

# Clustering 聚类

---

## Clustering 聚类

### 1. 基本概念

- 1.1 聚类任务
- 1.2 性能度量
  - 1.2.1 外部指标
  - 1.2.2 内部指标
- 1.3 距离计算

### 2. 聚类算法

- 2.1 原型聚类
  - k-means 算法
  - 学习向量量化 (LVQ) 算法
  - 高斯混合聚类
- 2.2 密度聚类
  - DBSCAN 算法
- 2.3 层次聚类
  - AGNES 算法

聚类是无监督学习中研究最多、应用最广的方法。

聚类算法可分为

- 1. 原型聚类
  - k-means 算法
  - 学习向量量化 (LVQ) 算法
  - 高斯混合聚类
- 2. 密度聚类
  - DBSCAN 算法
- 3. 层次聚类
  - AGNES 算法

## 1. 基本概念

---

### 1.1 聚类任务

聚类尝试将数据集中的样本划分为若干个通常是不想交的子集，每个子集称为一个“簇”，每个簇有其潜在的概念语义，算法事先不知道。

聚类任务用数学表达：假设样本集  $D = \{x_1, x_2, \dots, x_m\}$  包含  $m$  个无标记样本，每个样本  $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{in})$  是一个  $n$  维特征向量，则聚类算法将样本集  $D$  划分为  $k$  个不相交的簇  $\{C_l | l = 1, 2, \dots, k\}$ ，其中  $C_{l'} \cap_{l' \neq l} C_l = \emptyset$  且  $D = \bigcup_{l=1}^k C_l$ 。相应地，我们用  $\lambda_j \in \{1, 2, \dots, k\}$  表示样本  $x_j$  的簇标记 (cluster label)，即样本属于这个簇： $x_j \in C_{\lambda_j}$ ，于是聚类的结果可用包含  $m$  个元素的簇标记向量  $\boldsymbol{\lambda} = (\lambda_1; \lambda_2; \dots; \lambda_m)$ ：即样本  $x_j$  是属于标记为  $\lambda_j$  簇的。

## 1.2 性能度量

聚类的性能度量称为聚类“有效性指标” (validity index)。

物以类聚，我们想要结果“簇内相似度”高，“簇间相似度低”。

聚类性能度量分两类：

1. 结果与参考模型比较，称为“外部指标”
2. 直接考察结果，不利用参考模型，称为“内部指标”

### 1.2.1 外部指标

数据集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ，簇划分  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ ，外部的参考模型给出的簇为： $\mathcal{C}^* = \{C_1^*, C_2^*, \dots, C_s^*\}$ 。相应地， $\boldsymbol{\lambda}$  和  $\boldsymbol{\lambda}^*$  表示  $\mathcal{C}$  和  $\mathcal{C}^*$  对应的簇标记向量。

将样本两两配对，定义：

$$a = |SS|, \quad SS = \left\{ (\mathbf{x}_i, \mathbf{x}_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j \right\} \quad (1)$$

$$b = |SD|, \quad SD = \left\{ (\mathbf{x}_i, \mathbf{x}_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j \right\} \quad (2)$$

$$c = |DS|, \quad DS = \left\{ (\mathbf{x}_i, \mathbf{x}_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j \right\} \quad (3)$$

$$d = |DD|, \quad DD = \left\{ (\mathbf{x}_i, \mathbf{x}_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j \right\} \quad (4)$$

$a, b, c, d$  四个集合代表四种样本对，比如  $a$  是在  $\mathcal{C}$  和  $\mathcal{C}^*$  中都是隶属相同簇的样本对集合。由于每个样本对  $(x_i, x_j) (i < j)$  只能出现在一个集合中，因此有  $a + b + c + d = m(m - 1)/2$

我们有以下这些常用的聚类性能度量外部指标：

*Jaccard* 系数（简称 *JC*）

$$\text{JC} = \frac{a}{a + b + c} \quad (5)$$

*FM* 指数（简称 *FMI*）

$$\text{FMI} = \sqrt{\frac{a}{a + b} \cdot \frac{a}{a + c}} \quad (6)$$

*Rand* 指数

$$\text{RI} = \frac{2(a + d)}{m(m - 1)} \quad (7)$$

显然，上述性能度量结果在  $[0, 1]$  区间，值越大越好。

## 1.2.2 内部指标

聚类结果  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ , 定义：

$$\text{avg}(C) = \frac{2}{|C|(|C| - 1)} \sum_{1 \leq i < j \leq |C|} \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \quad (8)$$

$$\text{diam}(C) = \max_{1 \leq i < j \leq |C|} \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \quad (9)$$

$$d_{\min}(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \quad (10)$$

$$d_{\text{cen}}(C_i, C_j) = \text{dist}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) \quad (11)$$

$\text{dist}(\cdot, \cdot)$  用于计算两个样本之间的距离；

$\boldsymbol{\mu}$  代表簇  $C$  的中心点  $\boldsymbol{\mu} = \frac{1}{|C|} \sum_{1 \leq i \leq |C|} \mathbf{x}_i$ 。

$\text{avg}(C)$  对应于簇  $C$  内样本间的平均距离

$\text{diam}(C)$  对应于簇  $C$  内样本间的最远距离

$d_{\text{cen}}(C_i, C_j)$  对应于簇  $C_i$  和  $C_j$  中心点之间的距离

下面是常用的聚类性能度量内部指标：

*DB* 指数 (*DBI*)

$$\text{DBI} = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left( \frac{\text{avg}(C_i) + \text{avg}(C_j)}{d_{\text{cen}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)} \right) \quad (12)$$

*Dumm* 指数 (*DI*)

$$DI = \min_{1 \leq i \leq k} \left\{ \min_{j \neq i} \left( \frac{d_{\min}(C_i, C_j)}{\max_{1 \leq l \leq k} \text{diam}(C_l)} \right) \right\} \quad (13)$$

显然  $DBI$  值越小越好,  $DI$  值越大越好。

## 1.3 距离计算

对函数  $\text{dist}(\cdot, \cdot)$ , 若其为一个“距离度量”, 则要满足以下性质:

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

给定样本  $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{in})$  与  $\mathbf{x}_j = (x_{j1}; x_{j2}; \dots; x_{jn})$

闵可夫斯基距离 (Minkowski distance) :

$$\text{dist}_{\text{mk}}(\mathbf{x}_i, \mathbf{x}_j) = \left( \sum_{u=1}^n |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}} \quad (14)$$

即  $L_p$  范数,  $\|\mathbf{x}_i - \mathbf{x}_j\|_p$

$p = 2$  时, 即欧氏距离 (Euclidean distance)

$$\text{dist}_{\text{ed}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{u=1}^n |x_{iu} - x_{ju}|^2} \quad (15)$$

$p = 1$  时, 即曼哈顿距离 (Manhattan distance)

$$\text{dist}_{\text{man}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{u=1}^n |x_{iu} - x_{ju}| \quad (16)$$

属性划分有连续属性和离散属性。对于离散属性, 在讨论距离计算时, 属性上是否定义“序”关系至关重要, 比如像  $\{1, 2, 3\}$  的离散属性与连续属性的性质更接近, 能直接在属性上计算距离: “1”与“2”较近, “1”与“3”较远, 这种属性称为“有序属性”, 而像  $\{\text{飞机}, \text{火车}, \text{轮船}\}$  这样的离散属性不能直接计算距离, 称为“无序属性”。

显然, 闵可夫斯基距离可用于有序属性。

对无序属性可采用 VDM (Value Difference Metric), 令  $m_{u,a}$  表示在属性  $u$  上取值为  $a$  的样本数,  $m_{u,a,i}$  表示第  $i$  个样本簇中在属性  $u$  上取值为  $a$  的样本数,  $k$  为样本簇数, 则属性  $u$  上两个离散值  $a$  和  $b$  之间的 VDM 距离为:

$$\text{VDM}_p(a, b) = \sum_{i=1}^k \left| \frac{m_{u,a,i}}{m_{u,a}} - \frac{m_{u,b,i}}{m_{u,b}} \right|^p \quad (17)$$

于是, 将闵可夫斯基距离和 VDM 结合可处理混合属性, 假定有  $n_c$  个有序属性,  $n - n_c$  个无序属性, 令有序属性排在无序属性前:

$$\text{MinkovDM}_p(\mathbf{x}_i, \mathbf{x}_j) = \left( \sum_{u=1}^{n_c} |x_{iu} - x_{ju}|^p + \sum_{u=n_c+1}^n \text{VDM}_p(x_{iu}, x_{ju}) \right)^{\frac{1}{p}} \quad (18)$$

不同属性重要性不同时，可使用加权距离，以加权闵可夫斯基距离为例：

$$\text{dist}_{\text{wmk}}(\mathbf{x}_i, \mathbf{x}_j) = (w_1 \cdot |x_{i1} - x_{j1}|^p + \dots + w_n \cdot |x_{in} - x_{jn}|^p)^{\frac{1}{p}} \quad (19)$$

权重  $w_i \geq 0 (i = 1, 2, \dots, n)$  表征不同属性的重要性，通常和为 1

需要注意，现在的距离是来定义相似度度量，而不是“距离度量”，因为距离不一定满足所有距离度量的性质，特别是“直递性”，称为“非度量距离”。现实中有必要基于数据样本来确定合适的距离计算式，可通过“距离度量学习”来实现。

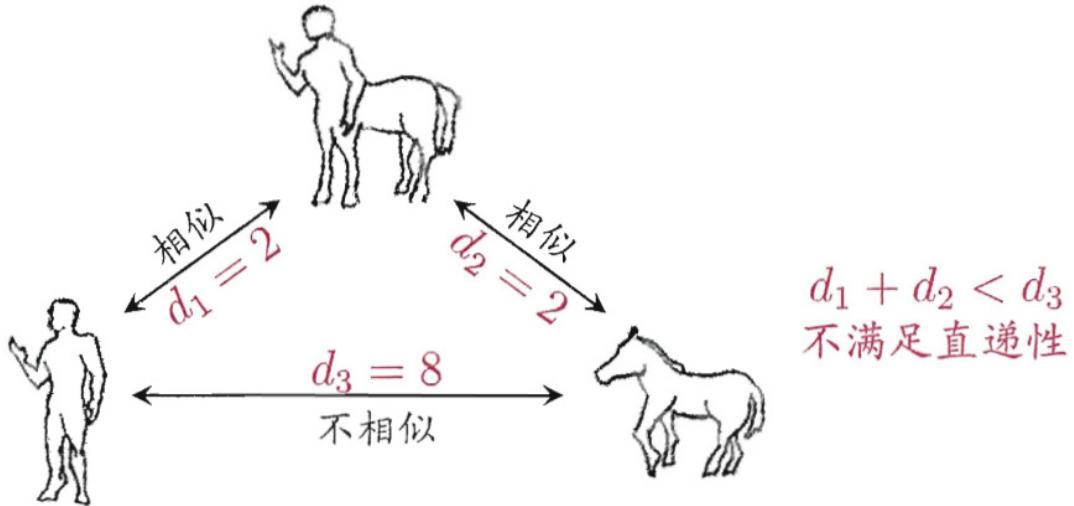


图 9.1 非度量距离的一个例子

## 2. 聚类算法

### 2.1 原型聚类

prototype-based clustering

原型聚类假设聚类结构能通过一组原型刻画，在现实聚类中极为常用。

通常算法先对原型进行初始化，然后对原型进行迭代跟新求解。

#### k-means 算法

样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , 簇划分  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ , k-均值算法最小化平方误差：

$$E = \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2 \quad (20)$$

其中  $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$  是簇  $C_i$  的均值向量。

上式刻画了簇内样本围绕簇均值向量的紧密程度， $E$  值越小，则簇内样本相似度越高。

最小化上式不容易，找到其最优解要考察  $D$  所有可能的簇划分，是一个 NP 难问题。 $k$  均值算法采用了贪心策略，通过迭代优化来近似求解  $E = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|_2^2$

算法如下：

---

输入：样本集  $D = \{x_1, x_2, \dots, x_m\}$ ;

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  $\{\mu_1, \mu_2, \dots, \mu_k\}$

2: repeat

3: 令  $C_i = \emptyset (1 \leq i \leq k)$

4: for  $j = 1, 2, \dots, m$  do

5: 计算样本  $x_j$  与各均值向量  $\mu_i (1 \leq i \leq k)$  的距离:  $d_{ji} = \|x_j - \mu_i\|_2$ ;

6: 根据距离最近的均值向量确定  $x_j$  的簇标记:  $\lambda_j = \arg \min_{i \in \{1, 2, \dots, k\}} d_{ji}$ ;

7: 将样本  $x_j$  划入相应的簇:  $C_{\lambda_j} = C_{\lambda_j} \cup \{x_j\}$ ;

8: end for

9: for  $i = 1, 2, \dots, k$  do

10: 计算新均值向量:  $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ ;

11: if  $\mu'_i \neq \mu_i$  then

12: 将当前均值向量  $\mu_i$  更新为  $\mu'_i$

13: else

14: 保持当前均值向量不变

15: end if

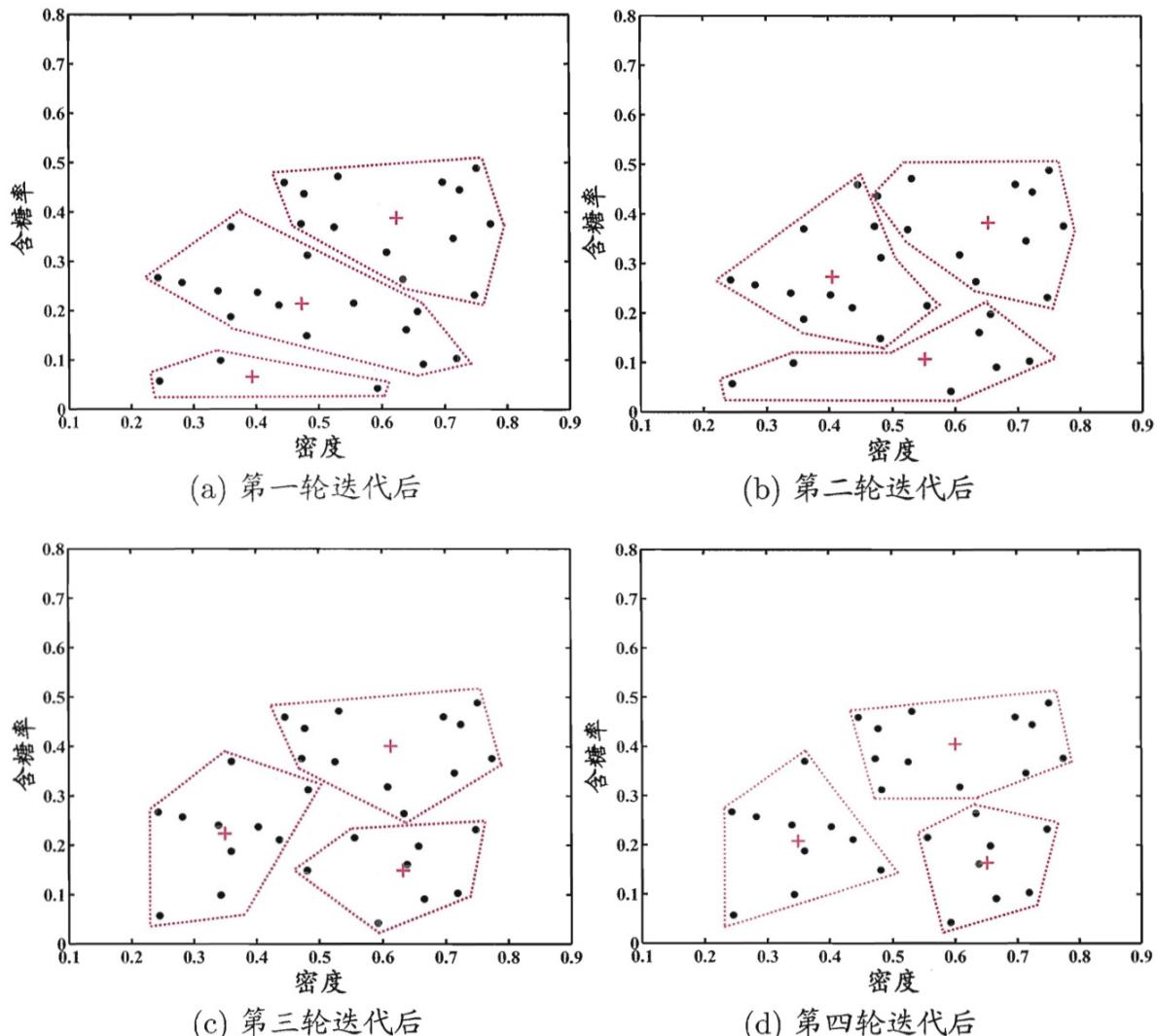
16: end for

17: until 当前均值向量均未更新

---

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

图 9.2  $k$  均值算法



## sklearn API

```

sklearn.cluster.KMeans(n_clusters=8,
    init='k-means++',
    n_init=10,
    max_iter=300,
    tol=0.0001,
    precompute_distances='auto',
    verbose=0,
    random_state=None,
    copy_x=True,
    n_jobs=1,
    algorithm='auto'
)
# n_clusters: 簇的个数, 即你想聚成几类
# init: 初始簇中心的获取方法
# n_init: 获取初始簇中心的更迭次数, 为了弥补初始质心的影响, 算法默认会初10次质心, 实现
# 算法, 然后返回最好的结果。
# max_iter: 最大迭代次数 (因为kmeans算法的实现需要迭代)
# tol: 容忍度, 即kmeans运行准则收敛的条件

```

```

# precompute_distances: 是否需要提前计算距离, 这个参数会在空间和时间之间做权衡, 如果
# 是True 会把整个距离矩阵都放到内存中, auto 会默认在数据样本大于features*samples 的数量
# 大于12e6 的时候False, False 时核心实现的方法是利用Cpython 来实现的
# verbose: 冗长模式
# random_state: 随机生成簇中心的状态条件。
# copy_x: 对是否修改数据的一个标记, 如果True, 即复制了就不会修改数据。bool 在scikit-
# learn 很多接口中都会有这个参数的, 就是是否对输入数据继续copy 操作, 以便不修改用户的输入
# 数据。这个要理解Python 的内存机制才会比较清楚。
# n_jobs: 并行设置
# algorithm: kmeans的实现算法, 有: 'auto', 'full', 'elkan', 其中 'full' 表示用EM
# 方式实现

import numpy as np
from sklearn.cluster import KMeans

data = np.random.rand(100, 3) # 样本大小为100, 特征数为 3

# 构造一个聚类数为3的聚类器
estimator = KMeans(n_clusters=3)#构造聚类器
estimator.fit(data)
label_pred = estimator.labels_ # 标签
centroids = estimator.cluster_centers_ # 聚类中心
inertia = estimator.inertia_ # 每个点到中心的距离平方和

```

## 学习向量量化 (LVQ) 算法

学习向量量化 (Learning Vector Quantization) 和 k-means 类似, 也是试图找一组原型向量来刻画聚类结构, 但是不同的是, LVQ 假设数据带有类别标记, 学习过程利用样本的监督信息来辅助聚类。

样本集带有辅助标记:  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ , LVQ 的目标是学得一组  $n$  维原型  $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$ , 每个原型向量代表一个聚类簇, 簇标记  $t_i \in \mathcal{Y}$

---

**输入:** 样本集  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ;  
 原型向量个数  $q$ , 各原型向量预设的类别标记  $\{t_1, t_2, \dots, t_q\}$ ;  
 学习率  $\eta \in (0, 1)$ .

**过程:**

- 1: 初始化一组原型向量  $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$
- 2: **repeat**
- 3:   从样本集  $D$  随机选取样本  $(\mathbf{x}_j, y_j)$ ;
- 4:   计算样本  $\mathbf{x}_j$  与  $\mathbf{p}_i$  ( $1 \leq i \leq q$ ) 的距离:  $d_{ji} = \|\mathbf{x}_j - \mathbf{p}_i\|_2$ ;
- 5:   找出与  $\mathbf{x}_j$  距离最近的原型向量  $p_{i^*}$ ,  $i^* = \arg \min_{i \in \{1, 2, \dots, q\}} d_{ji}$ ;
- 6:   **if**  $y_j = t_{i^*}$  **then**
- 7:      $\mathbf{p}' = \mathbf{p}_{i^*} + \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*})$
- 8:   **else**
- 9:      $\mathbf{p}' = \mathbf{p}_{i^*} - \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*})$
- 10:   **end if**
- 11:   将原型向量  $\mathbf{p}_{i^*}$  更新为  $\mathbf{p}'$
- 12: **until** 满足停止条件

**输出:** 原型向量  $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$

---

图 9.4 学习向量量化算法

解释一下:

第一行原型向量初始化, 如对第  $q$  个簇可以从类别标记为  $t_q$  的样本中随机选取一个作为原型向量。

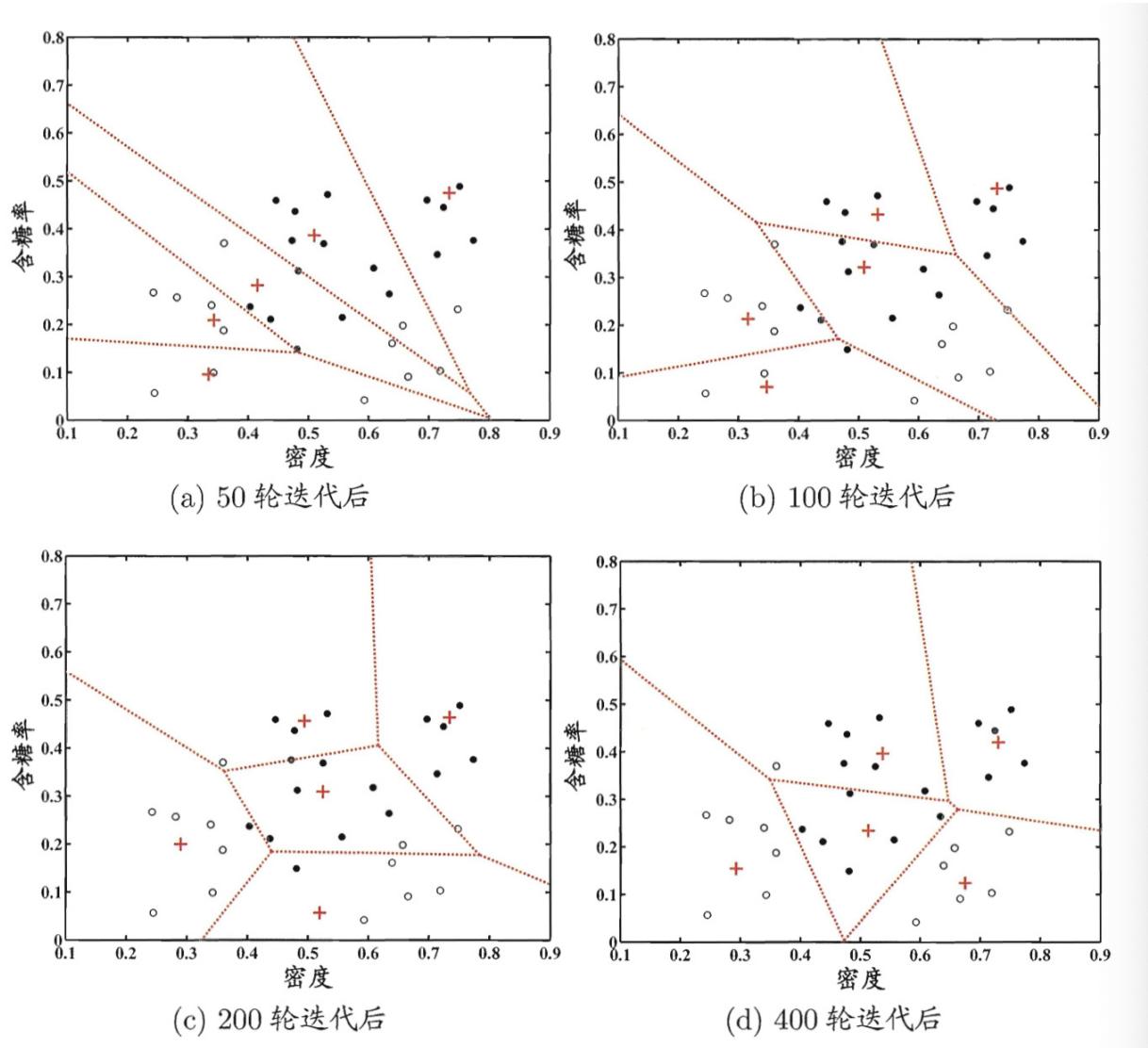
每一次迭代, 算法随机选择样本  $(\mathbf{x}_j, y_j)$ , 计算与它最近的原型向量  $p_{i^*}$ , 再观察  $y_i$  和  $p_{i^*}$  的类别是否一致, 如果一致, 则令  $p_{i^*}$  向  $x_j$  反方向靠拢

$$\mathbf{p}' = \mathbf{p}_{i^*} + \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*}) \quad (21)$$

不一致, 向相反方向走远:

$$\mathbf{p}' = \mathbf{p}_{i^*} - \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*}) \quad (22)$$

在学习一组原型向量  $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$  后, 即可实现对样本空间  $\mathcal{X}$  的粗划分, 每个样本被划入与其距离最近的原型向量所代表的簇中



## 高斯混合聚类

与 k 均值, LVQ 用原型向量来刻画聚类结构不同, 高斯混合 (Mixture-of-Gaussian) 聚类采用概率模型来表达聚类原型。

多元高斯分布:

$n$  维向量  $x$  若服从高斯分布, 则其概率密度为

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})} \quad (23)$$

$\boldsymbol{\mu}$  是  $n$  维均值向量,  $\Sigma$  是  $n \times n$  协方差矩阵

概率密度函数可定记为  $p(\mathbf{x}|\boldsymbol{\mu}, \Sigma)$ , 可定义高斯混合分布

$$p_{\mathcal{M}}(\mathbf{x}) = \sum_{i=1}^k \alpha_i \cdot p(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i) \quad (24)$$

该分布由  $k$  个混合分布组成，每个混合成分对应一个高斯分布， $\alpha_i > 0$  为相应的混合系数， $\sum_{i=1}^k \alpha_i = 1$ 。

假设样本的生成过程由高斯混合分布给出：首先，根据  $\alpha_1, \alpha_2, \dots, \alpha_k$  定义的先验分布，选择高斯混合成分，其中  $\alpha_i$  为选择第  $i$  个混合成分的概率；然后根据被选择的混合成分的概率密度函数进行采样，从而生成相应的样本。

若训练集  $D = \{x_1, x_2, \dots, x_m\}$  由上述过程生成，令随机变量  $z_j \in 1, 2, \dots, k$  表示生成样本  $x_j$  的高斯混合成分，其取值未知，显然  $z_j$  的先验概率  $P(z_j = i)$  对应于  $\alpha_i (i = 1, 2, \dots, k)$ ，根据贝叶斯定理， $z_j$  的后验分布对应于

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

$p_M(z_j = i | \mathbf{x}_j)$  给出了：样本  $x_j$  由第  $i$  个高斯混合成分生成的的后验概率，简记为  $\gamma_{ji} (i = 1, 2, \dots, k)$

当高斯混合分布  $p_M(\mathbf{x}) = \sum_{i=1}^k \alpha_i \cdot p(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$  已知时，划分为  $k$  个簇  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ ，每个样本  $x_j$  的簇标记  $\lambda_j$  如下确定：

$$\lambda_j = \arg \max_{i \in \{1, 2, \dots, k\}} \gamma_{ji} \quad (25)$$

从原型聚类的角度看，高斯混合聚类采用概率模型（高斯分布）对原型进行刻画，簇划分则由原型对应的后验概率确定。

$p_M(\mathbf{x}) = \sum_{i=1}^k \alpha_i \cdot p(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$  已知时，划分为  $k$  个簇  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$  的模型参数  $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) | 1 \leq i \leq k\}$  如何求解？对样本集  $D$ ，可采用极大似然估计，集最大化（对数）似然

$$\begin{aligned} LL(D) &= \ln \left( \prod_{j=1}^m p_M(\mathbf{x}_j) \right) \\ &= \sum_{j=1}^m \ln \left( \sum_{i=1}^k \alpha_i \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right) \end{aligned}$$

常采用 EM 算法进行迭代优化求解：

若参数  $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) | 1 \leq i \leq k\}$  能使得上式最大化，则由  $\frac{\partial LL(D)}{\partial \boldsymbol{\mu}_i} = 0$  有

$$\sum_{j=1}^m \frac{\alpha_i \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^k \alpha_l \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} (\mathbf{x}_j - \boldsymbol{\mu}_i) = 0 \quad (26)$$

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}} \quad (27)$$

即各混合成分的均值可通过样本加权平均来估计，样本权重是每个样本属于该成分的后验概率，类似由  $\frac{\partial LL(D)}{\partial \boldsymbol{\Sigma}_i} = 0$  得：

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \boldsymbol{\mu}_i) (\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^m \gamma_{ji}} \quad (28)$$

对于混合系数  $\alpha_i$ , 除了要最大化  $LL(D)$ , 还需满足  $\alpha_i \geq 0, \sum_{i=1}^k \alpha_i = 1$ , 考虑  $LL(D)$  的拉格朗日形式

$$LL(D) + \lambda \left( \sum_{i=1}^k \alpha_i - 1 \right) \quad (29)$$

其中  $\lambda$  为拉格朗日乘子, 对  $\alpha_i$  的导数为 0, 有

$$\sum_{j=1}^m \frac{p(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^k \alpha_l \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} + \lambda = 0 \quad (30)$$

两边同乘以  $\alpha_i$ , 对所有样本求和可知  $\lambda = -m$ , 有

$$\alpha_i = \frac{1}{m} \sum_{j=1}^m \gamma_{ji} \quad (31)$$

即每个高斯成分的混合系数由样本属于该成分的平均后验概率确定。

由上述推导可获得高斯混合模型的 EM 算法: 在每步迭代中, 先根据当前参数来计算每个样本属于每个高斯分布的后验概率  $\gamma_{ji}$  (E 步), 再根据  $\boldsymbol{\mu}_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}$ ,  $\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \boldsymbol{\mu}_i) (\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^m \gamma_{ji}}$ ,  $\alpha_i = \frac{1}{m} \sum_{j=1}^m \gamma_{ji}$  更新模型参数  $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) | 1 \leq i \leq k\}$  (M 步)

---

**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  $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$
- 2: **repeat**
- 3:   **for**  $j = 1, 2, \dots, m$  **do**
- 4:     根据式(9.30)计算  $\mathbf{x}_j$  由各混合成分生成的后验概率, 即  
$$\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \mathbf{x}_j) \quad (1 \leq i \leq k)$$
- 5:   **end for**
- 6:   **for**  $i = 1, 2, \dots, k$  **do**
- 7:     计算新均值向量:  $\mu'_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}$ ;
- 8:     计算新协方差矩阵:  $\Sigma'_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \mu'_i)(\mathbf{x}_j - \mu'_i)^T}{\sum_{j=1}^m \gamma_{ji}}$ ;
- 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  $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$  更新为  $\{(\alpha'_i, \mu'_i, \Sigma'_i) \mid 1 \leq i \leq k\}$
- 12: **until** 满足停止条件
- 13:  $C_i = \emptyset \quad (1 \leq i \leq k)$
- 14: **for**  $j = 1, 2, \dots, m$  **do**
- 15:   根据式(9.31)确定  $\mathbf{x}_j$  的簇标记  $\lambda_j$ ;
- 16:   将  $\mathbf{x}_j$  划入相应的簇:  $C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$
- 17: **end for**

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

---

图 9.6 高斯混合聚类算法

## 2.2 密度聚类

### DBSCAN 算法

---

输入: 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ;  
邻域参数  $(\epsilon, MinPts)$ .

过程:

- 1: 初始化核心对象集合:  $\Omega = \emptyset$
- 2: **for**  $j = 1, 2, \dots, m$  **do**
- 3:   确定样本  $\mathbf{x}_j$  的  $\epsilon$ -邻域  $N_\epsilon(\mathbf{x}_j)$ ;
- 4:   **if**  $|N_\epsilon(\mathbf{x}_j)| \geq MinPts$  **then**
- 5:     将样本  $\mathbf{x}_j$  加入核心对象集合:  $\Omega = \Omega \cup \{\mathbf{x}_j\}$
- 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  $\mathbf{o} \in \Omega$ , 初始化队列  $Q = < \mathbf{o} >$ ;
- 13:    $\Gamma = \Gamma \setminus \{\mathbf{o}\}$ ;
- 14:   **while**  $Q \neq \emptyset$  **do**
- 15:     取出队列  $Q$  中的首个样本  $\mathbf{q}$ ;
- 16:     **if**  $|N_\epsilon(\mathbf{q})| \geq MinPts$  **then**
- 17:       令  $\Delta = N_\epsilon(\mathbf{q}) \cap \Gamma$ ;
- 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**

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

---

## 2.3 层次聚类

层次聚类 (hierarchical clustering) 在不同层次对数据集进行划分，从而采用树形的聚类结构。

数据集划分可自底向上聚合、或自顶向下分拆

### AGNES 算法

AGNES 采用自底向上聚合策略，先将每个样本看作一个初始聚类簇，然后每一步找出距离最近的两个聚类簇进行合并，不断重复，直到达到预设的聚类簇个数。关键是如何计算聚类簇间的距离。

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

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

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

最小距离由两个簇的最近样本决定，最大距离由两个簇的最远样本决定，平均距离由两个簇的所有样本共同决定。当聚类簇由 $d_{\min}$ 、 $d_{\max}$ 、 $d_{avg}$ 计算时，AGNES 算法相应地被称为“单链接”，“全连接”，“均链接”算法

---

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

过程：

```
1: for  $j = 1, 2, \dots, m$  do
2:    $C_j = \{x_j\}$ 
3: end for
4: for  $i = 1, 2, \dots, m$  do
5:   for  $j = 1, 2, \dots, m$  do
6:      $M(i, j) = d(C_i, C_j)$ ;
7:      $M(j, i) = M(i, j)$ 
8:   end for
9: end for
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  $C_{i^*}$  和  $C_{j^*}$ ;
13:   合并  $C_{i^*}$  和  $C_{j^*}$ :  $C_{i^*} = C_{i^*} \cup C_{j^*}$ ;
14:   for  $j = j^* + 1, j^* + 2, \dots, q$  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;
18:   for  $j = 1, 2, \dots, q - 1$  do
19:      $M(i^*, j) = d(C_{i^*}, C_j)$ ;
20:      $M(j, i^*) = M(i^*, j)$ 
21:   end for
22:    $q = q - 1$ 
23: end while
```

---

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

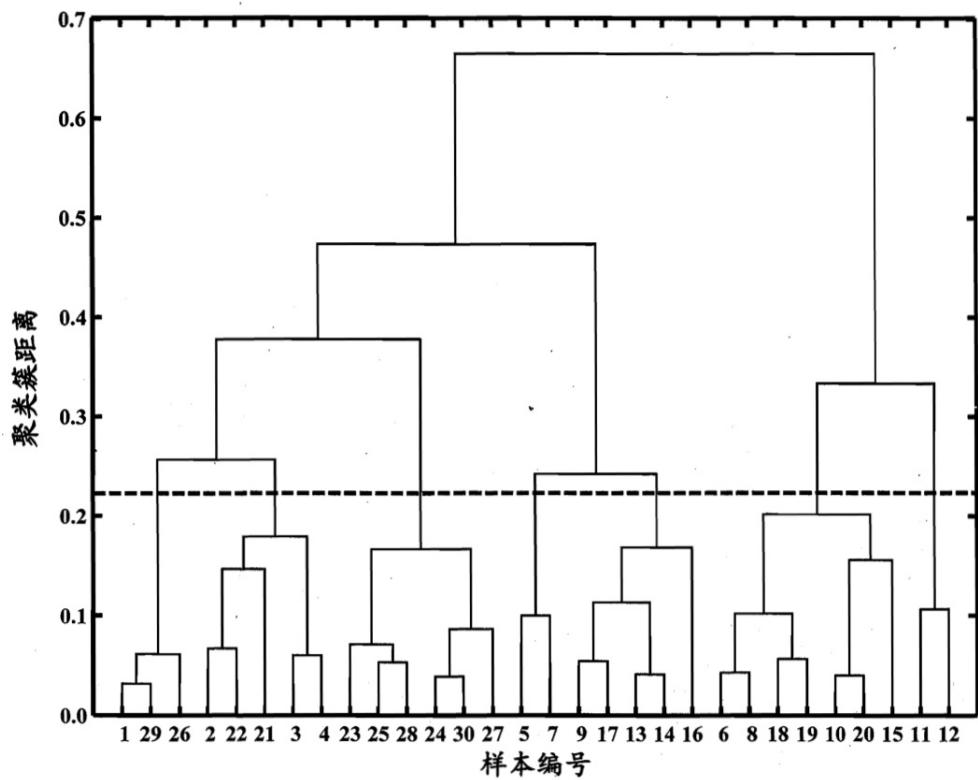


图 9.12 西瓜数据集 4.0 上 AGNES 算法生成的树状图(采用  $d_{\max}$ ). 横轴对应于样本编号, 纵轴对应于聚类簇距离.

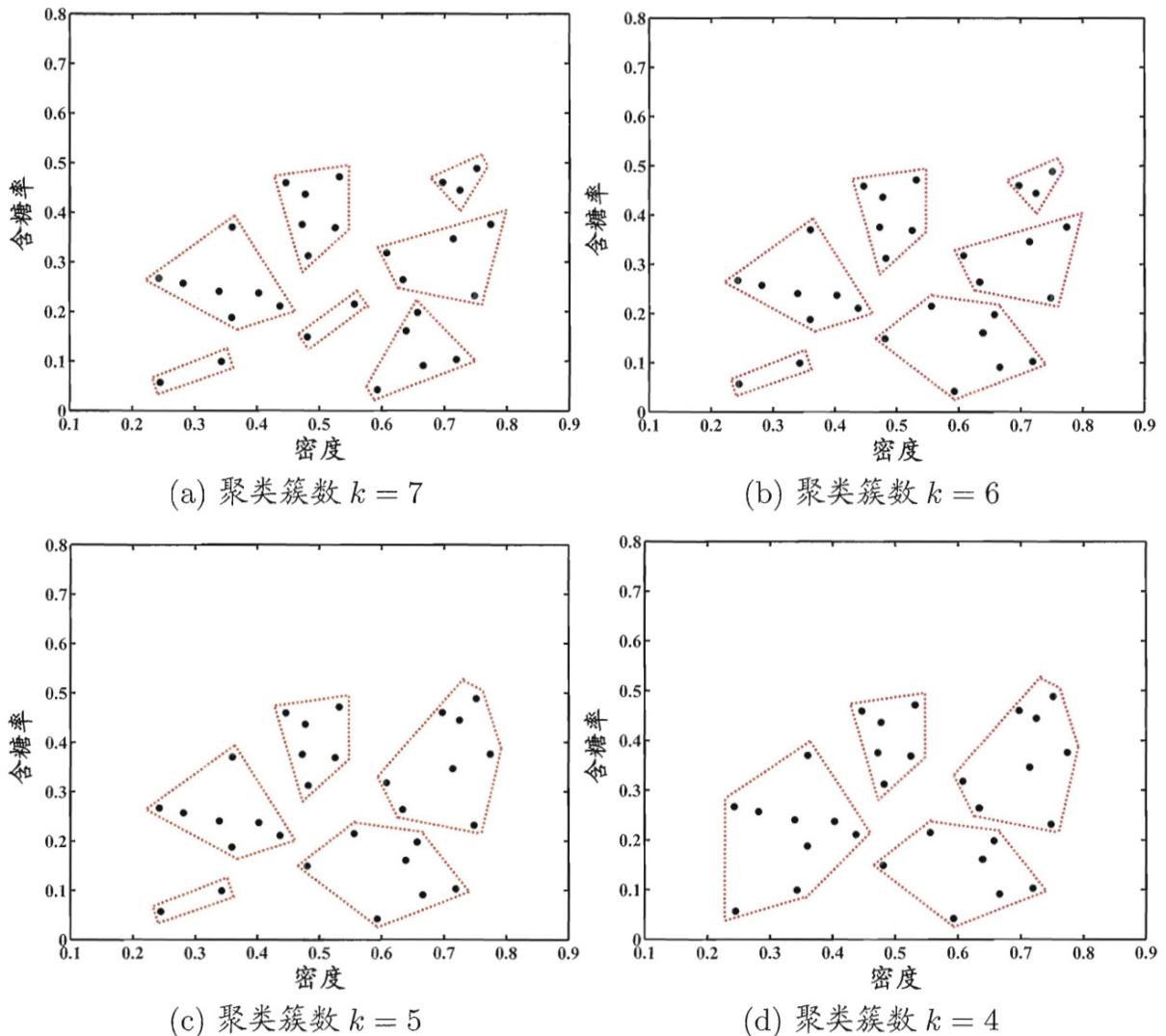


图 9.13 西瓜数据集 4.0 上 AGNES 算法(采用  $d_{\max}$ )在不同聚类簇数( $k = 7, 6, 5, 4$ )时