上篇主要介绍了一种机器学习的通用框架--集成学习方法，首先从准确性和差异性两个重要概念引出集成学习“**好而不同**”的四字真言，接着介绍了现阶段主流的三种集成学习方法：AdaBoost、Bagging及Random Forest，AdaBoost采用最小化指数损失函数迭代式更新样本分布权重和计算基学习器权重，Bagging通过自助采样引入样本扰动增加了基学习器之间的差异性，随机森林则进一步引入了属性扰动，最后简单概述了集成模型中的三类结合策略：平均法、投票法及学习法，其中Stacking是学习法的典型代表。本篇将讨论无监督学习中应用最为广泛的学习算法--聚类。

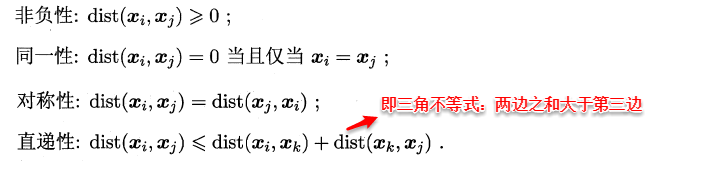
#**10、聚类算法**

聚类是一种经典的**无监督学习**方法，**无监督学习的目标是通过对无标记训练样本的学习，发掘和揭示数据集本身潜在的结构与规律**，即不依赖于训练数据集的类标记信息。聚类则是试图将数据集的样本划分为若干个互不相交的类簇，从而每个簇对应一个潜在的类别。

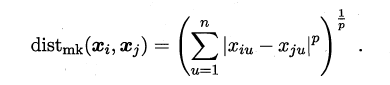
聚类直观上来说是将相似的样本聚在一起，从而形成一个**类簇（cluster）**。那首先的问题是如何来**度量相似性**（similarity measure）呢？这便是**距离度量**，在生活中我们说差别小则相似，对应到多维样本，每个样本可以对应于高维空间中的一个数据点，若它们的距离相近，我们便可以称它们相似。那接着如何来评价聚类结果的好坏呢？这便是**性能度量**，性能度量为评价聚类结果的好坏提供了一系列有效性指标。

##**10.1 距离度量**

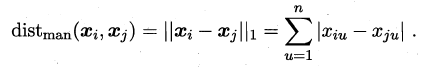
谈及距离度量，最熟悉的莫过于欧式距离了，从年头一直用到年尾的距离计算公式：即对应属性之间相减的平方和再开根号。度量距离还有其它的很多经典方法，通常它们需要满足一些基本性质：

[](https://camo.githubusercontent.com/6f86aa79547b1ed1a0a3a047990808212de4182d/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643463303339302e706e67)

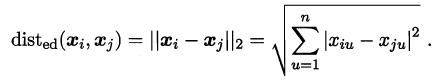
最常用的距离度量方法是\*\*“闵可夫斯基距离”（Minkowski distance)\*\*：

[](https://camo.githubusercontent.com/85a5567ba2d17dce3d4c056cdd4c26908adce66b/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643439653331662e706e67)

当p=1时，闵可夫斯基距离即**曼哈顿距离（Manhattan distance）**：

[](https://camo.githubusercontent.com/74351ff33bc682776a2a79955242769f5c51a5f8/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643439633331662e706e67)

当p=2时，闵可夫斯基距离即**欧氏距离（Euclidean distance）**：

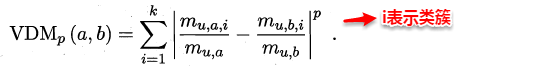
[](https://camo.githubusercontent.com/19fc27bb1b077ba2b18af1ad0cac627781c4e8d1/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643439373631332e706e67)

我们知道属性分为两种：**连续属性**和**离散属性**（有限个取值）。对于连续值的属性，一般都可以被学习器所用，有时会根据具体的情形作相应的预处理，例如：归一化等；而对于离散值的属性，需要作下面进一步的处理：

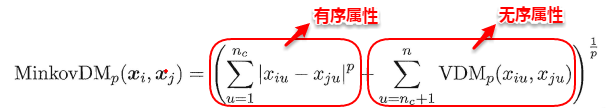
若属性值之间**存在序关系**，则可以将其转化为连续值，例如：身高属性“高”“中等”“矮”，可转化为{1, 0.5, 0}。 若属性值之间**不存在序关系**，则通常将其转化为向量的形式，例如：性别属性“男”“女”，可转化为{（1,0），（0,1）}。

在进行距离度量时，易知**连续属性和存在序关系的离散属性都可以直接参与计算**，因为它们都可以反映一种程度，我们称其为“**有序属性**”；而对于不存在序关系的离散属性，我们称其为：“**无序属性**”，显然无序属性再使用闵可夫斯基距离就行不通了。

**对于无序属性，我们一般采用VDM进行距离的计算**，例如：对于离散属性的两个取值a和b，定义：

[](https://camo.githubusercontent.com/3ec2fdab264b9332741674750f0b45e842f2cc03/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643465393536302e706e67)

于是，在计算两个样本之间的距离时，我们可以将闵可夫斯基距离和VDM混合在一起进行计算：

[](https://camo.githubusercontent.com/69576adc1560098087fd37da8d4b16f6a4f05e56/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643530376263372e706e67)

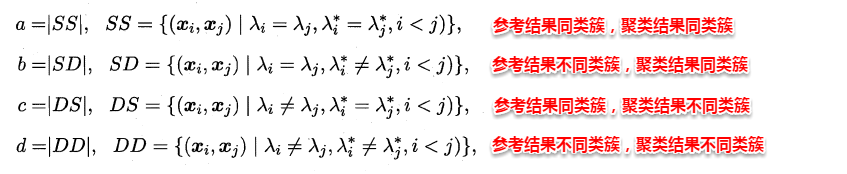
若我们定义的距离计算方法是用来度量相似性，例如下面将要讨论的聚类问题，即距离越小，相似性越大，反之距离越大，相似性越小。这时距离的度量方法并不一定需要满足前面所说的四个基本性质，这样的方法称为：**非度量距离（non-metric distance）**。

##**10.2 性能度量**

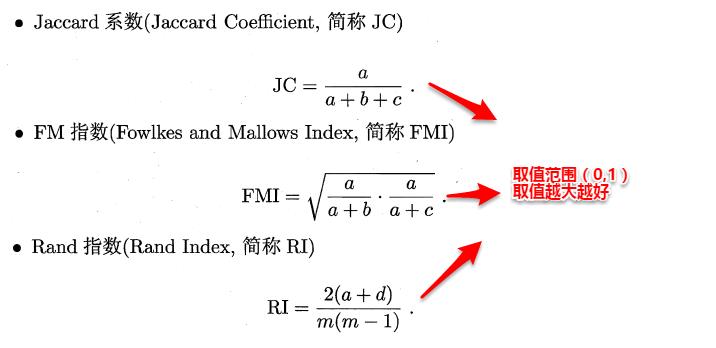
由于聚类算法不依赖于样本的真实类标，就不能像监督学习的分类那般，通过计算分对分错（即精确度或错误率）来评价学习器的好坏或作为学习过程中的优化目标。一般聚类有两类性能度量指标：**外部指标**和**内部指标**。

###**10.2.1 外部指标**

即将聚类结果与某个参考模型的结果进行比较，**以参考模型的输出作为标准，来评价聚类好坏**。假设聚类给出的结果为λ，参考模型给出的结果是λ\*，则我们将样本进行两两配对，定义：

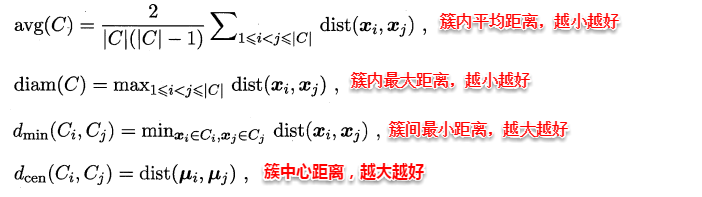
[](https://camo.githubusercontent.com/bf6ceedabce8df0a30439fa2608ae32383ac69c4/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643539313630652e706e67)

显然a和b代表着聚类结果好坏的正能量，b和c则表示参考结果和聚类结果相矛盾，基于这四个值可以导出以下常用的外部评价指标：

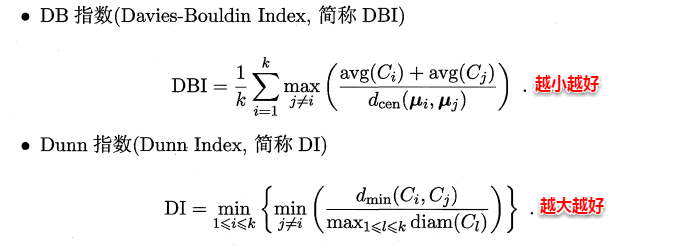
[](https://camo.githubusercontent.com/1339bee08ca6fd266ecb56f37aa6a0e53d445776/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643538373433382e706e67)

###**10.2.2 内部指标**

内部指标即不依赖任何外部模型，直接对聚类的结果进行评估，聚类的目的是想将那些相似的样本尽可能聚在一起，不相似的样本尽可能分开，直观来说：**簇内高内聚紧紧抱团，簇间低耦合老死不相往来**。定义：

[](https://camo.githubusercontent.com/7cc2d5d2acbbb24544194e200a71d642f901cca8/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643538313835322e706e67)

基于上面的四个距离，可以导出下面这些常用的内部评价指标：

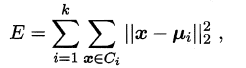
[](https://camo.githubusercontent.com/4c42a95771ff2c47ecc5fc12a4293ea0b1058163/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383465643538323835342e706e67)

##**10.3 原型聚类**

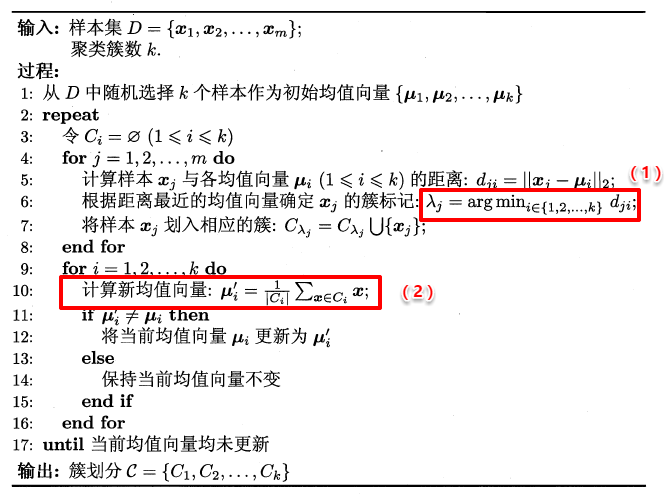
原型聚类即“**基于原型的聚类**”（prototype-based clustering），原型表示模板的意思，就是通过参考一个模板向量或模板分布的方式来完成聚类的过程，常见的K-Means便是基于簇中心来实现聚类，混合高斯聚类则是基于簇分布来实现聚类。

###**10.3.1 K-Means**

K-Means的思想十分简单，**首先随机指定类中心，根据样本与类中心的远近划分类簇，接着重新计算类中心，迭代直至收敛**。但是其中迭代的过程并不是主观地想象得出，事实上，若将样本的类别看做为“隐变量”（latent variable），类中心看作样本的分布参数，这一过程正是通过**EM算法**的两步走策略而计算出，其根本的目的是为了最小化平方误差函数E：

[](https://camo.githubusercontent.com/317726823268e68cbefe7e1544e0fe64f28420c6/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623832623564332e706e67)

K-Means的算法流程如下所示：

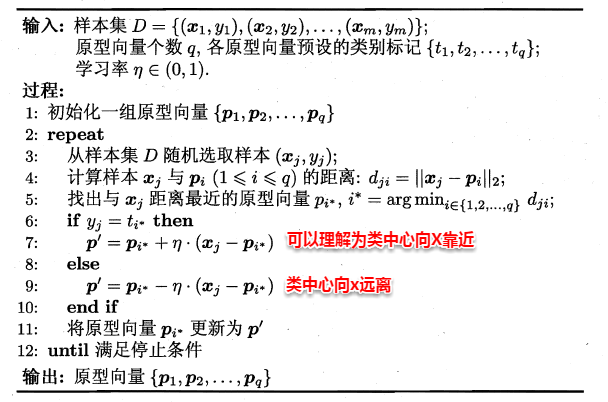
[](https://camo.githubusercontent.com/9fb366fcb8a4e4aa1fe8581382149dc526731a06/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623963303831372e706e67)

###**10.3.2 学习向量量化（LVQ）**

LVQ也是基于原型的聚类算法，与K-Means不同的是，**LVQ使用样本真实类标记辅助聚类**，首先LVQ根据样本的类标记，从各类中分别随机选出一个样本作为该类簇的原型，从而组成了一个**原型特征向量组**，接着从样本集中随机挑选一个样本，计算其与原型向量组中每个向量的距离，并选取距离最小的原型向量所在的类簇作为它的划分结果，再与真实类标比较。

**若划分结果正确，则对应原型向量向这个样本靠近一些** **若划分结果不正确，则对应原型向量向这个样本远离一些**

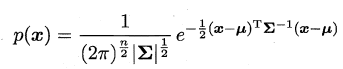
LVQ算法的流程如下所示：

[](https://camo.githubusercontent.com/60be1e5c41288fd45f9d7b67939eee9cdef27d98/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623964353966322e706e67)

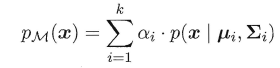
###**10.3.3 高斯混合聚类**

现在可以看出K-Means与LVQ都试图以类中心作为原型指导聚类，高斯混合聚类则采用高斯分布来描述原型。现假设**每个类簇中的样本都服从一个多维高斯分布，那么空间中的样本可以看作由k个多维高斯分布混合而成**。

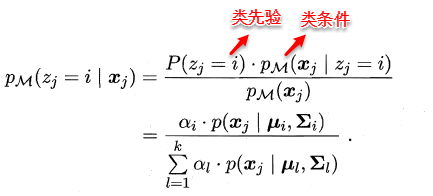
对于多维高斯分布，其概率密度函数如下所示：

[](https://camo.githubusercontent.com/3363d02e7d90de42fae23c73dfb24130f97c5dc1/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623837306439382e706e67)

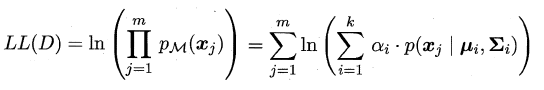
其中u表示均值向量，∑表示协方差矩阵，可以看出一个多维高斯分布完全由这两个参数所确定。接着定义高斯混合分布为：

[](https://camo.githubusercontent.com/7befe8f76f62a488bed57b4ae35d2eb412f5ce11/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623837363739342e706e67)

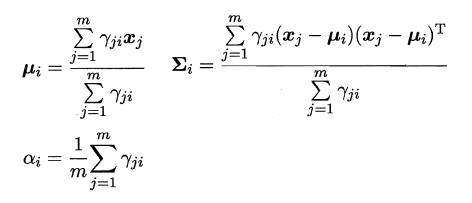
α称为混合系数，这样空间中样本的采集过程则可以抽象为：**（1）先选择一个类簇（高斯分布），（2）再根据对应高斯分布的密度函数进行采样**，这时候贝叶斯公式又能大展身手了：

[](https://camo.githubusercontent.com/4887b43083b2122d798fee46f387e3d6244f0d9d/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623931393164392e706e67)

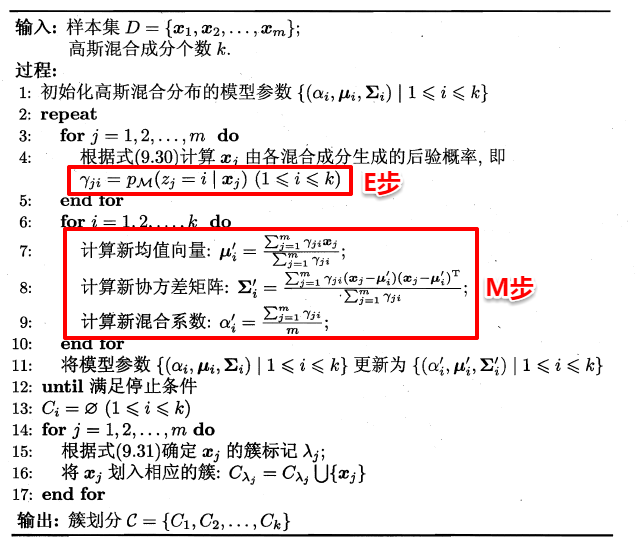
此时只需要选择PM最大时的类簇并将该样本划分到其中，看到这里很容易发现：这和那个传说中的贝叶斯分类不是神似吗，都是通过贝叶斯公式展开，然后计算类先验概率和类条件概率。但遗憾的是：**这里没有真实类标信息，对于类条件概率，并不能像贝叶斯分类那样通过最大似然法美好地计算出来**，因为这里的样本可能属于所有的类簇，这里的似然函数变为：

[](https://camo.githubusercontent.com/ff1d3291abcac95ba5ea6e69dc761be640c44aa6/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623837316434612e706e67)

可以看出：简单的最大似然法根本无法求出所有的参数，这样PM也就没法计算。**这里就要召唤出之前的EM大法，首先对高斯分布的参数及混合系数进行随机初始化，计算出各个PM（即γji，第i个样本属于j类），再最大化似然函数（即LL（D）分别对α、u和∑求偏导 ），对参数进行迭代更新**。

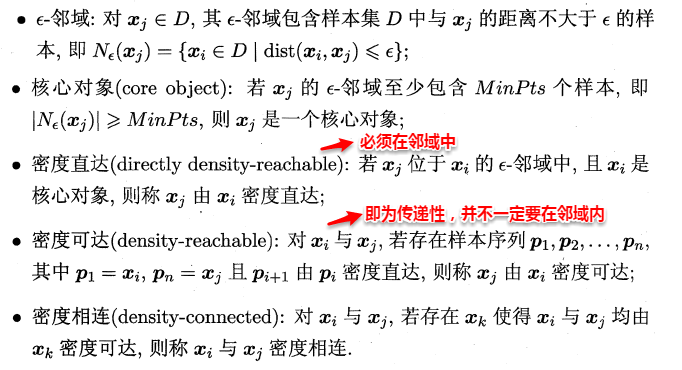
[](https://camo.githubusercontent.com/0b73fba14bb9869a376201d0d4e58bbf986e0d6a/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623861366633322e706e67)

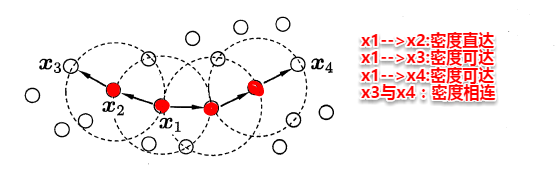
高斯混合聚类的算法流程如下图所示：

[](https://camo.githubusercontent.com/20fe520833e3934843f0ad1227b9e3f286a6902b/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623963346661342e706e67)

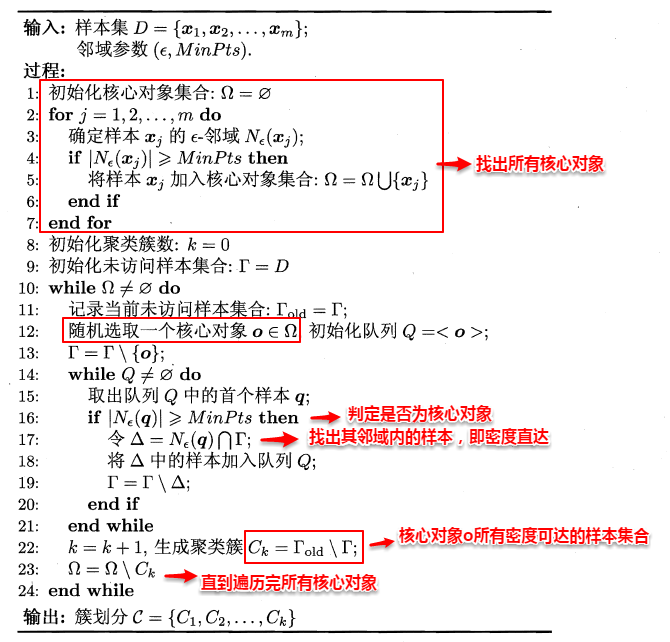
##**10.4 密度聚类**

密度聚类则是基于密度的聚类，它从样本分布的角度来考察样本之间的可连接性，并基于可连接性（密度可达）不断拓展疆域（类簇）。其中最著名的便是**DBSCAN**算法，首先定义以下概念：

[](https://camo.githubusercontent.com/bf17ce05f6397d12816583f06f118bf35f93bfec/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383466623962643639632e706e67)

[](https://camo.githubusercontent.com/a7d694efabb6689ba6393ede044fbe0d7ea7f9c4/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383530396638643631392e706e67)

简单来理解DBSCAN便是：**找出一个核心对象所有密度可达的样本集合形成簇**。首先从数据集中任选一个核心对象A，找出所有A密度可达的样本集合，将这些样本形成一个密度相连的类簇，直到所有的核心对象都遍历完。DBSCAN算法的流程如下图所示：

[](https://camo.githubusercontent.com/25dad8baa1028c1a8314eb7a0d50c3705da797b7/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383530396665623538372e706e67)

##**10.5 层次聚类**

层次聚类是一种基于树形结构的聚类方法，常用的是**自底向上**的结合策略（**AGNES算法**）。假设有N个待聚类的样本，其基本步骤是：

1.初始化-->把每个样本归为一类，计算每两个类之间的距离，也就是样本与样本之间的相似度； 2.寻找各个类之间最近的两个类，把他们归为一类（这样类的总数就少了一个）； 3.重新计算新生成的这个**类与各个旧类之间的相似度**； 4.重复2和3直到所有样本点都归为一类，结束。

可以看出其中最关键的一步就是**计算两个类簇的相似度**，这里有多种度量方法：

\* 单链接（single-linkage）:取类间最小距离。

[23.png](https://camo.githubusercontent.com/49e67b3c42b2fbff0c36dab3445c2acf55f6b286/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383530396562623032322e706e67)

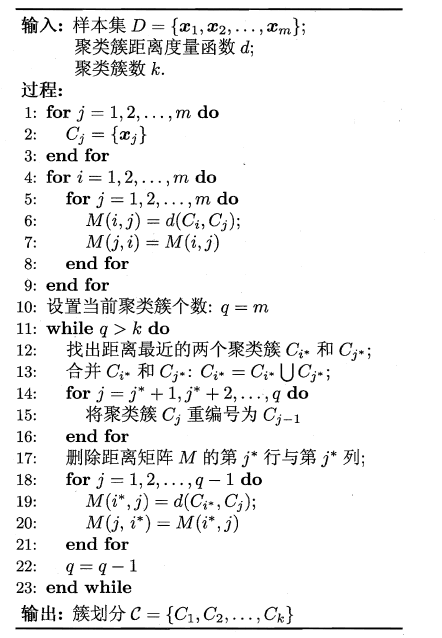
\* 全链接（complete-linkage）:取类间最大距离

[24.png](https://camo.githubusercontent.com/5c1dd4f002a5b45d041a4fbab591320720e4db43/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383530396562326233302e706e67)

\* 均链接（average-linkage）:取类间两两的平均距离

[25.png](https://camo.githubusercontent.com/30cab5dffd9ec6578090907008074f4ad9ed5a58/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383530396630383961372e706e67)

很容易看出：**单链接的包容性极强，稍微有点暧昧就当做是自己人了，全链接则是坚持到底，只要存在缺点就坚决不合并，均连接则是从全局出发顾全大局**。层次聚类法的算法流程如下所示：

[](https://camo.githubusercontent.com/c19548f8e2c8d896454e750da436432cf260e34b/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31382f356263383530396639643461302e706e67)

在此聚类算法就介绍完毕，分类/聚类都是机器学习中最常见的任务，我实验室的大Boss也是靠着聚类起家，从此走上人生事业钱途...之巅峰，在书最后的阅读材料还看见Boss的名字，所以这章也是必读不可了...