INT104W10_非监督学习-聚类

非监督学习，即输入（学习）的样本中不包含标签（label），在没有类别信息（标签）的情况下，通过对所研究对象样本的数据分析，实现对样本的分类或发现数据中的潜在结构和模式。

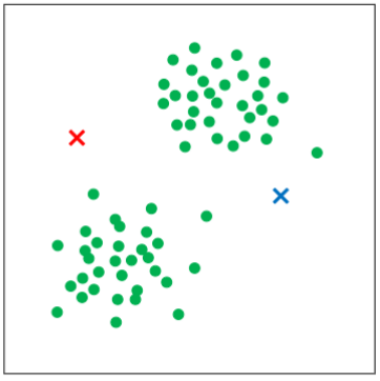
K-means聚类

一种无监督聚类算法，将一组数据通过聚类得到k个分组（其中k为超参数，需要手动设定分为多少类）。

聚类步骤

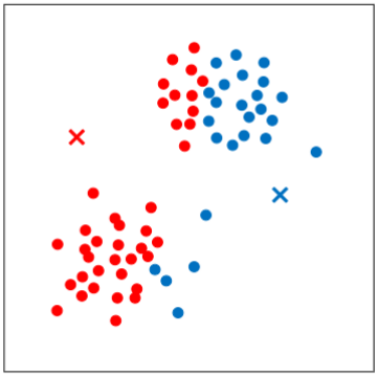
假设我们需要将原数据集分为两类，即设定k=2

步骤1：在数据集中随机选取k个**聚类中心点（红色和蓝色×）**

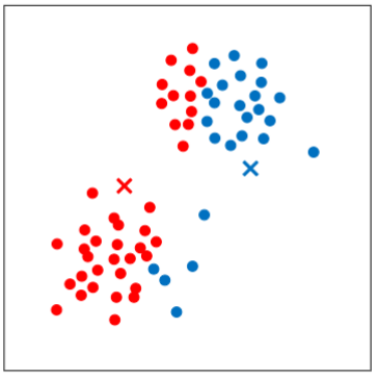


步骤2：分别计算每个样本点到k个**聚类中心点**的距离，根据距离对该数据点进行分类。

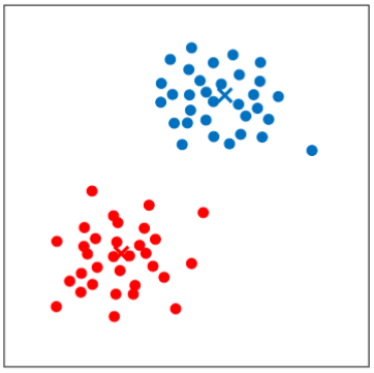
*注：这里的距离可以是欧式距离（$\sqrt{\sum_{j=1}^{k}(a_j-b_j)^2}$每个对应特征的差的平方和的开根），也可以是曼哈顿距离（$\sum_{j=1}^{k}|a_j-b_j|$每个对应特征的差的模的和）



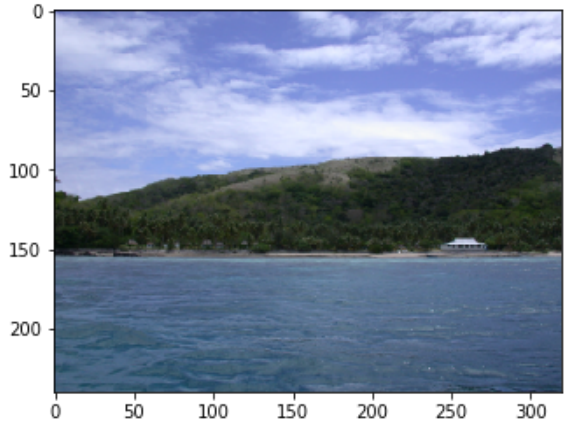
步骤3.计算同类数据点的重心作为待更新的该类**聚类中心点**位置。

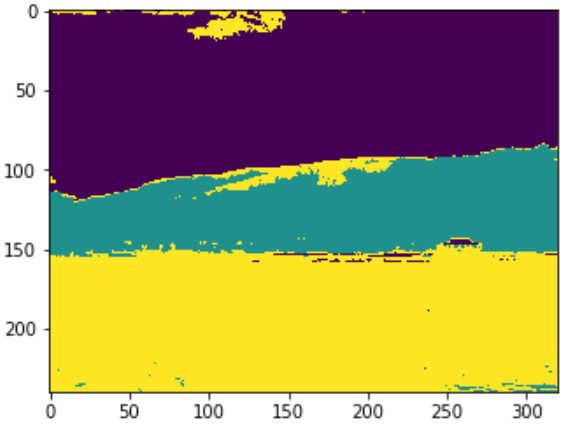


步骤4：更新**聚类中心点**，重复步骤2-4，若每个数据点与其所属类中心点的距离之和不再变化，则算法结束。



在实际应用中，除了可以对数据聚类，还可以对图像中的颜色与位置聚类





聚类原理

对希望分入k个类中的数据集$X = \{x_1, x_2, ..., x_i, ..., x_m\}$ (1)

随机初始化$\{\mu_1, \mu_2, ..., \mu_j, ..., \mu_k\}$个聚类中心 (2)

每个点x_i到对应聚类中心μ_j的距离$c_{j,i} = arg\ min_j||x_i - \mu_j||^2$ (3)

(3)用欧氏距离将x_i分配给最近的μ_j，得到k个类簇$\{S_1, S_2, ..., S_j, ..., S_k\}$ (4)

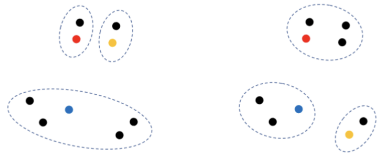
更新每个聚类中心$\mu_j = \frac{\sum_{x_i \in S_j} x_i}{|x_i|}$ (5)

重复(3)到(5)，直到μ_j不再变化（收敛） (6)

即最小化损失函数：$J(c, \mu) = \sum_{i=1}^{m} ||x_i - \mu_{c_i}||^2$（保证收敛）

改进方法（K-means++）

尽管对聚类中心的位置优化是自动的，但是初始聚类中心的不同会导致最终聚类结果的不同。如下可见，由于聚类中心的选择更加分散，右图聚类效果好于左图。



初始聚类中心的不同会导致最终聚类结果的不同

K-means++方法（Arthur & Vassilvitskii, 2007）为选择更优的初始聚类中心做了优化（Better initialization）：

1. 从样本点中随机选择一个作为聚类中心
2. 对每个样本点x_i计算该样本点到距他最近的聚类中心的距离$D(x_i)$
3. 随机选择一个新样本点x_j作为聚类中心，样本点被选中的概率与$D(x_j)$成正比（距离越远，被选中的概率越大）
4. 重复步骤2~3，直到选出了k个聚类中心。

参数调整与效果评估

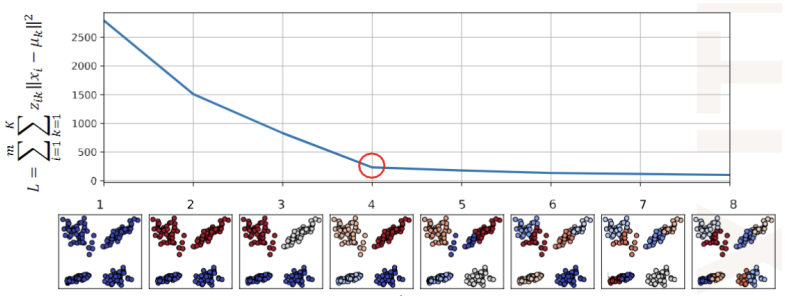
- 机器学习算法（十二）：聚类

  https://blog.csdn.net/weixin_39910711/article/details/123973374

在不知道聚类簇数的情况下，如何选择超参数K的大小？尤其在面对没有观测数据的分组信息时，在面对数据维度很高样本极多时，我们**需要通过算法模型来寻求数据内在的结构和模式。**

肘方法(Elbow method)

逐渐增加**聚类数K**并进行聚类，计算**损失函数**（每个样本到对应聚类中心的距离之和），损失函数的**拐点处即为推荐的聚类数**（即聚类数超过该点后，聚类数的增加不会解释样本中的更多方差，即不会对损失函数的下降带来很大的影响，所以会选择拐点）



肘部在K=4

轮廓系数（Silhouette Coefficient）

一个好的聚类算法应当确保得到的聚类：

- 每个簇中的样本相似度高（内聚）
- 不同簇间的样本相似度低（分离）

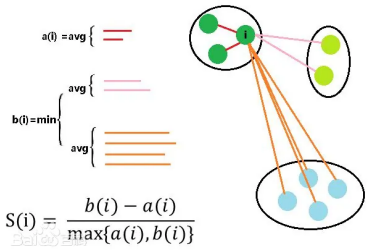
轮廓系数同时评判了一个聚类的内聚度（**cohesion**）和分离度（**separation**）：

$$\text{对每个样本点}x_i\text{的轮廓系数：}\ s(x_i)=\frac{b(x_i)-a(x_i)}{max\{b(x_i),a(x_i)\}} \tag{7}$$

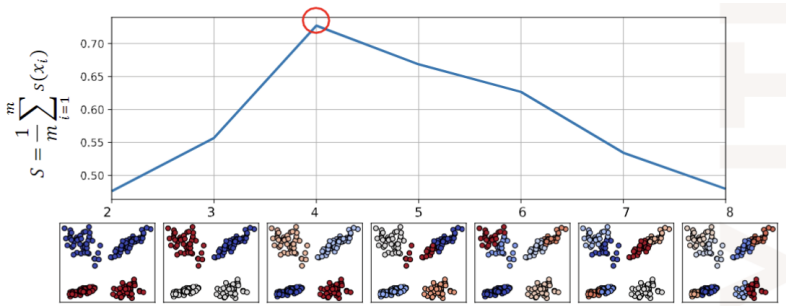
$$a(x_i)=x_i\text{到同簇中其他点的平均距离（棕线）} \tag{8}$$

$$b(x_i)=x_i\text{到}\textbf{最临近簇}\text{中的所有点的平均距离（粉线）} \tag{9}$$

$$\text{整体的轮廓系数：}\ S=\frac{1}{m}\sum_{i=1}^{m}s(x_i) \tag{10}$$



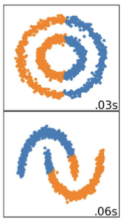
可见，$a(x_i)$越小，内聚程度越高；$b(x_i)$越大，分离程度越高。**轮廓系数的值介于 [-1,1]，越趋近于1代表内聚度和分离度都相对较优。**



K=4时轮廓系数最大

优缺点

- 优点：时间复杂度和空间复杂度都较低，对于大型数据集也较为简单高效。
- 缺点：需要手动设置超参数K；对噪声和离群值非常敏感；当数据集较大时，K-Means算法容易陷入局部最优；对初始值的设置很敏感。
- 聚类特性：在圆形或凸形状上的聚类表现较好，但对凹形的聚类簇表现较差（因为使用点到点间的距离而不是密度作为聚类判断条件）。



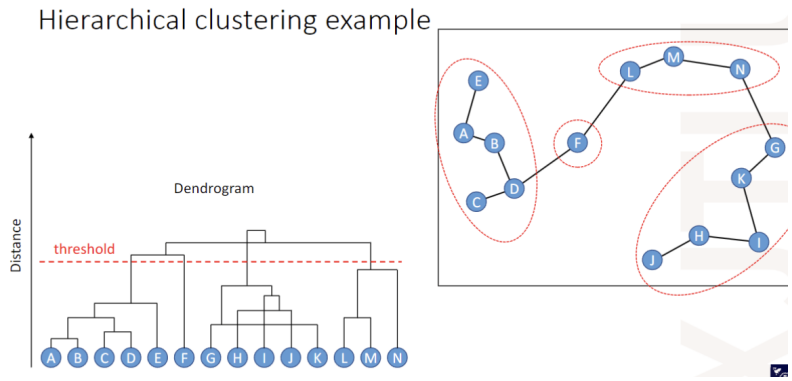
层次聚类（Hierarchical clustering）

可以认为，较大的簇由更小的簇组成。层次聚类基于簇间的相似度在不同层次上分析数据，形成树状的聚类结构。有**自底向上的聚合(Agglomerative)**和**自顶向下的分拆(Divisive)**两种策略，前者更常见。

聚合(Agglomerative)过程

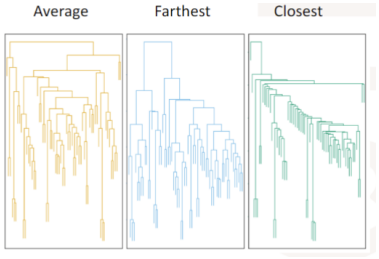
idea：我们希望临近（距离短）的数据点最终/尽早归于同一个簇中

1. 初始化集合C，仅包含**由单个样本点构成的簇**c：$C=\{c_1,c_2,...,c_i,...,c_m\}$，其中$c_i=\{x_i\}$
2. 查找集合C中**距离**最近的一对簇$arg\ min_{i,j}D(c_i,c_j)$，将这对簇合并为新的簇$c_{i\&j}$加入集合并移除原先的簇。
3. 重复2直到集合C只剩下一个簇，生成聚类谱系图（Dendrogram（hierarchical tree of clusters））。



簇间相似度的计算方法（**对距离的不同定义**）：

- 最小距离：两个簇的最近样本决定，又称为单链接算法。
- 最大距离：两个簇的最远样本决定，又称为全链接算法。
- 平均距离：两个簇的所有样本对距离平均值决定，又称为均链接算法。
- 中心距离：两个簇的中心间的距离决定。
- 最小方差/离差平方和（ward）：两个簇的所有样本对的距离平方和的平均决定。



对距离的不同定义会导致不同的聚类结果

例题

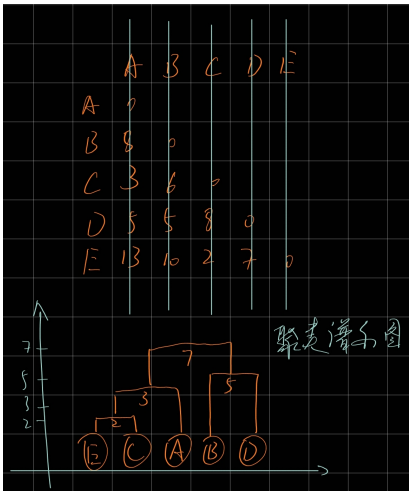
下表给定了样本之间的曼哈顿距离，请使用单链接的聚合层次聚类绘制聚类谱系图。

Given the following table that shows the distance between samples ("city block distance"), using agglomerative clustering method with ***single linkage***, draw the final dendrogram obtained.

city block distance :

A (3,5) B(2,7) -> D(A,B) = |3-2|+|5-7|=3

	A	B	C	D	E
A	0				
B	8	0			
C	3	6	0		
D	5	5	8	0	
E	13	10	2	7	0



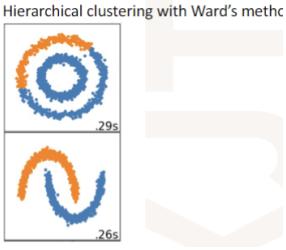
优缺点

优点：

- 层次结构相比平面结构（K-means）可以提供更多信息，从而更容易确定分簇数量（但也得不到聚类中心）
- 无需预先指定聚类簇数
- 结果可通过谱系图可视化
- 可以处理非凸数据集

缺点：

- 计算速度慢，时间复杂度$O(n^3)$，空间复杂度$O(n^2)$，不适合大数据集。
- 由于层次聚类尝试连接所有数据点，故对异常值和噪声敏感。
- 可能聚成链状（聚类失败）



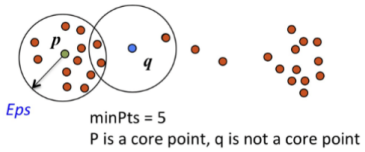
- 层次聚类 Hierarchical Clustering

  https://blog.csdn.net/JasonH2021/article/details/131017486

基于密度的聚类方法-DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise)

聚类原理



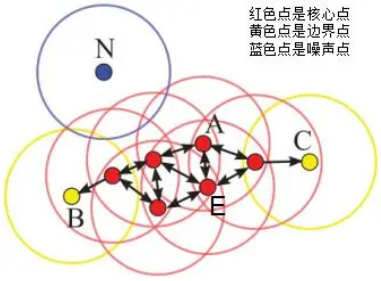
密度=单位空间中的样本点数量。故共有两个超参数：扫描半径（Eps）和最小包含点数（minPts）；扫描半径内代表了单位空间，最小包含点数代表了样本点数量，两者结合成为了本算法中密度的度量。

- **扫描半径 (eps)**：用于定位点/检查任何点附近密度的距离度量
- **最小包含点数(minPts)**：聚集在一起的最小点数(阈值)，该区域被认为是稠密的

以此可将所有样本点分为三类：

- **核心点**（core point）：在半径Eps内含有超过MinPts数目的点。
- **边界点/密度可达点**（density-reachable point）：在半径Eps内点的数量小于MinPts。但是落在核心点的邻域内的点。
- **噪音点**（outliers/noise point）：既不是核心点也不是边界点的点。

下面以minPts=4为例：



- 密度直达（directly density-reachable）：若q处于p的ε邻域内，且p为核心点，则称q由p密度直达；
- 密度可达（density-reachable）：若q处于p的ε邻域内，且p，q均为核心点，则称q的邻域点由p密度可达；
- 密度相连（density-connected）：若p，q均为非核心点，且p，q处于同一个簇类中，则称q与p密度相连。

原理上：只要任意两个样本点是密度直达或密度可达的关系，那么该两个样本点归为同一簇类，上图的样本点ABCE为同一簇类。因此，DBSCAN算法从数据集中随机选择一个核心点作为“种子”，由该种子出发确定相应的聚类簇，当遍历完所有核心点时，算法结束。

*若使用曼哈顿（manhattan）距离，则邻域性状为矩形；若使用欧式距离，则邻域形状为圆形。

1. 将所有点标记为核心点、边界点或噪声点
2. 如果选择的点是核心点，则找出所有从该点出发的密度可达点形成簇
3. 如果该点是非核心点，将其指派到一个与之关联的核心点的簇中
4. 重复以上步骤，直到所点都被处理过

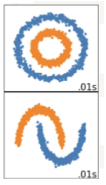
优缺点

优点：

- 相当快，优化后的时间复杂度可达到$O(n\ log(n))$
- 由于基于密度，可以找到任意形状的簇
- 对异常值（噪声）有鲁棒性

缺点：

- 如果不同数据区域的数据点密度不同，可能无法正常工作；对不稠密的数据也不推荐使用。
- 选择合适的距离阈值ε可能会困难



DBSCAN聚类