—武大本科生课程

第12讲　聚类分析

(Lecture 12 Clustering Analysis)

武汉大学计算机学院机器学习团队

2023.04

第10章　聚类分析

10.1 聚类分析概念
10.2 层次聚类/系统聚类 (层次聚类之凝聚方法)
10.3 分解聚类 (层次聚类之分裂方法)
10.4 动态聚类
10.5 DBSCAN密度聚类 (*: 选学)
10.6 基于试探的聚类算法 (*: 自学)
10.7 聚类分析Python程序举例
小结

10.1 聚类分析概念

按照“物以类聚、人以群分”的基本思想，对未知类别的样本集根据样本之间的相似程度分类，相似的归为一类，不相似的归为另一类，故这种分类称为聚类分析(Clustering Analysis)，又常常叫做“聚类”。

相似性测度、聚类准则和聚类算法称为聚类分析的三要素。

相似性测度用于衡量同类样本的类似性和不同类样本的差异性。常用的测度有：距离、夹角余弦、相似系数等(详见第一讲课件)。

为了评价聚类效果的好坏，必须定义准则函数。有了模式相似性测度和准则函数后，聚类就变为使准则函数取极值的优化问题。常用的准则函数是误差平方和准则。

一. 分类与聚类的区别

- 分类(Classifying)：用已知类别的样本训练集来设计分类器(有监督学习： supervised learning)

  我们在前面设计分类器时，训练样本集中每个样本的类别归属都是“被标记了”(labeled)的，这种利用已标记样本集的学习方法称为有监督学习方法。

- 聚类/集群(Clustering)：事先不知样本的类别，而利用样本的先验知识来构造分类器(无监督学习： unsupervised learning)
- 无监督学习(unsupervised learning)中没有导师，只有输入数据，其学习是发现输入数据集中潜藏的结构或者规律。最常见的无监督学习是聚类问题。除了聚类问题，无监督学习还包括异常点检测(Novelty detection)，用于发现样本点中具有明显区别的样本；维数简约/降维(Dimensionality reduction)，保持原训练样本本质信息的同时将数据从高维空间映射到低维空间即特征提取(Feature extraction)，常见的方法如主成分分析(PCA: Principal Component Analysis)、KPCA(Kernel PCA)等。这两种无监督学习方法均没有用到已标记的数据，仅利用了无标记样本进行结构或者本维特征的学习。

- 半监督学习(Semi-unsupervised learning)是机器学习和模式识别的一个研究与应用热点，Google、Baidu、科大讯飞等国内外大型IT公司都在做这方面的研究和应用软件开发，它的基本原理是通过大量的无标记数据辅助少量的已标记数据进行学习，从而提高学习效果，半监督学习典型算法有十几种，如基于图的半监督学习算法、协同训练算法、半监督支持向量机等)。

二. 常见聚类方法分类

根据聚类准则及准则的应用方法，常见聚类方法如下：

1. 基于划分的方法

给定一个包含n个样本的数据集，基于划分的方法 (Partitioning Method) 就是将n个样本按照特定的度量划分为k个簇 ($k \leq n$)，使得每个簇至少包含一个样本，并且每个样本属于且仅属于一个簇，而且簇之间不存在层次关系。基于划分的方法大多数是基于距离来划分样本的，首先对样本进行初始划分，然后计算样本间的距离，重新对数据集中的样本进行划分，将样本划分到距离更近的簇中，得到一个新的样本划分，迭代计算直到聚类结果满足用户指定的要求。典型的算法有k均值算法、k-medoids算法。

2. 基于层次的方法

基于层次的方法 (Hierarchical Method) 是按层次对数据集进行划分。根据层次聚类的过程，可分为自底向上的凝聚方法和自顶向下的分裂方法。凝聚方法(Agglomerative Method)将初始数据集中的每个样本独立当作一个簇，然后根据距离、密度等度量方法，逐步将样本合并，直到将所有样本都合并到一个簇中，或满足特定的算法终止条件。分裂方法(Divisive Method)将初始数据集中的所有样本点都当作一个簇，在迭代过程中逐步将上层的簇进行分解得到更小的新簇，直到所有的簇中都只包含一个单独的样本，或满足特定的算法终止条件。

3. 基于密度的方法

大部分基于密度的方法 (Density-based Method) 采用距离度量来对数据集进行划分，在球状的数据集中能够正确划分，但是在非球状的数据集中则无法对样本进行正确聚类，并且受到数据集中的噪声数据影响较大。基于密度的方法可以克服这两个弱点。

基于密度的方法提出“密度”的思想，即给定邻域中样本点的数量，当邻域中密度达到或超过密度阈值时，将邻域内的样本包含到当前的簇中。若邻域的密度不满足阈值要求，则当前的簇划分完成，对下一个簇进行划分。基于密度的方法可以对数据集中的离群点进行检测和过滤。

4. 基于网格的方法

基于网格的方法 (Grid-based Method) 是将数据集空间划分为有限个网格单元，形成一个网络结构，在后续的聚类过程中，以网格单元为基本单位进行聚类，而不是以样本为单位。由于算法处理时间与样本数量无关，只与网络单元数量有关，因此这种方法在处理大数据集时效率很高。基于网络的方法可以在网格单元划分的基础上，与基于密度的方法、基于层次的方法等结合使用。

5. 基于模型的方法

基于模型的方法 (Model-based Method) 假定数据集满足一定的分布模型，找到这样的分布模型，就能对数据集进行聚类。基于模型的方法主要包括基于统计和基于神经网络两大类，前者以高斯混合模型GMM (Gaussian Mixture Model) 为代表，后者以自组织映射网络SOM (Self Organizing Map) 为代表。目前以基于统计模型的方法为主。

10.2 层次聚类法 (层次聚类之凝聚方法)

层次聚类法(Hierarchical Clustering Method，又称系统聚类法/谱系聚类法)：先把每个样本 (或指标) 各自作为一类，然后根据样本间的相似性和相邻性聚合。即将亲疏程度最高的两类合并，如此重复进行，直到所有的样本都合成一类。

衡量亲疏程度的指标有两种：距离、相似系数 (包括夹角余弦，相关系数等)。

相似性、相邻性一般用距离表示。

一、两类间的距离

1.最短距离(Single linkage)：两类中相距最近的两样本间的距离。

$$D_{pq} = \min_{\substack{x_i \in \omega_p \\ x_j \in \omega_q}} d_{ij}$$

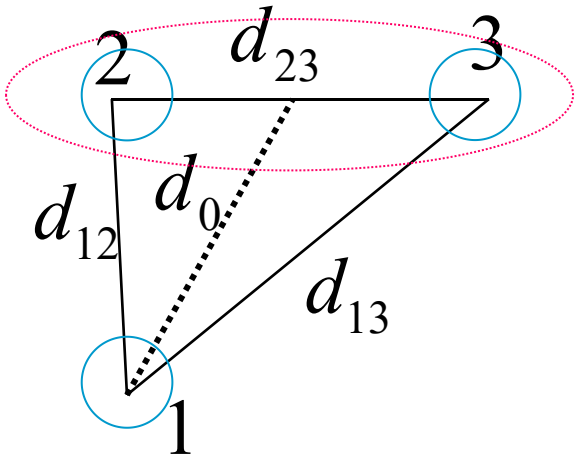
d_{ij}表示第i个样本与第j个样本之间的距离。

2.最长距离 (Complete method)：两类中相距最远的两个样本间的距离。

$$D_{pq} = \max_{\substack{x_i \in \omega_p \\ x_j \in \omega_q}} d_{ij}$$

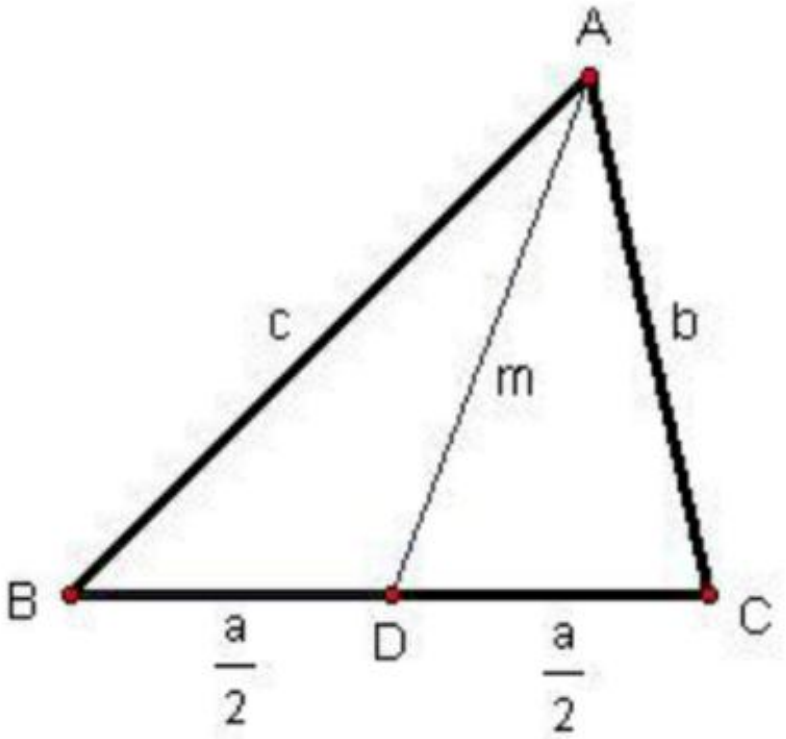
3.中间距离(Median method)：最短距离和最长距离都有片面性，因此有时采用中间距离。假设某一步将ω_2与ω_3合并为类ω_{23}，需要计算ω_{23}类与某类ω_1的距离。设ω_1类和ω_{23}类间的最短距离为d_{12}，最长距离为d_{13}，ω_{23}类的长度为d_{23}，则类ω_1与类ω_{23}的中间距离定义为：

$$d_0^2 = \frac{1}{2}d_{12}^2 + \frac{1}{2}d_{13}^2 - \frac{1}{4}d_{23}^2$$



回顾：

中线定理：$m^2=\dfrac{2b^2+2c^2-a^2}{4}$



证：由余弦定理：

$b^2=m^2+(\frac{a}{2})^2-2m(\frac{a}{2})\cos\angle ADC$

$c^2=m^2+(\frac{a}{2})^2-2m(\frac{a}{2})\cos\angle ADB$

又：$\cos\angle ADB+\cos\angle ADC=0$

$\rightarrow b^2+c^2=2m^2+2(\frac{a}{2})^2$

$\rightarrow m^2=\dfrac{2b^2+2c^2-a^2}{4}$

上式可推广为一般情况：

$$d_0^2 = \frac{1}{2}d_{12}^2 + \frac{1}{2}d_{13}^2 + \beta d_{23}^2$$

其中β为参数，$-\frac{1}{4} \le \beta \le 0$

4.重心距离(Centroid method)：两类的均值(向量)之间的距离

5.类平均距离(Average linkage)：两类中各个元素两两之间的距离平方相加后取平均值

$$D_{pq}^2 = \frac{1}{N_p N_q} \sum_{\substack{x_i \in \omega_p \\ x_j \in \omega_q}} d_{ij}^2$$

其中，$N_p : \omega_p$类的样本数，$N_q : \omega_q$类的样本数

d_{ij}为ω_p类样本点i与ω_q类样本点j之间的距离

6. 离差平方和(Ward's method, Ward法):

(1) 设N个样本已分为q类$G_1, G_2, \ldots, G_q$，则定义第i类样本的类内离差平方和为:

$$S_i = \sum_{j=1}^{N_i} (\mathbf{x}_{ji} - \bar{\mathbf{x}}_i)^T (\mathbf{x}_{ji} - \bar{\mathbf{x}}_i)$$

这里:

$\mathbf{x}_{ji}$表示第i类(即G_i)的第j个样本;

$\bar{\mathbf{x}}_i$为第i类所有样本的均值向量(即G_i的重心);

N_i为第i类的样本个数。

(2)离差平方和增量：设样本已分成ω_p、ω_q两类，若把ω_p、ω_q合并为ω_r类，则定义离差平方和增量：

$$D_{pq}^2 = S_r - (S_p + S_q)$$

其中S_p, S_q分别为ω_p类和ω_q类的离差平方和；

S_r为ω_r类的离差平方和；

用于层次聚类法/系统聚类法：增量愈小，合并愈合理

(或用于分解聚类法：增量愈大，分解越合理)。

马氏距离(Mahalanobis，马哈拉诺比斯)

表达式：$$D^2 = (\boldsymbol{X} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{X} - \boldsymbol{\mu})$$

其中，X：模式向量；μ：均值向量；

C：该类模式总体的协方差矩阵。

这是计算任一点X与数据中心的马氏距离。如果要计算数据集中任意两点（X，Y）之间的马氏距离，把其中的μ改为Y即可。

对n维向量：$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} m_1 \\ ,\mathrm{M} \\ m_n \end{bmatrix}$$

$$\boldsymbol{C} = E\left\{ (\boldsymbol{X} - \boldsymbol{\mu})\left((\boldsymbol{X} - \boldsymbol{\mu})\right)^{\mathrm{T}} \right\}$$

$$= E\left\{ \begin{bmatrix} (x_1 - m_1) \\ (x_2 - m_2) \\ \mathrm{M} \\ (x_n - m_n) \end{bmatrix} \begin{bmatrix} (x_1 - m_1) & (x_2 - m_2) & \mathrm{L} & (x_n - m_n) \end{bmatrix} \right\}$$

$$=\begin{bmatrix} E(x_1-m_1)(x_1-m_1) & E(x_1-m_1)(x_2-m_2) & \cdots & E(x_1-m_1)(x_n-m_n) \\ E(x_2-m_2)(x_1-m_1) & E(x_2-m_2)(x_2-m_2) & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ E(x_n-m_n)(x_1-m_1) & \cdots & \cdots & E(x_n-m_n)(x_n-m_n) \end{bmatrix}$$

$$=\begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \ddots & \sigma_{jk}^2 & \vdots \\ \vdots & \vdots & \sigma_{kk}^2 & \vdots \\ \sigma_{n1}^2 & \cdots & \cdots & \sigma_{nn}^2 \end{bmatrix}$$

马氏距离：在各分量特征维度计算样本模式与均值模式的距离上，剔除了该维度上模式类的方差影响——方差大，说明模式取值变化大，因此计算距离时要用方差归一化，这样得出的分量模式与均值模式的距离才具有比较意义。

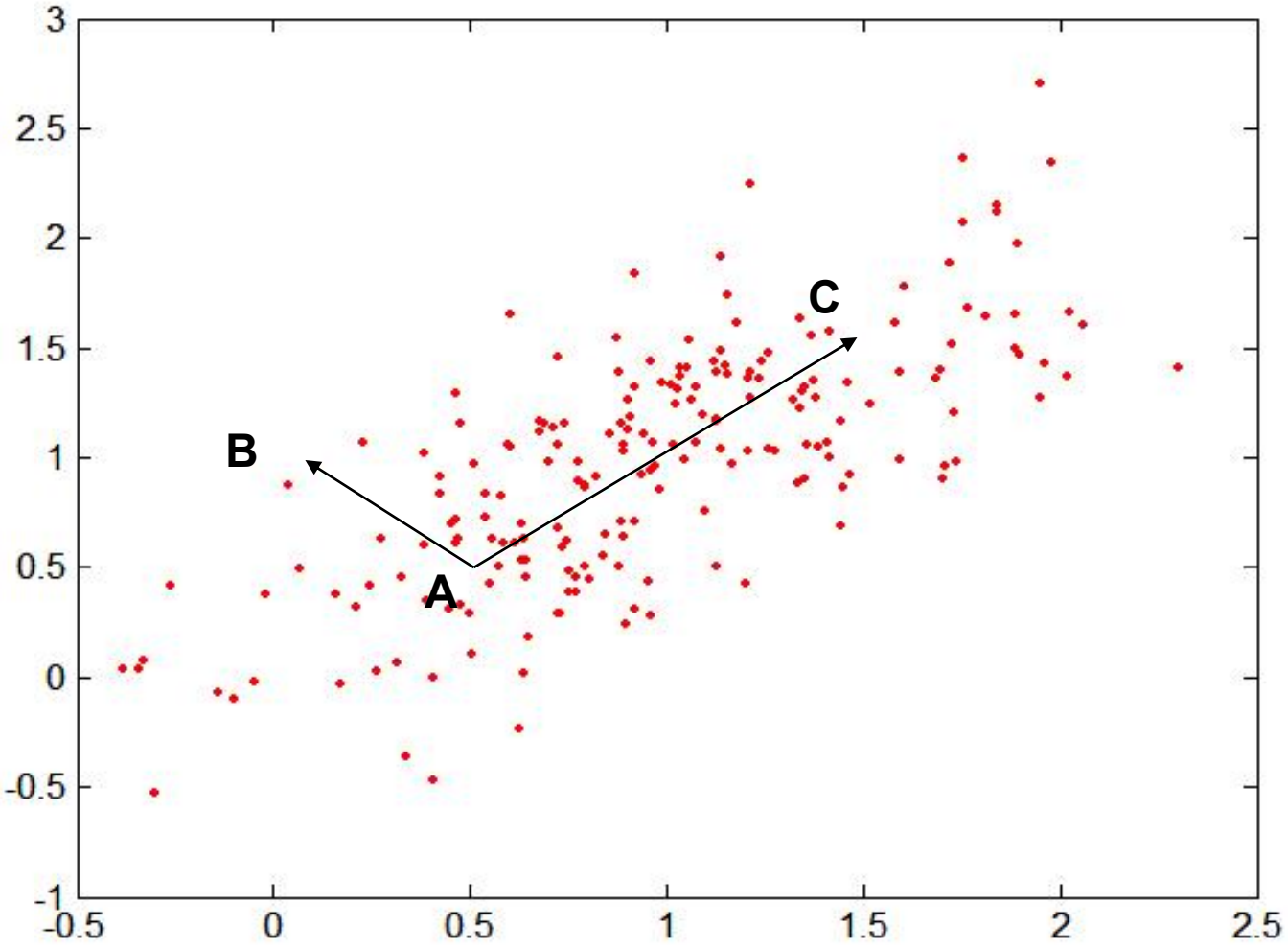
举例：以厘米为单位来测量人的身高，以克（g）为单位测量人的体重

A(160,60000) B(160,59000) C(170,60000)

优点：排除了模式样本之间的相关性影响 。

特例：当$C = I$ 时，马氏距离退化为欧氏距离。

Mahalanobis 距离示例



Cov Matrix:

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

A: (0.5, 0.5)

B: (0, 1)

C: (1.5, 1.5)

Mahal(A,B) = 5

Mahal(A,C) = 4

二、层次聚类/系统聚类算法

层次聚类基本思想：将距离阈值(distance threshold value)作为决定聚类数目的标准，基本思路是每个样本先自成一类，然后按距离准则逐步合并，减少类别数，直到达到分类要求为止。

层次聚类算法步骤描述：

(1)初始分类。 假设有N个样本，每个样本自成一类，则有N类：$G_1(0)$，$G_2(0)$，…，$G_N(0)$ {$G_i(k)$表示第k次合并时的第i类，此步k=0}。

(2)计算各类之间的距离, 得m×m维的距离矩阵D(k), m为类别个数(初始时m=N，k=0)。

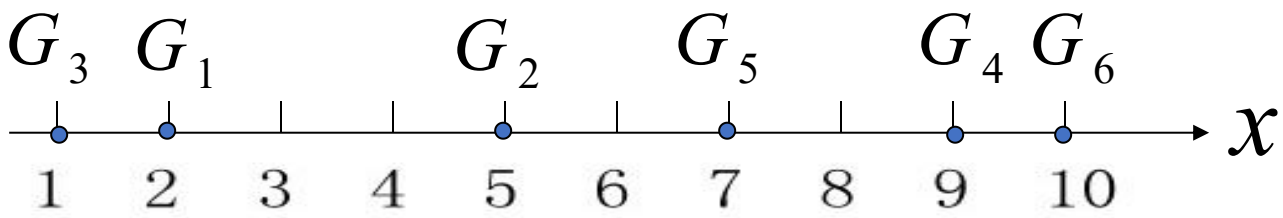
(3) 找出距离矩阵D(k)中类间距离最小的元素，如果它对应着$G_i(k)$和$G_j(k)$，则将类$G_i(k)$与$G_j(k)$合并为一类，由此得到新的聚类$G_1(k+1)$，$G_2(k+1)$，…。令k=k+1，m=m-1，计算距离矩阵D(k)。

(4) 若D(k)中类间距离最小值大于距离阈值*T*(*T*hreshold)，则算法停止(这意味着所有类间距均大于要求的阈值*T*，各类已经足够分开了)，所得分类即为聚类结果(或者，如果所有的样本被聚成两类，则算法停止)；否则，转(3)。

层次聚类法将样本逐步聚类，类别由多到少。层次聚类算法的特点是在聚类过程中类的中心不断地调整，但样本一旦归到某个类以后就不会再改变。

系统聚类举例：

如下图所示。



(1)设全部样本分为6类；

(2) 作距离矩阵D(0)，见下表：

	ω_1	ω_2	ω_3	ω_4	ω_5
ω_2	9				
ω_3	(1)	16			
ω_4	49	16	64		
ω_5	25	4	36	4	
ω_6	64	25	81	(1)	9

(3)求最小元素；

(4)把ω1,ω3合并ω7=(1, 3)； ω4,ω6合并ω8=(4, 6)；

(5)作距离矩阵D(1)；

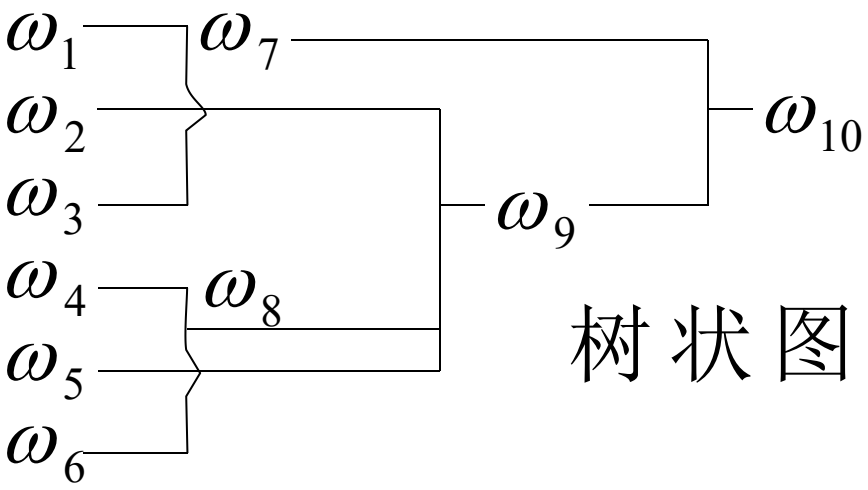
	ω_7	ω_2	ω_8
ω_2	9		
ω_8	49	16	
ω_5	25	4	4

说明: 这里采用最小距离法求距离D(1) 。

(6) 若合并的类数没有达到要求，转(3)；
否则停止。

(3) 求最小元素。

(4) ω_8，ω_5，ω_2合并，$\omega_9=(2, 5, 4, 6)$。



树状图/系统聚类树

10.3 分解聚类法 (层次聚类之分裂方法)

分解聚类法 (Decomposition Clustering Method)：把全部样本作为一类，然后根据相似性、相邻性进行分解。

通俗地讲，分解聚类法是首先把全部样本当作一类，然后再分为两类，三类，…，直至所有的样本自成一类为止。

目标函数：两类均值(重心)方差

$$E=\frac{N_1N_2}{N}(\overline{x_1}-\overline{x_2})^T(\overline{x_1}-\overline{x_2})$$

N：总样本数，N_1：ω1类样本数，

N_2：ω2类样本数。

下面介绍一分为二的分裂方法：

设有N个样本当作一类记为ω, 将它分成两个子类ω_1 和ω_2且各有N_1 和N_2个样本, 记ω, ω_1, ω_2 的重心分别为$\bar{x}, \bar{x}_1, \bar{x}_2$。 ω, ω_1, ω_2 的离差平方和分别为：

$$S = \sum_{\mathbf{x}_i \in \omega} (x_i - \bar{x})^T (x_i - \bar{x})$$

$$S_j = \sum_{\mathbf{x}_i \in \omega_j} (x_i - \bar{x}_j)^T (x_i - \bar{x}_j), \quad j = 1, 2$$

“一分为二”的思想就是要使S_1+S_2 尽可能小, 或使离差平方和增量$S-(S_1+S_2)= S- S_1- S_2$尽可能大。

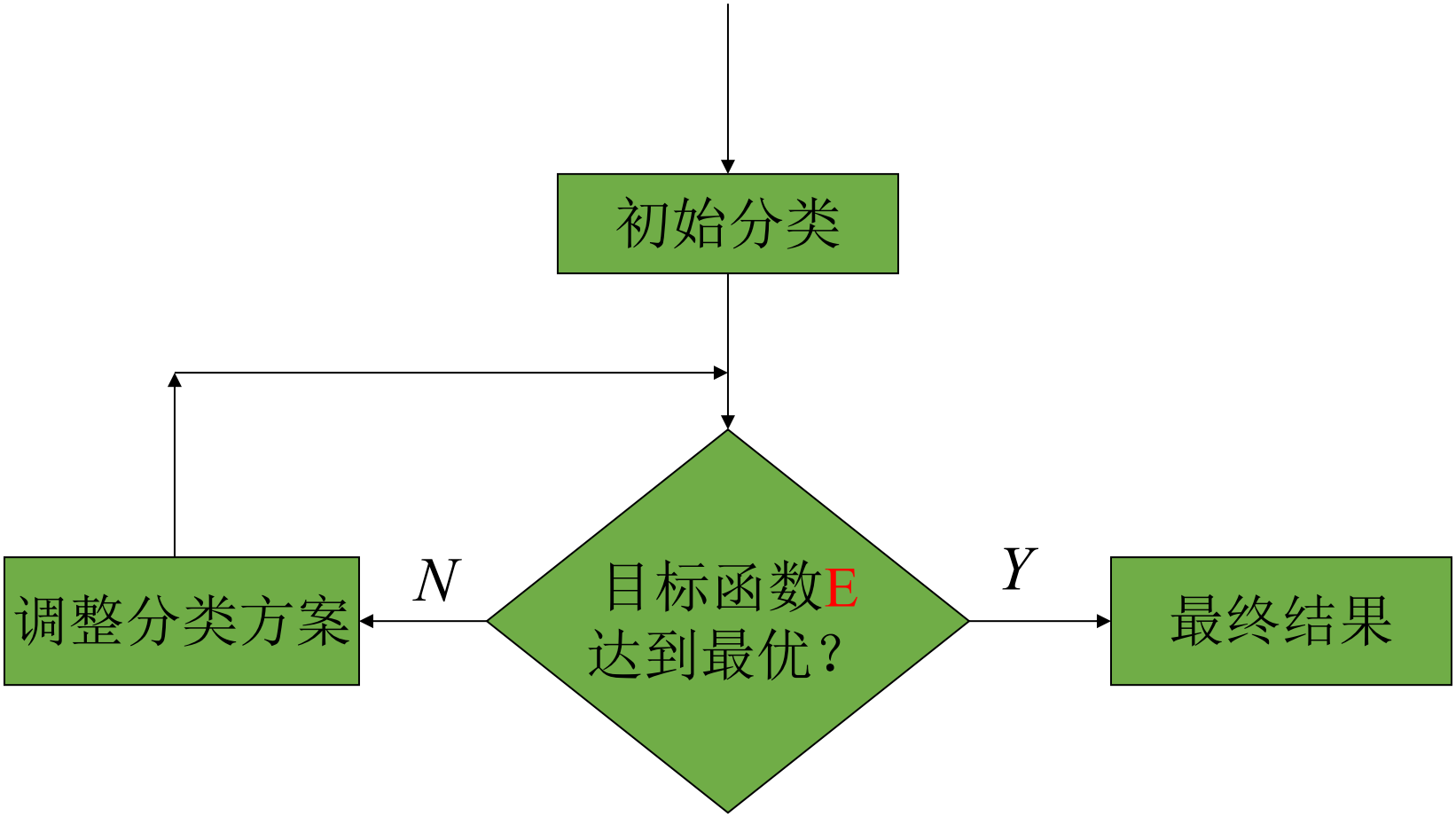
可以证明(此略)：

$$E = S - S_1 - S_2$$

$$= \frac{N_1 N_2}{N} (\overline{x}_1 - \overline{x}_2)^T (\overline{x}_1 - \overline{x}_2)$$

N：总样本数，N_1：ω1类样本数，
N_2：ω2类样本数。

把上面两类均值(重心)的方差E作为目标函数，选择某种分法使得E达到最大。



分解聚类框图

对分法(一种一分为二的分裂方法，它是将一批样本分成两类的一种方法)：

一开始所有n个样本均在ω1 中，然后找一个样本把它归入ω2使E达到最大，接着再找第二个样本归入ω2 使E 达到最大，如此继续下去(某样本一旦归入ω2后，该样本在以后的划分中就不再回到ω1) 。

令E(k)表示ω2中有k个样本，那么一定存在k*使得：

$$V(k^*)= \max_{1\le k\le n} E(k)$$

于是将前k*次进入ω2 的样本归为一类，其余n-k*个样本为另一类。再将上述步骤应用于每个子类直至每个样本都自成一类。

举例1：已知21个样本，每个样本有两个特征，原始资料列表如下：

样本号	1	2	3	4	5	6	7	8	9	10
x_1	0	0	2	2	4	4	5	6	6	7
x_2	6	5	5	3	4	3	1	2	1	0

11	12	13	14	15	16	17	18	19	20	21
−4	−2	−3	−3	−5	1	0	0	−1	−1	−3
3	2	2	0	2	1	−1	−2	−1	−3	−5

解：第一次分类时计算所有样本，分别划到G2时的E值，找出最大的。

1. 开始时，$G_1^{(0)}=(x_1,x_2,...,x_{21})$　　$G_2^{(0)}=$空

$$\therefore\ \bar{x}_1^{(0)}=\begin{pmatrix}0.714\\1.333\end{pmatrix}\quad \bar{x}_2^{(0)}=\begin{pmatrix}0\\0\end{pmatrix}\quad N_1^{(0)}=21, N_2^{(0)}=0$$

$$\therefore \text{目标函数}\quad E=\frac{N_1N_2}{N}(\bar{x}_1-\bar{x}_2)^T(\bar{x}_1-\bar{x}_2)=0$$

2. 分别计算当$x_1, x_2, \ldots, x_n$划入G_2时的E值

把x_1划入G_2时有：

$$\bar{x}_1^{(1)} = \bar{x}_1^{(0)} + \frac{\bar{x}_1^{(0)} - x_1}{N_1^{(0)} - 1}$$

$$= \begin{pmatrix} 0.714 \\ 1.333 \end{pmatrix} + \frac{\left[\begin{pmatrix} 0.714 \\ 1.333 \end{pmatrix} - \begin{pmatrix} 0 \\ 6 \end{pmatrix} \right]}{(21-1)} = \begin{pmatrix} 0.75 \\ 1.10 \end{pmatrix},$$

$$\bar{x}_2^{(1)} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

$$E = \frac{20 \times 1}{21}\left[0.75^2 + (1.10-6)^2\right] = 23.40$$

然后再把$x_2,x_3,\ldots,x_{21}$ 划入G2时对应的E值，找出一个最大的E值。

把 x_{21}划为G2的E值最大。

$\therefore\ G_1^{(1)}=(x_1,x_2,\ldots,x_{20}),\ G_2^{(1)}=(x_{21})$

$$\overline{x_1}=\begin{pmatrix}0.9\\1.65\end{pmatrix},\quad \overline{x_2}=\begin{pmatrix}-3\\-5\end{pmatrix},\ N_1^{(1)}=20,\ N_2^{(1)}=1$$

E(1)=56.

6此时，将x_{21}固定在$G_2^{(1)}$中，再继续进行第2、第3次迭代，…；计算出 E(2) , E(3) , …。

迭代次数(k)	G1→G2	E值
1	x_{21}	56.6
2	x_{20}	79.16
3	x_{18}	90.90
4	x_{14}	102.61
5	x_{15}	120.11
6	x_{19}	137.15
7	x_{11}	154.10
8	x_{13}	176.15
9	x_{12}	195.26
10	x_{17}	213.07
11	x_{16}	212.01

第10次迭代 x_{17} 划入G2时，E最大。于是分成以下两类：$G_1=(x_1,x_2,...,x_{10},x_{16})$

$$G_2=(x_{11},x_{12},x_{13},x_{14},x_{15},x_{17},x_{18},x_{19},x_{20},x_{21})$$

每次分类后要重新计算 $\overline{x_1}$，$\overline{x_2}$ 的值。可用以下递推公式：

$$x_1^{(k+1)}=\overline{x_1}^{(k)}+(\overline{x_1}^{(k)}-x_i)/(N_1^{(k)}-1)$$

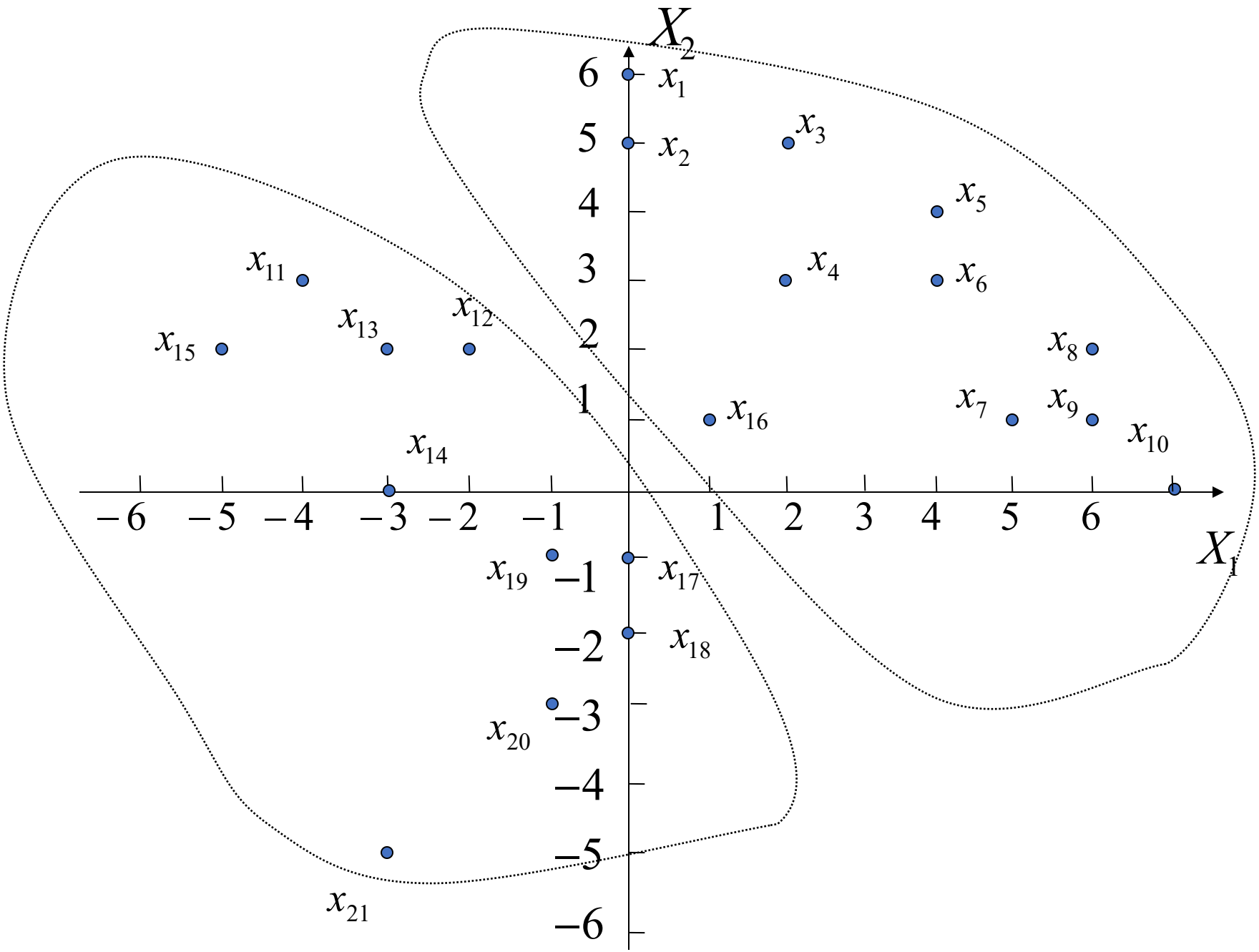
$$x_2^{(k+1)}=\overline{x_2}^{(k)}-(\overline{x_2}^{(k)}-x_i)/(N_2^{(k)}+1)$$

$\overline{x}_1^{(k)}$，$\overline{x}_2^{(k)}$是第 k 步对分时两类均值；

x_1^{k+1}，x_2^{k+1}是下一次对分时把 x_i

从 $G_1^{(k)}$划到 $G_2^{(k)}$时的两类均值；

$N_1^{(k)}$，$N_2^{(k)}$为二类样本数。


X_2
X_1
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
x_{10}
x_{11}
x_{12}
x_{13}
x_{14}
x_{15}
x_{16}
x_{17}
x_{18}
x_{19}
x_{20}
x_{21}
6
5
4
3
2
1
−1
−2
−3
−4
−5
−6
−6
−5
−4
−3
−2
−1
1
2
3
4
5
6

举例2：运用分解聚类法对经济差距进行分类统计与决策(小论文阅读)。

论文来源：
万树平. 运用分解聚类法对经济差距的分类统计与决策，统计与决策， 2007年第5期，P139。
(可在武汉大学图书馆网站进“中国知网” 检索“运用分解聚类法对经济差距的分类”，然后下载文件阅读，这里的下载文件名：“运用分解聚类法对经济差距的分类.PDF”)

10.4 动态聚类法

一、动态聚类法概要

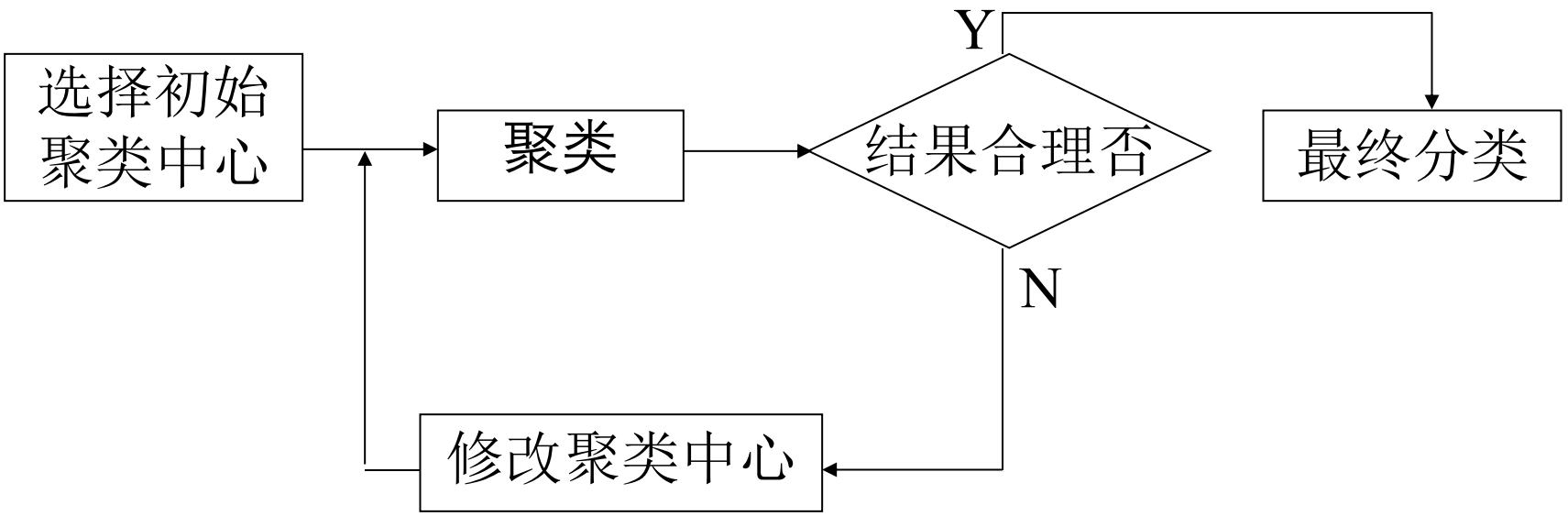
(Dynamic Clustering Algorithm)

动态聚类法首先选择若干个样本作为聚类中心，再按照事先确定的聚类准则进行聚类。在聚类过程中，根据聚类准则对聚类中心进行反复修改，直到分类合理为止。

动态聚类有如下三个要点：

1. 先选定某种距离作为样本间的相似性度量；
2. 确定评价聚类结果的准则函数；
3. 给出某种初始分类，用迭代法找出使准则函数取极值的最好的聚类结果。

动态聚类法基本思想如下图所示。“动态”即指聚类过程中，聚类中心不断被修改的变化状态。



动态聚类基本思路框图

二、代表点的选取方法：代表点就是初始分类的聚类中心数K

1. 用样本集的前K个样本点作为代表点;
2. 将全部样本随机分成K类，计算每类重心，把这些重心作为每类的代表点;
3. 凭经验选代表点，根据问题的性质、数据分布，从直观上看来较合理的代表点K;

4. 按密度大小选代表点

以每个样本作为球心，以d为半径做球形；落在球内的样本数称为该点的密度，并按密度大小排序。首先选密度最大的样本点作为第一个代表点，即第一个聚类中心。再考虑第二大密度点，若第二大密度点距第一个代表点的距离大于d_1(预先规定的正数)，则把第二大密度点作为第二个代表点，否则不能作为代表点。这样按密度大小依次考察下去，所选代表点间的距离都大于d_1。d_1太小，代表点太多，d_1太大，代表点太小，一般选$d_1=2d$。对代表点内的密度一般要求大于T。$T>0$为规定的一个正数。这样选择K个样本点(即K个代表点)作为初始聚类中心。

5. 将相距最远的K个样本点作为代表点。

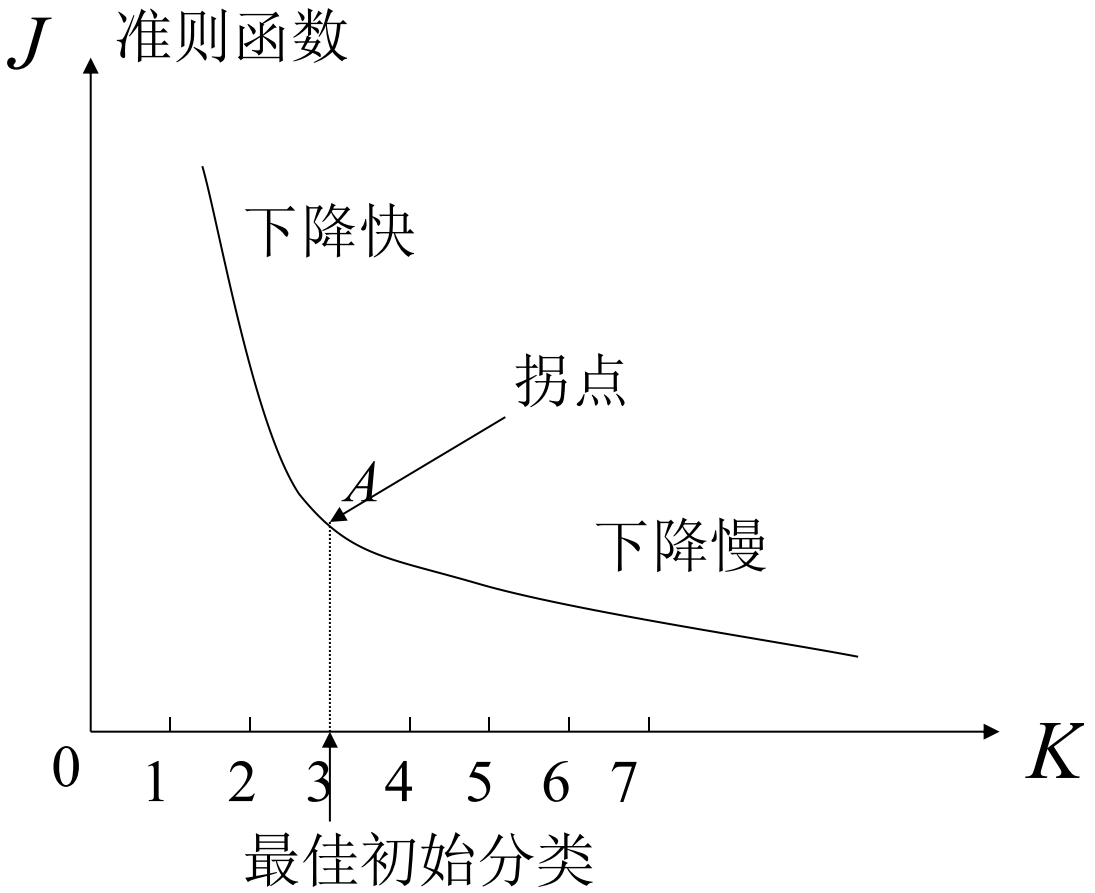
三、初始分类和调整

1. 选一批代表点后，代表点就是聚类中心，计算其他样本到聚类中心的距离，把所有样本归于最近的聚类中心点，形成初始分类，再重新计算各聚类中心，称为成批处理法。
2. 选一批代表点后，依次计算其他样本的归类，当计算完第一个样本时，把它归于最近的一类，形成新的分类。再计算新的聚类中心，并计算第二个样本到新的聚类中心的距离，对第二个样本归类。即每个样本的归类都改变一次聚类中心。此法称为逐个处理法。

3.直接用样本进行初始分类，先规定距离d，把第一个样本作为第一类的聚类中心，考察第二个样本，若第二个样本距第一个聚类中心距离小于d，就把第二个样本归于第一类，否则第二个样本就成为第二类的聚类中心，再考虑其他样本，根据样本到聚类中心距离大于还是小于d，决定分裂还是合并。

4.最佳初始分类

*K*均值算法，其聚类中心数目(初始分类数目)假定已知为*K*个。如下图所示，随着初始分类*K*的增大，准则函数下降很快，经过拐点A后，下降速度减慢。拐点*A*就是最佳初始分类。



四、*K*次平均算法/*K*均值算法 (K/C-means)

*K*均值算法也称*C*均值算法，是根据函数准则进行分类的聚类算法，基于使聚类准则函数最小化。这里所用的聚类准则函数是聚类中每一个样本点到各聚类中心的平方和。对于第j个聚类集，准则函数定义为

$$J_j = \sum_{i=1}^{N_j} \left\| \mathbf{x}_i - \mathbf{z}_j \right\|^2, \mathbf{x}_i \in G_j$$

式中,G_j表示第j个聚类集, 也称为聚类域, 其聚类中心为$\mathbf{z}_j$; N_j为第j个聚类域G_j所包含的样本个数.

对所有K个模式类有准则函数:

$$J=\sum_{j=1}^{K}\sum_{i=1}^{N_j}\left\|\mathbf{x}_i-\mathbf{z}_j\right\|^2,\mathbf{x}_i\in G_j$$

K均值算法的聚类准则是:聚类中心$\mathbf{z}_j$ 的选择应使准则函数J 极小,也就是使J_j的值极小,要满足这一点,应有:

$$\frac{\partial J_j}{\partial \mathbf{z}_j}=0$$

即: $$\frac{\partial}{\partial \mathbf{z}_j}\sum_{i=1}^{N_j}\left\|\mathbf{x}_i-\mathbf{z}_j\right\|^2=\frac{\partial}{\partial \mathbf{z}_j}\sum_{i=1}^{N_j}(\mathbf{x}_i-\mathbf{z}_j)^T(\mathbf{x}_i-\mathbf{z}_j)=0$$

可解得:

$$\mathbf{z}_j=\frac{1}{N_j}\sum_{i=1}^{N_j}\mathbf{x}_i,\quad \mathbf{x}_i\in G_j$$

上式表明,G_j类的聚类中心应选为该类样本的均值向量.

推导过程:

$$\frac{\partial}{\partial \mathbf{z}_j}\sum_{i=1}^{N_j}\left\|\mathbf{x}_i-\mathbf{z}_j\right\|^2$$

$$=\sum_{i=1}^{N_j}\frac{\partial}{\partial \mathbf{z}_j}(\mathbf{x}_i-\mathbf{z}_j)^T(\mathbf{x}_i-\mathbf{z}_j)$$

$$=\sum_{i=1}^{N_j}\frac{\partial}{\partial \mathbf{z}_j}(\mathbf{x}_i-\mathbf{z}_j)^T(\mathbf{x}_i-\mathbf{z}_j)=\sum_{i=1}^{N_j}\{-2(\mathbf{x}_i-\mathbf{z}_j)\}=0$$

即: $\sum_{i=1}^{N_j}(\mathbf{x}_i-\mathbf{z}_j)=0\Rightarrow, N_j\mathbf{z}_j=\sum_{i=1}^{N_j}\mathbf{x}_i$, 得:

$$\mathbf{z}_j=\frac{1}{N_j}\sum_{i=1}^{N_j}\mathbf{x}_i,\quad \mathbf{x}_i\in G_j$$

*K*均值算法使用的聚类准则函数是误差平方和准则，通过反复迭代优化聚类结果，使所有样本到各自所属类别的中心的距离平方和达到最小。

条件：设待分类的模式向量集为$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$，类的数目*K*是事先取定的。

K/*C*均值算法步骤描述：

(1) 任选*K*个初始聚类中心：$z_1(0), z_2(0), \ldots, z_K(0)$；

其中，圆括号中的数字表示聚类过程中的迭代运算次数，用*r*表示，令初值*r*=0。

(2) 将待分类的模式特征向量集$\{\mathbf{x}_i\}$中的样本$\mathbf{x}_i$逐个按最小距离原则划分给K类中的某一类，即：

如果 $d_{ij}(r)=\min_s\{d_{is}(r)\}=\min_s\left\|\mathbf{x}_i-\mathbf{z}_s(r)\right\| \quad (i=1,2,...,N)$

则判定 $\mathbf{x}_i\in G_j(r+1) \quad (j=1,2,...,K)$

式中$d_{ij}(r)$即为$\mathbf{x}_i$和$G_j(r)$的聚类中心$\mathbf{z}_j(r)$的最小距离，r表示迭代次数。于是产生新的聚类$G_j(r+1) \quad (j=1,2,...,K)$

(3) 计算重新分类后的各聚类中心，即：

$$\mathbf{z}_j(r+1)=\frac{1}{n_j(r+1)}\sum_{\mathbf{x}_i\in G_j(r+1)}\mathbf{x}_i \qquad (j=1,2,...,K)$$

式中$n_j(r+1)$为$\mathbf{x}_i\in G_j(r+1)$类中所含模式样本的个数。因为这一步采用取平均的方法计算调整后各类的中心，且定为K类，故称为*K*均值法。

(4) 如果$\mathbf{z}_j(r+1)=\mathbf{z}_j(r)(j=1,2,...,K)$，则结束；

否则，r=r+1，转至步骤(2)。

*K*均值算法讨论：

*K*均值算法是否有效主要受以下几个因素的影响：

(1)选取的聚类中心数(代表点)是否符合模式的实际分布；

(2)所选聚类中心的初始位置；

(3)模式样本分布的几何性质；

(4)样本读入的次序。

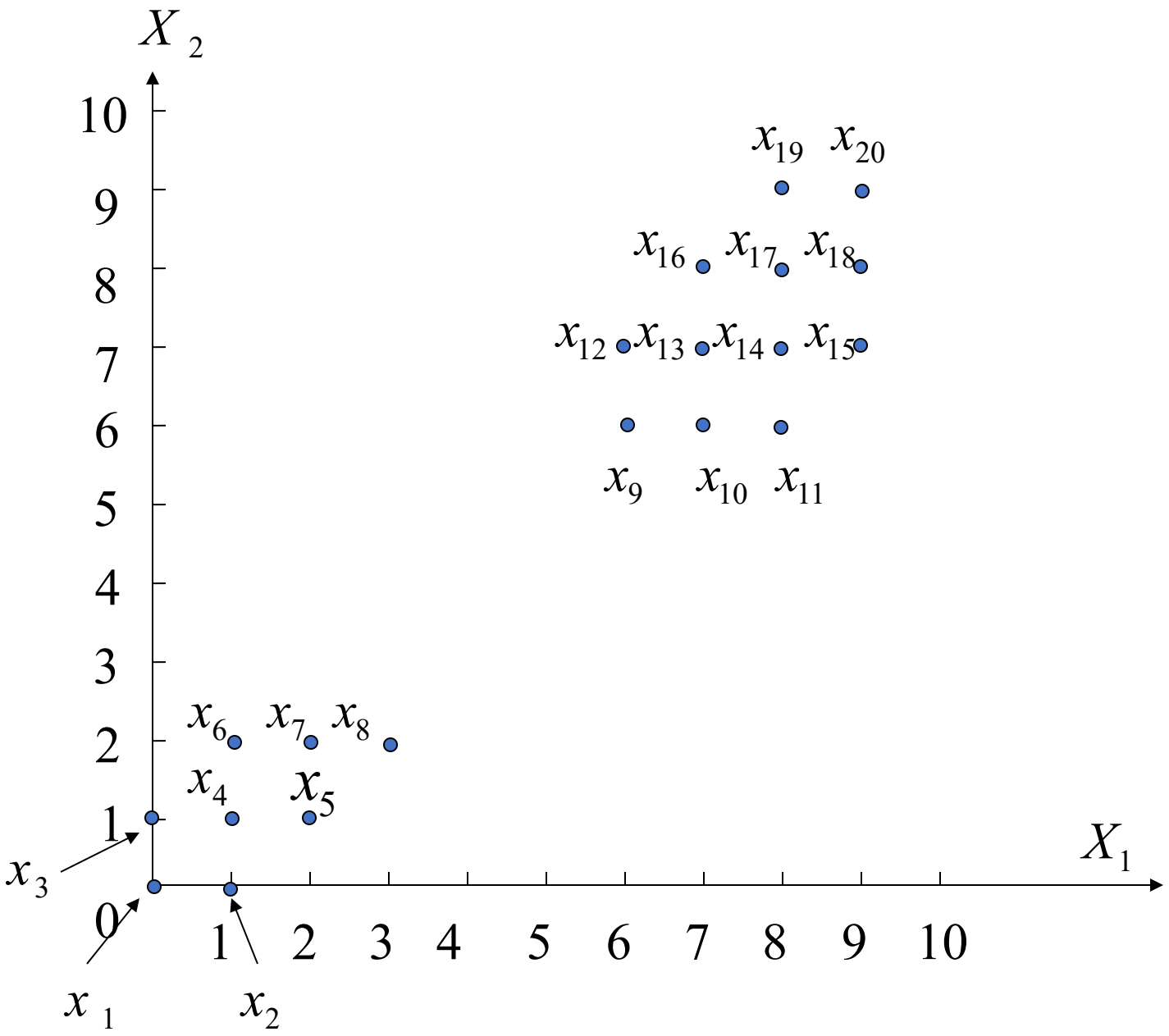
实际应用中，需要试探不同的*K*值和选择不同的聚类中心初始值。如果模式样本形成几个距离较远的孤立分布的小块区域，一般结果都会收敛。

*K*均值算法举例：

已知有20个样本，每个样本有2个特征，数据分布如下表，*K*=2，试用*K*均值算法进行聚类。

样本序号	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
特征x_1	0	1	0	1	2	1	2	3	6	7
特征x_2	0	0	1	1	1	2	2	2	6	6

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
8	6	7	8	9	7	8	9	8	9
6	7	7	7	7	8	8	8	9	9


X_2
10
9
8
7
6
5
4
3
2
1
0
x_{19} x_{20}
x_{16} x_{17} x_{18}
x_{12} x_{13} x_{14} x_{15}
x_9 x_{10} x_{11}
x_6 x_7 x_8
x_4 x_5
x_3
X_1
1 2 3 4 5 6 7 8 9 10
x_1
x_2

第一步：K=2，选初始聚类中心为

$$Z_1(0)=\mathbf{x}_1=(0,0)^T;Z_2(0)=\mathbf{x}_2=(1,0)^T$$

第二步：$\|\mathbf{x}_1 - Z_1(0)\| = \left\| \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\| = 0$

$$\|\mathbf{x}_1 - Z_2(0)\| = \left\| \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\| = 1$$

因为$\|\mathbf{x}_1 - Z_1(0)\| < \|\mathbf{x}_1 - Z_2(0)\|$

所以$\mathbf{x}_1 \in G_1(1)$

$$\|\mathbf{x}_2 - Z_1(0)\| = \left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\| = 1$$

$$\|\mathbf{x}_2 - Z_2(0)\| = \left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\| = 0$$

因为$\|\mathbf{x}_2 - Z_1(0)\| > \|\mathbf{x}_2 - Z_2(0)\|,$

所以$\mathbf{x}_2 \in G_2(1)$

同理

$\|\mathbf{x}_3 - Z_1(0)\| = 1 < \|\mathbf{x}_3 - Z_2(0)\| = 2, \therefore \mathbf{x}_3 \in G_1(1)$

$\|\mathbf{x}_4 - Z_1(0)\| = 2 > \|\mathbf{x}_4 - Z_2(0)\| = 1, \therefore \mathbf{x}_4 \in G_2(1)$

同样把所有$\mathbf{x}_5, \mathbf{x}_6, ..., \mathbf{x}_{20}$与第二个聚类中心的距离计算出来，判断$\mathbf{x}_5, \mathbf{x}_6, ..., \mathbf{x}_{20}$都属于$G_2(1)$

因此分为两类：

1、$G_1(1) = (\mathbf{x}_1, \mathbf{x}_3),$

2、$G_2(1) = (\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5, ..., \mathbf{x}_{20})$

$N_1 = 2, N_2 = 18$

第三步：计算新的聚类中心

$$Z_1(1)=\frac{1}{N_1}\sum_{\mathbf{x}\in G_1(1)}\mathbf{x}=\frac{1}{2}(\mathbf{x}_1+\mathbf{x}_3)=\frac{1}{2}\left[\begin{pmatrix}0\\0\end{pmatrix}+\begin{pmatrix}0\\1\end{pmatrix}\right]$$

$$=\frac{1}{2}\begin{pmatrix}0\\1\end{pmatrix}=(0,0.5)^T$$

$$Z_2(1)=\frac{1}{N_2}\sum_{\mathbf{x}\in G_2(1)}\mathbf{x}=\frac{1}{18}(\mathbf{x}_2+\mathbf{x}_4+\mathbf{x}_5+\ldots+\mathbf{x}_{20})$$

$$=(5.67,5.33)^T$$

第四步：因$Z_j(1) \neq Z_j(0)(j=1,2)$，故r=r+1=1，转第二步’。

第二步’：由新的聚类中心，得

$$\|\mathbf{x}_l - Z_1(1)\| < \|\mathbf{x}_l - Z_2(1)\| \qquad l = 1,2,...,8$$

$$\|\mathbf{x}_l - Z_1(1)\| > \|\mathbf{x}_l - Z_2(1)\| \qquad l = 9,10,...,20$$

故得：

$$G_1(2)=(\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_8),N_1=8$$

$$G_2(2)=(\mathbf{x}_9,\mathbf{x}_{10},...,\mathbf{x}_{20}),N_2=12$$

第三步’：计算聚类中心

$$Z_1(2)=\frac{1}{N_1}\sum_{\mathbf{x}\in G_1(2)}\mathbf{x}=\frac{1}{8}(\mathbf{x}_1+\mathbf{x}_2+\mathbf{x}_3+...+\mathbf{x}_8)$$

$$=(1.25,1.13)^T$$

$$Z_2(2)=\frac{1}{N_2}\sum_{\mathbf{x}\in G_2(2)}\mathbf{x}=\frac{1}{12}(\mathbf{x}_9+\mathbf{x}_{10}+...+\mathbf{x}_{20})$$

$$=(7.67,7.33)^T$$

第四步’：因$Z_j(2) \neq Z_j(1), j=1,2, r=r+1=2,$转第二步″。

第二步”：重新计算$\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{20}$到$Z_1(2), Z_2(2)$的距离，

分别把$\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{20}$归于最近的那个聚类中心，

重新分为二类$G_1(3) = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_8)$

$G_2(3) = (\mathbf{x}_9, \mathbf{x}_{10}, ..., \mathbf{x}_{20}), N_1 = 8, N_2 = 12$

第三步”：更新聚类中心

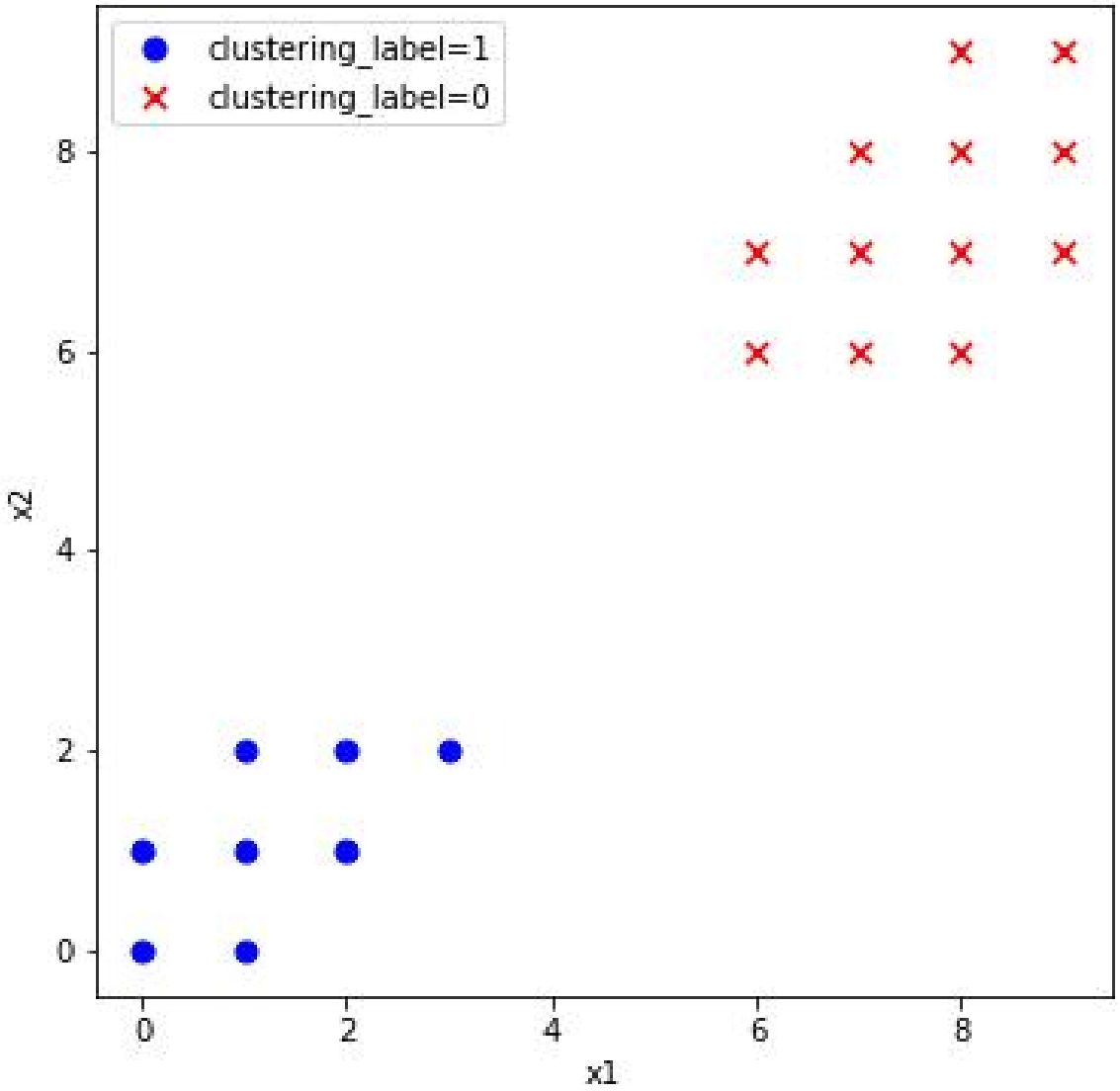
$Z_1(3) = Z_1(2) = (1.25, 1.13)^T$

$Z_2(3) = Z_2(2) = (7.67, 7.33)^T$

第四步”：

因$Z_j(3) = Z_j(2)(j=1,2)$，不再出现新的类别划分，故分类过程结束。

采用Python实现本例数据集的K均值聚类(K=2)：



K均值聚类算法Python核心代码：

In [1]:
```
#K均值聚类举例
#Filename:kmeans_example.ipynb
#Import Library
import numpy as np
from sklearn.cluster import KMeans
```

In [2]:
```
#Assumed you have X (features) or training data set and x_test(features) of test_dataset
X=np.array([[0,0], [1,0], [0,1], [1,1], [2,1], [1,2], [2,2], [3,2], [6,6], [7,6],
            [8,6], [6,7], [7,7], [8,7], [9,7], [7,8], [8,8], [9,8], [8,9], [9,9]])
x_test=np.array([[1,1.5], [8.5,8.5]])
```

In [3]:
```
# Create KMeans clusterer object model
k_means=KMeans(n_clusters=2, random_state=0)## default value for clusters is 3
# Train the model using the training sets and check score
k_means.fit(X)
clustering_label=k_means.predict(X)
print(clustering_label)
```

```
[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [4]:
```
#Predict Output
predicted=k_means.predict(x_test)
print(predicted)
```

```
[1 0]
```

10.5 DBSCAN密度聚类 (*：选学)

基于划分聚类和基于层次聚类的方法在聚类过程中根据距离来划分类簇，因此只能够用于挖掘球状簇。为了解决这一缺陷，基于密度聚类算法利用密度思想，将样本中的高密度区域(即样本点分布稠密的区域)划分为簇，将簇看作是样本空间中被稀疏区域 (噪声) 分隔开的稠密区域。这一算法的主要目的是过滤样本空间中的稀疏区域，获取稠密区域作为簇；

基于密度的聚类算法是根据密度而不是距离来计算样本相似度，所以基于密度的聚类算法能够用于挖掘任意形状的簇，并且能够有效过滤掉噪声样本对于聚类结果的影响；

常见的基于密度的聚类算法有DBSCAN(Density-Based Spatial Clustering of Applications with Noise, 具有噪声的基于密度的聚类方法)、OPTICS和DENCLUE等。其中，OPTICS 对DBSCAN算法进行了改进，降低了对输入参数的敏感程度。DENCLUE算法综合了基于划分、基于层次的方法。

本节仅介绍经典的DBSCAN密度聚类算法。

DBSCAN采用基于中心的密度定义，样本的密度通过核心对象在ϵ半径内的样本点个数(包括自身)来估计。DBSCAN算法基于领域来描述样本的密度，输入样本集$S=\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_m\}$和参数$(\varepsilon,MinPts)$刻画邻域的样本分布密度。其中，$\varepsilon$表示样本的邻域距离阈值，$MinPts$表示对于某一样本$p$，其$\epsilon$-邻域中样本个数的阈值。下面给出DBSCAN中的几个重要概念。

- ϵ-邻域：给定对象$\mathbf{x}_i$，在半径ε内的区域称为$\mathbf{x}_i$的ε-邻域。在该区域中，S的子样本集$N_\epsilon(\mathbf{x}_i)=\{\mathbf{x}_j\in S|distance(\mathbf{x}_i,\mathbf{x}_j)\leq\varepsilon\}$。
- 核心对象 (core object)：如果对象$\mathbf{x}_i\in S$，其ϵ-邻域对应的子样本集$N_\epsilon(\mathbf{x}_i)$至少包含$MinPts$个样本，$|N_\epsilon(\mathbf{x}_i)|\geq MinPts$，那么$\mathbf{x}_i$为核心对象。

- 直接密度可达(directly density-reachable)：对于对象$\mathbf{x}_i$和$\mathbf{x}_j$，如果$\mathbf{x}_i$是一个核心对象，且$\mathbf{x}_j$在$\mathbf{x}_i$的ε-邻域内，那么对象$\mathbf{x}_j$是从$\mathbf{x}_i$直接密度可达的。
- 密度可达(density-reachable)：对于对象$\mathbf{x}_i$和$\mathbf{x}_j$，若存在一个对象链$p_1, p_2, \ldots, p_n$，使得$p_1 = \mathbf{x}_i, p_n = \mathbf{x}_j$，并且对于$p_i \in S(1 \leq i \leq n)$，$p_{i+1}$从$p_i$关于$(\varepsilon, MinPts)$直接密度可达，那么$\mathbf{x}_j$是从$\mathbf{x}_i$密度可达的。
- 密度相连(density-connected)：对于对象$\mathbf{x}_i$和$\mathbf{x}_j$，若存在$\mathbf{x}_k$使得$\mathbf{x}_i$和$\mathbf{x}_j$是从$\mathbf{x}_k$关于$(\varepsilon, MinPts)$密度可达，那么$\mathbf{x}_i$和$\mathbf{x}_j$是密度相连的。

在下图中，若$MinPts = 3$，则**a**、**b**、**c**和**x**、**y**、**z**都是核心对象，因为在各自的ε-邻域中，都至少包含3个对象。对象**c**是从对象**b**直接密度可达的，对象**b**是从对象a直接密度可达的，则对象**c**是从对象**a**密度可达的。对象**y**是从对象**x**密度可达的，对象**z**是从对象**x**密度可达的，则对象**y**和**z**是密度相连的。

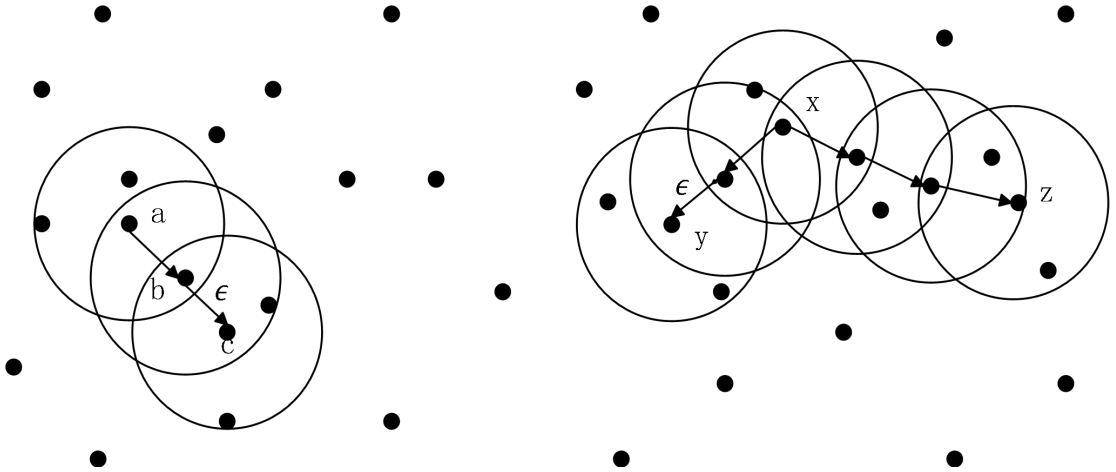


图 密度可达和密度相连

DBSCAN可以用于对任意形状的稠密数据集进行聚类，DBSCAN算法对输入顺序不敏感。DBSCAN能够在聚类的过程中发现数据集中的噪声点，且算法本身对噪声不敏感。当数据集分布为非球型时，使用DBSCAN算法效果较好。

DBSCAN算法要对数据集中的每个对象进行邻域检查，当数据集较大时， 聚类收敛时间长，需要较大的内存支持，I/O消耗也很大，此时可以采用KD树或球树对算法进行改进，快速搜索最近邻，帮助算法快速收敛。此外，当空间聚类的密度不均匀，聚类间距离相差很大时，聚类的质量较差。

DBSCAN算法的聚类结果受到邻域参数$(\varepsilon, MinPts)$的影响较大，不同的输入参数对聚类结果有很大的影响，邻域参数也需要人工输入，调参时需要对两个参数联合调参，比较复杂。

对于邻域参数选择导致算法聚类质量降低的情况，可从以下几个方面进行改进：

1. 对原始数据集抽取高密度点生成新的数据集，对新数据集进行聚类。在抽取高密度点生成新数据集的过程中，反复修改密度参数进行抽取，直到生成的新数据集可以很容易被聚类为止。以新数据集的结果为基础，将其他点归类到各个簇中，从而避免输入参数对于聚类结果的影响。

2. 采用核密度估计方法。采用核密度估计的思想对原始样本集进行非线性变换，使得到的新样本集中样本点的分布尽可能均匀，从而改善原始样本集中密度差异过大的情况。变换过后再使用全局参数进行聚类，得到较好的结果。

3. 并行化处理。对数据进行划分得到新的样本集，使得每个划分中的样本点分布相对均匀，根据每个新样本集中的样本分布密度来选择局部ε值。这样一方面降低了 全局ε参数对于聚类结果的影响，另一方面并行处理对多个划分进行聚类，在数据量较大的情况下提高了聚类效率，有效解决了DBSCAN算法对内存要求高的缺点。

DBSCAN程序示例：使用Sklearn库中的DBSCAN密度聚类算法实现聚类。

说明：

1. DBSCAN算法包含于sklearn.cluster库中；

2. 数据集用make_blobs方法随机生成，数量为750，有3个簇；

3. StandardScaler().fit_transformer()对数据进行预处理，以确保每个维度的均值为0、方差为1，使预测结果不会被某些维度过大的特征值而主导。

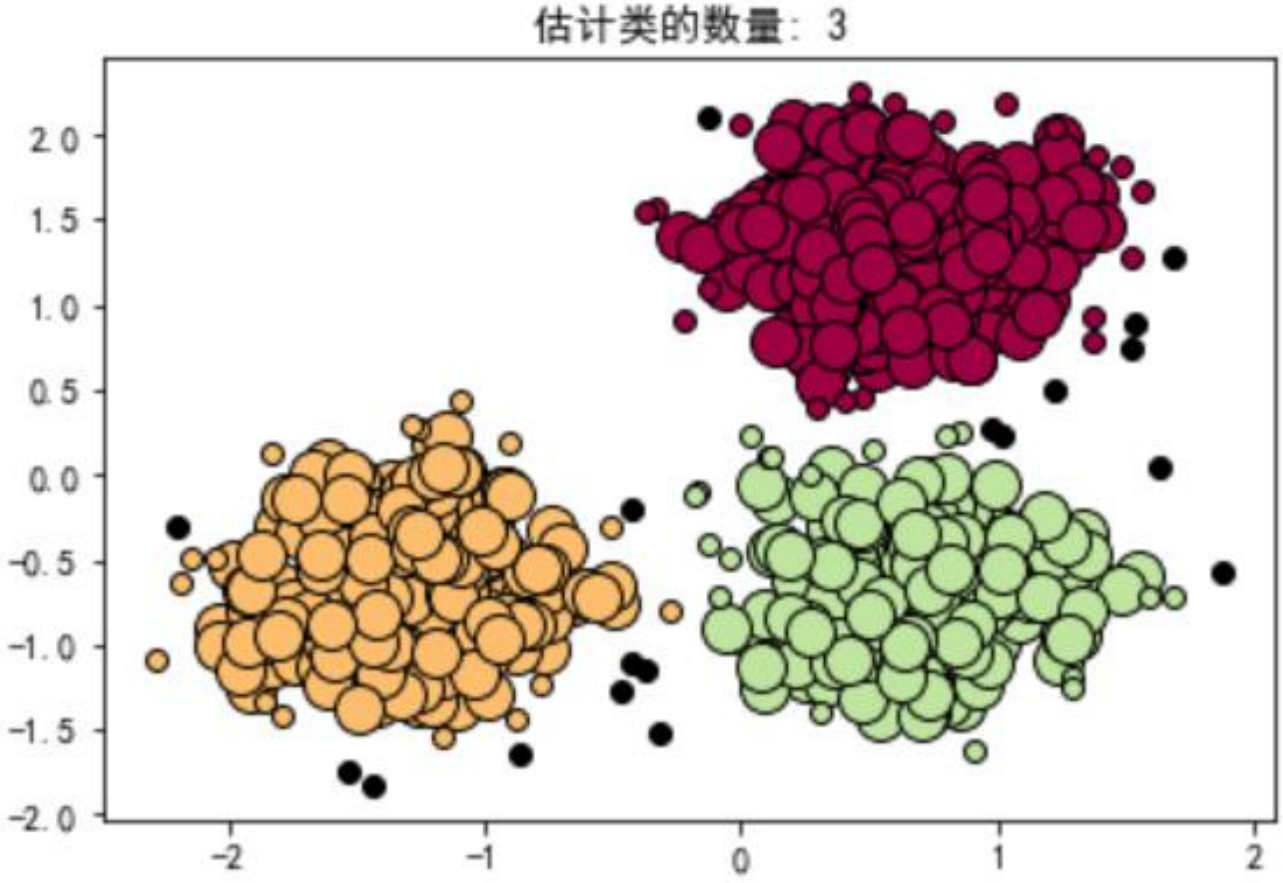
In [2]:

```
#Filename: DBSCAN_density_clustering.ipynb
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

# Generate sample data
centers = [[1, 1], [-1, -1], [1, -1]]
X, labels_true = make_blobs(n_samples=750, centers=centers, cluster_std=0.4, random_state=0)
X = StandardScaler().fit_transform(X)
# Compute DBSCAN
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
```

```
print('Estimated number of clusters: %d' % n_clusters_)
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels_true, labels))
print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels))
print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))
print("Adjusted Rand Index: %0.3f"
      % metrics.adjusted_rand_score(labels_true, labels))
print("Adjusted Mutual Information: %0.3f"
      % metrics.adjusted_mutual_info_score(labels_true, labels))
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(X, labels))

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
```

```
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

plt.title('估计类的数量: %d' % n_clusters_)
plt.show()
```

```
Estimated number of clusters: 3
Homogeneity: 0.953
Completeness: 0.883
V-measure: 0.917
Adjusted Rand Index: 0.952
Adjusted Mutual Information: 0.916
Silhouette Coefficient: 0.626
```



DBSCAN聚类效果

在本程序中，构建DBSCAN算法的参数有eps、min_samples。其中eps表示ε-邻域的距离阈值，其默认值为0.5；min_samples表示样本点要成为核心对象所需要的ε-邻域的样本数域值，默认值是5，一般在多组值中选一个最优值。

10.6 基于试探的聚类算法（*：自学）

确定一种聚类准则后, 每一种分类方法对应于一个J值, 其中使J取得极值的分类, 就被认为是最佳分类。 这种处理方式的计算量太大, 可以对某些关键性的参数进行试探性地选取, 使聚类准则函数达到最优, 这类方法可称为基于试探的聚类算法。

一、基于最近邻规则的试探法

设待分类样本集为$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, 按最近邻规则进行聚类, 算法具体步骤如下:

(1) 选取任意一个样本作为第一个聚类中心$\boldsymbol{z}_1$, 例如, 令ω_1类的中心$\boldsymbol{z}_1=\boldsymbol{x}_1$, 选定一非负的类内距离阈值$T$。

(2) 计算下一个样本$\boldsymbol{x}_2$到$\boldsymbol{z}_1$的距离d_{21}。 若$d_{21}>T$, 则建立一新的聚类中心$\boldsymbol{z}_2$, 且ω_2类的中心$\boldsymbol{z}_2=\boldsymbol{x}_2$; 若$d_{21}\leq T$, 则认为$\boldsymbol{x}_2$在以$\boldsymbol{z}_1$为中心的邻域中, 即$\boldsymbol{x}_2$属于$\omega_1$类。

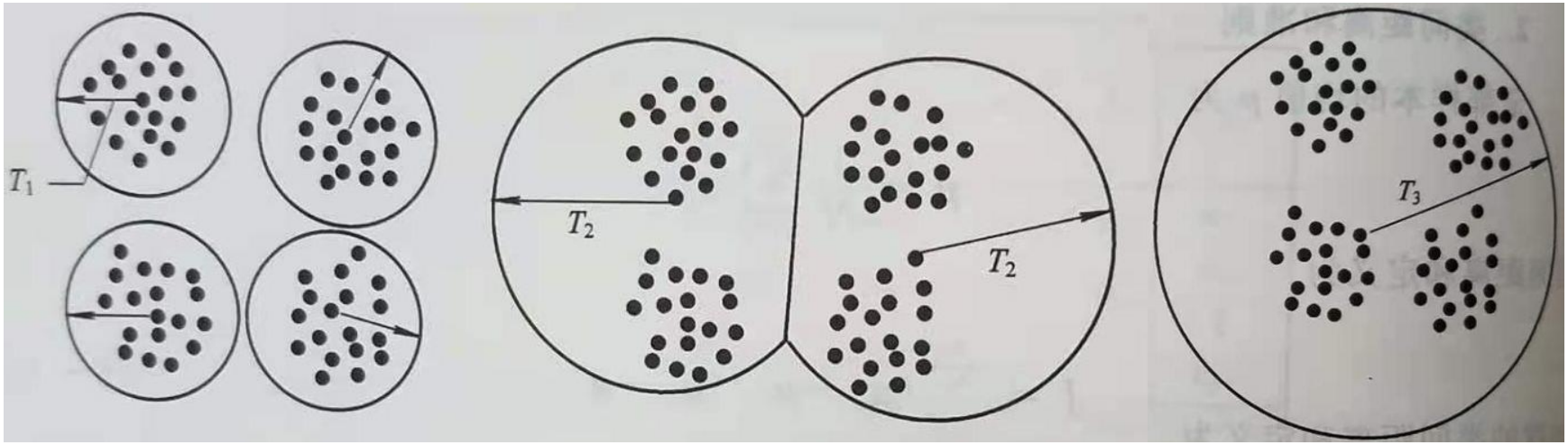
(3) 假定已有m类的中心$\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m$, 计算尚未确定类别的样本$\boldsymbol{x}_i$到$m$类中心$\boldsymbol{z}_j(j=1, 2, \ldots, m)$的距离$d_{ij}$。 如果所有的$d_{ij}$均满足$d_{ij}>T$, 则建立一新的聚类中心$\boldsymbol{z}_{m+1}$, 且$\omega_{m+1}$类的中心$\boldsymbol{z}_{m+1}=\boldsymbol{x}_i$; 否则将$\boldsymbol{x}_i$划分到距离其最近的聚类中心所代表的类中。

(4) 检查是否所有的样本已经确定类别, 如果都已确定类别, 则转下一步; 否则返回(3)。

(5) 按照某种聚类准则评价聚类结果, 如果不满意, 则重新选取第一个聚类中心和类间距离阈值T, 返回(2); 如果满意, 则算法结束。

这种方法的聚类结果与第一个聚类中心的选取、 待分类样本的排列次序、 阈值T的大小以及样本分布的几何特性有关。

下图表示了距离阈值T的大小和初始聚类中心位置对聚类结果的影响。



(a) (b) (c)

图 距离阈值和初始类心对聚类结果的影响

二、最大最小距离聚类算法

最大最小距离聚类算法是另一种简单的试探法，以类间距离最大作为选取新的聚类中心的条件，以最小距离原则进行样本归类。这种算法通常采用欧氏距离，算法具体步骤如下：

(1) 选取任意一个样本作为第一个聚类中心z_1。

(2) 选取距离z_1最远的样本作为第二个聚类中心z_2。

(3) 计算所有未作为聚类中心的样本$\boldsymbol{x}_i$与$\boldsymbol{z}_1$、$\boldsymbol{z}_2$的距离d_{i1},d_{i2}，并设

d_i=min(d_{i1},d_{i2})，若存在 $d_l = \max_i\{d_i\} > \theta \cdot d(\mathbf{z}_1, \mathbf{z}_2)$

则建立第三个聚类中心$\boldsymbol{z}_3$, 且$\boldsymbol{z}_3$=$\boldsymbol{x}_l$。其中：$d(\boldsymbol{z}_1, \boldsymbol{z}_2)$为$\boldsymbol{z}_1$与$\boldsymbol{z}_2$之间的距离；$\theta$

可用试探法取为一固定分数, 例如, 。

(4) 假定已有m个聚类中心$\boldsymbol{z}_1$, $\boldsymbol{z}_2$, …, $\boldsymbol{z}_m$, 计算尚未作为聚类中心的各样本$\boldsymbol{x}_i$

到m类中心$\boldsymbol{z}_j$(j=1, 2, …, m)的距离d_{ij}, 并计算

$$d_J = \min(d_{J1}, d_{J2}, \cdots, d_{Jm}) = \max_i\{\min(d_{i1}, d_{i2}, \cdots, d_{im})\}$$

如果$d_J>\theta\cdot d(\boldsymbol{z}_1, \boldsymbol{z}_2)$, 则建立第$m+1$个聚类中心$\boldsymbol{z}_{m+1}$, 且$\boldsymbol{z}_{m+1}=x_J$, 并转(4); 否则转下一步。

(5) 将全部样本按最近距离原则划分到各类中。

(6) 按照某种聚类准则评价聚类结果, 如果不满意, 则重新选取第一个聚类中心和θ值, 返回(2); 如果满意, 则算法结束。

举例：10个样本数据如图所示, 用最大最小距离聚类算法对样本集分类。

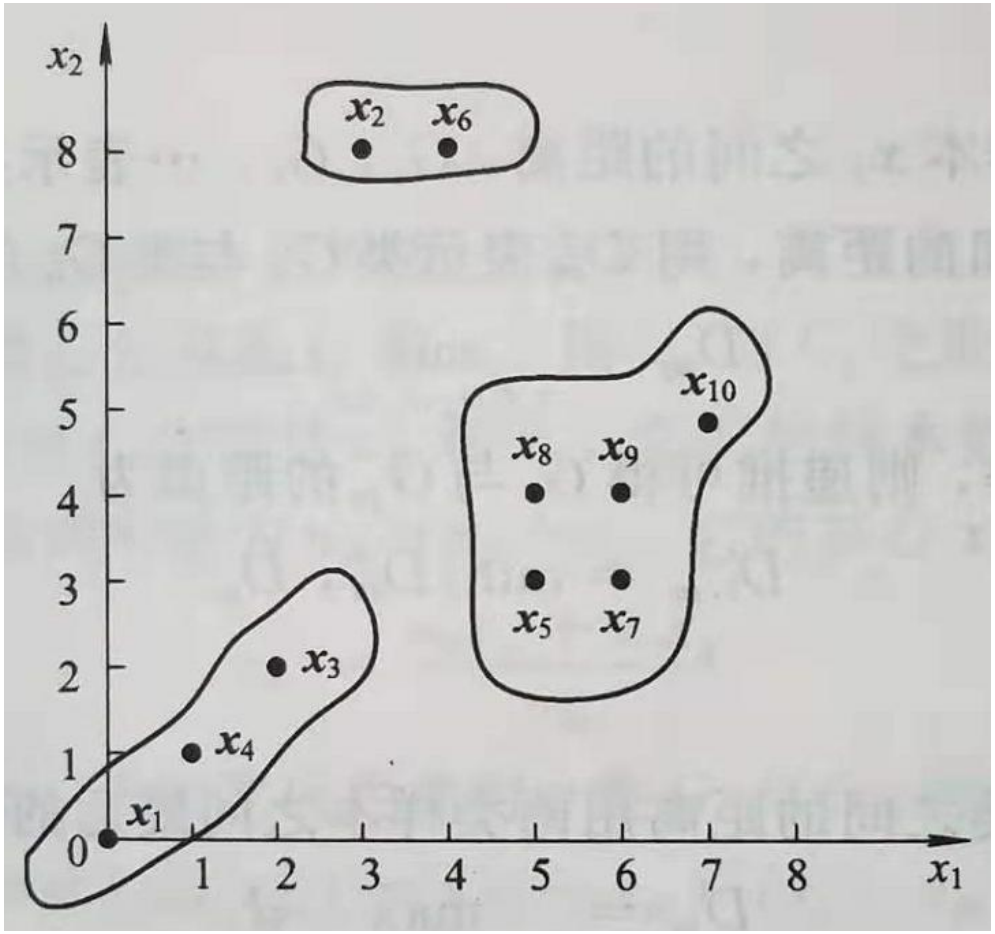


图　最大最小距离聚类算法举例

解：

(1) 给定θ=1/2，选取任意一个样本作为第一个聚类中心$\boldsymbol{z}_1$，令$\boldsymbol{z}_1=\boldsymbol{x}_1$。

(2) 计算其他样本到$\boldsymbol{z}_1$的距离，可得$\boldsymbol{x}_6$距离$\boldsymbol{z}_1$最远，则建立第二个聚类中心$\boldsymbol{z}_2=\boldsymbol{x}_6$。

(3) 计算所有未作为聚类中心的样本与$\boldsymbol{z}_1$、$\boldsymbol{z}_2$的距离，根据最大最小距离聚类算法，得到第三个聚类中心$\boldsymbol{z}_3=\boldsymbol{x}_7$。

(4) 判断出不再有新的聚类中心，将全部样本按最近距离原则划分到各类中，得到$\{\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_4\}$为第一类，$\{\boldsymbol{x}_2, \boldsymbol{x}_6\}$为第二类，$\{\boldsymbol{x}_5, \boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_9, \boldsymbol{x}_{10}\}$为第三类。

(5) 通过直观判断，认为本次分类结果比较合理，对样本集的分类完成。

10.7 聚类分析Python程序举例

一、层次聚类Python程序示例

1. 准备数据：生成一“坨”没有类别的数据点，并绘制其散点图

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
#导入数据集生成工具
from sklearn.datasets import make_blobs
#生成分类数据为1的数据集
blobs = make_blobs(random_state=1,centers=1)
X_blobs = blobs[0]
#绘制散点图
plt.scatter(X_blobs[:,0],X_blobs[:,1],c='r',edgecolor='k')
#显示图像
plt.show()
```

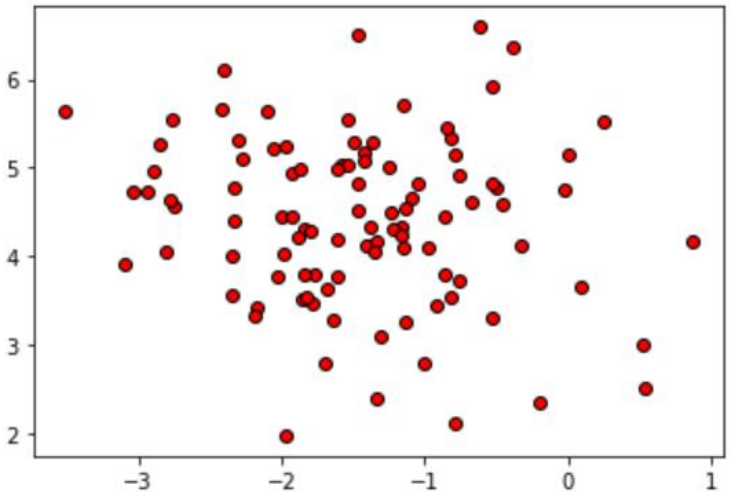


图1 使用make_blobs生成的无分类数据

2.导入dendrogram和ward工具，显示聚类树

In [2]:
```
# 导入dendrogram和ward工具
from scipy.cluster.hierarchy import dendrogram, ward
# 使用连线方式进行可视化
linkage = ward(X_blobs)
dendrogram(linkage)
ax = plt.gca()
plt.xlabel("Sample index")  #样本指数
plt.ylabel("Cluster distance") #距离
plt.show()
```

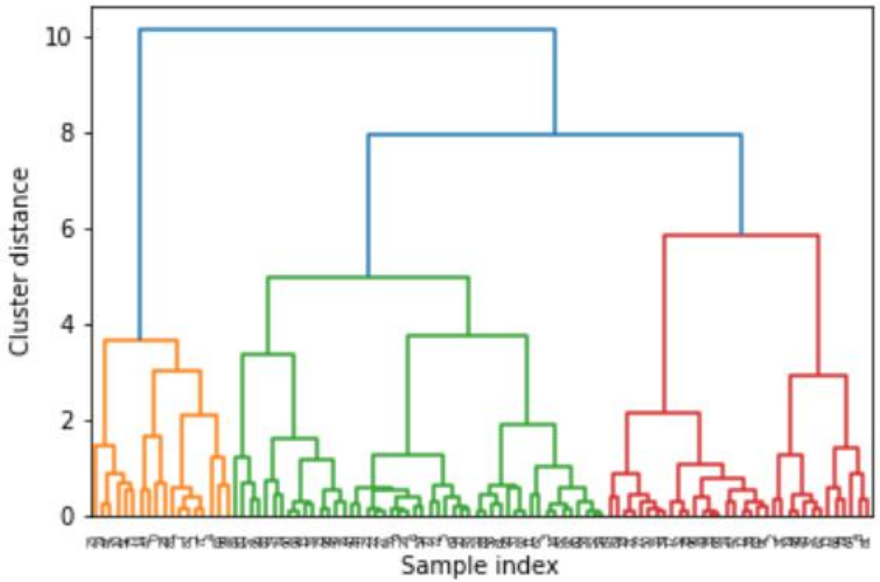


图2　凝聚聚类算法工作原理

由图2可知，自底向上的凝聚层次聚类方法是不断地合并相似的聚类中心，使类别越来越少，同时每个聚类中心的距离越来越远。

二、K均值聚类Python程序示例

In [3]:

```
#导入KMeans工具
from sklearn.cluster import KMeans
#要求将KMeans将数据聚为3类
kmeans = KMeans(n_clusters=3)
#拟合数据
kmeans.fit(X_blobs)

#下面是用来画图的代码
x_min, x_max = X_blobs[:, 0].min()-0.5 , X_blobs[:, 0].max()+0.5
y_min, y_max = X_blobs[:, 1].min()-0.5 , X_blobs[:, 1].max()+0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
                     np.arange(y_min, y_max, .02))


Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.summer,
           aspect='auto', origin='lower')
```

```
plt.plot(X_blobs[:, 0], X_blobs[:, 1], 'r.', markersize=5)
#用蓝色叉号代表聚类的中心
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=150, linewidths=3,
            color='b', zorder=10)

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

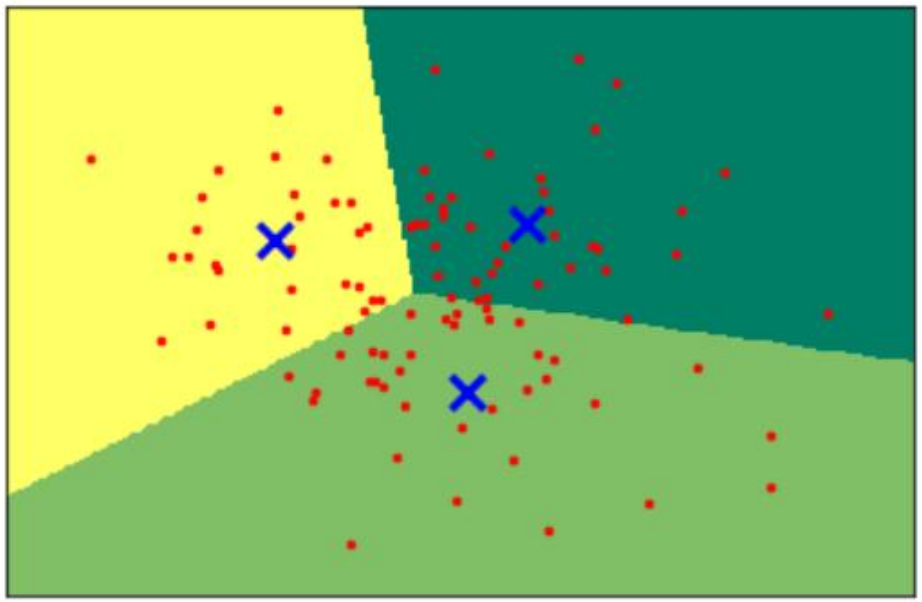


图3 使用K均值算法进行的聚类

In [4]:
```
print("K均值的聚类标签:\n{}".format(kmeans.labels_))
```

```
K均值的聚类标签:
[2 2 1 0 0 0 2 2 1 0 2 0 2 1 2 0 0 2 1 1 0 1 2 2 2 2 0 2 2 2 1 1 2 2 0 1 0
 1 2 1 0 2 1 1 0 0 0 2 1 2 1 2 0 1 0 2 1 0 0 2 0 1 0 2 1 0 1 1 2 0 0 2 0 0
 0 2 0 2 2 1 0 1 0 0 1 2 0 2 1 1 0 2 1 1 0 0 2 0 0 2]
```

从以上结果可以看出，*K*均值对数据进行聚类和分类有些类似，是用0、1、2三个数字来代表数据的类别，并存储在.labels_属性中。

三、DBSCAN密度聚类Python程序示例(*: 选学)

In [5]:
```
# 导入DBSCAN
from sklearn.cluster import DBSCAN
db = DBSCAN()
# 使用DBSCAN拟合数据
clusters = db.fit_predict(X_blobs)
# 绘制散点图
plt.scatter(X_blobs[:, 0], X_blobs[:, 1], c=clusters, cmap=plt.cm.cool,
            s=60,edgecolor='k')
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.show()
```

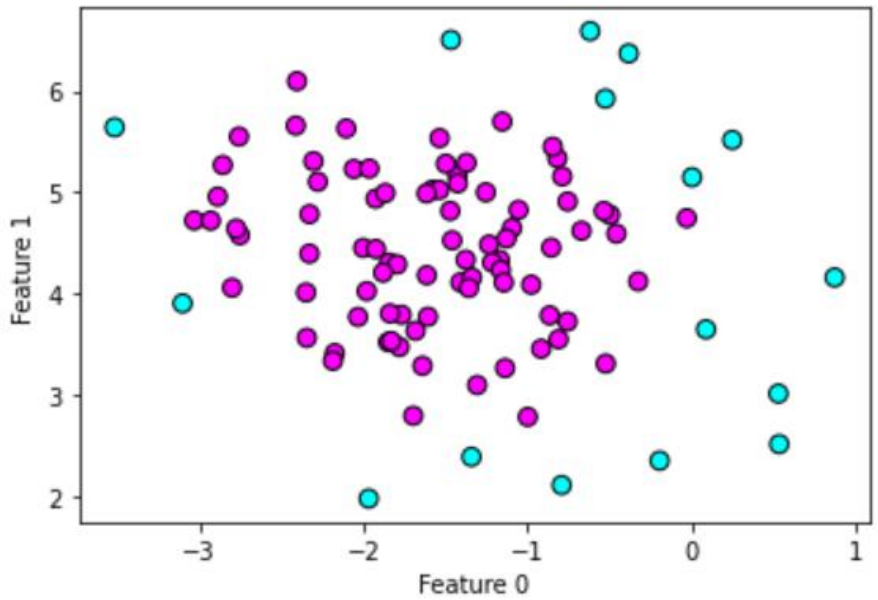


图4 DBSCAN算法对make_blobs数据集的聚类结果

结果分析：从图4可以看出，经过DBSCAN的聚类，样本点被标成了不同的深浅程度，那么是不是表示DBSCAN将数据集聚成了两类呢?

In [6]:
```
print('聚类标签为：\n{}'.format(clusters))
```

```
聚类标签为：
[-1  0 -1  0 -1  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0
  0 -1  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1
  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0 -1  0  0  0  0  0  0  0
  0  0 -1 -1  0  0  0  0 -1  0  0 -1  0  0 -1  0  0  0  0  0  0  0  0 -1
  0  0  0 -1]
```

结果分析：聚类标签出现了-1；在DBSCAN中，-1代表该数据点为噪声；在图4中，我们看到中间深色的样本点密度相对较大，因此DBSCAN将它们归到一“坨”，而外围的浅色的样本点，DBSCAN认为根本不属于任何一类，所以放进了“噪声”类别中。

DBSCAN中两个重要参数：

1. eps-指定的是考虑划分到同一“坨”的样本距离有多远，eps值设置得越大，则聚类所覆盖的样本点越多，反之则越少，eps默认值为0.5；
2. min_samples-指定在某个样本点周围，被当作聚类核心点的个数，min_samples值越大，则核心点越少，噪声就越多，反之min_samples值越小，噪声也就越小，该参数默认值为2。

In [7]:
```
#eps默认值为0.5，现将eps调到2，即调大一些，看看会发生什么？
db_1 = DBSCAN(eps = 2)
clusters_1 = db_1.fit_predict(X_blobs)
plt.scatter(X_blobs[:, 0], X_blobs[:, 1], c=clusters_1, cmap=plt.cm.cool,
            s=60,edgecolor='k')
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.show()
```

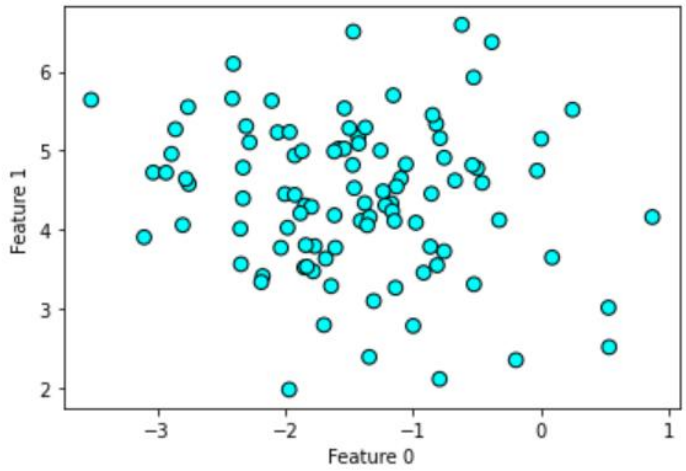


图5 eps值设置为2时的DBSCAN聚类

结果分析：从图5可以看出，所有样本点都变成了浅色，这并不是说所有样本点都变成了噪声，而是说所有样本点都被归入同一“坨”中；这是因为增大eps取值后，使DBSCAN将距离更远的样本点也拉到该聚类中了。

In [8]:
```
clusters_1
```

Out[8]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [9]:
```
db_2 = DBSCAN(min_samples=20)
clusters_2 = db_2.fit_predict(X_blobs)
plt.scatter(X_blobs[:, 0], X_blobs[:, 1], c=clusters_2, cmap=plt.cm.cool,
            s=60,edgecolor='k')
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.show()
```

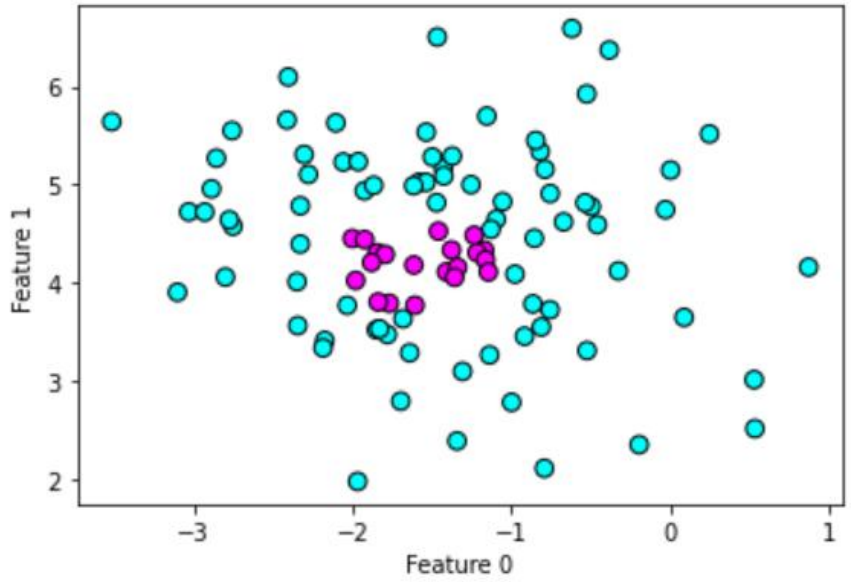


图6 min_samples值设置为20时的DBSCAN聚类

结果分析：将图6与图4进行对比，我们会发现浅色的样本点变多了，也就是噪声变多了；而深色的样本点，也就是聚类中被划分到类别1中的样本点变少了。

In [10]:
```
clusters_2
```

Out[10]:
```
array([-1, -1, -1, -1, -1, -1, -1, -1, -1,  0, -1, -1,  0, -1, -1, -1, -1,
       -1,  0, -1, -1,  0,  0,  0, -1, -1,  0,  0, -1, -1, -1, -1, -1, -1,
       -1,  0, -1, -1,  0,  0,  0, -1,  0, -1,  0, -1, -1, -1, -1, -1,  0,
       -1, -1, -1, -1,  0, -1, -1, -1,  0, -1, -1, -1, -1, -1, -1, -1, -1,
       -1,  0, -1, -1, -1, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1, -1, -1, -1],
      dtype=int64)
```

综上，虽然DBSCAN并不需要我们在开始训练算法时就指定clusters的数目，但是通过设置eps和min_samples参数，相当于间接地指定了clusters的数目。尤其是eps参数极为重要，因为它规定了某一“坨”的范围大小。而且在实际应用中，若将数据集先用MinMaxScaler或者StandardScaler进行预处理，那么DBSCAN算法的表现会更好(因为这两种预处理方法把数据的范围控制得较为集中)。

本章小结：

1. 层次聚类/系统聚类 (基于层次的方法-凝聚方法)

基本思想是将距离阈值(distance threshold value)作为决定聚类数目的标准，基本思路是每个样本先自成一类，然后按距离准则逐步合并，减少类别数，直到达到分类要求。

2. 分解聚类 (基于层次的方法-分裂方法)

先将全部样本看成一类，然后按照某种准则将其分解成二类、三类，直至每个样本自成一类为止。

3. *K*均值聚类

*K*均值聚类是最著名的基于划分的聚类算法。*K*均值聚类算法工作原理：1)任选*K*个样本点作为初始聚类中心；2)将每个样本点划分给距离最近的中心，衡量两个样本点的距离有多种不同的方法，常用的是欧氏距离；3)重新计算每个簇的中心作为新的聚类中心，使其总的平方距离达到最小；4)重复第2)步和第3)步，直到收敛。

4. DBSCAN密度聚类 (*: 选学)

5. 基于试探的聚类算法

最近邻规则聚类法(近邻聚类法)，最大最小距离聚类算法。

6. 聚类分析Python编程

掌握层次聚类、*K*均值聚类Python编程。

本章主要参考文献：

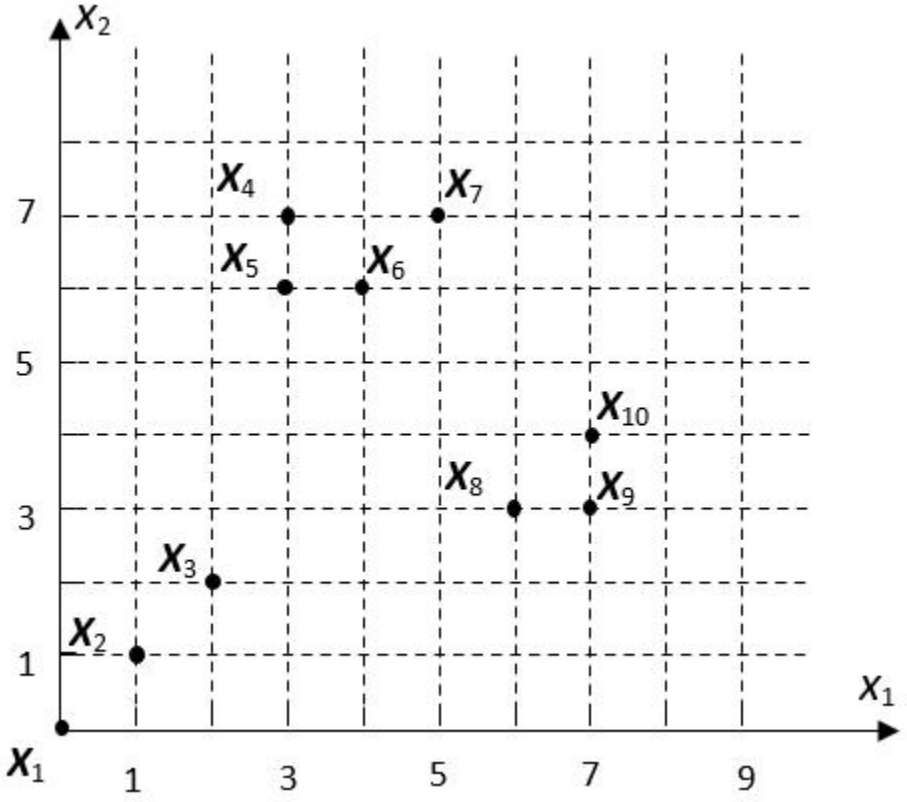
[01] 李弼程等. **模式识别原理与应用**: 第6章，西电版，2008

[02] 齐敏等. **模式识别导论**: 第2章，清华版，2009

课外作业题10 (聚类分析):

1.简述K均值聚类方法的优缺点。查阅资料，谈一谈可以从哪些方面对K均值进行改进。

2.设有10个二维模式样本，如图所示。 若θ=1/2，试用最大最小距离算法对该样本集进行聚类分析。按该聚类算法的步骤，详细给出各个中间结果。



(最大最小距离算法及举例:
1)课件10.6
2)李弼程.模式识别原理与应用-P106-107)
3)齐敏.模式识别导论-P21-23及例2.1
)

3. 设有五个六维样本： $\mathbf{x}_1=(0,1,3,1,3,4)^T$, $\mathbf{x}_2=(3,3,3,1,2,1)^T$,

$\mathbf{x}_3=(1,0,0,0,1,1)^T$, $\mathbf{x}_4=(2,1,0,2,2,1)^T$, $\mathbf{x}_5=(0,0,1,0,1,0)^T$,

试用最短距离法进行层次聚类 (系统聚类) 分析。

4. 设有样本集X={$(0,0)^T$,$(0,1)^T$, $(1,0)^T$, $(4,4)^T$, $(4,5)^T$, $(5,4)^T$, $(5,5)^T$}， 试用*K*均值算法对样本集进行聚类。(*: 选做)

课外单元编程作业3：用*K*均值算法对鸢尾花(Iris)数据集进行聚类，取*K*=3 。

(Iris数据集下载：http://archive.ics.uci.edu/ml/machine-learning-databases/iris/；点击[iris.data]下载)

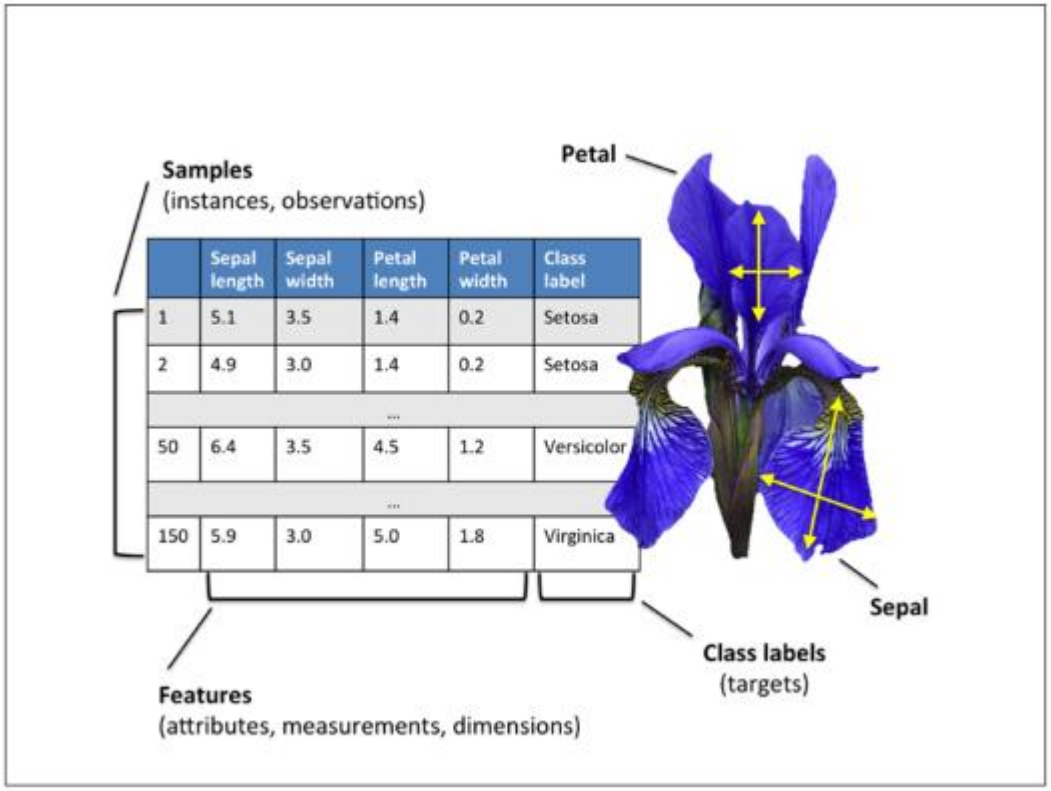
(提示：

1. Python+Sklearn Iris数据集可不用下载，直接用下面代码导入：

```
from sklearn import datasets
X = datasets.load_iris().data
y = datasets.load_iris().target
```

2. 本编程作业可用Python/R/MATLAB等语言编程完成)

鸢尾花(Iris)数据集说明：



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Iris数据集是常用的分类实验数据集，Iris也称鸢尾花卉数据集，由统计学家Fisher于1936年收集整理。数据集包含150个数据集，分为3类，每类50个样本数据，每个样本数据包含4个特征/属性。可通过萼片(Sepal)长度、萼片宽度、花瓣(Petal)长度、花瓣宽度4个特征预测鸢尾花卉属于（Setosa山鸢尾，Versicolour杂色鸢尾，Virginica维吉尼亚鸢尾）三个种类中的哪一类。数据集由3种不同类型的鸢尾花的50个样本数据构成。其中，第一个种类与另外两个种类线性可分，后两个种类非线性可分。

课外作业10 (聚类分析)参考解答:

1.答:

——优点：概念清楚。简单，易实现。

——缺点：对超参数K值与初始中心点敏感；对噪声点和异常点敏感。

——改进：略。书上，网上很多。

2.解:

① 取 $\boldsymbol{Z}_1 = \boldsymbol{X}_1 = [0,0]^{\mathrm{T}}$。

② 选离 $\boldsymbol{Z}_1$ 最远的样本作为第二聚类中心 $\boldsymbol{Z}_2$。

$$D_{21} = \sqrt{(1-0)^2 + (1-0)^2} = \sqrt{2}\,,\quad D_{31} = \sqrt{8}\,,\quad D_{41} = \sqrt{58}\,,\quad D_{51} = \sqrt{45}$$

$$D_{61} = \sqrt{52}\,,\quad D_{71} = \sqrt{74}\,,\quad D_{81} = \sqrt{45}\,,\quad D_{91} = \sqrt{58}\,,\quad D_{10,1} = \sqrt{65}$$

$\because$ 最大者为 D_{71}，$\therefore\ \boldsymbol{Z}_2 = \boldsymbol{X}_7 = [5,7]^{\mathrm{T}}$

$$T = \theta\|\boldsymbol{Z}_1 - \boldsymbol{Z}_2\| = \frac{1}{2}\sqrt{74}$$

③ 计算各样本与$\{\boldsymbol{Z}_1,\boldsymbol{Z}_2\}$间距离，选出其中的最小距离。

$$D_{12}=\sqrt{74}\ ,\quad D_{22}=\sqrt{52}\ ,\quad D_{32}=\sqrt{34}\ ,\quad \cdots,\quad D_{10,2}=\sqrt{13}$$

$$\min(D_{i1},D_{i2})=\{0,\sqrt{2},\sqrt{8},\sqrt{4},\sqrt{5},\sqrt{2},0,\sqrt{17},\sqrt{20},\sqrt{13}\}$$

④ $\max\{\min(\ D_{i1},D_{i2})\}=\sqrt{20}=D_{92}>T=\dfrac{1}{2}\sqrt{74}\ ,\ \therefore\quad \boldsymbol{Z}_3=\boldsymbol{X}_9=[7,3]^{\mathrm{T}}$

⑤ 继续判断是否有新的聚类中心出现：

$$\begin{cases}D_{11}=0\\D_{12}=\sqrt{74}\ ,\\D_{13}=\sqrt{58}\end{cases}\quad\begin{cases}D_{21}=\sqrt{2}\\D_{22}=\sqrt{52}\ ,\\D_{23}=\sqrt{40}\end{cases}\quad\cdots\begin{cases}D_{10,1}=\sqrt{65}\\D_{10,2}=\sqrt{13}\\D_{10,3}=\sqrt{1}\end{cases}$$

$$\min(D_{i1},D_{i2},D_{i3})=\{0,\sqrt{2},\sqrt{8},\sqrt{4},\sqrt{5},\sqrt{2},0,1,0,1\}$$

$$\max\{\min(\ D_{i1},D_{i2},D_{i3})\}=\sqrt{8}=D_{31}<T=\frac{1}{2}\sqrt{74}$$

寻找聚类中心的步骤结束。

⑥ 按最近距离分到三个聚类中心对应的类别中：

$$\omega_1:\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{X}_3;\quad \omega_2:\boldsymbol{X}_4,\boldsymbol{X}_5,\boldsymbol{X}_6,\boldsymbol{X}_7;\quad \omega_3:\boldsymbol{X}_8,\boldsymbol{X}_9,\boldsymbol{X}_{10}$$

3. 解：

1）将每个样本看作单独一类，全部样本分为5类，得

$$G_1(0)=\{\mathbf{x}_1\}, G_2(0)=\{\mathbf{x}_2\}, G_3(0)=\{\mathbf{x}_3\}, G_4(0)=\{\mathbf{x}_4\},$$
$$G_5(0)=\{\mathbf{x}_5\}$$

2）作距离矩阵D(0)，计算各类间欧氏距离，见下表

D(0)	$G_1(0)$	$G_2(0)$	$G_3(0)$	$G_4(0)$	$G_5(0)$
$G_1(0)$	0				
$G_2(0)$	$\sqrt{23}$	0			
$G_3(0)$	5	$\sqrt{24}$	0		
$G_4(0)$	$\sqrt{24}$	$\sqrt{15}$	$\sqrt{7}$	0	
$G_5(0)$	$\sqrt{26}$	5	$\sqrt{3}$	$\sqrt{12}$	0

3）将最小距离$\sqrt{3}$ 对应的类$G_3(0)$和$G_5(0)$合并为一类，得到新的聚类

$G_6(1)=\{G_3(0), G_5(0)\}$, $G_1(1)=\{G_1(0)\}$, $G_2(1)=\{G_2(0)\}$, $G_4(1)=\{G_4(0)\}$。按最小距离准则计算类间距，由D(0)矩阵递推得到聚类后的D(1)距离矩阵为：

D(1)	$G_6(1)$	$G_1(1)$	$G_2(1)$	$G_4(1)$
$G_6(1)$	0			
$G_1(1)$	5	0		
$G_2(1)$	$\sqrt{24}$	$\sqrt{23}$	0	
$G_4(1)$	$\sqrt{7}$	$\sqrt{24}$	$\sqrt{15}$	0

4）同样将最小距离$\sqrt{7}$对应的类$G_2(1)$和$G_4(1)$合并为一类，得到新的聚类

$G_7(2)=\{G_6(1), G_4(1)\}, G_1(2)=\{G_1(1)\}, G_2(2)=\{G_2(1)\}$ 。

若给定的阈值为$T=\sqrt{5}$，D(1)中的最小元素为$\sqrt{7}>T$，聚类结束，五个六维样本聚成三类，聚类结果为

$G_7=\{x_3, x_4, x_5\}, G_1=\{x_1\}, G_2=\{x_2\}$

若无阈值条件，继续聚类下去(如当聚成二类时，G_7和G_2又可合并为新的一类$G_8=\{x_2, x_3, x_4, x_5\}$，$G_1=\{x_1\}$为第二类)，直到最终全部样本归为一类，这时可画出聚类过程的聚类树（此略）。

End of this lecture.

Thanks!