金融科技学

李彦

liyan_zjgsu.163.com

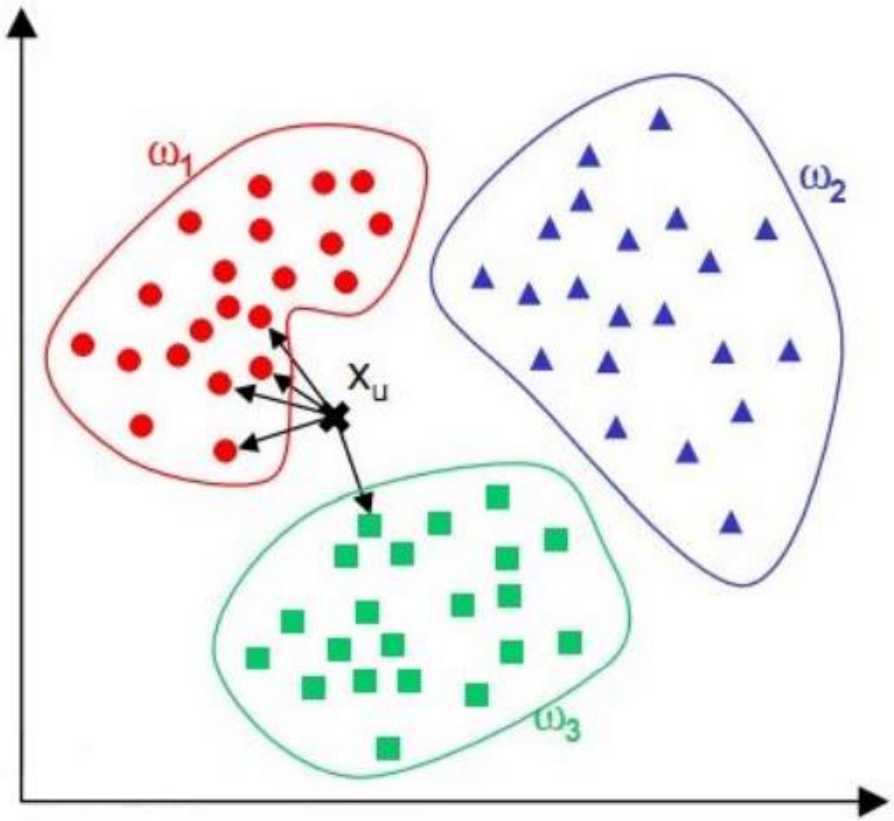
Preview

- 分类：KNN

- 聚类：
- Data scale
- Measure of Similarity
- K-means：how to define k?
  - K-means++，K-memoids…

KNN算法

- **K近邻(KNN，k-NearestNeighbor)分类算法**是分类技术中最简单的方法之一：
  - kNN算法的**核心思想**是：每个样本都可以用它最接近的k个邻居来代表
  - K值选取：交叉验证
  - *sklearn.neighbors.KNeighborsClassifier*



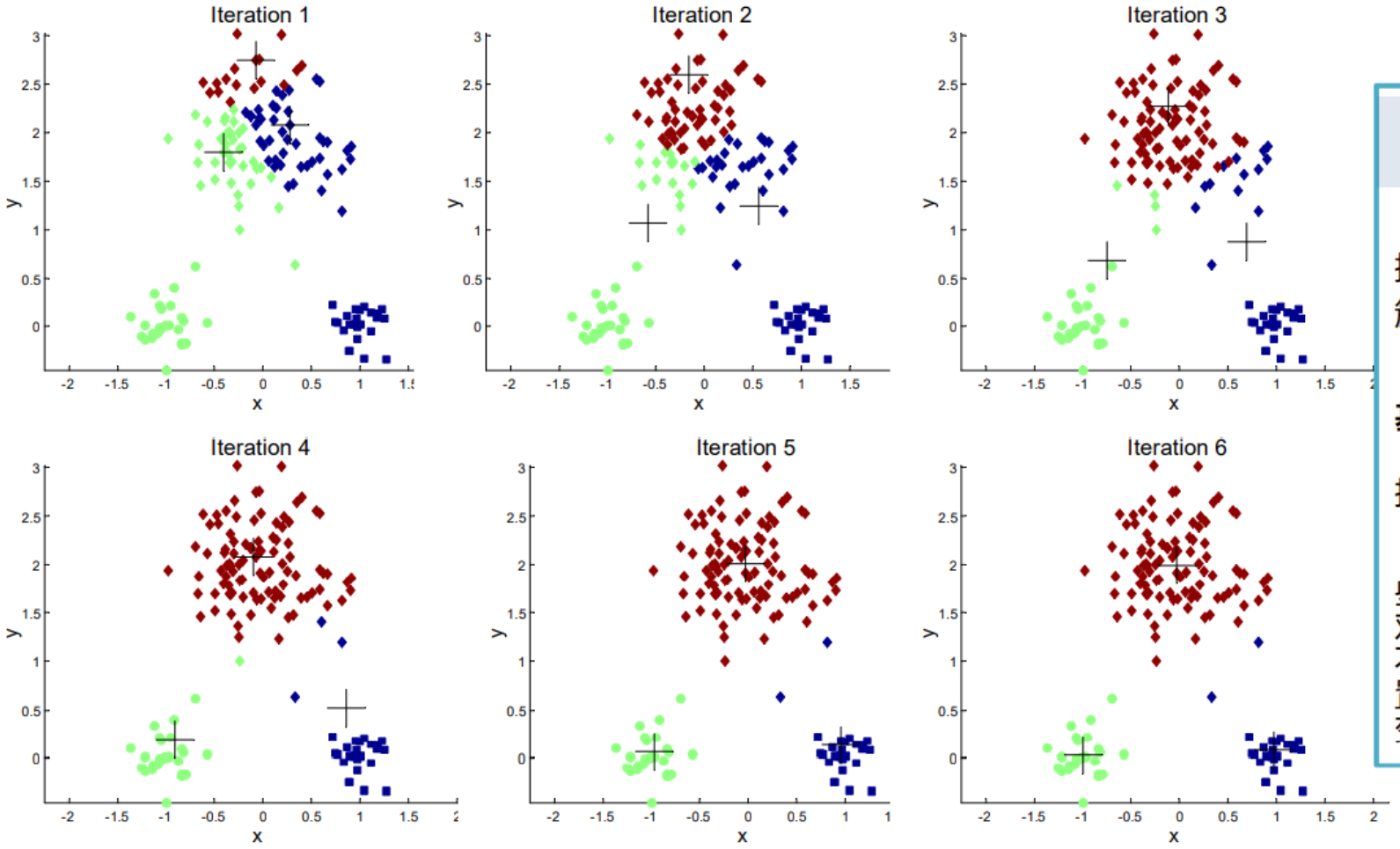
什么是聚类分析

- 聚类是无监督学习方法
- **目标：**是使同一个簇中的样本相似度较高，而不同簇间的样本相似度较低

- 距离：欧几里得、曼哈顿、切比雪夫、闵可夫斯基……
- 相似性：余弦相似度

- 数据预处理：标准化

K-means聚类过程图示



应用实例

利用K-means聚类算法，把原始数据聚成三个不同的簇的应用实例如左图示（K=3）。

基本思路：

（1）首先，随机选择k个数据点做为聚类中心；

（2）然后，计算其它点到这些聚类中心点的距离，通过对簇中距离平均值的计算，不断改变这些聚类中心的位置，直到这些聚类中心不再变化为止。

K-means算法停止条件

- 设定迭代次数。
- 聚类中心不再变化 。
- 前后两次聚类结果的目标函数SSE变化很小。

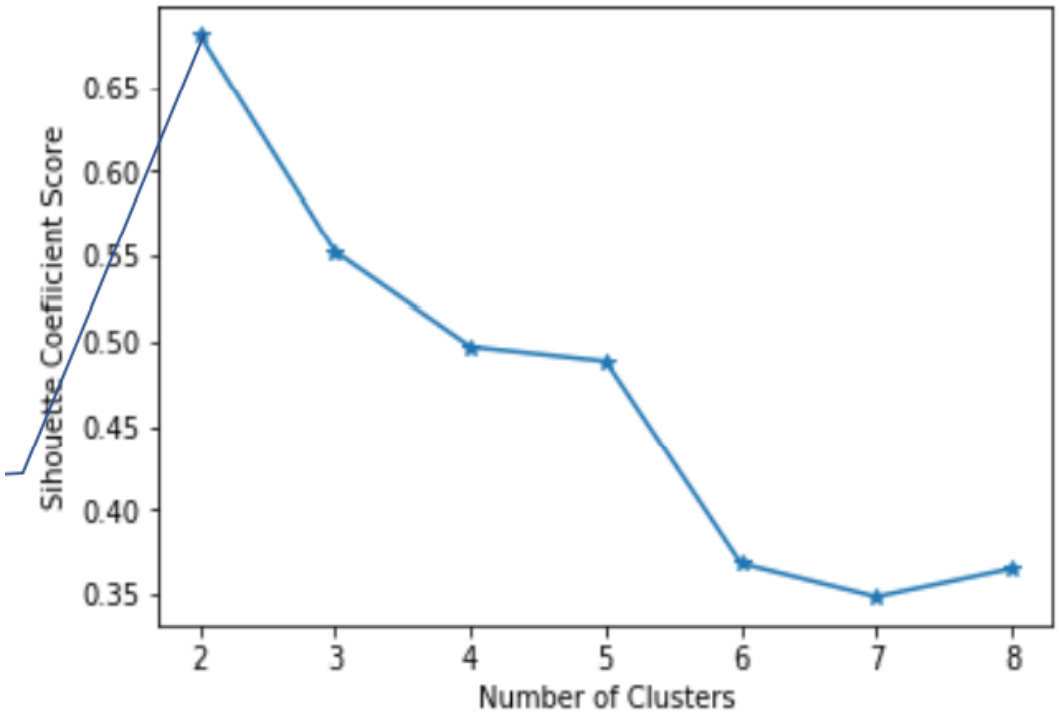
K-means Improvement

- K-means++
- 核心思想：**使初始的聚类中心之间的相互距离尽可能远**。
- Distance → Probability → Initial centroids

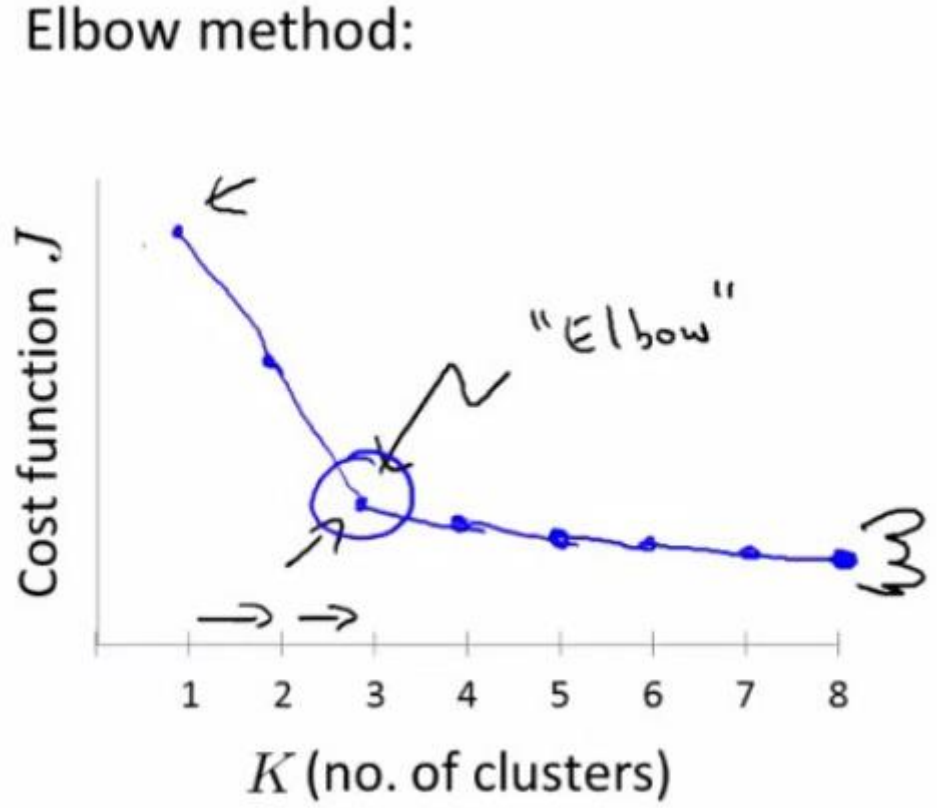
- K—medoids
- 核心思想：**不通过计算簇中所有样本的平均值得到簇的质心，而是选取原有样本中的样本点作为质心**
- 不断用非中心点替换中心点，迭代使得聚类的质量不能再被提升

k值选择：

- 选取轮廓系数最大的k值



- 肘部法则



K-means

- 局限性：不适用于非凸面形状（非球形）的数据集
- 实现：sklearn.cluster.KMeans

聚类方法性能评估

- **有分类标签**的数据集
  - 使用**兰德指数**（ARI，Adjusted Rand Index）
  - 计算真实标签与聚类标签两种分布之间的相似性，取值范围为[0,1]
  - 1表示最好的结果，即聚类类别和真实类别的分布完全一致
  - 鸢尾花数据集带有标签

```
from sklearn import metrics
metrics.adjusted_rand_score(y, kmeans.labels_)
```

- **没有分类标签**的数据集
  - 使用**轮廓系数**（Silhouette Coefficient）来度量聚类的质量
  - 轮廓系数同时考虑聚类结果的簇内凝聚度和簇间分离度
  - 取值范围：[-1,1]，轮廓系数越大，聚类效果越好

```
from sklearn import metrics
metrics.silhouette_score( X,kmeans.labels_,met
    ric='euclidean' )
```

机器学习算法的性能度量——分类

混淆矩阵（confusion matrix），误差矩阵

功能：主要用于比较分类结果和实例的真实信息

准确率（accuracy）：正确预测的正反例数 /总数

$$Accuracy=\frac{TP+TN}{TP+FN+FP+TN}$$

以二分类（正、负）为例：

真正(TP)：模型预测为正的正样本

假正(FP)：模型预测为正的负样本

假负(FN)：模型预测为负的正样本

真负(TN)：模型预测为负的负样本

兰德指数(Rand index, RI)

- RI取值范围为[0,1]，值越大意味着聚类结果与真实情况越吻合。
- 如果有了类别标签，那么聚类结果也可以像分类那样计算准确率和召回率。
- 假设U是外部评价标准，即true_label， 而V是聚类结果，根据**元素对**在类标签和聚类结果中的表现设定4个统计量：

	Same Cluster	Different Cluster	Sum U
Same Class	TP / a	FN / b	a+b
Different Class	FP / c	TN / d	c+d
Sum V	a+c	b+d	a+b+c+d

- RI定义为“正确决策”的比率：$RI = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{C_N^2} = \frac{a + b}{C_2^{n_{samples}}}$

轮廓系数（Silhouette Coefficient）

- 对于n个对象的数据集D，假设D被划分为k个簇。
- 对于D中的每个对象o，计算它和**它所属的簇的其他对象的平均距离a(o)**。类似的，计算它和**它不属于的簇的最小平均距离b(o)**。
  - a：簇内距离；b：簇间距离

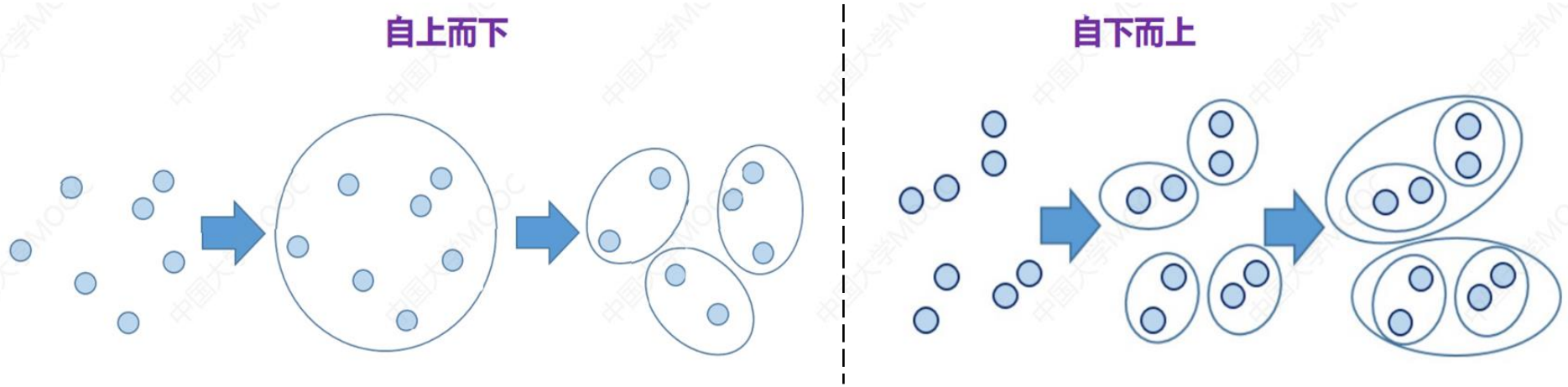
- 轮廓系数公式：$$s(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$$

- 对数据集中的每个对象计算轮廓系数然后取**平均值**作为聚类的质量度量。
- 轮廓系数的取值范围是[-1,1]，**越靠近1代表聚类质量越好，越靠近-1代表聚类质量越差**。

目录

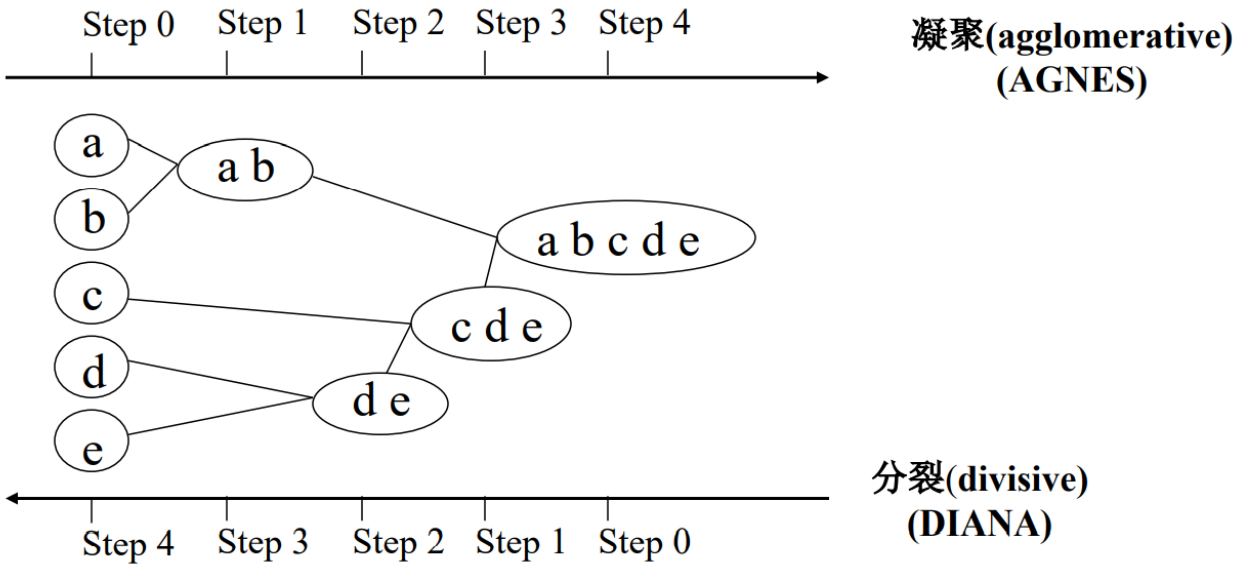
基于层次的方法

- 应用广泛程度仅次于基于划分的聚类，核心思想就是通过将数据集划分到不同层次的簇，形成一个**树形**的聚类结构。
- 层次聚类算法可以**揭示数据的分层结构**，从而得到不同粒度的聚类结果。
  - 按照层次聚类的过程分为：自底向上的**聚合聚类**和自顶向下的**分裂聚类**。
  - 聚合聚类：AGNES、BIRCH、ROCK等算法；分裂聚类：DIANA算法



基于层次的方法

- 层次聚类（hierarchical clustering）方法把数据组织成若干簇，并形成一个相应的**树状图**进行聚类。
- **聚合层次聚类采用自底向上的策略**，首先把每个对象单独作为一类，然后根据一定的规则，例如把簇间距离最小的相似簇合并成为越来越大的簇，直到所有样本凝聚成一个大的簇，针对给定应用选择最好结果的聚类层次。
- 与聚合型方法相反，**分裂聚类采用自顶向下的方法**，先把所有的对象都看成一个簇，然后不断分解直至满足一定的条件。



聚合算法agnes的实现

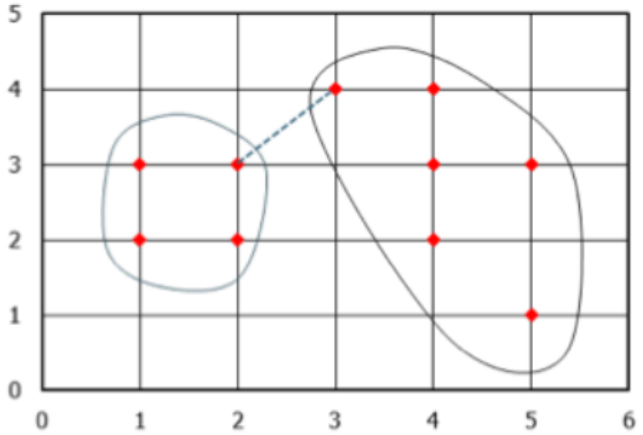
- 输入：包含n个对象的数据集合。
- 输出：满足终止条件的若干个簇。
  1. 将每个对象当成一个初始簇;
  2. **计算任意两个簇的距离**，并找到最近的两个簇;
  3. 合并两个簇，生成新的簇的集合
  4. 重复上述步骤直至终止条件得到满足。

- 如何计算**簇间距离**?

簇间距离度量方法

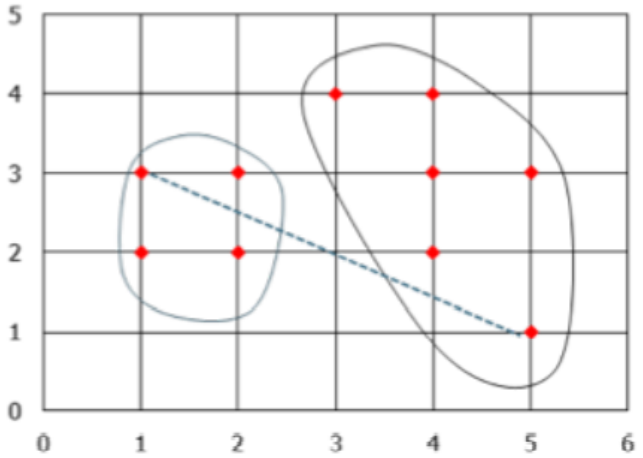
最小距离(minimum distance)，即单链接
Single link: 基于来自两个簇中的结点之间的最小距离来衡量两个簇的相似度, 即：

$$dis(C_i, C_j) = \min(o_{ip}, o_{jq})$$



最大距离(maximum distance)，即全链接
Complete link: 基于来自 两个簇中的结点之间的最大距离来衡量两个簇的相似度, 即：

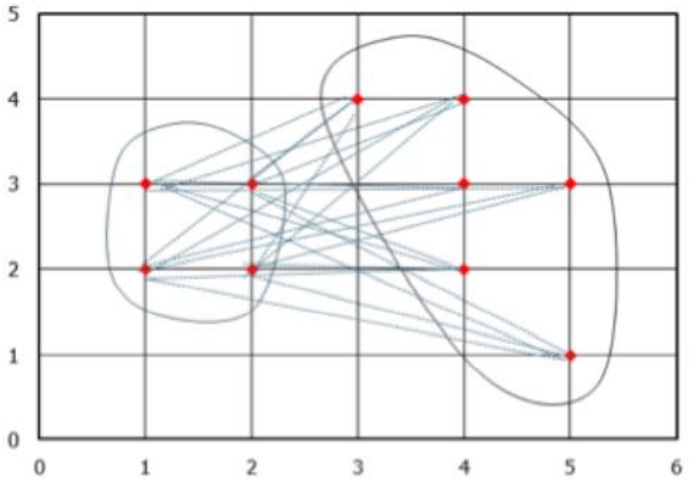
$$dis(C_i, C_j) = \max(o_{ip}, o_{jq})$$



簇间距离度量方法

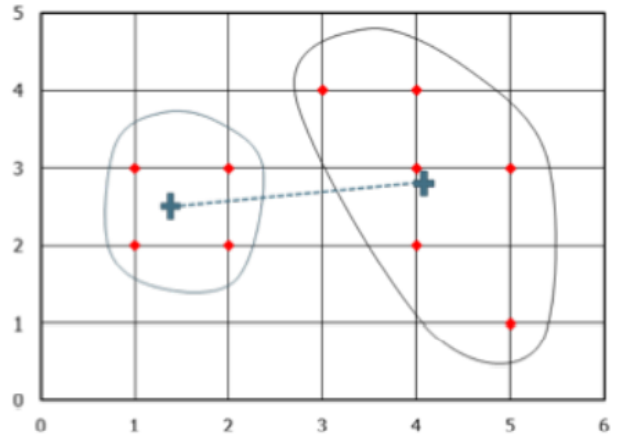
平均距离(average distance)，即链接Average link: 基于来自两个簇 中的结点之间的平均距离来衡量两个簇的相似度, 即：

$$dis(C_i, C_j) = avg(o_{ip}, o_{jq})$$



质心距离:计算两个簇的质心之间的距离来衡量两个簇的相似度, 即：

$$dis(C_i, C_j) = dis(c_i, c_j)$$



Agnes算法示例

- 使用AGNES算法对下面的数据集进行聚类，以**平均距离**计算簇间的距离。
- 刚开始共有5个簇：C1={A}、C2={B}、C3={C}、C4={D}和C5={E}
- 距离矩阵：

样本点	A	B	C	D	E
A	0	**0.4**	2	2.5	3
B	**0.4**	0	1.6	2.1	1.9
C	2	1.6	0	0.6	0.8
D	2.5	2.1	0.6	0	1
E	3	1.9	0.8	1	0

- Step1. 簇C1和簇C2的距离最近，将二者合并，得到新的簇结构：C1={A,B}、C2={C}、C3={D}和C4={E}。

Agnes算法示例

- 距离矩阵：

样本点	AB	C	D	E
AB	0	1.6	2.1	1.9
C	1.6	0	**0.6**	0.8
D	2.1	**0.6**	0	1
E	1.9	0.8	1	0

- 如何计算组合数据点与其他数据点间的距离？
  - 当我们计算(A,B)到C的距离时，需要分别计算A到C和B到C的距离**均值**。
- Step2. 簇C2和簇C3的距离最近，将二者合并，得到新的簇结构：C1={A,B}、C2={C,D}和C3={E}。

Agnes算法示例

- 距离矩阵：

样本点	AB	CD	E
AB	0	1.6	1.9
CD	1.6	0	**0.8**
E	1.9	**0.8**	0

- 平均距离： dist(AB, CD) = [dist(A,C)+dist(A,D)+dist(B,C)+dist(B,D)]/4
- Step3. 簇C2和簇C3的距离最近，将二者合并，得到新的簇结构：C1={A,B}和C2={C,D,E}。

Agnes算法示例

- 距离矩阵：

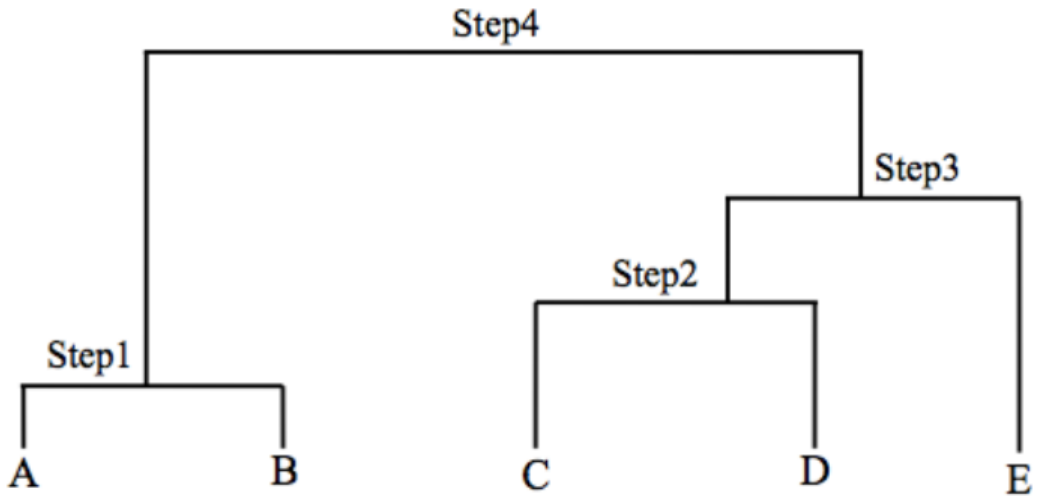
样本点	AB	CDE
AB	0	**1.6**
CDE	**1.6**	0

- Step4. 最后只剩下簇C1和簇C2，二者的最近距离为1.6，将二者合并，得到新的簇结构：C1={A,B,C,D,E}。

基于层次的方法评价

- 层次聚类产生具有**层次关系**的簇
- **不需要指定簇的个数 k**
- 复杂度高

- 在层次聚类算法的实现中，聚类通常**终止**于某个预先设定的条件。比如簇的数目、簇的直径、簇间距离等。

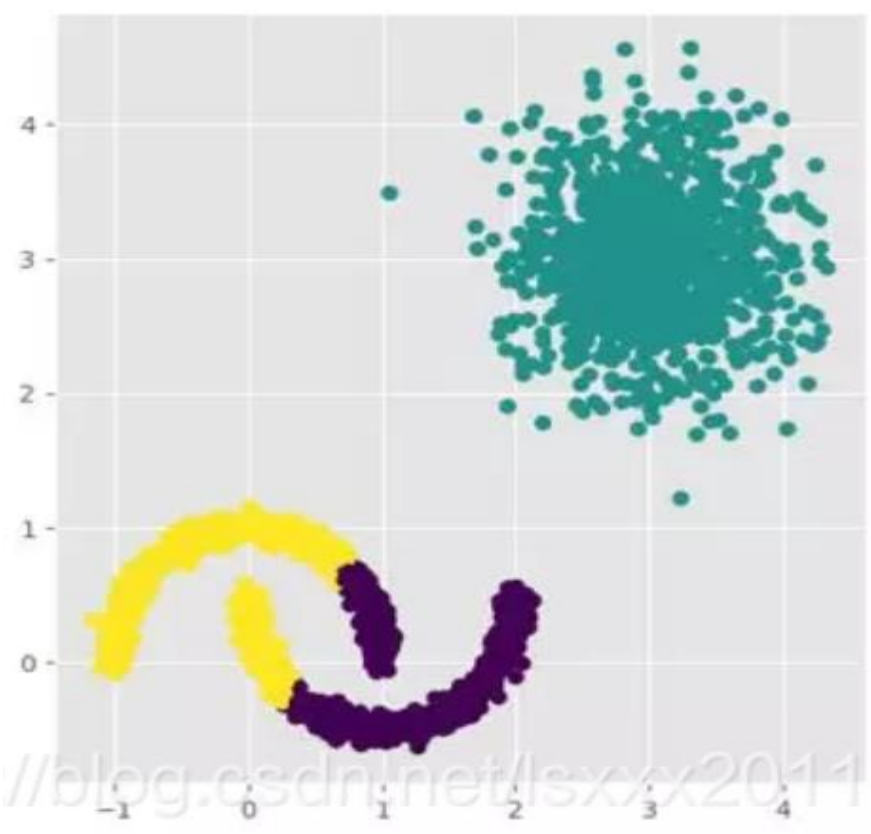


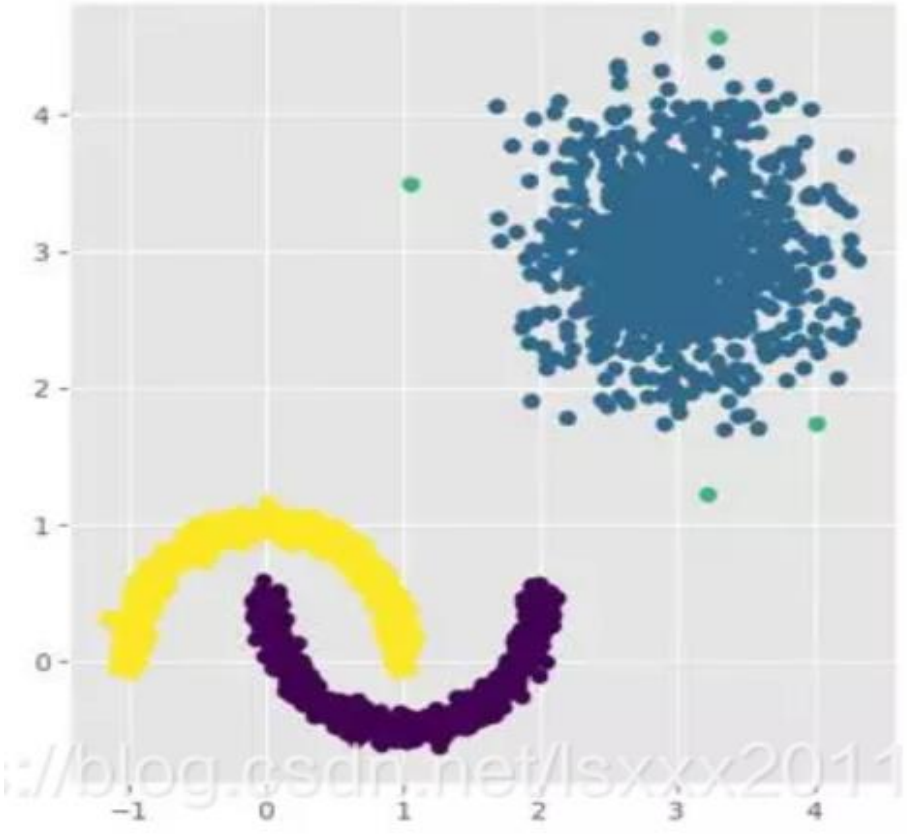
AGNES聚类过程

基于密度的方法

- 基于划分聚类和基于层次聚类的方法在聚类过程中根据**距离**来划分类簇，因此只能够用于挖掘球状簇。
- 为了解决这一缺陷，基于密度聚类算法利用密度思想，将样本中的高密度区域(即样本点分布稠密的区域)划分为簇，**将簇看作是样本空间中被稀疏区域(噪声)分隔开的稠密区域**。
- 这一算法的主要目的是过滤样本空间中的稀疏区域，获取稠密区域作为簇。
- 基于密度的聚类算法是**根据密度而不是距离**来计算样本相似度，所以基于密度的聚类算法**能够用于挖掘任意形状的簇**，并且能够有效过滤掉噪声样本对于聚类结果的影响。

基于密度的方法

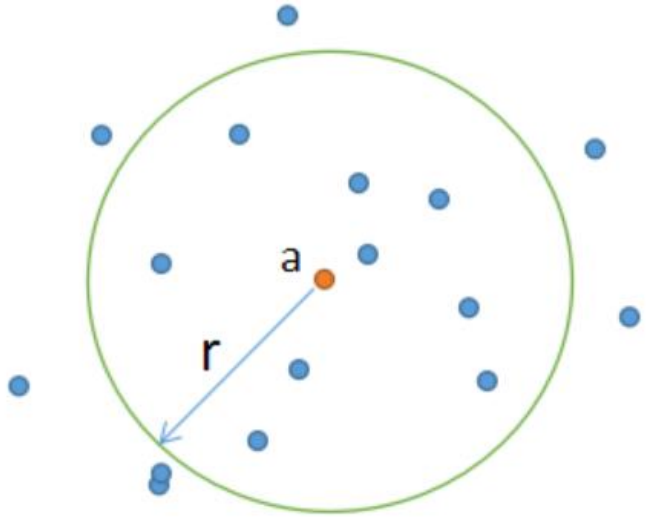




DBSCAN算法的基本概念

- DBSCAN采用基于中心的密度定义，样本的密度通过核心对象在邻域半径内的样本点个数（包括自身）来估计。
- DBSCAN算法基于邻域来描述样本的密度：**{ε, MinPts}**
- ε：样本的邻域距离阈值
- MinPts：对于某一样本，其ε-邻域中样本个数的阈值

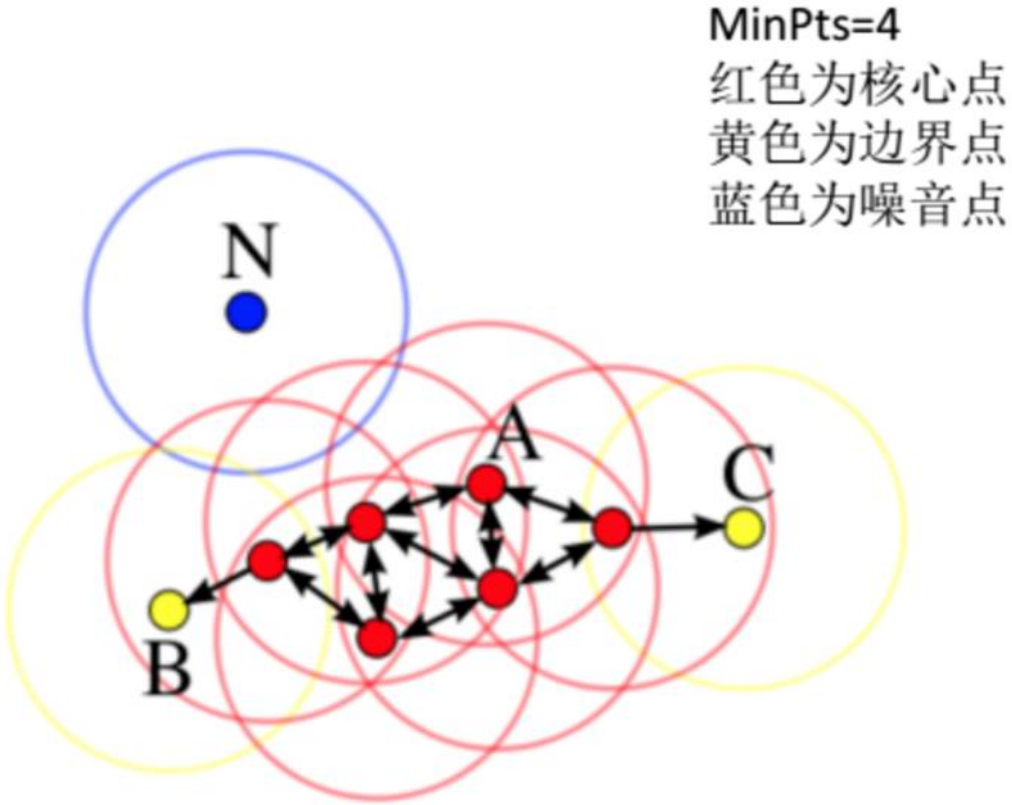
点a在半径r下的密度：



DBSCAN算法的基本概念

- **核心点**：
  - 在ε-邻域内，点的数量超过MinPts（包含自己）的点
- **边界点**：
  - 在ε-邻域内，点的数量小于MinPts，且为核心点的直接邻居的点
- **噪声点**：
  - 既不是核心点也不是边界点的点

- **噪声点是不会被聚类纳入的点，边界点与核心点组成聚类的“簇”。**



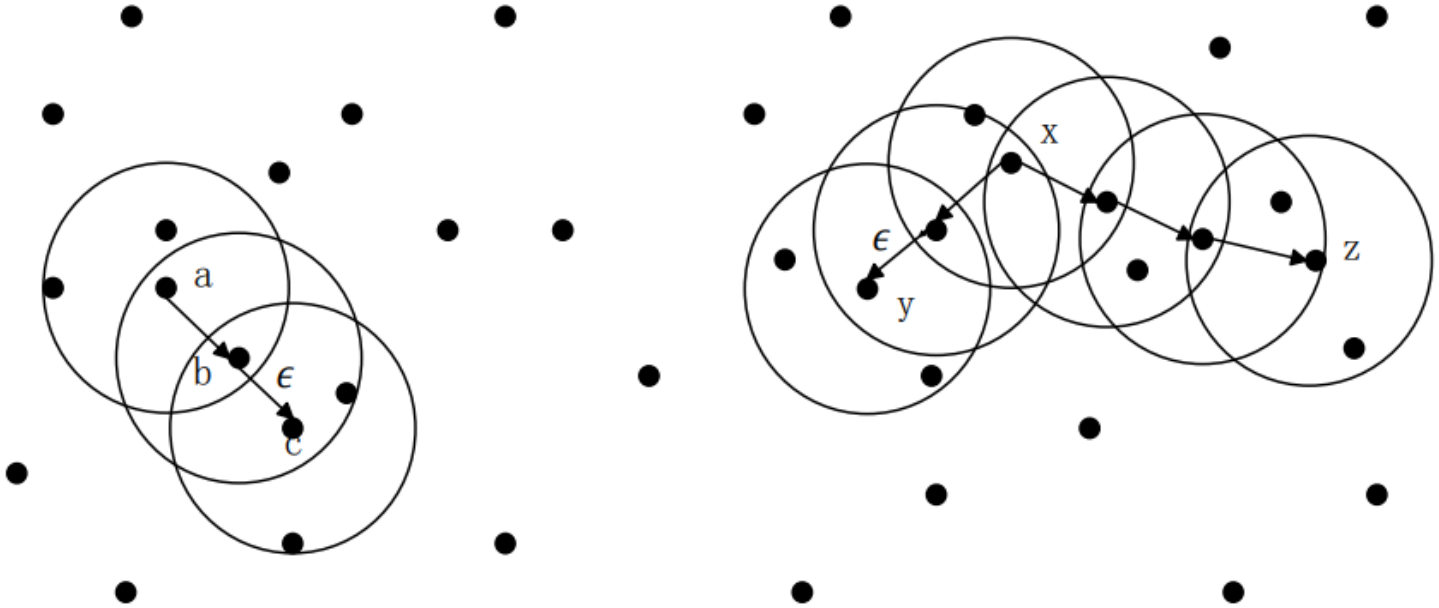
DBSCAN算法的基本概念

- 直接密度可达（directly density-reachable）：对于对象x_i和x_j，如果x_i是一个核心对象，且x_j在x_i的ε-邻域内，那么对象x_j是从x_i直接密度可达的。
- 密度可达（density-reachable）：对于对象x_i和x_j，若存在一个对象链$p_1, p_2, \ldots, p_n$，使得$p_1 = x_i, p_n = x_j$，并且对于$p_i(1 \leq i \leq n), p_{i+1}$从$p_i$关于$(\varepsilon, MinPts)$直接密度可达，那么$x_j$是从$x_i$密度可达的。
- 密度相连（density-connected）：对于对象x_i和x_j，若存在x_k使得x_i和x_j是从x_k关于$(\varepsilon, MinPts)$密度可达，那么x_i和x_j是密度相连的。

- 密度可达的核心点能够连通，它们构成的以r为半径的圆形邻域相互连接或重叠，这些**连通的核心点及其所处的邻域内的全部点构成一个簇**

DBSCAN算法的基本概念图示

- 在下图中，若$MinPts = 3$，则a、b、c和x、y、z都是核心对象，因为在各自的ε-邻域中，都至少包含3个对象。对象c是从对象b直接密度可达的，对象b是从对象a直接密度可达的，则对象c是从对象a密度可达的。对象y是从对象x密度可达的，对象z是从对象x密度可达的，则对象y和z是密度相连的



DBSCAN算法实现

- DBSCAN算法根据**密度可达关系**求出所有密度相连样本的**最大集合**，将这些样本点作为同一个簇：
  1. 将所有点标记为核心点、边界点或噪声点；
  2. 删除噪声点；
  3. 每组连通的核心点形成一个簇；
  4. 将每个边界点指派到一个与之关联的核心点的簇中（某个核心点的半径范围之内）

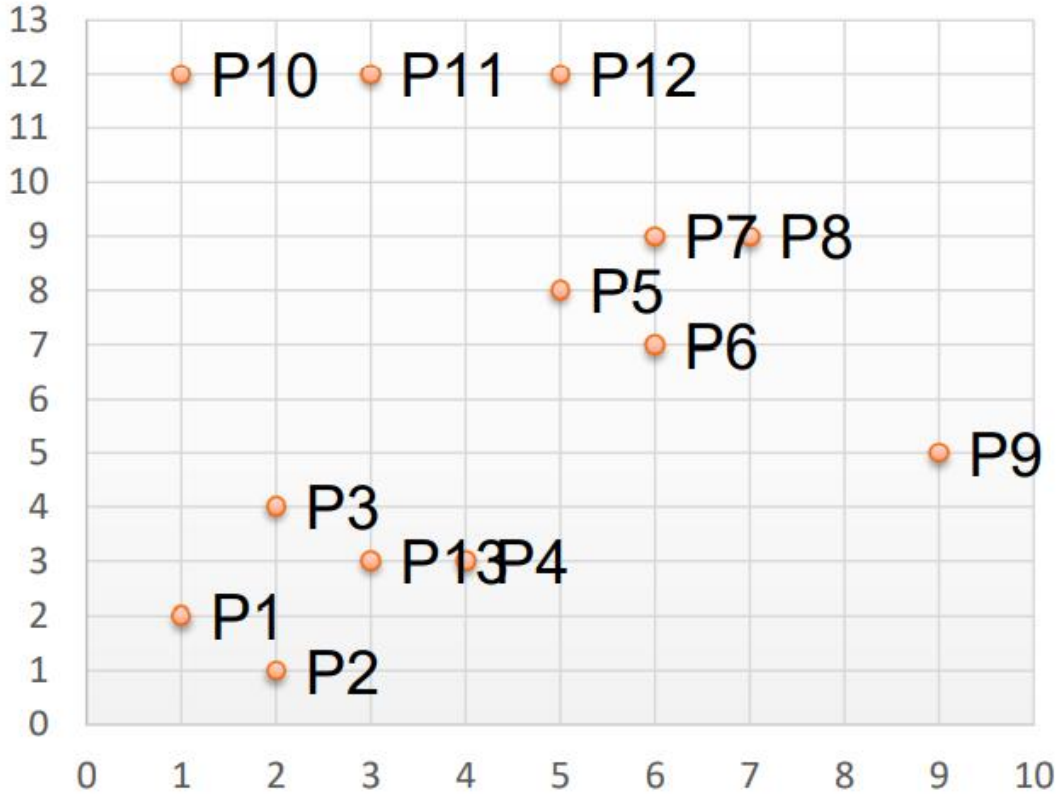
  - 优点：**不需要指定簇的个数**，可以发现任意形状的簇。
  - 缺点：对于高维度的数据，聚类效果不好

DBSCAN聚类示例

- 有如下13个样本点，
  使用DBSCAN进行聚类

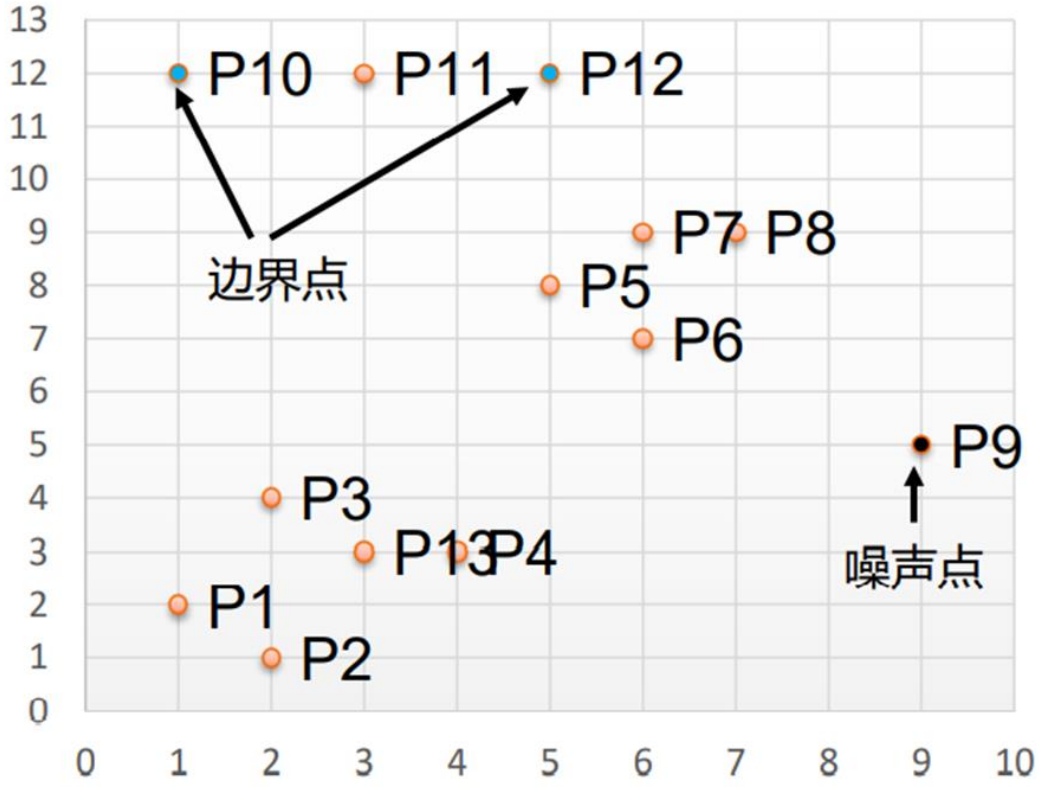
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13
X	1	2	2	4	5	6	6	7	9	1	3	5	3
Y	2	1	4	3	8	7	9	9	5	12	12	12	3

- 取Eps=3，MinPts=3
  依据曼哈顿距离对所有点进行聚类



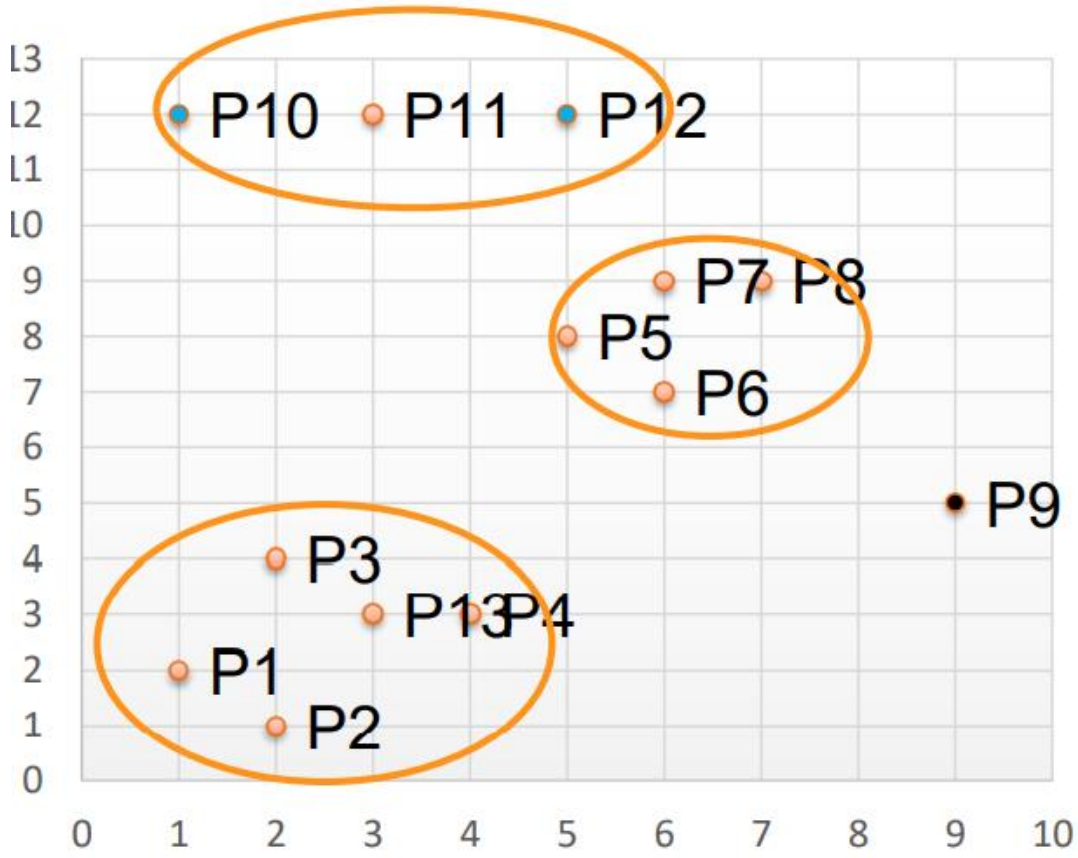
DBSCAN聚类示例

- 对每个点计算其邻域Eps=3内的点的集合。
- 集合内点的个数超过 MinPts=3的点为核心点
- 查看剩余点是否在核心点的邻域内：若在，则为边界点，否则为噪声点。



DBSCAN聚类示例

- 将距离不超过Eps=3的点相互连接，构成一个簇，核心点邻域内的点也会被加入到这个簇中。
- 则右侧形成3个簇。



可视化演示

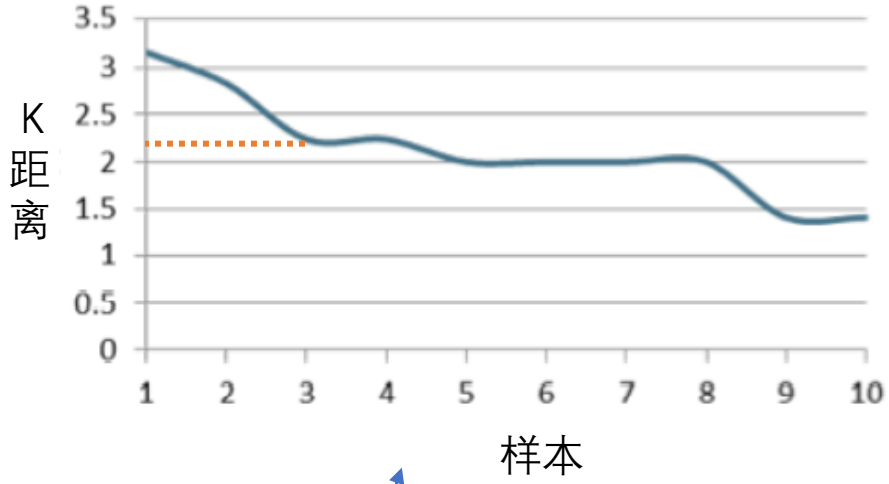
- DBSCAN
- https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

- K-means
- https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

DBSCAN算法评价

- DBSCAN可以用于对任意形状的稠密数据集进行聚类，它能够在聚类的过程中发现数据集中的噪声点，且算法本身**对噪声不敏感**。当数据集分布为**非球型**时，使用DBSCAN算法效果较好。
- DBSCAN算法的聚类结果**受到邻域参数{eps, Minpts}很大的影响**，邻域参数需要人工输入，调参时需要对两个参数联合调参，比较复杂。OPTICS算法对DBSCAN算法进行了改进，降低了对输入参数的敏感程度。
- DBSCAN算法要对数据集中的每个对象进行邻域检查，当数据集较大时，聚类收敛时间长，需要较大的内存支持，I/O 消耗也很大，此时可以采用KD树或球树对算法进行改进，快速搜索最近邻，帮助算法快速收敛。

DBSCAN算法的参数



- ε和minPts的设定可借助**k-距离**
  - k-距离：给定数据集D={p(i); i=0,1,…n}
  - 对于任意点P(i)，计算点P(i)到集合D的子集S={p(1), p(2), …, p(i-1), p(i+1), …, p(n)}中所有点之间的距离
  - 距离按照**从小到大**的顺序排序，假设排序后的距离集合为D={d(1), d(2), …, d(k-1), d(k), d(k+1), …,d(n)}，则d(k)就被称为k-距离。
  - **k-距离是点p(i)到所有点（除p(i)点外）第k近的距离**。
  - 对聚类集合中每个点p(i)都计算k-距离，最后得到所有点的k-距离集合E={e(1), e(2), …, e(n)}。
  - 对集合E进行顺序排序后得到，绘制k-距离的变化曲线图，将拐点位置所对应的k-距离的值，确定为半径ε的值。
  - minPts = k（二维数据k=4）

基于网格的方法

- 采用一个多分辨率的网格数据结构，将空间量化为有限数目的单元，这些单元形成了网络结构，所有的聚类操作都在网格上进行。

- 特点：直接聚类对象是空间而非数据对象
- 优点：处理速度快

基于模型的方法

- 试图优化给定的数据和某些数学模型之间的拟合，即假设数据是根据潜在的概率分布生成的;
- 基于模型的方法试图找到背后的模型，并使用概率分布特性进行聚类。

- GMM，Kohonen神经网络

聚类

- K-means：
  - 核心参数k：轮廓系数/肘部法则
  - 距离/相似性
- 层次聚类
- Dbscan：
  - 核心参数{eps, MinPts}：k-距离
  - 密度可达关系

机器学习常见问题

- 数据质量问题与预处理
- 机器学习常见陷阱
- 机器学习方法的选择
- 机器学习结果的评价

数据质量问题与预处理

- 数据质量要求数据是完整的和真实的，并且具有一致性和可靠性
- **数据预处理占用整个机器学习项目60%的工作量**
- 问题
  - 数据量过少
  - 数据量过多
  - 维度灾难
  - 数据不完整
  - 异常数据
  - 重复数据
  - 数据不一致

数据量过少

- 数据样本需要有足够的覆盖范围，需要覆盖与分析目标相关的维度
- 数据量增多，其中的规律会越发明显，也更易发现与分析目标相关的因素
  - 神经网络
  - 深度学习
- 一般来说，数据量是自变量数量的10~20倍为佳

数据量过多

- 数据量过多时，对全部数据集进行分析要耗费更多的计算资源，要求硬件配置较高，可应用数据采样技术随机提取样本子集。
- 对海量的同质化数据，可通过聚集技术按照时间、空间等属性进行均值等汇总，减少数据数量。
- 数据集不平衡问题可能导致出现较大的结果误差，因此要对数据集应用采样技术或对异常数据进行复制，提高其占比。

维度灾难

- 当数据中的**自变量过多**时，会出现维度灾难问题。
- 特别是在矩阵数据中，当冗余变量占比比较高时，可用数据会变成稀疏矩阵，在分类算法处理时就无法可靠地进行类别划分，在聚类算法中则容易使聚类质量下降。
- 可采用线性代数的相关方法将数据从高维空间影射到低维空间
  - 主成分分析（PCA）
  - 奇异值分解（SVD）

数据不完整

- 数据的种类要多，种类多少直接影响数据挖掘方法的选择，可以通过编写程序抓取外部数据作为补充。
  - **数据缺失**也是数据不完整的一种表现，包括了空白值、空值、无效值。
  - 需要针对不同原因对缺失值进行数据预处理，有多种方法可以操作
    - 采用众数、中位数、均值、最短距离等方法进行人为补充
    - 通过回归或贝叶斯定理等预测缺失值
    - 删除含有缺失值的数据

机器学习方法的选择

- 理解目标要求是机器学习方法选择的关键，首先要对问题进行分类，如果数据集中有标签则进行**有监督学习**，反之则进行**无监督学习**
- 熟悉各类机器学习方法的特性是分析方法选择的基础，不仅需要了解如何**使用**各类分析算法，还要了解其实现的**原理**
  - 在选择模型前，要对数据进行描述性统计
  - 可在几个可能模型中分析选出较优的模型
  - 选择模型后，比较不同模型的泛化能力，反复调整参数使模型结果趋于稳定

Come back to the task of PartI

- 请思考：
  - 如何处理原始数据中的缺失值？
    - Spliting first or filling first?
  - 如何确定特征？
    - 你能自己构造额外的特征吗？
    - 如何理解朴素贝叶斯分类器？
  - 面对这样一个问题，如何选择合适的机器学习方法？