第五章 聚类

5.1 聚类基本概念与评估

5.2 k均值聚类

5.3 层次聚类

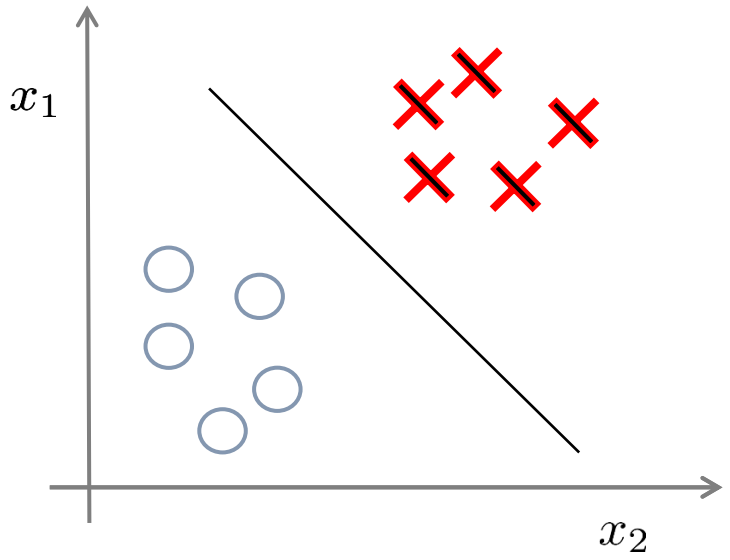
5.4 密度聚类*

监督学习 vs 无监督学习

分类：分类模型训练需要标记好的数据，是监督学习。

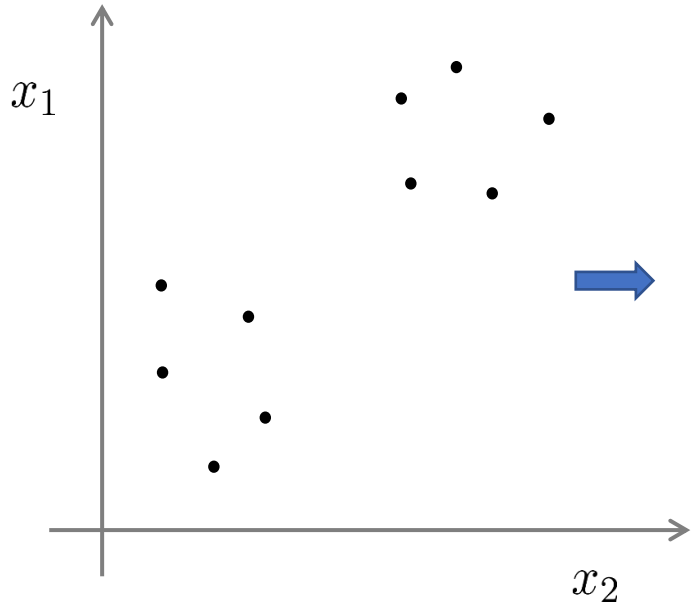
$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots (\boldsymbol{x}_n, y_n)\}$

标签哪里来？人工给出，成本高。



$\{\boldsymbol{x}_1, \boldsymbol{x}_2 \cdots \boldsymbol{x}_n\}$

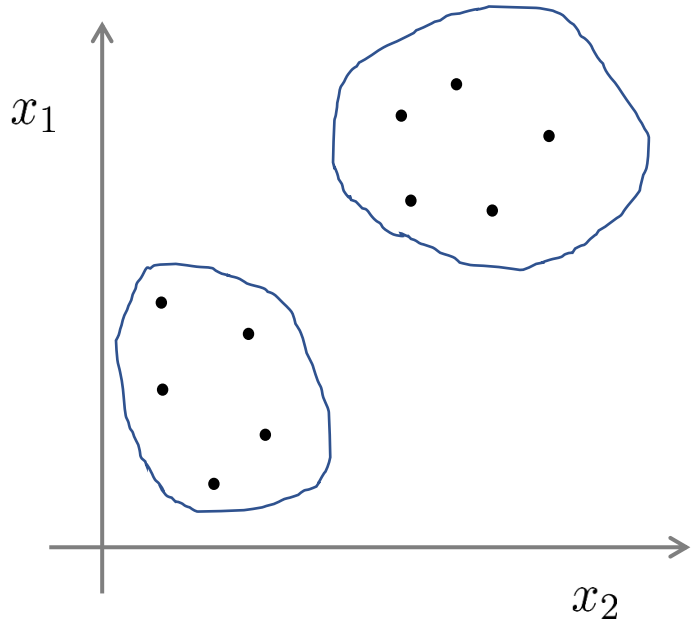
没有标签我们能做什么分析？



聚类分析：最常用的无监督学习方法。

聚类分析

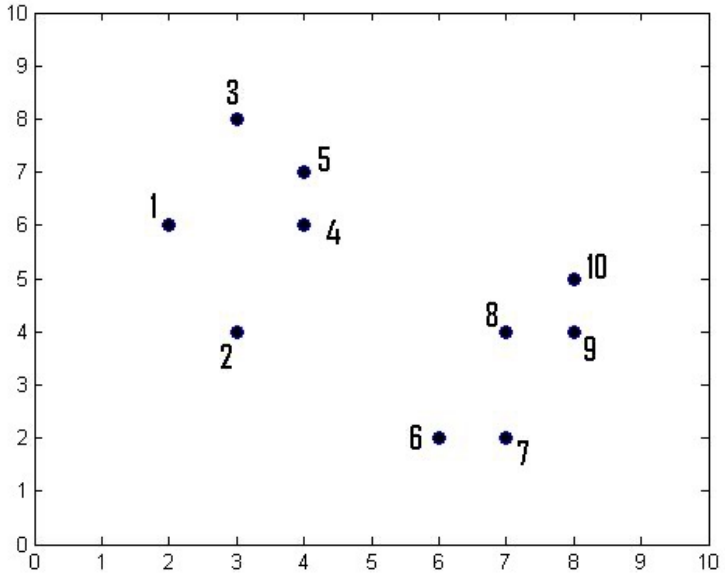
- 对数据集中的样本进行分组，使得同一个组内样本之间的相似度大于不同组样本之间的相似度。组又叫“簇”（cluster）。



- 用于探索数据集样本之间的簇结构，不需要标签信息，人工成本低。
- 做为其他方法的前驱处理，比如先聚类，然后基于聚类结果做其他处理。

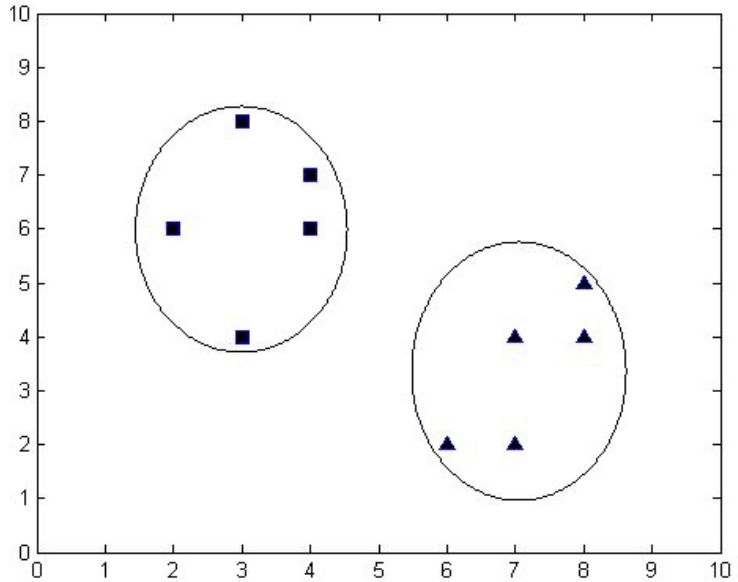
聚类方法：划分 vs 层次

划分聚类（partitioning clustering）一般得到若干个互斥的簇。



待聚类的数据集χ

划分聚类



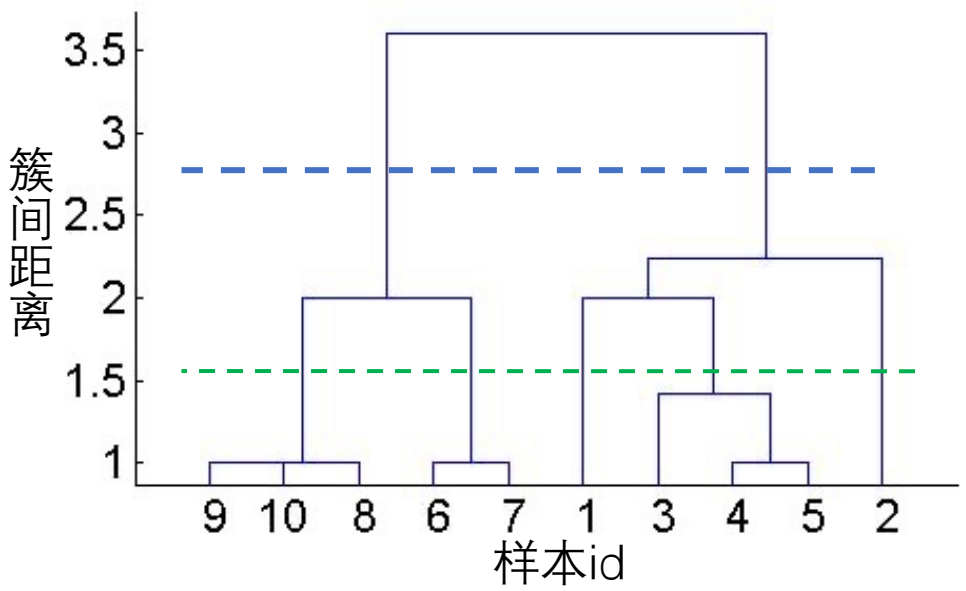
上图：

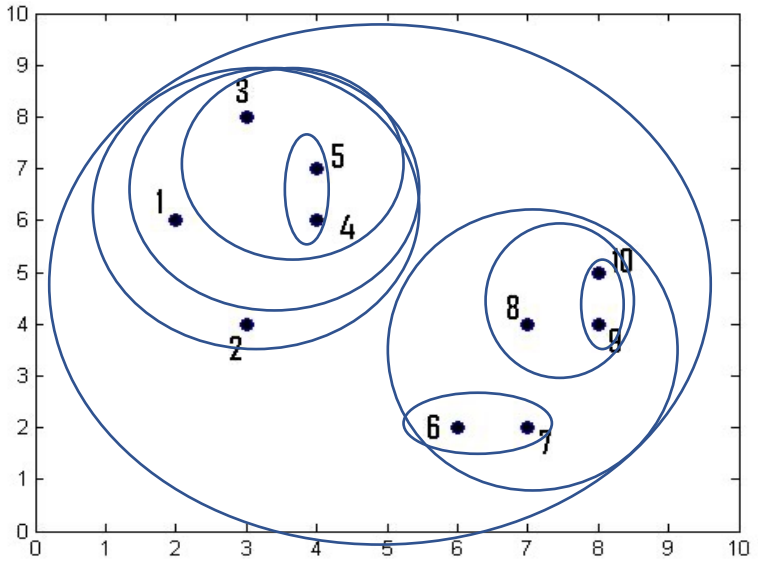
$\chi = C_1 \cup C_2$, 且 $C_1 \cap C_2 = \emptyset$;

每个样本属于且只属于一个簇；

聚类方法：划分 vs 层次

层次聚类（hierarchical clustering）得到树状结构(dendrogram)表示的一系列相互嵌套簇。





相互嵌套的簇

设定不同簇距离阈值，得到多种划分结果。

聚类结果评估

聚类结果的评估用来选择参数以及算法。根据评估时是否利用类别标签，把评估方法分为外部度量和内部度量。

- 外部度量：对比聚类算法产生的簇和已知样本类别。

注意：与分类时把类标签用于训练模型不一样，类标签只用于聚类结果评估而不用于聚类过程。

- 内部度量：对比衡量聚类结果与数据集自身分布情况。

聚类评估：外部度量

把样本的已知类别当作真实分组(ground truth)，即同一个标签的为一个类。

假设一个包含10个样本的数据集的真实分类与聚类结果如下，如何衡量两者之间的相近程度？

真实分类：{1，2，3}、{4，5，6}、{7，8，9，10}

算法得到簇：{1，2，4，7，8}、{3，9，10}、{5，6}

真实 \ 预测	k=1 {1，2，4，7，8}	k=2 {3，9，10}	k=3 {5，6}	n_c
c=1 {1，2，3}	2	1	0	**3**
c=2 {4，5，6}	1	0	2	**3**
c=3 {7，8，9，10}	2	2	0	**4**
n_k	**5**	**3**	**2**	

混淆矩阵（ n_k^c ）

注意：簇的个数K与类的个数C不一定相同。

聚类评估：外部度量

方法一：F值

$$Fmeasure(c,k)=\frac{(1+\beta)P(c,k)\times R(c,k)}{\beta P(c,k)+R(c,k)}$$

$$Fmeasure=\sum_{c=1}^{C}\frac{n_c}{n}\operatorname{argmax}_k\{Fmeasure(c,k)\}$$

$$P(c,k)=\frac{n_k^c}{n_k}\quad,\quad \mathrm{R}(c,k)=\frac{n_k^c}{n_c}$$

F值的定义与分类时有什么异同点？

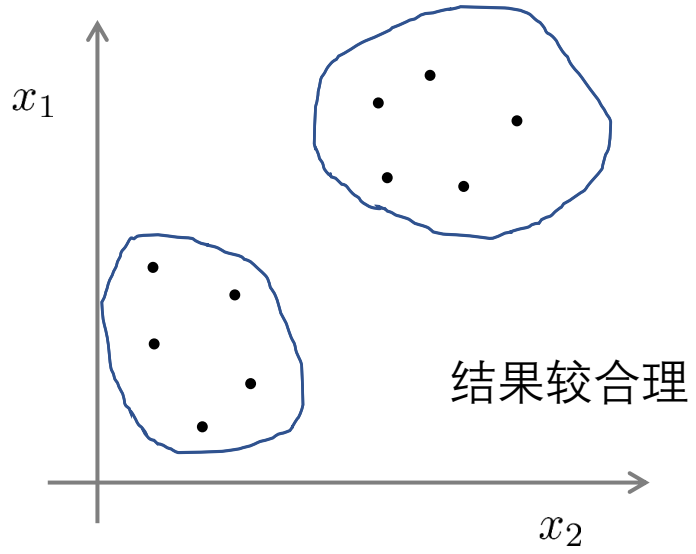
方法二：归一化互信息（Normalized mutual information）

$$NMI=\frac{\sum_{k=1}^{K}\sum_{c=1}^{C}n_k^c log\frac{n\cdot n_k^c}{n_k n_c}}{\sqrt{(\sum_{k=1}^{K}n_k log\frac{n_k}{n})(\sum_{c=1}^{C}n_c log\frac{n_c}{n})}}$$

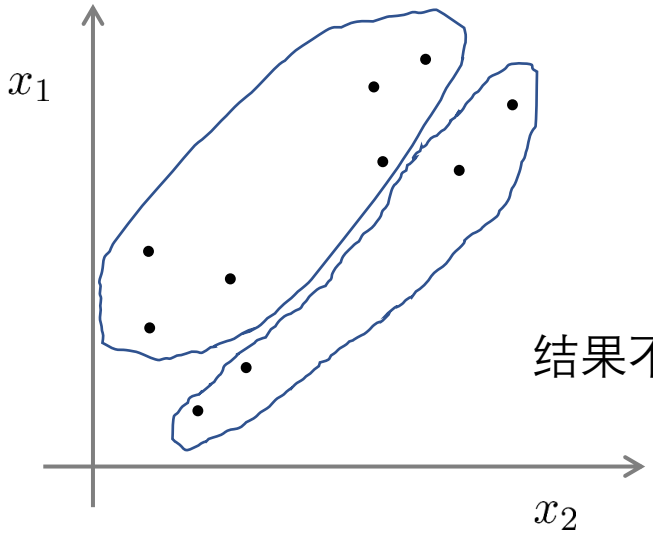
n是整个数据集的大小，n_k和n_c分别是第k个簇和第c个类的大小，n_k^c是第k个簇和第c个类共同包含的样本个数。在聚类结果接近于真实类别的时候，以上两种度量的值都接近于1。

聚类评估：内部度量

内部评估度量：对比聚类结果与数据集自身分布情况



结果较合理



结果不合理

同一个簇内部样本之间的相似度大于属于不同簇的样本之间的相似度

聚类评估：内部度量

- 簇内紧凑度：同一簇内样本之间(intra-cluster)的距离/相似度
- 簇间分离度：不同簇的样本之间(inter-cluster)的距离/相似度

评价标准：紧凑度越高，分离度越大，则聚类结果越好

假设数据集$\mathcal{X}$被分成k个簇, $\mathcal{X} = C_1 \cup C_2 \ldots \cup C_k$, 且对于c $\neq f, C_c \cap C_f = \emptyset$

- 内部度量：轮廓系数(s)

假设$x_i \in C_c$, 则

$$d_i^{intra} = \frac{1}{|C_c|-1}\sum_{x_j \in C_c, j \neq i} dist_{ij}, \quad d_i^{inter} = \min_{f=1,2\ldots k, f \neq c}\left\{\frac{1}{|C_f|}\sum_{x_j \in C_f} dist_{ij}\right\}$$

样本x_i的轮廓系数为 $s_i = \frac{d_i^{inter} - d_i^{intra}}{\max(d_i^{intra}, d_i^{inter})}$, 平均轮廓系数为 $s = \frac{1}{n}\sum_{i=1}^{n} s_i$

平均轮廓系数越大，聚类效果越好。

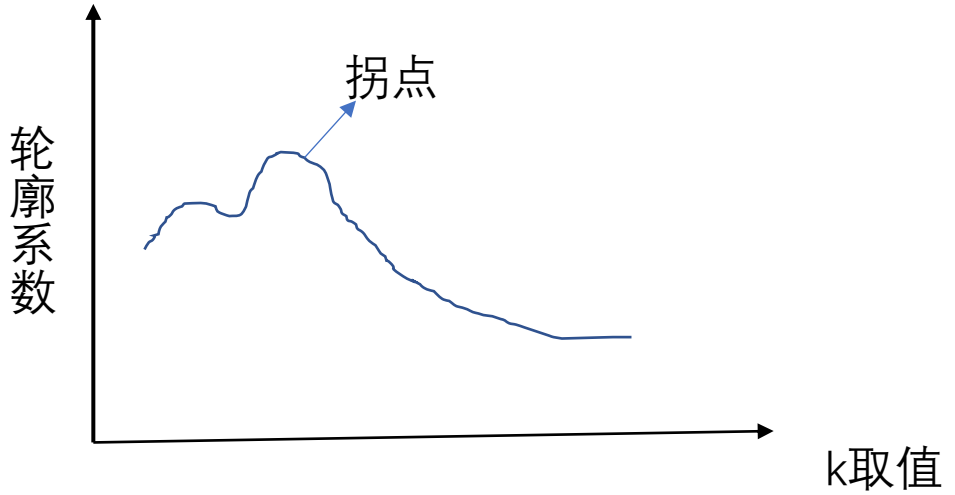
聚类相关其他评估/验证(cluster validation)

- 对簇结构存在性以及簇数目的评估

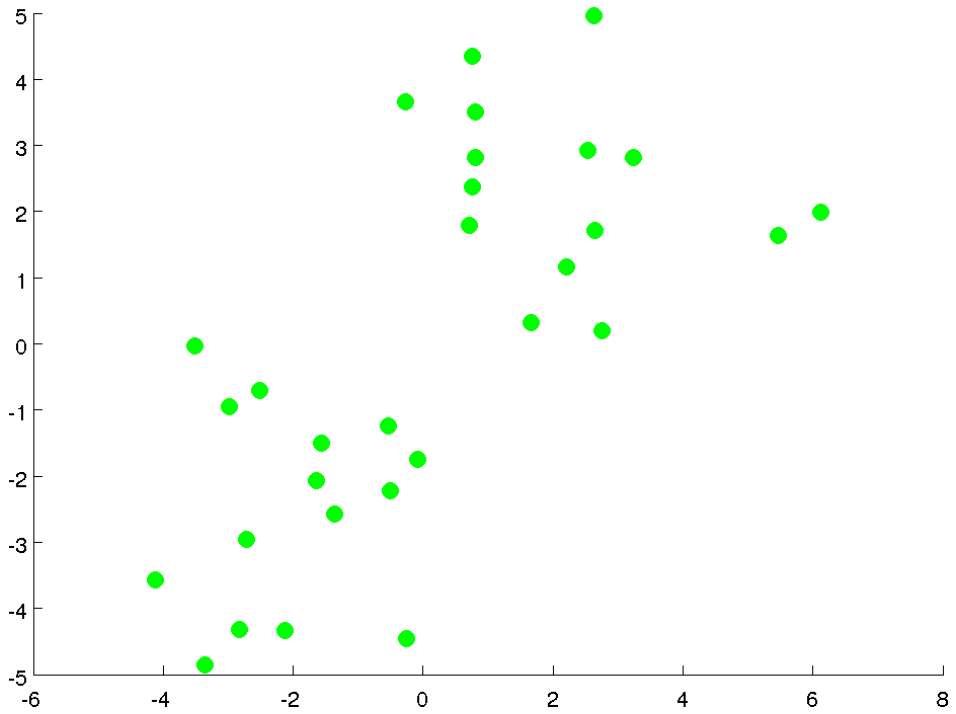
如果数据集本身就不具一定程度的簇结构，那么对其聚类的结果都是无意义的。
数据集到底包含了多少个簇？

很多算法如k均值、谱聚类等要求设定该超参数。

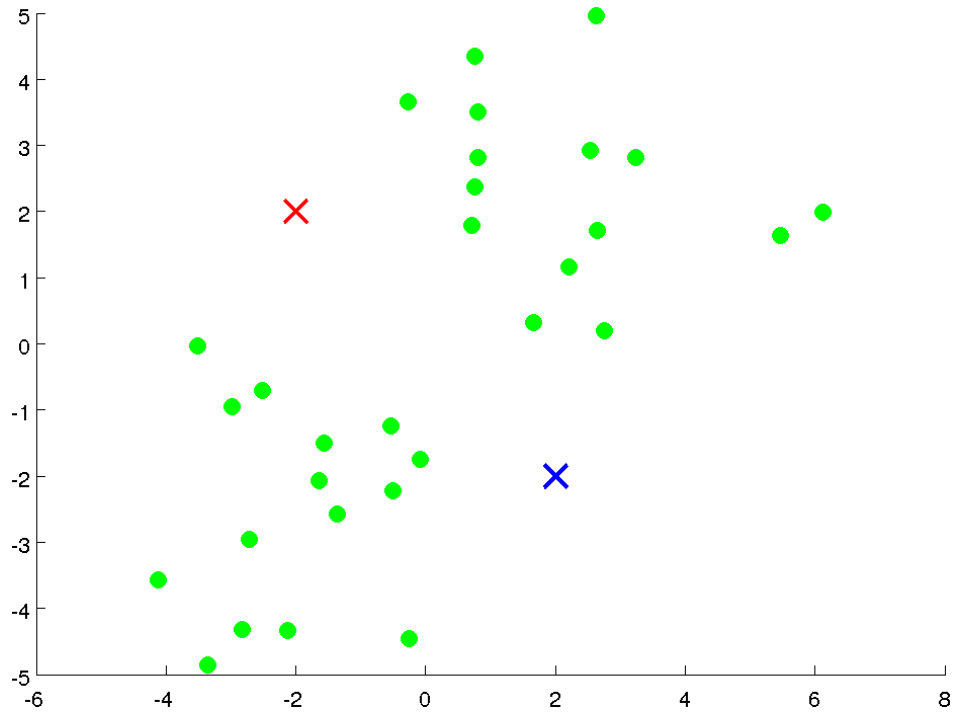
对不同的k值用内部评估方法
如轮廓系数进行评估。



第五章 聚类

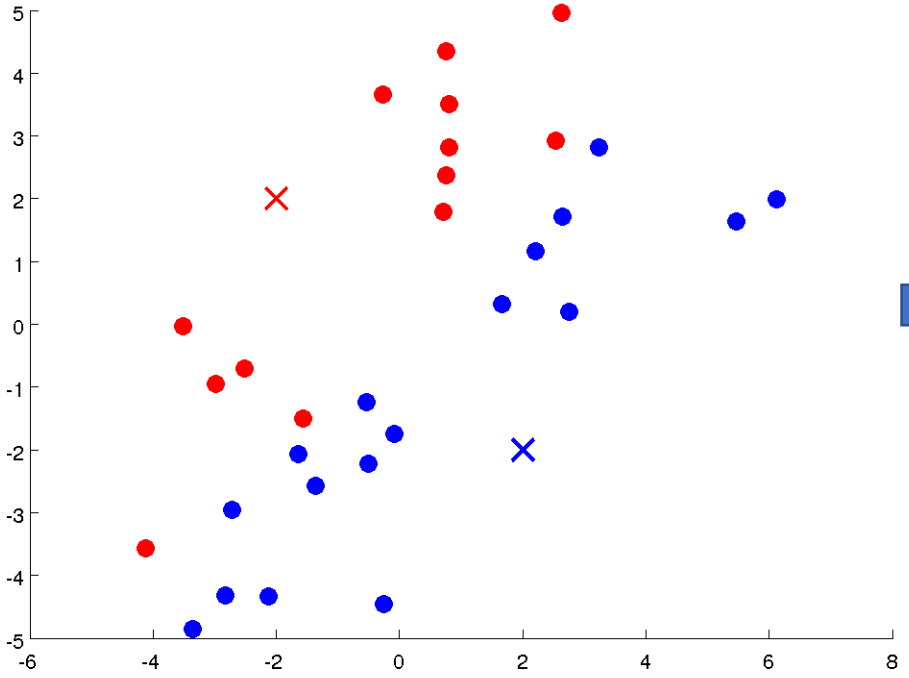


目标数据集

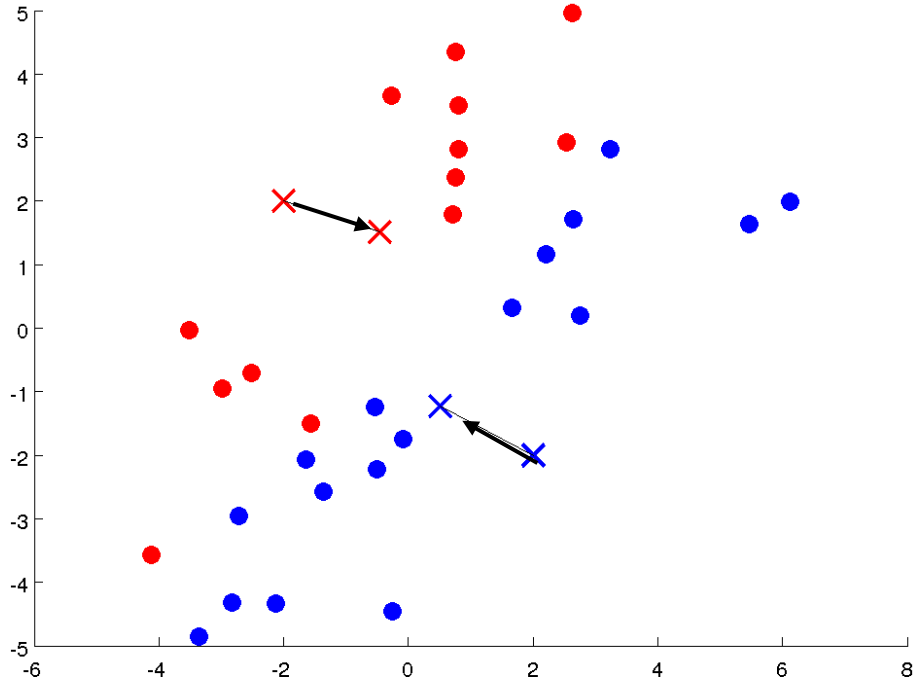


随机初始化中心点

第一次迭代

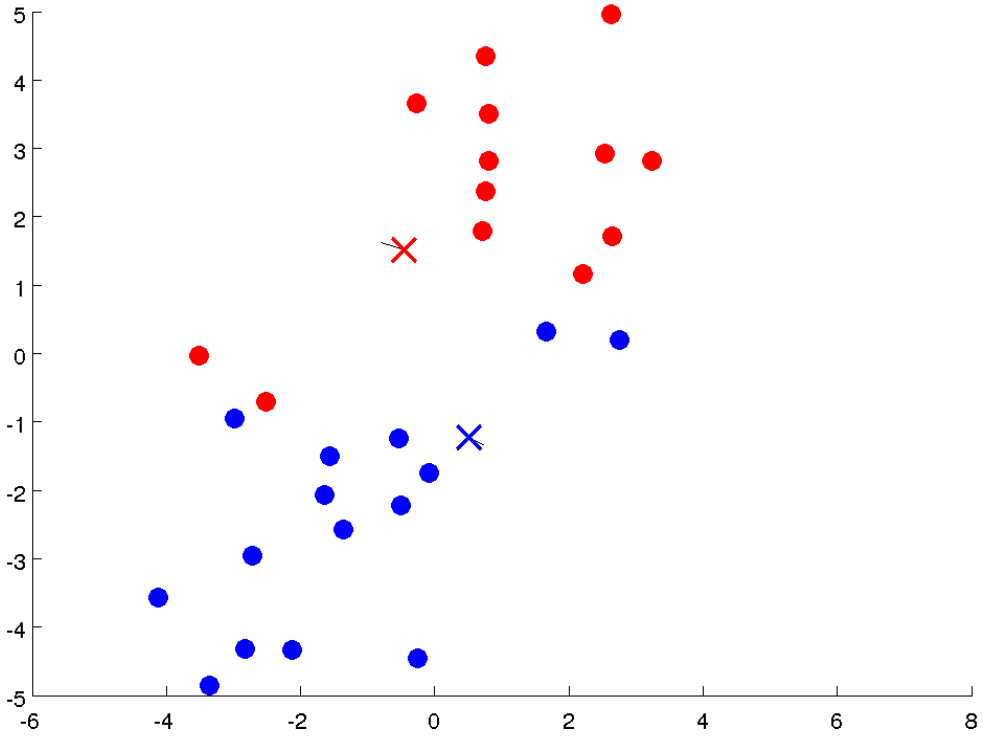


基于当前中心点计算每个点的类别

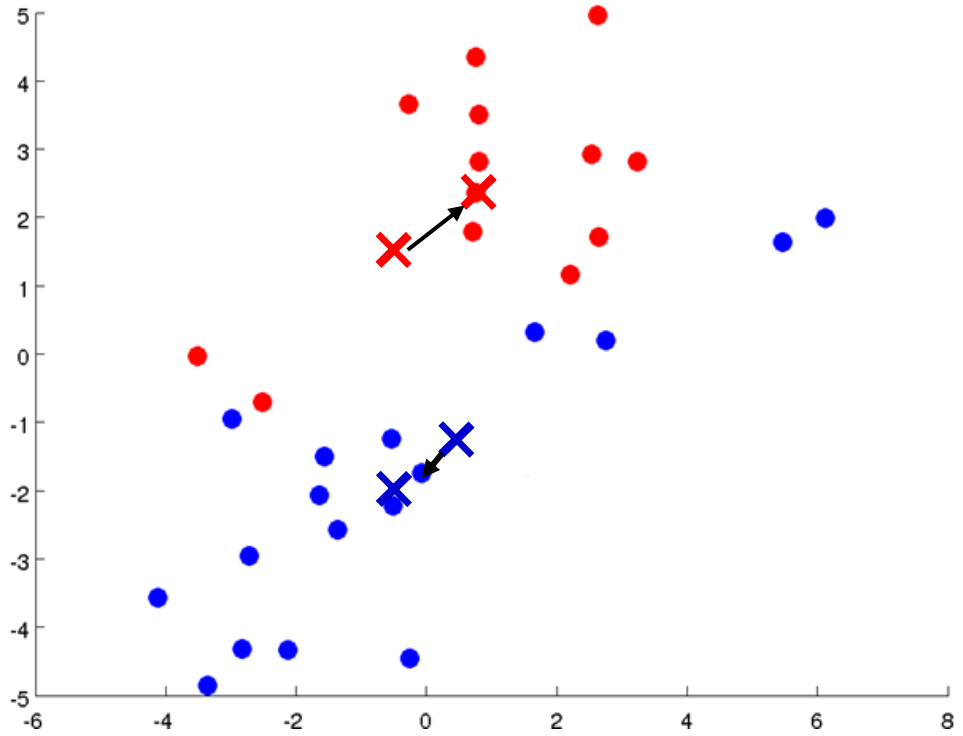


基于当前分组重新计算中心点

第二次迭代

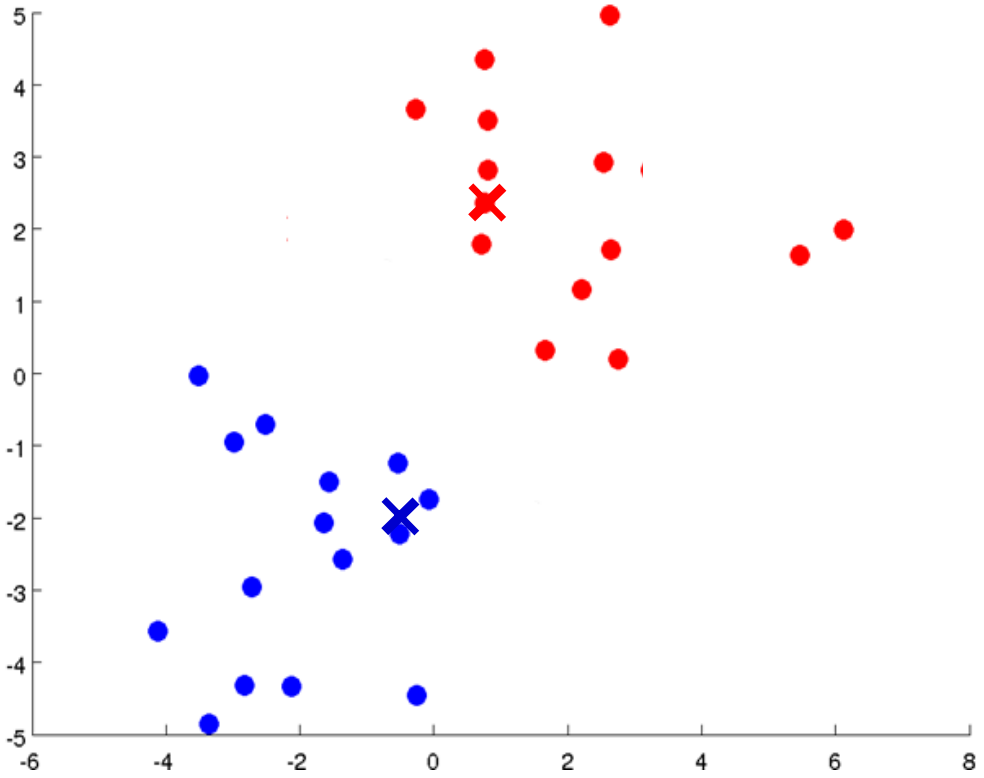


基于当前中心点计算每个点的类别

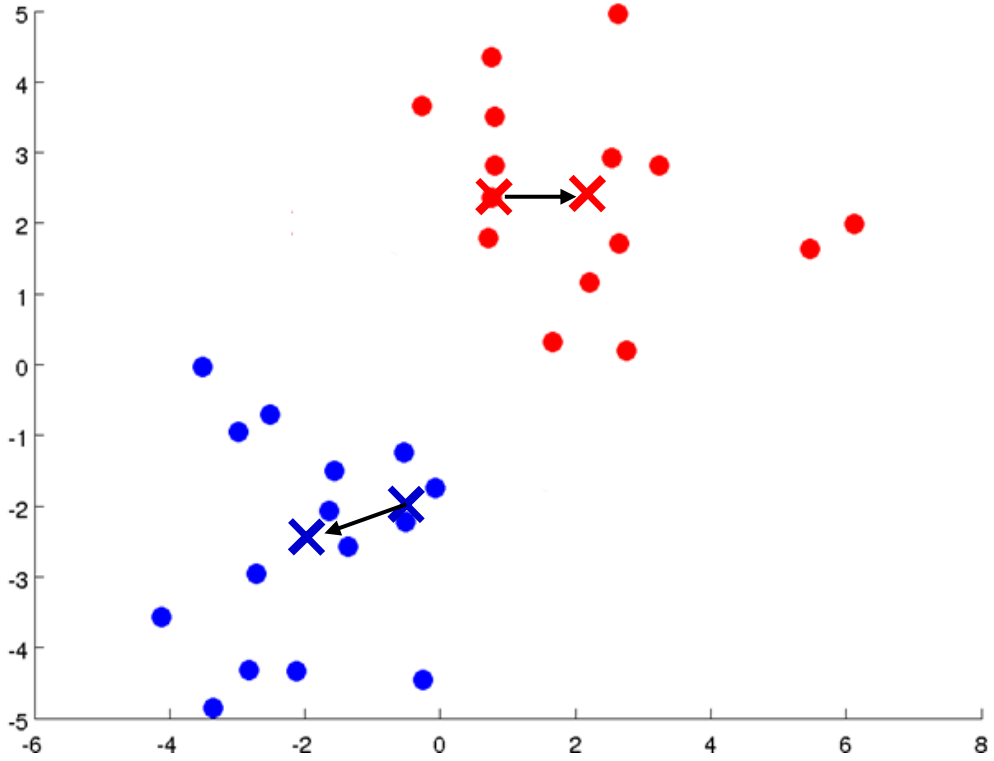


基于当前分组重新计算中心点

第三次迭代

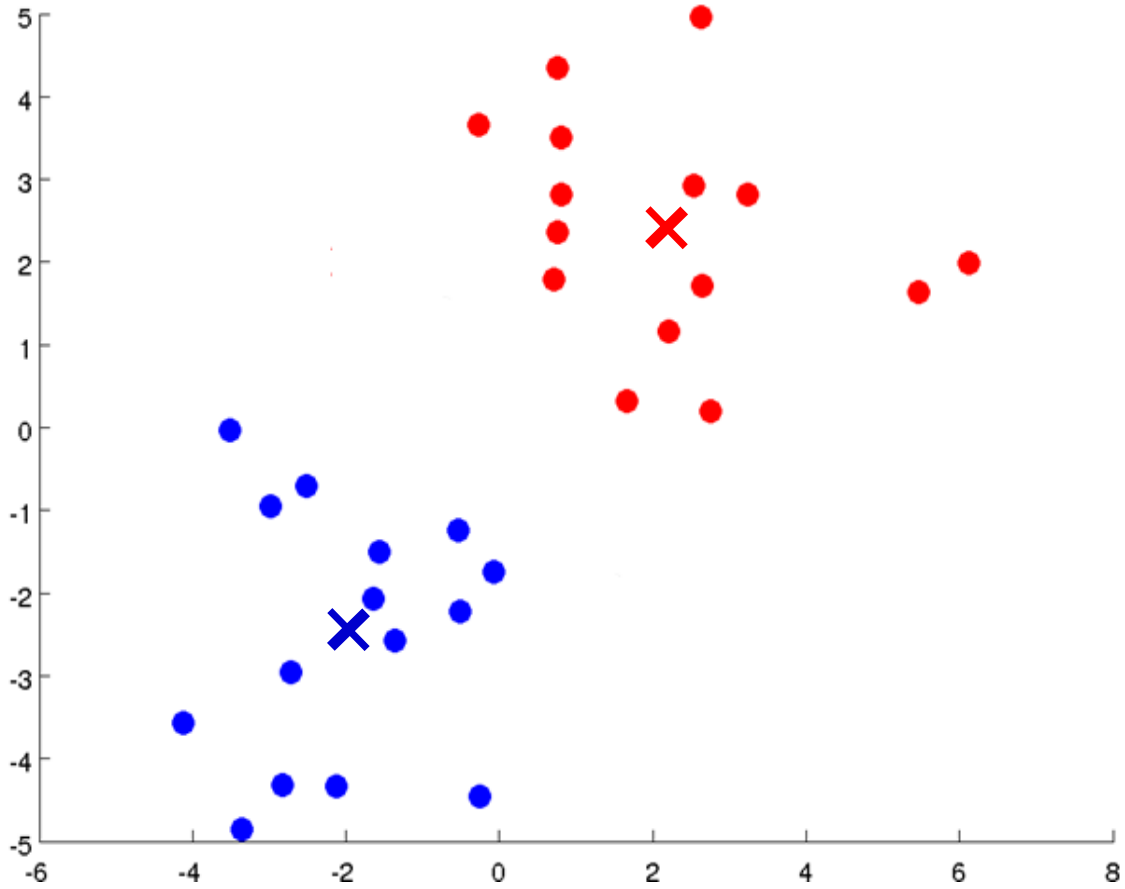


基于当前中心点计算每个点的类别



基于当前分组重新计算中心点

分组不再变化时，停止迭代



基于当前中心点计算每个点的类别

K-均值算法

对每个样本所属簇和每个簇中心的迭代更新。

随机初始化K个簇中心点 $\mu_1, \mu_2, \cdots \mu_K \in \mathbb{R}^s$

重复以下两步 {

//更新每个样本所属的簇

for $i = 1$ to n

$C_i \leftarrow$ K个中心点中与x_i最近的那个簇

//以均值作为簇中心进行更新

for $k = 1$ to K

$\mu_k \leftarrow$ 所有被标记到第k个簇的样本的均值

}直到分组不再变化或达到最大迭代次数

参数n, s, K分别表示样本集的大小，维度，以及簇的个数。

K均值的目标函数（损失函数）

$$\mathrm{J} = \min \sum_{k=1}^{K} \sum_{x_i \in C_k} d(\boldsymbol{x}_i, \boldsymbol{\mu}_k)$$

其中 $d(\boldsymbol{x}_i, \boldsymbol{\mu}_k) = \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$ 为样本点到簇中心欧式距离的平方

更新所属簇：$\boldsymbol{x}_i \in C_k \ , if \ k = \arg min_f \, d(\boldsymbol{x}_i, \boldsymbol{\mu}_f)$ ，

更新簇中心：$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i$

簇中心计算公式的推导

对目标函数求偏导

$$\frac{\partial J}{\partial \mu_k} = \sum_{\mathrm{k}=1}^{K} \sum_{\boldsymbol{x}_i \in C_k} \frac{\partial d(\boldsymbol{x}_i, \mu_k)}{\partial \mu_k} = \sum_{\boldsymbol{x}_i \in C_k} \frac{\partial \| \boldsymbol{x}_i - \mu_k \|^2}{\partial \mu_k} = \sum_{\boldsymbol{x}_i \in C_k} -2(\boldsymbol{x}_i - \mu_k)$$

令导函数为0

$$\sum_{\boldsymbol{x}_i \in C_k} -2(\boldsymbol{x}_i - \mu_k) = 0$$

$$|C_k|\mu_k - \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i = 0 \quad \Rightarrow \quad \boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i$$

第五章 聚类

5.1 聚类基本概念与评估

5.2 k均值聚类

5.2.1基本算法及目标函数

5.2.2关键问题及拓展算法

5.3 层次聚类

5.4 密度聚类*

K均值算法的关键问题

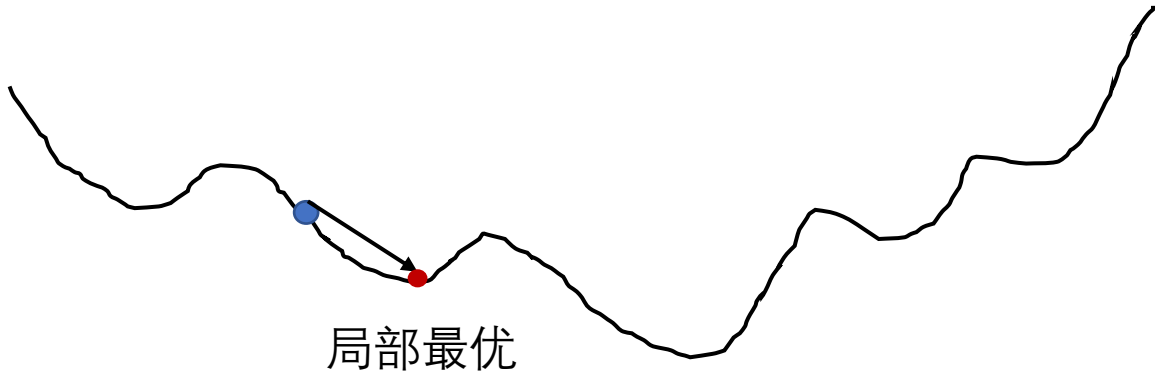
- 初始化

K均值算法对初始化敏感，即不同初始化可能得到不同结果。

- 簇数目K的设定

簇数目是K均值算法的一个认为设定参数，K的取值不同，得到不同结果。

初始化对结果的影响

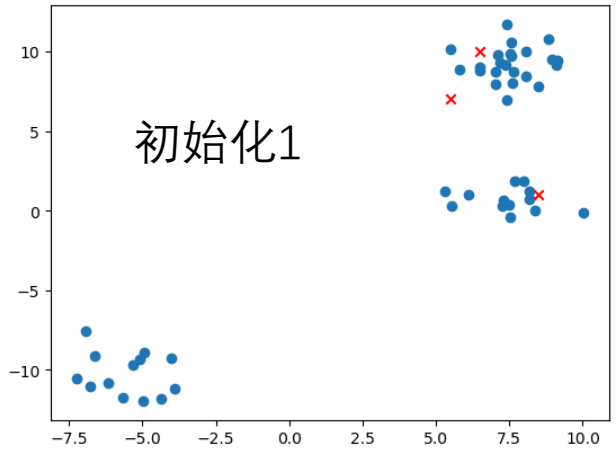


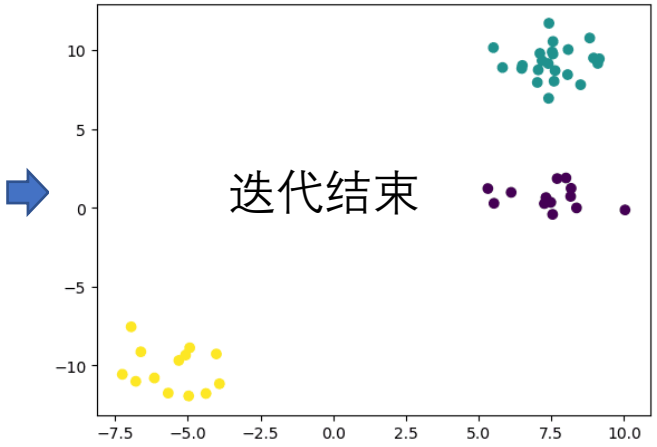
原因：通过前面迭代算法只能得到目标函数的局部最优。

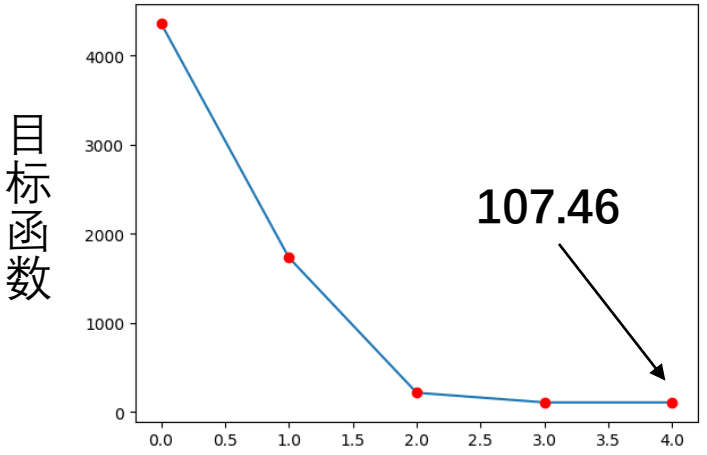
针对随机初始化的改进措施：

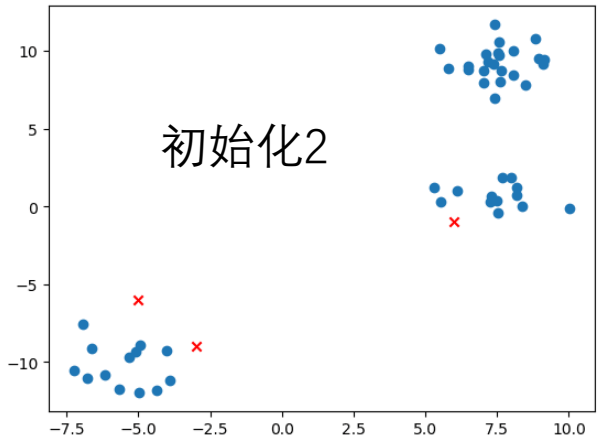
- 随机选择k个训练样本作为k个簇的初始中心点；
- 多次基于随机初始化运行，选择目标函数值最小的结果。

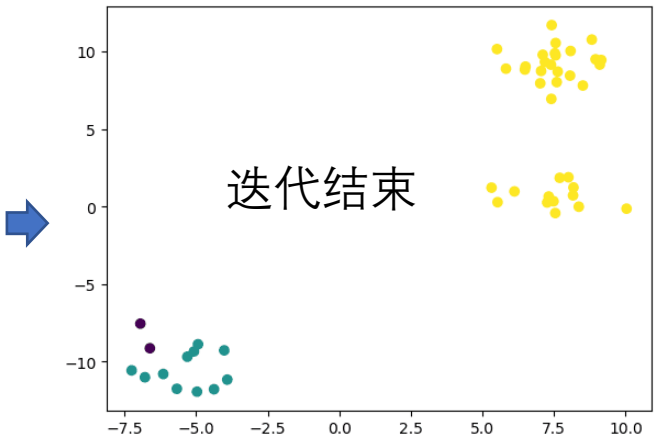
初始化对k均值结果的影响

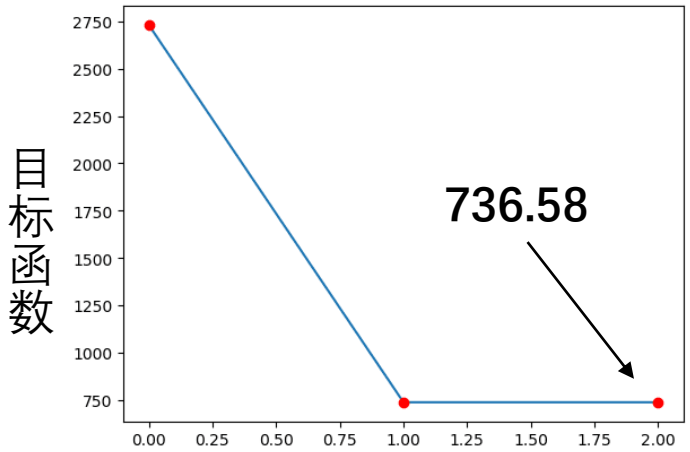




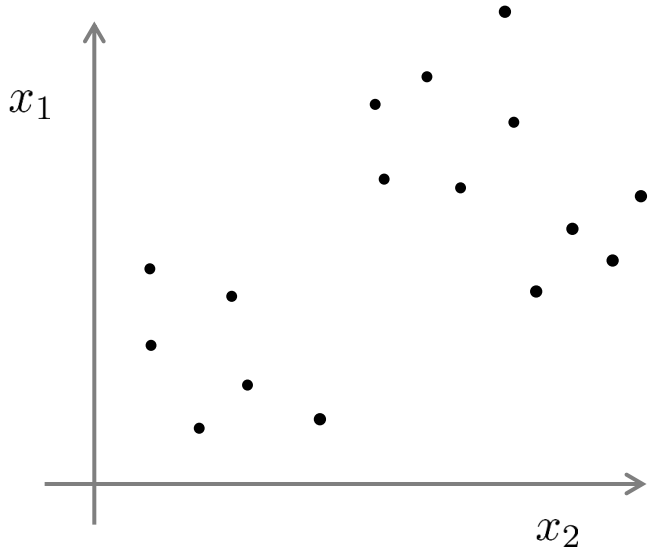




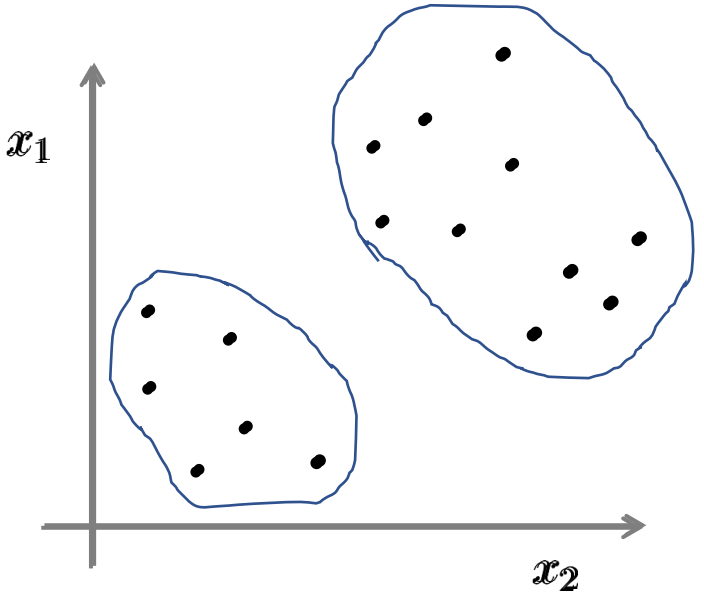




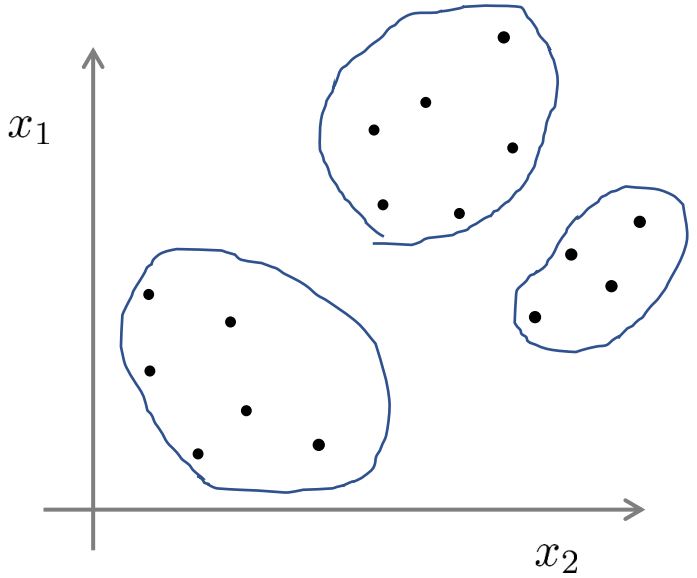
k的取值对结果的影响



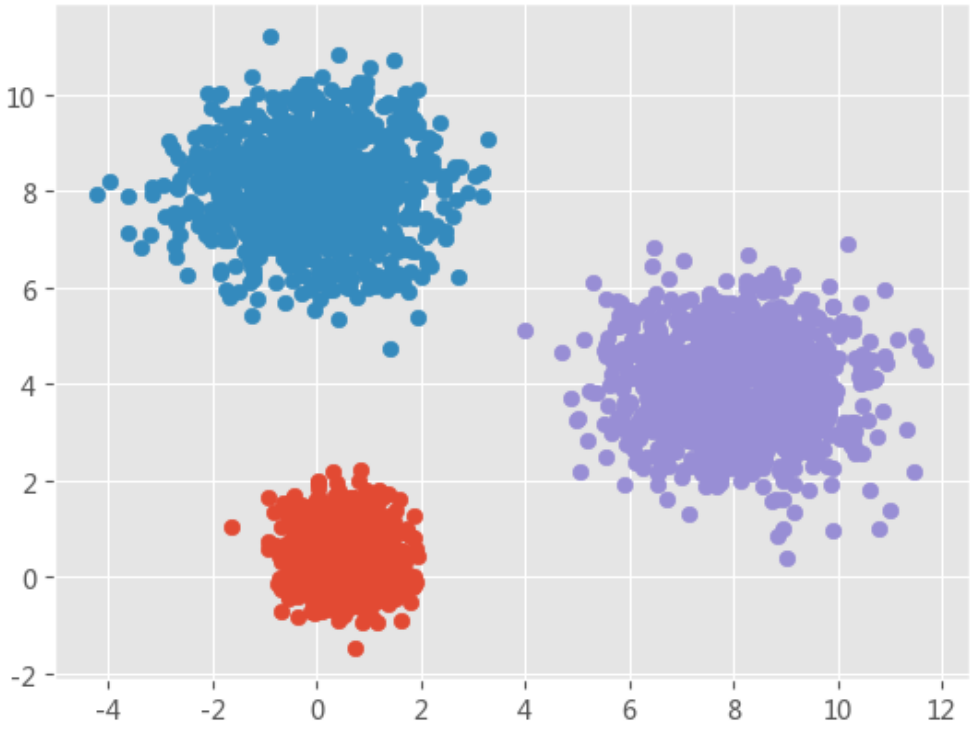
K=2

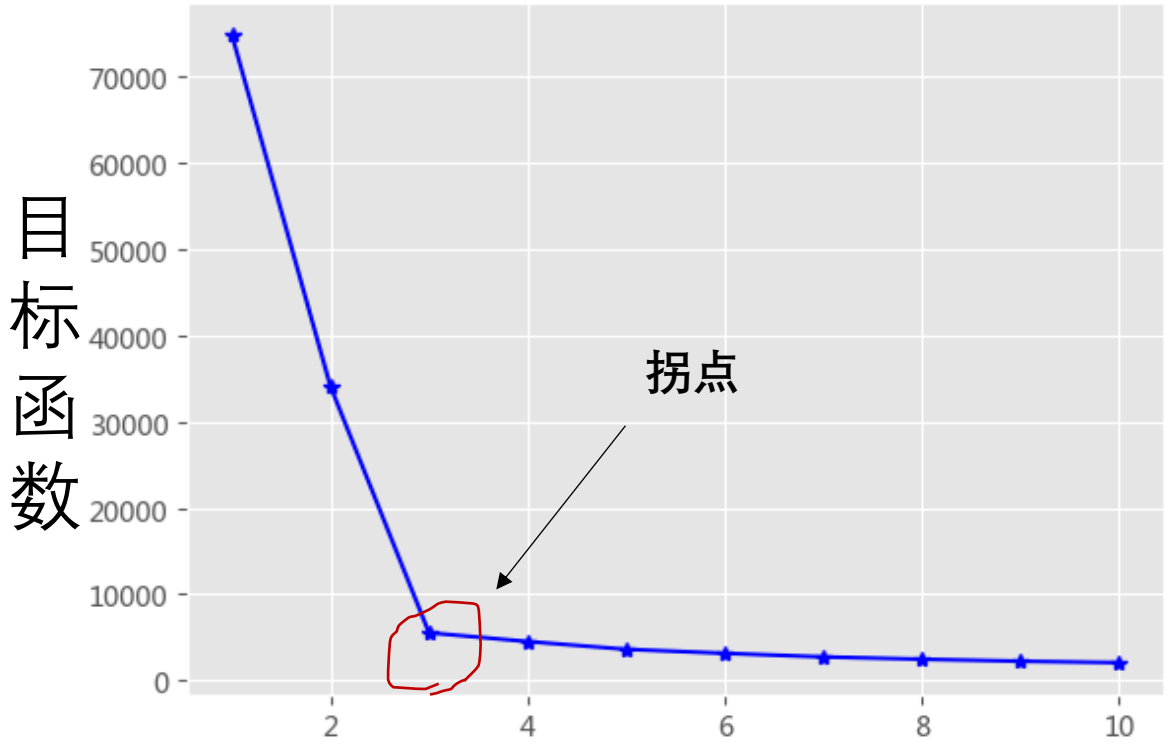


K=3



K的选择：拐点法





注意：与轮廓系数不同，k-means目标函数最小值随K的增加而下降，当K=N时，目标函数最小为0。很多情况下，拐点可能不明显。

K均值的拓展方法*

■ 为提高效果做的模型上的改进：

- **球面k均值**（Spherical k-means）
  用余弦相似度，即单位向量之间的内积来衡量对象-簇的距离。适用于文本等稀疏高维数据。
- **模糊k均值**（fuzzy k-means）
  引进隶属度表示对象到簇的隶属程度，允许一个对象同时属于多个簇。
- **核k均值**(Kernel k-means)
  通过核函数可以找出线性不可分的簇。

■ 为增加鲁棒性

- **K-中心(样本)点**（k-medoids）
  选用数据集中的样本点来作为簇中心，而不是均值。对噪声核离群点更加鲁棒。

■ 为处理大规模/不断增加的数据做的算法实现层面的改进

- **在线k均值**（online k-means）

球面k均值(spherical clustering)

余弦相似度

$$cosin(\boldsymbol{x}_i, \boldsymbol{\mu}_k) = \frac{\boldsymbol{x}_i{}^T \boldsymbol{\mu}_k}{\|\boldsymbol{x}_i\| \|\boldsymbol{\mu}_k\|}$$

假设每个样本$\boldsymbol{x}_i$和簇中心$\boldsymbol{\mu}_k$都被归一化到单位向量(在一个超球面)，则余弦距离等价于

$$d(\boldsymbol{x}_i, \boldsymbol{\mu}_k) = 1 - \boldsymbol{x}_i{}^T \boldsymbol{\mu}_k = 1 - \sum_p x_{ip} \mu_{kp}$$

x_{ip}, μ_{kp}分别为向量 $\boldsymbol{x}_i$和$\boldsymbol{\mu}_k$的第p个维度对应的值。按前面步骤推导得到以下公式用于基于余弦距离的中心点的更新

$$\boldsymbol{\mu}_k = \frac{\sum_{x_i \in C_k} \boldsymbol{x}_i}{\sqrt{\sum_p (\sum_{x_i \in C_k} x_{ip})^2}}$$

模糊k均值

K均值算法中，每个样本只允许属于一个簇，簇之间没有重叠，因此也叫硬聚类。现实应用中，没多时候簇之间有不同程度的重叠，即每个样本可能以不同程度地属于多个簇。

模糊k均值：引入模糊隶属度u_{ik}，表示第i个样本到第k个簇的隶属度。

目标函数为

$$\mathrm{J} = \min \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ik}^{m} d(\boldsymbol{x}_i, \boldsymbol{\mu}_k)$$

$$u_{ik} \geq 0, \qquad \text{且对所有} i, \text{满足} \sum_{k=1}^{K} u_{ik} = 1$$

参数 $m > 1$ 为模糊系数，用于控制隶属度的模糊化水平。

模糊隶属度

$$\mathbf{U}_{n\times K}=\begin{pmatrix}1&0&0\\1&0&0\\0&0&1\\0&1&0\\0&1&0\\1&0&0\end{pmatrix}$$

硬隶属度矩阵

$$\mathbf{U}_{n\times K}=\begin{pmatrix}0.9&0.1&0.0\\0.8&0.1&0.1\\0.1&0.2&0.7\\0.2&0.5&0.3\\0.1&0.7&0.2\\1.0&0.0&0.0\end{pmatrix}$$

模糊隶属度矩阵

试试看你能用拉格朗日方法推导出来吗？

模糊隶属度的迭代公式：

$$u_{ik}=\frac{(\frac{1}{d_{ik}})^{\frac{1}{m-1}}}{\sum_{f=1}^{K}(\frac{1}{d_{if}})^{\frac{1}{m-1}}}$$

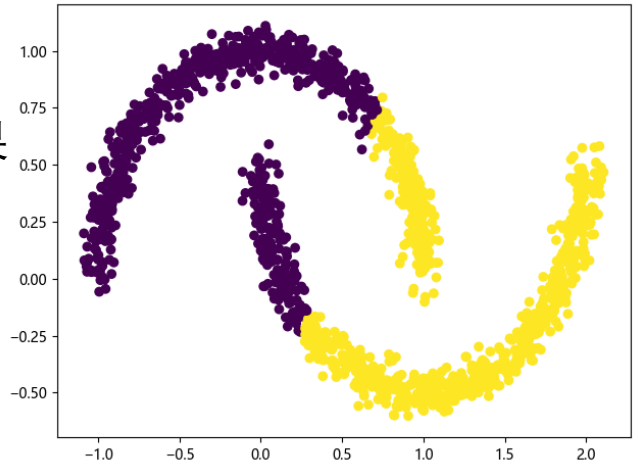
采用欧式距离 $d(\boldsymbol{x}_i,\boldsymbol{\mu}_k)=\|\boldsymbol{x}_i-\boldsymbol{\mu}_k\|^2$，簇中心的迭代公式为：

$$\boldsymbol{\mu}_k=\frac{\sum_{i=1}^{n}u_{ik}^{m}\boldsymbol{x}_i}{\sum_{i=1}^{n}u_{ik}^{m}}$$

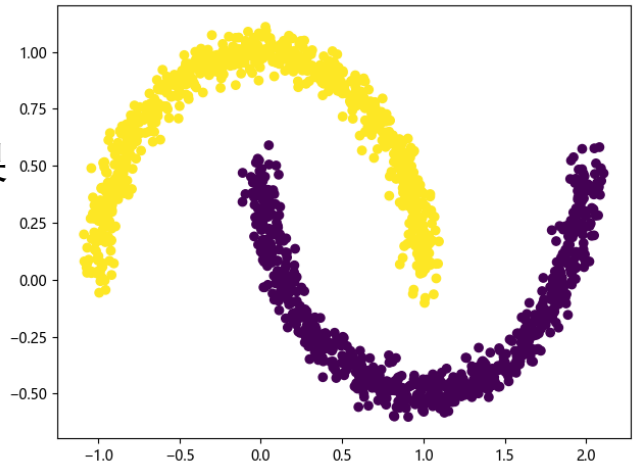
为了方便$d(\boldsymbol{x}_i,\boldsymbol{\mu}_k)$表示 为 d_{ik} 。

核k均值(kernel k-means)

k均值结果



核k均值结果



核k均值(Kernel k-means)把原始数据映射到一个新的特征空间进行k均值聚类：

$$\mathrm{J} = \min \sum_{k=1}^{K} \sum_{x_i \in C_k} \|\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_k\|^2$$

其中 $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\phi(\boldsymbol{x}_i) \in C_k} \phi(\boldsymbol{x}_i)$

为什么要映射：来线性不可分的簇变得线性可分。

怎么实现映射：不需要显性地学习映射函数ϕ。目标函数进行转换后表示为样本之间内积的形式，通过核函数（如高斯核函数）得到映射空间下样本之间的内积。

$$\mathrm{J} = \mathrm{f}(\phi^T(\boldsymbol{x}_i)\ \phi(\boldsymbol{x}_j))$$

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi^T(\boldsymbol{x}_i)\ \phi(\boldsymbol{x}_j)$$

k均值类型算法-总结

■基本框架：

- 交替更新类别和中心点；
- 计算复杂度随数据规模成线性增加$O(n)$；

■局限性以及改进

- 对初始化敏感；
- 不同的初始化以及k值可能得到不同的结果；
- 不同的距离度量适用于特定的形状的簇，如欧式距离对于2D数据，产生圆形的簇；
- 通过设计不同的目标函数得到改进的k均值。

第五章 聚类

5.1 聚类基本概念与评估

5.2 k均值聚类

5.3 层次聚类

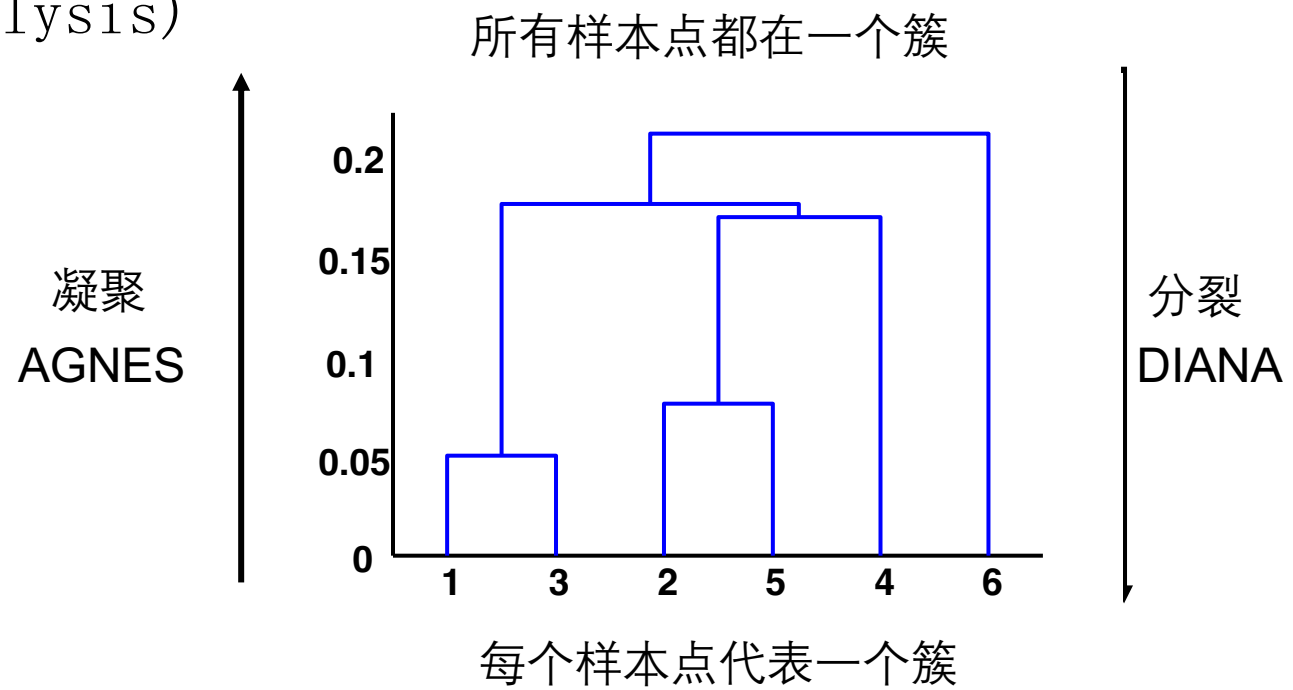
5.4 密度聚类*

层次聚类(Hierarchical Clustering)

关键问题：如何构造树状图(dendrogram)来表示相互嵌套的簇。

主要方法:

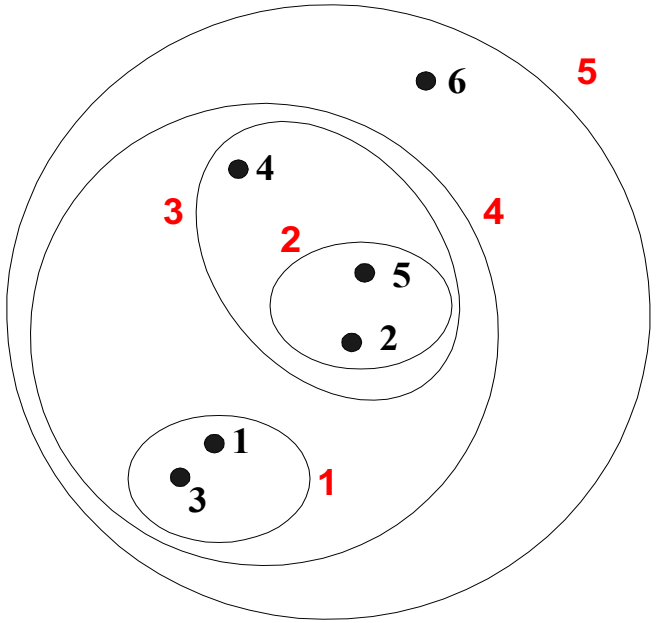
- 自下而上凝聚(Agglomerative Nesting)
- 自上而下分裂(Divisive Analysis)

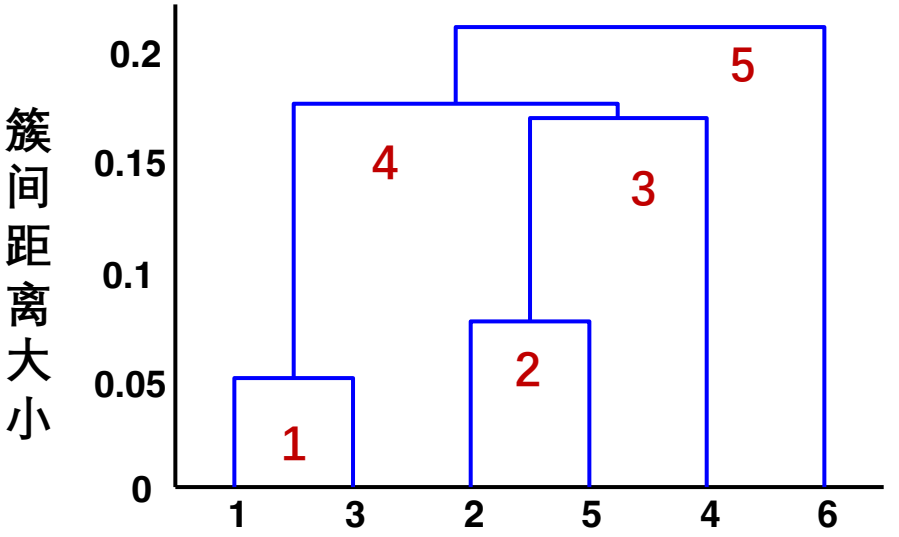


凝聚层次聚类算法
AGNES(Agglomerative Nesting)

AGNES算法基本步骤

1. 每个样本各自为一个簇，把簇距离矩阵设置为样本之间的距离矩阵
2. **Repeat**
3. 把两个最近的簇合并
4. 更新簇距离矩阵
5. **Until** 只剩下一个簇

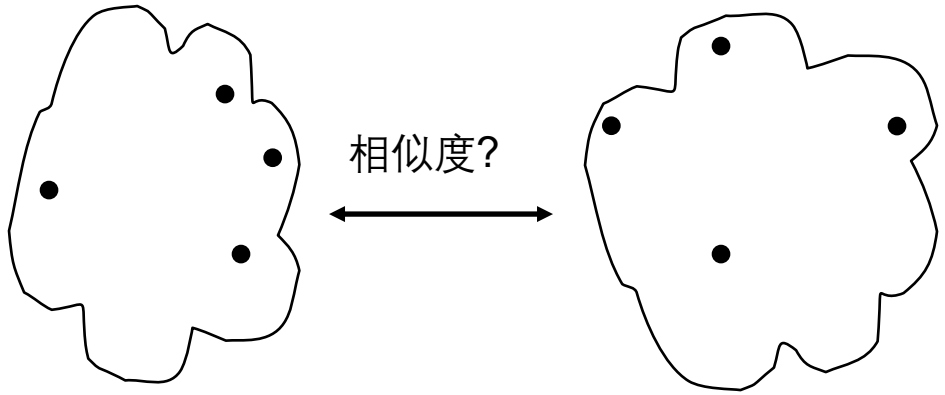




关键问题：如何计算两个簇之间的紧密度

不能的度量方法得到不同的层次聚类算法

怎么定义两个簇(样本子集)之间的相紧密度？



	C_1	C_2	C_3	C_4	C_5
C_1					
C_2					
C_3					
C_4					
C_5					

簇相似度矩阵

- 最常用的是基于互连性的方法

基于互连性(Linkage-based)的方法

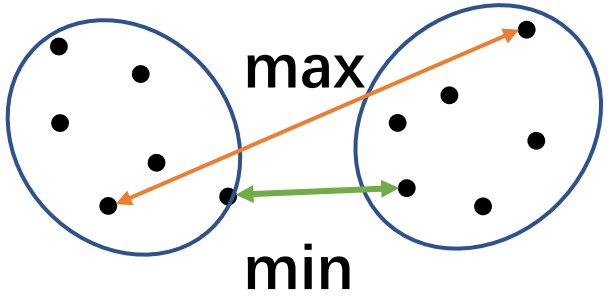
- 最短距离/单连接(single-linkage)：

  属于两个不同簇的样本之间距离的最小值：$l(C_r, C_s) = min_{x_{ri} \in C_r, x_{sj} \in C_s} d(x_{ri}, x_{sj})$

- 最长距离/全连接(Complete-linkage):

  属于两个不同簇的样本之间距离的最大值：

  $l(C_r, C_s) = max_{x_{ri} \in C_r, x_{sj} \in C_s} d(x_{ri}, x_{sj})$



- 平均距离/平均连接(Group-average):

  属于两个不同簇的样本之间距离的平均值：

  $l(C_r, C_s) = \frac{1}{|C_r||C_s|} \sum_{x_{ri} \in C_r, x_{sj} \in C_s} d(x_{ri}, x_{sj})$

例子：基于最短距离的凝聚层次聚类

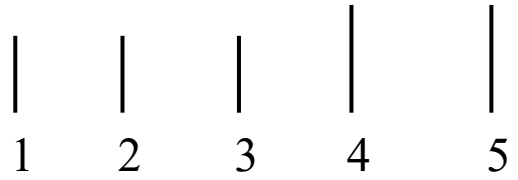
- 属于两个不同簇的样本之间距离的最小值/相似度的最大值

	C_1	C_2	C_3	C_4	C_5
C_1	-	0.10	0.90	0.35	0.80
C_2	0.10	-	0.30	0.40	0.50
C_3	0.90	0.30	-	0.60	0.70
C_4	0.35	0.40	0.60	-	0.20
C_5	0.80	0.50	0.70	0.20	-

初始距离矩阵 D

初始化: 5个点分别对应5个簇：

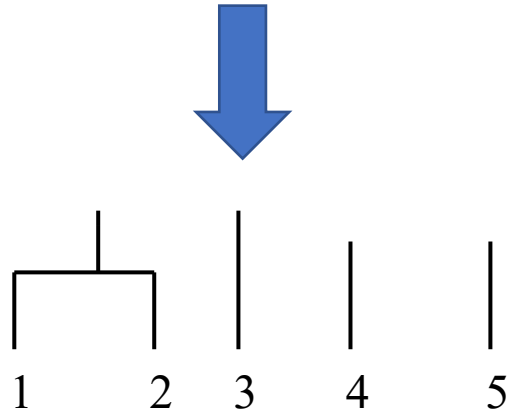
C1={1}, C2={2}, C3={3}, C4={4}, C5={5}



第1步：把D中距离最小的两个簇C1={1}和C2={2}合并后, 更新簇以及D：

C1={1，2}, C2={3}, C3={4}, C4={5}

	C_1	C_2	C_3	C_4
C_1	-	0.30	0.35	0.50
C_2		-	0.60	0.70
C_3			-	0.20
C_4				-

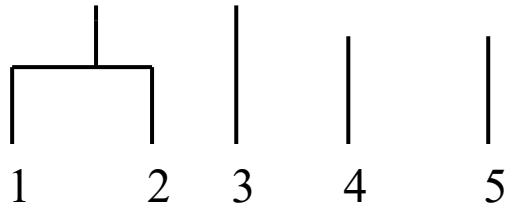


例子：基于最短距离的凝聚层次聚类(续)

第1步：把D中距离最小（0.1）的两个簇C1={1}和C2={2}合并后，更新簇以及D：

C1={1，2}；C2={3}, C3={4}, C4={5}

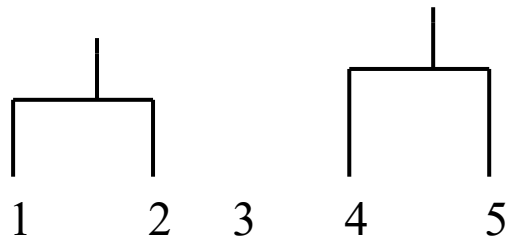
	C_1	C_2	C_3	C_4
C_1	-	0.30	0.35	0.50
C_2		-	0.60	0.70
C_3			-	0.20
C_4				-



第2步：找到D中最小（0.2）的两个簇C3={4}, C4={5}合并后更新簇以及D：

C1={1，2}；C2={3}, C3={4，5}

	C_1	C_2	C_3
C_1	-	0.30	0.35
C_2		-	0.60
C_3			-



例子：基于最短距离的凝聚层次聚类(续)

第2步：找到D中最小（0.2）的两个簇C3={4}, C4={5}
合并后更新簇以及D：

C1={1，2}，C2={3}, C3={4，5}

	C_1	C_2	C_3
C_1	-	0.30	0.35
C_2		-	0.60
C_3			-

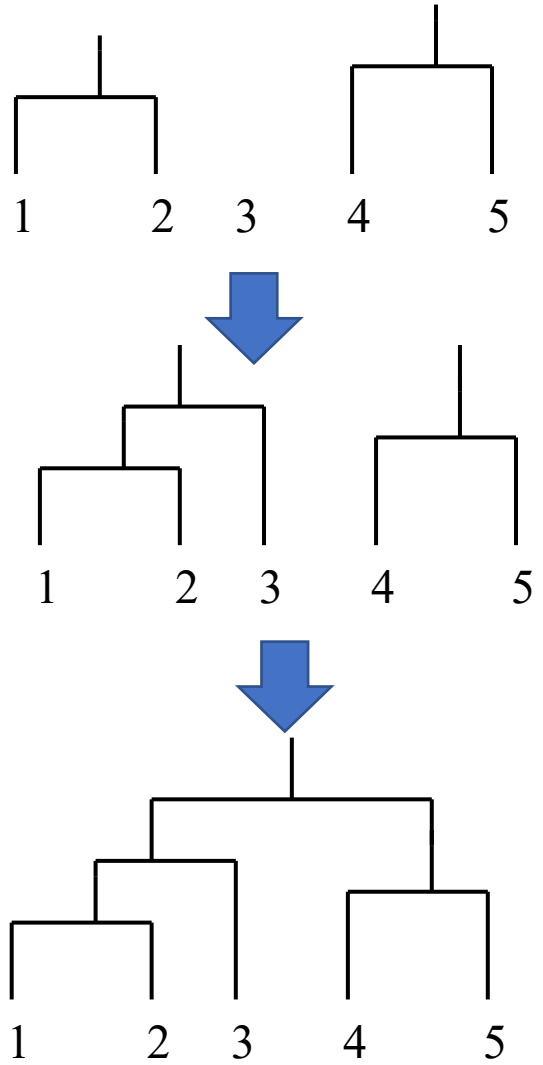
第3步：找到 D中距离最小(0. 3)的两个簇C1={1,2}, C2={3}合并后更新簇以及D：

C1={1，2，3}，C2={4，5}

	C_1	C_2
C_1	-	0.35
C_2		-

第4步：把 D中仅剩的距离(0. 35)的两个簇C1={1，2，3}，C2={4，5}合并成一个簇

C1={1，2，3，4，5}



对前面例子手动构造基于平均连接的树状结构。

不同linkage各自的局限性

- 最短距离：容易把两个簇混合在一起，两个簇从全局来看是分开的，但由于中间有个别点比较近就被合并了；对噪声敏感。

- 最长距离：可能把属于一个簇的样本分开；从整体上比较接近，但由于个别点离得比较开，根据最长距离原则使得两个子集距离很大，从而没有被合并；

- 平均距离：相对中和了以上两种情况，用得比较多。为了进一步减小对噪声得敏感，可以用中值代替平均值。

- 以上三种都需要计算对象之间的成对距离，复杂度为$O(n^2)$

层次聚类 vs 划分聚类

- 好处：可以描述簇之间的层次关系；
- 不足：需要计算簇之间相似度矩阵，不适用于大规模数据集；
- 产生簇得过程是不可逆的，即不会对已经产生的簇进行修改。

混合聚类：

首先把数据集通过几次凝聚层次聚类得到微小的簇（把最紧密连接部分先聚类），然后对这些微簇用划分聚类如k均值再进一步分组得到最终的聚类结果。

作业4

完成学习通作业。

第五章 聚类

5.1 聚类基本概念与评估

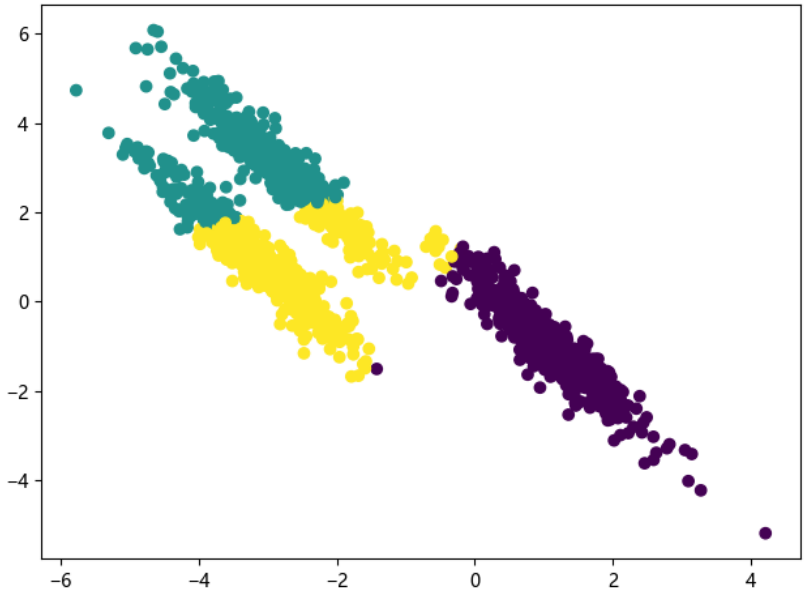
5.2 k均值聚类

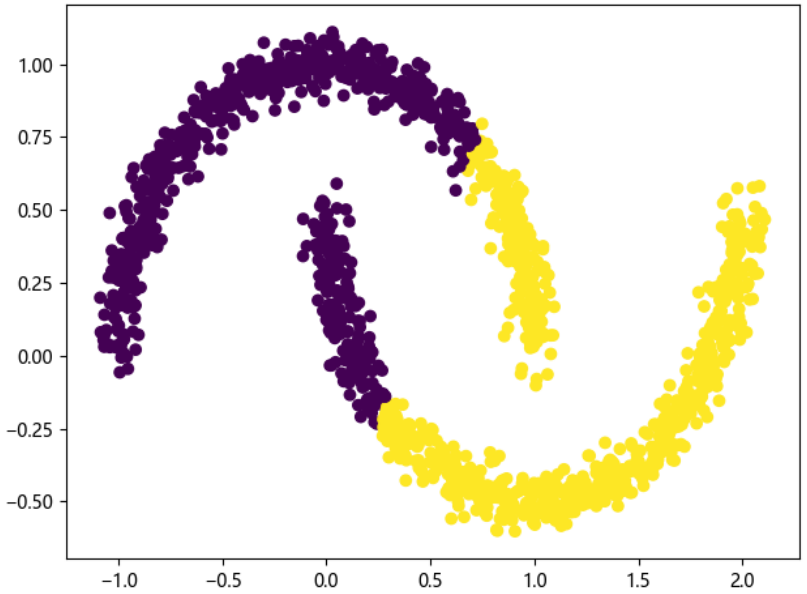
5.3 层次聚类

5.4 密度聚类*

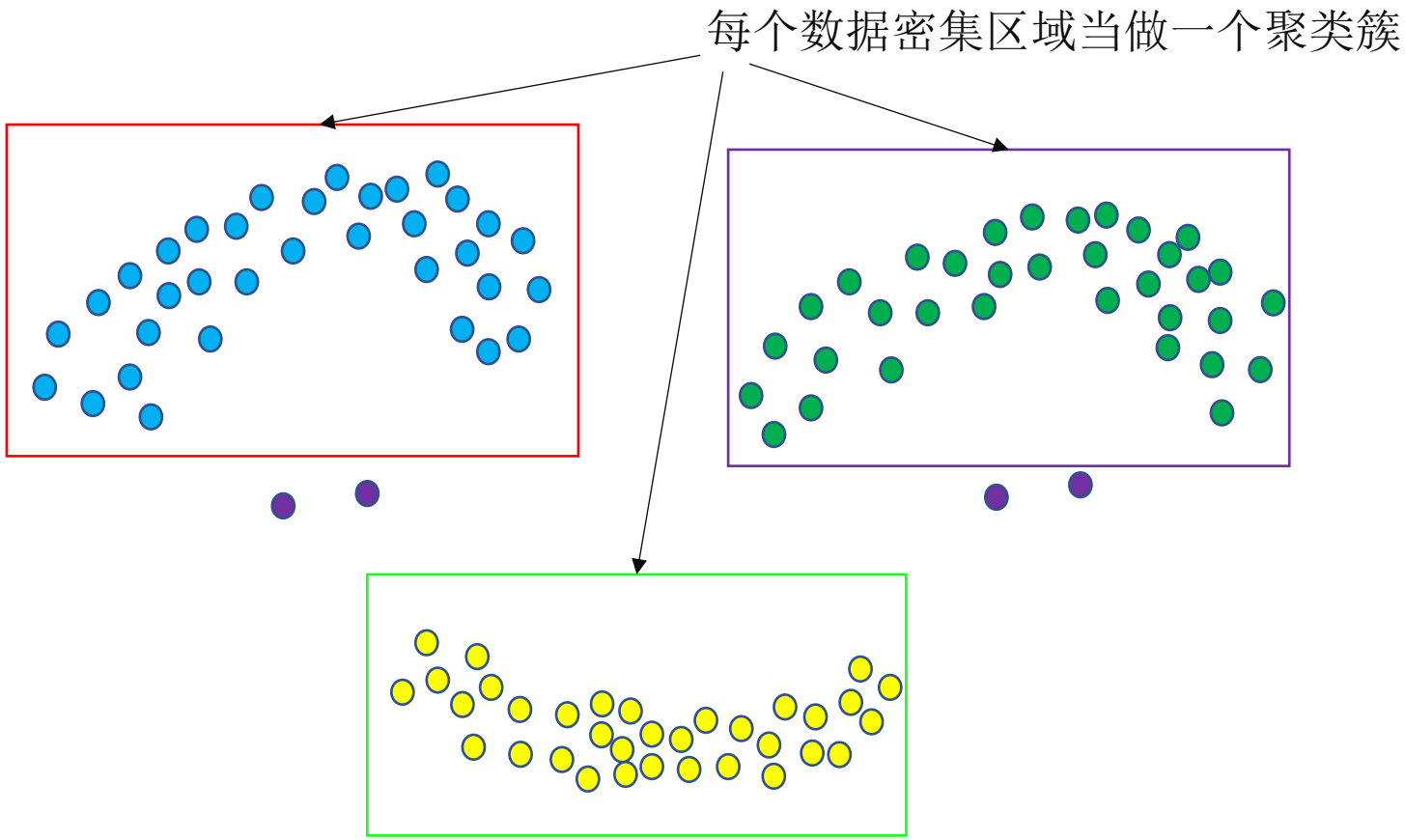
K-means聚类

只对近似圆形的簇有较好的表现。
对以下具有复杂现状的簇的数据集表现不理想。





基于密度的聚类



DBSCAN算法

基于密度的聚类算法的核心：通过某个点ϵ-邻域内样本点的数量来衡量该点所在空间的密度。

- 两个超参数，eps与MinPts。
- 数据点分为三类
- 点与点之间具有三种关系

两个超参数的定义

ϵ-领域：对于点p的ϵ-领域，定义为$N_{eps}(p)$，其中$N_{eps}(p) = \{q \in D | dist(p,q) \leq eps\}$。

$|N_{eps}(p)|$则定义为在p点的ϵ-领域内点的个数（包含p）。

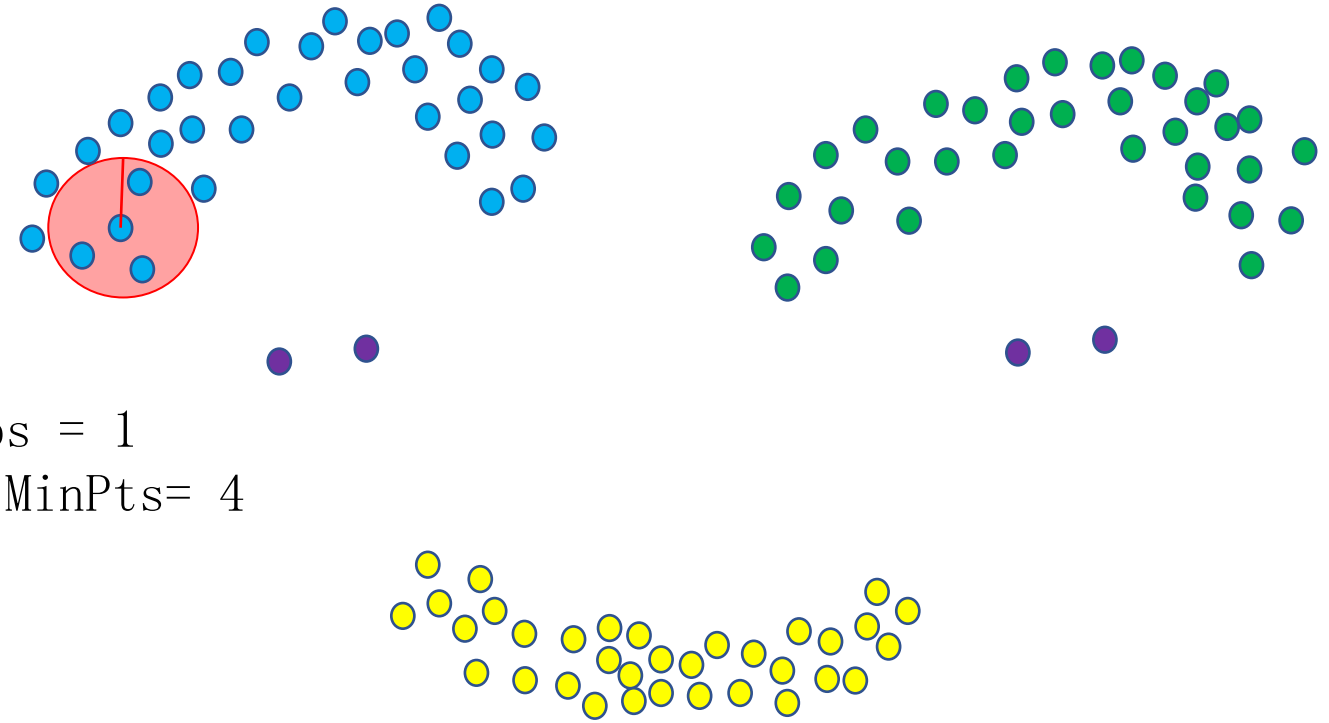
dist()函数为计算两点之间的距离的方法，

可以为欧式距离或余弦距离等。

MinPts：对于每一个ϵ-领域内点的数量至少为MinPts个。

若满足这个条件，则该点为核心点。

两个超参数的定义

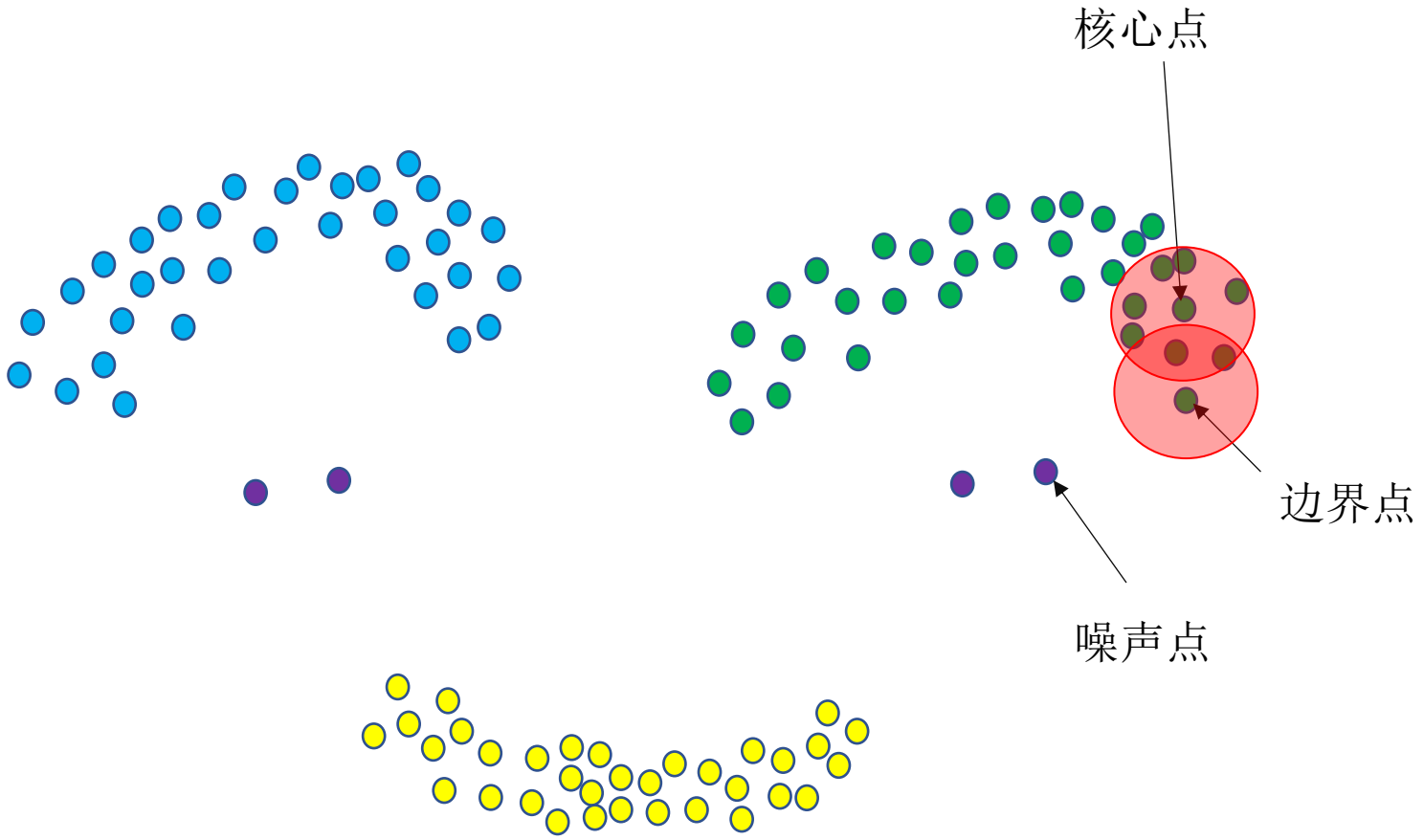


邻域半径eps = 1
最少点数目MinPts= 4

数据点的类别

- 核心点：其领域内的样本点的数量不少于MinPts。
- 边界点：处于核心点的领域内，但其本身的领域内样本点的个数少于MinPts。
- 噪声点：其领域内样本点的个数少于MinPts，且不处于核心点的领域内。

数据点的类别



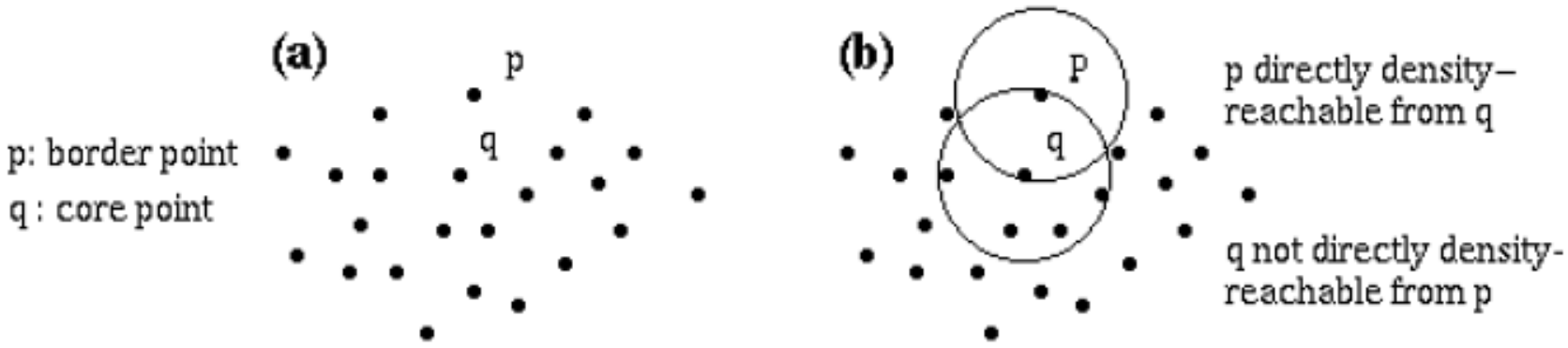
点与点之间的三种关系

1. 直接密度可达：若点p处于点q的ϵ-领域中，且q为核心点，则称点p可由点q直接密度可达（密度直达）。注意反之不一定成立，若点p为边界点，则点q不能由点p直接密度可达。-- 核心点到其领域内其他点的关系

点p与q应满足以下两个条件：

1) $\mathrm{p} \in N_{eps}(q)$

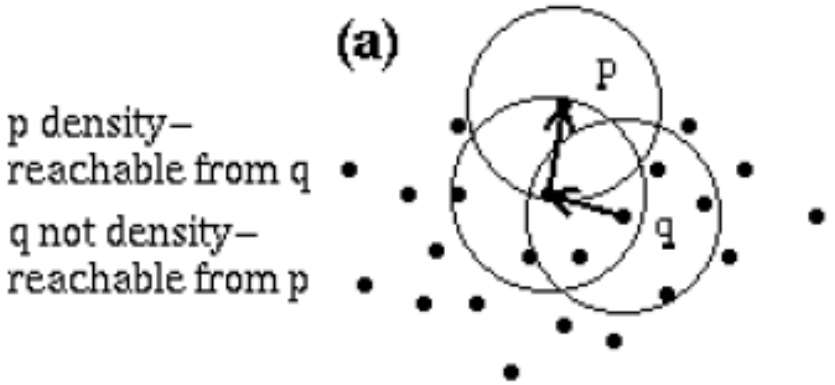
2) $\left|N_{eps}(q)\right| \geq MinPts$(核心点)

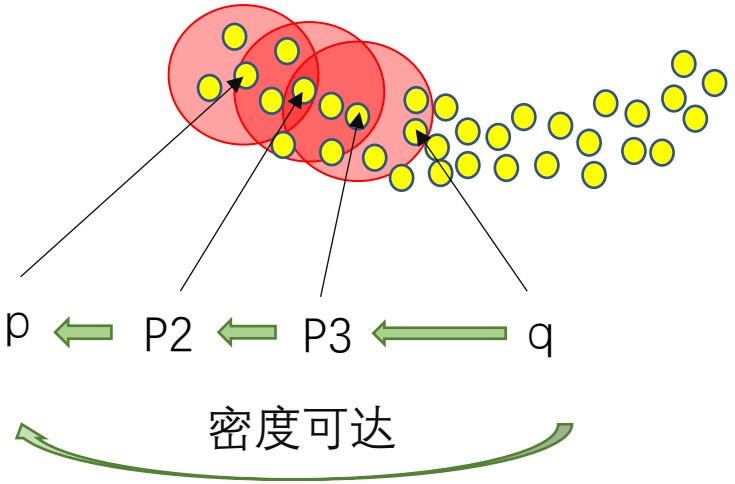


点与点之间的三种关系

2. 密度可达：对于点p和点q, 如果存在样本序列$p_1, p_2, \cdots, p_T$, 满足$p_1 = p, p_T = q$， 且p_{t-1}由p_t密度直达，则称p由q密度可达，即密度可达满足传递性。此时序列中的传递样本$p_2, \cdots, p_{T-1}$均为核心对象，因为只有核心对象才能使其他样本密度直达。

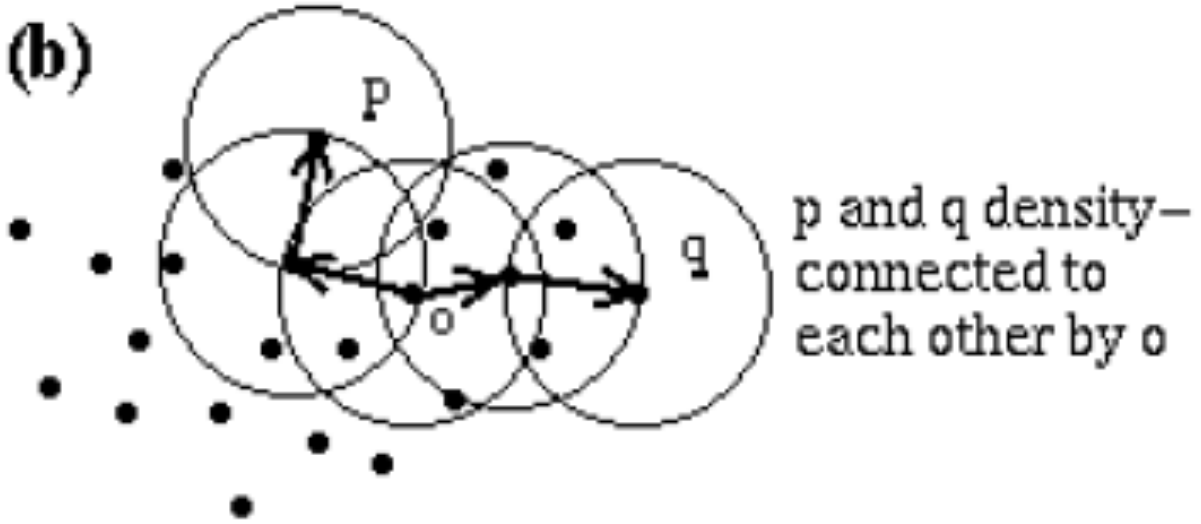
注意密度可达也不满足对称性，这个可以由密度直达的不对称性得出。



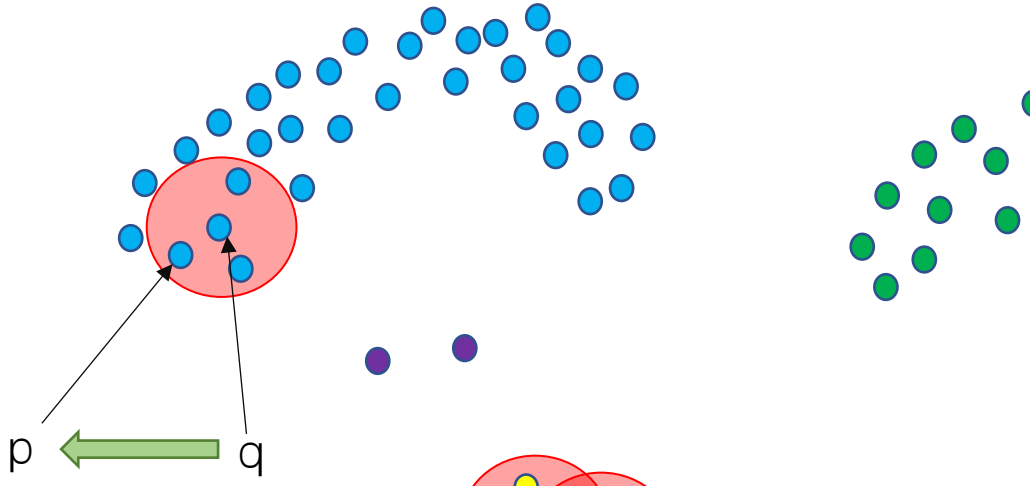


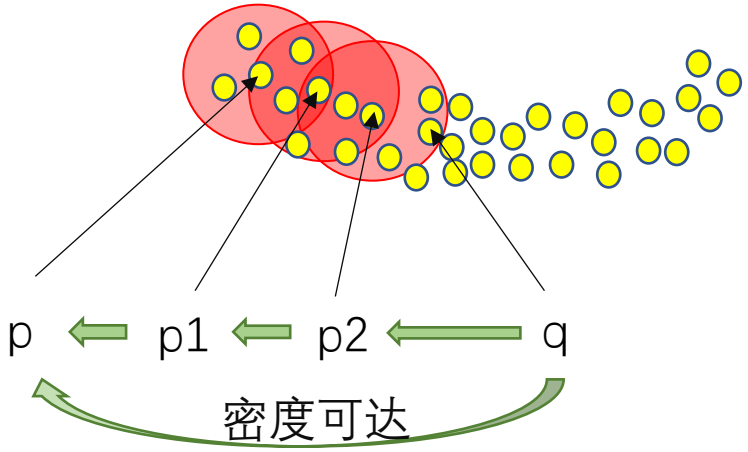
点与点之间的三种关系

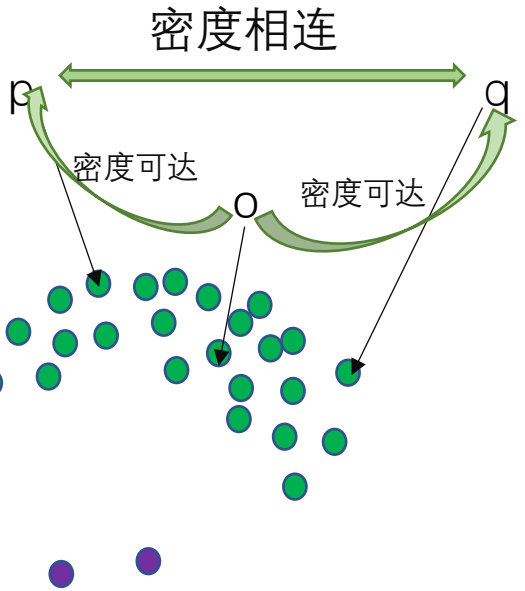
3. 密度相连：对于p和q, 如果存在核心对象样本点o，使点p和点q均由点o密度可达，则称p和q密度相连。注意密度相连关系是满足对称性的。



点与点之间的三种关系



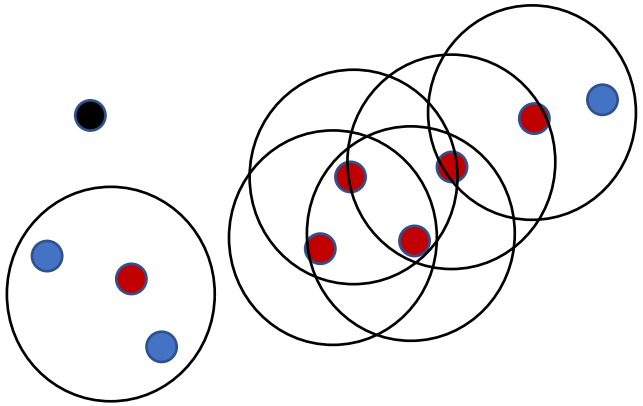




用密度相连的闭包来发现连通的稠密区域作为一个簇。

DBSCAN基本思路

- 选择一个未被标记的核心点p作为一个簇的起点，把其领域$N_{eps}(p)$ 的样本点作为候选集S，把S中未被标记的样本点归入该簇；
- 重复合并S中其他核心点领域内的样本点来扩充后候选集，把S中未被标记的样本点归入该簇，直到候选集S不存在未被标记的样本点。
- 重复以上两步来发现新的簇，直到所有点都被标记为某个簇或噪声。



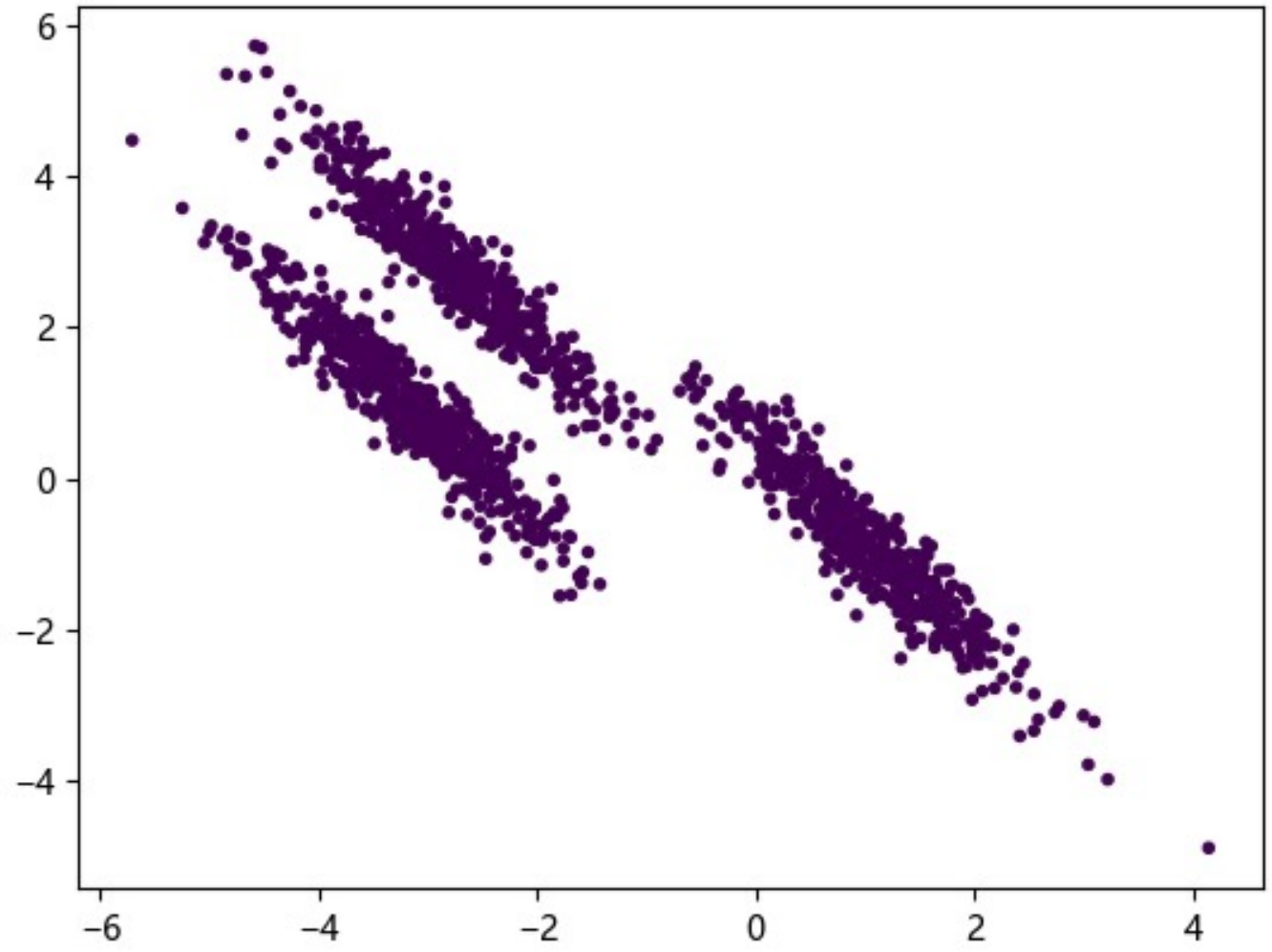
$MinPts$ =3

DBSCAN算法

输入：样本集$D = \{x_1, x_2, x_3, \cdots, x_n\}$，超参数 $(\epsilon, MinPts)$与距离度量方式
输出：聚类划分集合$C = \{C_1, C_2, \cdots, C_k\}$ // 最后划分出k个聚类簇

1. 标记所有对象的label为unvisited;
2. i = 0
3. for p in D:
4. if label(p) != unvisited then continue //找到一个之前没有被访问过的样本点
5. if $|N_{eps}(p)| < MinPts$: //非核心点，暂时标记为噪声
6. label(p) = noise
7. continue
8. i++; //找到一个之前没有被访问过的核心点,每个簇都从核心点开始增长
9. label(p) = C_i
10. set $S = N_{eps}(p) \backslash \{p\}$ // 排除当前点P
11. for each point q in S:
12. if label(q) == noise then label(q) = C_i //重置被访问过的p领域内的非核心点的label
13. if label(q) != unvisited then continue
14. label(q) = C_i //没有被访问过的p领域内的点
15. if $|N_{eps}(q)| \geq MinPts$: then $S = S \cup N_{eps}(q)$ //如果q是核心点，则向其领域扩张

DBSCAN过程演示



DBSCAN超参数

超参数eps与MinPts对结果的影响：

eps越小，MinPts越大，则要求构成同一个簇的样本之间的密度越大；这可能导致很多大的簇被碎片化，同时把处于密度比较小的区域的样本点标记为噪声；

eps越大，MinPts 越小，则要求构成同一个簇的样本之间的密度较小；这可能导致一些距离较近的簇被合并在一起，同时不同识别与簇离得较近的噪声点。

DBSCAN优缺点

主要优点：

1）可以对任意形状的稠密数据集进行聚类，相对而言，K-Means之类的聚类算法一般只适用于凸数据集。

2）可以在聚类过程中发现噪声点，对数据集中的噪声点不敏感。

3）与K-Means算法不同，不会受到初始值的影响。

4）不需要输入划分聚类的个数。

主要缺点：

1）如果样本集的密度不均匀、聚类间距差相差很大时，聚类质量较差，这时对于超参数的选取困难，用DBSCAN聚类一般不适合。

2）数据量增大时，要求的较大的内存支持，I/O消耗也很大，此时可以对搜索最近邻时建立KD树等方法来改进效率。

3）算法聚类效果依赖于距离公式的选取,实际应用中常用欧氏距离,对于高维数据，存在“维数灾难”。