一、聚类任务

1. 无监督学习，训练样本的标记信息是未知的
2. 聚类仅能自动形成簇结构，但是所对应的概念语义需由使用者来把握和命名

二、性能度量

- 簇内相似度高、簇间相似度低
- 性能度量
  - 外部指标
  - 内部指标

三、距离计算

> 将属性划分为连续属性和离散属性

- 离散属性中
  - {1，2，3}这种属于有序属性
  - 不能从属性值上计算距离的为无序属性
- 各种计算距离的公式
  - 连续属性--》闵可夫斯基距离（也可用于有序属性）
  - 无序属性--》VDM
  - 混合属性--》闵可夫斯基距离 + VDM
  - 重要性不同的属性--》可以使用加权距离

四、原型聚类

4.1 k均值

k-均值算法的思路就是让同一个簇中的样本都是离本簇的中心较近，而离其他簇的中心较远。

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

过程:

从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\ldots,\boldsymbol{\mu}_k\}$
repeat
  令 $C_i=\varnothing\ (1\leqslant i\leqslant k)$
  for $j=1,2,\ldots,m$ **do**
    计算样本 $\boldsymbol{x}_j$ 与各均值向量 $\boldsymbol{\mu}_i\ (1\leqslant i\leqslant k)$ 的距离: $d_{ji}=\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\|_2$;
    根据距离最近的均值向量确定 $\boldsymbol{x}_j$ 的簇标记: $\lambda_j=\arg\min_{i\in\{1,2,\ldots,k\}} d_{ji}$;
    将样本 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j}=C_{\lambda_j}\bigcup\{\boldsymbol{x}_j\}$;
  end for
  for $i=1,2,\ldots,k$ **do**
    计算新均值向量: $\boldsymbol{\mu}_i'=\frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$;
    if $\boldsymbol{\mu}_i'\neq\boldsymbol{\mu}_i$ **then**
      将当前均值向量 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}_i'$
    else
      保持当前均值向量不变
    end if
  end for
until 当前均值向量均未更新

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

为避免运行时间过长, 通常设置一个最大运行轮数或最小调整幅度阈值, 若达到最大轮数或调整幅度小于阈值, 则停止运行.

4.2学习向量量化(LVQ)

LVQ算法是聚类算法中的一个另类，LVQ(Learning Vector Quantization)算法需要样本的类别标记，算是半个监督学习算法。

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

过程:

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

输出: 原型向量 $\{\boldsymbol{p}_1,\boldsymbol{p}_2,\ldots,\boldsymbol{p}_q\}$

$\boldsymbol{x}_j$ 与 $\boldsymbol{p}_{i^*}$ 的类别相同.

$\boldsymbol{x}_j$ 与 $\boldsymbol{p}_{i^*}$ 的类别不同.

如达到最大迭代轮数.

4.3高斯混合聚类

高斯分布就是正态分布

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

4.3.1 最大似然估计(MLE：Maxmium likehood)

4.3.2 EM算法

> EM算法可以解决机器学习中的“隐变量”问题，在真实数据场景中，有很多的样本会确实一些属性值，但是这些数据是有价值的，只是需要处理一下这些缺失的属性值，这些缺失的属性值就成为隐变量。

在训练学习器的过程中，我们是需要根据函数p(x|θ)中的x去获得θ，这样就可以获得完整的学习器：模型+参数。但是此时x是不完整的，所以我们根据上面的描述: p(x|θ)，用θ去估计x，但是θ 也是未知的。就会陷入到一个死循环中，EM算法就是想采用一种逐步迭代的思路去打破这个循环：

- 初始化一个θ，用这个初始化参数去估计x中的隐变量x_c。
- 用x_c放入样本x中，去估计下一个θ_{t+1}
- 不停的迭代循环，知道达到某个停止条件：最大迭代次数或者说是变动量小于阈值。

4.3.3 高斯混合聚类

然后假设样本是服从参数不同的高斯分布，总共有k个不同的高斯分布(看出来了吧，就是想往这个k上来靠)。那么就可以定义一个混合高斯分布

> 定义了高斯混合分布
>
> $$p_M(x) = \sum_{i=1}^{k} \alpha_i * p(x|\mu_i, \Sigma_i), \text{其中} \sum_{i=1}^{k} \alpha_i = 1$$

这个概率公式定义了每个样本的生成概率。那么我们已经有了样本，要求样本x_j的后验概率$p(z_j = i|x_j)$，也就是这个样本x_j是从哪个分布中出来的概率。根据贝叶斯公式（

）可以得到：

$$p(z_j = i|x_j) = \frac{P(z_j = i) * p_M(x_j|z_j = i)}{p_M(x_j)} = \frac{\alpha_i p(x_j|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l p(x_j|\mu_l, \Sigma_l)}$$

把这个复杂的一腿的公式记为：γ_{ij}。

根据之前的聚类套路，就是计算每个样本x_j的这个后验概率γ_{ij}，哪个最大，就属于哪个类。但是计算这个东西的过程中，又碰到了手动增加进去的"隐变量"：α，所以就需要采用EM算法，将隐变量α和模型参数μ, Σ一起计算出来。模型参数μ, Σ的推导过程大家可以去看书。

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

过程：

初始化高斯混合分布的模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$
repeat
  for $j = 1, 2, \ldots, m$ **do** — EM 算法的 E步.
    根据式(9.30)计算 $\boldsymbol{x}_j$ 由各混合成分生成的后验概率, 即 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \boldsymbol{x}_j)$ $(1 \leqslant i \leqslant k)$
  end for
  for $i = 1, 2, \ldots, k$ **do** — EM 算法的 M步.
    计算新均值向量: $\boldsymbol{\mu}_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^{m} \gamma_{ji}}$;
    计算新协方差矩阵: $\boldsymbol{\Sigma}_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} (\boldsymbol{x}_j - \boldsymbol{\mu}_i')(\boldsymbol{x}_j - \boldsymbol{\mu}_i')^{\mathrm{T}}}{\sum_{j=1}^{m} \gamma_{ji}}$;
    计算新混合系数: $\alpha_i' = \frac{\sum_{j=1}^{m} \gamma_{ji}}{m}$;
  end for
  将模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$ 更新为 $\{(\alpha_i', \boldsymbol{\mu}_i', \boldsymbol{\Sigma}_i') \mid 1 \leqslant i \leqslant k\}$
until 满足停止条件 — 例如达到最大迭代轮数.
$C_i = \varnothing$ $(1 \leqslant i \leqslant k)$
for $j = 1, 2, \ldots, m$ **do**
  根据式(9.31)确定 $\boldsymbol{x}_j$ 的簇标记 λ_j;
  将 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \bigcup \{\boldsymbol{x}_j\}$
end for

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

4.4 密度聚类

密度聚类实际上一个从某个点出发，找到自己足够近的邻居的过程，这些邻居在算法中用了一个更专业的词：邻域。一个经典的邻域算法是DBSCAN算法，算法中定义了两个邻域参数：(ϵ,MinPts)。由这两个参数确定一个邻域。

- ϵ：定义了样本集中样本与某个样本x_j的最小距离，即如果距离不大于ϵ，即样本位于样本x_j的邻域内。
- MinPts：若样本x_j的邻域包含的样本数大于MinPts，则称x_j为**核心对象。**

另外，还有几个其他的概念:

- 密度直达：若样本x_j位于样本x_i的邻域内，且x_i是核心对象，则称x_j由x_i密度直达。
- 密度可达：若x_j可以由一系列的中间节点$p, p_1 ...$由x_i直达，则称x_j由x_i密度可达。
- 密度相连：若x_i和x_j均通过x_k可达，则称x_i，x_j相连。这里两个样本是距离是相互的，但是不是每个样本点都是核心对象。

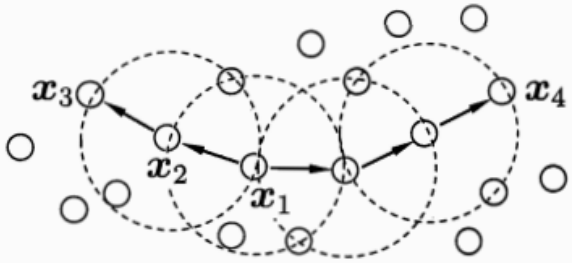


图 9.8 DBSCAN 定义的基本概念($MinPts = 3$): 虚线显示出 ϵ-邻域, x_1 是核心对象, x_2 由 x_1 密度直达, x_3 由 x_1 密度可达, x_3 与 x_4 密度相连.

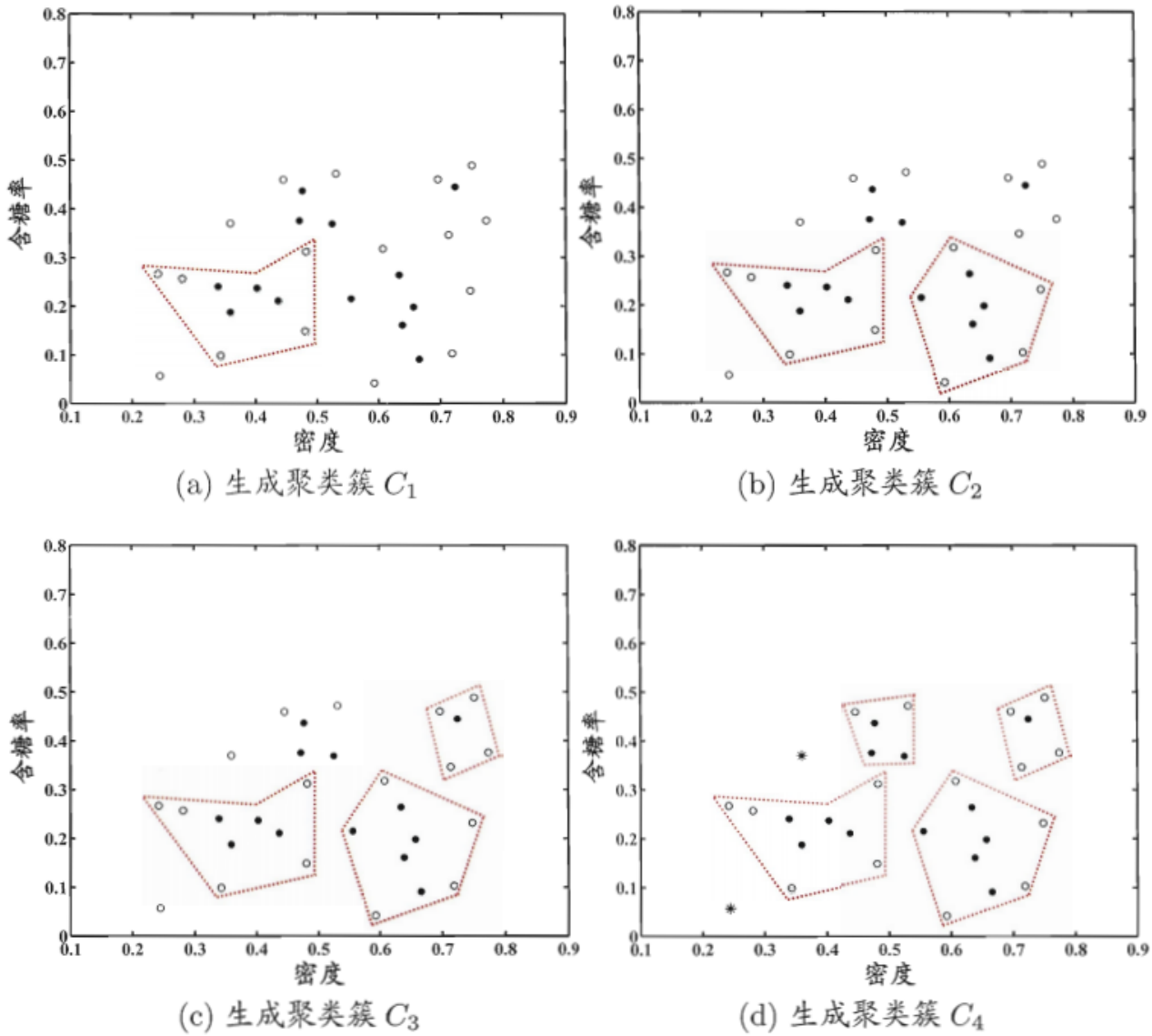
输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
邻域参数 $(\epsilon, MinPts)$.

过程:

初始化核心对象集合: $\Omega = \varnothing$
for $j = 1, 2, \ldots, m$ **do**
  确定样本 $\boldsymbol{x}_j$ 的 ϵ-邻域 $N_\epsilon(\boldsymbol{x}_j)$;
  if $|N_\epsilon(\boldsymbol{x}_j)| \geqslant MinPts$ **then**
    将样本 $\boldsymbol{x}_j$ 加入核心对象集合: $\Omega = \Omega \bigcup \{\boldsymbol{x}_j\}$
  end if
end for
初始化聚类簇数: $k = 0$
初始化未访问样本集合: $\Gamma = D$
while $\Omega \neq \varnothing$ **do**
  记录当前未访问样本集合: $\Gamma_{\text{old}} = \Gamma$;
  随机选取一个核心对象 $\boldsymbol{o} \in \Omega$, 初始化队列 $Q = < \boldsymbol{o} >$;
  $\Gamma = \Gamma \setminus \{\boldsymbol{o}\}$;
  while $Q \neq \varnothing$ **do**
    取出队列 Q 中的首个样本 $\boldsymbol{q}$;
    if $|N_\epsilon(\boldsymbol{q})| \geqslant MinPts$ **then**
      令 $\Delta = N_\epsilon(\boldsymbol{q}) \bigcap \Gamma$;
      将 Δ 中的样本加入队列 Q;
      $\Gamma = \Gamma \setminus \Delta$;
    end if
  end while
  $k = k + 1$, 生成聚类簇 $C_k = \Gamma_{\text{old}} \setminus \Gamma$;
  $\Omega = \Omega \setminus C_k$
end while

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

1. 核心对象集合初始化为空
2. 第2行到第7行是确定每个样本x_j的邻域$N_\epsilon(x_j)$，并从中挑出核心对象。
3. 从第10行开始到第24行，就是由核心对象来往外扩展生成簇的过程。
4. 从核心对象集合Ω中随机挑选一个核型对象，通过14-21行的代码将该核型对象的所有密度可达(注意密度可达的定义是需要对方也是核心对象)的样本全部加入到同一个簇内。在整个过程中，使用$\Gamma = D$不停的去减去这些簇中的样本点，最后再第22行的时候，用$\Gamma_{old} - \Gamma$就是这些样本点了，有点负负得正的意思。
    1. 刚开始$\Gamma = D$，$\Gamma_{old} = \Gamma$
    2. 选出一个核心对象样本点x_j后，$\Gamma = \Gamma - x_j$
    3. 将核心对象样本点x_j的邻域(比如有x_1, x_4, x_9)保存到Δ中，然后$\Gamma = \Gamma - \Delta$
    4. 最后用$\Gamma_{old} - \Gamma$就等于(x_j, x_1, x_4, x_9)了。
5. 形成完一个簇后，再重新从Ω中挑选核心对象进行扩展，直到Ω为空
6. 基于密度聚类算法会存在一些样本点没法成为核心对象，就会在算法结束之后成为一些孤立的样本点，成为噪声样本。



(a) 生成聚类簇 C_1　(b) 生成聚类簇 C_2

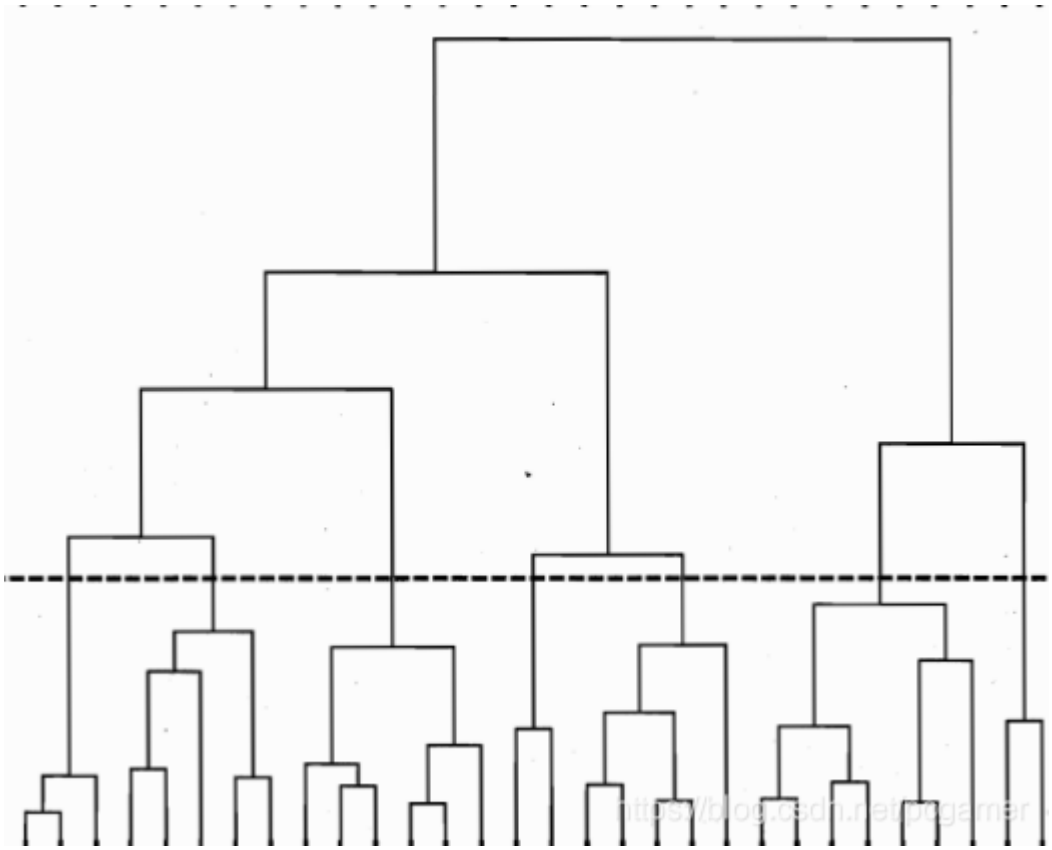
(c) 生成聚类簇 C_3　(d) 生成聚类簇 C_4

图 9.10 DBSCAN 算法($\epsilon = 0.11$, $MinPts = 5$)生成聚类簇的先后情况. 核心对象、非核心对象、噪声样本分别用"•" "○" "*"表示, 红色虚线显示出簇划分.

4.5 层次聚类

层次聚类是将样本点分拆或者合并成类似树的结构，树的每一个分支就是一个簇。书中提到的AGNES是基于合并的自底向上的方法。用的思路也比较简单：

- 首先所有的样本都单独成为一个簇。
- 计算簇和簇之间的距离，距离足够近的形成一个新的簇，最终形成一个类似树的结构：



- 通过控制簇的个数k来控制树分支的合并情况。

> 这里就有一个问题，之前计算的距离是样本之间的距离，在层次聚类里需要计算簇之间的距离，所以需要定义簇的距离如何进行计算。簇实际上就是一个样本集合，所以定义一下关于集合的某种运算即可

- 最小距离：$dist(C_i, C_j) = \min_{x \in C_i, z \in C_j} dist(x, z)$，将两个簇内样本之间的最小距离作为簇之间的距离。
- 最大距离：$dist(C_i, C_j) = \max_{x \in C_i, z \in C_j} dist(x, z)$，将两个簇内样本之间的最大距离作为簇之间的距离。
- 平均距离：$dist(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{z \in C_j} dist(x, z)$，将两个簇内样本之间的平均距离作为簇之间的距离。

算法过程

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
聚类簇距离度量函数 d;
聚类簇数 k.

过程:

for $j = 1, 2, \ldots, m$ **do**
  $C_j = \{\boldsymbol{x}_j\}$
end for
for $i = 1, 2, \ldots, m$ **do**
  for $j = 1, 2, \ldots, m$ **do**
    $M(i, j) = d(C_i, C_j)$;
    $M(j, i) = M(i, j)$
  end for
end for
设置当前聚类簇个数: $q = m$
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^*}: $C_{i^*} = C_{i^*} \bigcup C_{j^*}$;
  for $j = j^* + 1, j^* + 2, \ldots, q$ **do**
    将聚类簇 C_j 重编号为 C_{j-1}
  end for
  删除距离矩阵 M 的第 j^* 行与第 j^* 列;
  for $j = 1, 2, \ldots, q - 1$ **do**
    $M(i^*, j) = d(C_{i^*}, C_j)$;
    $M(j, i^*) = M(i^*, j)$
  end for
  $q = q - 1$
end while

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