聚类算法

K-Means
层次聚类
DBSCAN

聚类算法简介

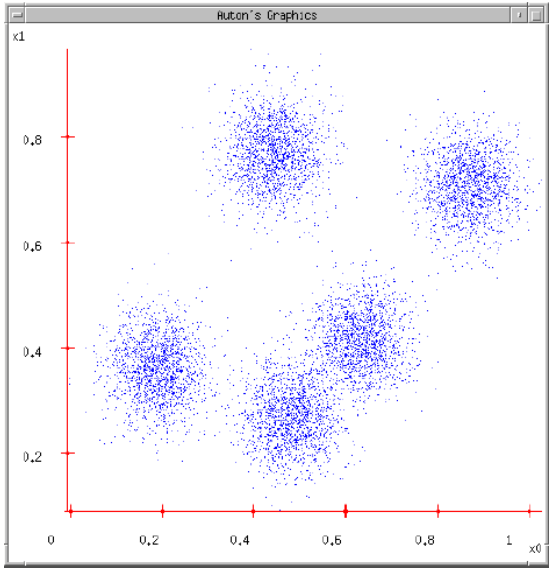
非监督学习

- 在监督学习任务中，我们有一系列标签的数据$D = \{X_i, y_i\}_{i=1}^{N}$，我们需要据此拟合一个函数，使得这个函数能表示输入x与输出y之间的关系。
- 在非监督学习中，我们只有一系列点$D = \{X_i\}_{i=1}^{N}$，却没有标签的数据y。
- 在非监督学习中，我们需要将一系列无标签的训练数据，输入到一个算法中，这个算法将寻找这个数据的内在结构。

聚类

- 聚类就是对大量未知标注的数据集，按数据的内在相似性将数据集划分为多个类别，使类别内的数据相似度较大而类别间的数据相似度较小。

划分原则



物以类聚、人以群分

- 作用
  - 计算：使用聚类中心而不是原始数据
  - 统计：识别/去除离群点（outliers）
  - 可视化/理解

相似度/距离计算方法总结

- 闵可夫斯基距离Minkowski/欧式距离

$$dist(X,Y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}$$

- 杰卡德相似系数(Jaccard)

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}$$

- 余弦相似度(cosine similarity)

$$\cos(\theta)=\frac{a^T b}{|a|\cdot|b|}$$

- Pearson相似系数

$$\rho_{XY}=\frac{\operatorname{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}=\frac{\sum_{i=1}^{n}(X_i-\mu_X)(Y_i-\mu_Y)}{\sqrt{\sum_{i=1}^{n}(X_i-\mu_X)^2}\sqrt{\sum_{i=1}^{n}(Y_i-\mu_Y)^2}}$$

聚类的基本思想

- 给定一个有N个对象的数据集，划分聚类技术将构造数据的k个划分，每一个划分代表一个簇，k≤n。也就是说，聚类将数据划分为k个簇，而且这k个划分满足下列条件：
  - 每一个簇至少包含一个对象
  - 每一个对象属于且仅属于一个簇

- 基本思想：对于给定的k，算法首先给出一个初始的划分方法，以后通过反复迭代的方法改变划分，使得每一次改进之后的划分方案都较前一次更好

02

PART TWO

K-MEANS

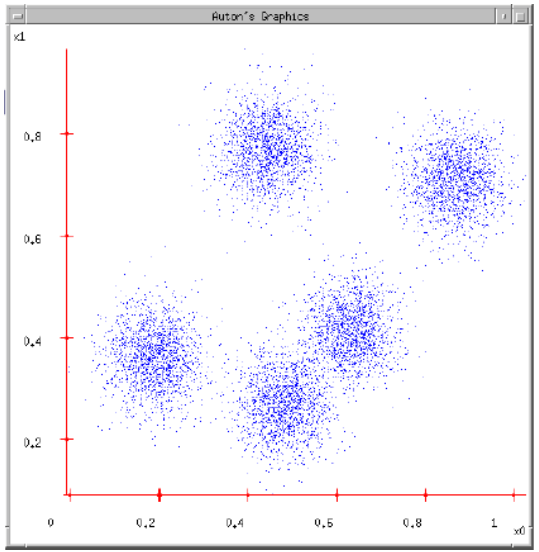
K-means

- K-means算法，也被称为k-平均或k-均值，是一种得到最广泛使用的聚类算法，或者成为其他聚类算法的基础。

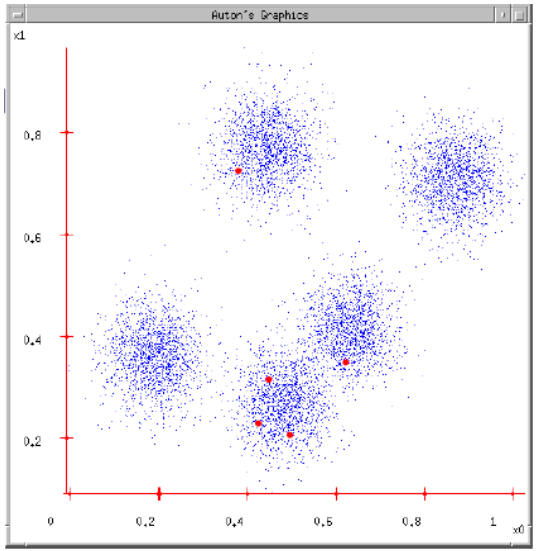
- 算法首先随机地选择k个对象，每个对象初始地代表了一个簇的平均值或中心。对剩余的每个对象根据其与各个簇中心的距离，将它赋给最近的簇。然后重新计算每个簇的平均值。这个过程不断重复，直到准则函数收敛。
  - 准则函数常常使用最小平方误差MSE
  - Minimum Squared-Error

K-means步骤(一)

1、给定数据

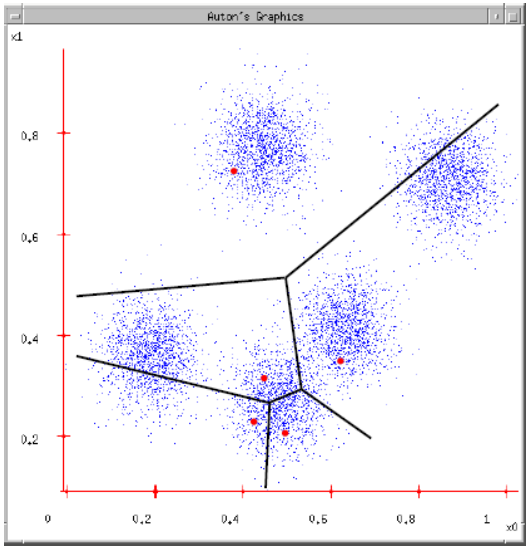


2、确定类别数K（如K=5），并初始化K个类的中心（如随机选择K个点）

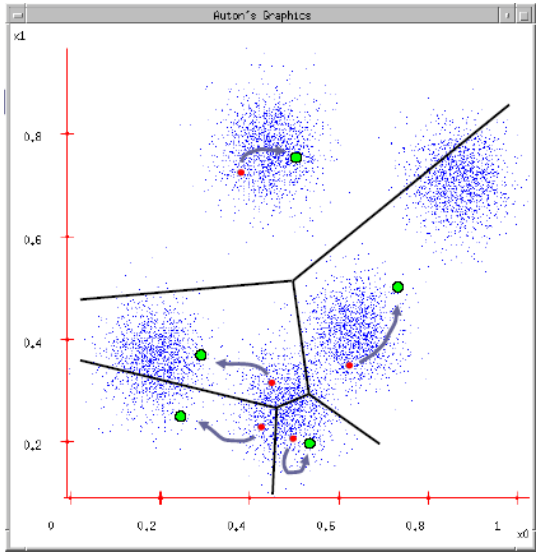


K-means步骤(二)

3、对每个数据点，计算离其最近的类（使得每个类拥有自己的一些数据）

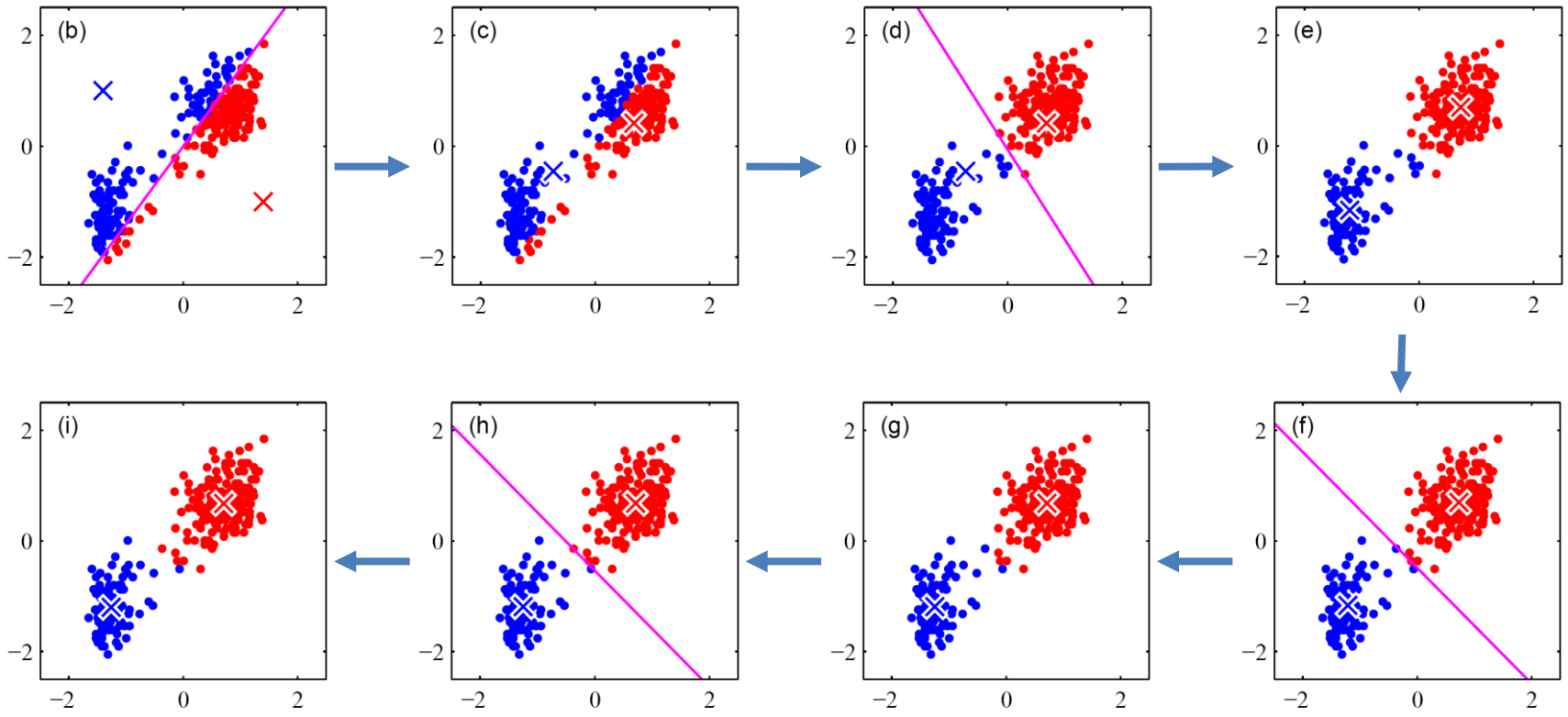


4、对每个类，计算其所有数据的中心，并跳到新的中心



5、重复3~4步，直到数据点所属类别类不再改变

K-means过程



K-means的优化目标

- 均值$\boldsymbol{\mu}$和数据点所属集合C的势能函数为

$$F(\boldsymbol{\mu}, C) = \sum_{i=1}^{N} \left(\boldsymbol{\mu}_{C(i)} - \mathbf{x}_i \right)^2$$

  - $C(i)$: 样本点i所属的簇

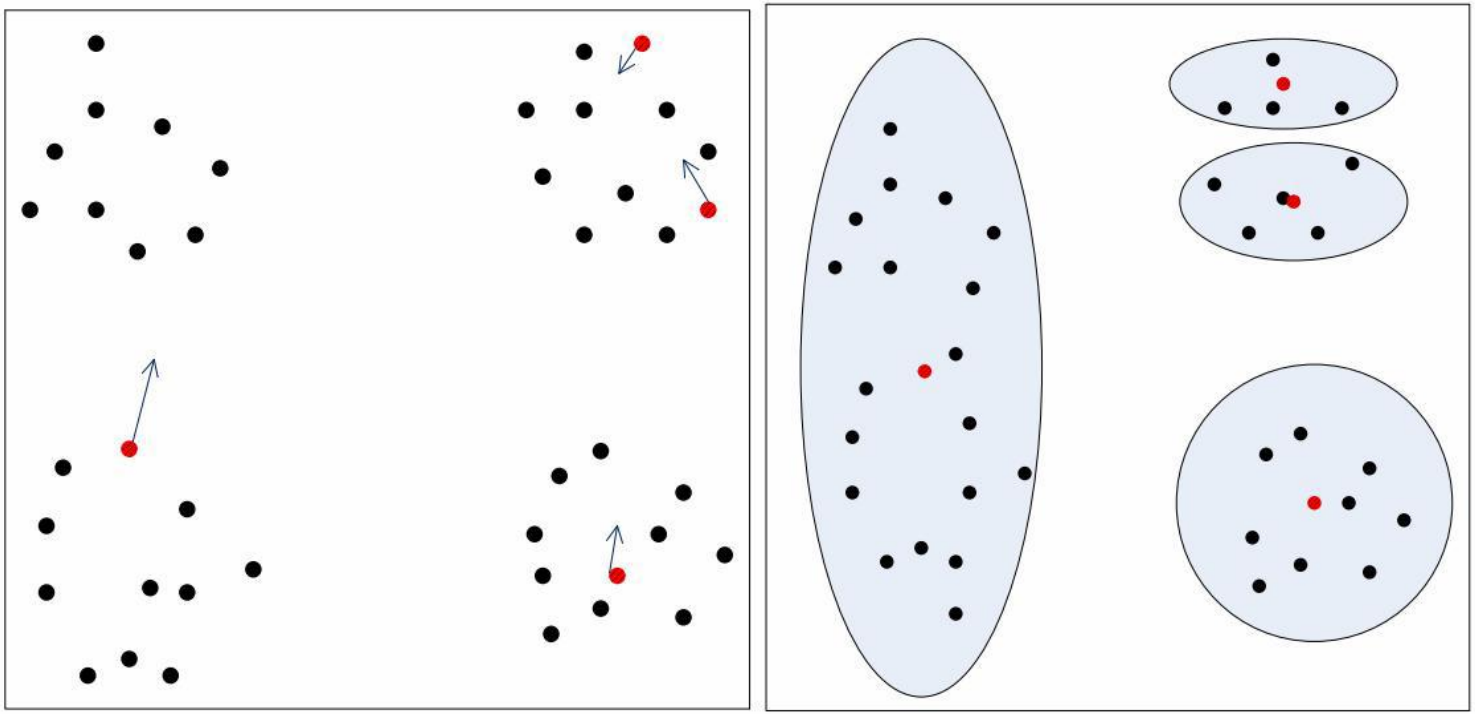
平方误差损失

- K-means的优化目标

$$\min_{\boldsymbol{\mu}} \min_{C} F(\boldsymbol{\mu}, C)$$

K-means初值敏感

K-means不保证达到全局最优（收敛，但没有达到全局最优的情况）



解决方案

- 仔细寻找初始值
  - 随机确定第一个类的中心，其他类中心的位置尽量远离已有类的中心;
  - Scikit learn中K-means实现中参数（ init='k-means++' ）将初始化centroids（质心）彼此远离，得到比随机初始化更好的结果。

- 重复多次（如10次），每次初始值不同，最后选择使目标函数最小的结果

K-means聚类的优缺点

- 优点
  - 是解决聚类问题的一种经典算法，简单、快速
  - 对处理大数据集，该算法保持可伸缩性和高效率
  - 当结果簇是密集的，它的效果较好

- 缺点
  - 在簇的平均值可被定义的情况下才能使用，可能不适用于某些应用
  - 必须事先给出k（要生成的簇的数目），而且对初值敏感，对于不同的初始值，可能会导致不同结果。
  - 不适合于发现非凸形状的簇或者大小差别很大的簇
  - 对躁声和孤立点数据敏感

Scikit learn中的K-means实现

- class sklearn.cluster.KMeans(n_clusters=8, init=’ kmeans++’ , n_init=10, max_iter=300, tol=0.0001, precompute_distances=’ auto’ , verbose=0, random_state=None, copy_x=True, n_jobs=1,algorithm=’ auto’)

  - n_clusters ：聚类数量，默认为8
  - init: 初始化质心，可以是 ‘k-means++’ 、 ‘random’ 或者数组
  - n_init: 重复次数。为了弥补初始质心的影响而使算法陷入局部极值点，算法默认会初始10个质心，实现聚类，然后返回多次聚类的最好结果。
  - precompute_distances： 这个参数会在空间和时间之间做权衡
    - True ： 存储样本之间的距离（快但是占用更多内存）
    - False： 不预计算存储样本之间的距离
    - auto ： 在数据样本大于featurs*samples 的数量大于12M时设置为False

03

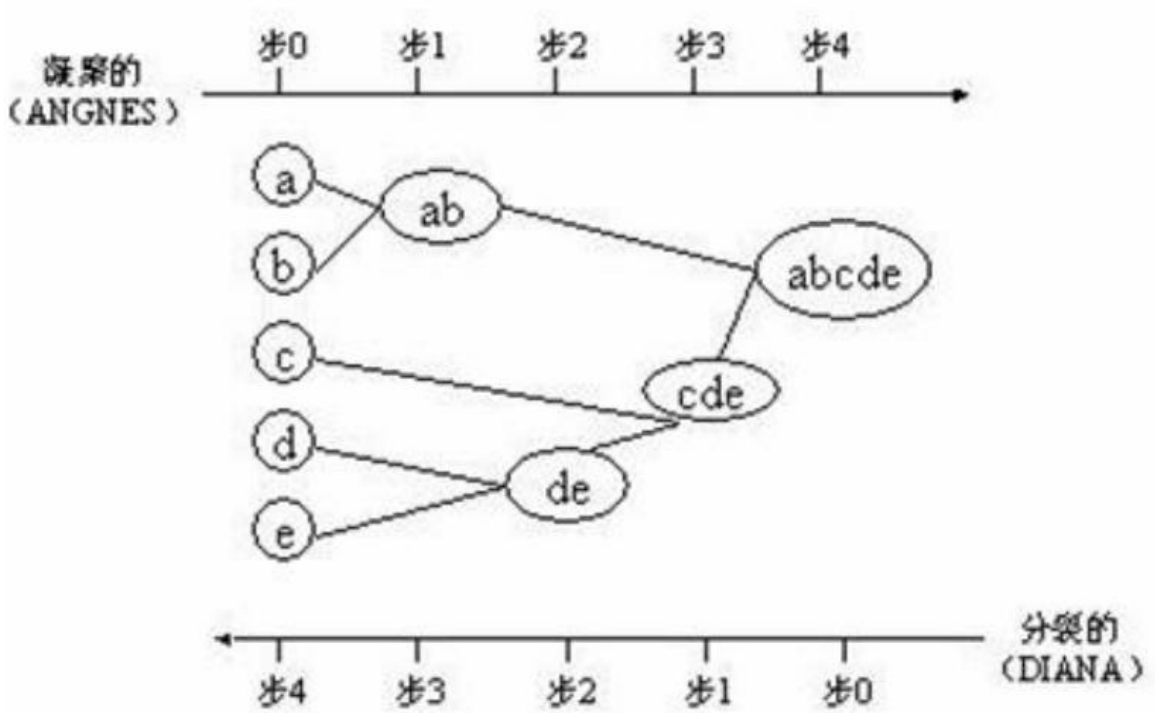
PART THREE

层级聚类

层次聚类方法

- 层次聚类 (Hierarchical clustering, 也称系统聚类法) 是最经 典和常用的聚类方法之一
  - 能找到任意形状的类，而且不需指定别数 能找到任意形状的类，而且不需指定别数 K
  - 需要 度量样本点之间的 距离以及类 与距离以及类 与之间的联接 之间的联接 (linkage) 程度
- 系统聚类法包括两种
  - 聚合方法 (agglomerative) / 自下而上 自下而上 :
    - 初始：每个样本作为 单独的一 类
    - 聚合：根据 类间联接程度， 合并相近的类，直到所有成一个合并相近的类，直到所有成一个
  - 分裂 方法 (divisive) / 自上而 自上而 下
    - 初始：所有样本置于一个类
    - 分裂：一 个类不断 地分为 更小的类，直到每个样本体单独为一更小的类，直到每个样本体单独为一更小的类，直到每个样本体单独为一更小的类，直到每个样本体单独为一更小的类，直到每个样本体单独为一更小的类，直到每个样本体单独为一更小的类，直到每个样本体单独为一个类

层次聚类


凝聚的
（ANGNES）
步0
步1
步2
步3
步4
a
b
c
d
e
ab
abcde
cde
de
步4
步3
步2
步1
步0
分裂的
（DIANA）

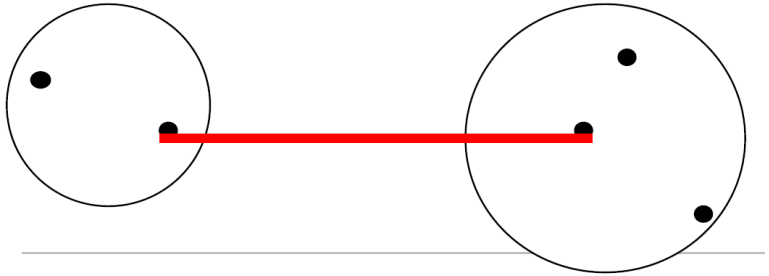
类间联系程度度量

在合并两个最近的类别时，需要度量两个类别之间的距离。距离度量可选：

1.最小距离法

- 极小异常值在实际中不多出现，避免极大值的影响
- 趋向于产生一个长链

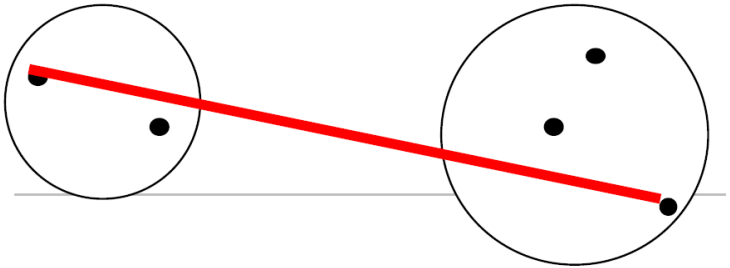
$$d_{\min}\left(C_k, C_{k'}\right) = \min_{x \in C_k, x' \in C_{k'}} \left\|\mathbf{x} - \mathbf{x}'\right\|$$



2. 最大距离法

- 可能被异常极大值扭曲，删除这些值之后再聚类
- 趋向于产生产生一些紧致的类别

$$d_{\max}\left(C_k, C_{k'}\right) = \max_{\mathbf{x} \in C_k, \mathbf{x}' \in C_{k'}} \left\|\mathbf{x} - \mathbf{x}'\right\|$$



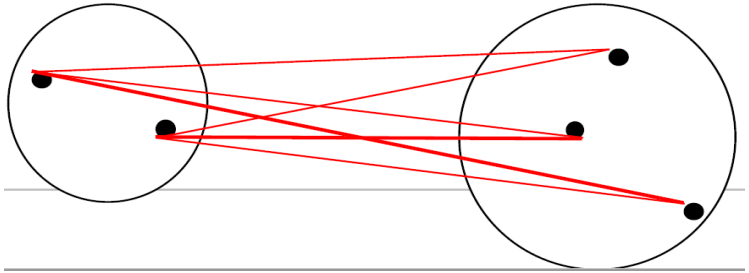
类间联系程度度量

在合并两个最近的类别时，需要度量两个类别之间的距离。距离度量可选：

3. 类平均距离法

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

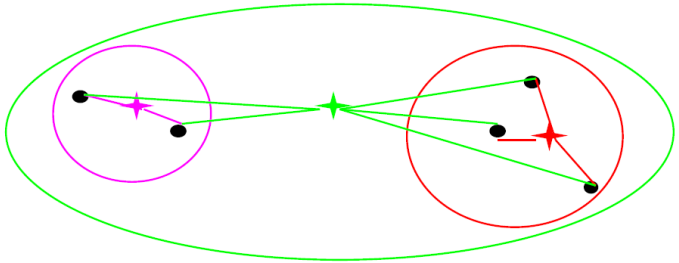
$$d_{avg}(C_k, C_{k'}) = \frac{1}{N_k N_{k'}} \sum_{\mathbf{x} \in C_k} \sum_{\mathbf{x}' \in C_{k'}} \|\mathbf{x} - \mathbf{x}'\|$$



4. 离差平方和法

- 分别计算类Ck和Ck’ 内各点到其均值的欧氏距离平方和Wk和Wk’ ；然后将两个类的点合并为一个类Cm并计算Wm

$$d_{ward} = W_{\mathrm{m}} - W_{C_k} - W_{C_{k'}}$$



- 对异常值很敏感；
- 对较大的类倾向产生较大的距离，从而不易合并，较符合实际需要。

Scikitlearn中的层次聚类

- sklearn.cluster.AgglomerativeClustering(n_clusters=2, affinity='euclidean', memory=None, connectivity=None, compute_full_tree='auto', linkage='ward', pooling_func=<function mean at 0x174b938>)

  - n_clusters ：聚类数量，默认为2
  - affinity：距离度量。默认为”euclidean”;如果linkage选择为 ‘ward’ ,只能用”euclidean”；
  - **linkage** ：{“ward”， “complete”， “average”}, optional, 默认为“ward”
    - ward：最小距离法
    - complete：最大距离法
    - average ：类平均距离法

层次聚类的总结

- 得到层次化的结果，而不是一堆无组织类别
  - 一棵表示类别及其之间距离的树 (称为 dendrogram dendrogram)
- 层次聚类是一个确定的过程：没有参数
  - 如果只想得到 K类，则剪掉 K-1个最长的类

- 但对层次聚类没有统计和信息基础
- 运算量较大，适用于不太大规模的数据
- 一旦完成 一个 步骤 (合并或分裂)，就不能撤销或修正

04

PART FOUR

DBSCAN

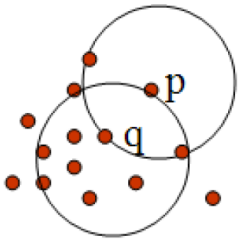
基于密度的聚类方法

- 密度聚类方法的指导思想是，只要一个区域中的点的密度大于某个阈值，就把它加到与之相近的聚类中去。
- 能克服基于距离的算法只能发现“类圆形”(凸)的聚类的缺点，可发现任意形状的聚类，且对噪声数据不敏感。但计算密度单元的计算复杂度大，需要建立空间索引来降低计算量。
- 但计算密度过程的计算复杂度大，需要建立空间索引来降低计算量，且对数据维的伸缩性较差 。
- DBSCAN是一个有代表性的基于密度的聚类算法 。可在有“噪声”的空间数据库中发现任意形状。

基本概念

DBSCAN基本思想涉及的一些概念：

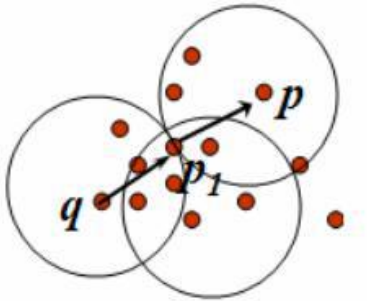
- 对象的ε-邻域：给定对象在半径ε内的区域。
- 核心对象：对于给定的数目Minpts为m，如果一个对象的ε-邻域至少包含m个对象，则称该对象为核心对象。
- 直接密度可达：给定一个对象集合D，如果p是在q的ε-邻域内，而q是一个核心对象，我们说对象p从对象q出发是直接密度可达的

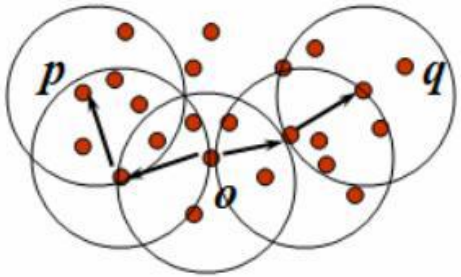


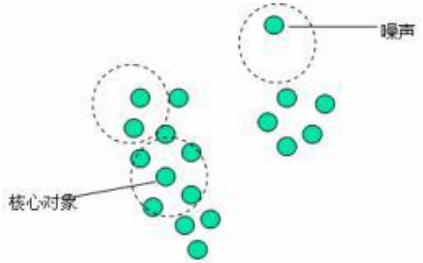
- ε=1，MinPts=5，q是一个核心对象
- 对象p从对象q出发是直接密度可达的

基本概念

- 密度可达：如果存在一个对象链$p_1, p_2, \cdots, p_n$， p_1 =q， p_n =p，对$p_i \in$D，(1≤i≤n)， p_{i+1}是从p_i关于ε和m直接密度可达的，则对象p是从对象q关于ε和m密度可达的。
- 密度相连：如果对象集合D中存在一个对象o，使得对象p和q是从o关于ε和m密度可达的，那么对象p和q是关于ε和m密度相连的。
- 簇：一个基于密度的簇是最大的密度相连对象的集合。
- 噪声







DBSCAN

DBSCAN通过检查数据集中每个对象的ε-邻域来寻找聚类。

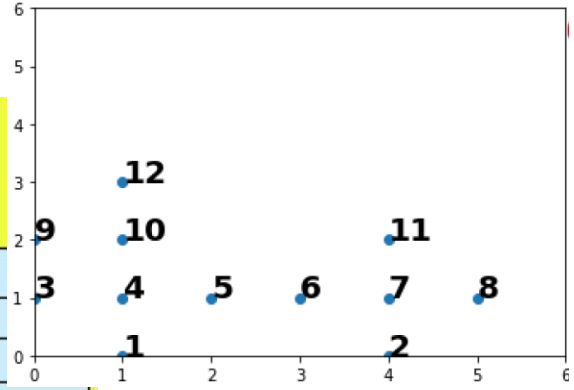
- 参数：
  - 给定聚类对象的半径ε-邻域
  - ε-邻域中最小包含的对象数 MinPts
- 如果一个点p的ε-邻域包含多于Minpts个对象，则创建一个p作为核心对象的新簇。
- 反复地寻找从这些核心对象直接密度可达的对象，这个过程可能涉及密度可达簇的合并。当没有新的点可以被添加到任何簇时，该过程结束。
  - 一个基于密度的簇是可达性最大相连对象集合
- 不包含在任何簇中的对象被认为是“噪声”。

DBSCAN

DBSCAN通过检查数据集中每个对象的ε-邻域来寻找聚类。

- 参数：
  - 给定聚类对象的半径ε-邻域
  - ε-邻域中最小包含的对象数 MinPts
- 如果一个点p的ε-邻域包含多于Minpts个对象，则创建一个p作为核心对象的新簇。
- 反复地寻找从这些核心对象直接密度可达的对象，这个过程可能涉及密度可达簇的合并。当没有新的点可以被添加到任何簇时，该过程结束。
  - 一个基于密度的簇是可达性最大相连对象集合
- 不包含在任何簇中的对象被认为是“噪声”。

例子



下面给出一个样本事务数据库（见左表），对它实施DBSCAN算法。
以下为算法的步骤（设n=12，用户输入ε =1，MinPts=4）

样本事务数据库

序号	属性 1	属性 2
1	1	0
2	4	0
3	0	1
4	1	1
5	2	1
6	3	1
7	4	1
8	5	1
9	0	2
10	1	2
11	4	2
12	1	3

DBSCAN算法执行过程示意

步骤	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
1	1	2	无
2	2	2	无
3	3	3	无
4	4	5	簇C_1：{1，3，4，5，9，10}
5	5	3	已在一个簇C_1中
6	6	3	无
7	7	5	簇C_2：{2，6，7，8，11}
8	8	2	已在一个簇C_2中
9	9	3	已在一个簇C_1中
10	10	4	簇C_1：{1，3，4，5，9，10，12}
11	11	2	已在一个簇C_2中
12	12	2	已在一个簇C_1中

聚出的类为{1，3，4，5，9，11，12}，{2，6，7，8，10}。

Scikitlearn中DBSCAN算法的实现

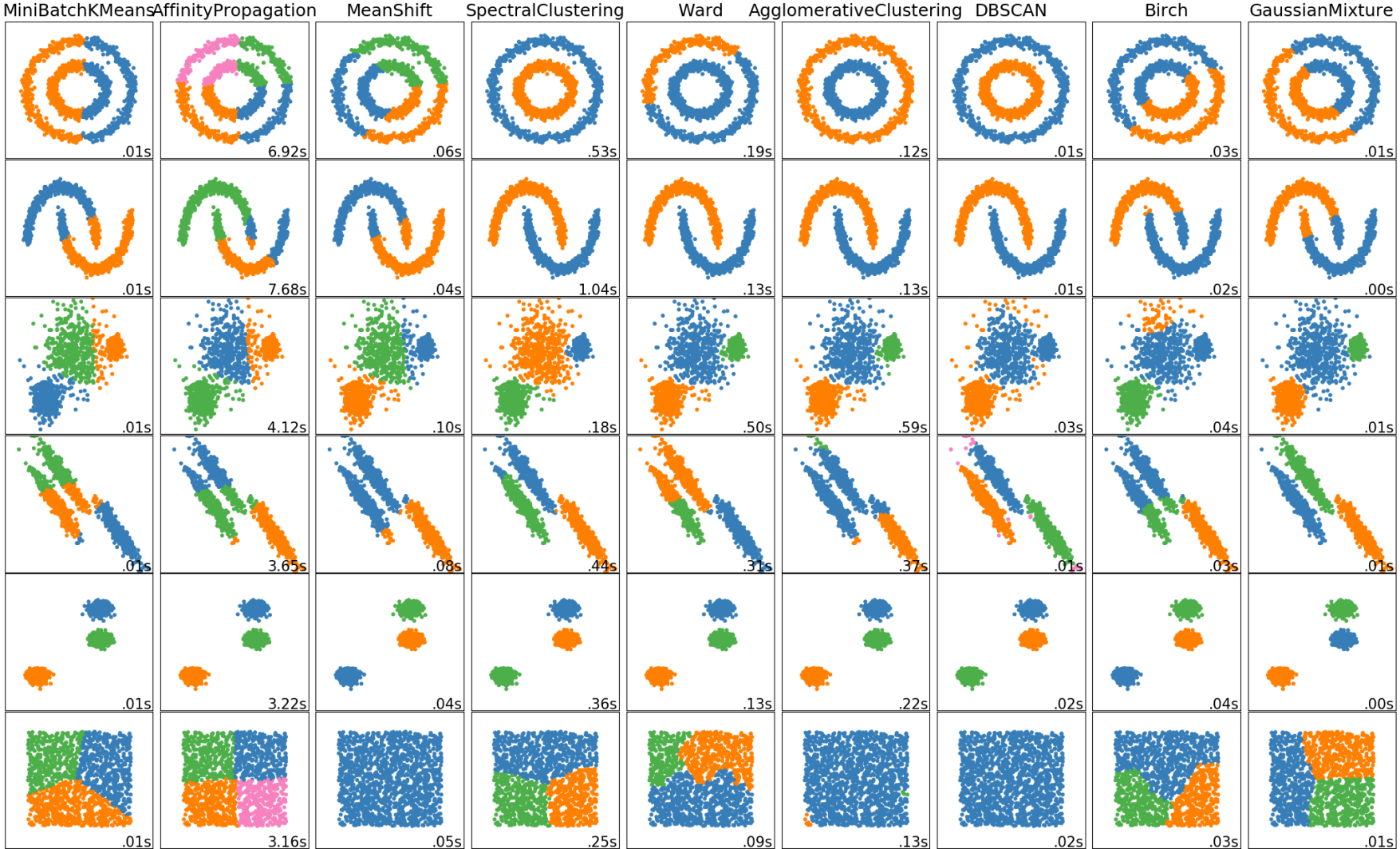
- class sklearn.cluster.DBSCAN(eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=1)

- Eps:为ε，定于邻域大小
- min_samples:为MinPts，定于核心节点的条件，即邻域内样 本点的最小数目。
- algorithm为最近邻的搜索算法，可‘auto’，‘ball_tree’，‘kd_tree’，‘brute’

05

PART FIVE

案例分析


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
GaussianMixture
.01s
6.92s
.06s
.53s
.19s
.12s
.01s
.03s
.01s
.01s
7.68s
.04s
1.04s
.13s
.13s
.01s
.02s
.00s
.01s
4.12s
.10s
.18s
.50s
.59s
.03s
.04s
.01s
.01s
3.65s
.08s
.44s
.31s
.37s
.01s
.03s
.01s
.01s
3.22s
.04s
.36s
.13s
.22s
.02s
.04s
.00s
.01s
3.16s
.05s
.25s
.09s
.13s
.02s
.03s
.01s

案列分析

MNIST手写数字识别聚类