《Python与数据挖掘》

第8章 聚类分析

讲师：武永亮

目录

聚类分析概述

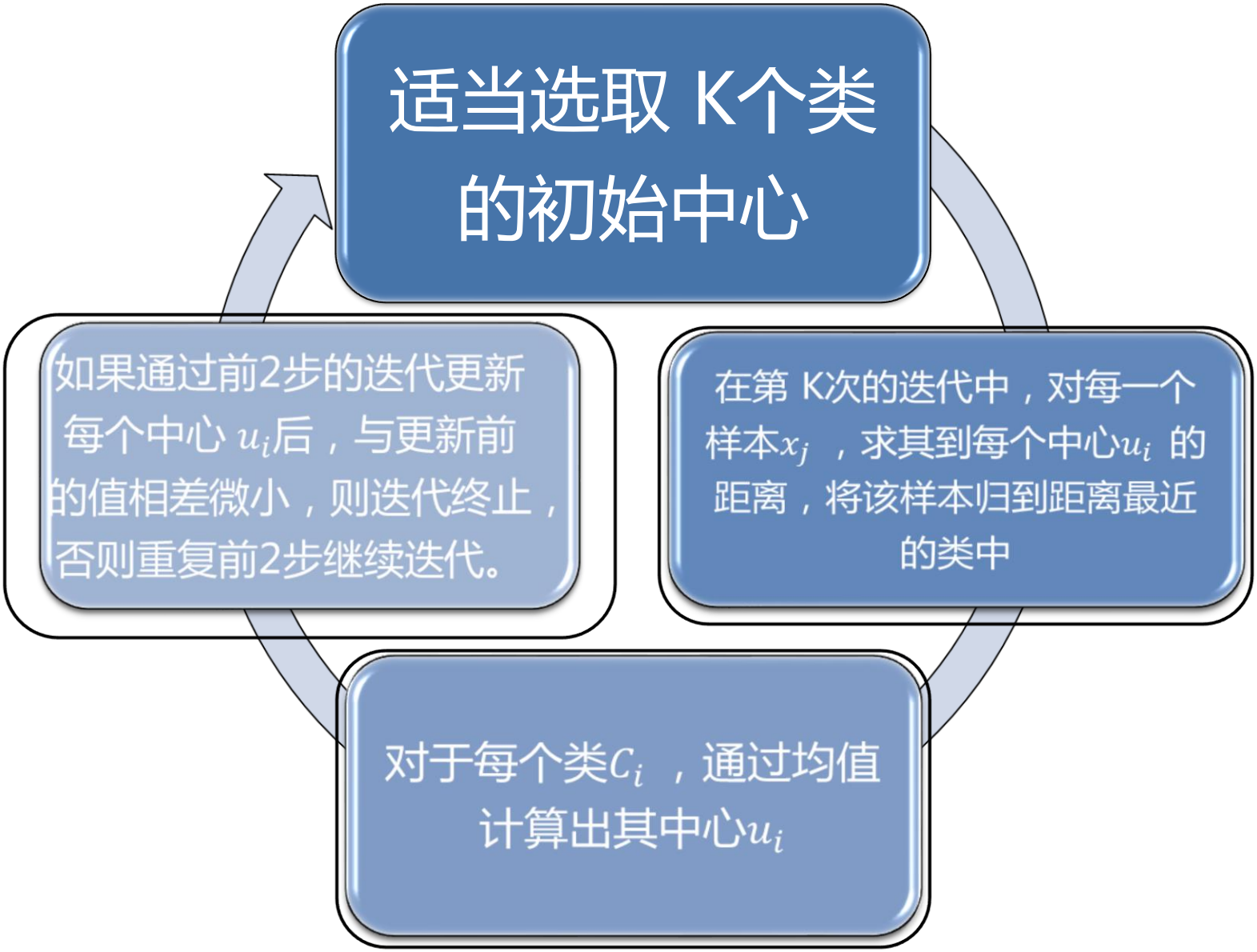
- 聚类分析是研究对事物进行分类的一种多元统计方法。
- 由于对象的复杂性，仅凭经验和专业知识有时不能达到确切分类的目的，于是数学方法就被引进到分类问题中来。
- 聚类分析根据事物彼此不同的属性进行辨认，将具有相似属性的事物聚为一类，使得同一类的事物具有高度的相似性。这使得聚类分析可以很好的解决无法确定事物属性的分类问题。

K-Means聚类分析

- 聚类分析中最广泛使用的算法为K-Means聚类分析算法。K-Means算法属于聚类分析中划分方法里较为经典的一种，由于该算法的效率高，所以在对大规模数据进行聚类时被广泛应用。
- K-Means算法通过将样本划分 k个方差齐次的类来实现数据聚类。该算法需要指定划分的类的个数。它处理大数据的效果比较好，已经被广泛用于实际应用。
- K-Means算法将数据集 N中的 n个样本划分成 k个不相交的类，将这 k个类用字母C表示， n个样本用字母 表示X，每一个类都具有相应的中心u_i 。K-Means算法是一个迭代优化算法，最终使得下面的均方误差最小:

$$\min \sum_{i=0}^{k} \sum_{x_j \in c_i} \left(\left\| x_j - u_i \right\|^2 \right)$$

迭代算法步骤


适当选取 K个类的初始中心
在第 K次的迭代中，对每一个样本x_j ，求其到每个中心u_i 的距离，将该样本归到距离最近的类中
对于每个类C_i ，通过均值计算出其中心u_i
如果通过前2步的迭代更新每个中心 u_i后，与更新前的值相差微小，则迭代终止，否则重复前2步继续迭代。

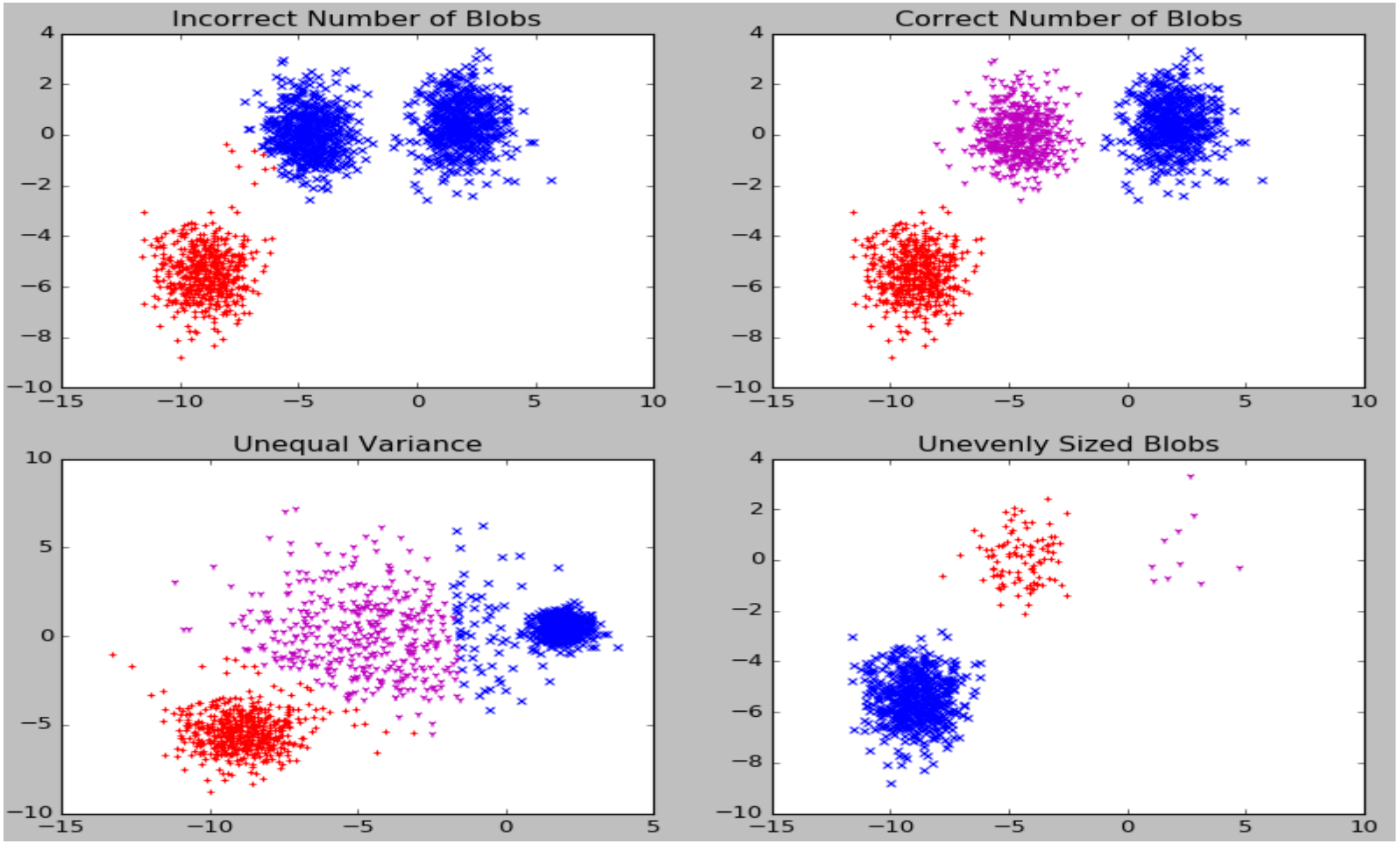
Python实现

- 这里选择scikit-learn中的K-Means算法进行聚类实验。看看程序代码和K-Means算法的效果：

```
# 聚类数量不正确时的效果
y_pred = KMeans(n_clusters=2, random_state=random_state).fit_predict(X)
plt.subplot(221)
plt.scatter(X[y_pred==0][:, 0], X[y_pred==0][:, 1], marker='x',color='b')
plt.scatter(X[y_pred==1][:, 0], X[y_pred==1][:, 1], marker='+',color='r')
plt.title("Incorrect Number of Blobs")
# 类间的方差存在差异的效果
X_varied, y_varied = make_blobs(n_samples=n_samples,
                                cluster_std=[1.0, 2.5, 0.5],
                                random_state=random_state)
y_pred = KMeans(n_clusters=3,
random_state=random_state).fit_predict(X_varied)
plt.subplot(223)
plt.scatter(X_varied[y_pred==0][:, 0], X_varied[y_pred==0][:, 1],
marker='x',color='b')
plt.scatter(X_varied[y_pred==1][:, 0], X_varied[y_pred==1][:, 1],
marker='+',color='r')
plt.scatter(X_varied[y_pred==2][:, 0], X_varied[y_pred==2][:, 1],
marker='1',color='m')
plt.title("Unequal Variance")
```

Python实现结果

- 我们实验采取的数据集是scikit-learn中的make_blods。使用scikit-learn中K-Means算法的程序语句很简单，实现结果如下图：



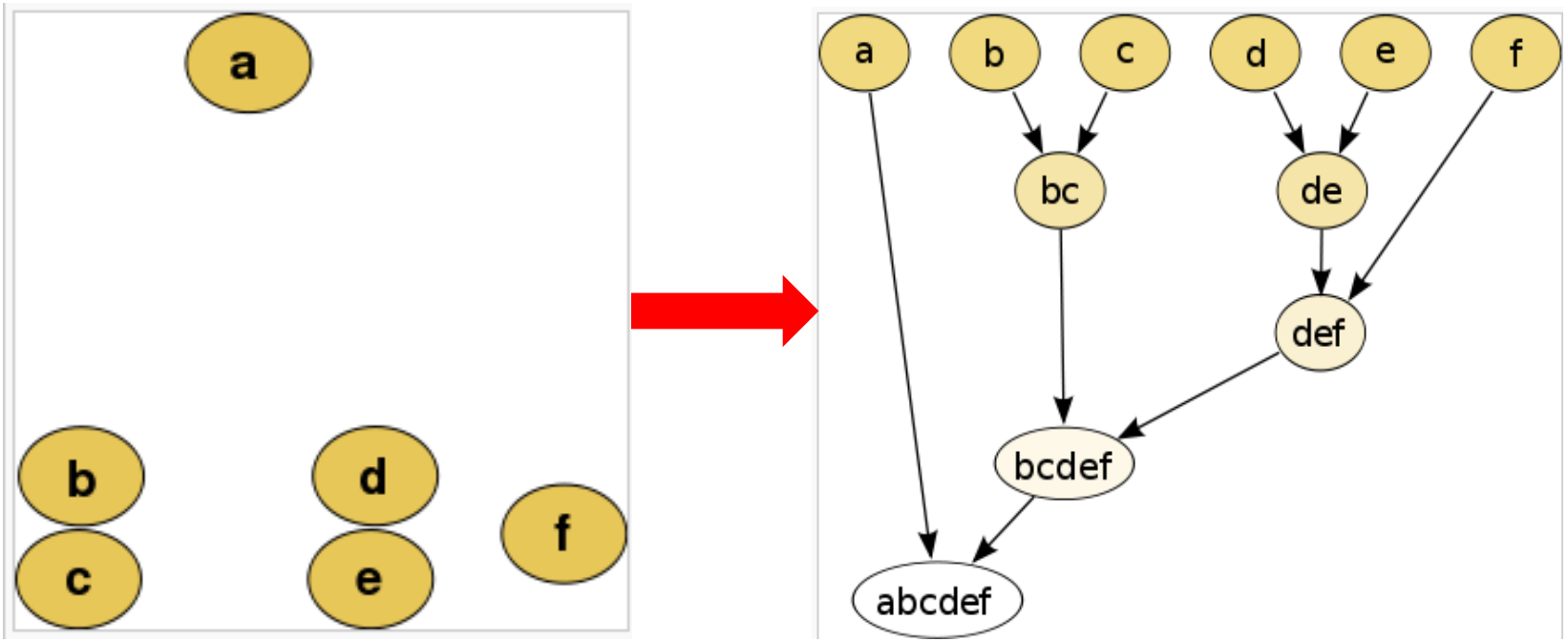
目录

系统聚类概述

- 系统聚类（又称为层次聚类，系谱聚类）是一个一般的聚类算法，通过合并或分割类，生成嵌套的集群。算法的层次结构可以以一棵树表示。树的根是一个唯一的类，包含了所有的样本，而树的叶子节点是单独的一个样本。通过树的叶子节点的相互合并，最终合并成为树的根节点。

系统聚类原理

- 系统聚类的的基本思想是先将样本看作各自一类，定义类间距离的计算方法，选择距离最小的一对类合并成为一个新的类。接着重新计算类间的距离，再将距离最近的两类合并，如此最终便最终合成一类。



系统聚类

- 由上图给出了一个系统聚类的例子。
- 首先定义样本间距离的计算方法，计算各个样本点间的距离。先将距离最近的b与c合并，此时有5个类： {a},{b,c} ,{d} ,{e} 和{f} 。
- 为了进一步的合并，所以需要计算类 {a}与}{b,c} 间的距离。
- 因此还需要定义类间距离的计算方法。按照合并距离最小的两个类的规则，我们按顺序合并 {d}与 {e} ，{d,e} 与 {f}， {b,c}与{d,e,f} ，{a} 与 {b,c,d,e,f} 。
- 最终我们通过类的合并得出上图的结果。整个过程如同生成树的过程，树的层次结构分明。

系统聚类

- 下表给出了样本间距离的常用定义：

距离名称	公式
欧几里得距离(Euclidean distance)	$d(a,b)=\|a-b\|_2=\sqrt{\sum_i (a_i-b_i)^2}$
均方距离(Square Euclidean distance)	$d(a,b)=\|a-b\|_2^2=\sum_i (a_i-b_i)^2$
曼哈顿距离(Manhattan distance)	$d(a,b)=\|a-b\|_1=\sum_i \|a_i-b_i\|$
余弦距离(Cosine distance)	$d(a,b)=\cos\Theta=\frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\bullet\sqrt{\sum_i b_i^2}}$
最大距离(Maximum distance)	$d(a,b)=\|a-b\|_\infty=\max_i \|a_i-b_i\|$

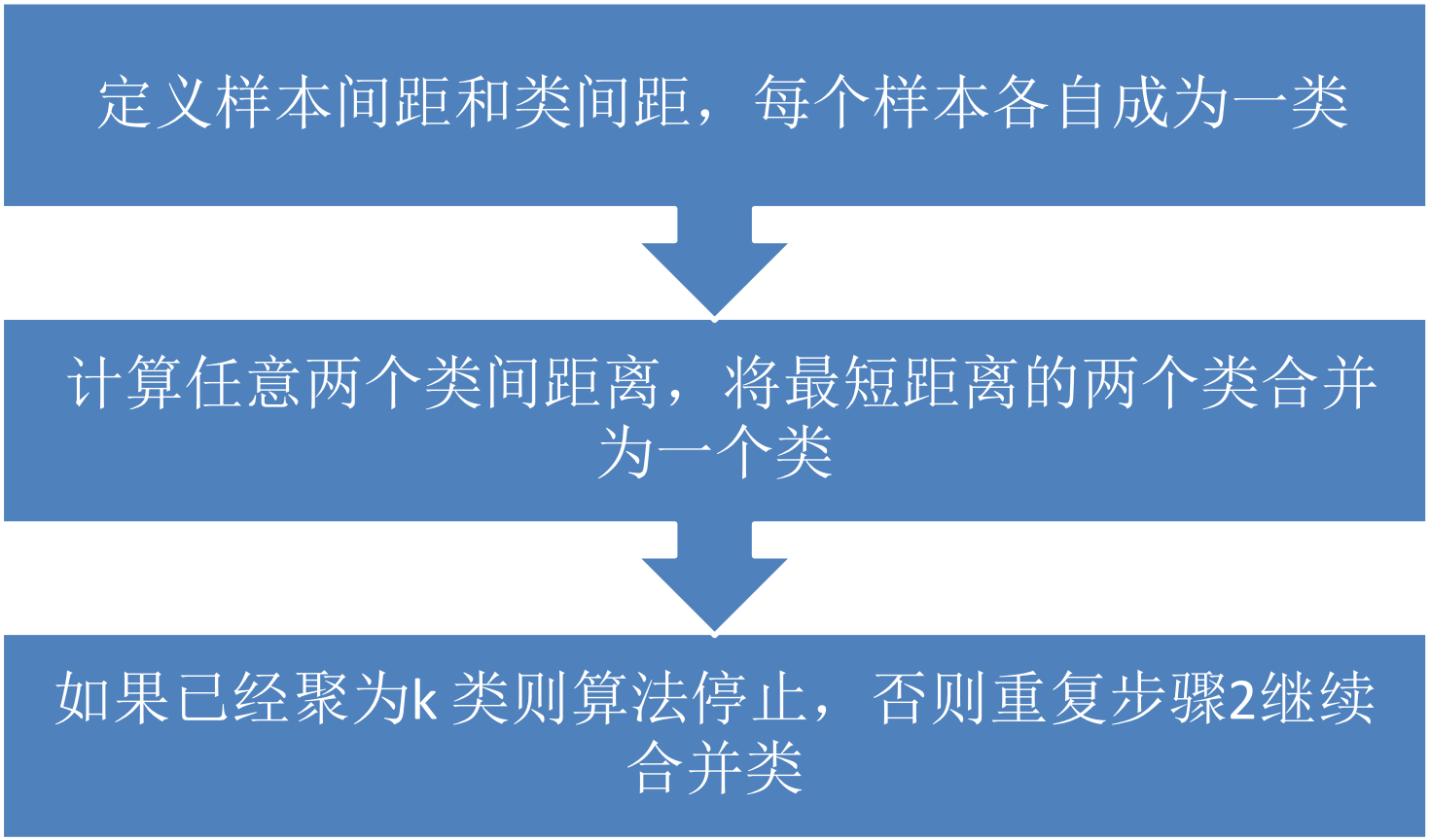
系统聚类

- 下表给出了类间距离的常用定义：

连接规则名称	公式
完全连接聚类 (Complete-linkage clustering)	$d(A,B)=\max(dist(a,b):a\in A,b\in B)$
单一连接聚类 (Single-linkage clustering)	$d(A,B)=\min(dist(a,b):a\in A,b\in B)$
平均连接聚类 (Average linkage clustering)	$d(A,B)=\frac{1}{\|A\|\|B\|}\sum_{a\in A}\sum_{b\in B}d(a,b)$
离差平方和法 (Ward's criterion)	递归算法： 1 初始情形，每个样本点单独作为一个类： $d_{ij}=d\{\{x_i\},\{x_j\}\}=\|x_i-x_j\|^2$ 1 递归合并： $d(A\cup B,C)=\frac{n_A+n_C}{n_A+n_B+n_C}d(A,C)+$ $\frac{n_B+n_C}{n_A+n_B+n_C}d(B,C)-\frac{n_C}{n_A+n_B+n_C}d(A,B)$

系统聚类步骤

- 考虑使用系统聚类算法将数据集 N中的n 个样本划分成 k个不相交的类
- 系统聚类算法步骤如下：



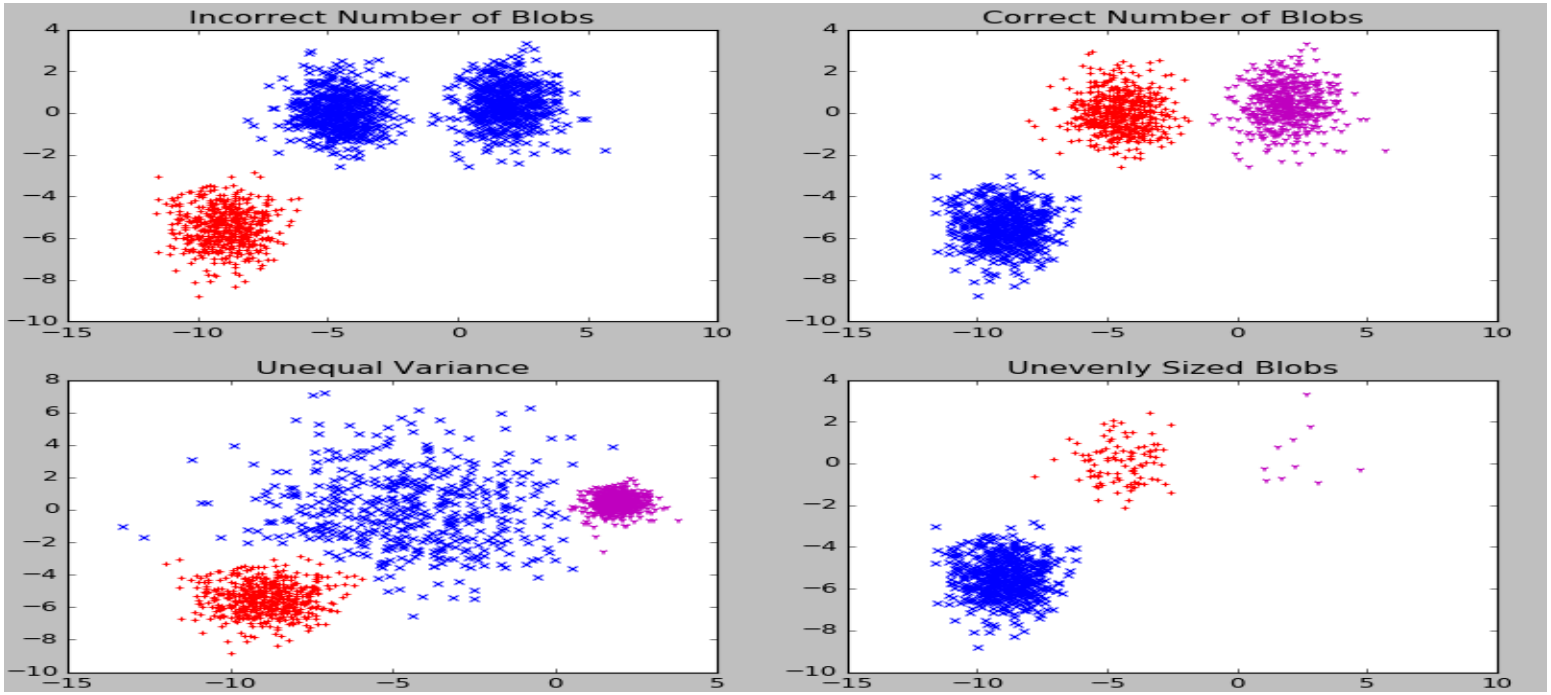
系统聚类

- 通过相同的数据，我们使用系统聚类与K-Means算法效果作对比。一般而言，系统聚类使用欧几里德距离（affinity= ‘euclidean’ ）和离差平方和法(linkage= ‘ward’)效果最好。代码如下：

```
# 聚类数量不正确时的效果
y_pred =
AgglomerativeClustering(affinity='euclidean',linkage='ward',n_clusters=2).fit_predict(X)
# 选取欧几里德距离和离差平均和法
plt.subplot(221)
plt.scatter(X[y_pred==0][:, 0], X[y_pred==0][:, 1], marker='x',color='b')
plt.scatter(X[y_pred==1][:, 0], X[y_pred==1][:, 1], marker='+',color='r')
plt.title("Incorrect Number of Blobs")
# 类间的方差存在差异的效果
X_varied, y_varied = make_blobs(n_samples=n_samples,
                        cluster_std=[1.0, 2.5, 0.5],
                        random_state=random_state)
y_pred= AgglomerativeClustering(
affinity='euclidean',linkage='ward',n_clusters=3).fit_predict(X_varied)
plt.subplot(223)
plt.scatter(X_varied[y_pred==0][:, 0], X_varied[y_pred==0][:, 1], marker='x',color='b')
plt.scatter(X_varied[y_pred==1][:, 0], X_varied[y_pred==1][:, 1], marker='+',color='r')
plt.scatter(X_varied[y_pred==2][:, 0], X_varied[y_pred==2][:, 1], marker='1',color='m')
plt.title("Unequal Variance")
```

系统聚类

- 从实验结果分析，系统聚类的结果比K-Means的聚类效果要好，在这两个实验尤为明显。
- 从算法上分析,K-Means需要随机选择类的初始中心，给算法带来一定的不稳定性，比起K-Means的迭代算法，系统聚类算法更为严谨，每一步合并都是贪心的。



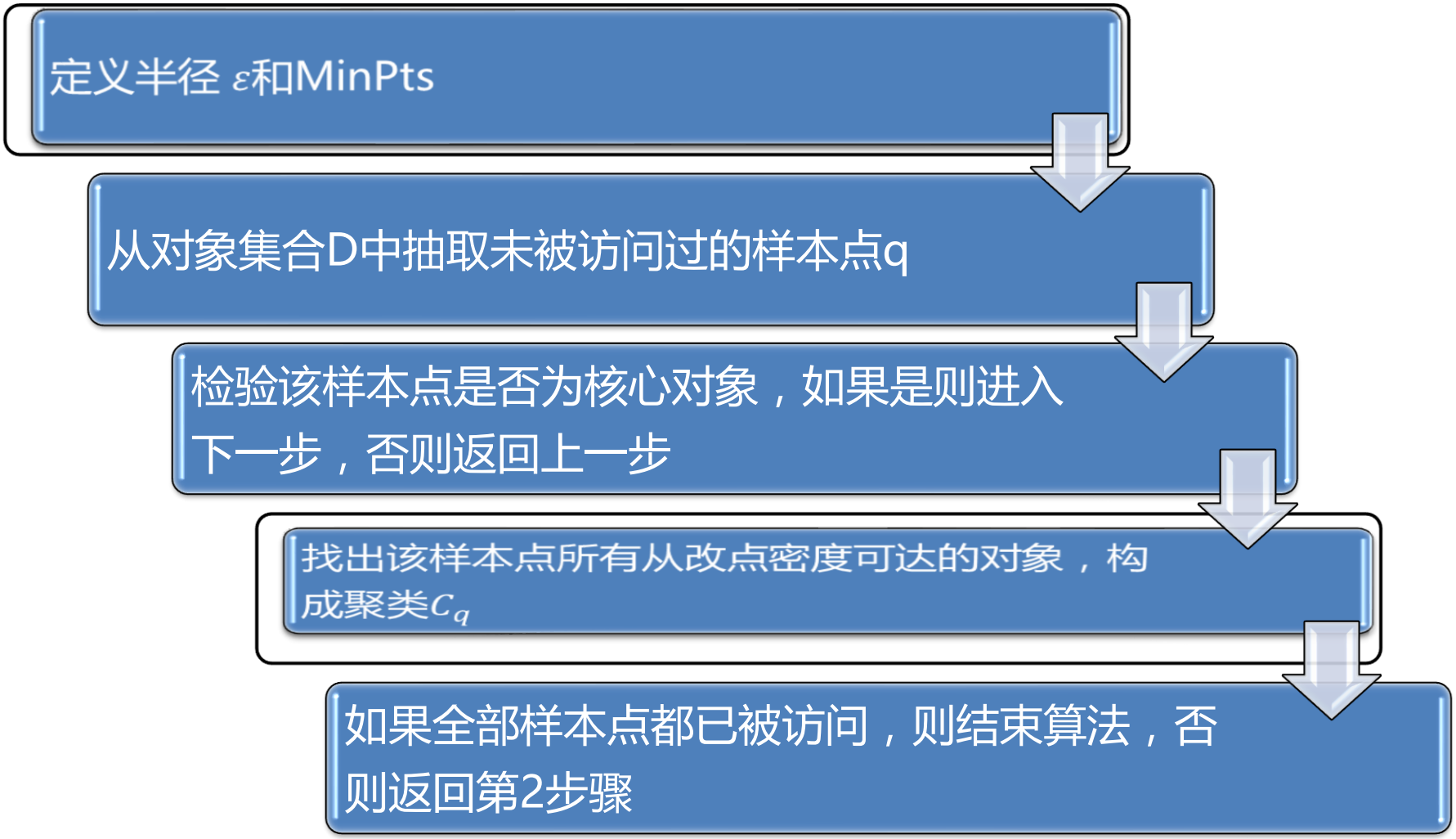
目录

DBSCAN聚类

- DBSCAN(Density-Based Spatial Clustering of Applications with Noise)是一个有代表性的密度聚类算法。它将类定义为密度相连的点的最大集合，通过在样本空间中不断寻找最大集合从而完成聚类。该算法在带噪声的样本空间中发现任意形状的聚类并排除噪声。
- 首先我们将列出DBSCAN算法涉及的基本定义：

ε 邻域	给定对象半径 ε 内的区域称为该样本点的 ε 邻域
核心对象	如果给定对象 ε 邻域内的样本点数大于设定的 MinPts，则称该对象为核心对象
直接密度可达	给定对象集合 D，如果对象 p 在对象 q 的 ε 邻域内，且 p 是 D 的一个核心对象，则称为对象 p 从对象 q 出发是直接密度可达的
密度可达	给定对象集合 D，如果存在一个对象链 $p_1, p_2, \cdots, p_n$，$p_1 = q$，$p_n = p$，$\forall p_i \in D(1 \le i \le n-1)$ 都有 p_{i+1} 与 p_i 是直接密度可达的，则称对象 p 从对象 q 出发是密度可达的
密度相连	如果存在对象 $o \in D$ 使得对象 p 和对象 q 都是从 o 出发密度可达的，则称对象 p 从对象 q 出发是密度相连

DBSCAN聚类步骤

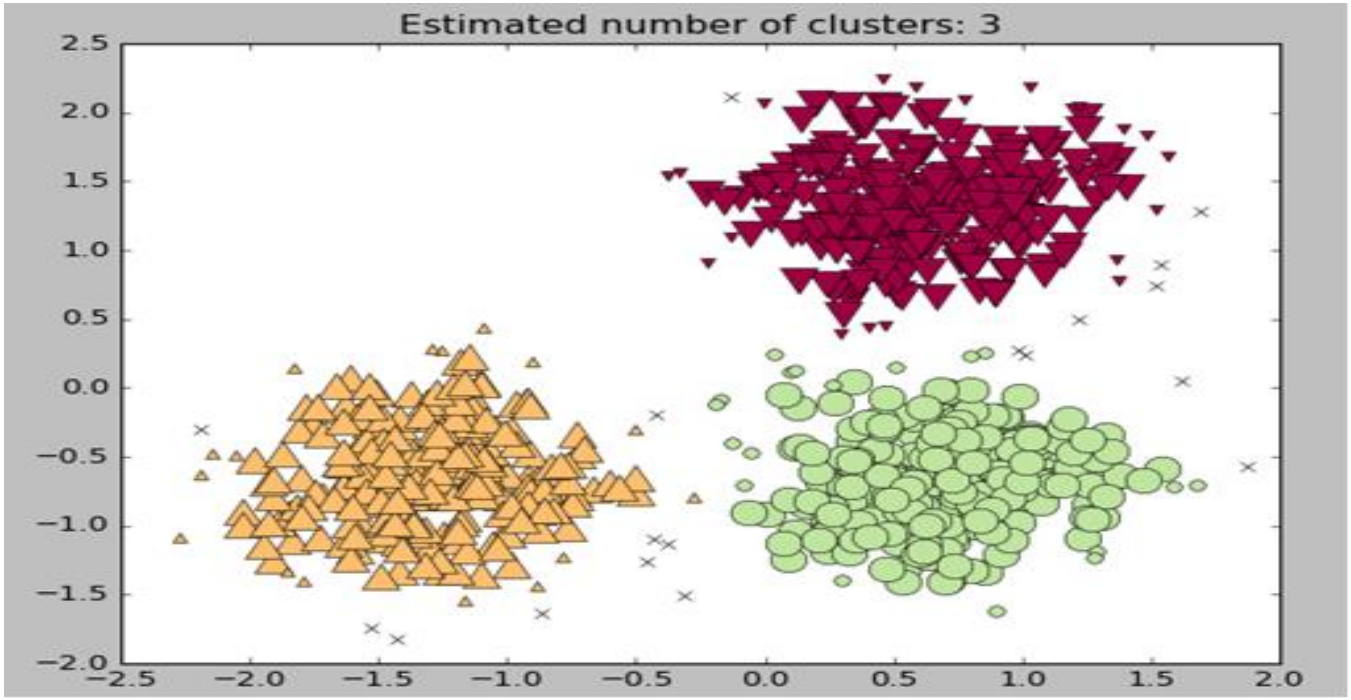

定义半径 ε和MinPts
从对象集合D中抽取未被访问过的样本点q
检验该样本点是否为核心对象，如果是则进入下一步，否则返回上一步
找出该样本点所有从改点密度可达的对象，构成聚类C_q
如果全部样本点都已被访问，则结束算法，否则返回第2步骤

DBSCAN聚类

- 我们还是利用scikit-learn中已经封装好的DBSCAN算法，设置DBSCAN分类器格式如下：
- clf = sklearn.cluster.DBSCAN(eps=0.3, min_samples=10)
- 如算法原理所述，我们需要设定两个参数，ε和MinPts。这两个参数需要根据我们的经验，或根据多次实验进行调整。

DBSCAN聚类

- 分析下图 ，DBSCAN算法能够很好地去除噪音，聚类效果也比较理想。图中较大的样本点是核心对象，较小的样本点是非核心对象，以非核心对象为界的思想能够较好地划分类。
- 借助scikit-learn模块我们能够轻松使用各自聚类算法，但我们必须了解算法背后的原理，知道如何调节算法的参数，这样才会取得好的聚类效果。



Thank You!