

机器学习

07 聚类

李祎

liyi@dlut.edu.cn



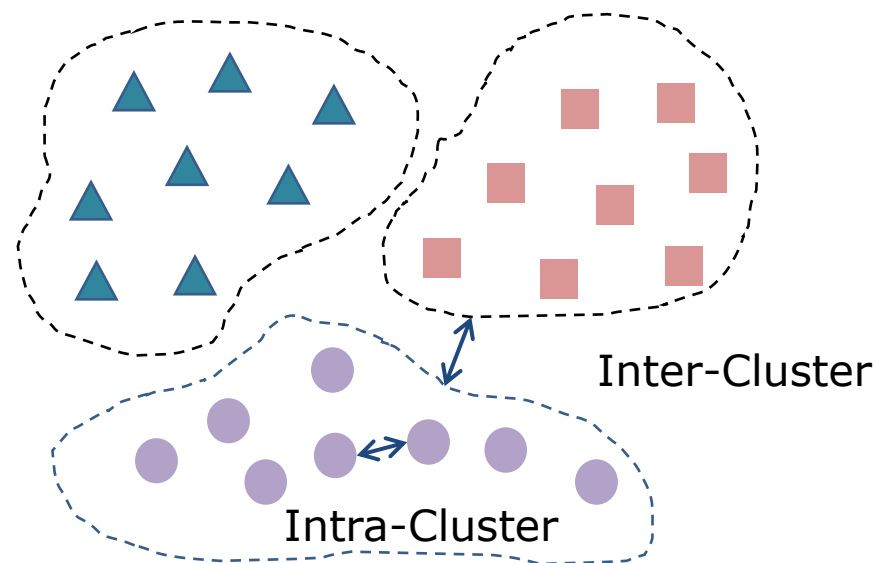
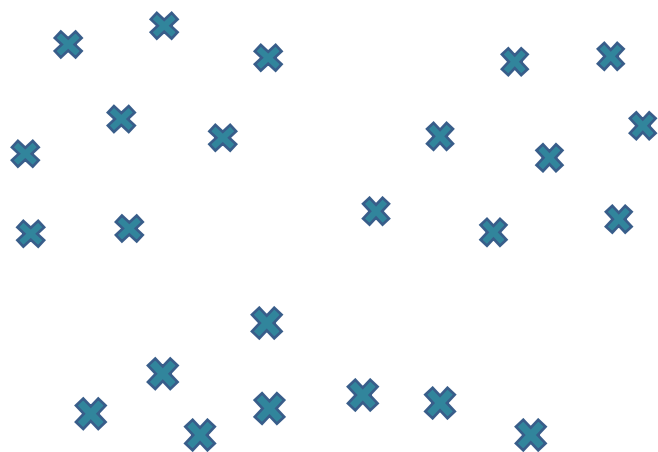
大连理工大学 人工智能学院
School of Artificial Intelligence, Dalian University of Technology

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

什么是聚类



- 无监督学习：No labels, Data driven
- 聚类目标：将数据集中的样本划分为若干个通常不相交的子集（“簇”，cluster），子集内部具有相似性，子集之间具有差异性。
- 聚类既可以作为一个单独过程（用于找寻数据内在的分布结构），也可作为分类等其他学习任务的前驱过程。



- 假定样本集 $D = \{x_1, x_2, \dots, x_m\}$ 包含 m 个无标记样本，每个样本 $x_i = (x_{i1}; x_{i2}; \dots; x_{in})$ 是一个 n 维的特征向量，聚类算法将样本集 D 划分成 k 个不相交的簇 $\{C_l | l = 1, 2, \dots, k\}$ 。其中 $C_{l'} \cap_{l' \neq l} C_l = \phi$ ，且 $D = \bigcup_{l=1}^k C_l$ 。
- 相应地，用 $\lambda_j \in \{1, 2, \dots, k\}$ 表示样本 x_j 的“簇标记”（即 **cluster label**），即 $x_j \in C_{\lambda_j}$ 。于是，聚类的结果可用包含 m 个元素的簇标记向量 $\lambda = \{\lambda_1; \lambda_2; \dots; \lambda_m\}$ 表示。

□ Marketing

- Finding groups of customers with similar behaviours.

□ Biology

- Finding groups of animals or plants with similar features.

□ Earthquake Studies

- Clustering observed earthquake epicenters to identify dangerous zones.

□ WWW

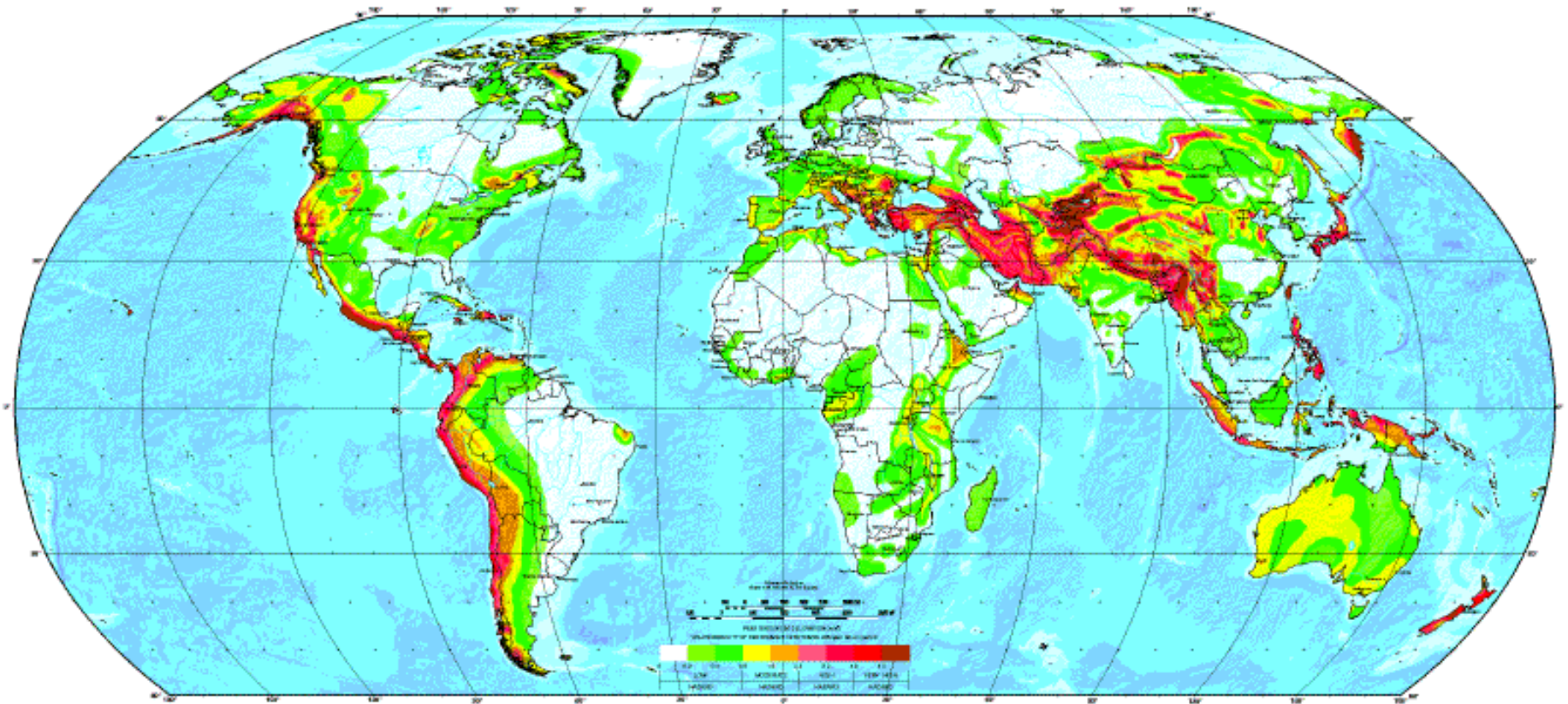
- Clustering weblog data to discover groups of similar access patterns.

□ Social Networks

- Discovering groups of individuals with close friendships internally.

□

GLOBAL SEISMIC HAZARD MAP



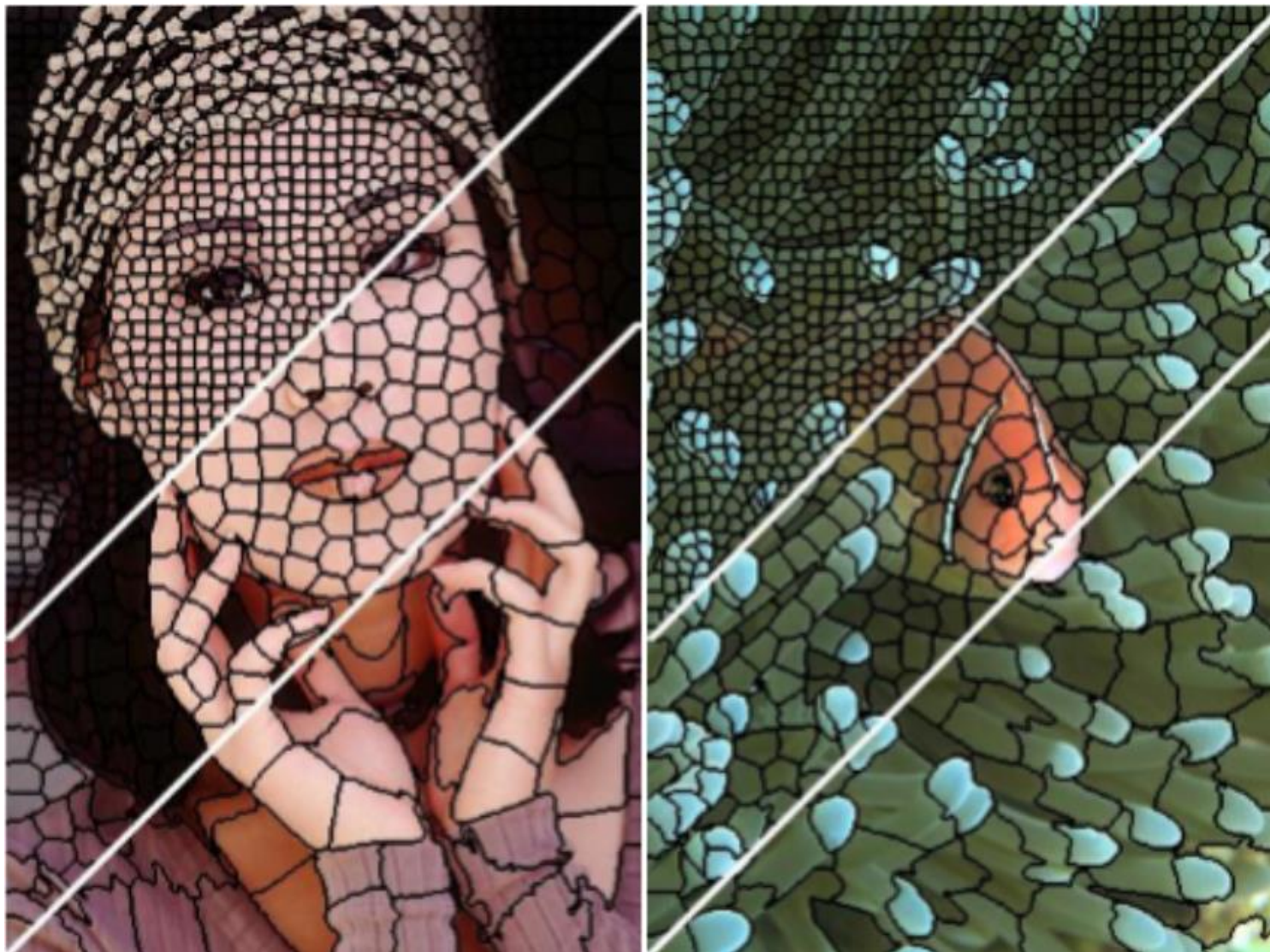
图像分割



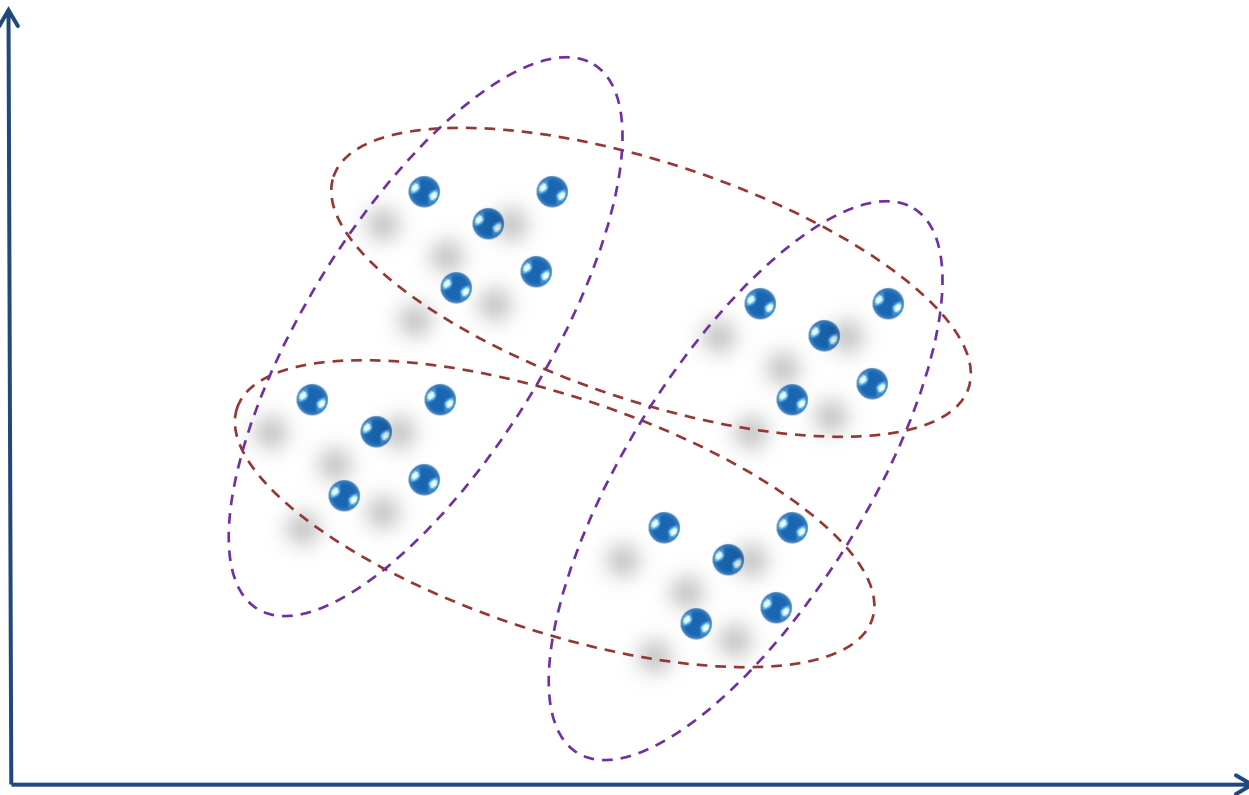
大连理工大学 人工智能学院
School of Artificial Intelligence, Dalian University of Technology



超像素



选择哪种？



□ 两个基本问题：

- 性能度量：1. 评估结果好坏，2. 作为优化目标
- 距离计算：有序、无序

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。

- 大致分为以下两类：

- 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。

对数据集 $D = \{x_1, x_2, \dots, x_m\}$ ，假定通过聚类得到的簇划分为 $C = \{C_1, C_2, \dots, C_k\}$ ，参考模型给出的簇划分为 $C^* = \{C_1^*, C_2^*, \dots, C_s^*\}$ 。相应地，令 λ 与 λ^* 分别表示与 C 和 C^* 对应的簇标记向量。

我们将样本两两配对考虑，定义

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

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

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

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

性能度量 - 外部指标



□ Jaccard系数 (Jaccard Coefficient, JC)

$$JC = \frac{a}{a+b+c}$$

□ FM指数 (Fowlkes and Mallows Index, FMI)

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$$

□ Rand指数 (Rand Index, RI)

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

[0,1]区间内,
越大越好.

□ 考虑聚类结果的簇划分 $C = \{C_1, C_2, \dots, C_k\}$ 定义

簇 C 内样本间的平均距离

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

簇 C 内样本间的最远距离

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

簇 C_i 与簇 C_j 最近样本间的距离

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

簇 C_i 与簇 C_j 中心点间的距离

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

□ DB指数 (Davies-Bouldin Index, DBI)

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

越小越好.

□ Dunn指数 (Dunn Index, DI)

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

越大越好.

距离计算

□ 距离度量的性质：

非负性： $dist(x_i, x_j) \geq 0$

同一性： $dist(x_i, x_j) = 0$ 当且仅当 $x_i = x_j$

对称性： $dist(x_i, x_j) = dist(x_j, x_i)$

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

□ 常用距离：

闵可夫斯基距离 (Minkowski distance) :

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

p=2: 欧氏距离 (Euclidean distance) .

p=1: 曼哈顿距离 (Manhattan distance) 或 街区距离.

□ 属性介绍

- 连续属性 (continuous attribute)

在定义域上有无穷多个可能的取值

- 离散属性 (categorical attribute)

在定义域上是有限个可能的取值

- 有序属性 (ordinal attribute)

例如定义域为 $\{1,2,3\}$ 的离散属性，“1”与“2”比较接近、与“3”比较远，称为“有序属性”。

- 无序属性 (non-ordinal attribute)

例如定义域为{飞机，火车，轮船}这样的离散属性，不能直接在属性值上进行计算，称为“无序属性”。

□ Value Difference Metric, VDM (处理无序属性) :

令 $m_{u,a}$ 表示属性 u 上取值为 a 的样本数, $m_{u,a,i}$ 表示在第 i 个样本簇中在属性 u 上取值为 a 的样本数, k 为样本数, 则属性 u 上两个离散值 a 与 b 之间的VDM距离为

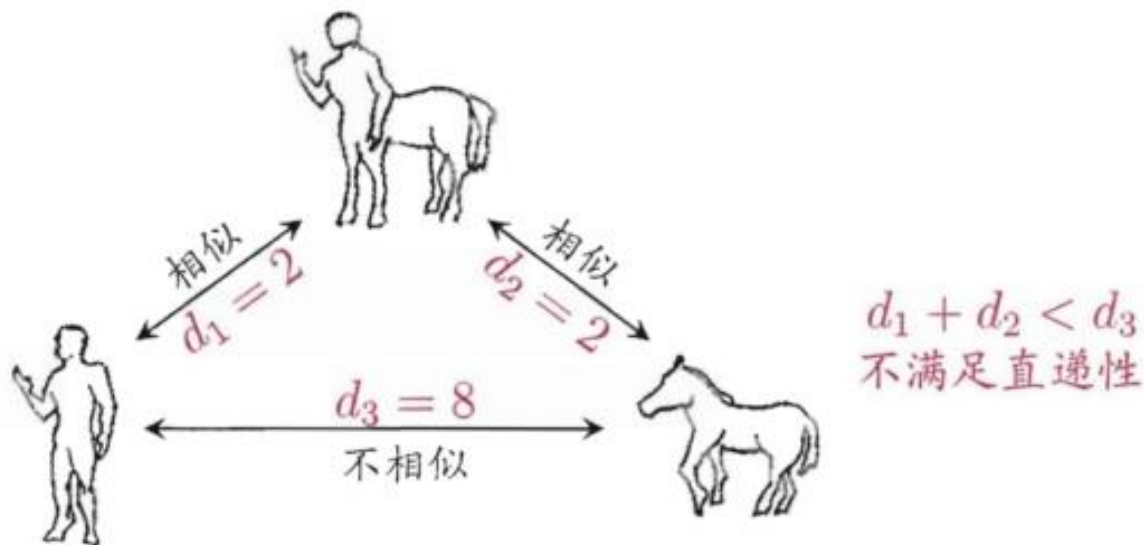
$$VDM_p(a, b) = \sum_{i=1}^k \left| \frac{m_{u,a,i}}{m_{u,a}} - \frac{m_{u,b,i}}{m_{u,b}} \right|^p$$

□ 例如:

- 计算出行方式上飞机和火车这两个离散值的VDM距离, 即为

$$\sum_{i=1}^k \left| \frac{\text{在簇}i\text{中出行方式为飞机的人数}}{\text{出行方式为飞机的总人数}} - \frac{\text{在簇}i\text{中出行方式为火车的人数}}{\text{出行方式为火车的总人数}} \right|$$

- 用于度量相似性的“距离”未必满足距离的所有基本性质，尤其是直递性，即“非度量距离”；
- 距离不一定是提前定义好的，可以根据数据样本进行动态学习，即“距离度量学习”。



原型聚类

□ 原型聚类

也称为“基于原型的聚类” (prototype-based clustering), 此类算法假设聚类结构能通过一组原型刻画。

□ 算法过程:

通常情况下, 算法先对原型进行初始化, 再对原型进行迭代更新求解。

□ 几种著名的原型聚类算法:

k -means、学习向量量化 (LVQ)、高斯混合聚类 (GMM)

给定数据集 $D = \{x_1, x_2, \dots, x_m\}$, k 均值算法针对聚类所得簇划分

$C = \{C_1, C_2, \dots, C_k\}$ 最小化平方误差

$$E = \sum_{i=1}^k \sum_{x \in C_j} \|x - \mu_i\|_2^2$$

其中, μ_i 是簇 C_i 的均值向量。

□ 算法流程（迭代优化）：

初始化每个簇的均值向量

repeat

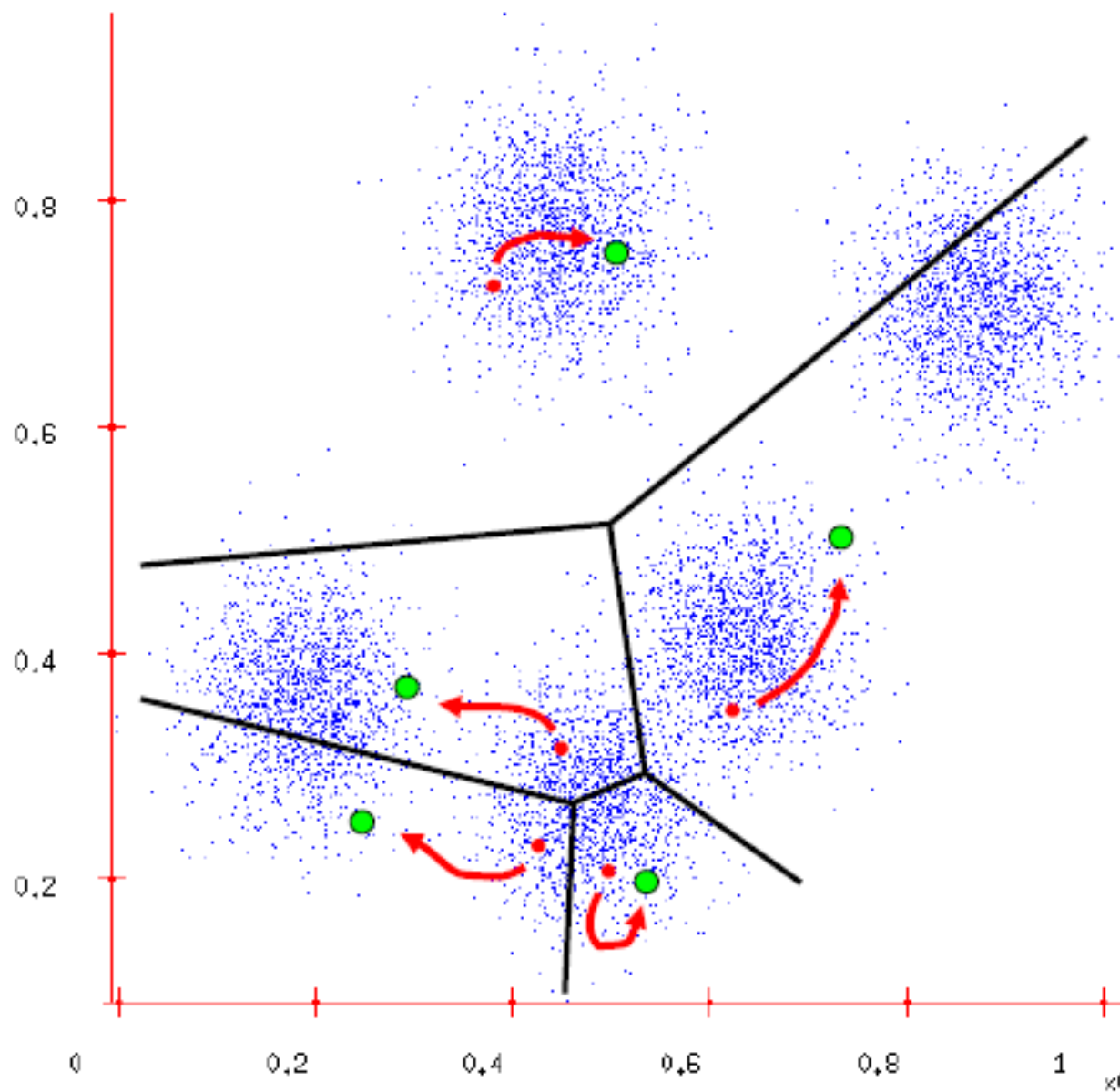
1. （更新）簇划分；
2. 计算每个簇的均值向量

until 当前均值向量均未更新

原型聚类 - k 均值算法



大连理工大学 人工智能学院
School of Artificial Intelligence, Dalian University of Technology



原型聚类 - k 均值算法



输入: 样本集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$;
聚类簇数 k .

过程:

- 1: 从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k\}$
 - 2: **repeat**
 - 3: 令 $C_i = \emptyset$ ($1 \leq i \leq k$)
 - 4: **for** $j = 1, \dots, m$ **do**
 - 5: 计算样本 \mathbf{x}_j 与各均值向量 $\boldsymbol{\mu}_i$ ($1 \leq i \leq k$) 的距离: $d_{ji} = \|\mathbf{x}_j - \boldsymbol{\mu}_i\|_2$;
 - 6: 根据距离最近的均值向量确定 \mathbf{x}_j 的簇标记: $\lambda_j = \arg \min_{i \in \{1, 2, \dots, k\}} d_{ji}$;
 - 7: 将样本 \mathbf{x}_j 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$;
 - 8: **end for**
 - 9: **for** $i = 1, \dots, k$ **do**
 - 10: 计算新均值向量: $\boldsymbol{\mu}'_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$;
 - 11: **if** $\boldsymbol{\mu}'_i \neq \boldsymbol{\mu}_i$ **then**
 - 12: 将当前均值向量 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}'_i$
 - 13: **else**
 - 14: 保持当前均值向量不变
 - 15: **end if**
 - 16: **end for**
 - 17: **until** 当前均值向量均未更新
 - 18: **return** 簇划分结果
- 输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

□ k 均值算法实例

接下来以表9-1的西瓜数据集4.0为例，来演示 k 均值算法的学习过程。将编号为 i 的样本称为 x_i 。

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	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	0.360	0.370	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

原型聚类 - k 均值算法



□ 假定聚类簇数 $k=3$ ，算法开始时，随机选择3个样本 x_6, x_{12}, x_{27} 作为初始均值向量，即 $\mu_1 = (0.403; 0.237), \mu_2 = (0.343; 0.099), \mu_3 = (0.533; 0.472)$ 。

考察样本 $x_1 = (0.697; 0.460)$ ，它与当前均值向量 μ_1, μ_2, μ_3 的距离分别为0.369, 0.506, 0.166，因此 x_1 将被划入簇 C_3 中。类似的，对数据集中的所有样本考察一遍后，可得当前簇划分为

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

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

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

于是，可以从分别求得新的均值向量

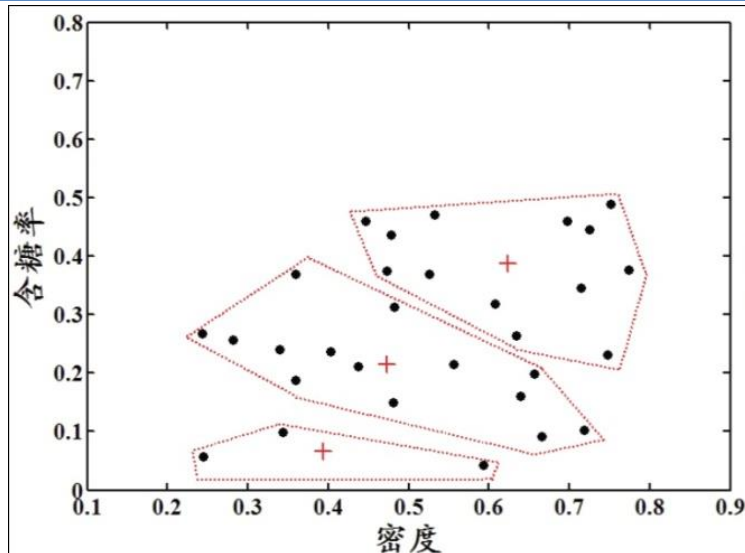
$$\mu'_1 = (0.473; 0.214), \mu'_2 = (0.394; 0.066), \mu'_3 = (0.623; 0.388)$$

不断重复上述过程，如下图所示。

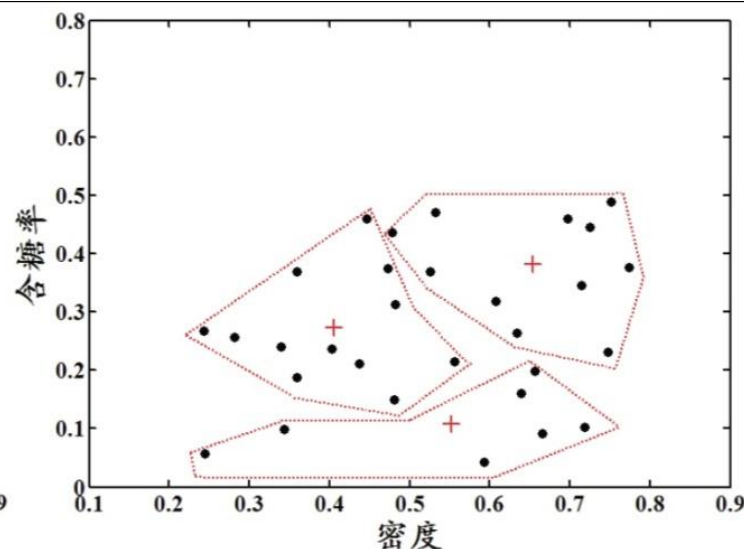
原型聚类 - k 均值算法



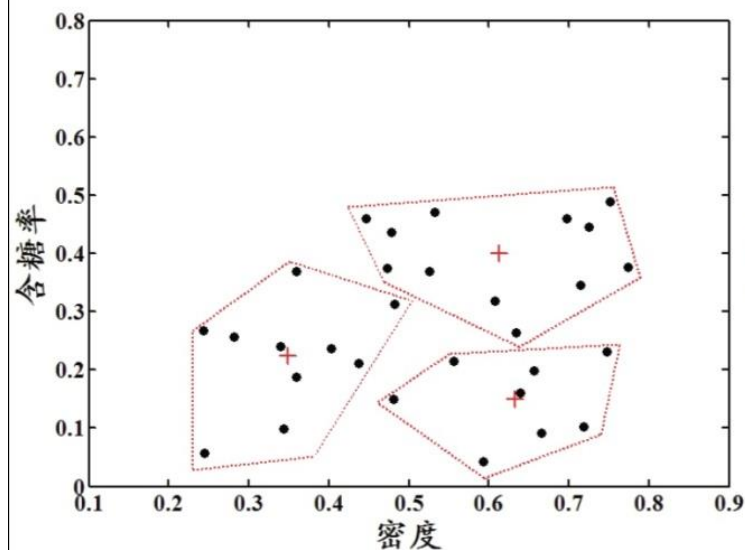
大连理工大学 人工智能学院
School of Artificial Intelligence, Dalian University of Technology



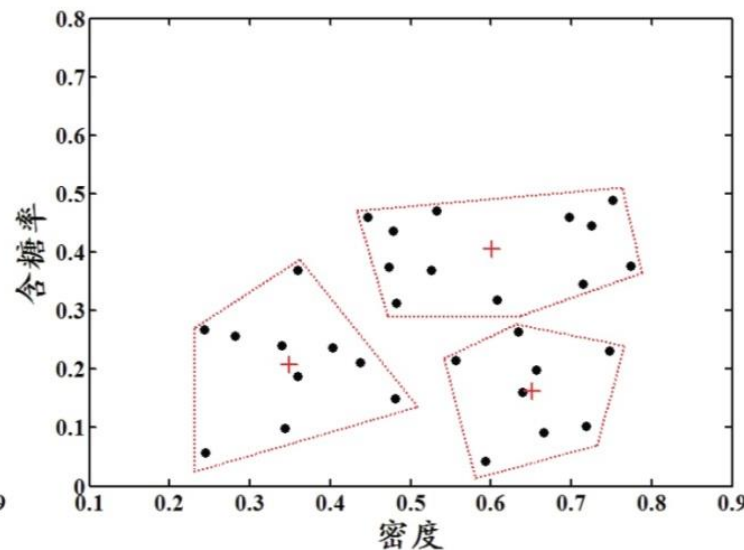
(a) 第一轮迭代后



(b) 第二轮迭代后



(c) 第三轮迭代后



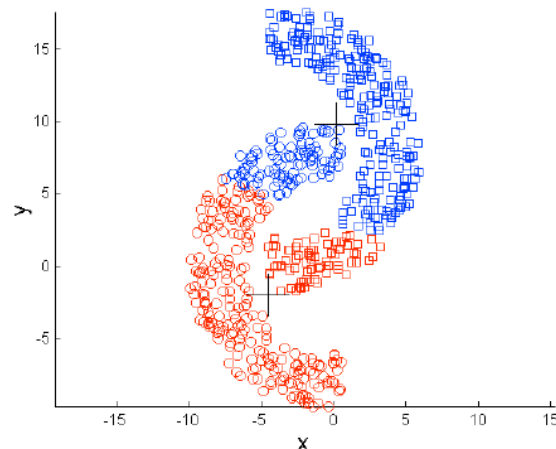
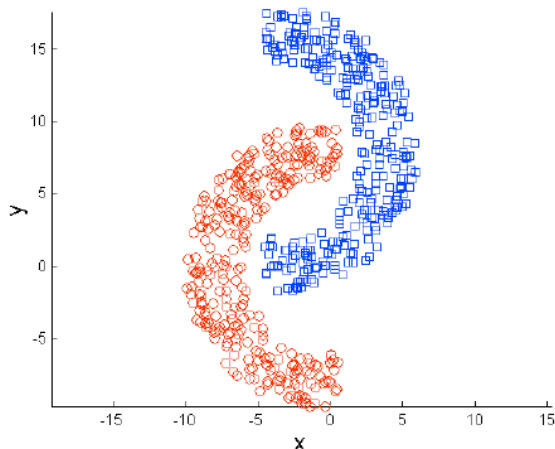
(d) 第四轮迭代后

□ 优点：

- 原理简单，容易实现
- 可解释度较强

□ 缺点：

- K值很难确定
- 可能陷入局部最优
- 对噪音和异常点敏感
- 聚类效果依赖于聚类中心的初始化
- 对于非凸数据集或类别规模差异太大的数据效果不好



与 k 均值用原型向量来刻画聚类结构不同，高斯混合模型（Gaussian Mixture Model）采用概率模型来表达聚类原型：

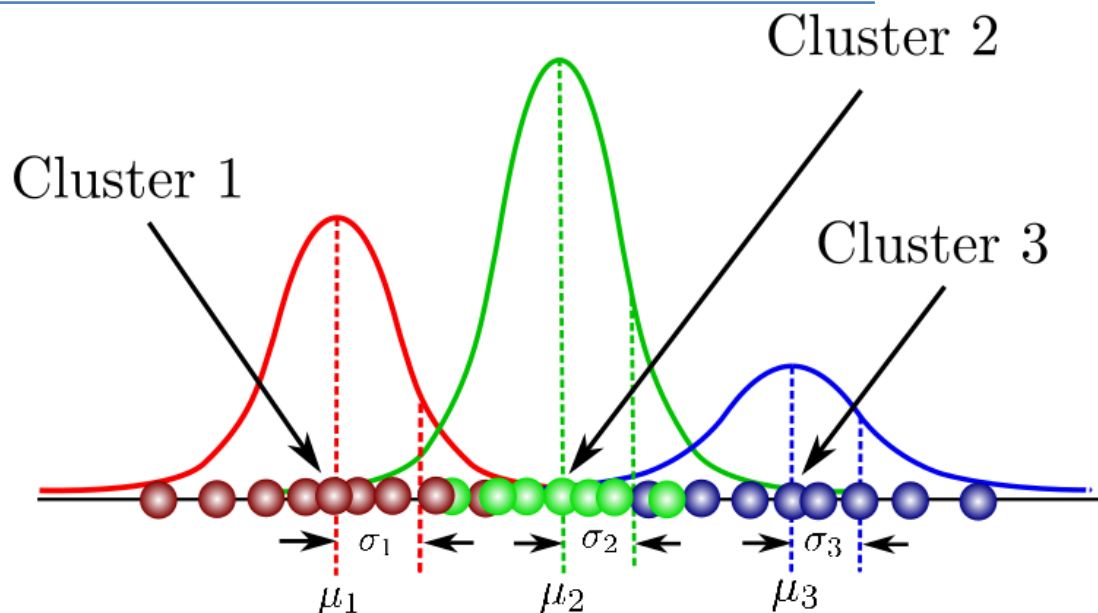
□ 多元高斯分布的定义

对 n 维样本空间中的随机向量 x ，若 x 服从高斯分布，其概率密度函数为

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

其中 μ 是 n 维均值向量， Σ 是 $n \times n$ 的协方差矩阵。也可将概率密度函数记作 $p(x|\mu, \Sigma)$ 。

原型聚类 - 高斯混合聚类



□ 高斯混合分布的定义

$$p_M(x) = \sum_{i=1}^k \alpha_i p(x|\mu_i, \Sigma_i)$$

该分布由 k 个混合分布组成，每个分布对应一个高斯分布。其中， μ_i 与 Σ_i 是第 i 个高斯混合成分的参数。而 $\alpha_i > 0$ 为相应的“混合系数”， $\sum_{i=1}^k \alpha_i = 1$ 。

高斯混合聚类



输入: 样本集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$;
高斯混合成分个数 k .

过程:

1: 初始化高斯混合分布的模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$

2: **repeat**

3: **for** $j = 1, \dots, m$ **do**

E 4: 根据(9.30)计算 \mathbf{x}_j 由各混合成分生成的后验概率, 即
步 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \mathbf{x}_j) \quad (1 \leq i \leq k)$ ←

5: **end for**

6: **for** $i = 1, \dots, k$ **do**

7: 计算新均值向量: $\boldsymbol{\mu}'_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}$;

M 8: 计算新协方差矩阵: $\boldsymbol{\Sigma}'_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \boldsymbol{\mu}'_i)(\mathbf{x}_j - \boldsymbol{\mu}'_i)^\top}{\sum_{j=1}^m \gamma_{ji}}$;

步 9: 计算新混合系数: $\alpha'_i = \frac{\sum_{j=1}^m \gamma_{ji}}{m}$;

10: **end for**

11: 将模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$ 更新为 $\{(\alpha'_i, \boldsymbol{\mu}'_i, \boldsymbol{\Sigma}'_i) \mid 1 \leq i \leq k\}$

12: **until** 满足停止条件

13: $C_i = \emptyset \quad (1 \leq i \leq k)$

14: **for** $j = 1, \dots, m$ **do**

15: 根据(9.31)确定 \mathbf{x}_j 的簇标记 λ_j ;

16: 将 \mathbf{x}_j 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$

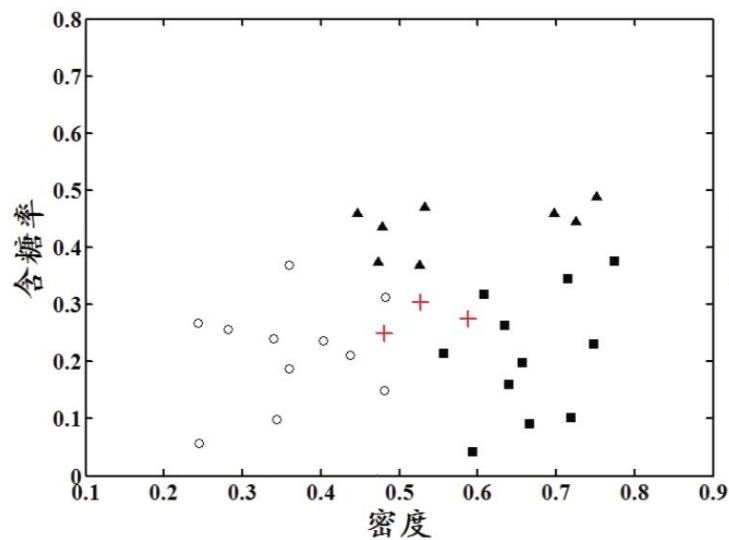
17: **end for**

18: **return** 簇划分结果

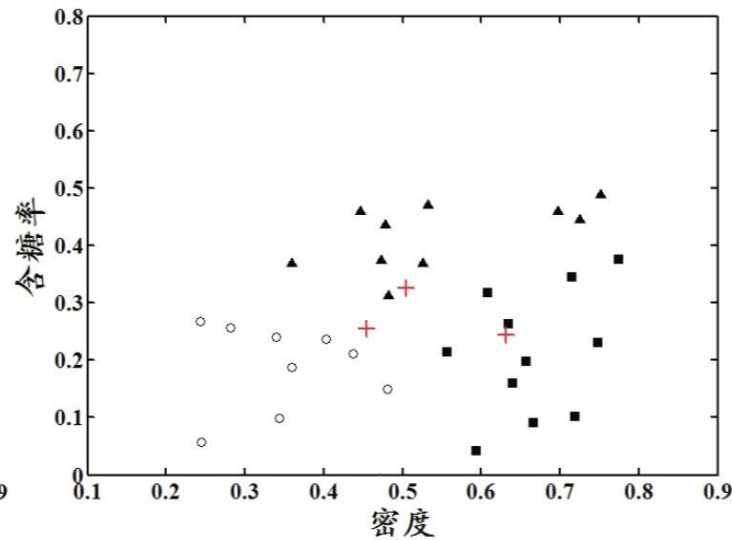
输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

$$\begin{aligned} p_{\mathcal{M}}(z_j = i \mid \mathbf{x}_j) &= \frac{P(z_j = i) \cdot p_{\mathcal{M}}(\mathbf{x}_j \mid z_j = i)}{p_{\mathcal{M}}(\mathbf{x}_j)} \\ &= \frac{\alpha_i \cdot p(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^k \alpha_l \cdot p(\mathbf{x}_j \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} \end{aligned}$$

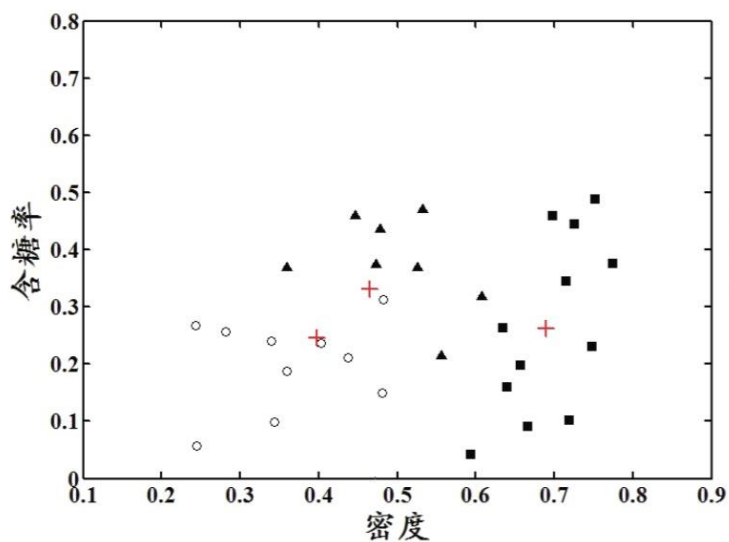
高斯混合聚类



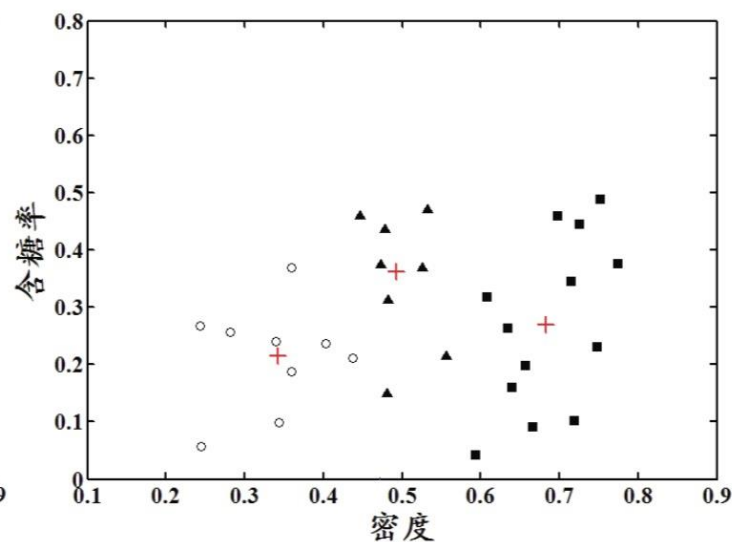
(a) 5轮迭代后



(b) 10轮迭代后



(c) 20轮迭代后



(d) 50轮迭代后

密度聚类

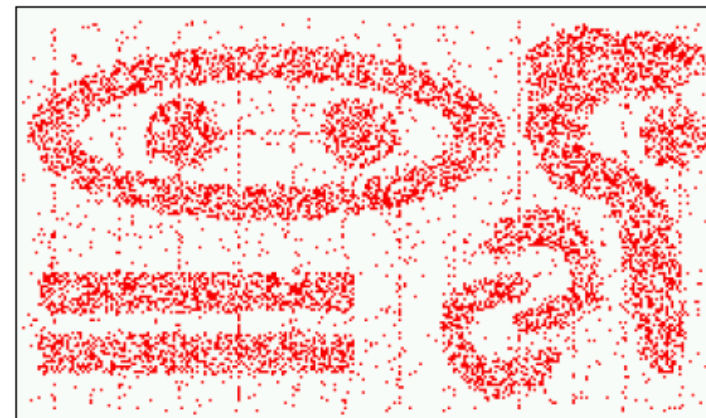
□ 原型聚类：假设**聚类结构**能够通过**一组原型**刻画

□ 密度聚类：也称为“基于密度的聚类”

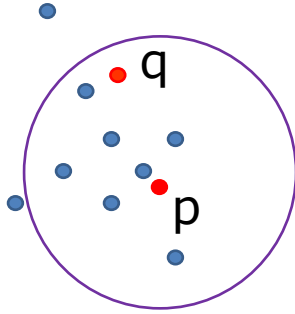
- 此类算法假设聚类结构能通过**样本分布的紧密程度**来确定。
- 通常情况下，密度聚类算法从样本密度的角度来考察**样本之间的可连接性**，并基于可连接样本**不断扩展**聚类簇来获得最终的聚类结果。

□ 特点

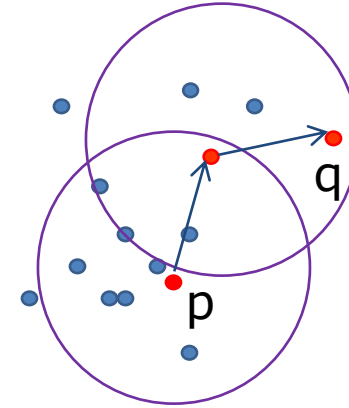
- Generate clusters of arbitrary shapes.
- Robust against noise.
- No K value required in advance.
- Somewhat similar to human vision.



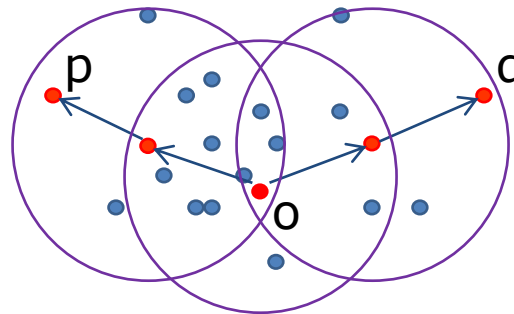
- DBSCAN算法：基于一组“邻域”参数($\epsilon, MinPts$)来刻画样本分布的紧密程度。
- 基本概念：
 - **邻域**：对样本 $x_j \in D$ 其 ϵ 邻域包含样本集 D 中与 x_j 的距离不大于 ϵ 的样本；
 - **核心对象**：若样本 x_j 的 ϵ 邻域至少包含 $MinPts$ 个样本，则该样本点为一个核心对象；
 - **密度直达**：若样本 x_j 位于样本 x_i 的 ϵ 邻域中，且 x_i 是一个核心对象，则称样本 x_j 由 x_i 密度直达；
 - **密度可达**：对样本 x_i 与 x_j ，若存在样本序列 p_1, p_2, \dots, p_n ，其中 $p_1 = x_i, p_n = x_j$ 且 p_{i+1} 由 p_i 密度直达，则该两样本密度可达；
 - **密度相连**：对样本 x_i 与 x_j ，若存在样本 x_k 使得两样本均由 x_k 密度可达，则称该两样本密度相连。



directly density reachable



density reachable



density connected

□ 例如：

□ 令 $MinPts = 3$ ，则

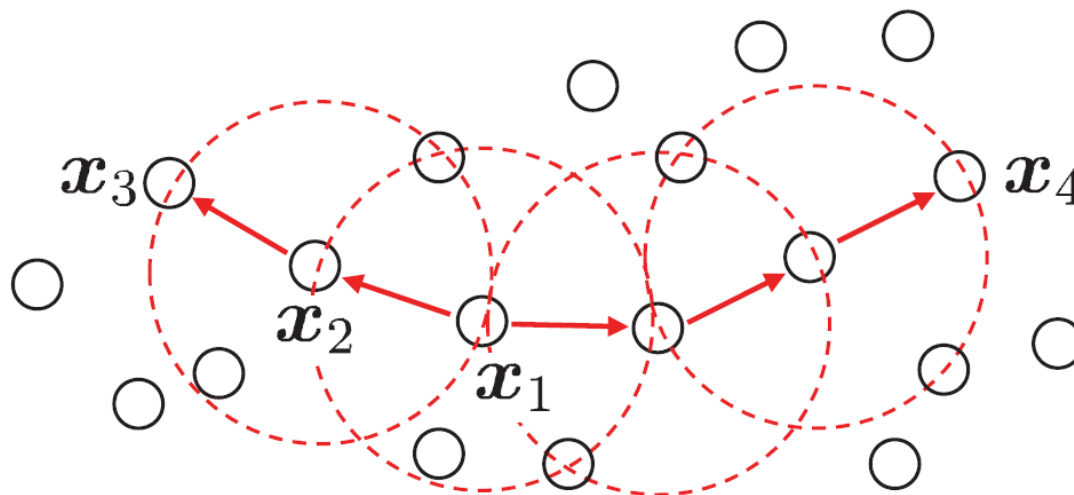
虚线显示出 ϵ 领域。

x_1 是核心对象。

x_2 由 x_1 密度直达。

x_3 由 x_1 密度可达。

x_3 与 x_4 密度相连。



□ 对“簇”的定义

由密度可达关系导出的最大密度相连样本集合。

□ 对“簇”的形式化描述

给定领域参数，簇是满足以下性质的非空样本子集：

连接性： $x_i \in C, x_j \in C \Rightarrow x_i$ 与 x_j 密度相连

最大性： $x_i \in C, x_i$ 与 x_j 密度可达 $\Rightarrow x_j \in C$

实际上，若 x 为核心对象，由 x 密度可达的所有样本组成的集合记为 $X = \{x' \in D \mid x' \text{ 由 } x \text{ 密度可达}\}$ ，则 X 为满足连接性与最大性的簇。

输入: 样本集 $D = \{x_1, x_2, \dots, x_m\}$;
邻域参数 $(\epsilon, MinPts)$.

过程:

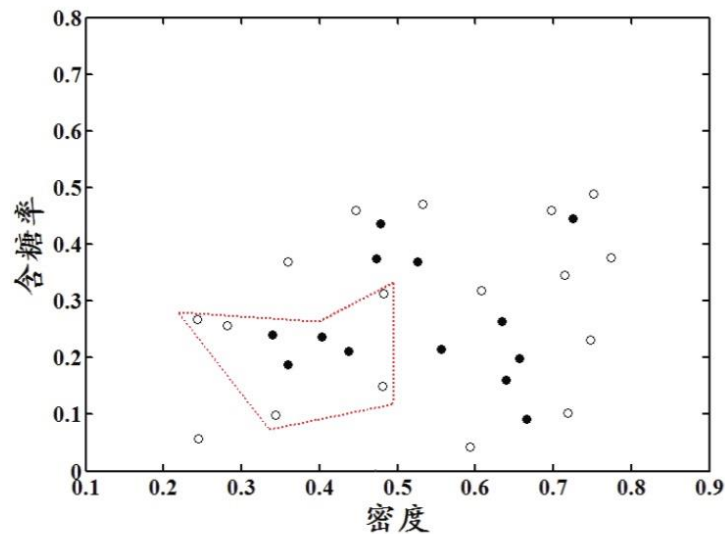
```
1: 初始化核心对象集合:  $\Omega = \emptyset$ 
2: for  $j = 1, \dots, m$  do
3:   确定样本  $x_j$  的  $\epsilon$ -邻域  $N_\epsilon(x_j)$ ;
4:   if  $|N_\epsilon(x_j)| \geq MinPts$  then
5:     将样本  $x_j$  加入核心对象集合:  $\Omega = \Omega \cup \{x_j\}$ 
6:   end if
7: end for

8: 初始化聚类簇数:  $k = 0$ 
9: 初始化未访问样本集合:  $\Gamma = D$ 
10: while  $\Omega \neq \emptyset$  do
11:   记录当前未访问样本集合:  $\Gamma_{old} = \Gamma$ ;
12:   随机选取一个核心对象  $o \in \Omega$ , 初始化队列  $Q = \langle o \rangle$ ;
13:    $\Gamma = \Gamma \setminus \{o\}$ ;
14:   while  $Q \neq \emptyset$  do
15:     取出队列  $Q$  中的首个样本  $q$ ;
16:     if  $|N_\epsilon(q)| \geq MinPts$  then
17:       令  $\Delta = N_\epsilon(q) \cap \Gamma$ ;
18:       将  $\Delta$  中的样本加入队列  $Q$ ;
19:        $\Gamma = \Gamma \setminus \Delta$ ;
20:     end if
21:   end while
22:    $k = k + 1$ , 生成聚类簇  $C_k = \Gamma_{old} \setminus \Gamma$ ;
23:    $\Omega = \Omega \setminus C_k$ 
24: end while
25: return 簇划分结果

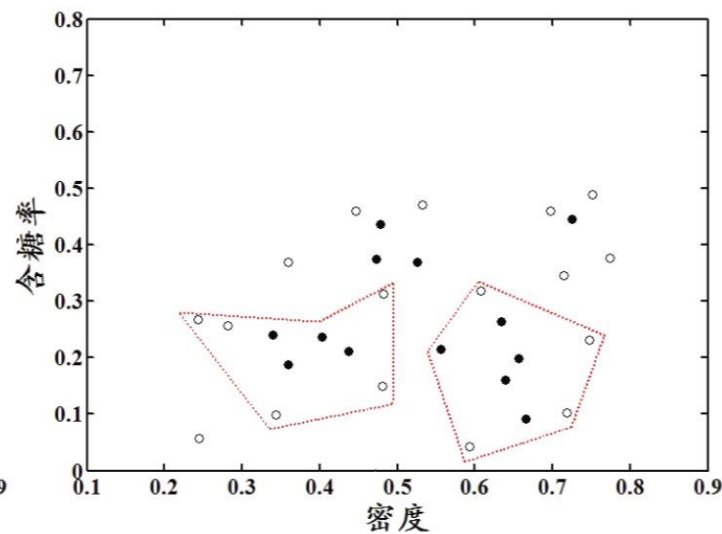
输出: 簇划分  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ 
```

遍历样本找出数据集中的
核心对象集

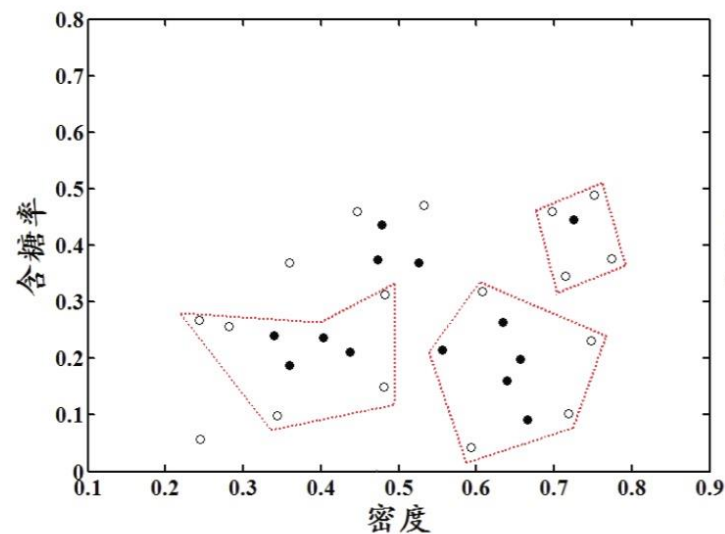
根据密度可达, 每次循环
生成一个簇



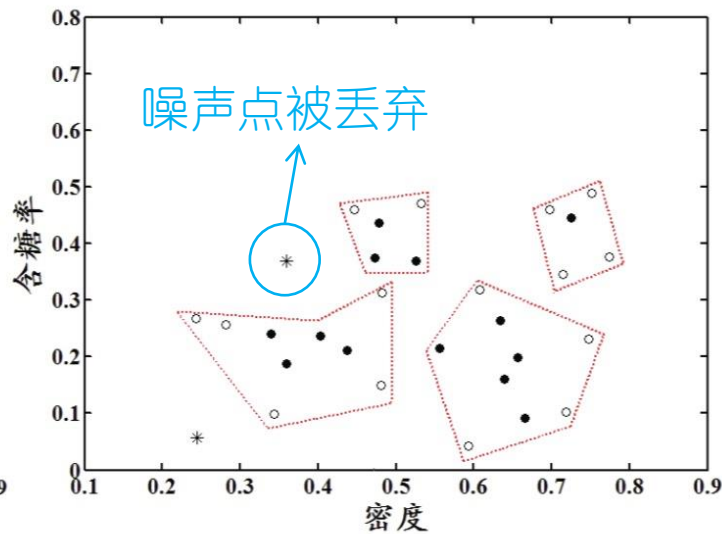
(a) 生成聚类簇 C_1



(b) 生成聚类簇 C_2



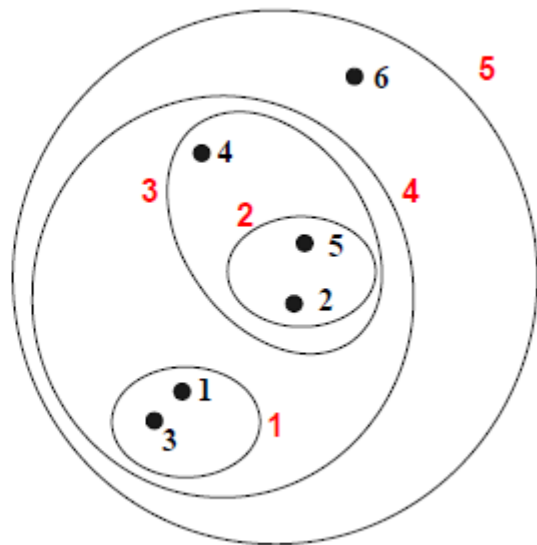
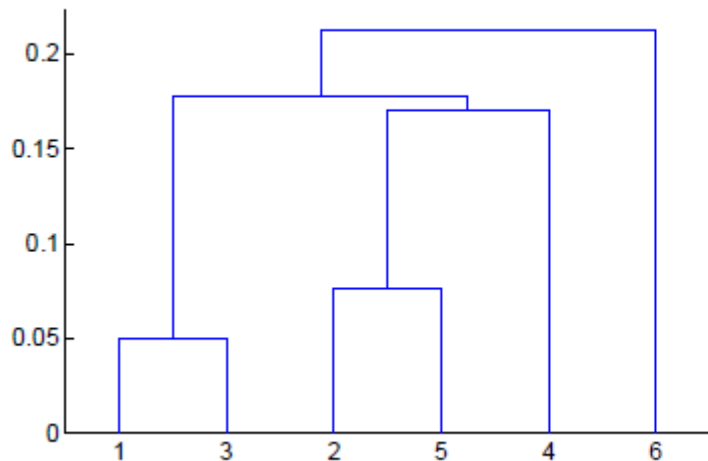
(c) 生成聚类簇 C_3



(d) 生成聚类簇 C_4

层次聚类

- 层次聚类试图在不同层次对数据集进行划分，从而形成**树形**的聚类结构。数据集划分既可采用“**自底向上**”的聚合策略，也可采用“**自顶向下**”的分拆策略。
- 灵活停止
- 不必事先设定簇个数



□ AGNES算法（自底向上的层次聚类算法）

首先，将样本中的每一个样本看做一个初始聚类簇，然后在算法运行的每一步中找出距离最近的两个聚类簇进行合并，该过程不断重复，直到达到预设的聚类簇的个数。

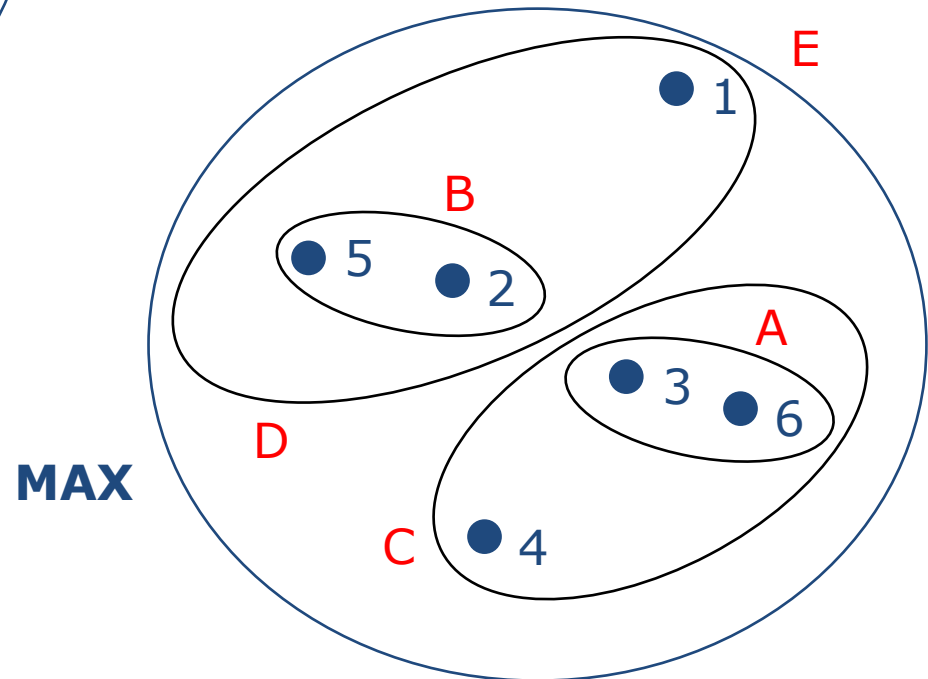
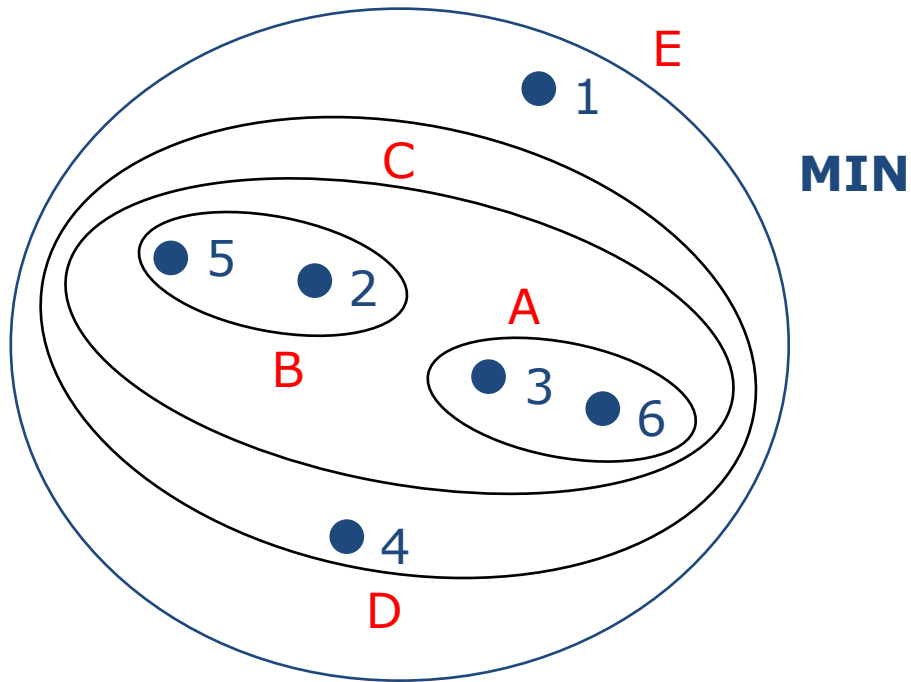
这里两个聚类簇 C_i 和 C_j 的距离，可以有3种度量方式。

最小距离：
$$d_{min}(C_i, C_j) = \min_{x \in C_i, z \in C_j} dist(x, z)$$

最大距离：
$$d_{max}(C_i, C_j) = \max_{x \in C_i, z \in C_j} dist(x, z)$$

平均距离：
$$d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{z \in C_j} dist(x, z)$$

MIN vs. MAX



层次聚类 – AGNES算法



输入: 样本集 $D = \{x_1, x_2, \dots, x_m\}$;
聚类簇距离度量函数 $d \in \{d_{\min}, d_{\max}, d_{\text{avg}}\}$;
聚类簇数 k .

过程:

```
1: for  $j = 1, \dots, m$  do
2:    $C_j = \{x_j\}$ 
3: end for
```

初始化样本，每个样本作为一个初始聚类簇

```
4: for  $i = 1, \dots, m$  do
5:   for  $j = i, \dots, m$  do
6:      $M(i, j) = d(C_i, C_j)$ ;
7:      $M(j, i) = M(i, j)$ 
8:   end for
9: end for
```

初始化簇间距离矩阵

10: 设置当前聚类簇个数: $q = m$

11: while $q > k$ do

```
12: 找出距离最近的两个聚类簇  $(C_{i^*}, C_{j^*})$ ;
13: 合并  $(C_{i^*}, C_{j^*})$ :  $C_{i^*} = C_{i^*} \cup C_{j^*}$ ;
14: for  $j = j^* + 1, \dots, q$  do
15:   将聚类簇  $C_j$  重编号为  $C_{j-1}$ 
16: end for
17: 删除距离矩阵  $M$  的第  $j^*$  行与第  $j^*$  列;
```

簇合并

```
18: for  $j = 1, \dots, q - 1$  do
19:    $M(i^*, j) = d(C_{i^*}, C_j)$ ;
20:    $M(j, i^*) = M(i^*, j)$ 
21: end for
```

更新簇间距离矩阵

22: $q = q - 1$

23: end while

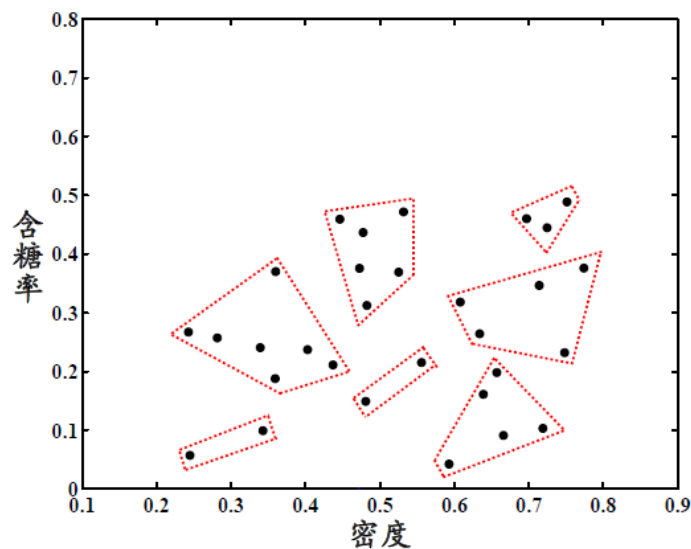
24: return 簇划分结果

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

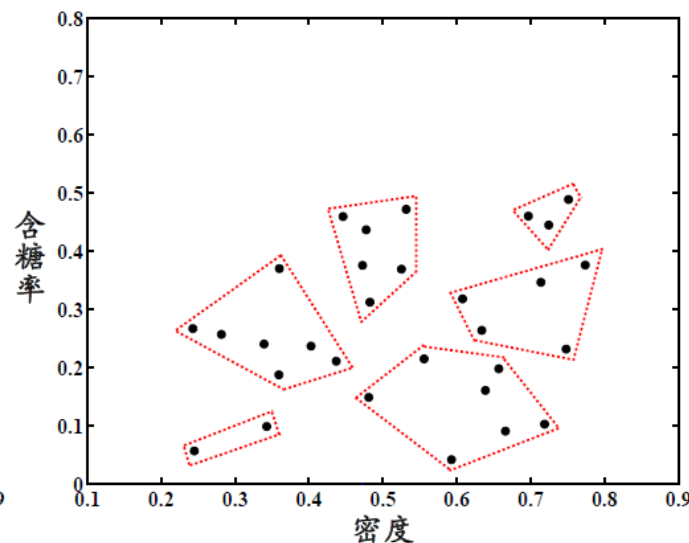
AGNES聚类结果



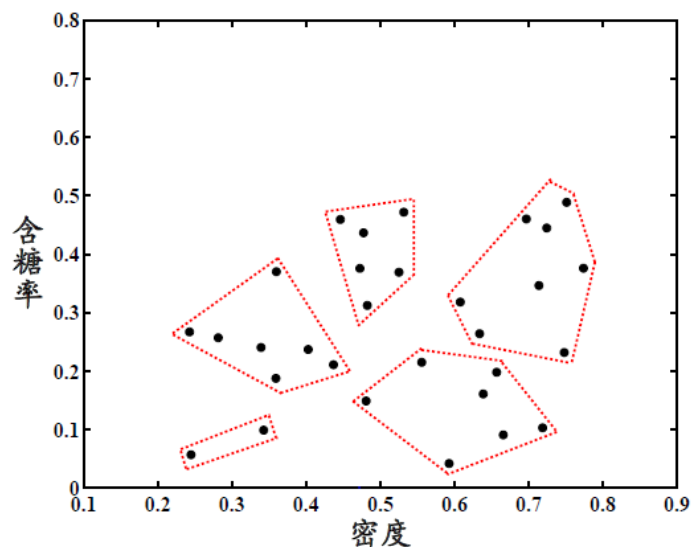
大连理工大学 人工智能学院
School of Artificial Intelligence, Dalian University of Technology



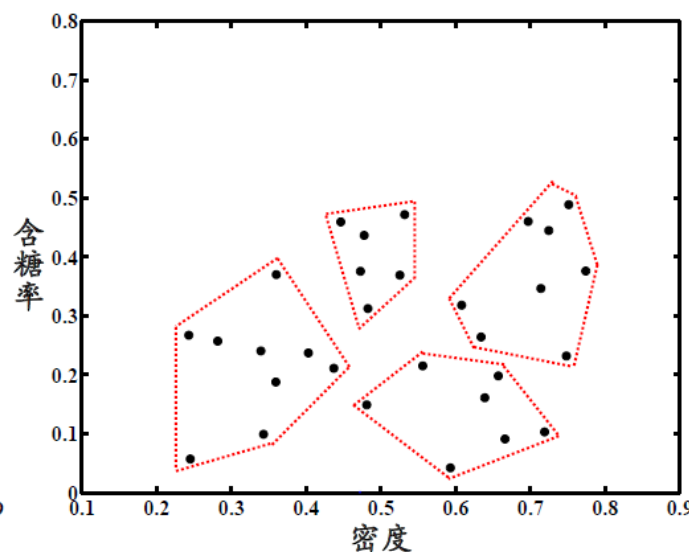
(a) 聚类簇数 $k = 7$



(b) 聚类簇数 $k = 6$



(c) 聚类簇数 $k = 5$



(d) 聚类簇数 $k = 4$

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类：K-means, GMM
- 密度聚类：DBSCAN
- 层次聚类：AGNES