聚类算法

Li Junli 李军利
Apr 2 2019

物以类聚，人以群分，聚类是无监督学习的范畴。聚类就是按照特定的准则(比如距离)把原始数据集分成不同的类或簇，使得同一个簇内的数据对象的相似性尽可能大，并且不在同一个簇中的数据对象的差异性也尽可能地大，即聚类后同一类的数据尽可能聚集到一起，不同数据尽量分离。熟悉统计学的人可能很快会联想到方差分析中的组内与组间误差，误差来源主要是组间误差时，两个组来着不同的总体

聚类算法主要有——基于划分、层次、密度、网格、模型和模糊 6种方法，下面作介绍(Kmeans是重点)

1. 基于划分的聚类

思想:

n维空间中有一堆需要聚类的散点，首先要确定这堆散点最后聚成几类，然后挑选几个点作为初始中心点，再做迭代，重新计算中心点，直到"类内的点都足够近，类间的点都足够远"，即所谓的"启发式算法"。

特点：很适合发现中小规模数据中的球状簇。

算法：k-means及其变种 k-medoids、k-modes、k-medians、kernel k-means等

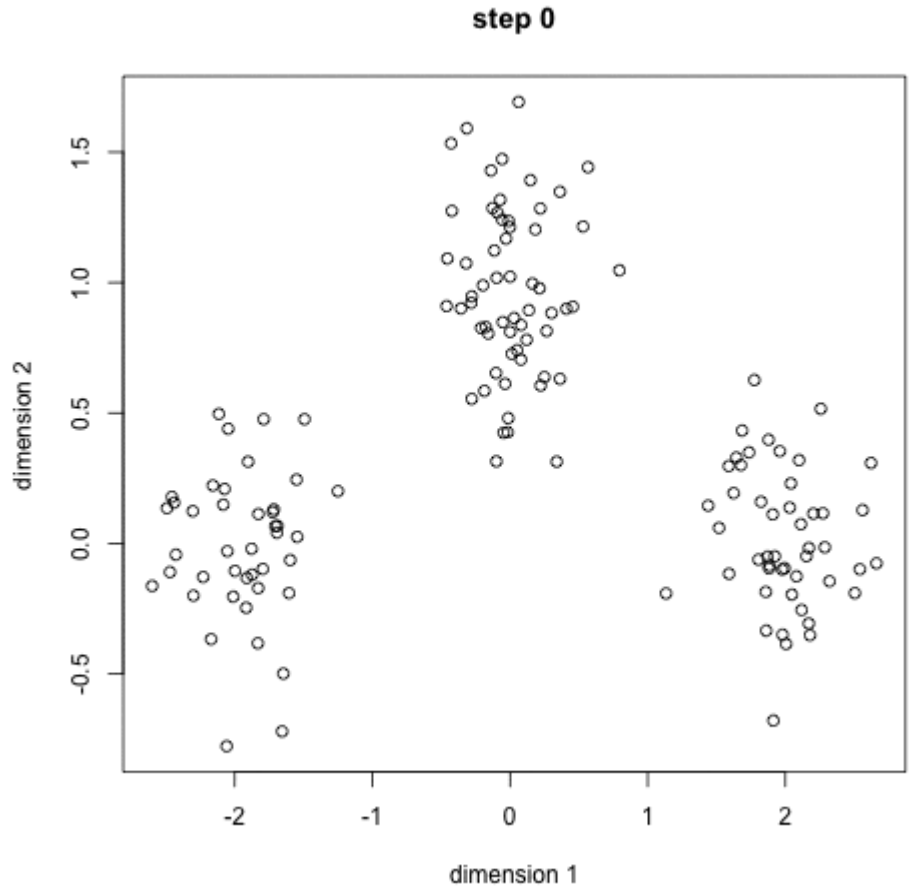
K-Means--经典算法

算法流程伪代码

k-means算法
输入：数据集D，聚簇数k
输出：聚簇中心集合C，聚簇标识向量m
/* 初始化 */
从数据集D中随机挑选k个点作为初始聚簇中心
将集合D中的每个数据点分配到距离它最近的聚簇中
repeat
/* 更新聚簇中心 */
更新C
/* 更新数据点的聚簇标识 */
将集合D中的每个数据点重新分配到距离它最近的聚簇中
更新m
until 目标函数收敛

时间复杂度O(nkt)，其中n是所有对象的数目，k是簇的数目，t是迭代的次数，通常k<<n

算法流程示意图



距离度量

通过计算样本之间的相似度(距离)，将相似度大的划分为一个类别，衡量样本相似度的方式有下面几种，其中K均值用的最多的距离度量方法是欧式距离，余弦距离等也会用到：

1. 闵可夫斯基距离/明氏距离/范式距离(Minkowski)：

$$X = (x_1, x_2 \ldots, x_n) \epsilon R^n,\ Y = (y_1, y_2 \ldots y_n) \epsilon R^n$$

样本点X与Y的明氏距离：$dis = (\sum_{i=1}^{100} |x_i - y_i|^p)^{1/p}$

p=1 曼哈顿距离；p=2 欧式距离；p无穷大 切比雪夫距离 此时dis等价于下面公式

$$dis = max(|x_1 - y_1|, |x_2 - y_2|, \ldots |x_n - y_n|)$$

一般情况下欧式距离用的比较多，当数据量出现扁平化时候，一般用切夫雪比距离

明氏距离比较直观，但是它与数据的分布无关，具有一定的局限性，如果某一维度方向，比如(x1, y1)的幅值远远大于其他维度方向(xi, yi)的值，这个距离公式就会过度放大(x1, y1)的作用。所以，在计算距离之前，我们可能还需要对数据进行 z-transform 处理，即减去该维度上的均值，除以该维度上的标准差：

$$(x_1, y_1) = (\frac{x_1 - u_x}{\sigma}, \frac{y_1 - u_y}{\sigma})$$

可以看到，上述处理开始体现数据的统计特性了。这种方法在假设数据各个维度不相关的情况下利用数据分布的特性计算出不同的距离。如果维度相互之间数据相关（例如：身高较高的信息很有可能会带来体重较重的信息，因为两者是有关联的），这时候就要用到马氏距离（Mahalanobis distance）了

2. 夹角余弦相似度：

$$对于向量a,c：cos(\theta) = \frac{\vec{a} \cdot \vec{c}}{|\vec{a}||\vec{c}|}$$

余弦相似度最常见的应用就是计算文本相似度。将两个文本根据他们词，建立两个向量，计算这两个向量的余弦值，就可以知道两个文本在统计学方法中他们的相似度情况。实践证明，这是一个非常有效的方法。

余弦距离和欧式距离一般是不等价的。比如说，夹角一样的两条边，边的距离是不一样的。但是两个向量的模长=1时，其欧式距离和余弦距离是等价的。如果向量模长归一化为1。那么将该训练数据丢到k-means里面去，使用欧式距离，也等价于使用了余弦距离。推导如下：

Cosine similarity is related to Euclidean distance as follows. Denote Euclidean distance by the usual $\|A - B\|$, and observe that

$$\|A - B\|^2 = (A - B)^\top (A - B) = \|A\|^2 + \|B\|^2 - 2A^\top B$$

by expansion. When A and B are normalized to unit length, $\|A\|^2 = \|B\|^2 = 1$ so the previous is equal to

$$2(1 - \cos(A, B))$$

Null distribution: For data which can be negative as well as positive, the null distribution for cosine similarity is the distribution of the dot product of two independent random unit vectors. This distribution has a mean of zero and a variance of $1/n$ (where n is the number of dimensions), and although the distribution is bounded between -1 and +1, as n grows large the distribution is increasingly well-approximated by the normal distribution.[5][6] For other types of data, such as bitstreams (taking values of 0 or 1 only), the null distribution will take a different form, and may have a nonzero mean.[7]

3. 杰卡德相似系数与距离（Jaccard）：

Jaccard（杰卡德）相似性系数主要用于计算符号度量或布尔值度量的样本间的相似度，适用于样本只有（0,1）的情况，又叫二元相似性。无法衡量差异具体值的大小，只能获得"是否相同"这样一种结果。等于样本集交集个数和样本集并集个数的比值，公式如下：

$$X = [0, 1, 1, 0, 1, \ldots, 1] \epsilon R^n, \ Y = [1, 1, 0, 1, 0, \ldots, 1] \epsilon R^n$$

$$J(X, Y) = \frac{X \cap Y}{X \cup Y}$$

与Jaccard系数相反的Jaccard距离：用两个集合中不同元素所占比例来衡量两个集合（样本）的区分度:

$$dis(X, Y) = 1 - J(X, Y) = 1 - \frac{X \cap Y}{X \cup Y}$$

将杰卡德相似性度量应用到基于物品的协同过滤系统中，并建立起相应的评价分析方法。 与传统相似性度量方法相比，杰卡德方法完善了余弦相似性只考虑用户评分而忽略了其他信息量的弊端，特别适合于应用到稀疏度过高的数据，下面是几个常用的应用场景：

（1）过滤相似度很高的新闻，或者网页去重

（2）考试防作弊系统

（3）论文查重系统

K-Means损失函数和终止条件

损失函数：

目的是最终得到的中心点使得，每个样本到中心点和最小，每个样本到中心点距离和表示为损失函数：

$$loss(a_1, a_2, \dots a_k) = \frac{1}{2}\sum_{j=1}^{k}\sum_{i=1}^{N_j}(x_i - a_i)^2，\ a_i是某个中心点$$

为了使损失函数最小，求偏导可以得到中心点的更新公式为(公式非常容易理解，也可以不从损失函数入手)：

$$a_j = \frac{1}{N(c_j)}\sum_{i \epsilon c_j} x_i$$

终止条件： 提前设置的迭代次数； 最小平方误差MSE；簇中心点变化率是常用的三个条件

K-Means优缺点及其优化变种

K-Means的主要优点：

1）原理比较简单，实现也是很容易，收敛速度快。

2）聚类效果较优。

3）算法的可解释度比较强。

4）主要需要调参的参数仅仅是簇数k

K-Means的主要缺点及优化方法：

1）K值的选取不好把握，除了依靠经验外，可以依据层次聚类，手肘法和轮廓系数法 确定k的值

2）只适合圆形形状(半径)的聚类，(基于密度的聚类算法更加适合，比如DBSCAN算法)

3）如果各隐含类别的数据不平衡，或者各隐含类别的方差不同，则聚类效果不佳

4） 采用迭代方法，得到的结果只是局部最优，很多教程都告诉我们Partition-based methods聚类多适用于中等体量的数据集，但“中等”到底有多“中”，不妨理解成，数据集越大，越有可能陷入局部最优

5） 对噪音和异常点比较的敏感(离群点检测的LOF算法，通过去除离群点后再聚类；改成求点的中位数，这种聚类方式即***K-Mediods***聚类(K中值))，只是每次都要先在组内排序再取中位数，增加了计算量

6） 初始聚类中心的选择(k-means++；二分K-means)

7) 计算所有的样本点到所有的质心的距离，如果样本量非常大，比如达到10万以上，特征有100以上，此时传统K-Means算法很耗时，减小K值和降维是一种思路。另外Mini Batch K-Means应运而生：选择一个合适的批样本大小batch size，仅用batch size个样本来做K-Means聚类。这batch size个样本一般是通过无放回的随机采样得到的。为了算法的准确性，多跑几次Mini Batch K-Means算法，用不同的随机采样集来

获得聚类簇。为保证算法的准确性，一般会多跑几次Mini Batch K-Means算法，用不同的随机采样集来得到聚类簇，选择其中最优的聚类簇

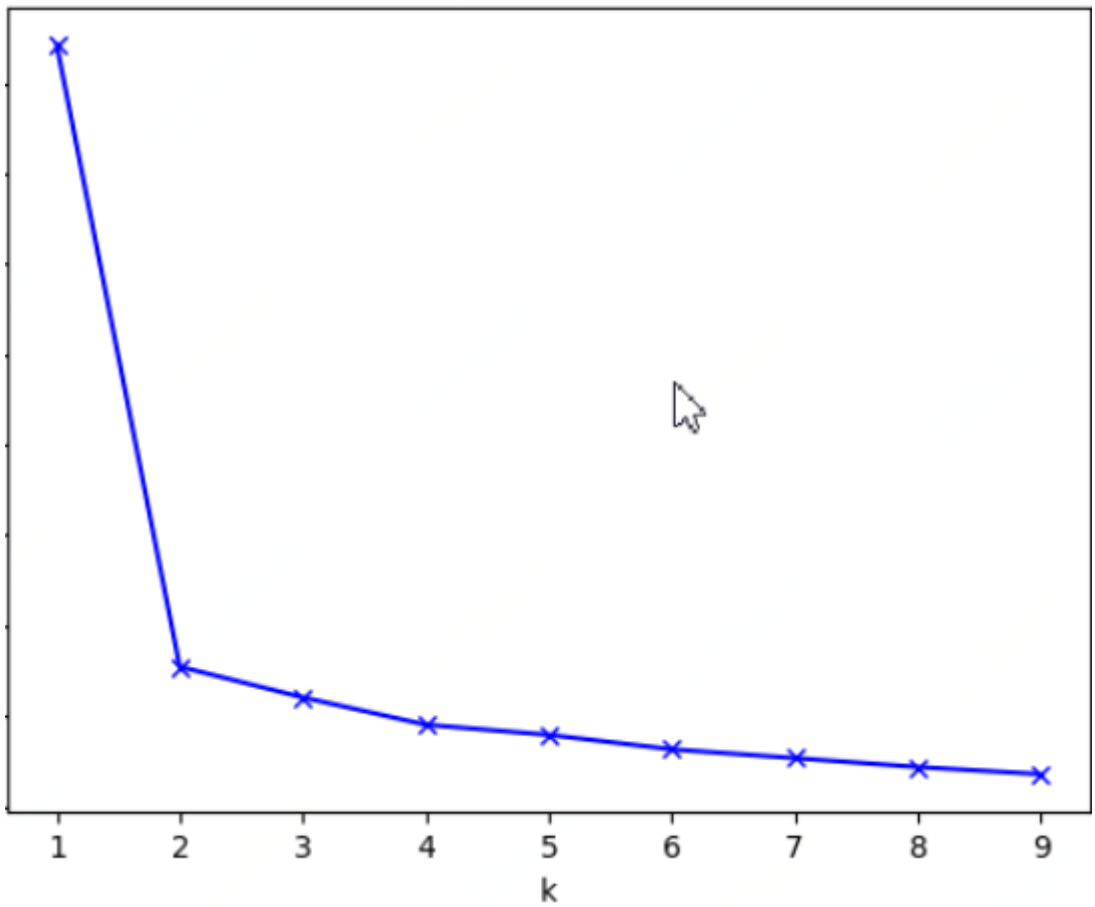
8）只用于numerical类型数据，可以用K-modes聚类解决categorical类型的数据

9）k-means不能解决非凸（non-convex）数据，所以有了kernel k-means

手肘法与二分K均值:

1)**手肘法：**核心指标是SSE，随着聚类数k的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和SSE自然会逐渐变小。当k小于真实聚类数时，由于k的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大，而当k到达真实聚类数时，再增加k所得到的聚合程度回报会迅速变小，所以SSE的下降幅度会骤减，然后随着k值的继续增大而趋于平缓，也就是说SSE和k的关系图是一个手肘的形状，而这个肘部对应的k值就是数据的真实聚类数。当然，这也是该方法被称为手肘法的原因。

(k=2)手肘法示意图



简单写几行代码实现一下吧

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
df = pd.read_csv() # 读入数据
# 利用SSE选择k
SSE = []  # 存放每次结果的误差平方和
for k in range(1,9):
    estimator = KMeans(n_clusters=k)  # 构造聚类器
    estimator.fit(df[[]])
    SSE.append(estimator.inertia_)
X = range(1,9)
plt.xlabel('k')
plt.ylabel('SSE')
plt.plot(X,SSE,'o-')
plt.show()
```

6）**二分K-Means算法**

二分的主要思想是：首先将所有点作为一个簇，然后将该簇一分为二。之后选择能最大程度降低聚类代价函数（也就是误差平方和）的簇划分为两个簇。以此进行下去，直到簇的数目等于用户给定的数目k为止。

二分k-means算法对初始质心的选择不太敏感，因为初始时只选择一个质心。

伪代码：

将所有数据点看成一个簇

while 簇数目小于k:

For 每一个簇:

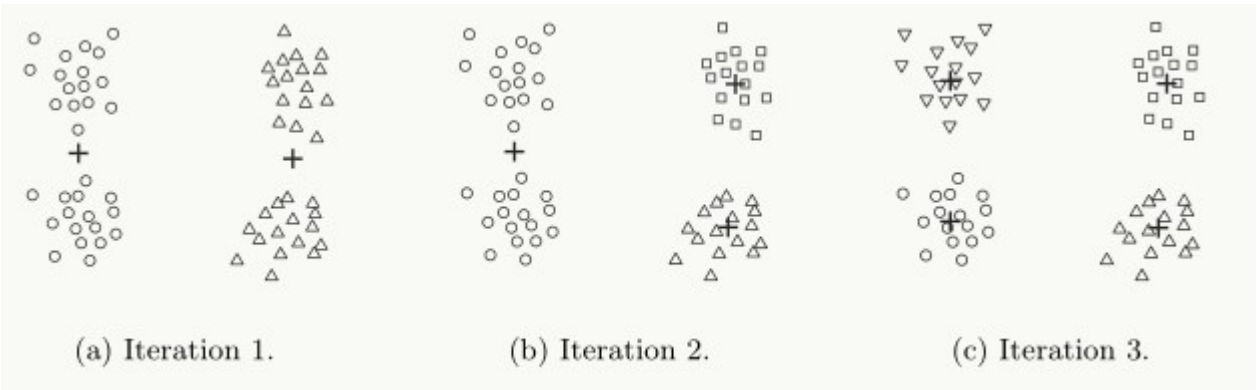
计算总误差

在给定的簇上面进行k-均值聚类（k=2）

计算将该簇一分为二后的总误差

选择使得误差最小的那个簇进行递归操作

二分k均值图示



2. 基于层次的聚类

思想：

还记得，这是本科多元统计里的一节。层次聚类主要有两种类型：合并的层次聚类和分裂的层次聚类。前者是一种自底向上的层次聚类算法，从最底层开始，每一次通过合并最相似的聚类来形成上一层次中的聚类，整个当全部数据点都合并到一个聚类的时候停止或者达到某个终止条件而结束，大部分层次聚类都是采用这种方法处理。后者是采用自顶向下的方法，从一个包含全部数据点的聚类开始，然后把根节点分裂为一些子聚类，每个子聚类再递归地继续往下分裂，直到出现只包含一个数据点的单节点聚类出现，即每个聚类中仅包含一个数据点。

特点：处理速度很快，通常这是与目标数据库中记录的个数无关的，只与把数据空间分为多少个单元有关。

主要算法：BIRCH、CURE、CHAMELEON

算法流程(自下向上为例)：

1. 将每个对象看作一类，计算两两之间的最小距离；
2. 将距离最小的两个类合并成一个新类；
3. 重新计算新类与所有类之间的距离；
4. 重复2、3，直到所有类最后合并成一类

优点：可解释性好(如当需要创建一种分类法时)，研究表明这些算法能产生高质量的聚类，K-means聚类k值的选

取可事先参考，可以解决K-means不能解决的非球形簇。

缺点：时间复杂度高，o(m^3)，改进后的算法也有o(m^2logm)，m为点的个数，一步错步步错(贪心算法的缺点)

3.基于密度的聚类

思想：

k-means解决不了不规则形状的聚类。于是就有了Density-based methods来系统解决这个问题，克服基于距离的算法只能发现"类圆形"的聚类的缺点。该方法同时也对噪声数据的处理比较好。其原理简单说画圈儿，其中要定义两个参数，一个是圈儿的最大半径，一个是一个圈儿里最少应容纳几个点。只要邻近区域的密度（对象或数据点的数目）超过某个阈值，就继续聚类,最后在一个圈里的，就是一个类。DBSCAN（Density-Based Spatial Clustering of Applications with Noise）是其中的典型

主要算法：DBSCAN、OPTICS、DENCLUE

算法流程(DBSCAN)：

1. 从任一对象点p开始
2. 寻找并合并核心p对象直接密度可达（eps）的对象
3. 如果p是一个核心点，则找到了一个聚类，如果p是一个边界点(即从p没有密度可达的点)则寻找下一个对象点
4. 重复2、3，直到所有点都被处理

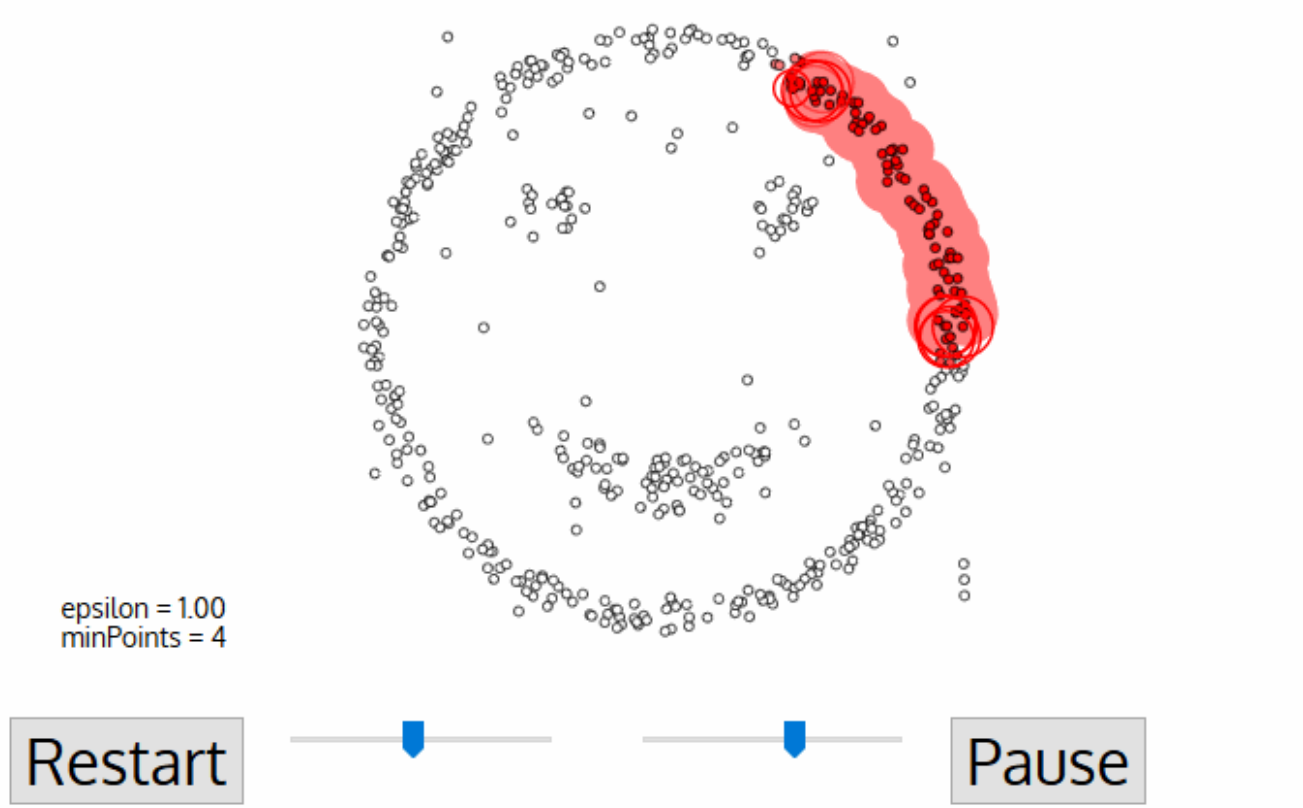
优点：对噪声不敏感；能发现任意形状的聚类

缺点：聚类的结果与参数关系很大，用固定参数识别聚类，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。但当聚类的稀疏程度不同时，相同的判定标准可能会破坏聚类的自然结构，即较稀的聚类

会被划分为多个类或密度较大且离得较近的类会被合并成一个聚类

DBSCAN图示



改进：DBSCAN的扩展叫OPTICS(Ordering Points To Identify Clustering Structure)，通过优先对高密度（high density）进行搜索，然后根据高密度的特点设置参数，改善了DBSCAN的不足

4. 基于网络的聚类

思想：

基于网络的方法：这类方法的原理就是将数据空间划分为网格单元，将数据对象集映射到网格单元中，并计算每个单元的密度。根据预设的阈值判断每个网格单元是否为高密度单元，由邻近的稠密单元组形成"类"

特点： 与层次聚类一样, 速度快，与目标数据库中记录的个数无关，只与把数据空间分为多少个单元有关。

主要算法：STING、CLIQUE、WAVE-CLUSTER

算法流程(DBSCAN)：

算法用不同的网格划分方法，将数据空间划分成为有限个单元（cell）的网格结构,并对网格数据结构进行了不同的处理，但核心步骤是相同的：

1. 划分网格
2. 使用网格单元内数据的统计信息对数据进行压缩表达
3. 基于这些统计信息判断高密度网格单元
4. 最后将相连的高密度网格单元识别为簇

优点：速度很快，因为其速度与数据对象的个数无关，而只依赖于数据空间中每个维上单元的个数。 **缺点：**参数敏感、无法处理不规则分布的数据、维数灾难等；这种算法效率的提高是以聚类结果的精确性为代价

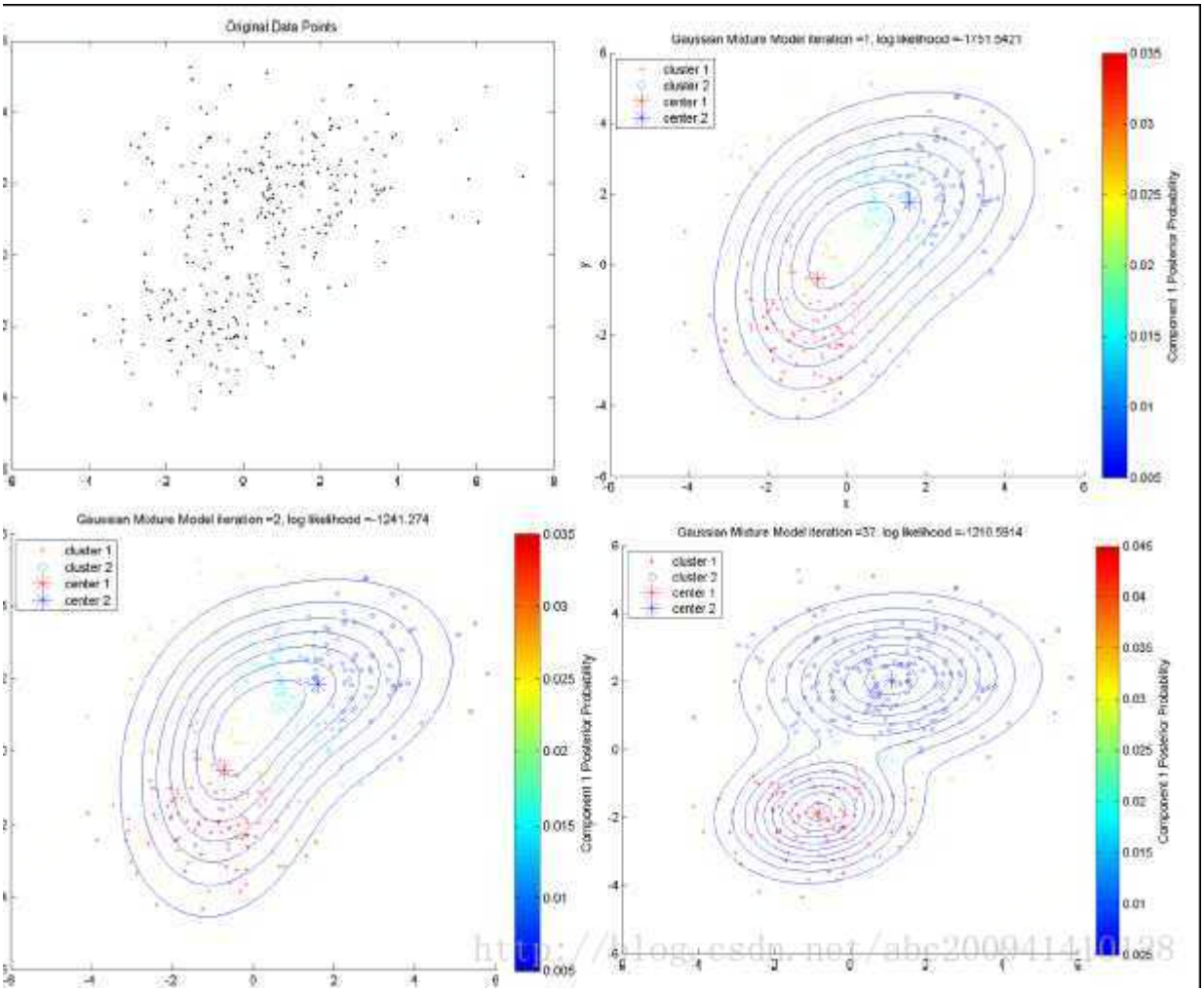
的。经常与基于密度的算法结合使用。

5. 基于模型的聚类(GMM)

思想：

基于模型的方法(Model-based methods)，为每簇假定了一个模型，寻找数据对给定模型的最佳拟合，这一类方法主要是指基于概率模型的方法和基于神经网络模型的方法，尤其以基于概率模型的方法居多。这里的概率模型主要指概率生成模型（generative Model），同一"类"的数据属于同一种概率分布，即假设数据是根据潜在的概率分布生成的。其中最典型、也最常用的方法就是高斯混合模型（GMM，Gaussian Mixture Models）。基于神经网络模型的方法主要就是指SOM（Self Organized Maps）了，也是我所知的唯一一个非监督学习的神经网络了。下图表现的就是GMM的一个demo，里面用到**EM**算法来做最大似然估计：

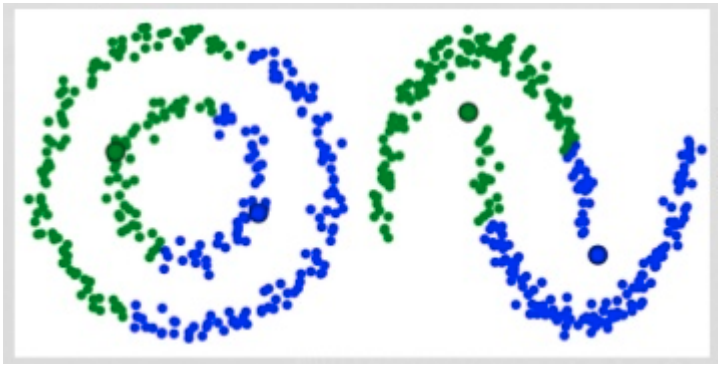
GMM图示



下面详细介绍 使用高斯混合模型（GMM）的期望最大化（EM）聚类：

K-Means的一个主要缺点是它对聚类中心的平均值的使用很简单幼稚。我们可以通过看下面的图片来了解为什么这不是最好的方法。在左边看起来很明显的是，有两个圆形的聚类，不同的半径以相同的平均值为中心。K-Means无法处理，因为聚类的均值非常接近。在聚类不是循环的情况下，K-Means也会失败，这也是使用均值作为聚类中心的结果.

K均值失败两个案例



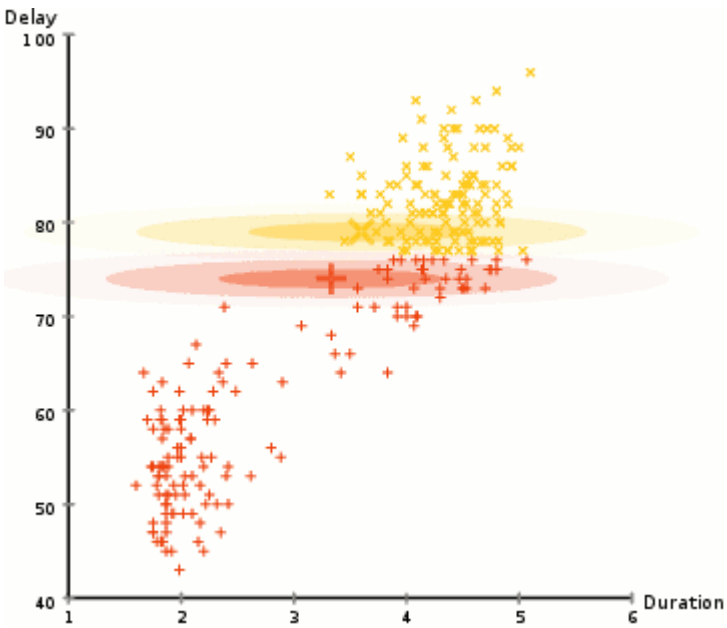
高斯混合模型（GMMs）比K-Means更具灵活性。使用高斯混合模型，我们可以假设数据点是高斯分布的;比起说它们是循环的，这是一个不那么严格的假设。这样，我们就有两个参数来描述聚类的形状：平均值和标准差！以二维的例子为例，这意味着聚类可以采用任何形式的椭圆形状（因为在x和y方向上都有标准差）。因此，每个高斯分布可归属于一个单独的聚类。

所谓混合高斯模型（GMM）就是指对样本的概率密度分布进行估计，而估计采用的模型（训练模型）是几个高斯模型的加权和（具体是几个要在模型训练前建立好）。每个高斯模型就代表了一个类（一个Cluster）。对样本中的数据分别在几个高斯模型上投影，就会分别得到在各个类上的概率。然后我们可以选取概率最大的类所为判决结果。

从中心极限定理的角度上看，把混合模型假设为高斯的是比较合理的，当然，也可以根据实际数据定义成任何分布的Mixture Model,不过定义为高斯的在计算上有一些方便之处，另外，理论上可以通过增加Model的个数，用GMM近似任何概率分布。

为了找到每个聚类的高斯分布的参数（例如平均值和标准差）我们将使用一种叫做期望最大化（EM）的优化算法。看看下面的图表，就可以看到高斯混合模型是被拟合到聚类上的。然后，我们可以继续进行期望的过程——使用高斯混合模型实现最大化聚类。

GMM动图



高斯混合模型来期望最大化聚类流程

1. 我们首先选择聚类的数量（如K-Means所做的那样），然后随机初始化每个聚类的高斯分布参数。通过快速查看数据，可以尝试为初始参数提供良好的猜测。注意，在上面的图表中可以看到，这并不是100%的必要，因为高斯开始时的表现非常不好，但是很快就被优化了。
2. 给定每个聚类的高斯分布，计算每个数据点属于特定聚类的概率。一个点离高斯中心越近，它就越有可能属于那个聚类。这应该是很直观的，因为有一个高斯分布，我们假设大部分的数据都离聚类中心很近。
3. 基于这些概率，我们为高斯分布计算一组新的参数，这样我们就能最大程度地利用聚类中的数据点的概率。我们使用数据点位置的加权和来计算这些新参数，权重是属于该特定聚类的数据点的概率。为了解释这一点，我们可以看一下上面的图，特别是黄色的聚类作为例子。分布在第一次迭代中是随机的，但是我们可以看到大多数的黄色点都在这个分布的右边。当我们计算一个由概率加权的和，即使在中心附近有一些点，它们中的大部分都在右边。因此，自然分布的均值更接近于这些点。我们还可以看到，大多数点都是“从右上角到左下角”。因此，标准差的变化是为了创造一个更符合这些点的椭圆，从而使概率的总和最大化。
4. 步骤2和3被迭代地重复，直到收敛，在那里，分布不会从迭代到迭代这个过程中变化很多。

使用高斯混合模型有两个关键的优势。首先，高斯混合模型在聚类协方差方面比K-Means要灵活得多;根据标准差参数，聚类可以采用任何椭圆形状，而不是局限于圆形。K-Means实际上是高斯混合模型的一个特例，每个聚类在所有维度上的协方差都接近0。其次，根据高斯混合模型的使用概率，每个数据点可以有多个聚类。因此，如果一个数据点位于两个重叠的聚类的中间，通过说X%属于1类，而y%属于2类，我们可以简单地定义它的类。

GMM优点：投影后样本点不是得到一个确定的分类标记，而是得到每个类的概率，这是一个重要信息。

GMM缺点：每一步迭代的计算量比较大，大于k-means。GMM的求解办法基于EM算法，因此有可能陷入局部极值，这和初始值的选取十分相关了。GMM不仅可以用在聚类上，也可以用在概率密度估计上。

6.基于模糊的聚类(FCM)

思想：

1965年美国加州大学柏克莱分校的扎德教授第一次提出了‘集合’的概念。经过十多年的发展，模糊集合理论渐渐被应用到各个实际应用方面。为克服非此即彼的分类缺点，出现了以模糊集合论为数学基础的聚类分析。用模糊数学的方法进行聚类分析，就是模糊聚类分析。 基于模糊集理论的聚类方法，样本以一定的概率属于某个类。比较典型的有基于目标函数的模糊聚类方法、基于相似性关系和模糊关系的方法、基于模糊等价关系的传递闭包方法、基于模 糊图论的最小支撑树方法，以及基于数据集的凸分解、动态规划和难以辨别关系等方法。FCM算法是一种以隶属度来确定每个数据点属于某个聚类程度的算法。该聚类算法是传统硬聚类算法的一种改进。

FCM模糊聚类算法流程：

1. 标准化数据矩阵
2. 建立模糊相似矩阵，初始化隶属矩阵
3. 算法开始迭代，直到目标函数收敛到极小值
4. 根据迭代结果，由最后的隶属矩阵确定数据所属的类，显示最后的聚类结果

FCM算法需要两个参数一个是聚类数目C，另一个是参数m。一般来讲C要远远小于聚类样本的总个数，同时要保证C>1。对于m，它是一个控制算法的柔性的参数，如果m过大，则聚类效果会很次，而如果m过小则算法会接近HCM聚类算法。

算法的输出是C个聚类中心点向量和C*N的一个模糊划分矩阵，这个矩阵表示的是每个样本点属于每个类的隶属度。根据这个划分矩阵按照模糊集合中的最大隶属原则就能够确定每个样本点归为哪个类。聚类中心表示的是每个类的平均特征，可以认为是这个类的代表点。

优点：算法对于满足正态分布的数据聚类效果会很好，另外，算法对孤立点是敏感的。

缺点：由于不能确保FCM收敛于一个最优解。算法的性能依赖于初始聚类中心。因此，要么用另外的快速算法确

定初始聚类中心，要么每次用不同的初始聚类中心启动该算法，多次运行FCM。

改进：模糊C均值（简称FCM）聚类算法是HCM聚类算法的改进

参考文献与链接

peghoty .Clustering by fast search and find of density peak.Science 2014

百度百科聚类

K均值距离

K均值优化

机器学习之聚类算法

聚类算法的分类

五种聚类算法

GMM

声明

写了两天，终于写完啦！本文对聚类的介绍已经很详细了，不了解聚类的同学可以把它当做聚类入门，本文为个人整理，仅限学习使用，转载请标明作者和来源。码字不易，如果觉得不错，git上请点个***star***吧，谢谢配合！