2024-2025学年 第1学期(秋)

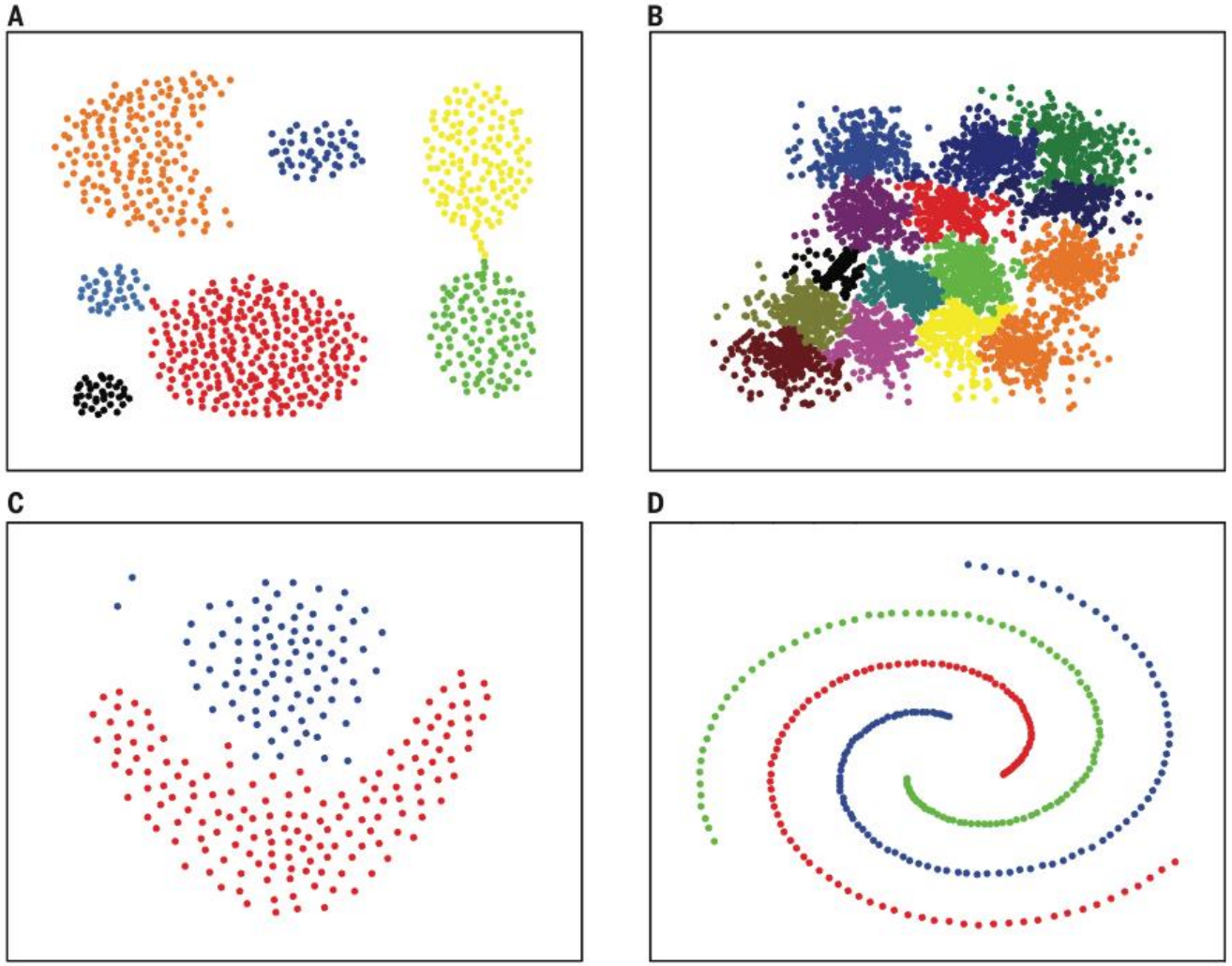
数据挖掘

聚类分析

2024 年 11 月

物以类聚、人以群分

聚类效果



A. Rodriguez and A. Laio, *Science*, 2014

学习目标

- 掌握聚类分析的概念，区分聚类和分类
- 熟悉划分聚类、层次聚类等基本的聚类方法
- 了解聚类评估的任务有哪些

南京大学
NANJING UNIVERSITY

目录

聚类分析概述

- **什么是聚类?**
  - **是把数据对象集合按照相似性划分成多个子集的过程。**
  - **每个子集是一个簇（cluster），使得簇中的对象彼此相似，但与其他簇中的对象不相似。：**
- **聚类是无监督学习：给定的数据没有类标号信息**

分类 vs. 聚类

- 分类
  - 有监督学习
  - 通过有标签样本学习分类器
- 聚类
  - 无监督学习
  - 通过观察学习，将数据分割成多个簇

聚类的应用

- 商业领域
  - 聚类分析被用来发现不同的客户群，并且通过购买模式刻画不同的客户群的特征。
- 电子商务
  - 聚类出具有相似浏览行为的客户，并分析客户的共同特征，可以更好的帮助电子商务的用户了解自己的客户，向客户提供更合适的服务。
- 舆情监控
  - 发现热点主题、话题、事件等
  - 发现未知异常

数据挖掘对聚类的要求

- 可扩展性
  - 许多聚类算法在小于几百个数据对象的小数据集合上工作得很好
  - 而一个大规模数据库可能包含几百万个对象
- 处理不同数据类型的能力
  - 许多聚类算法专门用于数值类型的数据
  - 而实际应用涉及不同的数据类型，如二元的、分类的、图像的等
- 发现任意形状的能力
  - 基于距离的聚类算法往往发现的是球形的聚类
  - 而现实的聚类是任意形状的
- 用于决定输入参数的领域知识最小化
  - 聚类结果对于输入参数十分敏感
  - 而参数很难决定，聚类的质量也很难控制

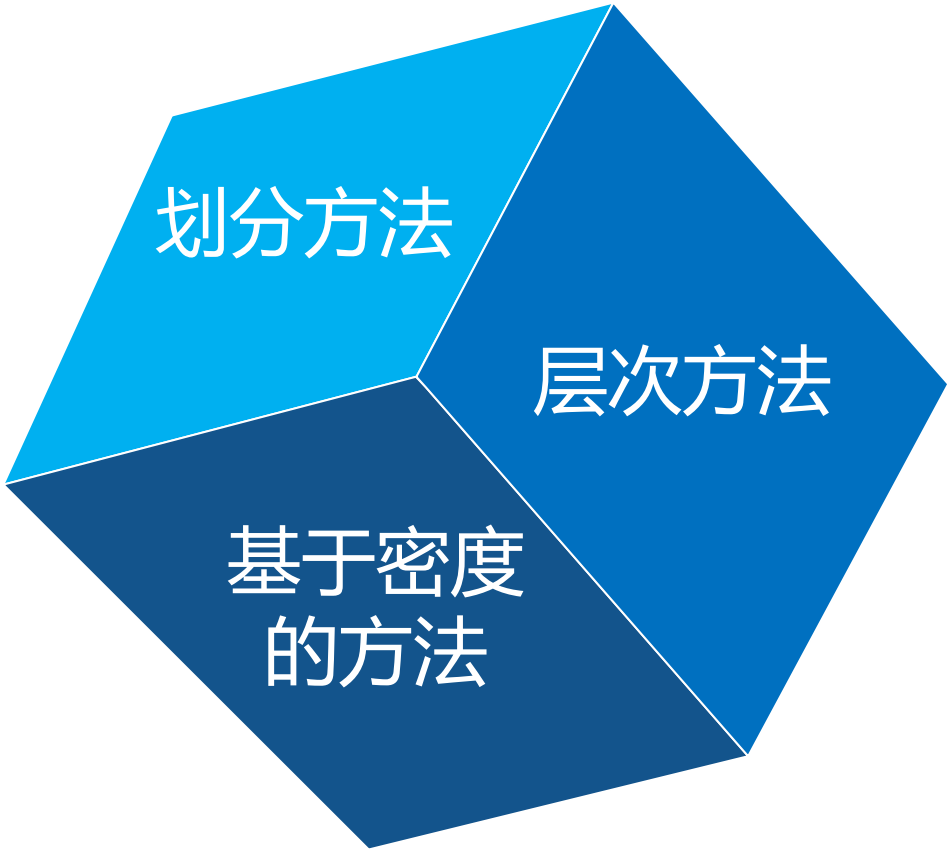
数据挖掘对聚类的要求

- 处理噪声数据的能力
  - 很多数据库都包含了孤立点，缺失或错误的数据
  - 而一些聚类算法对于这样的数据敏感，可能导致低质量的聚类结果
- 对输入数据的顺序不敏感和增量聚类
  - 同一个数据集合，以不同的次序提交给同一个算法，应该产生相似的结果
  - 能将新加入的数据合并到已有聚类中
- 高维度
  - 许多聚类算法擅长处理低维数据，可能只涉及两到三维
  - 而数据库或者数据仓库可能包含若干维或属性

南京大学
NANJING UNIVERSITY

目录

基本聚类方法



划分方法

- **划分方法：将有n 个对象的数据集D划分成k个簇，并且k≤n，满足如下的要求：**
  - **每个簇至少包含一个对象**
  - **每个对象属于且仅属于一个簇**
- **基本思想**
  - **首先创建一个初始k划分(k为要构造的划分数)**
  - **然后不断迭代地计算各个簇的聚类中心并依新的聚类中心调整聚类情况，直至收敛**
- **目标**
  - **同一个簇中的对象之间尽可能“接近”或相关**
  - **不同簇中的对象之间尽可能“远离”或不同**

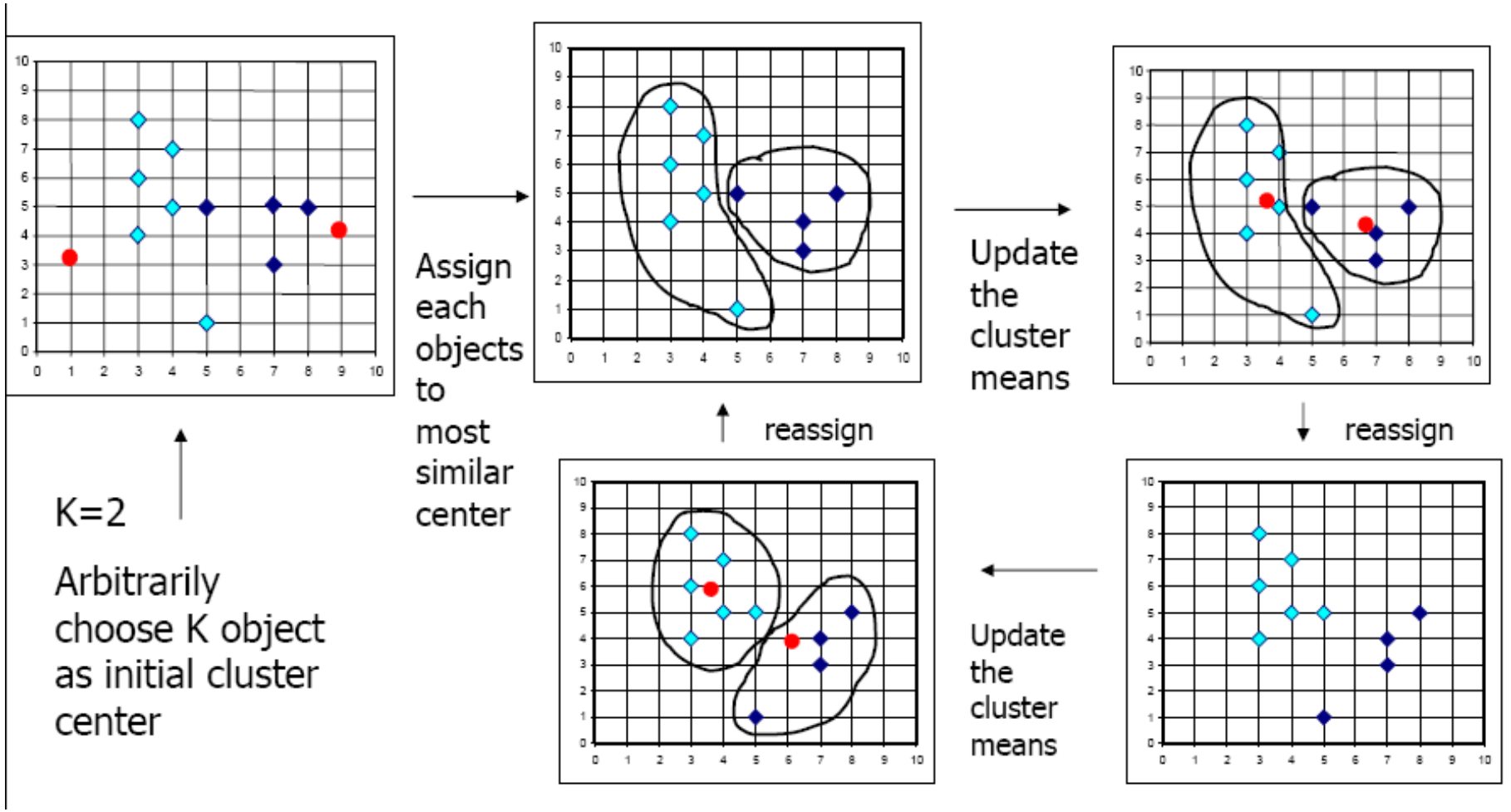
划分方法

- **启发式方法** $E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (d(p, c_i))^2$
  - **k-均值(k-means)**
    - 每个簇用该簇中对象的均值来表示
    - 基于质心的技术
  - **k-中心点(k-medoids)**
    - 每个簇用接近簇中心的一个对象来表示
    - 基于代表对象的技术
- **适用性**
  - **这些启发式算法适合发现中小规模数据库中的球状聚类**
  - **对于大规模数据库和处理任意形状的聚类，这些算法需要进一步扩展**

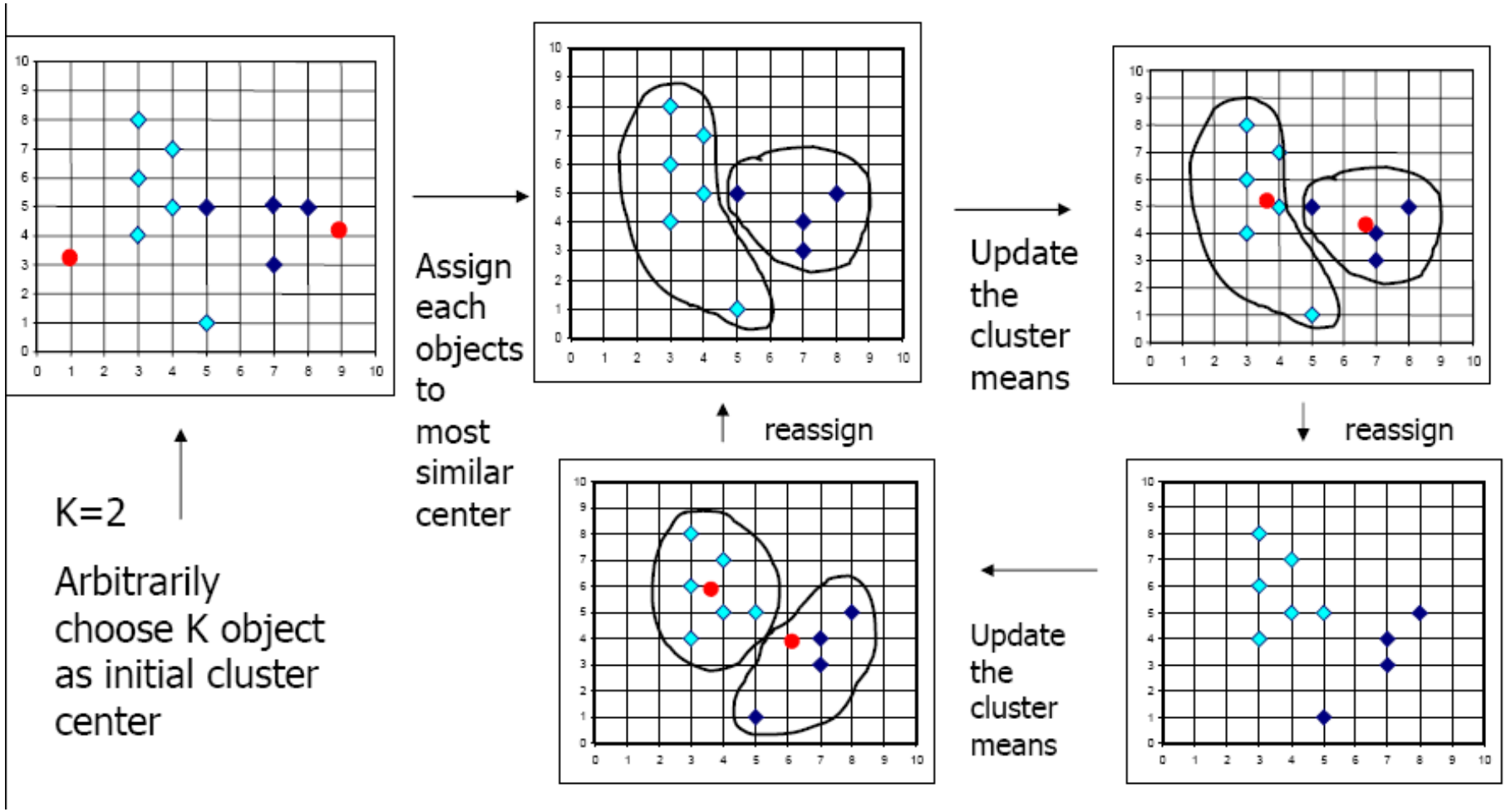
K-means

1. 从数据集中随机取K个样本作为初始的聚类中心$C = \{c_1, c_2, \dots, c_k\}$
2. 针对数据集中每个样本x_i，计算它到K个聚类中心的距离并将其分到距离最小的聚类中心对应的类中
3. 针对每个类别c_i，重新计算其聚类中心$c_i = \frac{1}{|c_i|}\sum_{x \in c_i} x$
4. 重复第2和第3步骤，直到$\sum_{i=0}^{n} \min_{c_j \in C}(\| x_j - c_i \|^2)$小于特定的阈值

K-means


K=2
Arbitrarily choose K object as initial cluster center
Assign each objects to most similar center
reassign
Update the cluster means
reassign
Update the cluster means

K-means



K-means算法为启发式算法，遵循的寻优原则：

每次聚类保证局部最优，随后调整聚类，利用局部最优聚类的上限来不断逼近全局最优

k-means计算示例

- **假设：给定如下要进行聚类的对象：**

  {2，4，10，12，3，20，30，11，25}，k = 2，请使用k均值划分聚类

- **步骤如下：**

m1	m2	K1	K2
2	4	{2,3}	{4,10,12,20,30,11,25}
2.5	16	{2,3,4}	{10,12,20,30,11,25}
3	18	{2,3,4,10}	{10,12,20,30,11,25}
4.75	19.6	{2,3,4,10,11,12}	{20,30,25}
7	25	{2,3,4,10,11,12}	{20,30,25}

k-means算法效率分析

- 算法的计算复杂度为O(N*K*T)
- 其中

  N: 为数据集中对象的数目

  K: 为期望得到的簇的数目

  T: 为迭代的次数
- 在处理大数据库时也是相对有效的(可扩展性)

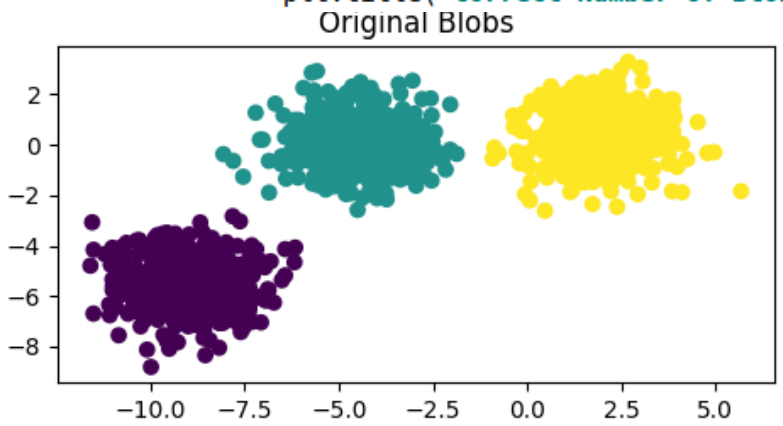
k-means优缺点

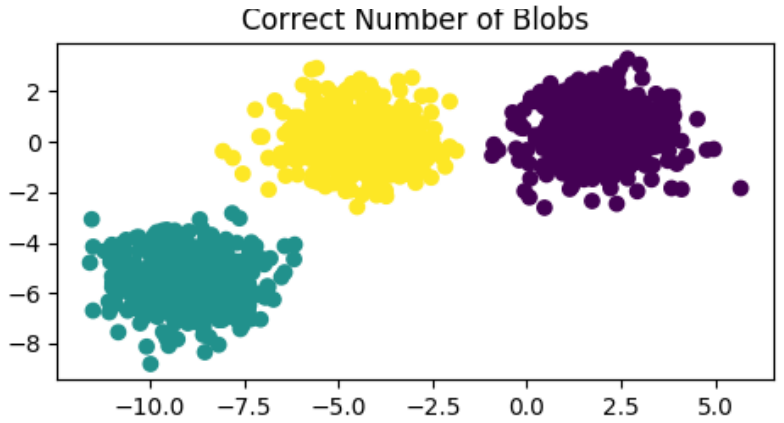
- **优点**
  - 聚类时间快
  - 当结果簇是密集的，而簇与簇之间区别明显时，效果较好
  - 相对可扩展和有效，能对大数据集进行高效划分
- **缺点**
  - 用户必须事先指定聚类簇的个数
  - 常常终止于局部最优
  - 只适用于数值属性聚类(计算均值有意义)
  - 对噪声和异常数据也很敏感
  - 不同的初始值，结果可能不同
  - 不适合发现非凸面形状的簇

K-means的问题

1. K-means中初始簇规模估计正确

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

plt.figure(figsize=(12, 12))

n_samples = 1500
random_state = 170
X, y = make_blobs(n_samples=n_samples, random_state=random_state)

plt.subplot(321)
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.title("Original Blobs")


y_pred = KMeans(n_clusters=3, random_state=random_state).fit_predict(X)

plt.subplot(322)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.title("Correct Number of Blobs")
```

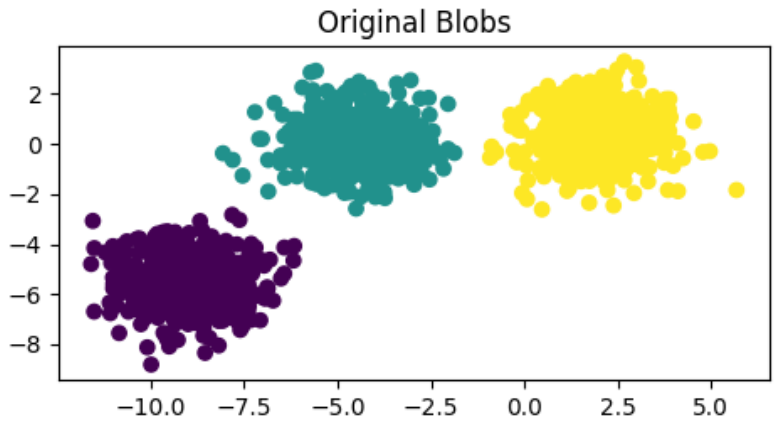


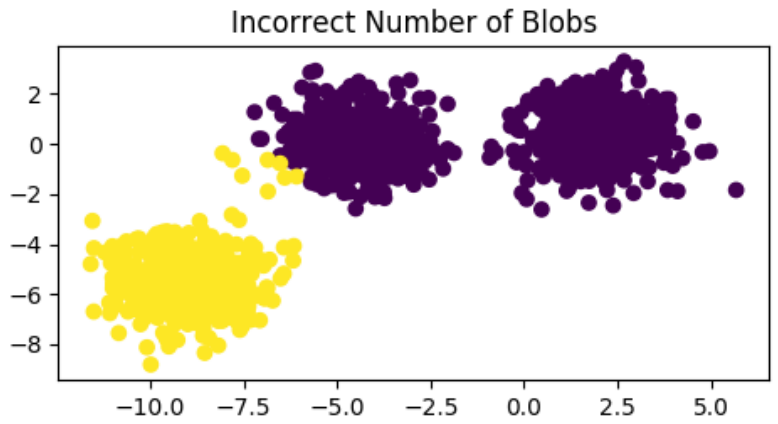


K-means的问题

2. K-means中初始簇规模估计错误

```
n_samples = 1500
random_state = 170
X, y = make_blobs(n_samples=n_samples, random_state=random_state)

plt.subplot(321)
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.title("Original Blobs")

# Incorrect number of clusters
y_pred = KMeans(n_clusters=2, random_state=random_state).fit_predict(X)

plt.subplot(322)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.title("Incorrect Number of Blobs")
```

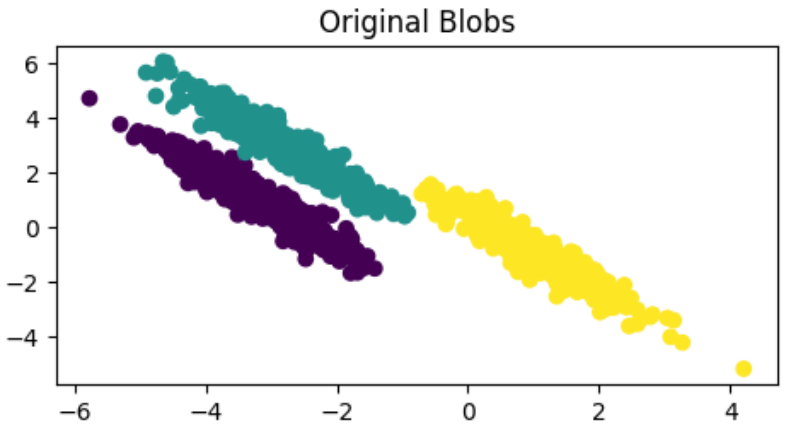


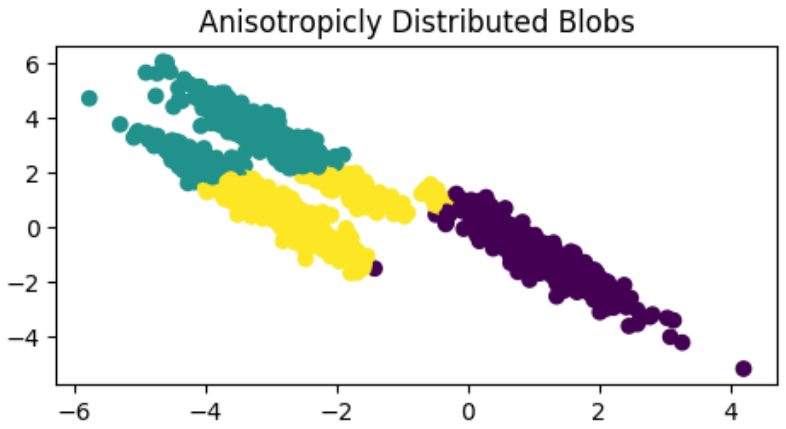


K-means的问题

3. 数据分布形状对分簇结果的影响

```
X, y = make_blobs(n_samples=n_samples, random_state=random_state)
# Anisotropicly distributed data
transformation = [[0.60834549, -0.63667341], [-0.40887718, 0.85253229]]
X_aniso = np.dot(X, transformation)

plt.subplot(321)
plt.scatter(X_aniso[:, 0], X_aniso[:, 1], c=y)
plt.title("Original Blobs")

y_pred = KMeans(n_clusters=3, random_state=random_state).fit_predict(X_aniso)

plt.subplot(322)
plt.scatter(X_aniso[:, 0], X_aniso[:, 1], c=y_pred)
plt.title("Anisotropicly Distributed Blobs")
```

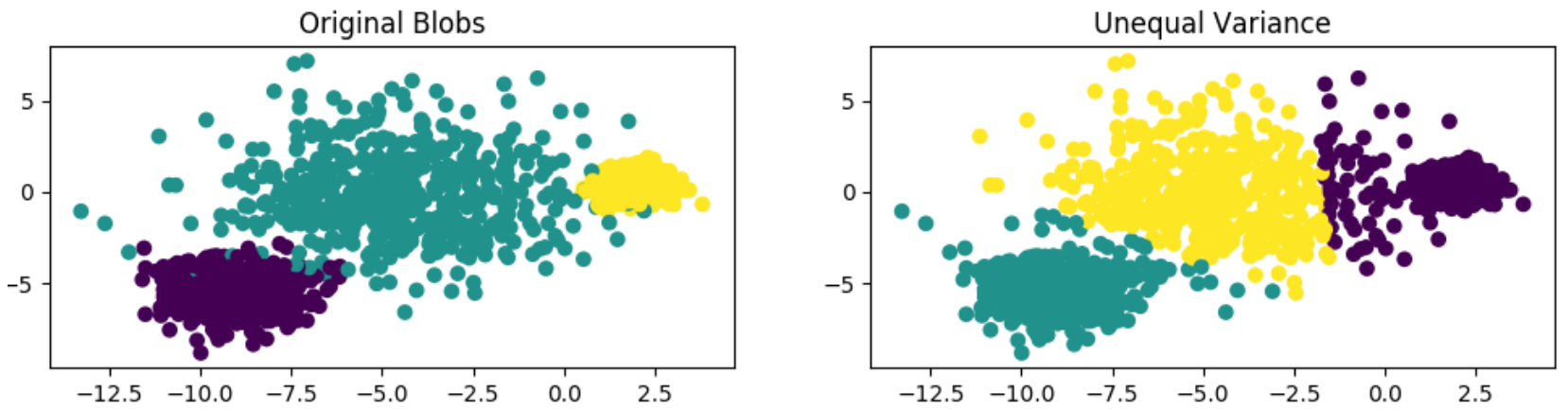




K-means的问题

4. 数据分散程度对分簇结果的影响

```
# Different variance
X_varied, y_varied = make_blobs(n_samples=n_samples,
                                cluster_std=[1.0, 2.5, 0.5],
                                random_state=random_state)
plt.subplot(321)
plt.scatter(X_varied[:, 0], X_varied[:, 1], c=y_varied)
plt.title("Original Blobs")

y_pred = KMeans(n_clusters=3, random_state=random_state).fit_predict(X_varied)

plt.subplot(322)
plt.scatter(X_varied[:, 0], X_varied[:, 1], c=y_pred)
plt.title("Unequal Variance")
```



K-means的问题

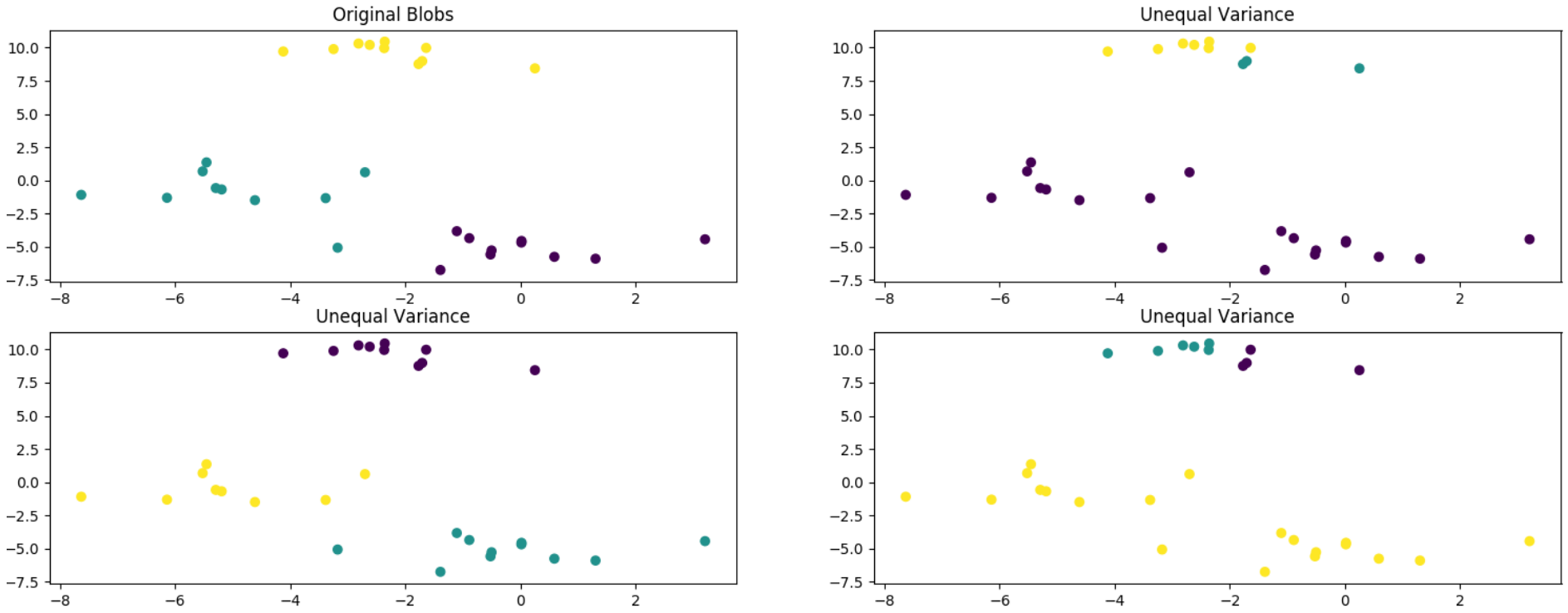
5. 随机初始种子的影响

```
n_samples = 30
random_state = 50
X, y = make_blobs(n_samples=n_samples, random_state=random_state)

# Different variance
X_varied, y_varied = make_blobs(n_samples=n_samples,
                                cluster_std=[1, 2, 1],
                                random_state=random_state)
plt.subplot(321)
plt.scatter(X_varied[:, 0], X_varied[:, 1], c=y)
plt.title("Original Blobs")

y_pred = KMeans(init='random',n_clusters=3, n_init=1).fit_predict(X_varied)
plt.subplot(322)
plt.scatter(X_varied[:, 0], X_varied[:, 1], c=y_pred)
plt.title("Unequal Variance")


y_pred = KMeans(init='random',n_clusters=3, n_init=1).fit_predict(X_varied)
plt.subplot(323)
plt.scatter(X_varied[:, 0], X_varied[:, 1], c=y_pred)
plt.title("Unequal Variance")

y_pred = KMeans(init='random',n_clusters=3, n_init=1).fit_predict(X_varied)
plt.subplot(324)
plt.scatter(X_varied[:, 0], X_varied[:, 1], c=y_pred)
plt.title("Unequal Variance")
```

K-means的问题

5. 随机初始种子的影响

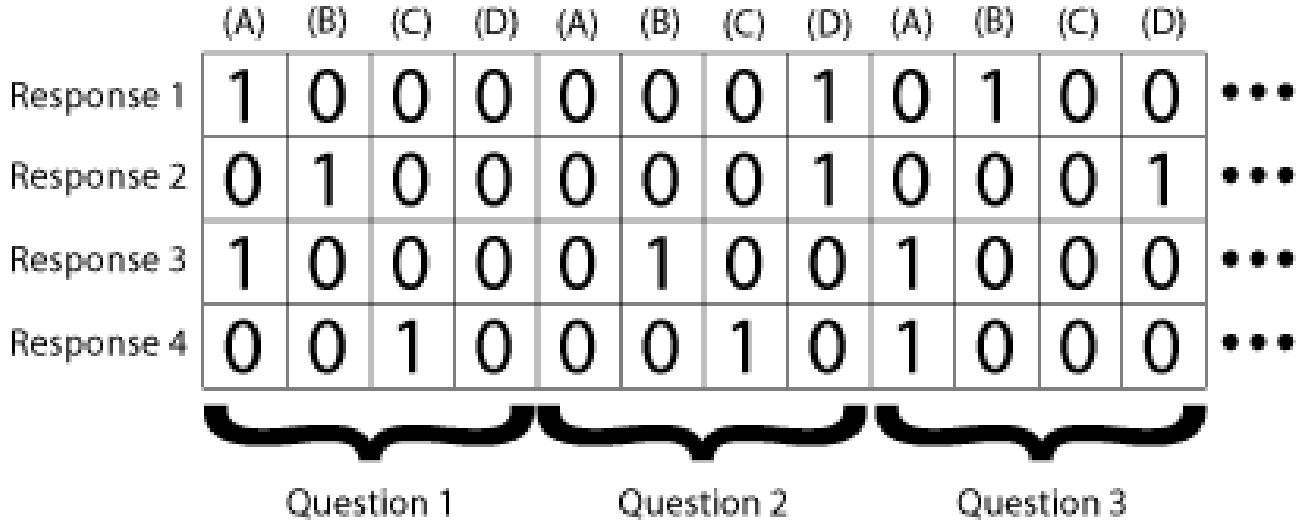
```
n_samples = 30
random_state = 50
X, y = make_blobs(n_samples=n_samples, random_state=random_state)
```




```
y_pred = KMeans(init='random',n_clusters=3, n_init=1).fit_predict(X_varied)
plt.subplot(324)
plt.scatter(X_varied[:, 0], X_varied[:, 1], c=y_pred)
plt.title("Unequal Variance")
```

K-modes算法 —— 解决数据敏感的问题

- K-means的改进算法主要区别在于：
  - 初始k均值选择
  - 相异度计算
  - 计算均值方法
- 处理分类变量: K-modes
  - 针对分类数据
  - 用众数代替均值
  - 使用新的相异性度量



K-means++算法 —— 解决初始点选择问题

- **基本原理**

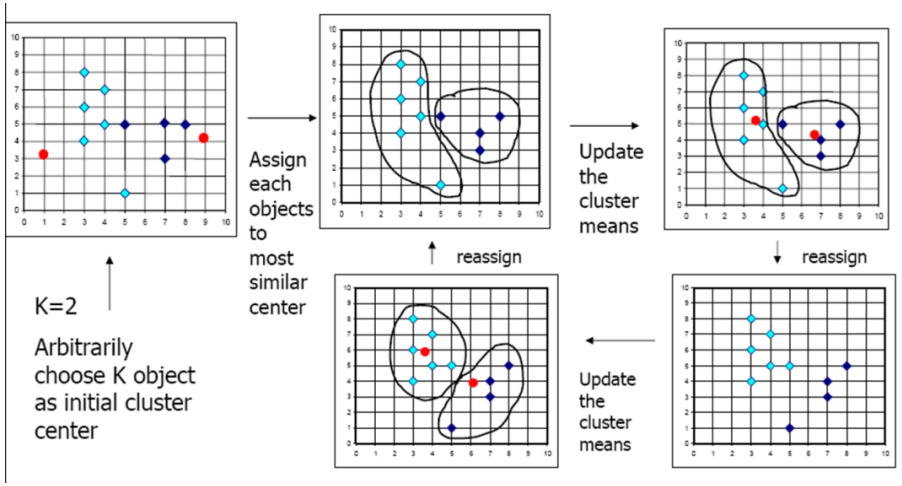
① 从输入的数据点集合中随机选择一个点作为第一个聚类中心

② 对于数据集中的每一个点X，计算其与聚类中心的距离D（X）

③ 选择一个D（X）最大的点作为新的聚类中心

④ 重复2和3步直到K个聚类中心被选出

⑤ 利用K个初始聚类中心运行K-Means



K-means++算法 —— 解决初始点选择问题

- **基本原理**

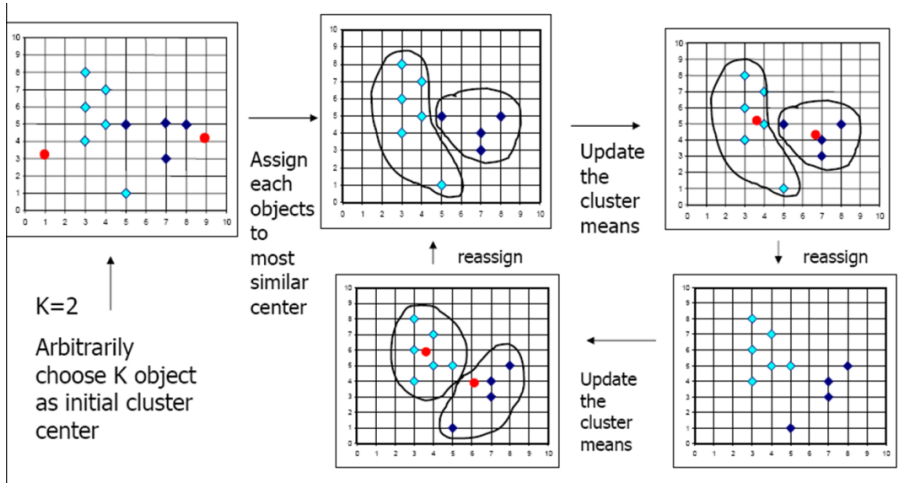
① 从输入的数据点集合中随机选择一个点作为第一个聚类中心

② 对于数据集中的每一个点X，计算其与聚类中心的距离D（X）

③ 选择一个D（X）最大的点作为新的聚类中心

④ 重复2和3步直到K个聚类中心被选出

⑤ 利用K个初始聚类中心运行K-Means

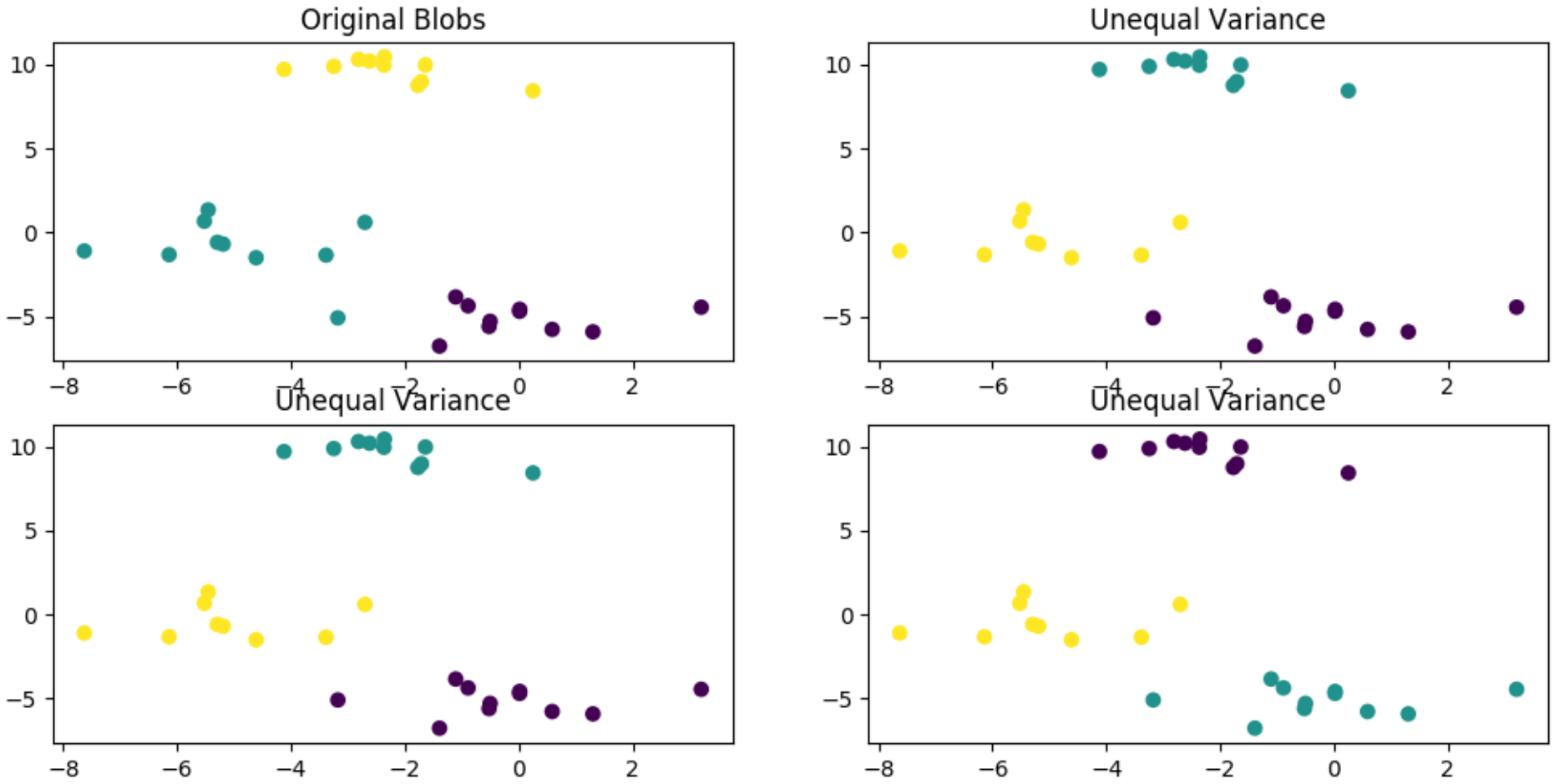


尽可能选择距离远的点来作为初始种子节点

K-means++算法 —— 解决初始点选择问题

5. 随机初始种子的影响

```
n_samples = 30
random_state = 50
X, y = make_blobs(n_samples=n_samples, random_state=random_state)
```



k-中心点方法 —— 解决对离群点敏感问题

一维空间的7个点 1 2 3 8 9 10 25 离群点

k-中心点方法 —— 解决对离群点敏感问题

一维空间的7个点 1 2 3 8 9 10 25 离群点

设置k=2，直觉应该划分为 (1,2,3) (8,9,10,25) 两个类别

k-中心点方法 —— 解决对离群点敏感问题

一维空间的7个点 1 2 3 8 9 10 25 离群点

设置k=2，直觉应该划分为 (1,2,3) (8,9,10,25) 两个类别

划分方法聚类质量评价准则：最小化E值

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (d(p, c_i))^2$$

k-中心点方法 —— 解决对离群点敏感问题

一维空间的7个点 1 2 3 8 9 10 25 离群点

设置k=2，直觉应该划分为 (1,2,3) (8,9,10,25) 两个类别

划分方法聚类质量评价准则：最小化E值

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (d(p, c_i))^2$$

(1,2,3) (8,9,10,25) 聚类的E值为：

$(1-2)^2+ (2-2)^2 + (3-2)^2 + (8-13)^2 + (9-13)^2 + (10-13)^2 + (25-13)^2 =196$

(1,2,3,8) (9,10,25) 的E值为：

$(1-3.5)^2+ (2-3.5)^2 + (3-3.5)^2 + (8-3.5)^2 + (9-14.67)^2 + (10-14.67)^2 + (25-14.67)^2 =189.67$

k-中心点方法 —— 解决对离群点敏感问题

一维空间的7个点 1 2 3 8 9 10 25 离群点

设置k=2，直觉应该划分为 (1,2,3) (8,9,10,25) 两个类别

划分方法聚类质量评价准则：最小化E值

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (d(p, c_i))^2$$

(1,2,3) (8,9,10,25) 聚类的E值为：

$(1-2)^2+ (2-2)^2 + (3-2)^2 + (8-13)^2 + (9-13)^2 + (10-13)^2 + (25-13)^2 =196$

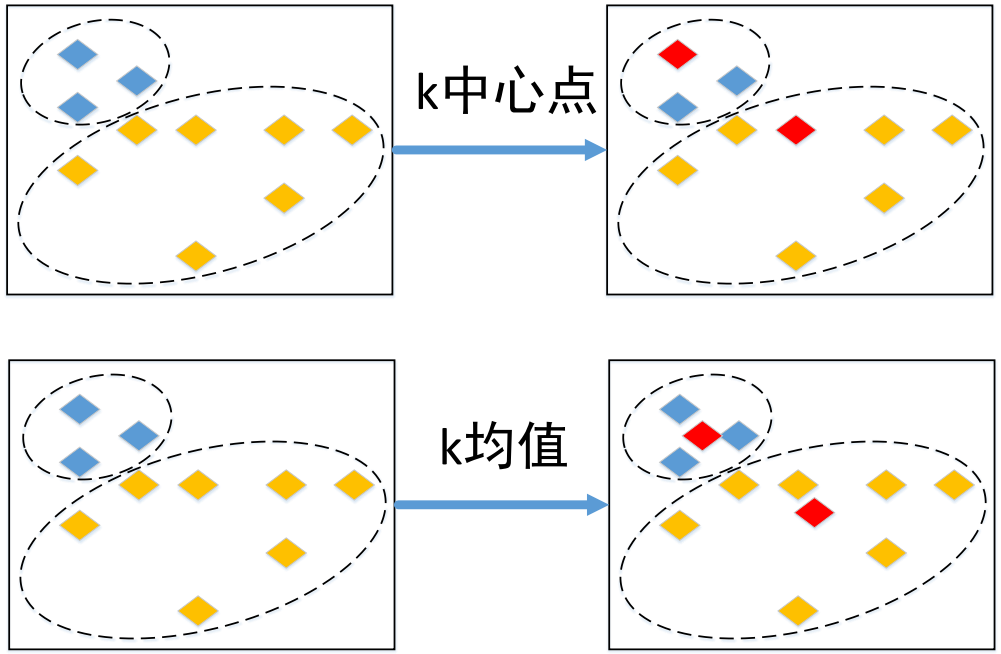
(1,2,3,8) (9,10,25) 的E值为：

$(1-3.5)^2+ (2-3.5)^2 + (3-3.5)^2 + (8-3.5)^2 + (9-14.67)^2 + (10-14.67)^2 + (25-14.67)^2 =189.67$

k-中心点

- k-中心点
  - 选用簇中位置最中心的实际对象即中心点作为参照点
  - 基于最小化所有对象与其参照点之间的相异度之和的原则来划分(使用绝对误差标准)

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} | p - o_i |$$



k-中心点方法

- **基本流程**
  - 首先为每个簇随意选择一个代表对象，剩余的对象根据其与代表对象的距离分配给最近的一个簇
  - 然后迭代地用非代表对象来替代代表对象，以改进聚类的质量（找更好的代表对象）
  - 聚类结果的质量用一个代价函数来估算，该函数评估了对象与其参照对象之间的平均相异度

PAM算法

- **PAM算法(Partitioning Around Medoids)**
- **最早提出的K-中心点算法之一**

① 随机选择**k** 个对象作为初始的代表对象

Repeat

① 指派每个剩余的对象给离它最近的代表对象所代表的簇

② 随机地选择一个非代表对象O_{random}

③ 计算用O_{random}代替O_j的**总代价S**

④ 如果**S<0**，则用O_{random}替换O_j形成新的**k**个代表对象的集合**u**

Until不发生变化

代价函数

- 代价函数

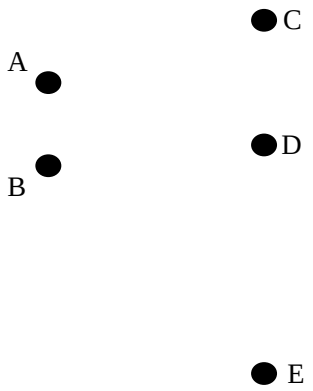
$$TC_{jh} = \sum_{p=1}^{n} C_{pjh}$$

其中：n是数据集中样本的个数；C_{pjh}表示中心点O_j被非中心点O_h替代后，样本点p的代价。

问题：如何计算每个样本点p产生的代价C_{pjh}？

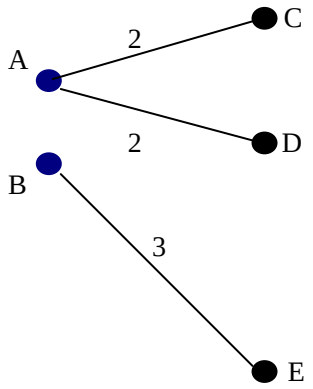
PAM算法示例

假如空间中的五个点｛A、 B、 C、 D、 E｝如图所示，各点之间的距离关系如表所示，根据所给的数据对其运行PAM算法实现划分聚类（设k=2）。 样本点间距离如下表所示：



样本点

样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



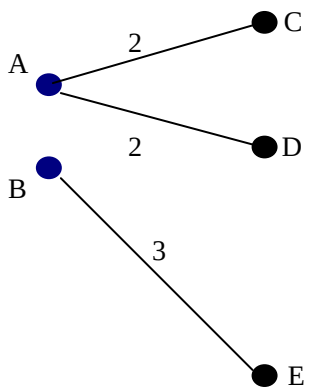
起始中心点

PAM算法示例

第一步 建立阶段：假如从5个对象中随机抽取的2个中心点为{A，B},则样本被划分为{A、C、D}和{B、E}

第二步 交换阶段：假定中心点A、B分别被非中心点C、D、E替换，根据PAM算法需要计算下列代价TC_{AC}、 TC_{AD}、 TC_{AE}、TC_{BC}、TC_{BD}、 TC_{BE}，即验证中心点变换后，代价如何变化

- 以TC_{AC}为例说明计算过程



起始中心点

a) 当A被C替换以后，A不再是一个中心点，因为A离B比A离C近，A被分配到B中心点代表的簇，$C_{AAC}=d(A,B)-d(A,A)=1$

b) B是一个中心点，当A被C替换以后，B不受影响，$C_{BAC}=0$

样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

PAM算法示例

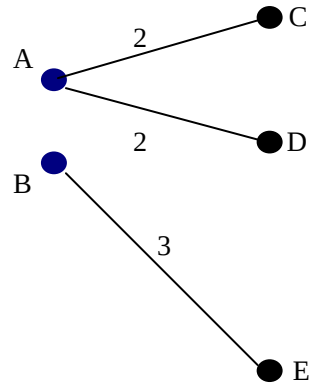
a) $C_{AAC}=d(A,B)-d(A,A)=1$

b) $C_{BAC}=0$

c) C原先属于A中心点所在的簇，当A被C替换以后，C是新中心点，$C_{CAC}=d(C,C)-d(C,A)=0-2=-2$

d) D原先属于A中心点所在的簇，当A被C替换以后，离D最近的中心点是C，$C_{DAC}=d(D,C)-d(D,A)=1-2=-1$

e) E原先属于B中心点所在的簇，当A被C替换以后，离E最近的中心仍然是 B，根据PAM算法代价函数的第三种情况$C_{EAC}=0$



样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

因此，$TC_{AC}=C_{AAC}+C_{BAC}+C_{CAC}+C_{DAC}+C_{EAC}=1+0-2-1+0=-2$

代价函数变小，说明C点替换以前的中心点A，聚类更加紧凑，那么C作为中心点更加合理

PAM算法示例

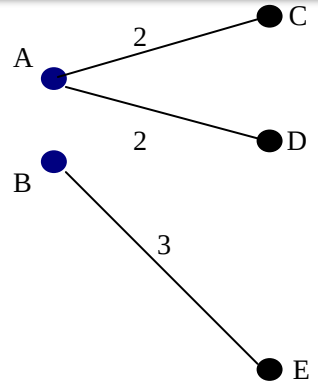
a) $C_{AAC}=d(A,B)-d(A,A)=1$

b) $C_{BAC}=0$

c) C原先属于A中心点所在的簇，当A被C替换以后，C是新中心点，$C_{CAC}=d(C,C)-d(C,A)=0-2=-2$

d) D原先属于A中心点所在的簇，当A被C替换以后，离D最近的中心点是C，$C_{DAC}=d(D,C)-d(D,A)=1-2=-1$

e) E原先属于B中心点所在的簇，当A被C替换以后，离E最近的中心仍然是 B，根据PAM算法代价函数的第三种情况$C_{EAC}=0$



样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

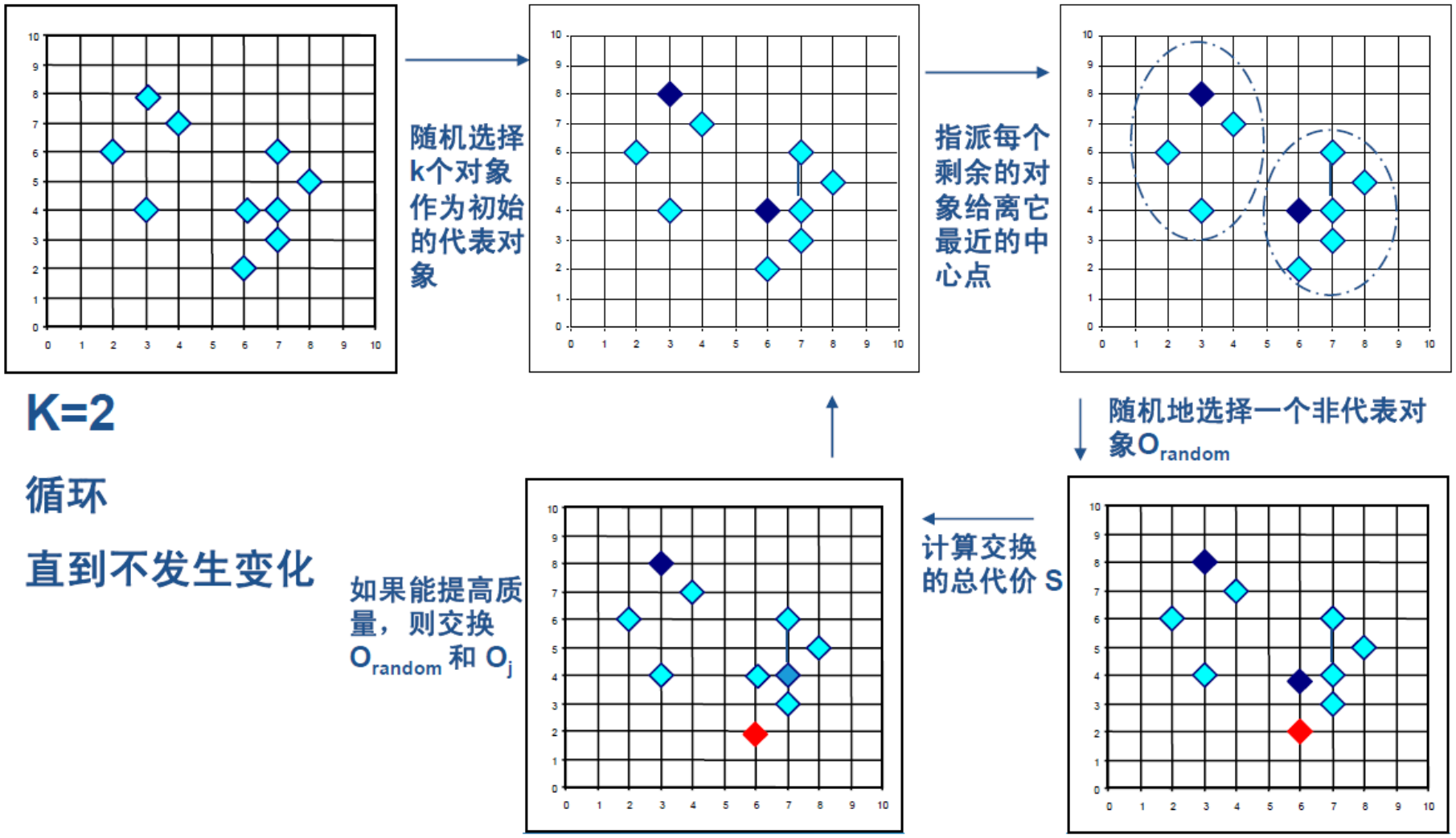
因此，$TC_{AC}=C_{AAC}+C_{BAC}+C_{CAC}+C_{DAC}+C_{EAC}=1+0-2-1+0=-2$

代价函数变小，说明C点替换以前的中心点A，聚类更加紧凑，

那么C作为中心点更加合理

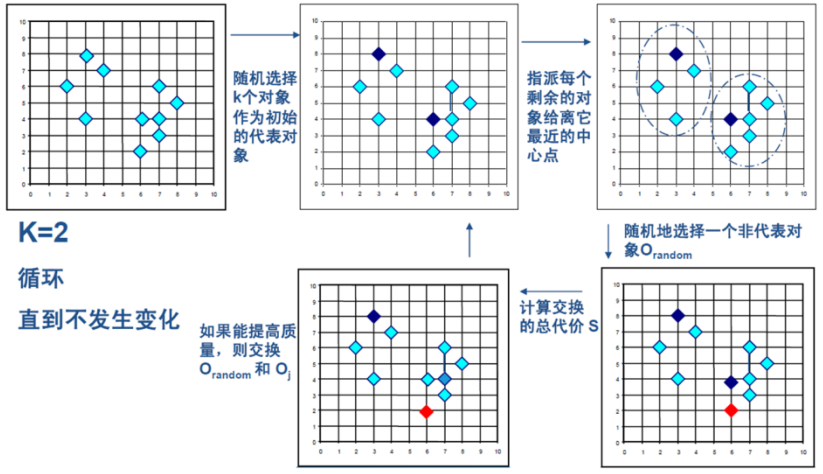
同理计算TC_{AC}、TC_{AD}、TC_{AE}、TC_{BC}、TC_{BD}、TC_{BE}，代价函数最小值作为新中心点

PAM算法示意


随机选择k个对象作为初始的代表对象
指派每个剩余的对象给离它最近的中心点
随机地选择一个非代表对象O_{random}
计算交换的总代价 S
如果能提高质量，则交换 O_{random} 和 O_j
K=2
循环
直到不发生变化

PAM算法

- 优点
  - 当存在噪音和孤立点时，PAM 比k-均值方法更健壮
- 缺点
  - PAM 对于较小的数据集非常有效，但不能很好地扩展到大型数据集，原因是计算复杂度较高
    - 每次中心点交换的时候，就要计算数据中每个点的代价



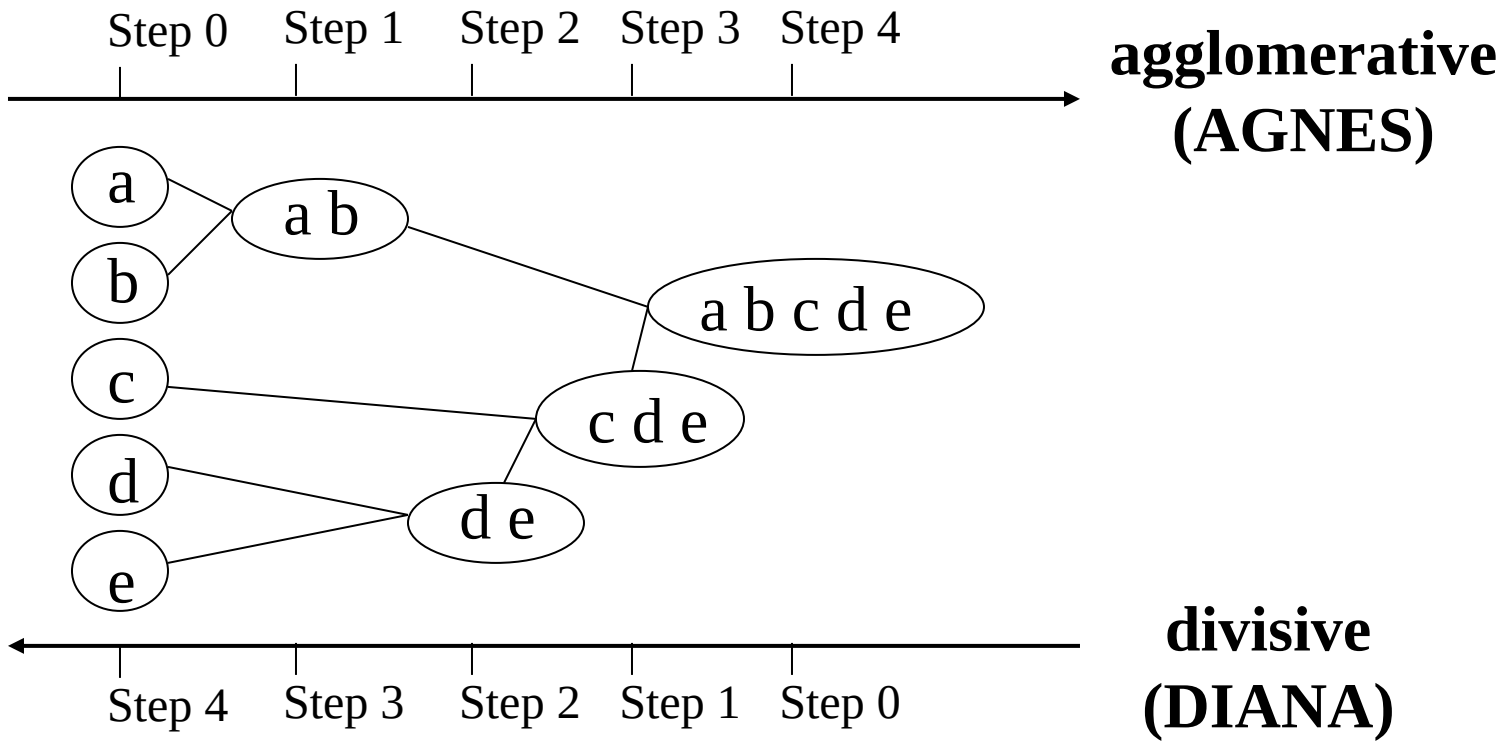
- PAM算法中，每一次迭代都需要确定出**k(n-k)**个交换对
- 对于交换对i，h需要计算**Tc^{ih}**
- 在计算TC^{ih}过程中，需要计算**n-k**个非中心点的代价
- 所以每次迭代的复杂度是**$k(n-k)^2$**

层次方法

- 层次方法
  - 对给定数据对象集进行层次的分解
  - 使用距离矩阵作为聚类标准
  - 不需要输入聚类数目k，但需要终止条件
- 两种层次方法
  - 自底向上方法（凝聚）
    - 初始将每个对象作为单独的一个簇，然后相继的合并相近的对象或簇，直到所有的簇合并为一个，或者达到一个终止条件
    - 代表算法：AGNES算法
  - 自顶向下方法（分裂）
    - 初始将所有的对象置于一个簇中，在迭代的每一步，一个簇被分裂为多个更小的簇，直到最终每个对象在一个单独的簇中，或达到一个终止条件
    - 代表算法：DIANA算法

层次聚类过程图示

- 凝聚的(agglomerative)和分裂的(divisive)层次聚类图示



AGNES算法

- AGNES (Agglomerative Nesting)算法
  - 首先，将数据集中的每个样本作为一个簇；
  - 然后，根据某些准则将这些簇逐步合并；
  - 合并的过程反复进行，直至不能再合并或者达到结束条件为止。
- 合并准则
  - 每次找到距离最近的两个簇进行合并。
  - 两个簇之间的距离由这两个簇中距离最近的样本点之间的距离来表示。

AGNES算法

AGNES算法（自底向上合并算法）

输入：包含n个样本的数据集，终止条件簇的数目k。

输出：k个簇，达到终止条件规定的簇的数目。

(1) 初始时，将每个样本当成一个簇；

(2) REPEAT

根据不同簇中最近样本间的距离找到最近的两个簇；

合并这两个簇，生成新的簇的集合；

(3) UNTIL 达到定义的簇的数目。算法终止条件是什么？

AGNES算法

AGNES算法（自底向上合并算法）

输入：包含n个样本的数据集，终止条件簇的数目k。

输出：k个簇，达到终止条件规定的簇的数目。

(1) 初始时，将每个样本当成一个簇；

(2) REPEAT

根据不同簇中最近样本间的距离找到最近的两个簇；

合并这两个簇，生成新的簇的集合；

(3) UNTIL 达到定义的簇的数目。

算法终止条件是什么？

（1）指定簇的数目k

AGNES算法

AGNES算法（自底向上合并算法）

输入：包含n个样本的数据集，终止条件簇的数目k。

输出：k个簇，达到终止条件规定的簇的数目。

(1) 初始时，将每个样本当成一个簇；

(2) REPEAT

根据不同簇中最近样本间的距离找到最近的两个簇；

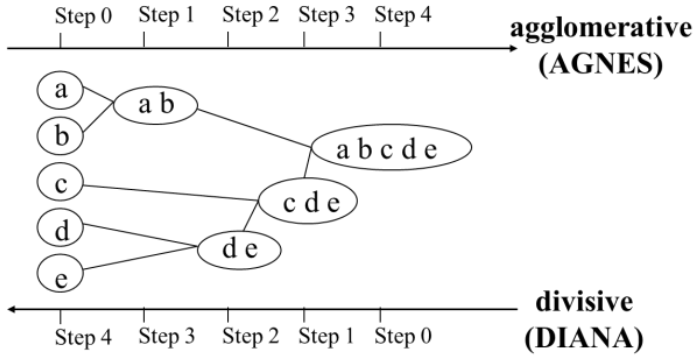
合并这两个簇，生成新的簇的集合；

(3) UNTIL 达到定义的簇的数目。

算法终止条件是什么？

（1）指定簇的数目k

（2）簇之间的距离超过一定阈值



AGNES算法

AGNES算法（自底向上合并算法）

输入：包含n个样本的数据集，终止条件簇的数目k。

输出：k个簇，达到终止条件规定的簇的数目。

(1) 初始时，将每个样本当成一个簇；

(2) REPEAT

簇之间的距离如何计算?

根据不同簇中最近样本间的距离找到最近的两个簇；

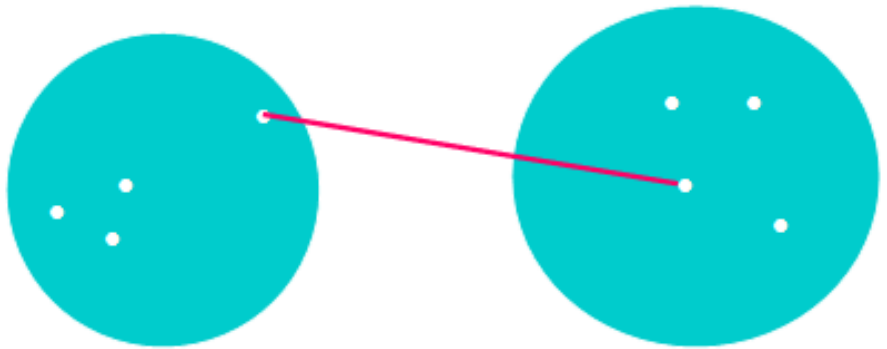
合并这两个簇，生成新的簇的集合；

(3) UNTIL 达到定义的簇的数目。

AGNES算法 —— 最小距离

- **单链接**
  - 其每个簇可以用簇中所有对象代表，簇间的相似度用属于不同簇中**最近的**数据点对之间的相似度来度量
  - 也称为**最短距离法**，定义簇的邻近度为取自不同簇的所有点对的俩个最近的点之间的邻近度
  - 设d_{ij}表示样本$X_{(i)}$和$X_{(j)}$之间的距离，D_{ij}表示类G_i和G_j之间距离

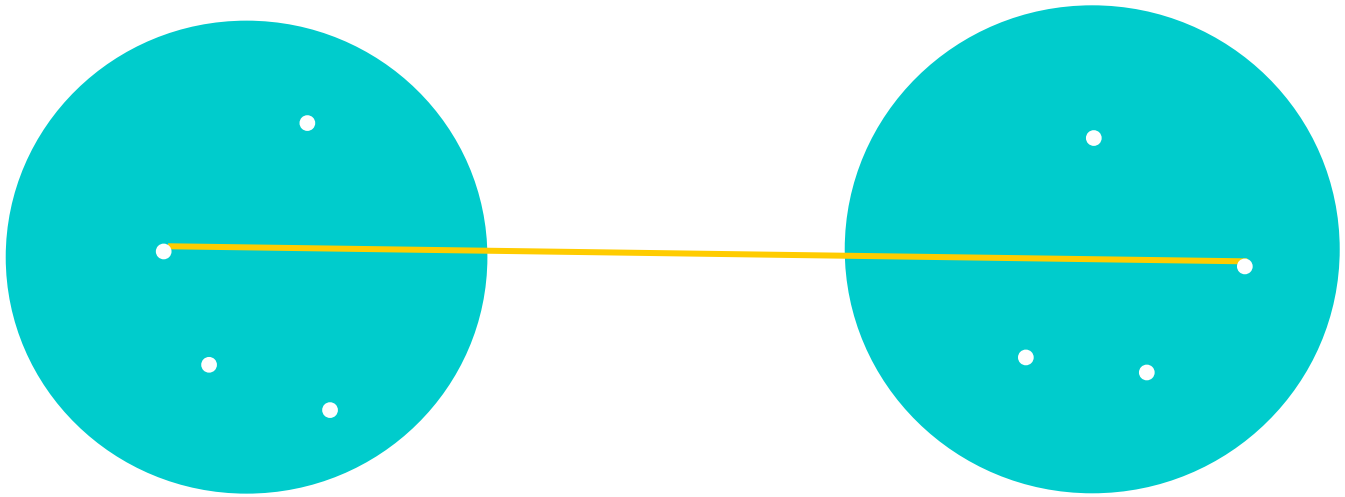
$$D_{ij} = \min_{X_{(i)} \in D_i, X_{(j)} \in D_j} d_{ij}$$



AGNES算法 —— 最大距离

- **全链接**
  - 取自不同簇中的两个最远的点之间邻近度作为簇的邻近度，或者使用图的术语，不同的结点子集中俩个结点之间的最长边

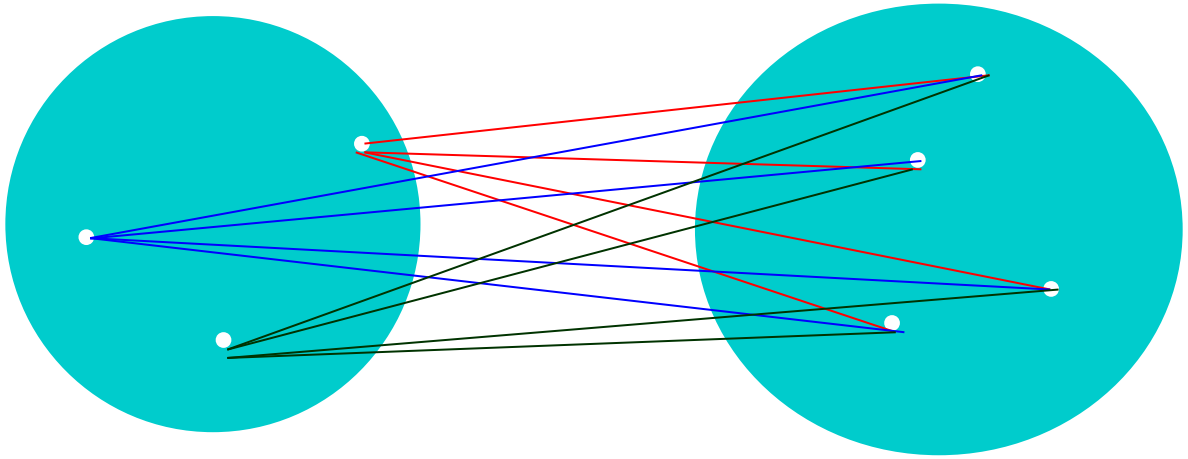
$$D_{ij} = \max_{X_{(i)} \in G_i, X_{(j)} \in G_j} d_{ij}$$



AGNES算法 —— 平均距离

- **平均链接**
  - 类间所有样本点的平均距离
  - 该法利用了所有样本的信息，被认为是较好的系统聚类法

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} \|p - p'\|$$



层次方法的问题及改进

- **层次聚类存在的主要问题**
  - **合并或分裂的决定需要检查和估算大量的对象或簇**
  - **一个步骤一旦完成便不能被撤消**
    - 避免考虑选择不同的组合，减少计算代价
    - 不能更正错误的决定
  - **不具有很好的可伸缩性**
- **改进方法：将层次聚类和其他的聚类技术进行集成，形成多阶段聚类**
  - **BIRCH：使用CF-tree 对对象进行层次划分，然后采用其他的聚类算法对聚类结果进行求精**
  - **CURE：采用固定数目的代表对象来表示每个簇，然后依据一个指定的收缩因子向着聚类中心对它们进行收缩**
  - **CHAMELEON：使用动态模型进行层次聚类**

南京大学
NANJING UNIVERSITY

目录

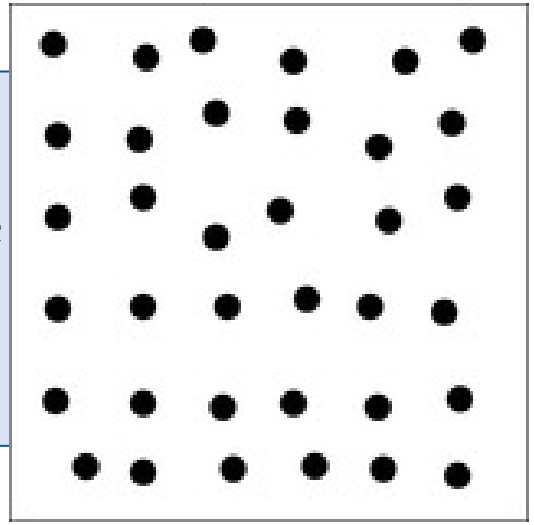
聚类评估

- 估计在数据集上进行聚类的可行性和被聚类方法产生的结果的质量
- 聚类评估的任务
  - 估计聚类趋势：评估数据集是否存在非随机结构
  - 确定数据集中的簇数：在聚类之前，估计簇数
  - 测定聚类质量：聚类之后，评估结果簇的质量

估计聚类趋势

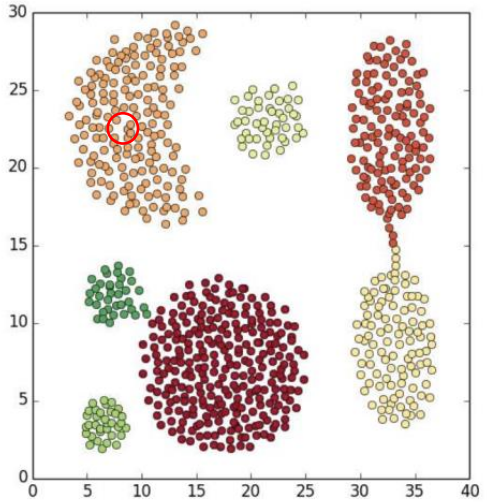
- 从统计学的角度，就是检测数据是否是随机或线性分布的

如果数据服从均匀分布，显然对其进行的聚类操作都是没有意义的



$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$$

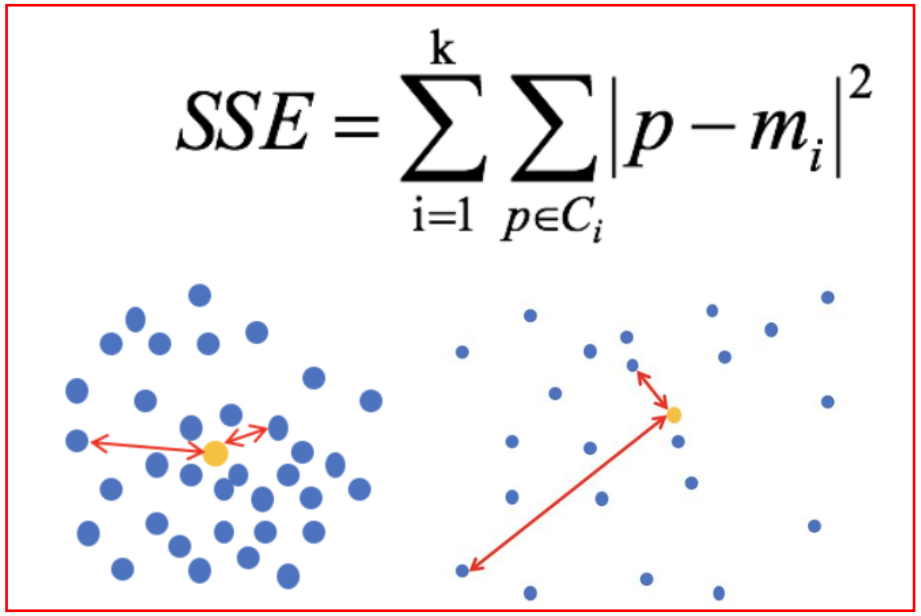
霍普金斯统计量

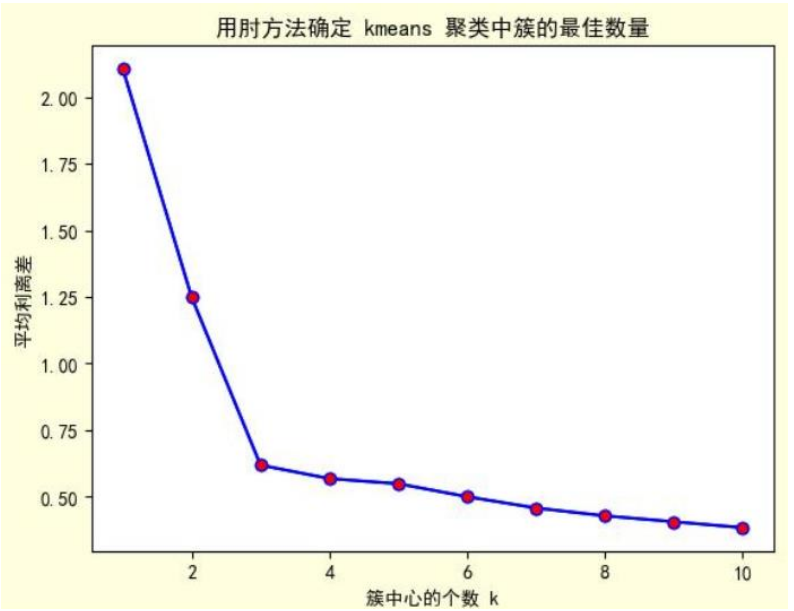


确定数据集中的簇数

- 经验发则
  - 对于n个点的数据集，簇数$\sqrt{n/2}$，每个簇约有$\sqrt{2n}$个点
- 肘部（Elbow）方法
  - 增加簇数可以降低簇内方差之和，但是如果形成太多的簇，降低簇内方差之和的边缘效应可能下降。

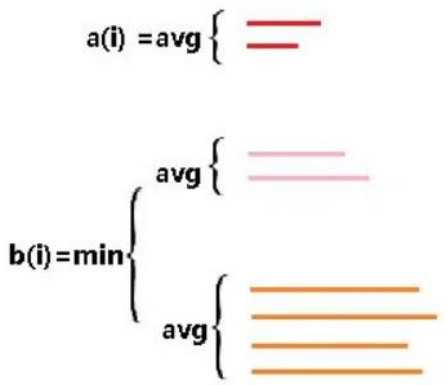
$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2$$

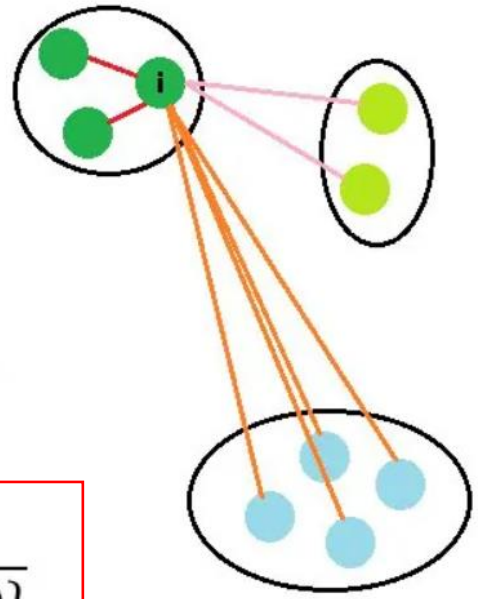




测定聚类质量

- 外在方法：有监督的
  - 用某种聚类质量度量对聚类结果和基准进行比较
  - 基准：一种理想的聚类，由专家构建
- 内在方法：无监督的
  - 通过考察簇的分离情况和簇的紧凑情况来评估聚类
  - 例：轮廓系数
    - ✓ 值介于 [-1,1]
    - ✓ 越趋近于1代表内聚度和分离度都相对较优





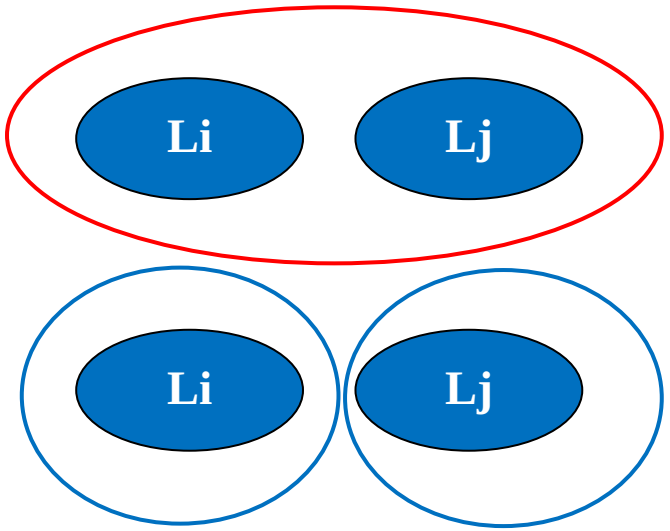
$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

测定聚类质量

- 簇的同质性
- 簇的完全性
- 碎布袋性
- 小簇保持性

测定聚类质量

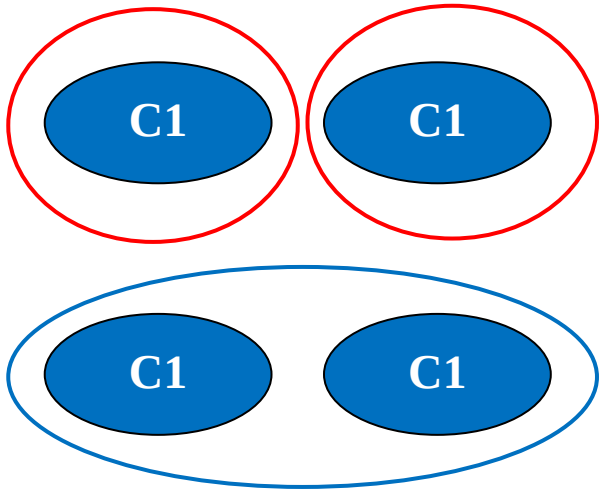
- **簇的同质性**



- **簇的完全性**
- **碎布袋性**
- **小簇保持性**

测定聚类质量

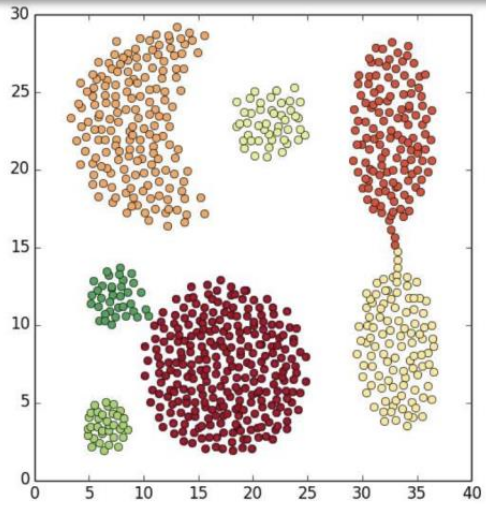
- 簇的同质性
- 簇的完全性



- 碎布袋性
- 小簇保持性

测定聚类质量

- 簇的同质性
- 簇的完全性
- 碎布袋性
  - 把一个异构对象放到一个纯聚簇中要比把它放到一个碎布袋里（例如，“杂项”或“其他”类别）更糟糕
  - 比如更希望噪音不被聚到已存在的聚簇中
- 小簇保持性



测定聚类质量

- 簇的同质性
- 簇的完全性
- 碎布袋性
- 小簇保持性
  - 把一个小类别聚簇分成更小的聚簇比把一个大类别分成小聚簇更有害
  - 也就是我们希望小类别聚簇不再被划分