第8章 聚类分析

麦国炫

2017/10/24

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。

聚类分析的概述

- 相近样本聚为一类，远离的聚于不同类。

应用领域：

- 在客户分类
- 文本分类
- 基因识别
- 空间数据处理
- 卫星图片分析
- 数据分析、统计学、机器学习、空间数据库技术、生物学和市场学也推动了聚类分析研究的进展

变量

聚类分析是研究对样本或变量的聚类，在进行聚类时，可使用的方法有很多，而这些方法的选择往往与变量的类型是有关系的，由于数据的来源及测量方法的不同，变量大致可以分为两类：

- 1）定量变量，也就是通常所说的连续变量。
- 2）定性变量，这些量并非真有数量上的变化，而只有性质上的差异。这些量可以分为两种，一种是有序变量，另一种是名义变量。

连续性变量距离

对于连续性变量数据，有一些典型的距离定义

距离	定义式	说明
绝对值距离	$d_{ij}(1)=\sum_{k=1}^{p}\lvert x_{ik}-x_{jk}\rvert$	绝对值距离是在一维空间下进行的距离计算
欧式距离	$d_{ij}(2)=\sqrt{\sum_{k=1}^{p}(x_{ik}-x_{jk})^2}$.	欧式距离是在二维空间下进行的距离计算
闵可夫斯基距离	$d_{ij}(q)=\left[\sum_{k=1}^{p}(x_{ik}-x_{jk})^q\right]^{1/q}$，$q>0$.	闵可夫斯基距离是在 q 维空间下进行的距离计算
切比雪夫距离	$d_{ij}(\infty)=\max_{1\leqslant k\leqslant p}\lvert x_{ik}-x_{jk}\rvert$.	切比雪夫距离是 q 取正无穷大时的闵可夫斯基距离，即切比雪夫距离是在 $+\infty$ 维空间下进行的距离计算
Lance 距离	$d_{ij}(L)=\sum_{k=1}^{p}\frac{\lvert x_{ik}-x_{jk}\rvert}{x_{ik}+x_{jk}}$	减弱极端值的影响能力
归一化距离	$d_{ij}=\sum_{k=1}^{p}\frac{\lvert x_{ik}-x_{jk}\rvert}{\max(x_k)-\min(x_k)}$	自动消除不同变量间的纲量影响，其中每个变量 k 的距离取值均是 [0,1]

聚类算法

聚类算法种类繁多，且其中绝大多数可以用python实现。下面将选取普及性最广、最实用、最具有代表性的5中聚类算法进行介绍，其中包括：

- K-均值聚类(K-Means)
- K-中心点聚类(K-Medoids)
- 密度聚类(Densit-based Spatial Clustering of Application with Noise, DBSCAN)
- Hierarchical Clustering(系谱聚类,层次聚类)
- 期望最大化聚类(Expectation Maximization, EM)

需要说明的是，这些算法本身无所谓优劣，而最终运用于数据的效果却存在好坏差异，这在很大程度上取决于数据使用者对于算法的选择是否得当。

目录

K-Means聚类分析

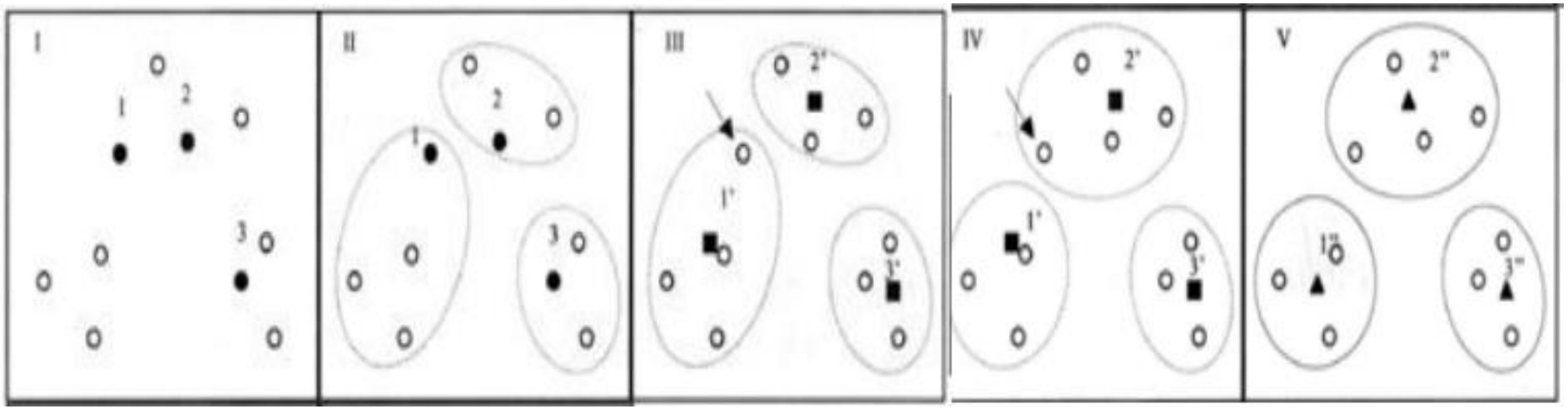
- 聚类分析中最广泛使用的算法为k-means聚类分析算法。K-means算法属于聚类分析中划分方法里较为经典的一种，由于该算法的效率高，所以在对大规模数据进行聚类的被广泛应用。
- K-means算法通过将样本划分k个方差齐次的类来实现数据聚类。该算法需要指定划分的类的个数。它处理的大数据的效果比较好，已经被广泛用于实际应用。
- K-means算法将数据集N中的n个样本划分为k个不相交的类，将这k个类用字母C来表示，n个样本用字母X表示，每一个类都具有相应的中心u_i。K-means算法是一个迭代优化算法，最终使得下面的均方误差最小：

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

- K-means算法以欧式距离作为相似度测度

迭代算法步骤

- 1、随机选取K个样本作为类中心；
- 2、计算各样本与各类中心的距离；
- 3、将各样本归于最近的类中心点；
- 4、求各类的样本的均值，作为新的类中心；
- 5、判定：若类中心不再发生变动或达到迭代次数，算法结束，否则回到第2步。



K-means算法

选定样本a和b为初始类中心，中心值分别为1、2

a	1
b	2
c	4
d	5

1. 选中心
2. 求距离
3. 归类
4. 求新类中心
5. 判定结束

	1	2	class
a	0	1	1
b	1	0	2
c	3	2	2
d	4	3	2

center1=1

center2=11/3

	1	11/3	class
a	0	8/3	1
b	1	5/3	1
c	3	1/3	2
d	4	4/3	2

center1=3/2

center2=9/2

	3/2	9/2	class
a	1/2	7/2	1
b	1/2	5/2	1
c	5/2	1/2	2
d	7/2	1/2	2

center1=3/2

center2=9/2

结束

K-means算法

- 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，主要语句是:

```
# 设置分类器
clf  =  sklearn.cluster.Kmeans(n_cluster=8,random_state=None,...)
# random_state参数是随机因子，使得初始中心是随机选取的
# 训练分类器并对样本的标签进行预测
y_pred = clf.fit_predict(X)
```

目录


1
K-Means聚类分析
2
K-Medoids聚类分析
3
DBSCAN聚类分析
4
Hierarchical Clustering聚类分析
5
EM原理聚类分析

K-medoids

- k中心算法的基本过程是: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每个代表对象的距离（此处距离不一定是欧氏距离，也可能是曼哈顿距离）分配给最近的代表对象所代表的簇；然后反复用非代表对象来代替代表对象，以优化聚类质量。聚类质量用一个代价函数来表示。当一个中心点被某个非中心点替代时，除了未被替换的中心点外，其余各点被重新分配。
- 为了减轻k均值算法对孤立点的敏感性，k中心点算法不采用簇中对象的平均值作为簇中心，而选用簇中离平均值最近的对象作为簇中心。

步骤

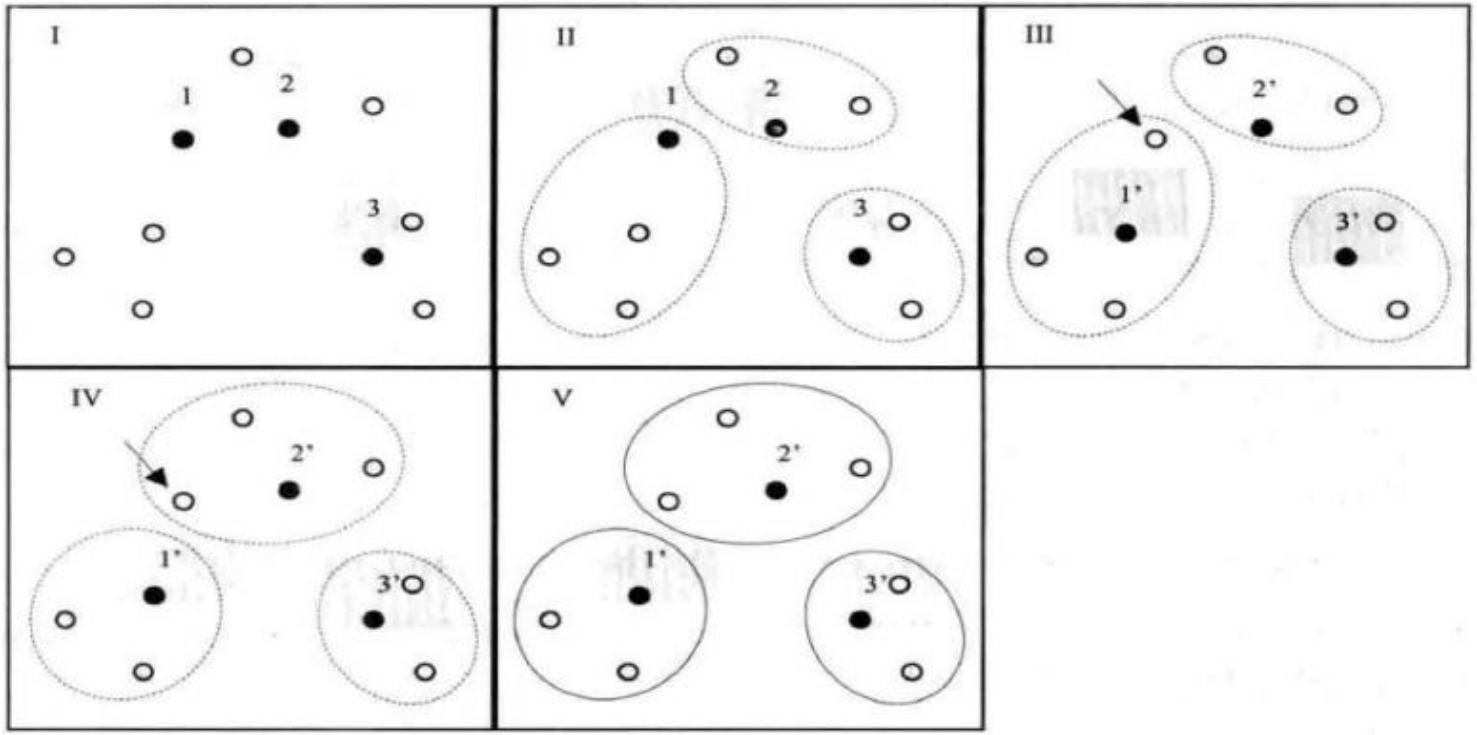
算法如下:

- 输入:包含n个对象的数据库和簇数目k。
- 输出:k个簇
- （1）随机选择k个代表对象作为初始的中心点
- （2）指派每个剩余对象给离它最近的中心点所代表的簇
- （3）随机地选择一个非中心点对象y
- （4）计算用y代替中心点x的总代价s
- （5）如果s为负，则用可用y代替x，形成新的中心点
- （6）重复（2）（3）（4）（5），直到k个中心点不再发生变化.

K-Medoids

K-中心点算法与K-均值算法在原理上十分相近，它是针对K-均值算法易受极值影响这一缺点的改进算法。在原理上的差异在于选择各类别中心点时不取样本均值点，而在类别内选取到其余样本距离之和最小的样本为中心。

- 下图表示出算法的基本运行步骤。



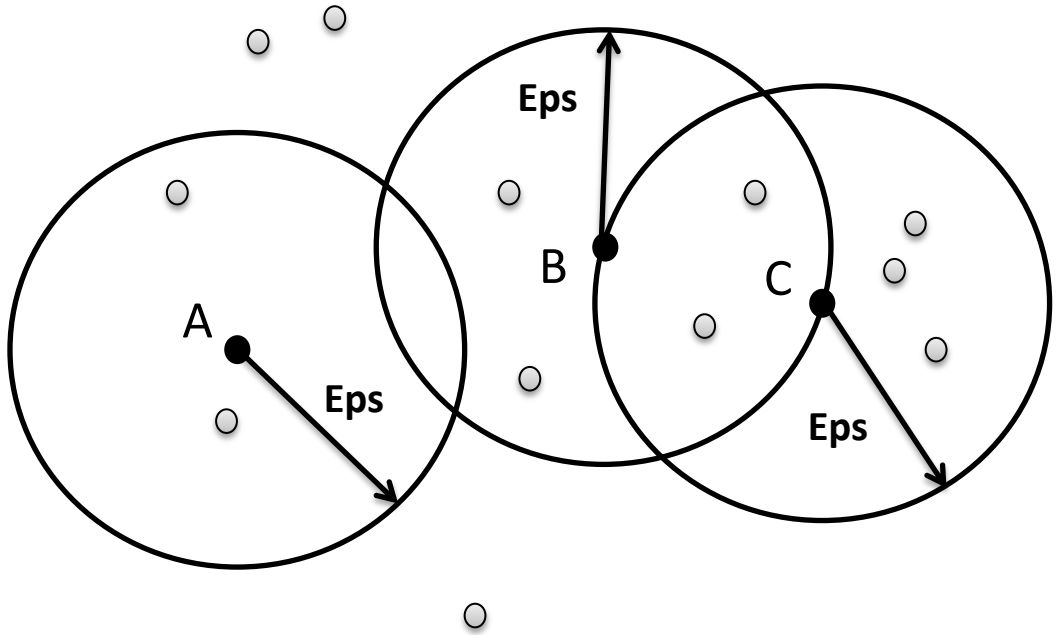
目录


1
K-Means聚类分析
2
K-Medoids聚类分析
3
DBSCAN聚类分析
4
Hierarchical Clustering聚类分析
5
EM原理聚类分析

DBSCAN

- 密度聚类亦称“基于密度的聚类”（density-based clustering），此类算法假设聚类结构能通过样本分布的紧密程度确定。
- DBSCAN，全称为“density-based Spatial clustering of application with Noise”。DBSCAN是一种著名的密度聚类算法，主要是依赖两个主要的参数来进行聚类的，即对象点的区域半径Eps和区域内点的个数的阈值MinPts。
- DBSCAN算法通过查找数据聚类的，即对象点的区域半径Eps和区域内点的个数的阈值M集中任意一个点的距离在Eps区域来进行聚类，如果这个区域内的点数大于MinPts，则将这些点放在同一个簇中，形成新的一类。

DBSCAN的基本直观图



A为噪声点
B为边界点
C为核心点

DBSCAN

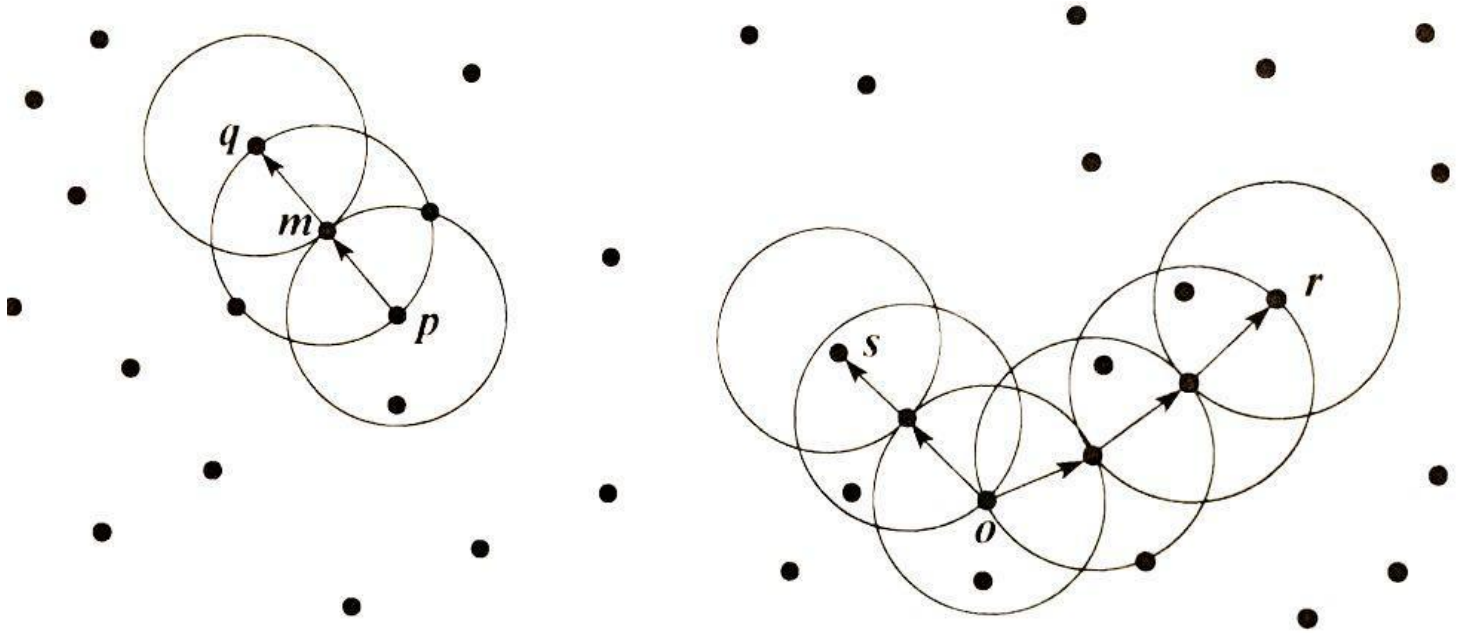
- 核心点：如果点p的密度等于或者大于阈值MinPts，则P为核心点。
- 边界点：如果点p不是核心点，但落在其他核心点的区域内，那么p点为边界点。
- 噪声点：如果点p既不是核心点，也不是边界点，则p点为噪声点。

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算法原理直观图



- 1.对象q是从m直接密度可达的。对象m从p直接密度可达的。
- 2.对象q是从p（间接）密度可达的,因为q到m是密度直达，m到p也是密度直达。
- 3.r和s是从o密度可达的，而o是从r密度可达的，所有o,r和s都是密度相连的。

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该点密度可达的对象，构成聚类C_q

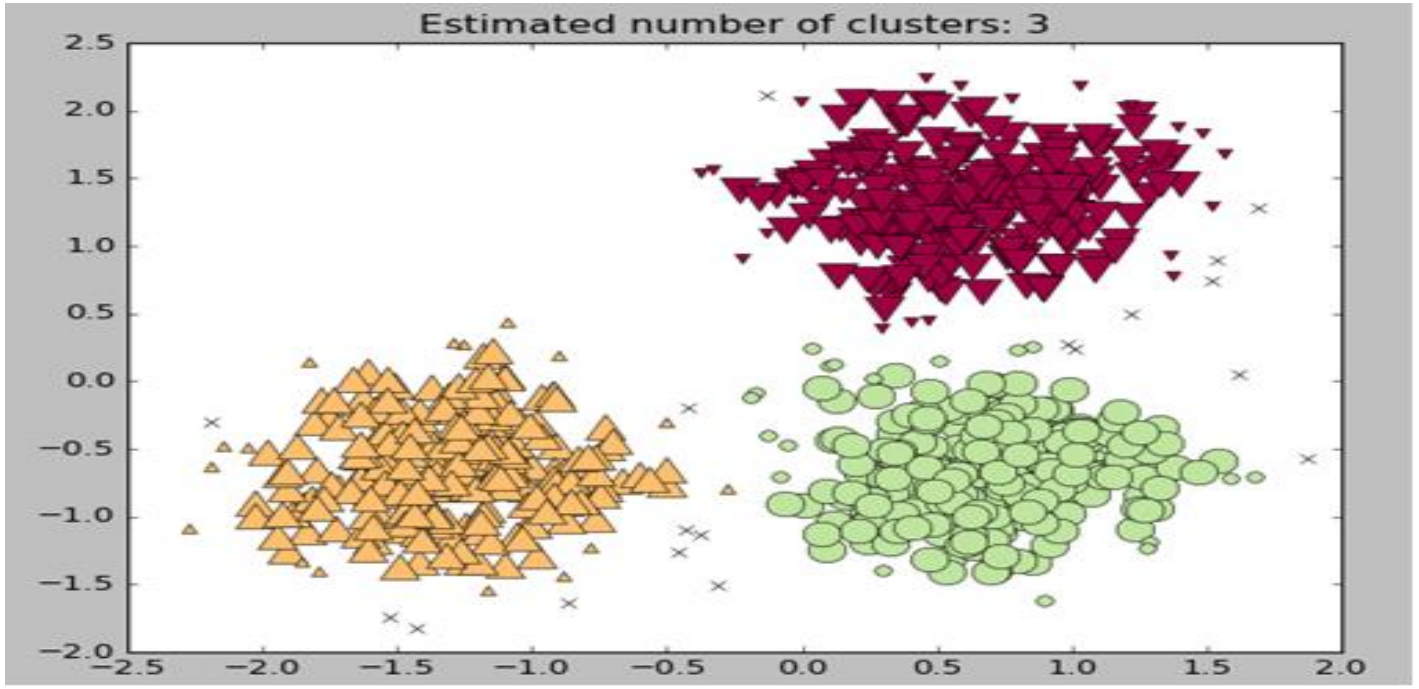
如果全部样本点都已被访问，则结束算法，否则返回第2步骤

DBSCAN聚类

- 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，需要设定两个参数，ε和MinPts。这两个参数需要根据经验，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。



目录

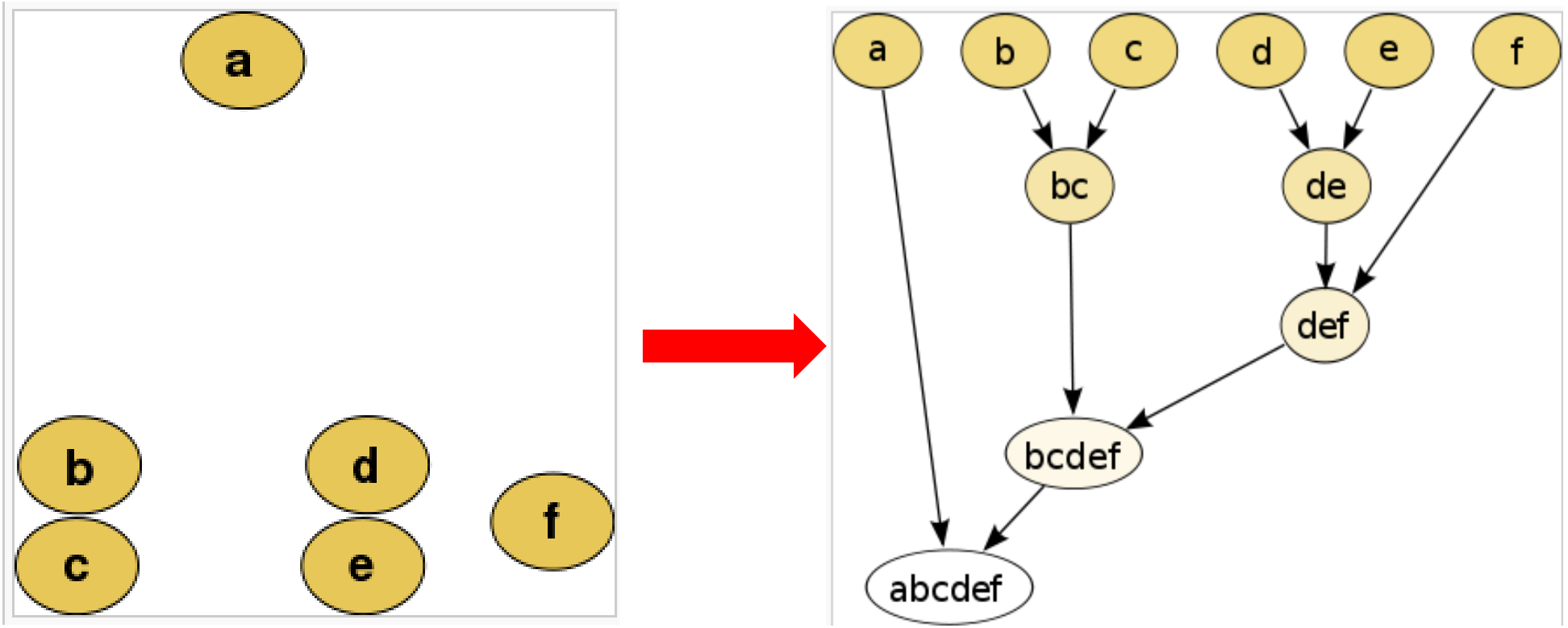

1
K-Means聚类分析
2
K-Medoids聚类分析
3
DBSCAN聚类分析
4
Hierarchical Clustering聚类分析
5
EM原理聚类分析

Hierarchical Clustering聚类概述

- Hierarchical Clustering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。

Hierarchical Clustering聚类原理

Hierarchical Clustering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。



Hierarchical Clustering聚类

由上图给出了一个Hierarchical Clustering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。

Hierarchical Clustering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)	$d(a,b)=\|a-b\|_\infty=\max_i \|a_i-b_i\|$

Hierarchical Clustering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： $d(A\cup B,C)=\frac{n_A+n_C}{n_A+n_B+n_C}d(A,C)+$ $\frac{n_B+n_C}{n_A+n_B+n_C}d(B,C)-\frac{n_C}{n_A+n_B+n_C}d(A,B)$

Hierarchical Clustering聚类步骤

考虑使用Hierarchical Clustering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：

- 1、（初始化）把每个样本归为一类，计算每两个类之间的距离，也就是样本与样本之间的相似度；
- 2、寻找各个类之间最近的两个类，把他们归为一类（这样类的总数就少了一个）；
- 3、重新计算新生成的这个类与各个旧类之间的相似度；
- 4、重复2和3直到所有样本点都归为一类，结束。

- 使用scikit-learn的系统聚类函数能够轻松实现，与K-Means算法实现比较，只需更改一行代码，修改分类器：

K-Means：

y_pred = sklearn.cluster. KMeans(n_clusters=2,random_state=random_state).fit_predict(X)

系统聚类：

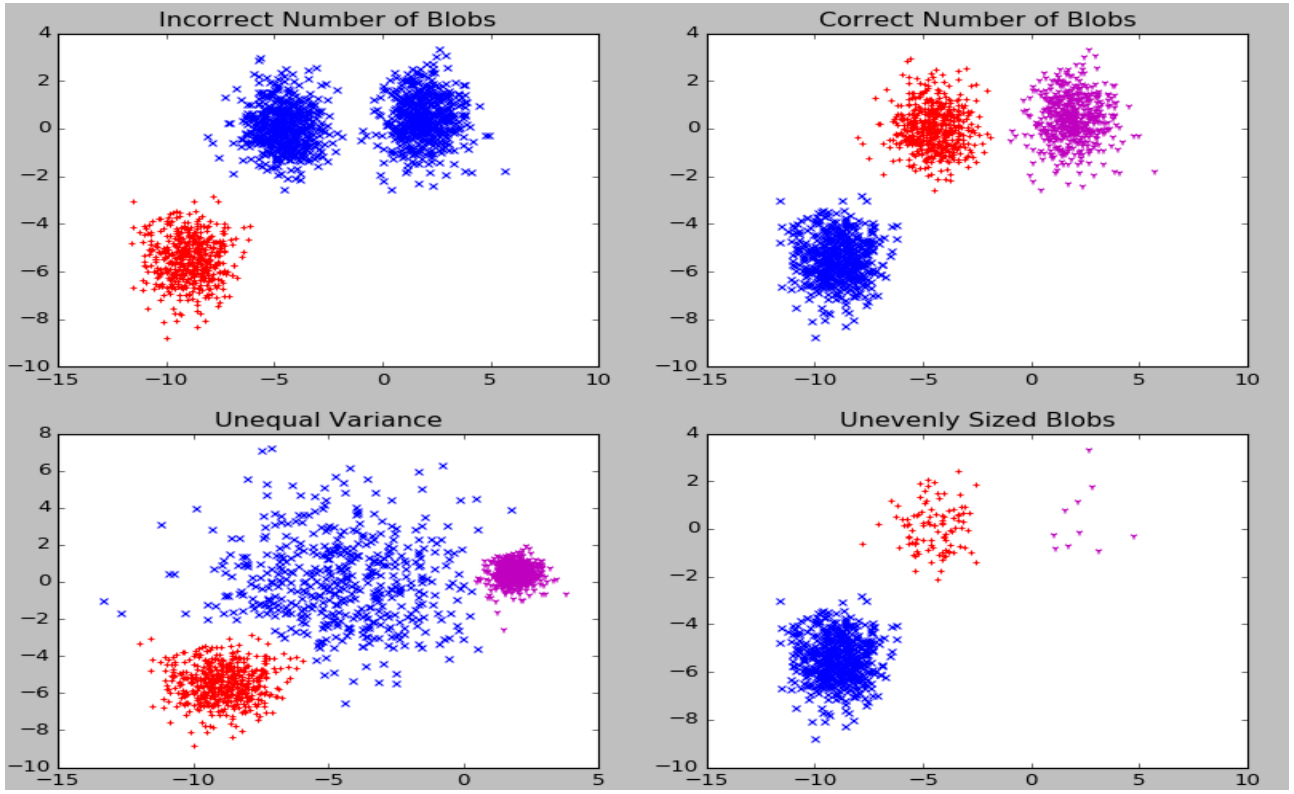
y_pred=sklearn.cluster.AgglomerativeClustering(affinity='euclidean',linkage='ward',n_clusters=2).fit_predict(X)

系统聚类的函数是AgglomerativeClustering()，最重要的参数是这三个：n_clusters聚类数目，affinity样本距离的定义，linkage类间距离的定义（联接规则）。

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


1
K-Means聚类分析
2
K-Medoids聚类分析
3
DBSCAN聚类分析
4
Hierarchical Clustering聚类分析
5
EM原理聚类分析

EM算法

- E M算法指的是最大期望算法（Expectation Maximization Algorithm，又译期望最大化算法），是一种迭代算法，用于含有隐变量（latent variable）的概率参数模型的最大似然估计或极大后验概率估计。
- 在统计计算中，最大期望（EM）算法是在概率（probabilistic）模型中寻找参数最大似然估计或者最大后验估计的算法，其中概率模型依赖于无法观测的隐藏变量（Latent Variable）。最大期望经常用在机器学习和计算机视觉的数据聚类（Data Clustering）领域。

步骤与流程

最大期望算法经过两个步骤交替进行计算：

- 第一步是计算期望（E），利用概率模型参数的现有估计值，计算隐藏变量的期望；
- 第二步是最大化（M），利用E 步上求得的隐藏变量的期望，对参数模型进行最大似然估计。
- M 步上找到的参数估计值被用于下一个 E 步计算中，这个过程不断交替进行。

总体来说，EM的算法流程如下：

- 1.初始化分布参数
- 2.重复直到收敛：
- E步骤：估计未知参数的期望值，给出当前的参数估计。
- M步骤：重新估计分布参数，以使得数据的似然性最大，给出未知变量的期望估计。

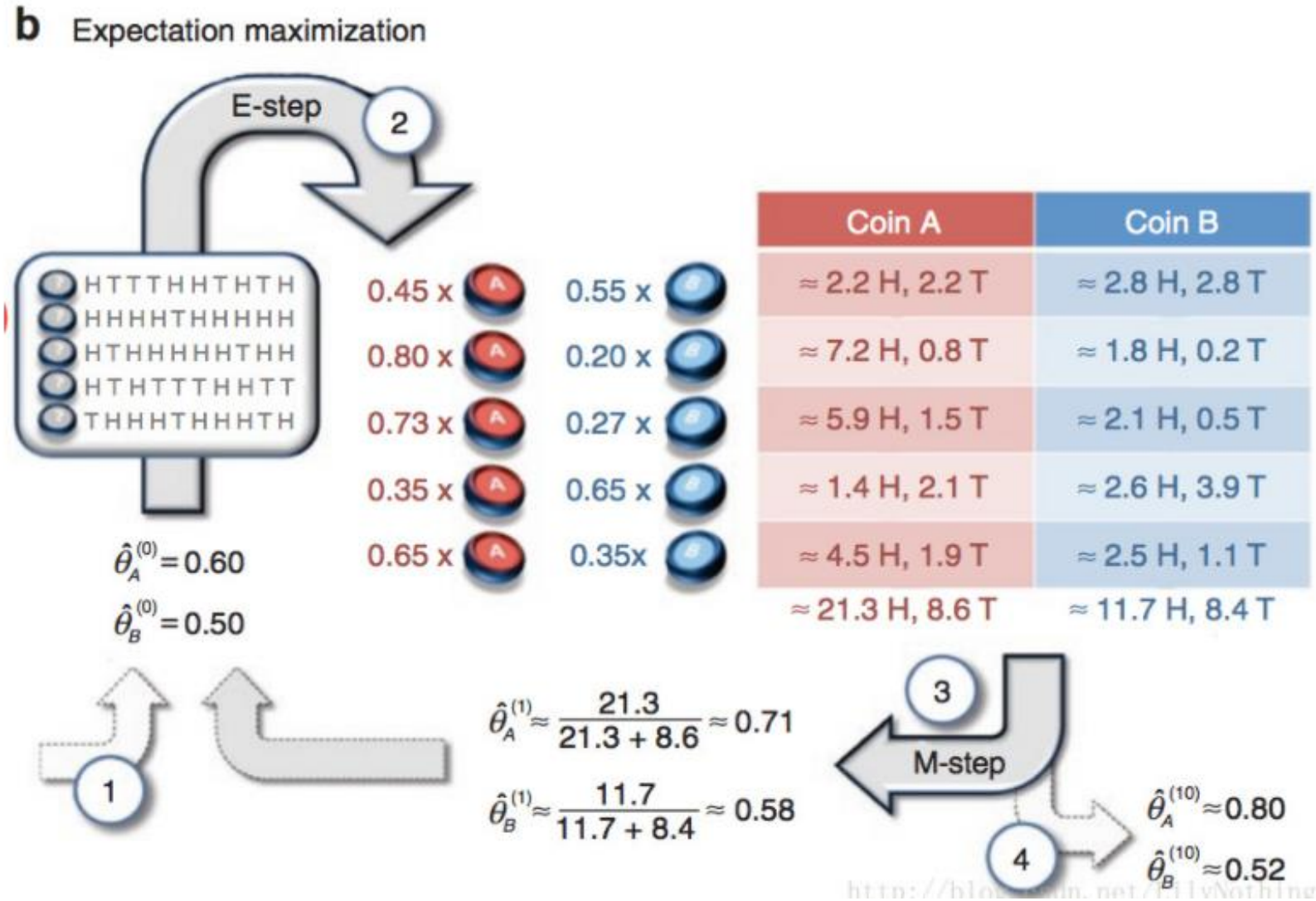
双硬币问题

- 假设有两枚硬币A、B，以相同的概率随机选择一个硬币，进行如下的抛硬币实验：共做5次实验，每次实验独立的抛十次，结果如图中a所示，例如某次实验产生了H、T、T、T、H、H、T、H、T、H，H代表正面朝上。
- 假设试验数据记录员可能是实习生，业务不一定熟悉，造成a和b两种情况
- a表示实习生记录了详细的试验数据，我们可以观测到试验数据中每次选择的是A还是B
- b表示实习生忘了记录每次试验选择的是A还是B，我们无法观测实验数据中选择的硬币是哪个
- 问在两种情况下分别如何估计两个硬币正面出现的概率？

双硬币问题

对于已知是A硬币还是B硬币抛出的结果的时候，可以直接采用概率的求法来进行求解。对于含有隐变量的情况，也就是不知道到底是A硬币抛出的结果还是B硬币抛出的结果的时候，就需要采用EM算法进行求解了

双硬币问题


b Expectation maximization
E-step
2
HTTTHHTHTH
HHHHTHHHHH
HTHHHHHTHH
HTHTTTHHTT
THHHTHHHTH
0.45 x
0.55 x
0.80 x
0.20 x
0.73 x
0.27 x
0.35 x
0.65 x
0.65 x
0.35x
Coin A
Coin B
≈ 2.2 H, 2.2 T
≈ 2.8 H, 2.8 T
≈ 7.2 H, 0.8 T
≈ 1.8 H, 0.2 T
≈ 5.9 H, 1.5 T
≈ 2.1 H, 0.5 T
≈ 1.4 H, 2.1 T
≈ 2.6 H, 3.9 T
≈ 4.5 H, 1.9 T
≈ 2.5 H, 1.1 T
≈ 21.3 H, 8.6 T
≈ 11.7 H, 8.4 T
$\hat{\theta}_A^{(0)} = 0.60$
$\hat{\theta}_B^{(0)} = 0.50$
1
$\hat{\theta}_A^{(1)} \approx \frac{21.3}{21.3 + 8.6} \approx 0.71$
$\hat{\theta}_B^{(1)} \approx \frac{11.7}{11.7 + 8.4} \approx 0.58$
3
M-step
4
$\hat{\theta}_A^{(10)} \approx 0.80$
$\hat{\theta}_B^{(10)} \approx 0.52$

实现

- 给定数据集和初值，就可以调用EM算法了：print em(observations,[0.6,0.5])
- 得到 [[0.7222502854992596, 0.5554380899384829 8], 36]
- 我们可以改变初值，试验初值对EM算法的影响。print em(observations,[0.5,0.6])
- 结果：[[0.5554372786904242 5, 0.7222509913921462 1], 37]
- 看来EM算法还是很健壮的。如果把初值设为相等会怎样？print em(observations,[0.3,0.3])
- 输出：[[0.6400000000000001, 0.6400000000000001], 1]
- 显然，两个值相加不为1的时候就会破坏这个EM函数。 换一下初值：
- print em(observations,[0.99999,0.00001])
- 输出：[[0.7222560629286650 7, 0.5554314500618421 4], 33]
- EM算法对于参数的改变还是有一定的健壮性的。

Thank you!

泰迪科技：www.tipdm.com
热线电话：40068-40020