第8章　聚类

刘家锋

哈尔滨工业大学

第8章　聚类

1. 8.1 聚类的概念
2. 8.2 原型聚类
3. 8.3 密度聚类
4. 8.4 层次聚类

8.1 聚类的概念

聚类任务

- 无监督学习
  - 训练样本的标记信息是未知的；
  - 学习的目标是要揭示训练数据的内在性质和规律；
- 聚类任务
  - 聚类是无监督学习中研究最多，应用最广的一类任务；
  - 聚类试图将数据集中的样本划分为若干个不相交的子集，称为簇(cluster)；
  - 每个簇对应一些潜在的概念，而这些概念对聚类算法来说也是未知的，是在聚类过程中自动形成的；

聚类任务

- **形式化描述**
  - 给定样本集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$，均为无标记样本；
  - 将D划分为k个不相交的簇$\{C_l | l = 1, \cdots, k\}$，其中：

$$C_{l'} \bigcap\nolimits_{l' \neq l} C_l = \varnothing, \qquad D = \bigcup\nolimits_{l=1}^{k} C_l$$

  - 聚类结果可以表示为：$\lambda_1, \cdots, \lambda_m$；
  - $\lambda_j \in \{1, \cdots, k\}$，表示样本$\mathbf{x}_j$的簇标记；

性能度量

- **如何评价聚类结果的优劣？**
  - 外部指标：将聚类结果与参考模型(答案)比较，包括Jaccard系数，FM指数，Rand指数等；
  - 内部指标：直接考察聚类结果，不需要参考模型，包括如DB指数，Dunn指数等；
- **聚类的原则**
  - “物以类聚”，同一簇的样本尽可能相似，不同簇的样本尽可能不同；
  - “簇内相似度”越大越好，“簇间相似度”越小越好；

距离度量

- **距离度量**
  - 满足下列性质的函数$dist(\cdot,\cdot)$称为距离：

$$
\begin{aligned}
&\text{非负性：} dist(\mathbf{x}_i,\mathbf{x}_j) \geq 0\\
&\text{同一性：} dist(\mathbf{x}_i,\mathbf{x}_j) = 0, \text{当且仅当} \mathbf{x}_i = \mathbf{x}_j\\
&\text{对称性：} dist(\mathbf{x}_i,\mathbf{x}_j) = dist(\mathbf{x}_j,\mathbf{x}_i)\\
&\text{直递性：} dist(\mathbf{x}_i,\mathbf{x}_j) \leq dist(\mathbf{x}_i,\mathbf{x}_k) + dist(\mathbf{x}_i,\mathbf{x}_k)
\end{aligned}
$$

- **距离与相似度**
  - 距离可以用来度量样本之间的相似度；
  - 距离越小，相似度越大；距离越大，相似度越小；

连续属性距离度量

给定样本有d个连续属性，$\mathbf{x}_i = (x_{i1}, \cdots, x_{id})^t$，$\mathbf{x}_j = (x_{j1}, \cdots, x_{jd})^t$

闵可夫斯基距离(Minkowski distance)

$$dist_{mk}(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{u=1}^{d} |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}}$$

欧几里得距离(Euclidean distance)，$p = 2$

$$dist_{od}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{u=1}^{d} (x_{iu} - x_{ju})^2}$$

曼哈顿距离(Manhattan distance)，$p = 1$

$$dist_{man}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{u=1}^{d} |x_{iu} - x_{ju}|$$

8.2 原型聚类

原型聚类

- **基于原型的聚类**
  - 假设：聚类结构能够通过一组原型刻画；
  - 首先随机地初始化一组原型；
  - 然后根据原型划分样本集；
  - 根据样本集的划分，更新原型；
  - 迭代更新原型和划分样本集的过程，直到收敛；
  - 代表算法：k均值算法，学习矢量量化，高斯混合聚类；

k均值算法

- **k-means算法的目标**
  - 给定聚类样本集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$;
  - 将样本集划分为k个簇$\mathcal{C} = \{C_1, \cdots, C_k\}$;
  - 聚类的目标是使得，用每个簇的均值$\boldsymbol{\mu}_i$代替簇内样本的平方误差最小：

$$\min_{\mathcal{C}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2$$

其中，每个簇的均值：

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

  - k-means的优化目标是要使得每个簇内样本的相似度最大；

k均值算法

- **k-means算法的过程**
  - 找到优化目标的最优解，是一个NP难问题，需要遍历所有将m个样本到k个簇的划分；
  - k-means采用贪心算法，寻求一个优化目标的次优解；
  - 给定各个簇的均值$\boldsymbol{\mu}_i$，可以很容易地优化平方误差函数：

$$E(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2$$

    只要将$\mathbf{x} \in D$划分到距离最近的$\boldsymbol{\mu}_i$代表的簇，即可；
  - 给定样本集的划分$\mathcal{C}$，也可以得到最优的$\{\boldsymbol{\mu}_i\}$(试证明)：

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum\nolimits_{\mathbf{x} \in C_i} \mathbf{x}$$

k均值算法

Algorithm 1 k均值聚类算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$，聚类数$k$

Output: 簇划分$\mathcal{C} = \{C_1, \cdots, C_k\}$

从D中随机选择k个样本作为$\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$;
repeat
    $C_i = \varnothing, \quad i = 1, \cdots, k$;
    for $j = 1, \cdots, m$ **do**
        标记样本$\mathbf{x}_j$：$\lambda_j = \arg\min_{1 \le i \le k} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|_2$
        划入相应的簇：$C_{\lambda_j} \leftarrow C_{\lambda_j} \cup \{\mathbf{x}_j\}$
    end for
    for $i = 1, \cdots, k$ **do**
        更新均值：$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$
    end for
until 均值向量未改变

例8.1

无标记西瓜数据集

编号	密度	含糖量	编号	密度	含糖量	编号	密度	含糖量
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

例8.1

将“无标记西瓜数据集”聚类为3个簇；

随机选择$\mathbf{x}_6, \mathbf{x}_{12}, \mathbf{x}_{24}$作为初始均值：

$$\boldsymbol{\mu}_1 = (0.403, 0.237)^t, \quad \boldsymbol{\mu}_2 = (0.343, 0.099)^t, \quad \boldsymbol{\mu}_3 = (0.478, 0.437)^t$$

第一轮：
计算$\mathbf{x}_1$与均值之间的距离：

$$\|\mathbf{x}_1 - \boldsymbol{\mu}_1\|_2 = 0.369, \quad \|\mathbf{x}_1 - \boldsymbol{\mu}_2\|_2 = 0.506, \quad \|\mathbf{x}_1 - \boldsymbol{\mu}_3\|_2 = 0.220$$

应该将$\mathbf{x}_1$划入簇C_3；遍历所有样本，可以得到簇划分：

$$C_1 = \{\mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}, \mathbf{x}_{23}\}$$
$$C_2 = \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{16}\}$$
$$C_3 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_{15}, \mathbf{x}_{21}, \mathbf{x}_{22}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{26}, \mathbf{x}_{27}, \mathbf{x}_{28}, \mathbf{x}_{29}, \mathbf{x}_{30}\}$$

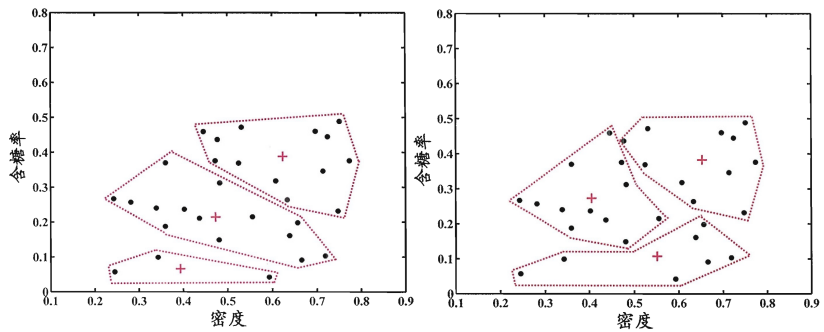
例8.1

重新计算C_1, C_2, C_3的均值向量：

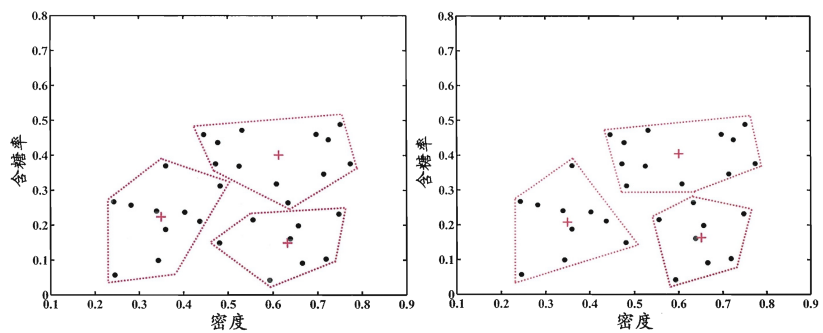
$$\boldsymbol{\mu}_1 = (0.493, 0.207)^t, \quad \boldsymbol{\mu}_2 = (0.394, 0.066)^t, \quad \boldsymbol{\mu}_3 = (0.602, 0.396)^t$$

… …

第五轮: 均值计算结果与第四轮相同，得到最终的簇划分。



(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

例8.2

将下列样本聚类为2个簇：

编号	属性1	属性2	编号	属性1	属性2
$\mathbf{x}_1$	0	0	$\mathbf{x}_{11}$	8	6
$\mathbf{x}_2$	1	0	$\mathbf{x}_{12}$	7	7
$\mathbf{x}_3$	0	1	$\mathbf{x}_{13}$	8	7
$\mathbf{x}_4$	1	1	$\mathbf{x}_{14}$	9	7
$\mathbf{x}_5$	2	1	$\mathbf{x}_{15}$	7	8
$\mathbf{x}_6$	1	2	$\mathbf{x}_{16}$	8	8
$\mathbf{x}_7$	2	2	$\mathbf{x}_{17}$	9	8
$\mathbf{x}_8$	3	2	$\mathbf{x}_{18}$	8	9
$\mathbf{x}_9$	6	6	$\mathbf{x}_{19}$	9	9
$\mathbf{x}_{10}$	7	6			

k均值算法

- **k均值算法的特点**
  - k均值算法的收敛性是由保证的；
  - k均值是一种贪心算法，得到的是优化目标的次优解；
  - 初始均值的选择对聚类结果的影响很大；
  - 算法初始化时，也可以先随机地将样本集划分为k个簇，然后计算初始均值；
  - 实际使用中，一般需要多次初始化，在多个聚类结果中选择最优的；

高斯混合聚类

- **k均值与GMM**
  - k均值聚类可以看作是包含k个高斯的特殊GMM学习过程；
    - 假设高斯的组合系数相同：$\alpha_1 = \cdots = \alpha_k = 1/k$
    - 假设每个高斯的协方差矩阵均为单位阵：$\Sigma_1 = \cdots = \Sigma_k = I$
    - 需要估计每个高斯的均值向量：$\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$
  - 每个簇的样本服从特殊GMM中的某个高斯分布；
- **高斯混合聚类**
  - 如果假设每个簇的样本服从一般的高斯分布，并且每个簇的先验概率是不同的；
  - 可以用数据集D来估计GMM的参数；
  - 然后依据每个样本由各个高斯产生的概率来划分簇；

高斯混合聚类

Algorithm 2 高斯混合聚类算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$，聚类数$k$

Output: 簇划分$\mathcal{C} = \{C_1, \cdots, C_k\}$

1: 用数据集D及EM算法学习GMM的参数：$\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i)\}_{i=1,\cdots,k}$

2: $C_i = \varnothing, \quad i = 1, \cdots, k$;

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

4: 计算样本$\mathbf{x}_j$由各个高斯生成的后验概率：

$$\gamma_{ji} = P(z_j = i|\mathbf{x}_j) = \frac{\alpha_i \cdot p(\mathbf{x}_j|\boldsymbol{\mu}_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(\mathbf{x}_j|\boldsymbol{\mu}_l, \Sigma_l)}$$

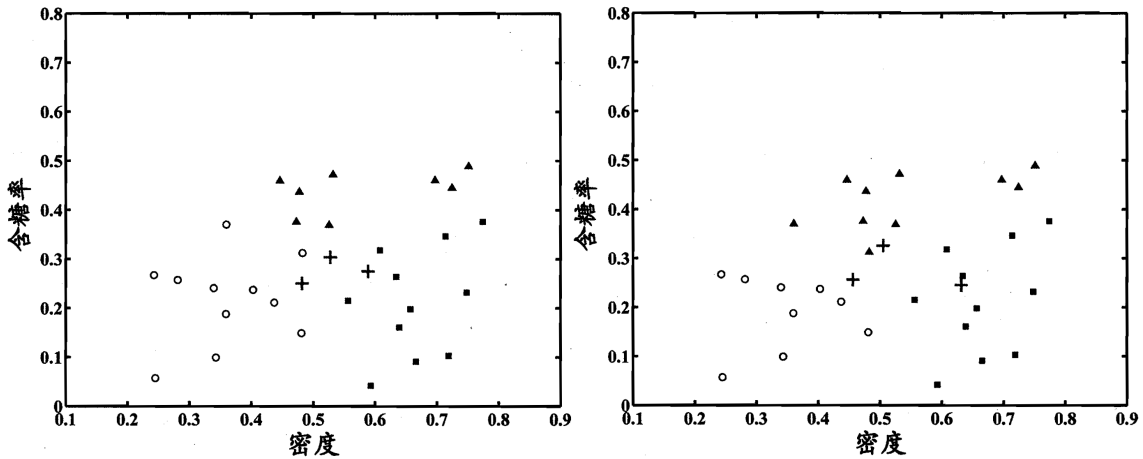
5: 标记$\mathbf{x}_j$，并划入相应的簇：

$$\lambda_j = \arg\max_{1 \le i \le k} \gamma_{ji}, \quad C_{\lambda_j} \leftarrow C_{\lambda_j} \cup \{\mathbf{x}_j\}$$

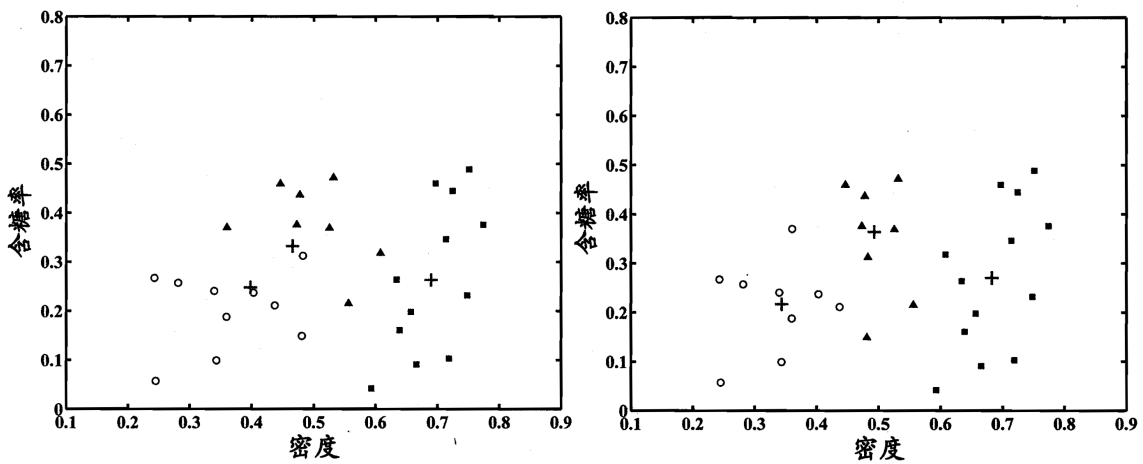
6: **end for**

例8.3

采用高斯混合聚类，将无标记西瓜数据集划分为3个簇：



(a) 5 轮迭代后　(b) 10 轮迭代后

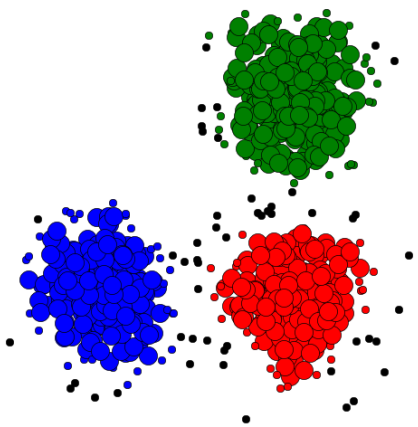


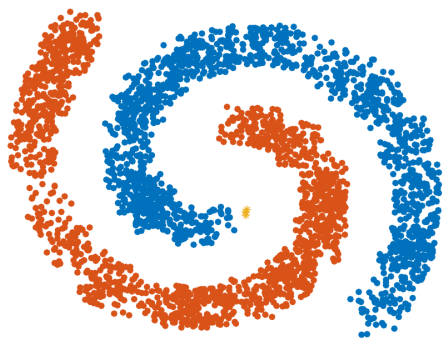
(c) 20 轮迭代后　(d) 50 轮迭代后

8.3 密度聚类

密度聚类

- **基于密度的聚类**
  - 假设：聚类结构能通过样本分布的紧密程度确定；
  - 从样本密度的角度来考察样本之间的可连接性，基于可连接样本不断扩展聚类簇；
  - 代表算法：DBSCAN, OPTICS, DENCLUE





定义名词

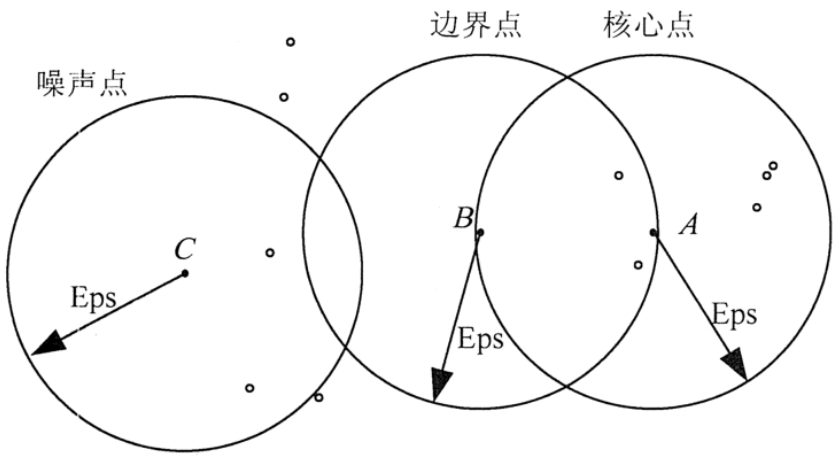
- **点的密度**
  - 给定数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$，算法参数$(\epsilon, MinPts)$；
  - ϵ-邻域：与$\mathbf{x}_j \in D$距离小于ϵ的样本集合

$$N_\epsilon(\mathbf{x}_j) = \{\mathbf{x}_i \in D | dist(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon\}$$

  - 核心点：如果$|N_\epsilon(\mathbf{x}_j)| \geq MinPts$，则$\mathbf{x}_j$是核心点；
  - 边界点：如果$\mathbf{x}_i$不是核心点，但在某个核心点的ϵ-邻域内，则$\mathbf{x}_i$为边界点；
  - 噪声点：既不是核心点，又不是边界点的样本是噪声点；

定义名词

- **点的密度**
  - 核心点对应高密度区域的样本；边界点对应高密度区域边缘的样本；噪声点对应高密度区域之外的样本；
  - 例如：$MinPts = 5$，下图中A是核心点，B是边界点，C是噪声点；

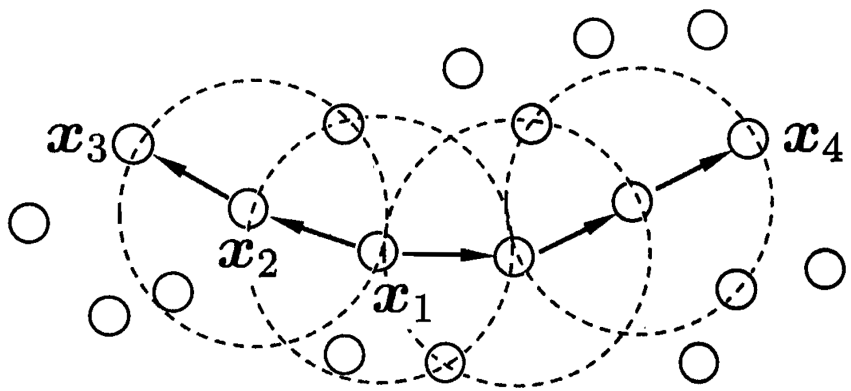


定义名词

- **密度直达**
  - 若$\mathbf{x}_j$位于核心点$\mathbf{x}_i$的ϵ-邻域内，则称$\mathbf{x}_j$由$\mathbf{x}_i$密度直达；
  - 核心点ϵ-邻域内的点可以由核心点密度直达，包括边界点和邻域内的其它核心点；
  - 噪声点不可能有核心点密度直达；
- **密度可达**
  - 如果存在样本序列$\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n$，其中$\mathbf{p}_1 = \mathbf{x}_i, \mathbf{p}_n = \mathbf{x}_j$，并且$\mathbf{p}_{i+1}$由$\mathbf{p}_i$密度直达，则称$\mathbf{x}_j$由$\mathbf{x}_i$密度可达；
  - $\mathbf{x}_j$可以是边界点或核心点，其它的都是核心点；

定义名词

- **密度相连**
  - 若存在$\mathbf{x}_k$使得$\mathbf{x}_i$和$\mathbf{x}_j$均由$\mathbf{x}_k$密度可达，称$\mathbf{x}_i$与$\mathbf{x}_j$密度相连；
  - $\mathbf{x}_k$必须是核心点，$\mathbf{x}_i$和$\mathbf{x}_j$可以是核心点，也可以是边界点；
  - 例如下图中$MinPts = 3$，$\mathbf{x}_1$是核心点，$\mathbf{x}_1$可密度直达$\mathbf{x}_2$，密度可达$\mathbf{x}_3$；
  - $\mathbf{x}_3$和$\mathbf{x}_4$之间是密度相连的；



DBSCAN算法

- **簇的定义**
  - 簇是由密度可达关系导出的最大的密度相连样本集，簇$C \subseteq D$满足：
    a. 连接性：$\mathbf{x}_i \in C, \mathbf{x}_j \in C \quad \Rightarrow \quad \mathbf{x}_i$与$\mathbf{x}_j$密度相连
    b. 最大性：$\mathbf{x}_i \in C, \mathbf{x}_j$由$\mathbf{x}_i$密度可达 $\quad \Rightarrow \quad \mathbf{x}_j \in C$
- **聚类过程**
  1. 从D中任意选择一个数据对象点$\mathbf{p}$；
  2. 如果$\mathbf{p}$是核心点，则找出所有从$\mathbf{p}$密度可达的数据对象点，形成一个新的簇；
  3. 如果$\mathbf{p}$不是核心点，则处理下一个数据对象；
  4. 重复，直到所数据都被处理；剩余的数据对象为噪声点；

DBSCAN算法

Algorithm 3 DBSCAN算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$，参数$(\epsilon, MinPts)$

Output: 簇划分$\mathcal{C} = \{C_1, \cdots, C_k\}$

根据参数$(\epsilon, MinPts)$形成数据集D的核心点集合Ω;
初始化聚类数$k = 1$，未访问数据集$\Gamma = D$;
while $\Omega \neq \varnothing$ **do**
    随机选择$\mathbf{o} \in \Omega$, $C_k = \varnothing$，队列$Q =< \mathbf{o} >$;
    while $Q \neq \varnothing$ **do**
        取出Q中的首个样本$\mathbf{q}$，$C_k \leftarrow C_k \cup (N_\epsilon(\mathbf{q}) \cap \Gamma)$;
        将$\Omega \cap N_\epsilon(\mathbf{q})$添加到队列$Q$的尾部，$\Omega \leftarrow \Omega \setminus N_\epsilon(\mathbf{q})$
    end while
    $\Gamma \leftarrow \Gamma \setminus C_k$, $k \leftarrow k + 1$;
end while

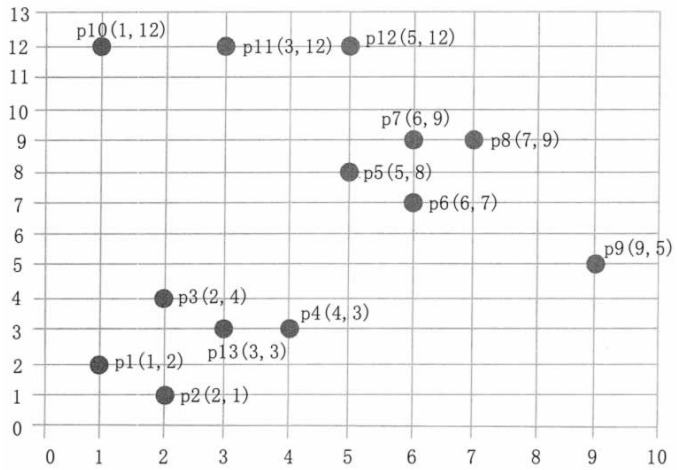
DBSCAN算法

- 优点
  - 可以对任意形状的稠密数据集进行聚类；
  - 无需设定聚类数k；
  - 对数据集中的异常点不敏感，也可以用于发现异常点；
  - 聚类结果稳定，受初始值影响小；
- 缺点
  1. 受参数$(\epsilon, MinPts)$的影响大，特别是当数据集中存在不同密度的簇时，很难选择一组适合的参数；
  2. 数据集大时，样本的ϵ-近邻计算量大；

例8.4

使用DBSCAN算法聚类下列数据：$(\epsilon = 3, MinPts = 3)$

编号	属性1	属性2	编号	属性1	属性2
$\mathbf{x}_1$	1	2	$\mathbf{x}_8$	7	9
$\mathbf{x}_2$	2	1	$\mathbf{x}_9$	9	5
$\mathbf{x}_3$	2	4	$\mathbf{x}_{10}$	1	12
$\mathbf{x}_4$	4	3	$\mathbf{x}_{11}$	3	12
$\mathbf{x}_5$	5	8	$\mathbf{x}_{12}$	5	12
$\mathbf{x}_6$	6	7	$\mathbf{x}_{13}$	3	3
$\mathbf{x}_7$	6	9			



例8.4

Step 1: 扫描$\mathbf{x}_1$

计算$\mathbf{x}_1$的ϵ-邻域:

$$N_\epsilon(\mathbf{x}_1) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_{13}\} \quad |N_\epsilon(\mathbf{x}_1)| = 4 > MinPts$$

$\mathbf{x}_1$是核心点，建立一个新簇:

$$C_1 = N_\epsilon(\mathbf{x}_1) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_{13}\}$$

簇C_1中$\mathbf{x}_2$是核心点，$N_\epsilon(\mathbf{x}_2) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{13}\}$，$\mathbf{x}_4$可达，加入$C_1$:

$$C_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{13}\}$$

簇C_1中$\mathbf{x}_3$是核心点，$N_\epsilon(\mathbf{x}_3) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{13}\}$，已在$C_1$中;
簇C_1中$\mathbf{x}_{13}$是核心点，$N_\epsilon(\mathbf{x}_{13}) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{13}\}$，已在$C_1$中;
簇C_1中$\mathbf{x}_4$是核心点，$N_\epsilon(\mathbf{x}_4) = \{\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{13}\}$，已在$C_1$中;

得到簇$C_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{13}\}$

例8.4

Step 2: 扫描$\mathbf{x}_5$

计算$\mathbf{x}_5$的ϵ-邻域:

$$N_\epsilon(\mathbf{x}_5) = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\} \quad |N_\epsilon(\mathbf{x}_5)| = 4 > MinPts$$

$\mathbf{x}_5$是核心点，建立一个新簇:

$$C_2 = N_\epsilon(\mathbf{x}_5) = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\}$$

$\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8$都是核心点，邻域均为$\{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\}$，已在簇$C_2$中；

Step 3: 扫描$\mathbf{x}_9$

计算$\mathbf{x}_9$的ϵ-邻域：$N_\epsilon(\mathbf{x}_9) = \{\mathbf{x}_9\}$，非核心点；

Step 4: 扫描$\mathbf{x}_{10}$

计算$\mathbf{x}_{10}$的ϵ-邻域：$N_\epsilon(\mathbf{x}_{10}) = \{\mathbf{x}_{10}, \mathbf{x}_{11}\}$，非核心点；

例8.4

Step 5: 扫描$\mathbf{x}_{11}$

计算$\mathbf{x}_{11}$的ϵ-邻域:

$$N_\epsilon(\mathbf{x}_{11}) = \{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\} \quad |N_\epsilon(\mathbf{x}_{11})| = 3 = MinPts$$

$\mathbf{x}_{11}$是核心点，建立一个新簇:

$$C_3 = N_\epsilon(\mathbf{x}_{11}) = \{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}$$

$\mathbf{x}_{10}$处理过，$\mathbf{x}_{12}$的邻域$N_\epsilon(\mathbf{x}_{12}) = \{\mathbf{x}_{11}, \mathbf{x}_{12}\}$，非核心点，得到簇$C_3$;

Step 6:

$\mathbf{x}_{12}$和$\mathbf{x}_{13}$均已处理过，算法结束;

输出簇划分:

$$C_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{13}\}, \qquad C_2 = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\},$$
$$C_3 = \{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}, \qquad \mathbf{x}_9\text{是噪声}$$

8.4 层次聚类

层次聚类

- **Hierarchical clustering**
  - 层次聚类在不同层次对数据划分，形成树形的聚类结构；
  - 数据集的划分可以采用“自底向上”的聚合策略，也可以采用“自顶向下”的分拆策略；
- **AGNES(AGglomerative NESting)**
  - AGNES算法采用的是“自底向上”的聚合策略；
  - 初始时，将数据集中的每一个样本作为一个簇；
  - 每一轮迭代，选择距离最近的两个簇合并；
  - 直到达到预设的聚类簇个数为止；

AGNES算法

Algorithm 4 AGNES算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$，簇距离度量函数$d$，聚类簇数$k$

Output: 簇划分$\mathcal{C} = \{C_1, \cdots, C_k\}$

初始化簇：$C_j = \{\mathbf{x}_j\},\ j = 1, \cdots, m$；簇个数：$q = m$；

计算簇距离矩阵：$M(i,j) = M(j,i) = d(C_i, C_j),\ i,j = 1, \cdots, m$

while $q > k$ **do**

    找出距离最近的两个聚类簇C_{i^*}和C_{j^*}；

    合并C_{i^*}和C_{j^*}：$C_{i^*} \leftarrow C_{i^*} \cup C_{j^*}$

    删除M的第j^*行和列，重编号j^*之后的聚类簇；

    重新计算M的第i^*行和列：

$$M(i^*, j) = M(j, i^*) = d(C_{i^*}, C_j), \quad j = 1, \cdots, q-1$$

    $q \leftarrow q - 1$

end while

簇的距离度量函数

- **最小距离(single-linkage)**
  - 以两个簇中距离最近的两个样本的距离作为簇之间的距离：

$$d_{min}(C_i, C_j) = \min_{\mathbf{x}\in C_i, \mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$$

- **最大距离(complet-linkage)**
  - 以两个簇中距离最远的两个样本的距离作为簇之间的距离：

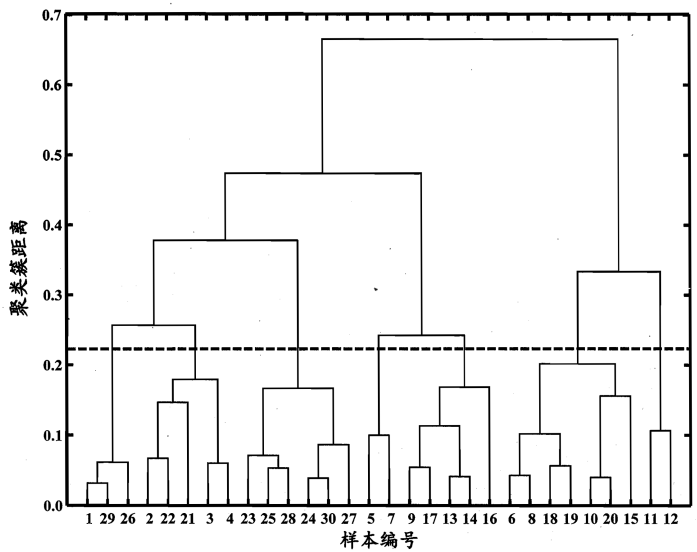
$$d_{min}(C_i, C_j) = \max_{\mathbf{x}\in C_i, \mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$$

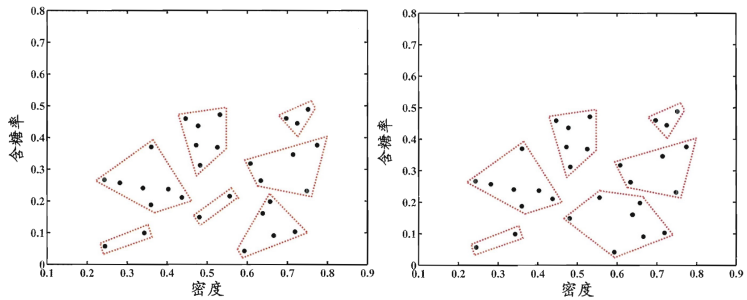
- **平均距离(average-linkage)**
  - 以两个簇之间样本对距离的平均值作为簇之间的距离：

$$d_{min}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x}\in C_i} \sum_{\mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$$

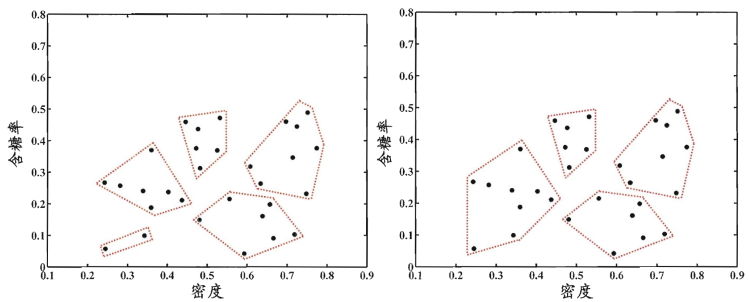
层次聚类

例8.1数据集采用AGNES算法聚类的过程，最大距离度量，聚类数设置$k = 7$;





(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 $k = 4$