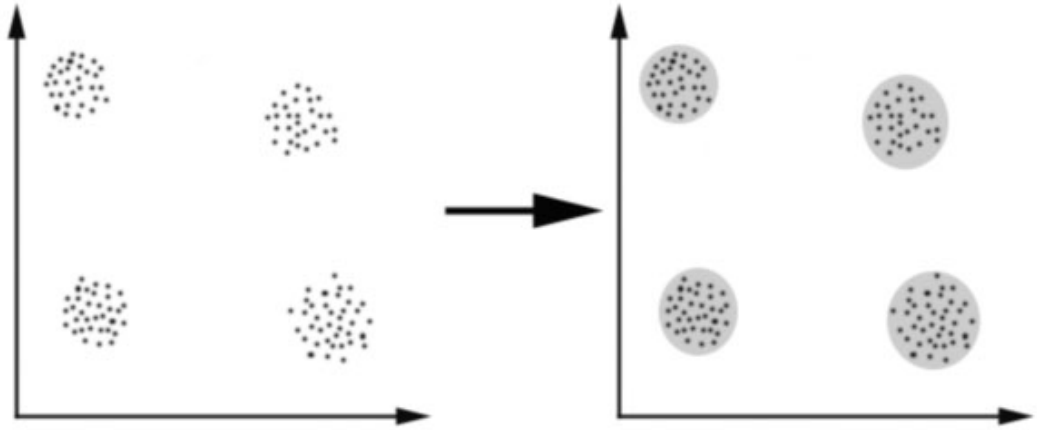
目录

聚类简介

聚类是什么?

聚类是一种涉及数据点分组的机器学习技术。给定一组数据点，我们可以使用聚类算法将每个数据点分类到特定的组中。理论上，同一组中的数据点应具有相似的属性或特征，而不同组中的数据点应具有较大差异的属性或特征。聚类是无监督学习的一种方法，也是用于许多领域中统计数据分析的常用技术。

我们可以用一个简单的图形例子来说明:



在上图中，我们很容易看出可以将数据划分为4个Cluster（簇）。

相似性的标准是距离:如果两个或多个物体根据给定的距离(在本例中是几何距离)“接近”，则它们属于同一簇。这称为基于距离的聚类。

另一种聚类是概念聚类:两个或多个对象属于同一个簇，如果这个簇定义了所有这些对象共有的概念。换句话说，对象是根据它们对描述性概念的适合程度来分组的，而不是根据简单的相似性度量。

聚类的目标

聚类的目标是确定一组未标记数据中的固有分组。但是如何决定什么构成一个好的簇呢？可以看出，并没有绝对的“最佳”标准，它独立于聚类的最终目标。因此，用户必须提供这个标准，以便聚类的结果能够满足他们的需要。

应用

聚类算法可以应用于很多领域，例如:

- 市场营销:给定一个包含客户属性和以往购买记录的大型客户数据数据库，寻找行为相似的客户群
- 生物学:根据植物和动物的特征进行分类
- 图书馆:图书订购
- 保险:确定平均理赔费用较高的汽车保单持有人群体，识别欺诈行为
- 城市规划:根据房屋类型、价值和地理位置，对房屋进行分类
- 地震研究:聚类观测震源，识别危险区域
- WWW:文档分类，对weblog数据进行聚类，以发现具有类似访问模式的组。

要求

聚类算法需要满足的主要要求是:

- 可伸缩性
- 处理不同类型的数据
- 发现任意形状的簇
- 领域最小化

- 处理噪音和异常值的能力
- 对输入数据的顺序不敏感
- 高维度
- 可解释性和可用性

难点

- 当前的聚类技术不能充分(同时)满足所有需求
- 由于时间的复杂性，处理大量维度和大量数据项可能是有问题的
- 该方法的有效性取决于“距离”的定义(基于距离的聚类)
- 如果一个明显的距离度量不存在，我们必须“定义”它，这并不总是容易的，尤其是在多维空间中
- 聚类算法的结果(在许多情况下，聚类算法本身可以是任意的)可以用不同的方式解释

聚类算法

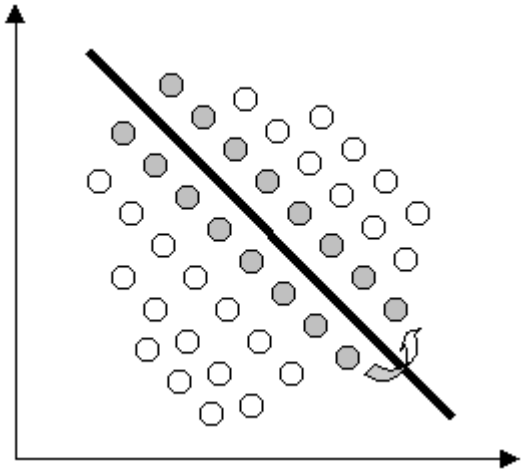
所谓聚类问题，就是给定一个元素集合D，其中每个元素具有n个可观察属性，使用某种算法将D划分成k个子集，要求每个子集内部的元素之间相异度尽可能低，而不同子集的元素相异度尽可能高。其中每个子集叫做一个簇。

算法分类

聚类算法可以分为以下几种:

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

1.在第一种情况下，数据以排他性的方式分组，因此，如果某个数据属于某个簇，那么它就不能包含在另一个簇中。一个简单的例子如下图所示，点的分离是通过二维平面上的一条直线实现的。



2.与此相反，Overlapping Clustering采用模糊集合（fuzzy sets）对数据进行聚类，使每个点可以同时属于多个群组。当然，这个点与每一个群组的耦合程度是不同的，有的紧密一点，有的疏松一点，但都属于阈值范围内的。

3.Hierarchical Clustering算法是基于两个最近的簇之间的并集。先把每个点都看作一个簇，然后再把最近的两个簇聚合成为一个新的簇，如此这般，不断迭代，直到满足最后设定的要求。

4.Probabilistic Clustering使用完全概率的方法。

在本教程中，我们提出了四种最常用的聚类算法:

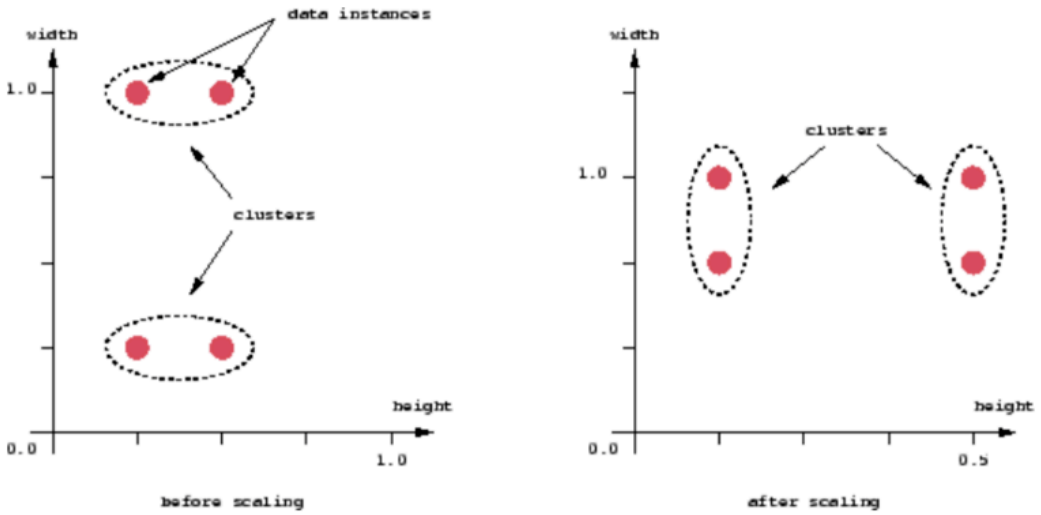
- k-means
- Fuzzy C-means
- 分层聚类
- 混合高斯模型

这些算法都属于上面列出的一种聚类类型。因此，k均值是Exclusive Clustering算法，模糊c均值是Overlapping Clustering算法，层次聚类属于Hierarchical Clustering算法，高斯混合模型属于Probabilistic Clustering算法。我们将在下文中讨论每一种聚类方法。

距离度量

聚类算法的一个重要内容是数据点之间的距离度量。
如果数据实例向量都在相同的物理单元中，那么简单的欧式距离度量就有可能足以成功地对相似的数据实例进行分组。然而，即使在这种情况下，欧式距离有时也会产生误导。

在下图中，左图height刻度每单位0.25，与纵坐标相同，看起来分成上下两组刚刚好。然而，如果通过重新调整坐标，将横坐标做拉伸处理，将每单位定义成0.125，肉眼看起来分成左右两组才更合适了。在左右两种聚类方式中，点的坐标没有变，唯一改变的就是横坐标比例。而就因为如此，分组方式完全变了。



聚类聚得“好不好”，通常无法用绝对值来衡量，主要还是看你的目标是什么，也就是为何聚类。
同样，我们需要借助相关领域的知识来指导如何为每个特定应用制定适当的距离度量。

闵可夫斯基度规

对于高维数据，最常用的测量方法是闵可夫斯基度规：

$$d_p(x_i, x_j) = (\sum_{K=1}^{d} |x_{i,k} - x_{j,k}|^p)^{\frac{1}{p}}$$

其中d为数据的维数。欧式距离是p=2的特殊情况，曼哈顿度规是p=1。然而，对于为指定的应用选择度量标准并没有通用的理论。
通常情况下，数据特征向量的组成部分不能直接进行比较。它可能不是连续变量(如长度)，而是名义类别(如星期几)。在这些情况下，必须再次凭借相关领域的知识来制定适当的度量。

K-Means

算法

K-means (MacQueen, 1967)是最简单的非监督学习算法之一，解决了众所周知的聚类问题。该过程遵循一种简单易行的方法，通过一定数量的簇(假设k个簇)对给定的数据集进行分类。

K-Means算法的思想很简单，对于给定的样本集，按照样本之间的距离大小，将样本集划分为K个簇。让簇内的点尽量紧密的连在一起，而让簇间的距离尽量的大。

该算法的目标是最小化目标函数，在本例中是平方误差函数。
目标函数：

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2$$

式中$||x_i^{(j)} - c_j||^2$为数据点$x_i^{(j)}$与聚类中心c_j之间选择的距离度量，为n个数据点与各自聚类中心之间距离的指示器。

算法由以下步骤组成:

- 将K个点放入由聚集的对象表示的空间中。这些点表示初始群心。
- 将每个对象分配给最接近质心所在的簇。
- 分配完所有对象后，重新计算K个质心的位置。
- 重复步骤2和3，直到重心不再移动。这产生了将对象分成组的簇，从中可以计算要最小化的度量。

虽然可以证明过程总是可以终止的，但是k-means算法不一定能找到最优的配置，对应的是全局目标函数的最小值。

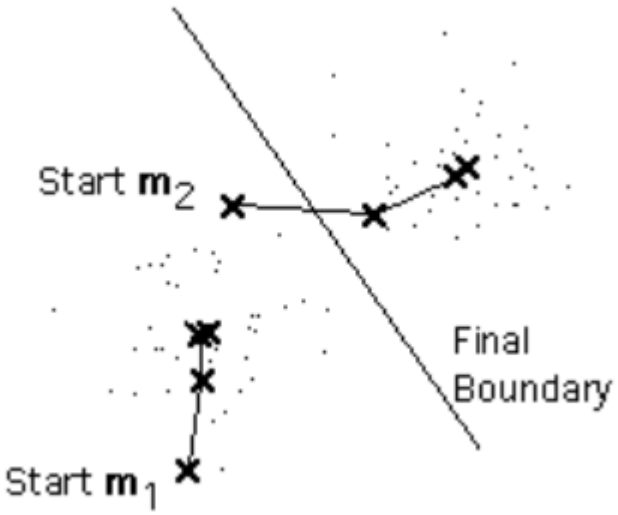
该算法对初始随机选择的聚类中心也具有显著的敏感性。k-means算法可以多次运行以减少这种影响。
K-means是一种简单的算法，适用于许多问题领域。正如我们将要看到的，它非常适合处理模糊特征向量。

示例

假设我们有n个样本特征向量$x_1, x_2 \ldots x_n$都来自同一个类，我们知道它们属于k个紧簇，$k < n$，设m_i为簇i中向量的均值，如果簇之间的距离很好，我们可以使用最小距离分类器将它们分离。也就是说，如果$||x - m_i||$是所有k个距离的最小值，我们可以说x在簇i中。求k的方法如下:

- 对平均值$m_1, m_2, \ldots, m_n$进行初步猜测
- until均值没有变化
  - 利用估计的方法对样本进行聚类
  - for i from 1 to k
    - 将m_i替换为集群i中所有样本的均值
    - end_for
  - end_until

下面是一个例子，说明均值m_1和m_2如何移动到两个集群的中心。



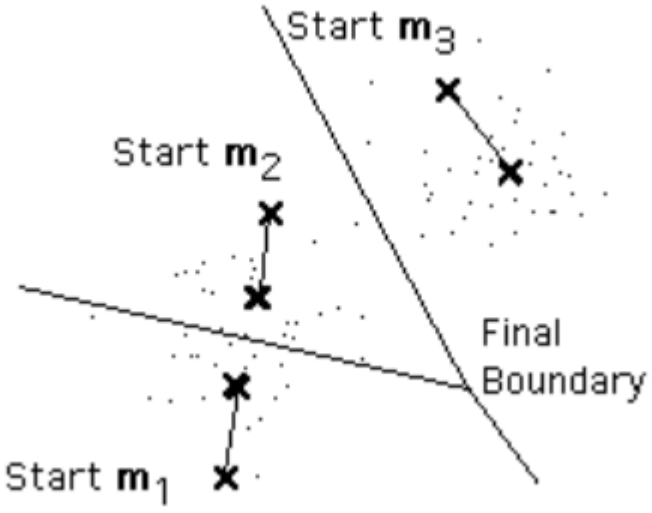
注意

这是k-means程序的一个简单版本。它可以看作是一种贪婪算法，将n个样本划分为k个簇，从而最小化到簇中心距离的平方和。它确实有一些缺点:

- 没有指定初始化方法。一种常用的开始方法是随机选择k个样本值。
- 产生的结果依赖于平均值的初值，并且经常会发现分区不是最优的。标准的解决方案是尝试许多不同的起点。
- 可能是最接近m_i的样本集为空，导致m_i无法更新。这是一个必须在实现中处理的问题，但是我们应该忽略它。

- 结果取决于用来测量$||x - m_i||$的度规。一个流行的解决方案是通过每个变量的标准差来标准化，尽管这并不总是可取的。
- 结果取决于k的值。

最后一个问题特别麻烦，因为我们通常无法知道存在多少个簇。在上面的例子中，相同的算法应用于相同的数据，会产生以下3种聚类方法。它比2均值聚类更好还是更差呢?



不幸的是，对于任何给定的数据集，没有找到最优的簇的数量的一般理论解决方案。一种简单的方法是用不同的k类比较多个运行的结果，并根据给定的标准选择最佳的一个。但是我们需要小心，因为根据定义，增加k会导致更小的误差函数值，但也会增加过拟合的风险。

Fuzzy C-Means

算法

Fuzzy C-Means (FCM)是一种聚类方法，它允许一段数据属于两个或更多的聚类。这种方法(Dunn在1973年开发，Bezdek在1981年改进)经常用于模式识别。它基于以下目标函数的最小化:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m ||x_i - c_j||^2, 0 \leq m < \infty$$

m是任何大于1的实数，u_{ij}是x_i在集群j中的隶属度,x_i是第i个d维测量数据,c_j是簇的d维中心,$||*||$是表示任意测量数据与中心相似度的任意范数。
对上述目标函数进行迭代优化，对隶属度u_{ij}和聚类中心c_j进行更新，进行模糊划分:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{||x_i - x_j||}{||x_i - c_k||})^{\frac{2}{m-1}}}$$

$$C_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

迭代将在$max_{ij}\left(\left|u_{ij}^{(k+1)} - u_{ij}^k\right|\right) < \varepsilon$时停止，$\varepsilon$为0 ~ 1之间的终止准则，而k是迭代步骤。这个过程收敛到J_m的局部极小值或鞍点。
算法由以下步骤组成:

1. *Initialize* $U=[u_{ij}]$ *matrix,* $U^{(0)}$
2. *At k-step: calculate the centers vectors* $C^{(k)}=[c_j]$ *with* $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

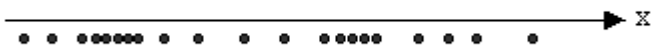
3. *Update* $U^{(k)}$, $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

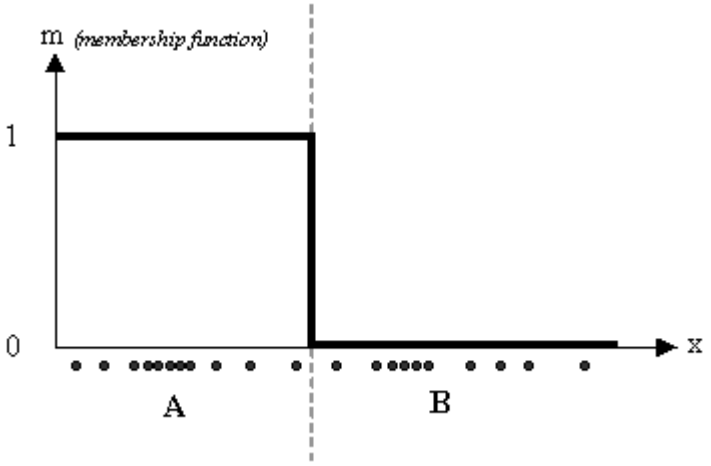
4. *If* $||\ U^{(k+1)} - U^{(k)}|| < \varepsilon$ *then STOP; otherwise return to step 2.*

如前所述，数据通过隶属函数绑定到每个簇，该隶属函数表示该算法的模糊行为。要做到这一点，我们只需构建一个名为U的适当矩阵，它的因数是0到1之间的数字，并表示数据和簇中心之间的隶属度。

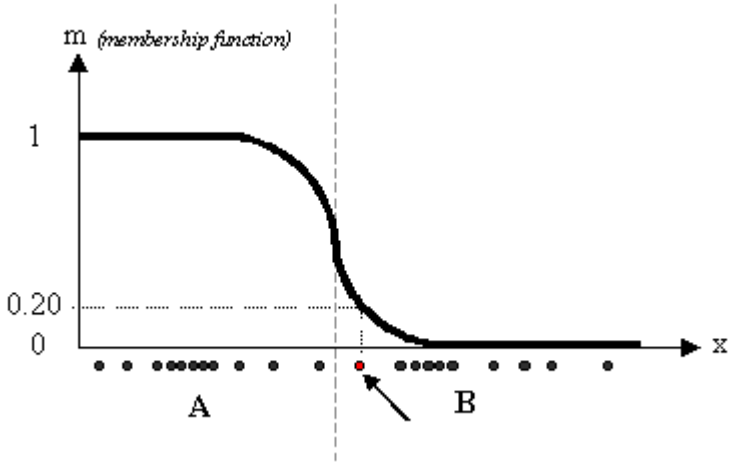
为了更好地理解，我们可以参考一个简单的一维示例。
给定一个数据集，假设将其表示为分布在轴上。如下图所示:



看这幅图，我们可以在两个数据集中附近找到两个簇。我们将用“A”和“B”来指代它们。在本教程中展示的第一种方法k-means算法—中，我们将每个数据关联到一个特定的质心;因此，这个隶属度函数是这样的:



相反，在FCM方法中，相同的给定数据并不完全属于定义良好的簇，但是可以将其放在中间位置。在这种情况下，隶属度函数沿着一条更平滑的线表示每个数据可能属于几个具有不同隶属度系数值的簇。



在上图中，以红色标记的点表示的数据更多的属于B簇，而不是A簇。值“m”=0.2表示该数据到A的隶属度。现在，我们不使用图形表示，我们引入一个矩阵U，其因子是从隶属函数中提取的:

$$U_{MC} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ .. & .. \\ 0 & 1 \end{bmatrix} \qquad U_{MC} = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \\ 0.6 & 0.4 \\ .. & .. \\ 0.9 & 0.1 \end{bmatrix}$$

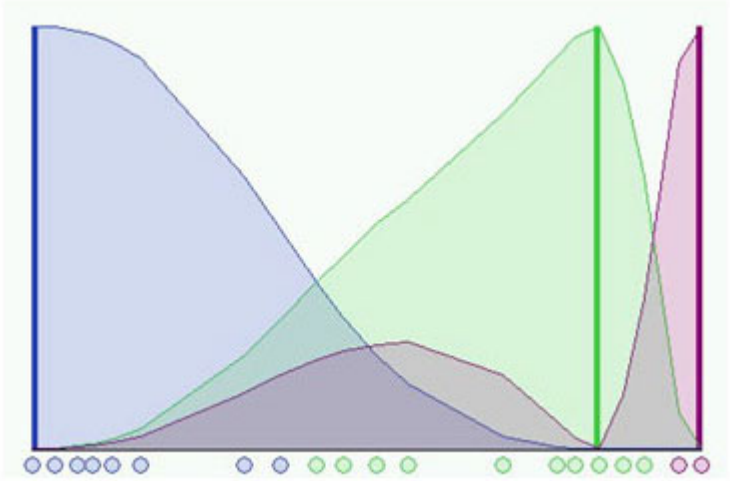
(a) (b)

行和列的数量取决于我们正在考虑的数据和簇的数量。更准确地说，我们有C = 2列(C = 2个簇)和N行，其中C是簇的总数，N是数据的总数。通用元素是这样表示的:u_{ij}。
在上面的例子中，我们考虑了k均值(a)和FCM (b)的情况。我们可以注意到在第一种情况下(a)系数总是单一的。这样做是为了表明每个数据只能属于一个簇。其他属性如下所示:

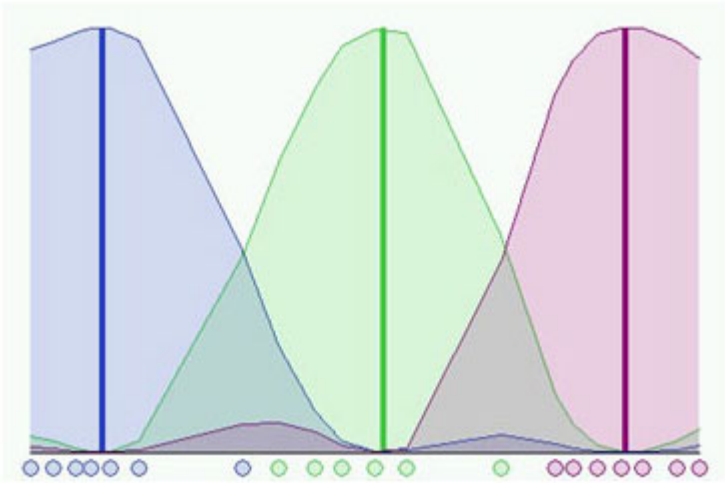
- $u_{ij} \in [0,1] \quad \forall i,j$
- $\sum_{j=1}^{C} u_{ik} = 1 \quad \forall i$
- $0 < \sum_{i=1}^{N} u_{ij} < N \quad \forall N$

示例

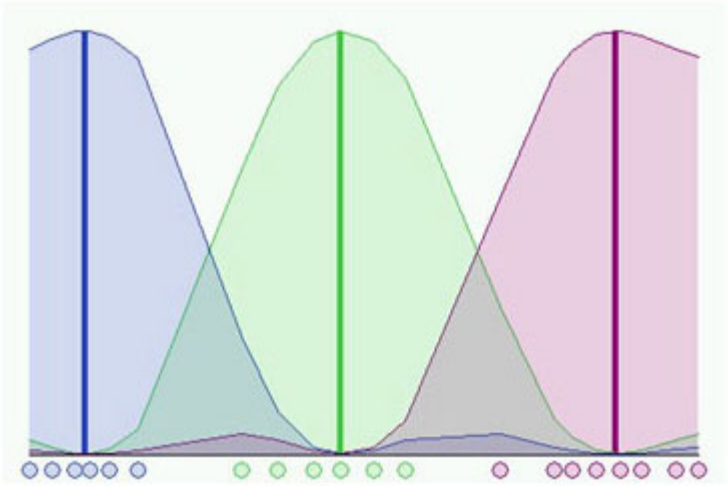
在这里，我们考虑FCM的一维应用程序的简单情况。20个数据和3个簇用于初始化算法和计算U矩阵。下图(取自我们的交互式演示)显示了每个数据和每个簇的成员资格值。根据隶属度函数，数据的颜色是最近的簇的颜色。



在上图所示的仿真中，我们使用了一个模糊系数m = 2，并且当$max_{ij}\left(|u_{ij}^{(k+1)} - u_{ij}^{k}|\right) < 0.3$时强制终止算法。图中显示了模糊分布依赖于聚类特定位置的初始条件。还没有执行任何步骤，因此不能很好地标识簇。现在我们可以运行该算法，直到停止条件得到验证。下表为第8步得到的最终条件，m=2， ε=0.3:



有可能做得更好吗?当然，我们可以使用更高的精度，但我们也必须投入更大的计算量。在下一个图中，我们可以看到使用相同的初始条件和ε=0.01得到了更好的结果，但是我们需要37个步骤!



同样需要注意的是，不同的初始化会导致算法的不同演进。事实上，它可以收敛到相同的结果，但可能需要不同数量的迭代步骤。

层次聚类算法

原理

给定一组要聚类的N个项，以及一个N*N的距离(或相似性)矩阵，层次聚类的基本过程(1967年S.C. Johnson定义)为:

1. 首先将每个项分配给一个簇，这样，如果有N个项，那么现在就有N个簇，每个簇只包含一个项。让簇之间的距离(相似性)与其包含的项之间的距离(相似性)相同。

2. 找到最接近(最相似)的一对簇，并将它们合并到单个簇中，这样可以减少一个簇。
3. 计算新簇与每个旧簇之间的距离(相似性)。
4. 重复步骤2和步骤3，直到所有项都聚集到一个大小为n(*)的簇中。

步骤3可以通过不同的方式来完成，这就是单连锁（single-linkage）与完全连锁（complete-linkage）和平均连锁（average-linkage）聚类的区别所在。

- 在single-linkage聚类(也称为连通性或最小值法)中，我们认为一个聚类与另一个聚类之间的距离等于从一个聚类中的任何成员到另一个聚类中的任何成员的最短距离。如果数据包含相似性，我们认为一个簇与另一个簇之间的相似性等于一个簇的任何成员与另一个簇的任何成员之间的最大相似性。
- 在complete-linkage聚类(也称为直径或最大值法)中，我们认为一个聚类与另一个聚类之间的距离等于一个聚类中的任何成员与另一个聚类中的任何成员之间的最大距离。
- 在average-linkage聚类中，我们认为一个聚类与另一个聚类之间的距离等于从一个聚类中的任何成员到另一个聚类中的任何成员的平均距离。

平均链路聚类的一种变化是R.D'Andrade（1978）的UCLUS方法，它使用中值距离，它比平均距离更具异常性。

这种层次聚类被称为聚合聚类，因为它是迭代地合并簇。还有一种分裂的层次聚类，它从一个簇中的所有对象开始，并将它们细分为更小的部分，从而实现相反的效果。分裂的方法一般是不可用的，也很少得到应用。

- 当然，将所有N个项目分组在一个簇中是没有意义的，但是，一旦你获得了完整的层次树，如果想要k个簇，只需删除k-1最长的链接。

Single-Linkage

算法

现在让我们更深入地看看Johnson算法在Single-Linkage聚类情况下是如何工作的。
该算法是一种新的聚类算法，当旧的聚类被合并到新的聚类中时，该算法将消除邻近矩阵中的行和列。
N*N邻近矩阵$D = [D(i,j)]$。聚类的序列编号为0,1，……， (n-1)和L(k)为第k个聚类层次。序号为m的簇记为(m)，簇(r)与簇(s)的邻近度记为$d[(r),(s)]$。
算法由以下步骤组成:

> 1、首先以级别$L(0) = 0$的不相交聚类开始。
> 2、根据$d[(r),(s)] = min d[(i),(j)]$，找出当前聚类中最不相似的对，即对$(r),(s)$。其中最小值为当前簇中的所有对簇。
> 3、增加序号:m = m +1。将聚类(r)和(s)合并到单个聚类中，形成下一个聚类m。将该聚类的级别设置为$L(m) = d[(r),(s)]$

> 4、更新邻近矩阵D，删除与簇(r)和(s)对应的行和列，并添加与新形成的簇对应的行和列。新簇(r,s)与旧簇(k)的邻近度定义如下:
>
> $$d[(k),(r,s)] = mind[(k),(r)],d[(k),(s)]$$
>
> 5、如果所有对象都在一个簇中，则停止。否则，转到步骤2。

示例

现在让我们看一个简单的例子:一些意大利城市之间以公里为单位的距离的层次聚类。使用的方法是Single-Linkage 。

输入距离矩阵(L = 0):

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0

最近的两个城市是MI和TO，距离138。它们被合并到一个名为“MI/TO”的簇中。新簇的级别为L(MI/TO) = 138，新序列号为m = 1。

然后我们计算这个新的复合对象到所有其他对象的距离。在单链路聚类中，规则是复合对象到另一个对象的距离等于从簇中的任何成员到外部对象的最短距离。因此，从MI/到RM的距离被选择为564，也就是从MI到RM的距离，以此类推。

将MI与合并后得到如下矩阵:

	BA	FI	MI/TO	NA	RM
BA	0	662	877	255	412
FI	662	0	295	468	268
MI/TO	877	295	0	754	564
NA	255	468	754	0	219
RM	412	268	564	219	0

min d(i,j) = d(NA,RM) = 219 =>将NA和RM合并成一个新的簇NA/RM

L(NA/RM) = 219

m = 2

	BA	FI	MI/TO	NA/RM
BA	0	662	877	255
FI	662	0	295	268
MI/TO	877	295	0	564
NA/RM	255	268	564	0

min d(i,j) = d(BA,NA/RM) = 255 =>将BA和NA/RM合并到一个名为BA/NA/RM的新簇中

L(BA/NA/RM) = 255

m = 3

	BA/NA/RM	FI	MI/TO
BA/NA/RM	0	268	564
FI	268	0	295
MI/TO	564	295	0

min d(i,j) = d(BA/NA/RM,FI) = 268 =>将BA/NA/RM与FI合并成一个新的簇BA/FI/NA/RM

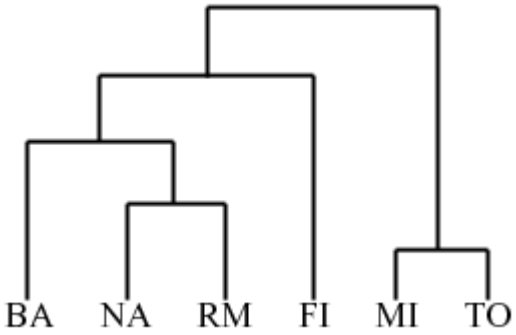
L(BA/FI/NA/RM) = 268

m = 4

	BA/FI/NA/RM	MI/TO
BA/FI/NA/RM	0	295
MI/TO	295	0

最后，我们在295级合并最后两个簇。

该过程总结为以下层次树:



问题

聚类方法的主要缺点是:

- 它们不能很好地伸缩:时间复杂度至少为O(n2)，其中n为总对象数;
- 不能撤销以前做过的事情。

混合高斯模型聚类

基于模型的簇

还有另一种处理聚类问题的方法:基于模型的方法，其中包括使用某些模型进行聚类并尝试优化数据和模型之间的拟合。
在实践中，每个聚类可以通过参数分布在数学上表示，如高斯（连续）或泊松（离散）。因此，整个数据集由这些分布的混合物建模。用于对特定簇建模的单个分布通常称为组件分布。

具有高可能性的混合模型倾向于具有以下特征：

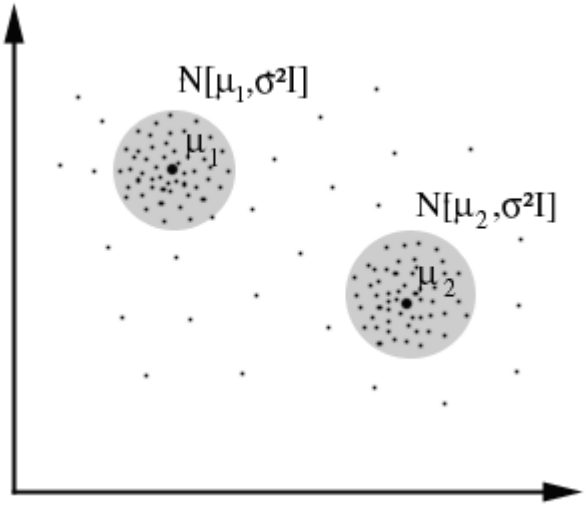
- 组件分布有很高的“峰值”(一个集群中的数据很“紧”);
- 混合模型很好地“覆盖”了数据(数据中的主导模式由组件分布捕获)。

基于模型聚类的主要优点:

- 已有的统计推断技术;
- 组件分布选择的灵活性;
- 得到每个簇的密度估计;
- “软”分类是可用的。

混合高斯模型

这类聚类方法中应用最广泛的是一种基于高斯混合学习的聚类方法:我们实际上可以把聚类看作是以其质心为中心的高斯分布，如图所示，其中灰色圆圈表示分布的第一个方差:



算法是这样工作的:

- 它随机选择概率为$p(\omega_i)$的分量(高斯);
- 它抽样一个点$N(\mu_i, \sigma^2)$。

我们假设有:

- $x_1, x_2, \ldots, x_N$
- $p(\omega_1)$,...,$p(\omega_k)$,σ

我们可以得到样本的概率:$P(x|\omega_i, \mu_1, \mu_2, \ldots, \mu_k)$
我们真正想要的是最大化是$P(x|\mu_1, \mu_2, \ldots, \mu_k)$(给定高斯中心的基准概率)

$$P(x|\mu_i) = \sum_i P(x|\omega_i, \mu_1, \mu_2, \ldots, \mu_k)$$

以此为基础写出似然函数:

$$P(data|\mu_i) = \prod_{i=1}^{N} \sum_i P(w_i) P(x|\omega_i, \mu_1, \mu_2, \ldots, \mu_k)$$

现在我们应该通过计算$\frac{\partial L}{\partial u_i} = 0$来最大化似然函数,但这太困难了。这就是为什么我们使用简化算法EM(期望最大化)。

EM算法

在实际应用中，用来寻找对数据集建模的高斯混合的算法称为EM (expectzation, Dempster, Laird and Rubin, 1977)。让我们看一个例子。
假设x_k是某门课的学生得到的分数，其概率为:

$x_1 = 30 \quad P(x_1) = \frac{1}{2}$
$x_2 = 18 \quad P(x_2) = \mu$
$x_3 = 0 \quad P(x_3) = 2\mu$
$x_4 = 23 \quad P(x_4) = \frac{1}{2} - 3\mu$

第一个案例:我们观察到分数在学生中分布得如此之广:

$\boldsymbol{x_1}$: a students

$\boldsymbol{x_2}$: b students

$\boldsymbol{x_3}$: c students

$\boldsymbol{x_4}$: d students

$$P(a,b,c,d\mid\mu)\propto\left(\frac{1}{2}\right)^{a}*(\mu)^{b}*(2\mu)^{c}*\left(\frac{1}{2}-3\mu\right)^{d}$$

通过计算$\frac{\partial L}{\partial u_i}=0$使这个函数最大化。我们来计算函数的对数，并使其最大化:

$$P_L=\log\left(\frac{1}{2}\right)^{a}+\log(\mu)^{b}+\log(2\mu)^{c}+\log\left(\frac{1}{2}-3\mu\right)^{d}$$

$$\frac{\partial P_L}{\partial\mu}=\frac{b}{\mu}+\frac{2c}{2\mu}-\frac{3d}{\frac{1}{2}-3\mu}=0$$

$$\Rightarrow\mu=\frac{b+c}{6(b+c+d)}$$

假设a = 14, b = 6, c = 9, d = 10，我们可以计算出$\boldsymbol{\mu=\frac{1}{10}}$。

第二种情况:我们观察到分数在学生中是如此的分散:

$\boldsymbol{x_1+x_2}$: h students

$\boldsymbol{x_3}$: c students

$\boldsymbol{x_4}$: d students

这样我们就得到了一个循环，它分为两个步骤:

- 期望值：$\boldsymbol{\mu\rightarrow a=\frac{\frac{1}{2}}{\frac{1}{2}+\mu}h,b=\frac{\mu}{\frac{1}{2}+\mu}h}$
- 极大值：$\boldsymbol{a,b\rightarrow\mu=\frac{b+c}{6(b+c+d)}}$

这种循环性可以用迭代的方法求解。

现在我们来看看EM算法是如何适用于混合高斯函数的（在第p次迭代中估计的参数用上标(p)标记）：

1、参数初值:

$$\lambda_0=\left\{\mu_1^{(0)},\mu_2^{(0)},\ldots,\mu_K^{(0)},p_1^{(0)},p_2^{(0)},\ldots,p_K^{(0)}\right\}$$

2、求期望:

$$P(\omega_j \mid x_k, \lambda_t) = \frac{P(x_k \mid \omega_j, \lambda_t)P(\omega_j \mid \lambda_t)}{P(x_k \mid \lambda_t)} = \frac{P(x_k \mid \omega_i, \mu_i^{(t)}, \sigma^2)p_i^{(t)}}{\sum_k P(x_k \mid \omega_j, \mu_j^{(t)}, \sigma^2)p_j^{(t)}}$$

3、求最大值：

$$\mu_i^{(t+1)} = \frac{\sum_k P(\omega_i \mid x_k, \lambda_t)x_k}{\sum_k P(\omega_i \mid x_k, \lambda_t)}$$

$$p_i^{(t+1)} = \frac{\sum_k P(\omega_i \mid x_k, \lambda_t)}{R}$$

其中R是记录的个数。

参考：

http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/index.html