**<https://blog.csdn.net/u011511601/article/details/81951939>**

**无监督学习（Unsupervised learning）**：训练样本的标记信息是未知的，目标是为了揭露训练样本的内在属性，结构和信息，为进一步的数据挖掘提供基础。

· 聚类（clustering）

· 降维（dimensionality reduction）

· 异常检测（outlier detection）

**监督学习（supervised learning）**：训练样本带有信息标记，利用已有的训练样本信息学习数据的规律预测未知的新样本标签

· 回归分析（regression）

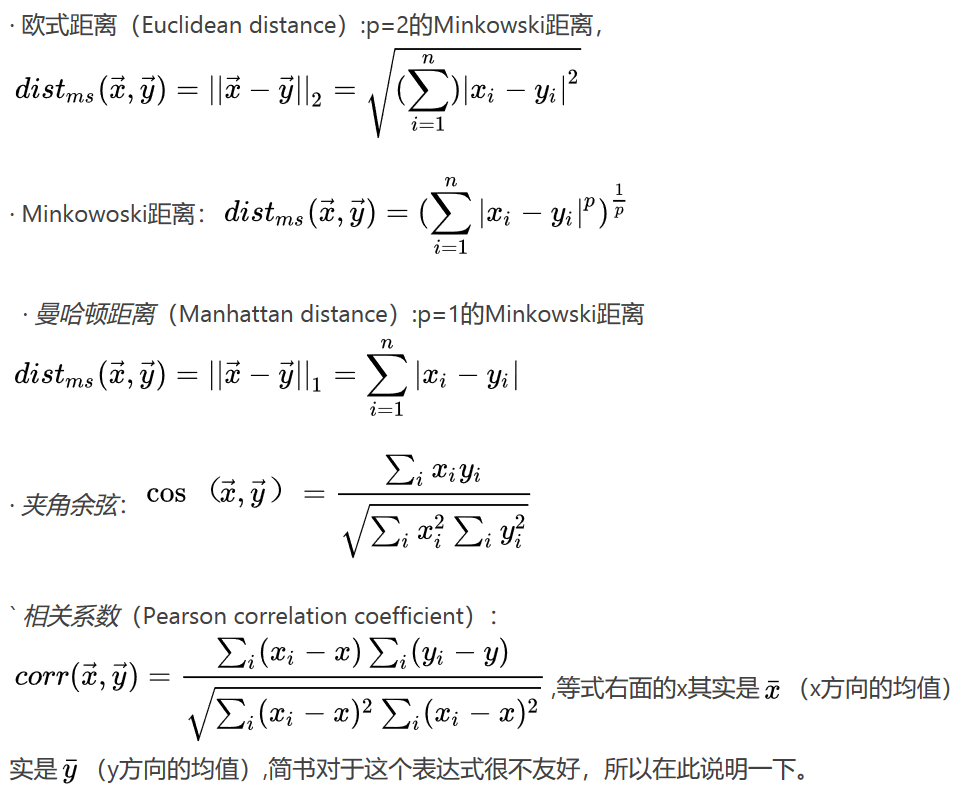
· 分类（classification）

**聚类**：物以类聚。按照某一个特定的标准（比如距离），把一个数据集分割成不同的类或簇，使得同一个簇内的数据对象的相似性尽可能大，同时不再同一个簇内的数据对象的差异性也尽可能的大。

**簇**（或类cluster）：子集合。最大化簇内的相似性；最小化簇与簇之间的相似性。

聚类可以作为一个单独过程，用于寻找数据内在分布结构，也可以作为其他学习任务前驱过程，可以作为一个数据探索性分析的有力工具。

**聚类相似度度量：**几何距离



**聚类类别：**

· 基于划分的聚类（partitioning based clustering）：

k均值（K-means）, Mean shift

· 层次聚类（hierarchical clustering）：

Agglomerative clustering, BIRCH

· 密度聚类（density based clustering）：

DBSCAN

· 基于模型的聚类（model based clustering）：

高斯混合模型（GMM）

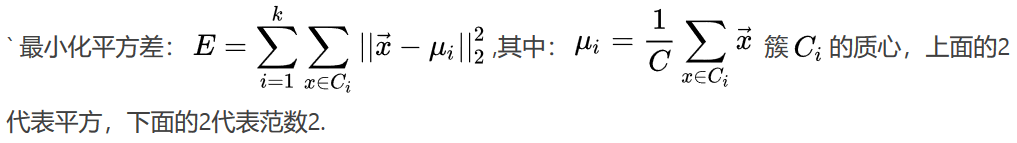
· Affinity propagation

· Spectral clustering（谱聚类）

**K均值算法（Kmeans）:**

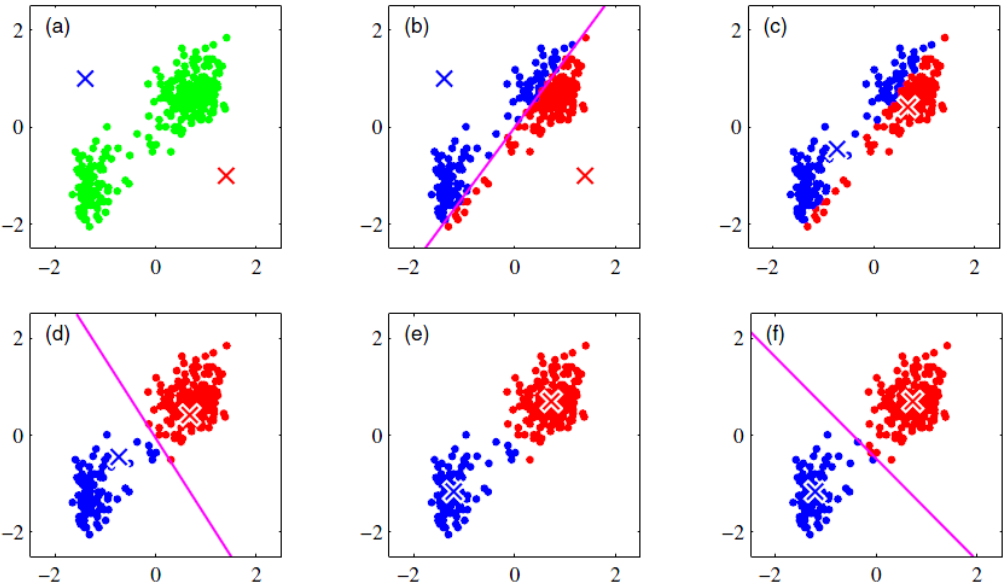
**K代表需要指定的K个簇，K个质心，means代表算法使用距离均值作为评价标准来确定质心和划分簇，均值意味着K-means算法可能会受到一些异常点的干扰**





算法过程：

1. 指定k个簇，随机从样本中选取k个质心位置
2. 计算每一个样本到这k个质心的距离，每个样本获得了k个距离值，选距离最小的质心，加入这个簇，k个簇出现，每个样本唯一属于某一个簇，完成第一次划分。
3. 重新确定k个簇的质心，利用最小化平方差确定新的质心。即均值
4. 重复2，每个簇的样本再一次发生变化
5. 重复3，簇的质心发生变化
6. 一直节点直到质心不再发生变化，即目标函数收敛了，基本不再发生变化



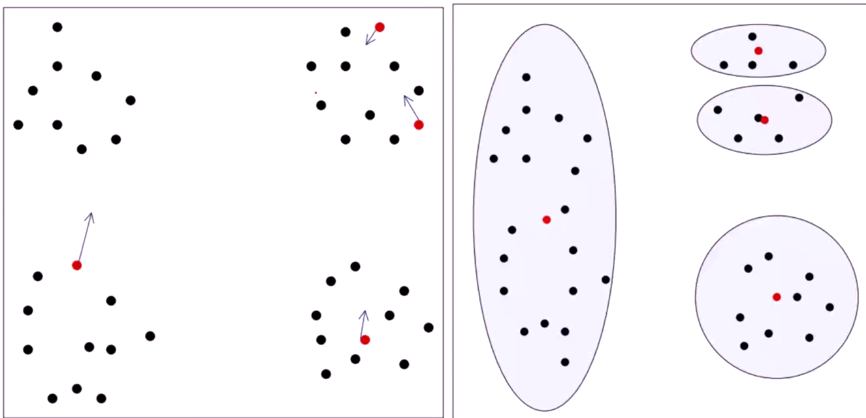
**K均值算法（Kmean）的优缺点**：

优点：1. 简单直观，易于理解实现；2. 复杂度相对比较低，在K不是很大的情况下，Kmeans的计算时间相对很短；3. Kmean会产生紧密度比较高的簇，反映了簇内样本围绕质心的紧密程度的一种算法。

缺点：

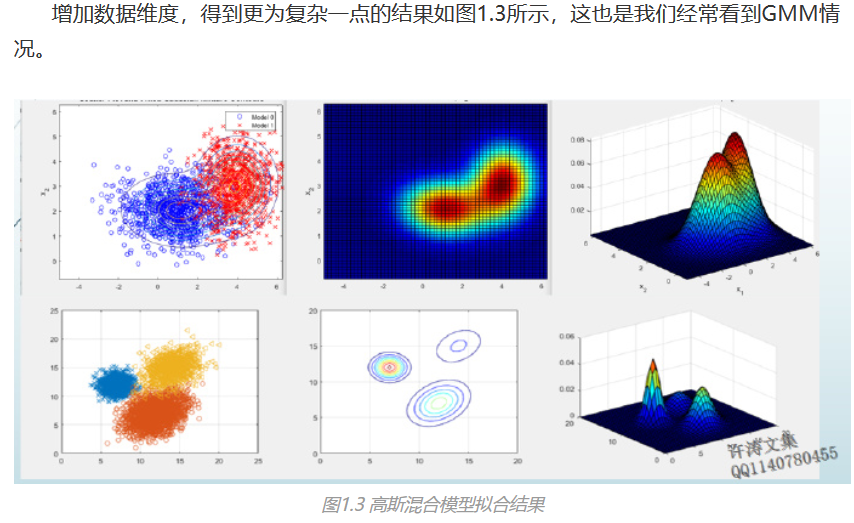
1. 你必须指定簇的数目很难预测到准确的簇的数目；

2. 对初始值设置很敏感（K-means++）



如果初始4个点如上图选择，那么最终的聚类结果是不好的，那理想中应该这k个点应该尽可能离得远一点，K-means++算法在初始选择k个点时，依次选的第i个质心都尽可能的远离了上一个质心i-1，就是离上一个质心远的点被选中的概率较大。

3. Kmeans主要发现圆形或者球形簇，对不同形状和密度的簇效果不好，就是说对于那些样本特征分布比较简单，效果还不错，对于一些样本点交织在一起，比如环形的曲线型的分布，无法发现复杂的簇。归根结底就是说，K-means算法适合那些分布服从混合高斯分布的数据，在这样的数据上效果比较好，类似环数据本身很难用混合高斯分布来拟合它，K-means算法在这样的数据上效果不太好。

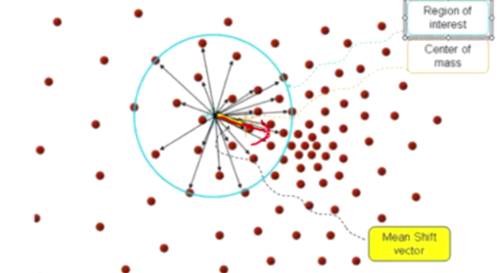
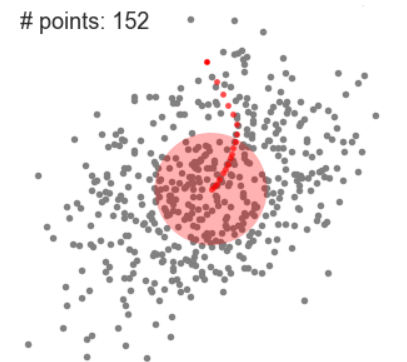


K-means算法主要有两个动作，一个是划分，一个是确定质心，确定质心的过程是通过找到簇内质点的最小距离平方差，只有簇的点分布更接近高斯分布，这样找到的簇才更加紧密，更加符合一个簇的概念。

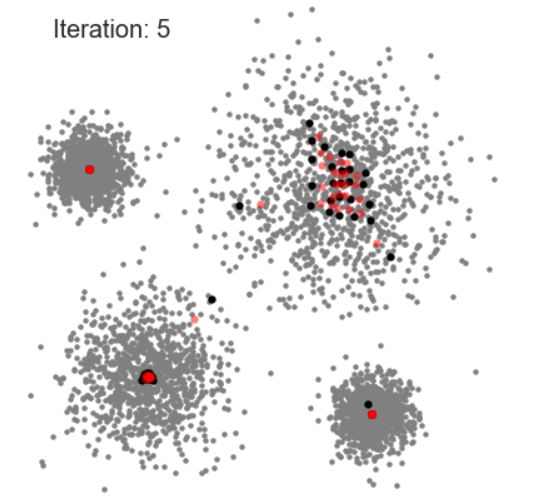
4. Kmeans对噪声和离群值非常敏感（Kmeadians对噪声和离群值不敏感）

均值偏移（Mean shift）聚类算法

是一种基于滑动窗口（sliding-window）的算法，它试图找到密集的数据点。而且，它还是一种基于中心的算法，它的目标是定位每一组群/类的中心点，通过更新中心点的候选点来实现滑动窗口中的点的平均值。这些候选窗口在后期处理阶段被过滤，以消除几乎重复的部分，形成最后一组中心点及其对应的组。



1. 以点C（随机选择）为中心的圆形滑窗开始，以半径r为内核，画出c到任意一个圆圈内的点的向量，最终得出一个合力方向，作为中心移动的方向和距离，均值偏移是一种爬山算法（hill climbing algorithm），它需要在每个步骤中反复地将这个内核移动到一个更高的密度区域，直到收敛。在每一次迭代中，滑动窗口会移向密度较高的区域，将中心点移动到窗口内的点的平均值（因此得名）。滑动窗口中的密度与它内部的点的数量成比例。自然地，通过移向窗口中点的平均值，它将逐渐向更高的点密度方向移动。每一次滑动完成，都可以找到该簇的最大密度点，该点可作为质心。
2. 重复上述过程多次，可能找到多个局部密度极大点，



与K-Means聚类相比，均值偏移不需要选择聚类的数量，因为它会自动地发现这一点。这是一个巨大的优势。聚类中心收敛于最大密度点的事实也是非常可取的，因为它非常直观地理解并适合于一种自然数据驱动。缺点是选择窗口大小/半径r是非常关键的，所以不能疏忽。

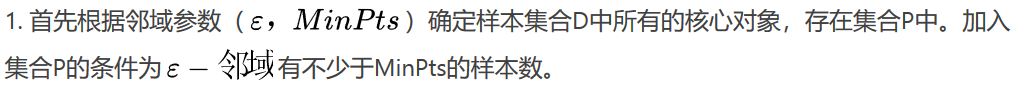
**DBSCAN聚类算法**

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)是一个比较有代表性的基于密度的聚类算法，类似于均值转移聚类算法，DBSCAN对于离群点的检测是非常显著的。

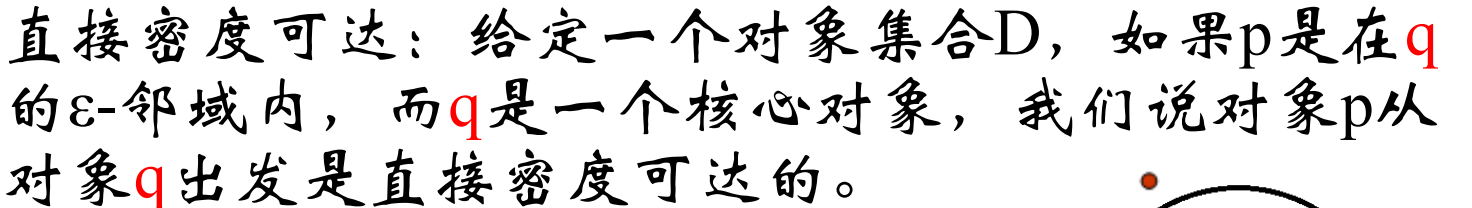
minPoints：阈值，需要指定的参数，代表该圆内样本点个数，当样本点个数大于minPoints，该圆将被定义为核心对象，小于minPoints定义为非核心对象

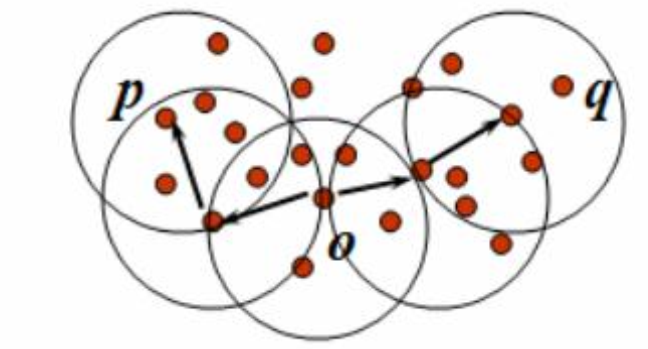
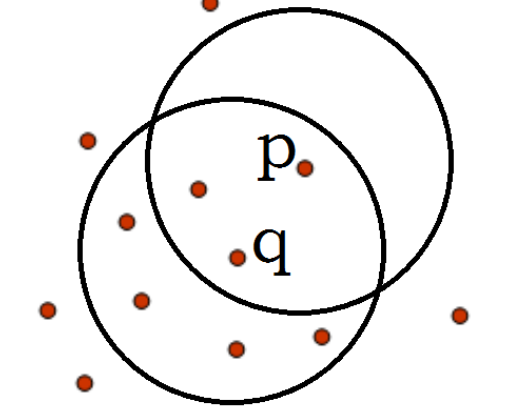
ε：领域半径，DBSCAN以一个从未访问过的任意起始数据点开始。这个点的邻域是用距离ε所有在ε距离的点都是邻点）来提取的。

DBSCAN算法的过程：

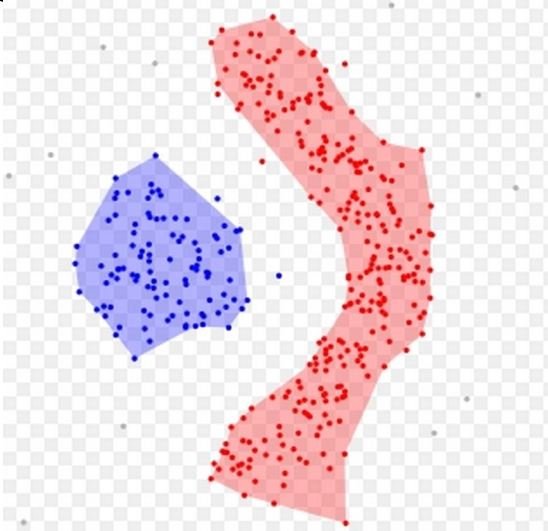


1. 然后从核心对象集合P中任意选取一个核心对象作为初始点，找出其密度可达的样本生成聚类簇，构成第一个聚类簇C1，所谓密度可达就是



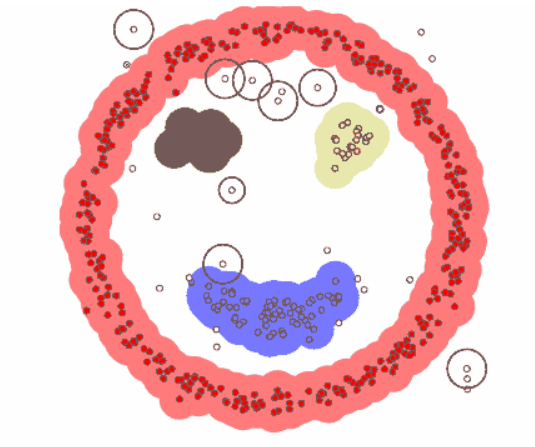


相当于按照这样一个密度和半径去深度优先搜索，找到这样一个密度可达链



1. 将C1内多个核心对象从P中去除，再从更新后的核心对象集合P中任意选取下一个种子样本。
2. 重复（2-3），直到核心对象被全部选择完，也就是P为空集。

一个笑脸的DBSCAN聚类结果



最终异常点或者异常群体会被孤立出来

DBSCAN比其他聚类算法有一些优势。

首先，它不需要一个预设定的聚类数量。它还将异常值识别为噪声，而不像均值偏移聚类算法，即使数据点非常不同，它也会将它们放入一个聚类中。

此外，它还能很好地找到任意大小和任意形状的聚类。

它可以给出核心对象，非核心对象以及异常点，三种评估结果

DBSCAN的主要缺点是，当聚类具有不同的密度时，它的性能不像其他聚类算法那样好。这是因为当密度变化时，距离阈值ε和识别邻近点的minPoints的设置会随着聚类的不同而变化。这种缺点也会出现在非常高维的数据中，因为距离阈值ε变得难以估计。