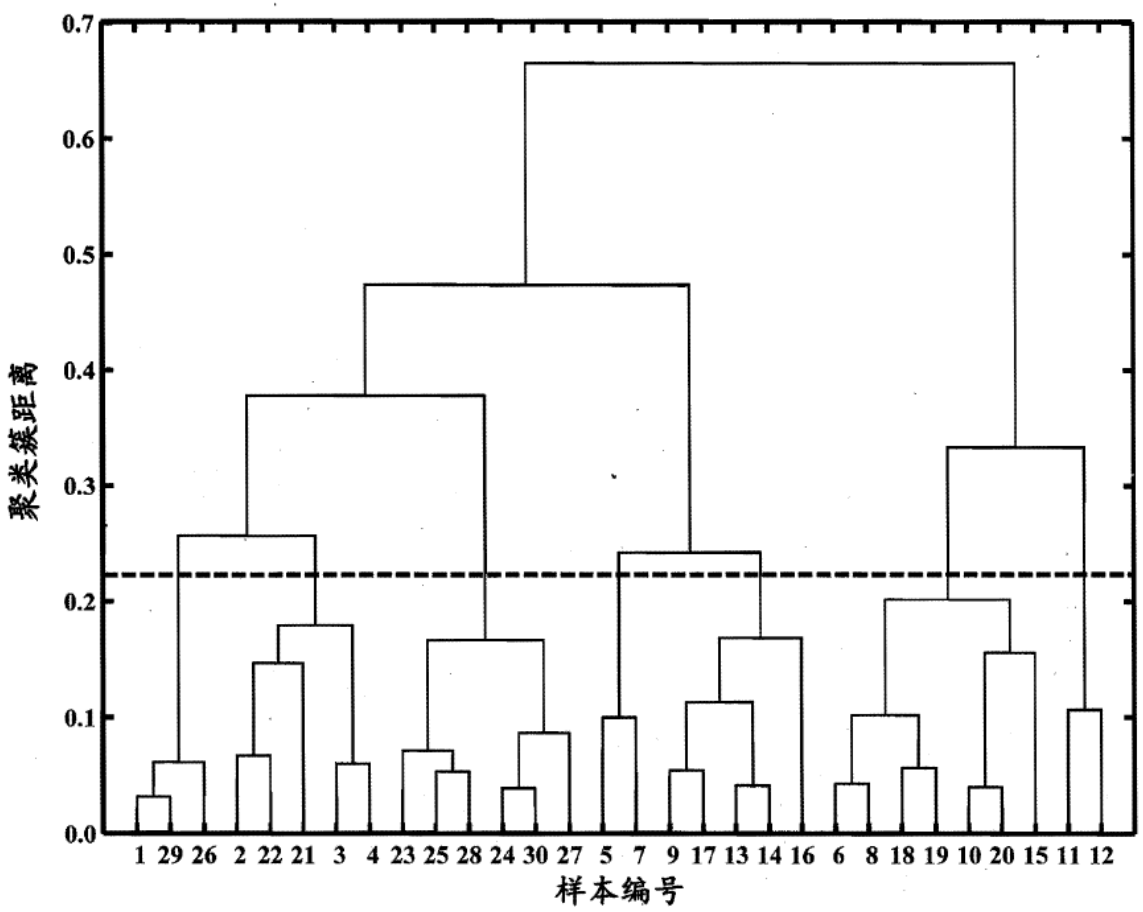
横轴对应于样本编号

纵轴对应于聚类簇距离.

在树状图的特定层次上进行分割，则可得到相应的簇划分结果.

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

三、层次聚类方法



例如，以图中所示虚线分割树状图，将得到包含7 个聚类簇的结果:

- $C_1 = \{x_1, x_{26}, x_{29}\}$
- $C_2 = \{x_2, x_3, x_4, x_{21}, x_{22}\}$
- $C_3 = \{x_{23}, x_{24}, x_{25}, x_{27}, x_{28}, x_{30}\}$
- $C_4 = \{x_5, x_7\}$
- $C_5 = \{x_9, x_{13}, x_{14}, x_{16}, x_{17}\}$
- $C_6 = \{x_6, x_8, x_{10}, x_{15}, x_{18}, x_{19}, x_{20}\}$
- $C_7 = \{x_{11}, x_{12}\}$

四、密度聚类方法

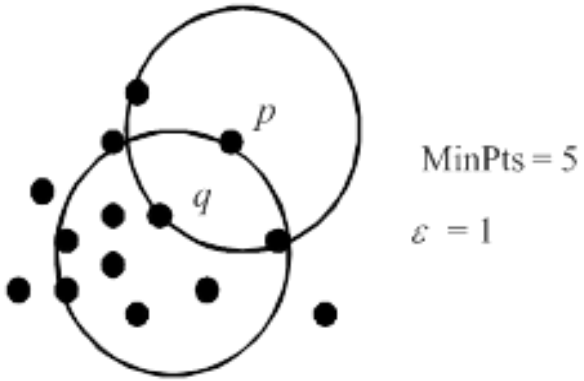
密度聚类方法

- 基于密度的聚类方法以数据集在空间分布上的稠密程度为依据进行聚类，无需预先设定簇的数量，特别适合对于未知内容的数据集进行聚类。
- 密度聚类方法的基本思想是：只要一个区域中的点的密度大于某个域值，就把它加到与之相近的聚类中去，对于簇中每个对象，在给定的半径ε的邻域中至少要包含最小数目（$MinPts$）个对象。
- 这类算法能克服基于距离的算法只能发现“类圆形”的聚类的缺点，可发现任意形状的聚类，且对噪声数据不敏感。
- 代表算法有：DBSCAN、OPTICS、DENCLUE算法等。

四、密度聚类方法

DBSCAN算法

- DBSCAN是一种基于高密度连通区域的基于密度的聚类方法，该算法将具有足够高密度的区域划分为簇，并在具有噪声的空间数据集中发现任意形状的簇，它将簇定义为密度相连的点的最大集合。
- 对象的ε邻域：给定对象在半径ε内的区域。
- 核心对象：如果一个对象的ε邻域至少包含最小数目$MinPts$（$>= MinPts$）个对象，则称该对象为核心对象。如下图中，$\varepsilon = 1$，$MinPts = 5$，q是一个核心对象。



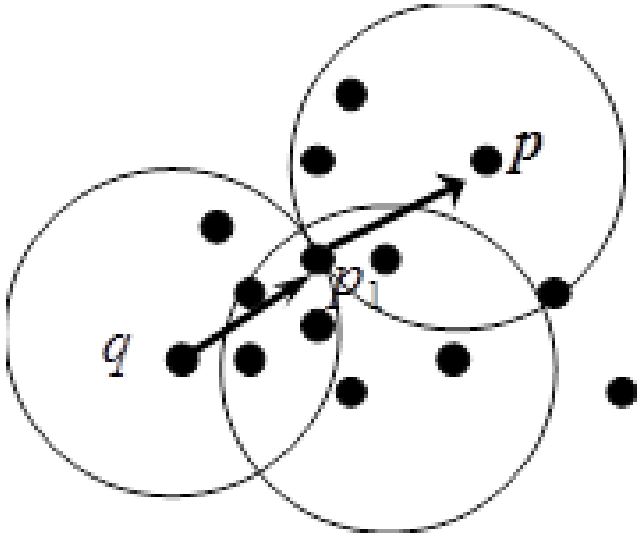
核心点、边界点和噪声点

四、密度聚类方法

- 边界点：不是核心点，但落在某个核心点的ε邻域内。
- 噪声：不包含在任何簇中的对象被认为是“噪声”。
- 直接密度可达：给定一个对象集合D，如果p是在q的ε邻域内，而q是一个核心对象，我们说对象p从对象q出发是直接密度可达的。如果q是一个核心对象，p属于q的邻域，那么称p直接密度可达q。

四、密度聚类方法

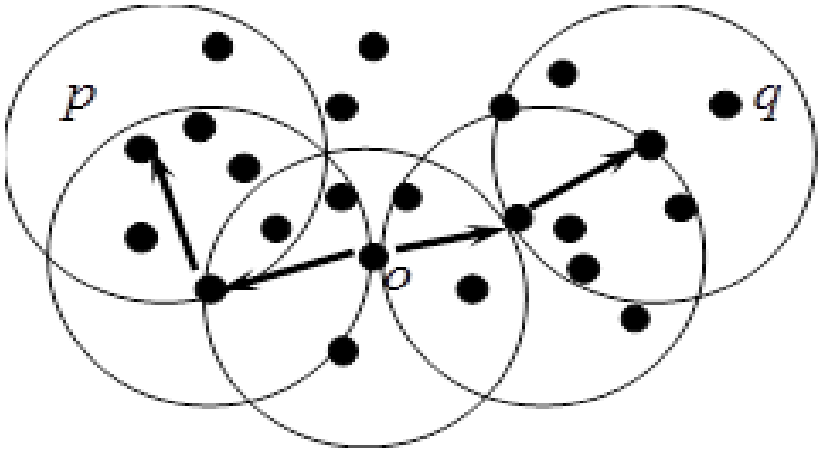
- 密度可达：如果存在一个对象链$p_1, p_2, \ldots, p_n$，$p_1 = q, p_n = p$，对$p_i \in D, (1 \le i \le n), p_{i+1}$是从$p_i$关于$\varepsilon$和$MitPts$直接密度可达的，则对象$p$是从对象$q$关于$\varepsilon$和$MinPts$密度可达的，如同下图所示。
- 由一个核心对象和其密度可达的所有对象构成一个聚类。



密度可达的

四、密度聚类方法

- 密度相连的：如果对象集合D中存在一个对象o，使得对象p和q是从o关于ε和$MinPts$密度可达的，那么对象p和q是关于ε和$MinPts$密度相连的。



密度相连的

四、密度聚类方法

请使用DBSCAN算法对所列西瓜数据集进行聚类，假定 ε=1，$MinPts$=5

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0. 697	0. 46	11	0. 245	0. 057	21	0. 748	0. 232
2	0. 774	0. 376	12	0. 343	0. 099	22	0. 714	0. 346
3	0. 634	0. 264	13	0. 639	0. 161	23	0. 483	0. 312
4	0. 608	0. 318	14	0. 657	0. 198	24	0. 478	0. 437
5	0. 556	0. 215	15	0. 36	0. 37	25	0. 525	0. 369
6	0. 403	0. 237	16	0. 593	0. 042	26	0. 751	0. 489
7	0. 481	0. 149	17	0. 719	0. 103	27	0. 532	0. 472
8	0. 437	0. 211	18	0. 359	0. 188	28	0. 473	0. 376
9	0. 666	0. 091	19	0. 339	0. 241	29	0. 725	0. 445
10	0. 243	0. 267	20	0. 282	0. 257	30	0. 446	0. 459

四、密度聚类方法

DBSCAN 算法先找出各样本的ε-邻域并确定核心对象集合:

$$\Omega = \{x_3, x_5, x_6, x_8, x_9, x_{13}, x_{14}, x_{18}, x_{19}, x_{24}, x_{25}, x_{28}, x_{29}\}$$

然后，从Ω中随机选取一个核心对象作为种子，找出由它密度可达的所有样本，这就构成了第一个聚类簇. 不失一般性，假定核心对象x_8被选中作为种子，则DBSCAN生成的第一个聚类簇为：

$$C_1 = \{x_6, x_7, x_8, x_{10}, x_{12}, x_{18}, x_{19}, x_{20}, x_{23}\}$$

接着， DBSCAN 将C_1中包含的核心对象从Ω 中去除:

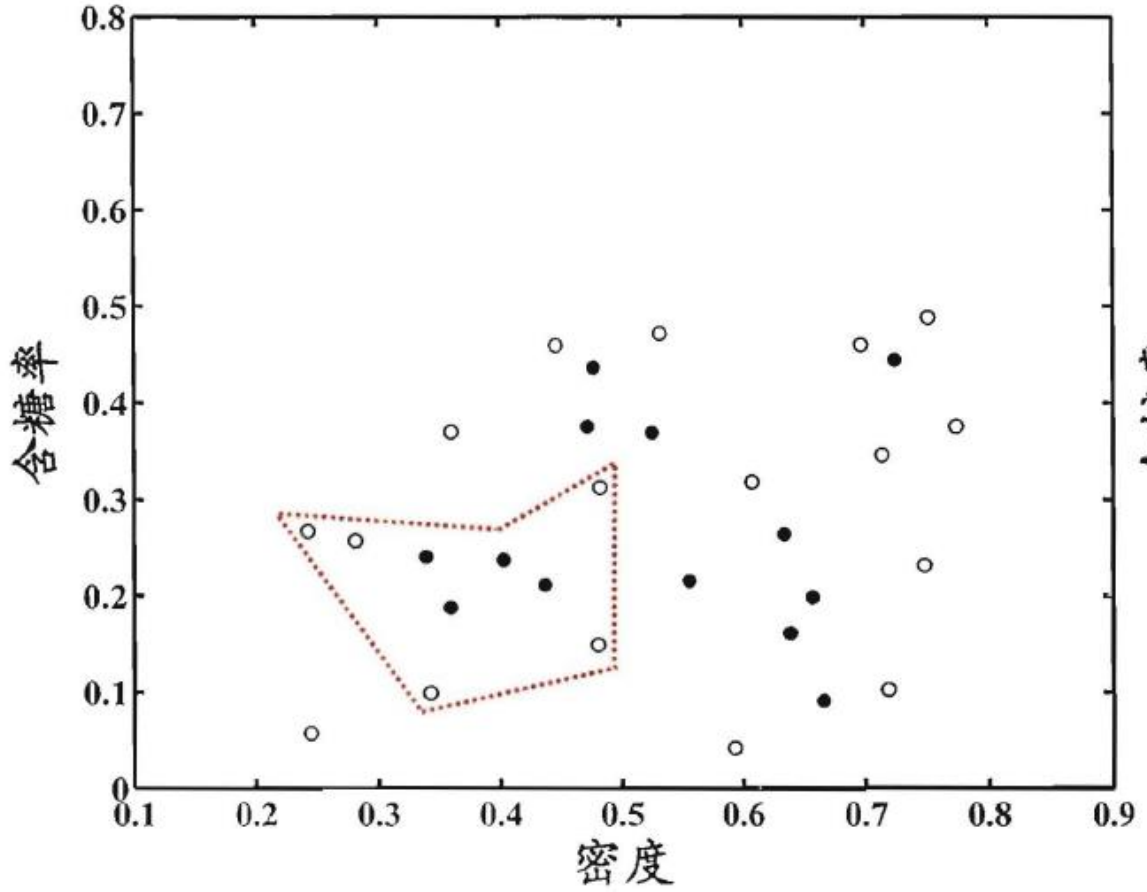
$$\Omega = \Omega \setminus C_1 = \{x_3, x_5, x_9, x_{13}, x_{14}, x_{24}, x_{25}, x_{28}, x_{29}\}$$

再从更新后的集合中随机选取一个接心对象作为种子来生成下一个聚类簇。

上述过程不断重复，直至 Ω 为空。

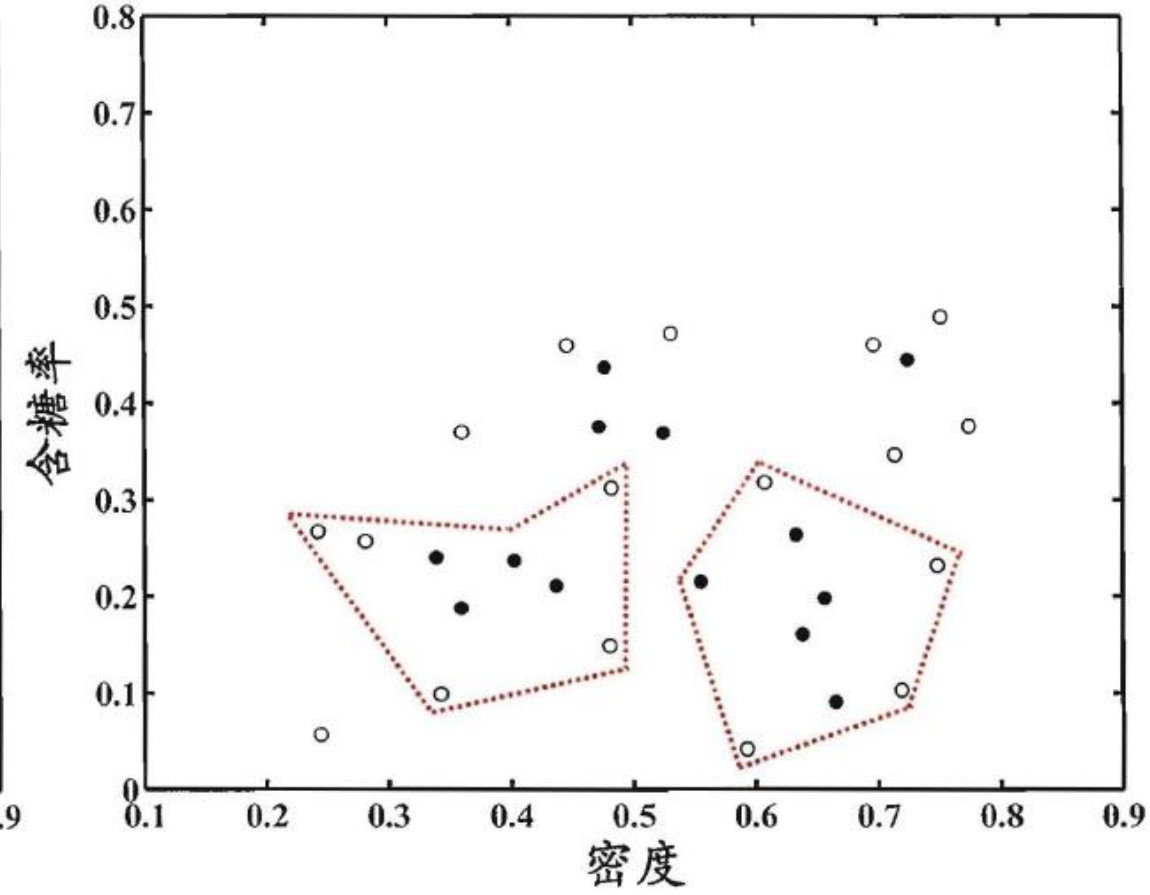
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

四、密度聚类方法



生成聚类簇C_1

$C_1 = \{x_6, x_7, x_8, x_{10}, x_{12}, x_{18}, x_{19}, x_{20}, x_{23}\}$

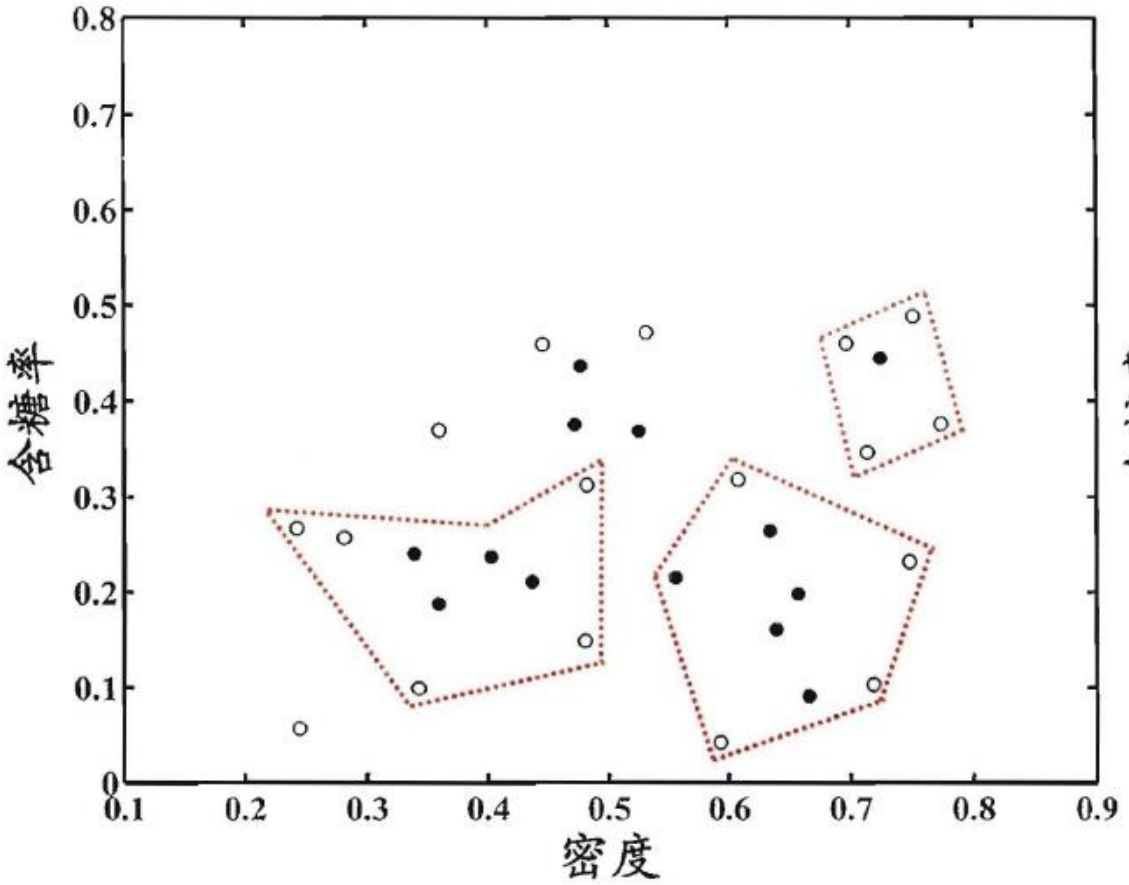


生成聚类簇C_2

$C_2 = \{x_3, x_4, x_5, x_9, x_{13}, x_{14}, x_{16}, x_{17}, x_{21}\}$

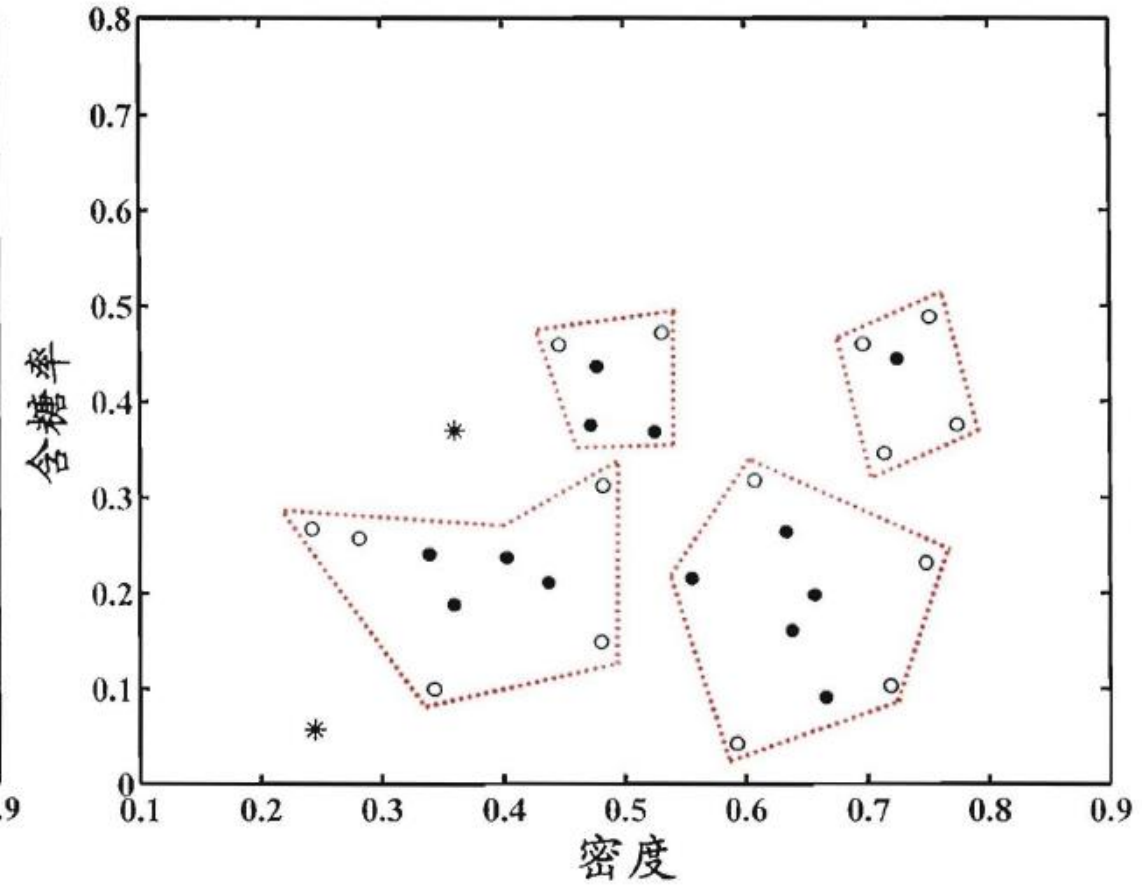
北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

四、密度聚类方法



生成聚类簇C_3

$C_3 = \{x_1, x_2, x_{22}, x_{26}, x_{29}\}$



生成聚类簇C_4

$C_4 = \{x_{24}, x_{25}, x_{27}, x_{28}, x_{30}\}$

五、GMM聚类方法

高斯混合模型（GMM）聚类

- 高斯混合模型（GMM）聚类的核心思想是假设所有的数据点来自多个参数不同的高斯分布，来自同一分布的数据点被划分为同一类。算法结果返回的是数据点属于不同类别的概率。
- 高斯混合模型（GMM）聚类可以看做是k-means模型的一个优化。它既是一种工业界常用的技术手段，也是一种生成式模型。高斯混合模型试图找到多维高斯模型概率分布的混合表示，从而拟合出任意形状的数据分布。

五、GMM聚类方法

- GMM混合高斯分布是多个高斯分布函数的线性组合，假设有随机变量$\boldsymbol{x}$，GMM模型可以用如下公式表示：

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- 其中，$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$是模型中的第$k$个分量(component)；
- π_k是混合系数（mixture coefficient)；
- 而且有 $\sum_{k=1}^{K} \pi_k = 1$，$0 \leq \pi_k \leq 1$

五、GMM聚类方法

- 如果样本x是一维数据(Univariate)，高斯分布的概率密度函数PDF(Probability Density Function)为：

$$P(x \mid \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- 当x为多维数据(Multivariate)时，其PDF如下：

$$P(x \mid \theta) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right)$$

- 其中，μ为期望，Σ为协方差矩阵，是一个对称矩阵，D为数据的维度。

五、GMM聚类方法

- GMM聚类的核心思想是假设所有的数据点来自多个参数不同的高斯分布，来自同一分布的数据点被划分为同一类。算法结果返回的是数据点属于不同类别的概率。
- 提到GMM，必然绕不过最大期望算法EM(Expectation Maximum)算法。下面我们用简单的方式来描述一下EM算法。
- 假设我们得到一组样本x_t，而且x_t服从高斯分布，但是高斯分布的参数未知，即$x \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$。我们的目标是找一个合适的高斯分布，确定分布参数，使得这个高斯分布能产生这组样本的可能性尽可能大。
- 要使产生这组样本的可能性尽可能大，这个时候就需要用到极大似然估计(Maximum Likehood Estimate, MLE) 了。

五、GMM聚类方法

- 假设样本集$X = x_1, x_2, \cdots, x_n$，而$p(x_n \mid \boldsymbol{\mu}, \Sigma)$是高斯分布的概率分布函数。

  如果假设样本抽样独立，样本集X的联合概率就是似然函数：

$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = L(x_1, x_2, \cdots, x_n; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=0}^{n} p(x_n; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- 接下来的求解过程就是求极值，对上式求导，并令导数为0，就可以求得高斯分布的参数。
- 所以最大化似然函数的意义在于：通过使得样本集的联合概率最大来对参数进行估计，从而选择最佳的分布模型。

五、GMM聚类方法

- 回到GMM。上面是只有一个高斯分布的情况，在GMM中，有多个高斯分布，我们并不知道样本来自哪个高斯分布。换句话说，

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- π_k的值我们并不知道，我们不光需要求解不同高斯分布的参数，还需要求解不同高斯分布的概率。
- 我们引入一个隐变量γ，它是一个K维二值随机变量，在它的K维取值中只有某个特定的元素γ_k的取值为1，其它元素的取值为0。
- 实际上，隐变量描述的就是：每一次采样，选择第k个高斯模型的概率，故有：

$$p(\gamma_k = 1) = \pi_k$$

五、GMM聚类方法

- 当给定了γ的一个特定的值之后（也就是知道了这个样本从哪一个高斯模型进行采样），可以得到样本x的条件分布是一个高斯分布，满足：

$$p(x \mid \gamma_k = 1) = N(x \mid \mu_k, \Sigma_k)$$

- 而实际上，每个样本到底是从这K个高斯模型中哪个模型进行采样的，是都有可能的。故样本y的概率为：

$$p(x) = \sum_{\gamma} p(\gamma)p(y \mid \gamma) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k)$$

- 样本集X（n个样本点）的联合概率为：

$$L(\mu, \Sigma, \pi) = L(x_1, x_2 \dots x_N; \mu, \Sigma, \pi) = \prod_{n=1}^{N} p(x_n; \mu, \Sigma, \pi) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k N(x_n \mid \mu_k, \Sigma_k)$$

五、GMM聚类方法

- 对数似然函数表示为：

$$\ln L(\mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k N(x_n \mid \mu_k, \Sigma_k)$$

- 接下来再求导，就可以得到模型参数(μ, Σ, π)了。

五、GMM聚类方法

Kmeans与GMM的区别与联系：

- 相同点：
  - 两者都是迭代算法，并且迭代策略大致相同：算法开始对需要计算的参数赋初值，然后交替执行两个步骤。第一个步骤是对数据的估计，k-means是估计每个点所属簇，GMM是计算隐含变量的期望。第二个步骤是用第一步的估计值重新计算参数值，更新目标参数。其中k-means是计算簇心位置；GMM是计算各个高斯分布的中心位置和协方差矩阵。
- 不同点：
  - 1.k-means是计算簇心位置，GMM是计算各个高斯分布的参数。
  - 2.k-means是计算当前簇中所有元素的位置的均值，GMM是通过计算似然函数的最大值实现分布参数的求解的。

五、GMM聚类方法

请使用GMM聚类方法对所列西瓜数据集进行聚类，令高斯混合成分个数为 $K=3$

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0. 697	0. 46	11	0. 245	0. 057	21	0. 748	0. 232
2	0. 774	0. 376	12	0. 343	0. 099	22	0. 714	0. 346
3	0. 634	0. 264	13	0. 639	0. 161	23	0. 483	0. 312
4	0. 608	0. 318	14	0. 657	0. 198	24	0. 478	0. 437
5	0. 556	0. 215	15	0. 36	0. 37	25	0. 525	0. 369
6	0. 403	0. 237	16	0. 593	0. 042	26	0. 751	0. 489
7	0. 481	0. 149	17	0. 719	0. 103	27	0. 532	0. 472
8	0. 437	0. 211	18	0. 359	0. 188	28	0. 473	0. 376
9	0. 666	0. 091	19	0. 339	0. 241	29	0. 725	0. 445
10	0. 243	0. 267	20	0. 282	0. 257	30	0. 446	0. 459

五、GMM聚类方法

- 算法开始时，假定将高斯混合分布的模型参数初始化为:

$$\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$$

$$\mu_1 = x_6, \mu_2 = x_{22}, \mu_3 = x_{27}$$

$$\Sigma_1 = \Sigma_2 = \Sigma_3 = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

- 在第一轮迭代中，先计算样本由各混合成分生成的后验概率.

五、GMM聚类方法

- 以x_1为例，算出后验概率：

$$\gamma_{11} = 0.219, \gamma_{12} = 0.404, \gamma_{13} = 0.377,$$

- 所有样本的后验概率算完后，得到如下新的模型参数：

$$\alpha_1' = 0.361, \alpha_2' = 0.323, \alpha_3' = 0.316,$$

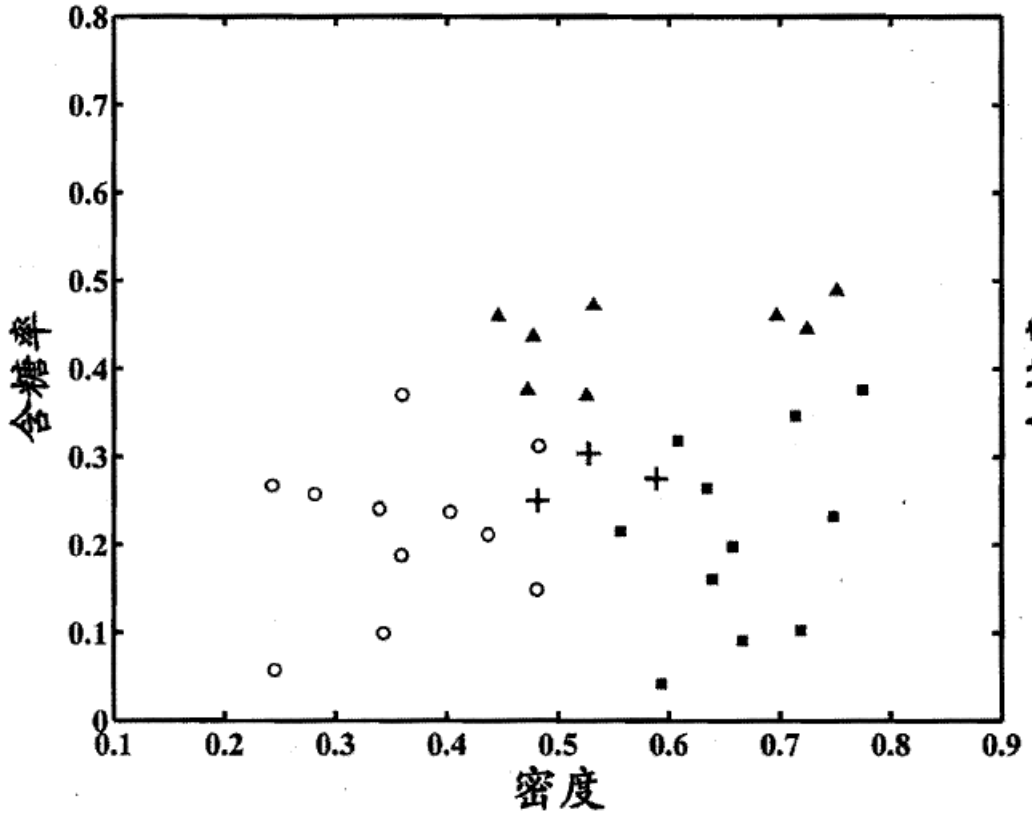
$$\mu_1' = (0{,}491, 0.251),\ \mu_2' = (0{,}571, 0.281),\ \mu_3' = (0{,}534, 0.295)$$

$$\Sigma_1' = \begin{bmatrix} 0.025 & 0.004 \\ 0.004 & 0.016 \end{bmatrix}, \Sigma_2' = \begin{bmatrix} 0.023 & 0.004 \\ 0.004 & 0.017 \end{bmatrix}, \Sigma_3' = \begin{bmatrix} 0.024 & 0.005 \\ 0.005 & 0.016 \end{bmatrix},$$

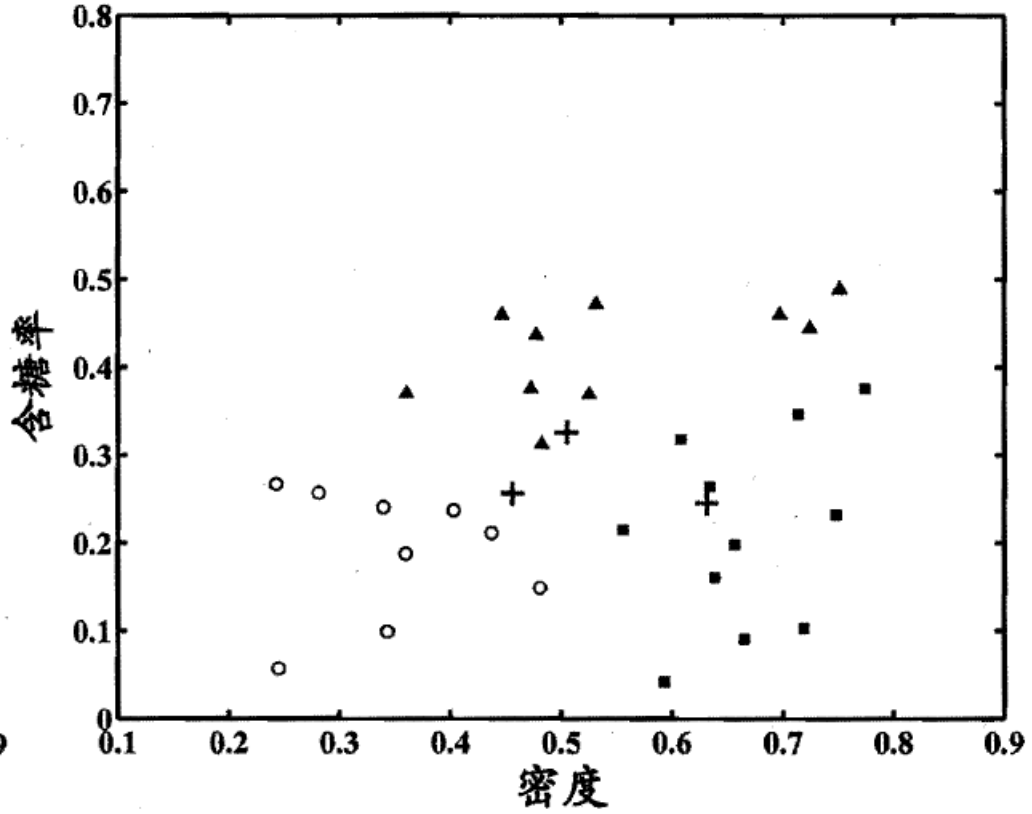
- 模型参数更新后，不断重复上述过程。

五、GMM聚类方法

- 不同轮数之后的聚类结果如图所示：



(a) 5 轮迭代后

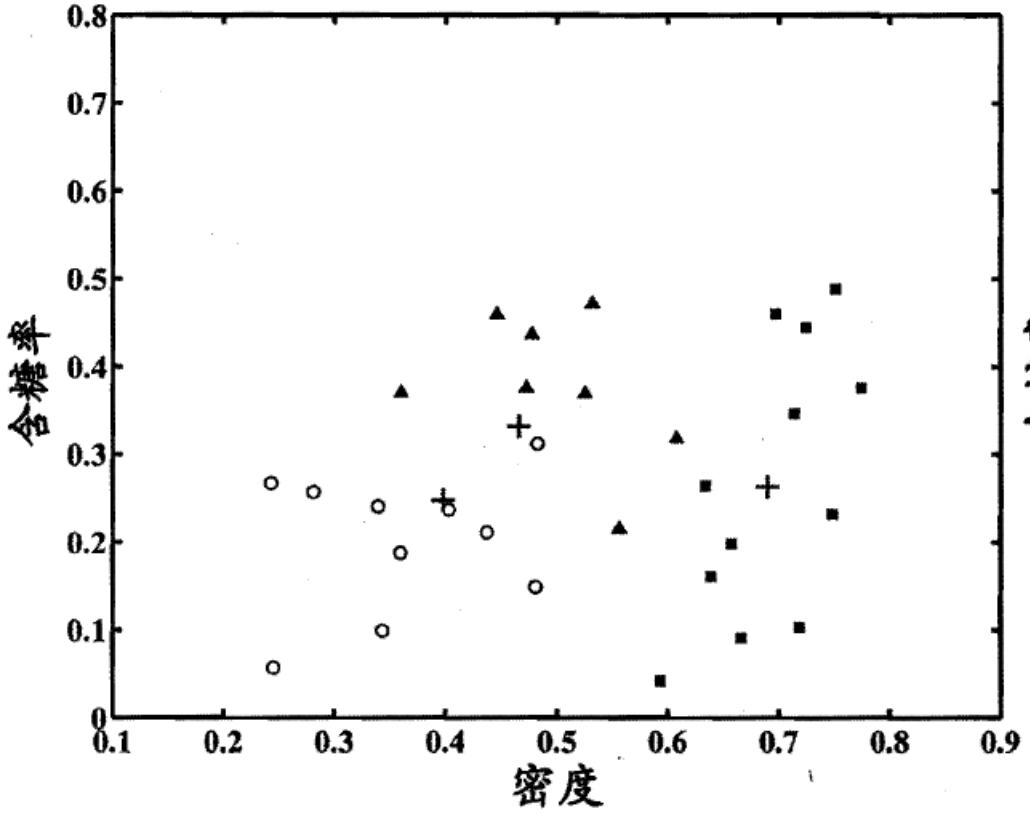


(b) 10 轮迭代后

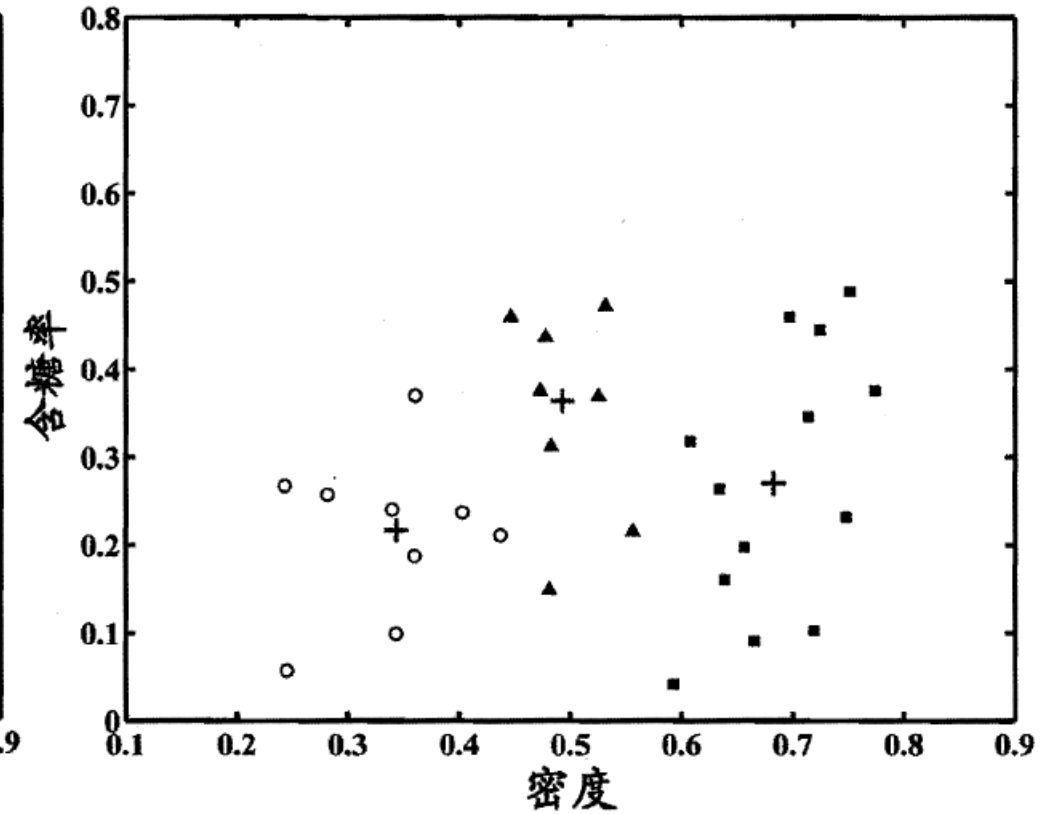
北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

五、GMM聚类方法

- 不同轮数之后的聚类结果如图所示：



(c) 20 轮迭代后



(d) 50 轮迭代后

六、案例演示

案例一：k-means

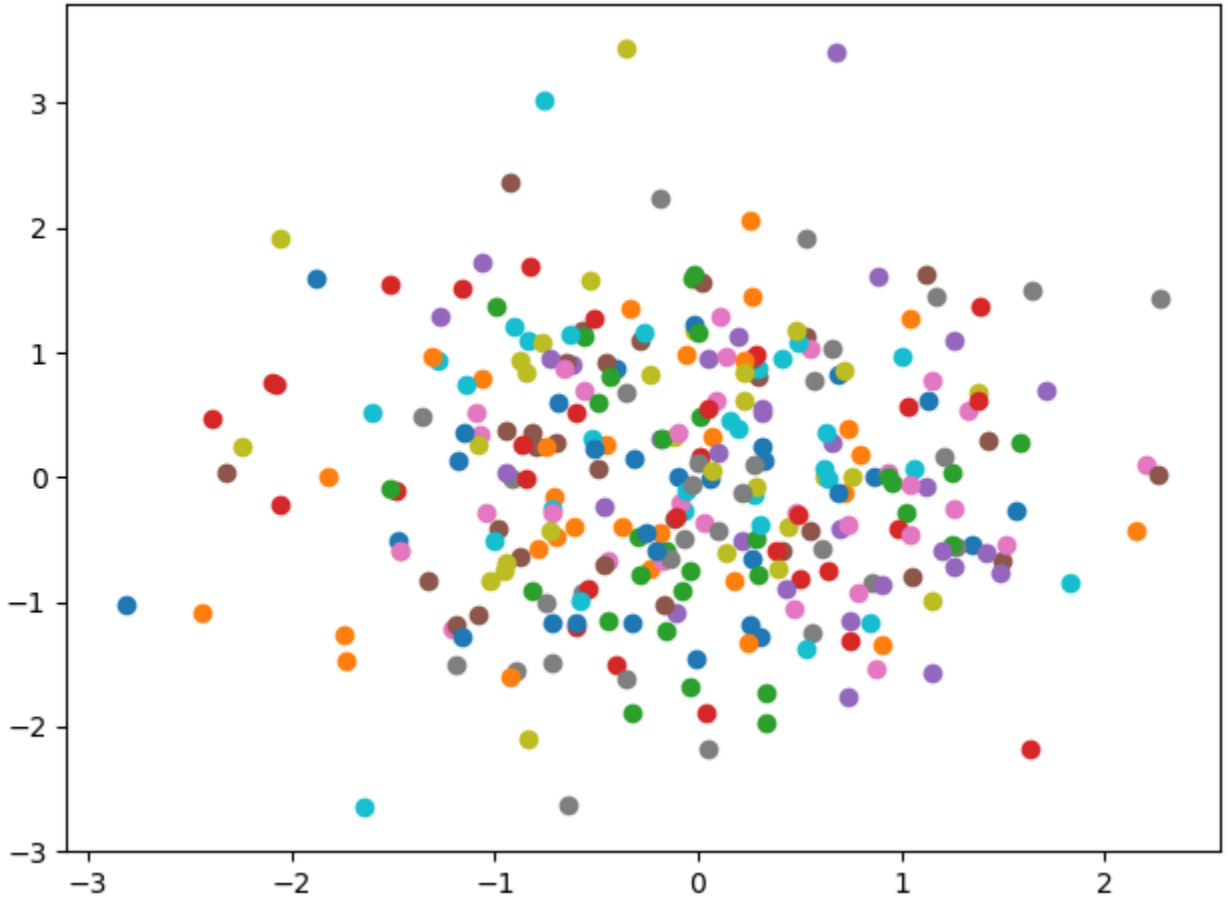
主要代码：

```
class kmeans:
    def __init__(self, k, n, node_list, iterations):
        self.k = k
        self.n = n
        self.all_nodes = node_list
        self.iter = iterations
        self.center = []
        self.res = []

    def draw(self, input_list):
            。。。

    def distance(self, x, y):
            。。。
        return res

    def get_mean(self, input_nodelist):
            。。。
        return new_center

    def exist(self, target, input_list):
            。。。
        return False

    def compute(self):
            。。。
        return self.res
```

六、案例演示

案例一：k-means

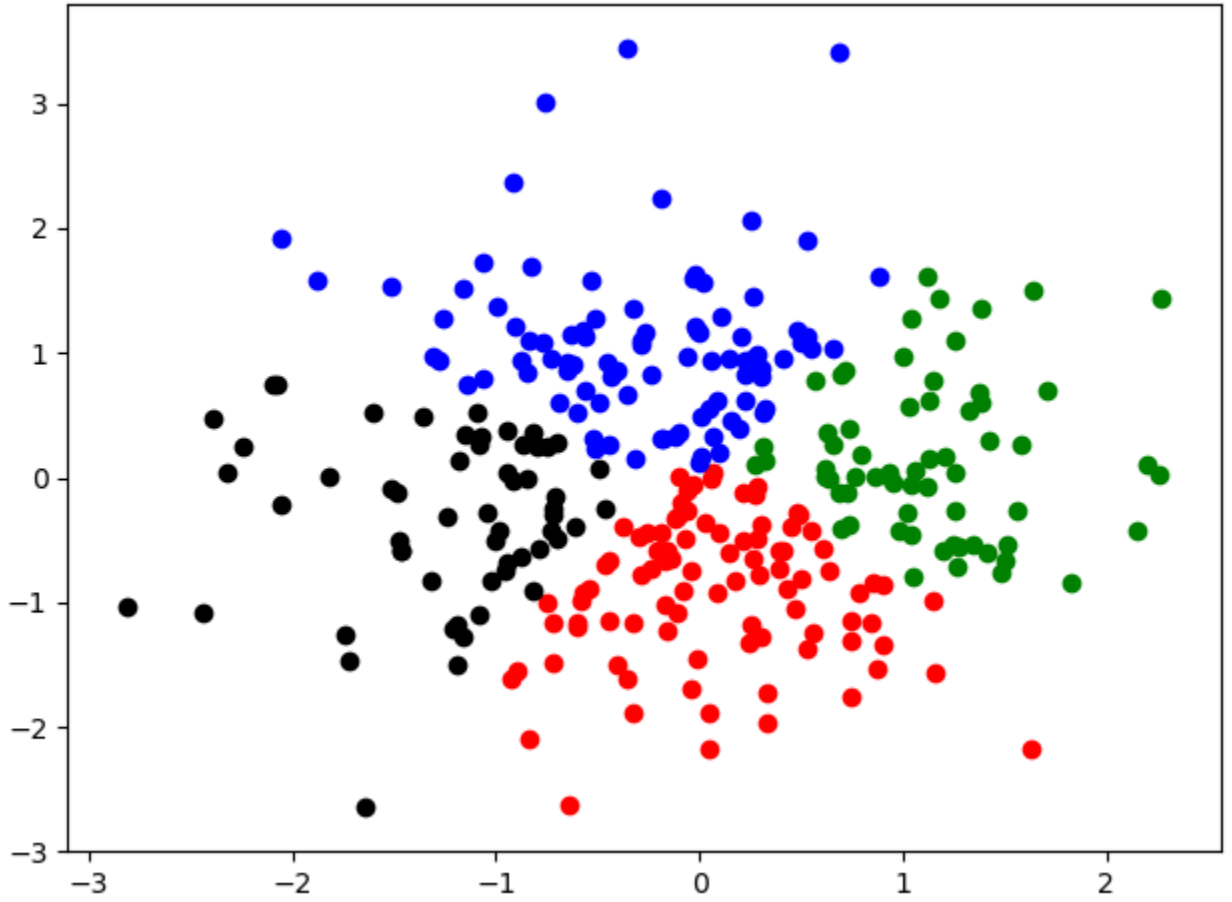
原始数据：



六、案例演示

案例一：k-means

聚类后：



六、案例演示

案例二：AGNES

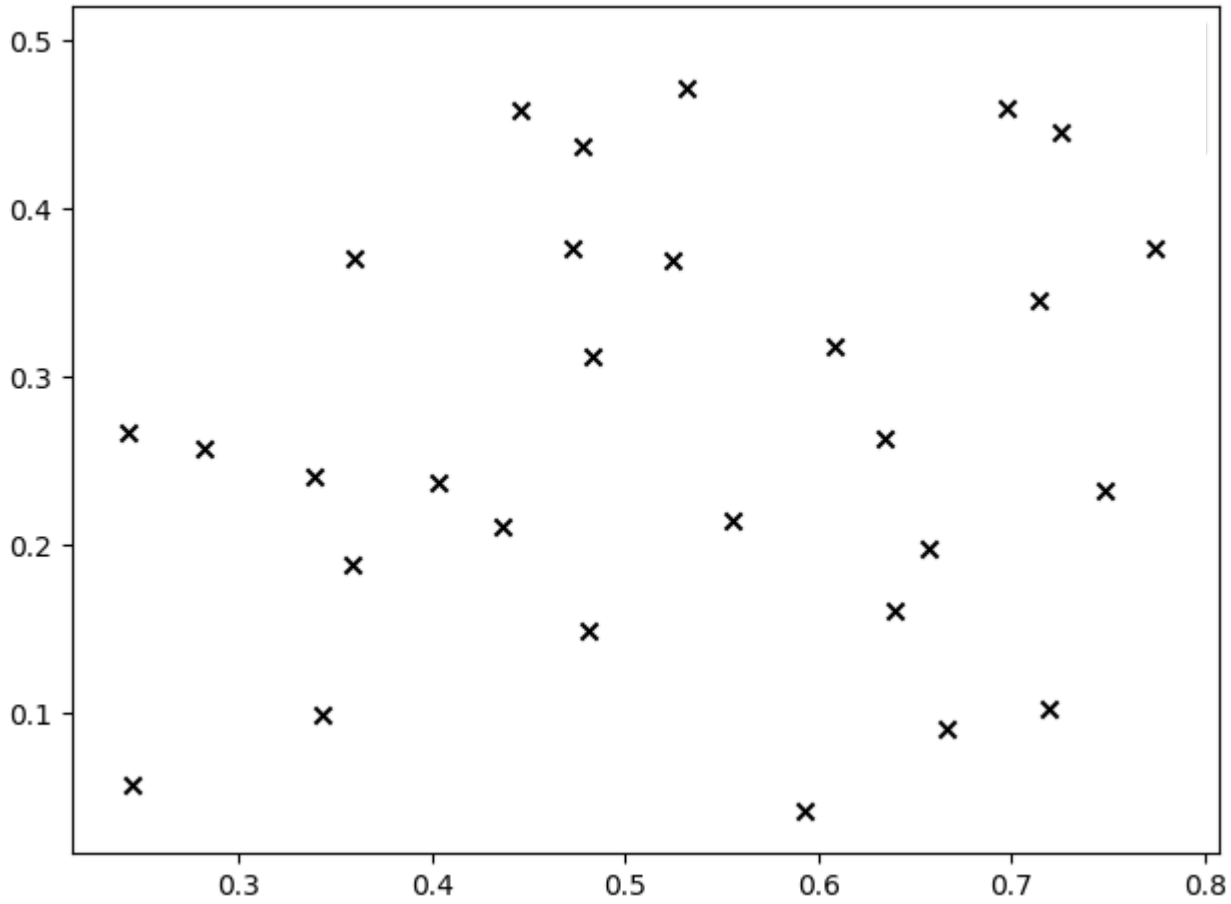
主要代码：

```
#计算欧几里得距离,a,b分别为两个元组
def dist(a, b):
    return math.sqrt(math.pow(a[0]-b[0], 2)+math.pow(a[1]-b[1], 2))

。。。

#找到距离最小的下标
def find_Min(M):
    。。。
    return (x, y, min)

#算法模型:
def AGNES(dataset, dist, k):
    #初始化C和M
    C = [];M = []
    for i in dataset:
        。。。
    #合并更新
  while q > k:
      。。。
        q -= 1
    return C
#画图
def draw(C):
    colValue = ['r', 'y', 'g', 'b', 'c', 'k', 'm']
    。。。
    pl.legend(loc='upper right')
    pl.show()
```

六、案例演示

案例二：AGNES

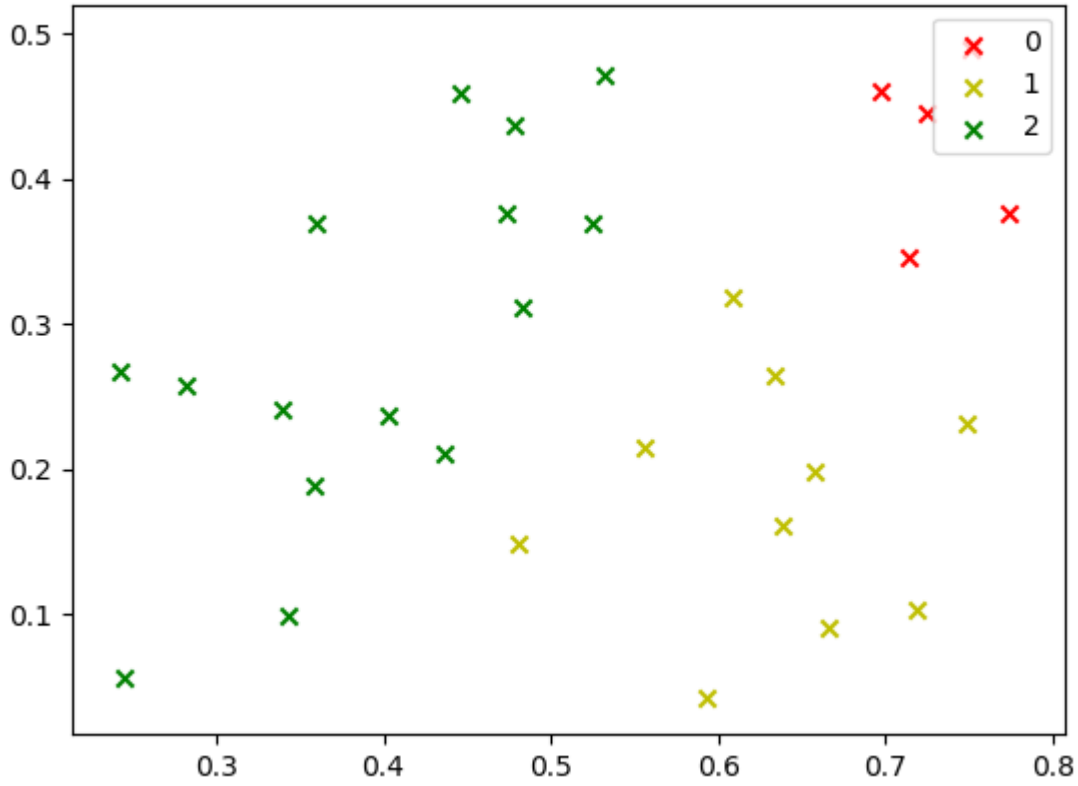
原始数据：



六、案例演示

案例二：AGNES

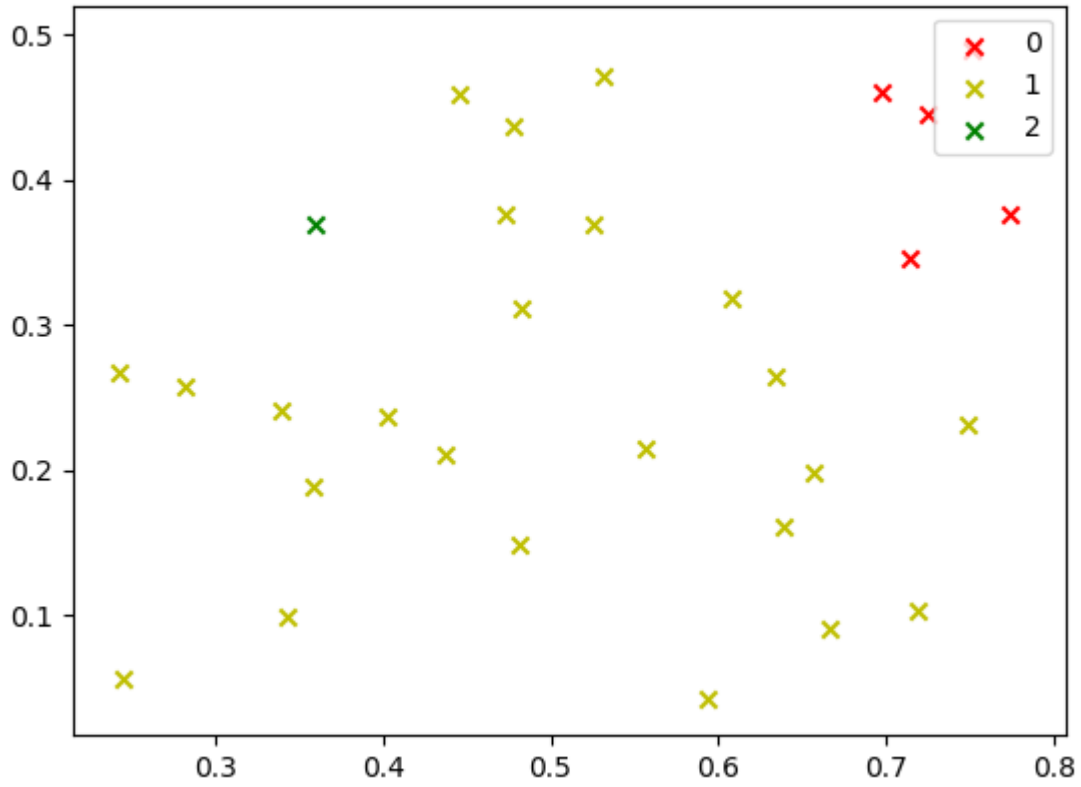
聚类后（采用dist_avg运行结果）：



六、案例演示

案例二：AGNES

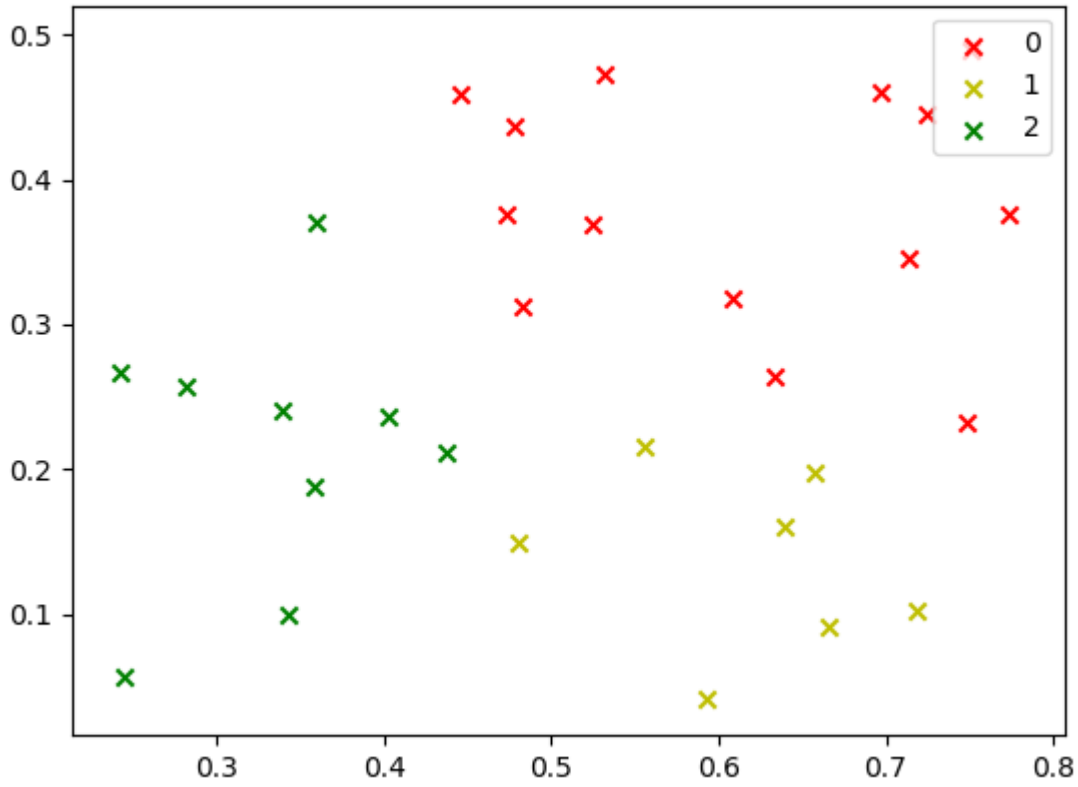
聚类后（采用dist_min运行结果）：



六、案例演示

案例二：AGNES

聚类后（采用dist_max运行结果）：



六、案例演示

案例三：DBSCAN

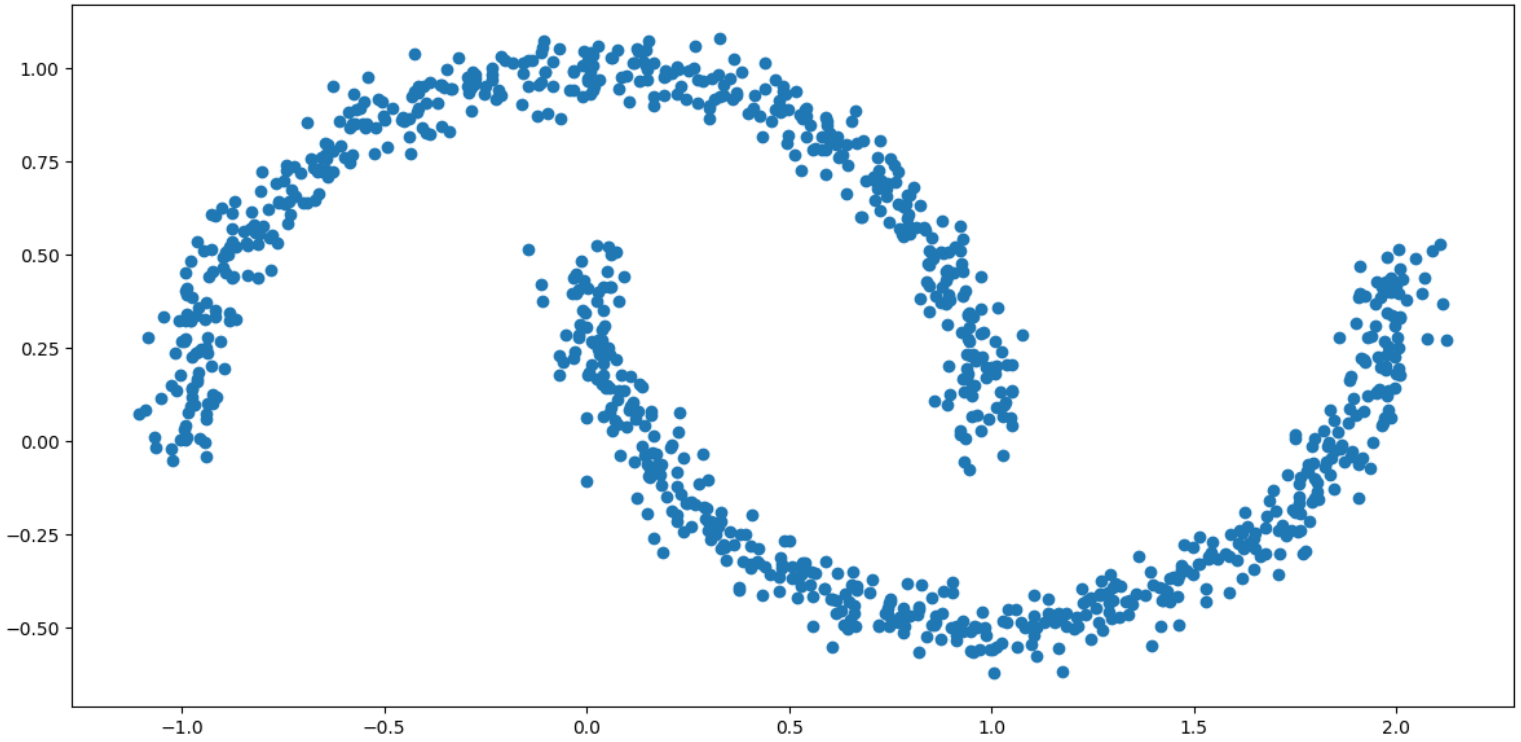
主要代码：

```
def cluster_and_plot(epsilon, min_samples):
    dbscan = DBSCAN(eps=epsilon, min_samples=min_samples)
    dbscan.fit(X)
    plt.figure(figsize=(12, 6))
    # 绘制簇群
    plt.scatter(X[dbscan.labels_>-1, 0], X[dbscan.labels_>-1, 1], c=dbscan.labels_[dbscan.labels_>-1])
    # 绘制异常点
    anomaly_predicates = dbscan.labels_==-1
    n_anomalies = sum(anomaly_predicates)
。。。
# 设置参数
epsilon = 0.1
min_samples = 10
#在大多数聚类算法中，第一步是计算距离矩阵。
distance_matrix=euclidean_distances(X)
。。。

labels = -np.ones((1000,), dtype=np.int)
cluster_id = 0
loop_index = 0
while sum(labels < 0) > 0:
    # unlabelled_predicates = (labels==-1)
    。。。
clusters = np.unique(labels)
for cluster_id in [c for c in clusters if c>-1]:
    #找到已经分配了 cluster_id 的点
    cluster_predicates = labels==cluster_id
    #找到上面的点的邻居
    neighbours_matrix = neighbour_predicate_matrix[cluster_predicates]
    neighbours_predicates = neighbours_matrix.any(axis=0)
    #从邻居中删除已经聚类的点。
    clustered_non_core_indices=indices[neighbours_predicates & ~cluster_predicates]
    #标记剩余的邻居
    labels[clustered_non_core_indices] = cluster_id
```

六、案例演示

案例三：DBSCAN

原始数据：



六、案例演示

案例三：DBSCAN

聚类后：