第 18 章　聚类算法

聚类[1]属于无监督学习问题，其目标是将样本集划分成多个类，保证同一类的样本之间尽量相似，不同类的样本之间尽量不同，这些类称为簇（cluster）。与有监督的分类算法不同，聚类算法没有训练过程，直接完成对一组样本的划分。与有监督学习算法相比，无监督学习算法的研究进展更为缓慢，但在很多实际问题中得到了成功的应用。

18.1　问题定义

聚类也是分类问题，它的目标也是确定每个样本所属的类别。和有监督的分类算法不同，这里的类别不是人工预定好的，而由聚类算法确定。假设有一个样本集：

$$C = \left\{ \mathbf{x}_1, ..., \mathbf{x}_l \right\}$$

聚类算法把这个样本集划分成 m 个不相交的子集 $C_1, ..., C_m$。这些子集的并集是整个样本集：

$$C_1 \cup C_2 ... \cup C_m = C$$

每个样本只能属于这些子集中的一个，即任意两个子集之间没有交集：

$$C_i \cap C_j = \phi, \forall i, j, i \neq j$$

其中 m 的值可以由人工设定，也可以由算法确定。下面用一个实际的例子来说明聚类任务。假设有一堆水果，我们事先并不知道有几类水果，聚类算法要完成对这堆水果的归类，而且要在没有人工的指导下完成。

聚类本质上是集合划分问题。因为没有人工定义的类别标准，因此要解决的核心问题是如何定义簇。通常的做法是根据簇内样本间的距离，样本点在数据空间中的密度来确定。对簇的不同定义导致了各种不同的聚类算法。常见的聚类算法有以下几种：

连通性聚类。典型的代表是层次聚类算法，它根据样本之间的联通性（相似度）来构造簇，所有联通的样本属于同一个簇。

基于质心的聚类。典型的代表是 k 均值算法，它用类中心向量来表示一个簇，样本所属的簇由它到每个簇的中心向量的距离确定。

基于概率分布的聚类。这种算法假设每种类型的样本服从某一概率分布，如多维正态分布，典型的代表是 EM 算法。

基于密度的聚类。典型代表是 DBSCAN 算法，OPTICS 算法，以及均值漂移（Mean shift）算法，它们将簇定义为空间中样本密集的区域。

基于图的算法。这类算法用样本点构造出带权重的无向图，每个样本是图中的一个顶点，然后使用图论中的方法完成聚类，对应于图切割问题，即将图切分成多个子图。

18.2　层次聚类

对于有些问题，类型的划分具有层次结构。如水果分为苹果，杏，梨等，苹果又可以细分成黄元帅、红富士、蛇果等很多品种，杏和梨也是如此。将这种谱系关系画出来，是一棵分层的树。层次聚类[1]使用了这种做法，它反复将样本进行合并，形成一种层次的表示。

初始时每个样本各为一簇，然后开始反复合并的过程。计算任意两个簇之间的距离，并将聚类最小的两个簇合并。下图 18.1 是对一堆水果进行层次聚类的示意图：

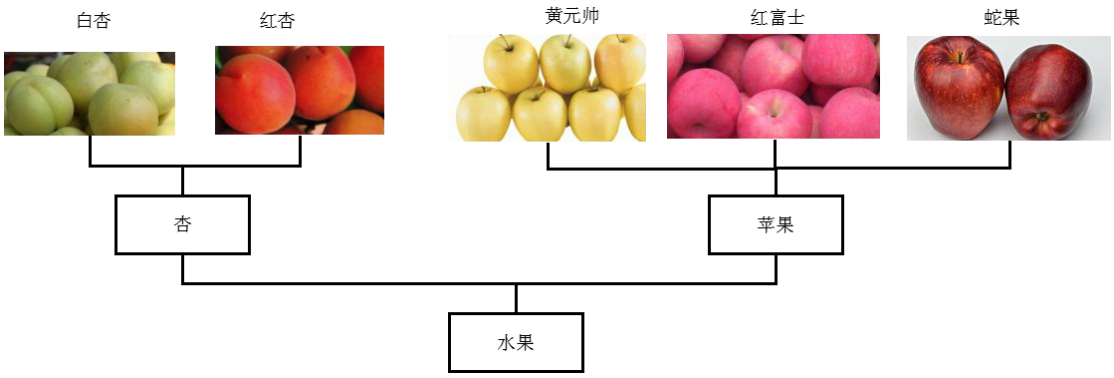

图 18.1　对水果进行层次聚类的结果

算法依赖于两个簇的距离值，因此需要定义它的计算公式。常用的方案可有 3 种。第一种方案是使用两个簇中任意两个样本之间的距离的最大值，第二种方案是使用两个簇中任意两个样本之间的距离的最小值，第三种方案是使用两个簇中所有样本之间距离的均值。

18.3　基于质心的算法

基于质心的算法计算每个簇的中心向量，以此为依据来确定每个样本所属的类别，典型的代表是 k 均值算法。

18.3.1　k 均值算法

k 均值算法[2]是一种被广泛用于实际问题的聚类算法。它将样本划分成 k 个类，参数 k 由人工设定。算法将每个样本划分到离它最近的那个类中心所代表的类，而类中心的确定又依赖于样本的划分方案。假设样本集有 l 个样本，特征向量 x_i 为 n 维向量，给定参数 k 的值，算法将这些样本划分成 k 个集合：

$$\mathrm{S}=\{S_1,...,S_k\}$$

最优分配方案是如下最优化问题的解：

$$\min_{\mathrm{S}}\sum_{i=1}^{k}\sum_{\mathrm{x}\in S_i}\|\mathrm{x}-\mu_i\|^2$$

其中 μ_i 为第 i 个类的中心向量，S_i 为第 i 个类的样本（样本下标）集合。本质上这是子集划分问题，所有可能的划分方案随着样本数呈指数级增长，因此这个问题是 NP 难问题，不易求得全局最优解，只能近似求解。实现时采用迭代法，只能保证收敛的局部最优解处。

算法的流程如下：

初始化 k 个类的中心向量 $\mu_1,...,\mu_k$

循环，直到收敛

分配阶段。根据当前的类中心估计值确定每个样本所属的类：

循环，对每个样本 x_i

计算样本离每个类中心 μ_j 的距离：

$$d_{ij} = \left\| \mathrm{x}_i - \mu_j \right\|$$

将样本分配到距离最近的那个类

结束循环

更新阶段。更新每个类的类中心：

循环，对每个类

根据上一步的分配方案更新每个类的中心：

$$\mu_i = \sum_{j=1, y_j=i}^{l} \mathrm{x}_j / N_i$$

结束循环

结束循环

与 k 近邻算法一样，这里也依赖于样本之间的距离，因此需要定义距离的计算方式，最常用的是欧氏距离，也可以采用其他距离定义，这在第 6 章已经介绍。k 均值算法倾向于得到形状规则（一般偏向圆形）的簇，对复杂的形状聚类效果不好。

算法在实现时要考虑下面几个问题：

1.类中心向量的初始值。一般采用随机初始化[19][20]。最简单的是 Forgy 算法，它从样本集中随机选择 k 个样本作为每个类的初始类中心。第二种方案是随机划分，它将所有样本随机的分配给 k 个类中的一个，然后按照这种分配方案计算各个类的中心向量。

2.参数 k 的设定。可以根据先验知识人工指定一个值，或者由算法自己确定[11][12]。

3.迭代终止的判定规则。一般做法是计算本次迭代后的类中心和上一次迭代时的类中心之间的距离，如果小于指定阈值，则算法终止。

k 均值算法有多种改进版本，包括模糊 c 均值聚类[6]，用三角不等式加速[7]，感兴趣的读者可以进一步阅读这些参考文献。

18.4 基于概率分布的算法

基于概率分布的聚类算法假设每个簇的样本服从相同的概率分布，这是一种生成模型。经常使用的是多维正态分布，如果服从这种分布，则为高斯混合模型，求解时一般采用 EM 算法。

18.4.1 高斯混合模型

高斯混合模型（Gaussian Mixture Model，简称 GMM）通过多个正态分布（高斯分布）的加权和来描述一个随机变量的概率分布，概率密度函数定义为

$$p(\mathbf{x})=\sum_{i=1}^{k} w_i N_i(\mathbf{x};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$$

其中 $\mathbf{x}$ 为随机向量，k 为高斯分布的数量，w_i 为高斯分布的权重，是一个正数，$\boldsymbol{\mu}$ 为高斯分布的均值向量，$\boldsymbol{\Sigma}$ 为协方差矩阵。所有高斯分布的权重之和为 1，即

$$\sum_{i=1}^{k} w_i = 1$$

任意一个样本可以看作是这样产生的：先以 w_i 的概率从 k 个高斯分布中选择出一个高斯分布，再由这个高斯分布 $N(\mathbf{x},\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$ 产生出样本数据 $\mathbf{x}$。高斯混合模型可以逼近任何一个连续的概率分布，因此它可以看做是连续型概率分布的万能逼近器。之所有要保证权重的和为 1，是因为概率密度函数必须满足在 $(-\infty,+\infty)$ 内的积分值为 1。

模型的参数为所有高斯分量的权重 w_i，均值向量 $\boldsymbol{\mu}_i$，协方差矩阵 $\boldsymbol{\Sigma}_i$。训练时采用最大似然估计，对数似然函数为

$$\sum_{i=1}^{l} \ln\left(\sum_{j=1}^{k} w_j N_j(\mathbf{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)\right)$$

由于对数函数中有 k 个求和项，以及参数 w_j 的存在，无法像单个高斯模型那样求得公式解。如果用梯度下降法或牛顿法求解，因为 w_i 要满足等式和不等式约束，也存在困难。

从另外一个角度看，如果知道每个样本属于哪个高斯分布，问题也会变简单，就是分别估计 k 个高斯分布的参数，标准的最大似然估计即可完成，得到均值和协方差的公式解。但每个样本属于哪个高斯分布是未知的，而计算高斯分布的参数时需要用到这个信息；反过来，样本属于哪个高斯分布又是由高斯分布的参数确定的。因此存在循环依赖，解决此问题的办法是打破此循环依赖，从所有高斯分布的一个不准确的初始猜测值开始，计算样本属于每个高斯分布的概率，然后又根据这个概率更新每个高斯分布的参数，EM 算法求解时采用了这种思路。

18.4.2 EM 算法

EM 算法[8]即期望最大化算法，是一种迭代法，它同时估计出每个样本所属的簇以及每个簇的概率分布的参数。如果要聚类的样本数据服从它所属的簇的概率分布，则可以通过估计每个簇的概率分布以及每个样本所属的簇来完成聚类。估计每个簇概率分布的参数需要知道哪些样本属于这个簇，而确定每个样本属于哪个簇又需要知道每个簇的概率分布的参数，这存在循环依赖。EM 算法在每次迭代时交替的解决上面的两个问题，直至收敛到局部最优解。

在介绍算法之前首先介绍 Jensen 不等式，后面的推导会用到它。假设 $f(\mathbf{x})$ 是凸函数，$\mathbf{x}$ 是随机变量，则下面的不等式成立：

$$\mathrm{E}\left(f\left(\mathbf{x}\right)\right)\geq f\left(\mathrm{E}\left(\mathbf{x}\right)\right)$$

如果 $f\left(\mathbf{x}\right)$ 是一个严格凸函数，当且仅当 $\mathbf{x}$ 是常数时不等式取等号：

$$\mathrm{E}\left(f\left(\mathbf{x}\right)\right)=f\left(\mathrm{E}\left(\mathbf{x}\right)\right)$$

EM 算法是一种迭代法，其目标是求解似然函数或后验概率的极值，而样本中具有无法观测的隐含变量。例如有一批样本分属于 3 个类，每个类都服从正态分布，均值和协方差未知，并且每个样本属于哪个类也是未知的，需要在这种情况下估计出每个正态分布的均值和协方差。下图 18.2 是一个例子，3 类样本都服从正态分布，但每个样本属于哪个类是未知的：

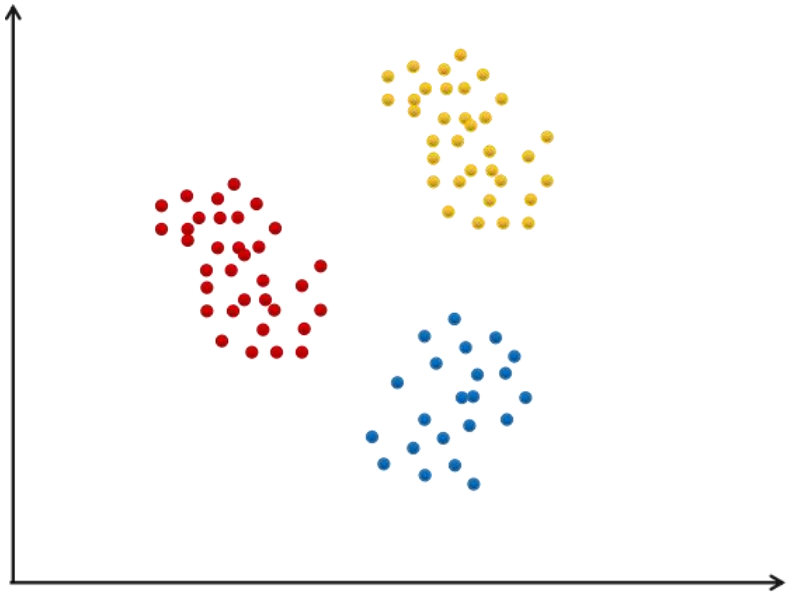

图 18.2　三类正态分布样本，每个样本所属类别未知

样本所属的类别就是隐变量，这种隐变量的存在导致了用最大似然估计求解时的困难。

假设有一个概率分布 $p\left(\mathbf{x};\boldsymbol{\theta}\right)$，从它生成了 l 个样本。每个样本包含观测数据 $\mathbf{x}_i$，以及无法观测到的隐变量 z_i，这个概率分布的参数 $\boldsymbol{\theta}$ 是未知的，现在需要根据这些样本估计出参数 $\boldsymbol{\theta}$ 的值。如果用最大似然估计，可以构造出对数似然函数：

$$\begin{aligned}L\left(\boldsymbol{\theta}\right)&=\sum_{i=1}^{l}\ln p\left(\mathbf{x}_i;\boldsymbol{\theta}\right)\\&=\sum_{i=1}^{l}\ln\sum_{z}p\left(\mathbf{x}_i,z_i;\boldsymbol{\theta}\right)\end{aligned}$$

这里的 z_i 是一个无法观测到（即我们不知道它的值）的隐含变量，是离散型随机变量，上式是对隐含变量 z 的所有情况下的联合概率 $p\left(\mathbf{x},z;\boldsymbol{\theta}\right)$ 求和得到 $\mathbf{x}$ 的边缘概率。因为隐含变量的存在，我们无法直接通过最大化似然函数得到参数的公式解。如果使用梯度下降法或牛顿法求解，则要保证隐变量所满足的等式和不等式约束

$$\sum_z p(z) = 1$$
$$p(z) \geq 0$$

这同样存在困难。可以采用一种策略，构造出对数似然函数的一个下界函数，这个下界函数更容易优化，然后优化这个下界。不断的改变优化变量的值使得下界函数的值升高，从而使得对数似然函数的值也升高，这就是 EM 算法所采用的思路。

对每个样本 i，假设 Q_i 为变量 z_i 的一个概率分布，根据对概率分布的要求它必须满足：

$$\sum_z Q_i(z) = 1$$
$$Q_i(z) \geq 0$$

利用这个概率分布，将对数似然函数变形，可以得到：

$$\begin{aligned}\sum_{i=1}^{l} \ln p(\mathbf{x}_i;\boldsymbol{\theta}) &= \sum_{i=1}^{l} \ln \sum_{z_i} p(\mathbf{x}_i, z_i;\boldsymbol{\theta}) \\ &= \sum_{i=1}^{l} \ln \sum_{z_i} Q_i(z_i) \frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)} \\ &\geq \sum_{i=1}^{l} \sum_{z_i} Q_i(z_i) \ln \frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)}\end{aligned}$$

上式第二步凑出了数学期望的形式，最后一步利用了 Jensen 不等式。令：

$$f(x) = \ln x$$

按照数学期望的定义（注意，在这里 z_i 是随机变量），有：

$$\begin{aligned}&\ln \sum_{z_i} Q_i(z_i) \frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)} = \\ &f\left(\mathrm{E}_{Q_i(z_i)}\left(\frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)}\right)\right) = \ln\left(\mathrm{E}_{Q_i(z_i)}\left(\frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)}\right)\right) \\ &\geq \mathrm{E}_{Q_i(z_i)} f\left(\frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)}\right) = \mathrm{E}_{Q_i(z_i)} \ln\left(\frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)}\right) \\ &= \sum_{z_i} Q_i(z_i) \ln \frac{p(\mathbf{x}_i, z_i;\boldsymbol{\theta})}{Q_i(z_i)}\end{aligned}$$

对数函数是凹函数，因此不等式成立，Jensen 不等式反号。上式给出了对数似然函数的一个下界，Q_i 可以是任意一个概率分布，因此可以利用参数 $\boldsymbol{\theta}$ 的当前估计值来构造 Q_i。显然，这个下界函数更容易求极值，因为对数函数里面已经没有求和项。

算法在实现时首先随机初始化参数 $\boldsymbol{\theta}$ 的值，接下来循环迭代，每次迭代时分为两步：

E 步，基于当前的参数估计值 $\boldsymbol{\theta}_t$，计算在给定 $\mathbf{x}$ 时对 Z 的条件概率：

$$Q_i(z_i) = p(z_i|\mathrm{x}_i;\boldsymbol{\theta}_t)$$

接下来根据该概率论构造目标函数（下界函数），它是对 Z 的数学期望。

M 步，求解如下极值问题，更新参数 $\boldsymbol{\theta}$ 的值：

$$\boldsymbol{\theta}_{t+1} = \arg\max_{\boldsymbol{\theta}} \sum_i \sum_{z_i} Q_i(z_i) \ln \frac{p(\mathrm{x}_i, z_i; \boldsymbol{\theta})}{Q_i(z_i)}$$

上面的目标函数中对数内部没有求和项，更容易求得 $\boldsymbol{\theta}$ 的公式解。由于 Q_i 可以是任意个概率分布，实现时 Q_i 可以按照下面的公式计算：

$$Q_i(z_i) = \frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta})}{\sum_z p(\mathbf{x}_i, z; \boldsymbol{\theta})}$$

迭代终止的判定规则是相邻两次函数值之差小于指定阈值。下面给出算法收敛性的证明。假设第 t 次迭代时的参数值为 $\boldsymbol{\theta}_t$，第 $t+1$ 次迭代时的参数值为 $\boldsymbol{\theta}_{t+1}$。如果能证明每次迭代时对数似然函数的值单调增，即：

$$L(\boldsymbol{\theta}_t) \le L(\boldsymbol{\theta}_{t+1})$$

则算法能收敛到局部极值点。由于在迭代时选择了：

$$Q_{it}(z_i) = p(z_i|\mathbf{x}_i;\boldsymbol{\theta}_t)$$

因此有：

$$\frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta})}{Q_i(z_i)} = \frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta})}{p(z_i|\mathbf{x}_i;\boldsymbol{\theta}_t)} = \frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta})}{p(\mathbf{x}_i, z_i; \boldsymbol{\theta}) / p(\mathbf{x}_i; \boldsymbol{\theta})} = p(\mathbf{x}_i; \boldsymbol{\theta})$$

这和 z_i 无关，因此是一个常数，从而保证 Jensen 不等式可以取等号。因此有下面的等式成立：

$$L(\boldsymbol{\theta}_t) = \sum_i \ln \sum_{z_i} Q_{it}(z_i) \frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta}_t)}{Q_{it}(z_i)} = \sum_i \sum_{z_i} Q_{it}(z_i) \ln \frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta}_t)}{Q_{it}(z_i)}$$

从而有：

$$\begin{aligned} L(\boldsymbol{\theta}_{t+1}) &\ge \sum_i \sum_{z_i} Q_{it}(z_i) \ln \frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta}_{t+1})}{Q_{it}(z_i)} \\ &\ge \sum_i \sum_{z_i} Q_{it}(z_i) \ln \frac{p(\mathbf{x}_i, z_i; \boldsymbol{\theta}_t)}{Q_{it}(z_i)} \\ &= L(\boldsymbol{\theta}_t) \end{aligned}$$

上式第一步利用了 Jensen 不等式，第二步成立是因为 $\boldsymbol{\theta}_{t+1}$ 是函数的极值，因此会大于等于任意点处的函数值，第三步在上面已经做了说明，是 Jensen 不等式取等式。上面的结论保证了每次迭代时函数值会上升，直到到达局部极大值点处，但只能保证收敛到局部极值。

在每次循环时首先计算对隐变量的数学期望（下界函数），然后将该期望最大化，这就是期望最大化算法这一名称的来历。下图 18.3 直观的解释了 EM 算法的原理

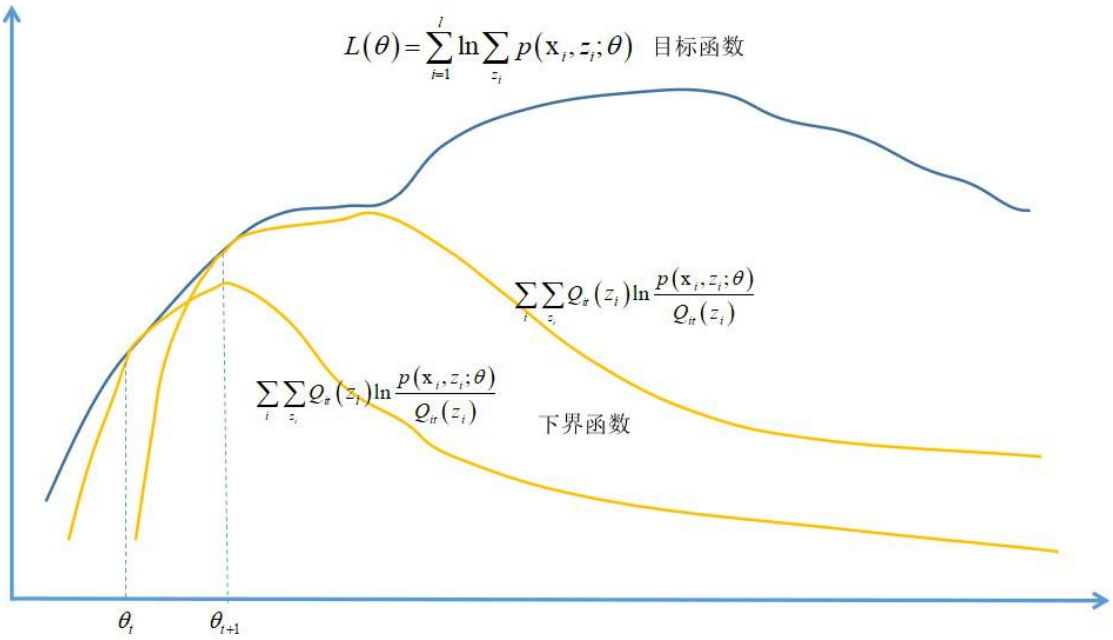

图 18.3 EM 算法原理示意图

图中的蓝色曲线为要求解的对数似然函数，黄色曲线为构造出的下界函数。首先用参数的估计值 $\boldsymbol{\theta}_t$ 计算出每个训练样本的隐变量的概率分布估计值 Q_t，然后用该值构造下界函数，在参数的当前估计值 $\boldsymbol{\theta}_t$ 处，下界函数与对数似然函数的值相等（对应图中左侧第一条虚线）。然后求下界函数的极大值，得到参数新的估计值 $\boldsymbol{\theta}_{t+1}$，再以当前的参数值 $\boldsymbol{\theta}_{t+1}$ 计算隐变量的概率分布 Q_{t+1}，构造出新的下界函数，然后求下界函数的极大值得到 $\boldsymbol{\theta}_{t+2}$。如此反复，直到收敛。

EM 算法的精髓在于：

构造下界函数（Jensen 不等式成立），通过巧妙的取 Q 的值而保证在参数的当前迭代点处下界函数与要求解的目标函数值相等（Jensen 不等式取等号），从而保证优化下界函数后在新的迭代点处目标函数值是上升的。

下面介绍 EM 算法在高斯混合模型中的使用。假设有一批样本 $\{\mathbf{x}_1,...,\mathbf{x}_l\}$。为每个样本 $\mathbf{x}_i$ 增加一个隐变量 z_i，表示样本来自于哪个高斯分布。这是一个离散型的随机变量，取值范围为 $\{1,...,k\}$，取每个值的概率为 w_i。 $\mathbf{x}$ 和 z 的联合概率可以写成

$$\begin{aligned}p(\mathbf{x},z=j)&=p(z=j)p(\mathbf{x}|z=j)\\&=w_jN(\mathbf{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)\end{aligned}$$

这是样本的隐变量取值为 j，并且样本向量值为 $\mathbf{x}$ 的概率。在 E 步构造 Q 函数

$$Q_i(z_i=j)=q_{ij}=\frac{p(\mathbf{x}_i,z_i=j;\theta)}{\sum_z p(\mathbf{x}_i,z;\theta)}$$
$$=\frac{w_j N(\mathbf{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{\sum_{t=1}^{k} w_t N(\mathbf{x};\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t)}$$

这个值根据 $\boldsymbol{\mu},\boldsymbol{\Sigma},\mathbf{w}$ 的当前迭代值计算，是一个常数。得到 z 的分布即 Q 值之后，要求解的目标函数为

$$L(\mathbf{w},\boldsymbol{\mu},\boldsymbol{\Sigma})=\sum_i\sum_{z_i} Q_i(z_i)\ln\frac{p(\mathbf{x}_i,z_i;\boldsymbol{\theta})}{Q_i(z_i)}$$
$$=\sum_{i=1}^{l}\sum_{j=1}^{k} q_{ij}\ln\frac{w_j N(\mathbf{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{q_{ij}}$$
$$=\sum_{i=1}^{l}\sum_{j=1}^{k} q_{ij}\ln\frac{w_j\frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_j|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)}{q_{ij}}$$
$$=\sum_{i=1}^{l}\sum_{j=1}^{k} q_{ij}\left(\ln\frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_j|^{1/2}q_{ij}}+\ln w_j-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)$$

在这里 q_{ij} 已经是一个常数而不是 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$ 的函数。对 $\boldsymbol{\mu}_j$ 求梯度并令梯度为 $\mathbf{0}$，可以得到

$$\nabla_{\boldsymbol{\mu}_j}L(\mathbf{w},\boldsymbol{\mu},\boldsymbol{\Sigma})=\nabla_{\boldsymbol{\mu}_j}\sum_{i=1}^{l}\sum_{j=1}^{k} q_{ij}\left(\ln\frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_j|^{1/2}q_{ij}}+\ln w_j-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)$$
$$=-\sum_{i=1}^{l} q_{ij}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)=\mathbf{0}$$

可以解得

$$\boldsymbol{\mu}_j=\frac{\sum_{i=1}^{l} q_{ij}\mathbf{x}_i}{\sum_{i=1}^{l} q_{ij}}$$

对 $\boldsymbol{\Sigma}_j$ 求梯度并令梯度为 $\mathbf{0}$，根据正态分布最大似然估计的结论，可以解得

$$\boldsymbol{\Sigma}_j=\frac{\sum_{i=1}^{l} q_{ij}(\mathbf{x}_i-\boldsymbol{\mu}_j)(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}}{\sum_{i=1}^{l} q_{ij}}$$

最后处理 $\mathbf{w}$。上面的目标函数中，只有 $\ln w_j$ 和 $\mathbf{w}$ 有关，因此可以简化。由于 w_i 有等式约束 $\sum_{i=1}^{k} w_i = 1$，因此构造拉格朗日乘子函数

$$L(\mathbf{w},\lambda) = \sum_{i=1}^{l}\sum_{j=1}^{k} q_{ij} \ln w_j + \lambda\left(\sum_{j=1}^{k} w_j - 1\right)$$

对 $\mathbf{w}$ 求梯度并令梯度为 $\mathbf{0}$，可以得到下面的方程组

$$\sum_{i=1}^{l}\sum_{j=1}^{k} \frac{q_{ij}}{w_j} + \lambda = 0$$

$$\sum_{i=1}^{k} w_i = 1$$

最后解得

$$w_j = \frac{1}{l}\sum_{i=1}^{l} q_{ij}$$

由此得到求解高斯混合模型的 EM 算法流程。首先初始化 $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w}$，接下来循环进行迭代，直至收敛，每次迭代时的操作为：

E 步，根据模型参数的当前估计值，计算第 i 个样本来自第 j 个高斯分布的概率：

$$q_{ij} = p\left(z_i = j \middle| \mathbf{x}_i; \mathbf{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$$

M 步，计算模型的参数。权重的计算公式为：

$$w_j = \frac{1}{l}\sum_{i=1}^{l} q_{ij}$$

均值的计算公式为：

$$\mu_j = \frac{\sum_{i=1}^{l} q_{ij}\mathbf{x}_i}{\sum_{i=1}^{l} q_{ij}}$$

协方差的计算公式为：

$$\boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^{l} q_{ij}\left(\mathbf{x}_i - \boldsymbol{\mu}_j\right)\left(\mathbf{x}_i - \boldsymbol{\mu}_j\right)^{\mathrm{T}}}{\sum_{i=1}^{l} q_{ij}}$$

18.5 基于密度的算法

基于密度的聚类算法的核心思想是根据样本点某一邻域内的邻居数定义样本空间的密

度，这类算法可以找出空间中形状不规则的簇，并且不用指定簇的数量。算法的核心是计算每一点处的密度值，以及根据密度来定义簇。

18.5.1 DBSCAN 算法

DBSCAN 算法[3]是一种基于密度的算法，对可以有效的处理噪声，发现任意形状的簇。它将簇定义为样本点密集的区域，算法从一个种子样本开始，持续向密集的区域生长，直至到达边界。

算法使用了两个人工设定的参数 ε 和 M 。前者是样本点邻域的半径，后者是定义核心点的样本数阈值，下面介绍它们的概念。

假设有样本集 $X=\left\{\mathrm{x}_1,...,\mathrm{x}_N\right\}$，样本点 x 的 ε 邻域定义为样本集中与该样本的距离小于等于 ε 的样本构成的集合：

$$N_\varepsilon\left(\mathrm{x}\right)=\left\{\mathrm{y}\in X:d\left(\mathrm{x},\mathrm{y}\right)\le\varepsilon\right\}$$

其中 $d\left(\mathrm{x},\mathrm{y}\right)$ 是两个样本之间的距离，可以采用任何一种距离定义。样本的密度定义为它的 ε 邻域的样本数：

$$\rho\left(\mathrm{x}\right)=\left|N_\varepsilon\left(\mathrm{x}\right)\right|$$

密度是一个非负整数。核心点定义为数据集中密度大于指定阈值的样本点，即如果：

$$\rho\left(\mathrm{x}\right)\ge M$$

则称 x 为核心点，核心点是样本分布密集的区域。样本集 X 中所有的核心点构成的集合为 X_c，非核心点构成的集合为 X_{nc}。如果 x 是非核心点，并且它的 ε 邻域内存在核心点，则称 x 为边界点，边界点是密集区域的边界。如果一个点既不是核心点，也不是边界点，则称为噪声点，噪声点是样本稀疏的区域。

如果 x 是核心点，y 在它的 ε 邻域内，则称 y 是从 x 直接密度可达的。对于样本集中的一组样本 $\mathrm{x}_1,...,\mathrm{x}_n$，如果 x_{i+1} 是从 x_i 直接密度可达的，则称 x_n 是从 x_1 密度可达的。密度可达是直接密度可达的推广。

对于样本集中的样本点 x，y 和 z，如果 y 和 z 都从 x 密度可达，则称它们是密度相连的，根据定义，密度相连具有对称性。

基于上面的概念可以给出簇的定义。样本集 C 是整个样本集的一个子集，如果它满足下列条件：对于样本集 X 中的任意两个样本 x 和 y，如果 $\mathrm{x}\in C$，且 y 是从 x 密度可达的，则 $\mathrm{y}\in C$；如果 $\mathrm{x}\in C$，$\mathrm{y}\in C$，则 x 和 y 是密度相连的，则称集合 C 是一个簇。

根据簇的定义可以构造出聚类算法，具体做法是从某一核心点出发，不断向密度可达的区域扩张，得到一个包含核心点和边界点的最大区域，这个区域中任意两点密度相连。

假设有样本集 X，聚类算法将这些样本划分成 K 个簇以及噪声点的集合，其中 K 由算法确定。每个样本要么属于这些簇中的一个，要么是噪声点。定义变量 m_i 为样本 x_i 所属的簇，如果它属于第 j 个簇，则 m_i 的值为 j，如果它不属于这些簇中的任何一个，即是噪声

点，则其值为-1，m_i 就是聚类算法的返回结果。变量 k 表示当前的簇号，每发现一个新的簇，其值加 1。聚类算法的流程如下：

第一阶段，初始化

计算每个样本的邻域 $N_\varepsilon\left(\mathrm{x}_i\right)$

令 $k=1$，$m_i=0$，初始化待处理样本集合 $I=\{1,...,N\}$

第二阶段，生成所有的簇

循环，当 I 不为空

从 I 中取出一个样本 i，并将其从集合中删除

如果 i 没被处理过，即 $m_i=0$

初始化集合 $T=N_\varepsilon\left(\mathrm{x}_i\right)$

如果 i 为非核心点

令 $m_i=-1$，暂时标记为噪声

如果即 i 为核心点

令 $m_i=k$，将当前簇编号赋予该样本

循环，当 T 不为空

从集合 T 中取出一个样本 j，并从该集合中将其删除

如果 $m_j=0$ 或 $m_j=-1$

令 $m_j=k$

如果 j 是核心点

将 j 的邻居集合 $N_\varepsilon\left(\mathrm{x}_j\right)$ 加入集合 T

结束循环

令 $k=k+1$

结束循环

算法的核心步骤是依次处理每一个还未标记的点，如果是核心点，则将其邻居点加入到加入到连接集合中，反复扩张，直到找到一个完整的簇。

在实现时有几个问题需要考虑，第一个问题是如何快速找到一个点的所邻居集合，可以用 R 树或者 KD 树等数据结构加速。第二个问题是参数 ε 和 M 的设定，ε 的取值在有些时候非常难以确定，而它对聚类的结果有很大的影响。M 值的选择有一个指导性的原则，如果样本向量是 n 维的，则 M 的值至少是 $n+1$。

DBSCAN 无需指定簇的数量，可以发现任意形状的簇，并且对噪声不敏感。其缺点是聚类的质量受距离函数的影响很大，如果数据维数很高，将面临维数灾难的问题。参数 ε 和 M 的设定有时候很困难。

18.5.2 OPTICS 算法

OPTICS 算法[4]是对 DBSCAN 算法的改进，对参数更不敏感。它不直接生成簇，而是对样本进行排序，从这个排序可以得到各种邻域半径 ε 和密度阈值 M 时的聚类结果。

OPTICS 算法复用了 DBSCAN 的一些概念，除此之外，还定义了两个新的概念。给定参数 ε 和 M，使得样本 x 成为核心点的最小邻域半径称为 x 的核心距离，即：

$$cd\left(\mathrm{x}\right)=\begin{cases}\text{UNDEFINED} & \left|N_{\varepsilon}\left(\mathrm{x}\right)\right|<M\\ d\left(\mathrm{x},N_{\varepsilon}^{M}\left(\mathrm{x}\right)\right) & \left|N_{\varepsilon}\left(\mathrm{x}\right)\right|\geq M\end{cases}$$

其中 $N_{\varepsilon}^{i}\left(\mathrm{x}\right)$ 为 x 的 ε 邻域内距离它第 i 近的点。按照定义，如果 x 是核心点，则其核心距离小于等于 ε，否则核心距离没有定义。给定样本集中的两个点 x 和 y，y 对于 x 的可达距离定义为：

$$rd\left(\mathrm{y},\mathrm{x}\right)=\begin{cases}\text{UNDEFINED} & \left|N_{\varepsilon}\left(\mathrm{x}\right)\right|<M\\ \max\left(cd\left(\mathrm{x}\right),d\left(\mathrm{x},\mathrm{y}\right)\right) & \left|N_{\varepsilon}\left(\mathrm{x}\right)\right|\geq M\end{cases}$$

如果 x 是核心点，y 对它的可达距离是 x 的核心距离与 y 和 x 之间的距离的最大值，如果不是核心点则该值未定义。这是使得 x 成为核心点，并且 y 从 x 直接密度可达的最小邻域半径。显然，可达距离与参考点 x 有关，不同的 x 将导致不同的计算结果。可达距离和 y 点处的密度有关，密度越大，它从邻居节点直接密度可达的距离越小。聚类时同样向密集的区域扩张，优先考虑可达距离小的样本。

给定样本集 $X=\left\{\mathrm{x}_1,...,\mathrm{x}_N\right\}$，以及人工设定的参数 ε 和 M，OPTICS 算法输出所有样本的一个排序，以及每个样本的核心距离，可达距离。其中第 i 个样本在输出序列中的位置为 p_i，它的核心距离为 c_i，可达距离为 r_i。辅助数组 v_i 表示第 i 个样本是否被处理过，用于算法的实现。算法维持了一个列表 seedList，存储所有待处理的样本，按样本点离它最近直接密度可达的核心点的可达距离升序排列。

算法依次处理每一个没有被处理的点，如果是核心点，则按照可达距离升序的顺序依次扩展到每一个能到达的新的点。OPTICS 算法的流程如下：

第一阶段：初始化

计算每个样本的邻域 $N_{\varepsilon}\left(\mathrm{x}_i\right)$

计算每个样本的核心距离 c_i

将所有样本的处理标志 v_i 初始化为 0

将所有样本的可达距离 r_i 初始化为 UNDEFINED

令 $k=1$，待处理样本的集合初始化为 $I=\left\{1,...,N\right\}$

第二阶段：输出排序

循环，当 I 中还有样本未处理

从 I 中取出一个样本 i，将 i 从 I 中删除

如果 $v_i = 0$，即样本没被处理过

令 $v_i = 1$， $p_k = i$， $k = k + 1$

如果 i 是核心点

调用 $\text{insert}\left(N_\varepsilon\left(\text{x}_i\right), \text{seedlist}\right)$，将 $N_\varepsilon\left(\text{x}_j\right)$ 中未处理点插入列表

循环，当列表 seedList 不为空

从 seedList 中取出第一个样本 j

令 $v_j = 1$， $p_k = j$， $k = k + 1$

如果 j 是核心点

调用 $\text{insert}\left(N_\varepsilon\left(\text{x}_j\right), \text{seedlist}\right)$，将 $N_\varepsilon\left(\text{x}_j\right)$ 中未处理点插入列表

结束循环

结束循环

注意，这里的 ε 只用于生成样本的顺序，真正的聚类使用了另一个邻域半径阈值。函数 insert 将一个样本点邻域集合中的所有未处理点按照可达距离插入到列表中。这里分两种情况，如果之前没有计算过可达距离，则直接按照本次计算的可达距离将样本插入列表，否则取之前的可达距离与本次可达距离的最小值，即使用从最近的那个核心点计算出来的可达距离。算法处理流程如下：

循环，对 $N_\varepsilon\left(\text{x}_k\right)$ 中的所有样本 i

如果 $v_i = 0$

计算 i 对 k 的可达距离 $rd = \max\left(cd_k, d\left(\text{x}_k, \text{x}_i\right)\right)$

如果 r_i 为 UNDEFINED

令 $r_i = rd$

将 i 按照可达距离值插入到 seedList 列表中的适当位置

否则

如果 $rd < r_i$

令 $r_i = rd$

将 i 按照可达距离值插入到 seedList 列表中的适当位置

结束循环

算法返回的序列是按照所有点对各个种子点的可达距离升序排序的。如果将横坐标设为有序样本的编号，纵坐标为可达距离，则寻找每个谷底的位置可以得到聚类结果。这里需要一个人工设定的参数 ε'，并且要保证 $\varepsilon' \le \varepsilon$，这是聚类时使用的最小邻域半径。算法依次处理 OPTICS 算法返回的有序列表中的每个样本，如果其可达距离大于 ε'，则认为是一个新

的簇的开始，因为它不能被加入到之前被处理的那些那边所在的簇中。这里又分两种情况，如何其可达距离小于 ε，则是一个新的簇，否则是噪声。如果可达距离小于 ε'，则把样本加入到已经存在的簇中，因为它和前面的样本是密度可达的。

和 DBSCAN 算法相同，用变量 m_i 标记对每个样本的分配结果。根据排序结果生成聚类结果的算法流程如下：

初始化 $clusertID=-1$， $k=1$

循环，对 $i=1,...,N$，依次处理有序列表中的每个样本

如果 $r_i>\varepsilon'$

如果 $c_i\le\varepsilon'$

$clusterID=k$， $k=k+1$， $m_j=clusterID$

否则

$m_i=-1$

否则

$m_i=clusterID$

结束循环

$clusterID$ 为当前簇号，每生成一个新的簇，其值加 1。算法执行结束之后得到每个样本的簇编号，和 DBSCAN 算法一样，它要么属于某一个簇，要么是噪声。

18.5.3 Mean Shift 算法

均值漂移（Mean Shift）算法[5][23]基于核密度估计技术（也称为 Parzen 窗技术），是一种寻找概率密度函数极值点的算法。它在聚类分析，图像分割，视觉目标跟踪中都有成功的应用。在用于聚类任务时，它寻找概率密度函数的极大值点，即样本分布最密集的位置，以此得到簇，因此是一种基于密度的方法。

对于某些应用，我们不知道概率密度函数的具体形式，但有一组采样自此分布的离散样本数据，核密度估计可以根据这些样本值估计概率密度函数，均值漂移算法可以找到概率密度函数的极大值点。和第 2 章介绍的数值优化算法类似，这也是一种迭代算法，从一个初始点 x 开始，按照某种规则移动到下一点，直到到达极值点处。

假设有 n 个样本点 $\mathbf{x}_i,i=1,...,n$，由核函数 K 与窗口半径 h 定义的核密度估计函数为：

$$p(\mathbf{x})=\frac{1}{nh^d}\sum_{i=1}^{n}K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

其中 d 为向量的维数，前面的常数是为了保证积分值为 1，这是概率密度函数的要求。这里使用了核函数：

$$K\left(\frac{\mathbf{x}_i-\mathbf{x}}{h}\right)$$

常用的是高斯核，其形式与正态分布的概率密度函数相同。如果使用径向对称核，核函数可以写成剖面函数的形式

$$K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)=ck\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$$

其中 c 为常数，k 称为剖面函数，它保证当 $\mathbf{x}$ 远离中心点 $\mathbf{x}_i$ 时函数值单调递减。寻找概率密度函数的极大值点即寻找核密度函数的极大值点，可以采用梯度上升法（和梯度下降法相反，在这里沿着梯度方向迭代），可以证明，其梯度为如下形式：

$$\mathbf{m}=\frac{\sum_{i=1}^{n}g\left(\mathbf{x}_i-\mathbf{x}\right)\mathbf{x}_i}{\sum_{i=1}^{n}g\left(\mathbf{x}_i-\mathbf{x}\right)}-\mathbf{x}$$

其中

$$g\left(x\right)=-k^{'}\left(x\right)$$

证明如下

$$\nabla p\left(\mathbf{x}\right)=\frac{2c}{nh^{d+2}}\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\left(\mathbf{x}_i-\mathbf{x}\right)$$

$$=\frac{2c}{nh^{d+2}}\left(\sum_{i=1}^{n}\left(g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\mathbf{x}_i\right)-\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\mathbf{x}\right)$$

$$=\frac{2c}{nh^{d+2}}\left(\sum_{i=1}^{n}\left(g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\frac{\sum_{j=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_j}{h}\right\|^2\right)}{\sum_{j=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_j}{h}\right\|^2\right)}\mathbf{x}_i\right)-\left(\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right)\mathbf{x}\right)$$

$$=\frac{2c}{nh^{d+2}}\left(\left(\sum_{j=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_j}{h}\right\|^2\right)\right)\left(\sum_{i=1}^{n}\frac{g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{j=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_j}{h}\right\|^2\right)}\mathbf{x}_i\right)-\left(\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right)\mathbf{x}\right)$$

$$=\frac{2c}{nh^{d+2}}\left[\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}-\mathbf{x}\right]$$

在这里 $\frac{2}{nh^{d+2}}\left[\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]$ 是一个标量。$\mathbf{m}$ 为梯度上升法迭代时的梯度方向，称

为均值漂移向量。根据它可以计算出下一次迭代的值

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{m}_t$$

其中 t 为迭代次数。可以证明算法最终会收敛到局部极大值点处。在聚类时，从某一初始点开始，反复用均值漂移算法进行迭代，直到到达密度最大的点处，这就找到了一个簇，聚类的过程类似于 k 均值算法。

18.6 基于图的算法

基于图的算法把样本数据看作图的顶点，根据数据点之间的距离构造边，形成带权重的图。通过图的切割实现聚类，即将图切分成多个子图，这些子图就是对应的簇。这类算法的典型代表是谱聚类算法[21][22]。谱聚类算法首先构造样本集的邻接图（也称为相似度图），得到图的拉普拉斯矩阵，这和第 7.3 节中介绍的方法相同。接下来对矩阵进行特征值分解，通过对特征向量进行处理构造出簇。

算法首先根据样本集构造出带权重的图 G，聚类算法的目标是将其切割成多个子图，每个子图即为聚类后的一个簇。假设图的顶点集合为 V，边的集合为 E。聚类算法将顶点集合切分成 k 个子集，它们的并集是整个顶点集：

$$V_1 \cup V_2 \ldots \cup V_k = V$$

任意两个子集之间的交集为空：

$$V_i \cap V_j = \phi, \forall i, j, i \neq j$$

对于任意两个子图，其顶点集合为 A 和 B，它们之间的切图权重定义为连接两个子图节点的所有边（即跨两个子图的边）的权重之和：

$$W(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

这可以看作两个子图之间的关联程度，如果两个子图之间没有边连接，则该值为 0。从另一个角度看，这是对图进行切割时去掉的边的权重之和。

下图 18.4 为图切割示意图

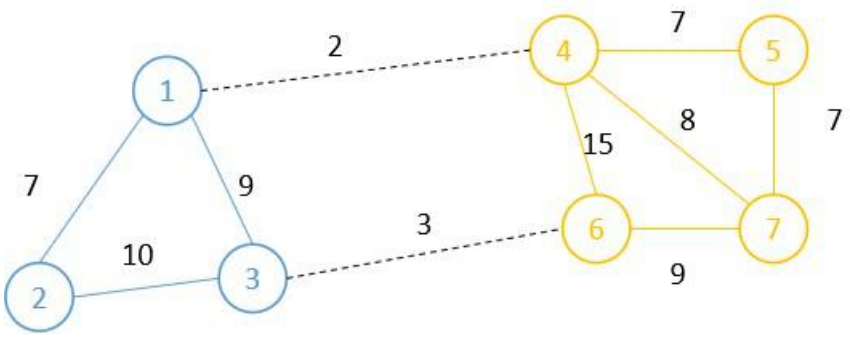

图 18.4　图切割

上图 18.4 中有 7 个顶点，被切割成蓝色和黄色两个子图，虚线边为被切割掉的边，因此切图权重为

$$2 + 3 = 5$$

对图顶点子集 $V_1, \ldots, V_k$，定义这种分割的代价为：

$$\text{cut}\left(V_1,V_2,...,V_k\right)=\frac{1}{2}\sum_{i=1}^{k}W\left(V_i,\overline{V}_i\right)$$

其中$\overline{V}_i$为V_i的补集。该值与聚类的目标一致，即每个子图内部的连接很强，而子图之间的连接很弱，换一种语言来表述就是同一个子图内的样本相似，不同子图之间的样本不相似。但直接通过最小化这个值实现聚类还有问题，它没有考虑子图规模对代价函数的影响，使得这个指标最小的切分方案不一定就是最优切割。

解决这个问题的方法是对代价函数进行归一化。第一种方法是用图的顶点数进行归一化，由此得到优化的目标为：

$$\text{cut}\left(V_1,V_2,...,V_k\right)=\frac{1}{2}\sum_{i=1}^{k}\frac{W\left(V_i,\overline{V}_i\right)}{\left|V_i\right|}$$

其中$\left|V_i\right|$为子集的元素数，称为 RatioCut。另外一种归一化方案为

$$\text{cut}\left(V_1,V_2,...,V_k\right)=\frac{1}{2}\sum_{i=1}^{k}\frac{W\left(V_i,\overline{V}_i\right)}{\text{vol}\left(V_i\right)}$$

其中 vol 是图中所有顶点的加权度之和

$$\text{vol}\left(V\right)=\sum_{i\in V_i}d_i$$

称为 NCut。这两种情况都可以转化成求解归一化后的拉普拉斯矩阵的特征值问题。假设$\mathbf{L}$为图的拉普拉斯矩阵，$\mathbf{W}$为邻接矩阵，$\mathbf{D}$为加权度矩阵，它们的定义与 7.3 节相同。定义归一化后的拉普拉斯矩阵为

$$\mathbf{L}_{\text{sym}}=\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}=\mathbf{I}-\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

对于 RatioCut，求解的是如下特征值问题

$$\min_{\mathbf{H}\in\mathbb{R}^{n\times k}}\text{tr}\left(\mathbf{H}^{\text{T}}\mathbf{L}\mathbf{H}\right)$$
$$\mathbf{H}^{\text{T}}\mathbf{H}=\mathbf{I}$$

其中n为样本数，$\mathbf{I}$为单位矩阵，tr 为矩阵的迹，下面给出证明。首先考虑最简单的情况，将图切分成两个子图A和$\overline{A}$，此时要求解的最优化问题为

$$\min_{A\subset V}\text{RatioCut}\left(A,\overline{A}\right)$$

为方便表述，给定一个子集A，构造指示向量$\mathbf{f}=\left(f_1,...,f_n\right)^{\text{T}}$，表示每个样本所属的簇即子图，其元素的取值为

$$f_i=\begin{cases}\sqrt{\left|\overline{A}\right|/\left|A\right|} & v_i\in A\\ -\sqrt{\left|A\right|/\left|\overline{A}\right|} & v_i\in\overline{A}\end{cases}$$

根据该向量的定义有

$$\begin{aligned}\mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{f} &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\left(f_i - f_j\right)^2 \\ &= \frac{1}{2}\sum_{i\in A, j\in\overline{A}} w_{ij}\left(\sqrt{\left|\overline{A}\right|/\left|A\right|} + \sqrt{\left|A\right|/\left|\overline{A}\right|}\right)^2 + \frac{1}{2}\sum_{i\in\overline{A}, j\in A} w_{ij}\left(-\sqrt{\left|\overline{A}\right|/\left|A\right|} - \sqrt{\left|A\right|/\left|\overline{A}\right|}\right)^2 \\ &= \mathrm{cut}\left(A,\overline{A}\right)\left(\left|\overline{A}\right|/\left|A\right| + \left|A\right|/\left|\overline{A}\right| + 2\right) \\ &= \mathrm{cut}\left(A,\overline{A}\right)\left(\left(\left|A\right| + \left|\overline{A}\right|\right)/\left|A\right| + \left(\left|A\right| + \left|\overline{A}\right|\right)/\left|\overline{A}\right|\right) \\ &= \left|V\right|\cdot\mathrm{RatioCut}\left(A,\overline{A}\right)\end{aligned}$$

即给定任意子图 A，上面这个二次型与 RatioCut 的目标函数一致。另外根据 $\mathbf{f}$ 的定义有

$$\sum_{i=1}^{n} f_i = \sum_{i\in A}\sqrt{\left|\overline{A}\right|/\left|A\right|} - \sum_{i\in\overline{A}}\sqrt{\left|A\right|/\left|\overline{A}\right|} = \left|A\right|\sqrt{\left|\overline{A}\right|/\left|A\right|} - \left|\overline{A}\right|\sqrt{\left|A\right|/\left|\overline{A}\right|} = 0$$

即向量 $\mathbf{f}$ 与全 1 向量 $\mathbf{1}$ 正交。另外

$$\left\|\mathbf{f}\right\|^2 = \sum_{i=1}^{n} f_i^2 = \left|A\right|\left|\overline{A}\right|/\left|A\right| + \left|\overline{A}\right|\left|A\right|/\left|\overline{A}\right| = \left|\overline{A}\right| + \left|A\right| = n$$

因此向量 $\mathbf{f}$ 需要满足等式约束。求解的切图问题等价于如下带约束的最优化问题

$$\begin{aligned}&\min_{A\subset V}\ \mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{f} \\ &\mathbf{f}\perp\mathbf{1} \\ &\left\|\mathbf{f}\right\| = \sqrt{n}\end{aligned}$$

其中 $\perp$ 表示向量正交。向量 $\mathbf{f}$ 所有分量的取值必须为定义的两种情况，此问题是一个离散优化问题，为 NP 难问题，不易求解。对问题进行放松，变成连续优化问题

$$\begin{aligned}&\min_{\mathbf{f}\in\mathbb{R}^n}\ \mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{f} \\ &\mathbf{f}\perp\mathbf{1} \\ &\left\|\mathbf{f}\right\| = \sqrt{n}\end{aligned}$$

这个问题的解是 $\mathbf{L}$ 的第二小的特征值所对应的特征向量。因为该矩阵最小的特征值是 0，对应的特征向量是 $\mathbf{1}$。因此，第二小的特征值对应的特征向量近似是 RatioCut 的最优解。但是，切图所对应的结果应该是离散的，而这里得到的解是连续的，需要转换成离散的。一种解决方案是

$$\begin{cases} v_i \in A & f_i \geq 0 \\ v_i \in \overline{A} & f_i < 0 \end{cases}$$

类似于 sgn 函数。对于超过两个簇的情况，这种简单的阈值化不合适，此时可以将 f_i 当做点的坐标，用聚类算法将其聚成两类。然后按照如下的规则得到聚类结果

$$\begin{cases} v_i \in A & f_i \in C \\ v_i \in \overline{A} & f_i \in \overline{C} \end{cases}$$

推广到多个子图的情况，通过构造指示向量可以得到类似的优化目标。对于 NCut 最后求解的是如下广义特征值问题

$$\min_{\mathbf{H}\in\mathbb{R}^{n\times k}} \operatorname{tr}\left(\mathbf{H}^{\mathrm{T}}\mathbf{L}_{\mathrm{sym}}\mathbf{H}\right)$$
$$\mathbf{H}^{\mathrm{T}}\mathbf{H}=\mathbf{I}$$

在完成特征值分解之后，保留 k 个最小的特征值和它们对应的特征向量，构成一个 $n\times k$ 的矩阵，矩阵的每一行为降维后的样本数据。最后用其他聚类算法如 k 均值算法对降维之后的数据进行聚类。

下面介绍归一化谱聚类算法的流程，即文献[24]提出的方法。算法输入为相似度矩阵 $\mathbf{S}\in\mathbb{R}^{n\times n}$，需要生成的簇数 k，输出为聚类结果。流程如下：

1.构造相似度图，可以采用之前介绍的三种方式。假设 $\mathbf{W}$ 为带权重的邻接矩阵。

2.计算未归一化的拉普拉斯矩阵 $\mathbf{L}$。

3.计算下面广义特征值问题的前 k 个特征向量 $\mathbf{u}_1,...,\mathbf{u}_k$

$$\mathbf{L}\mathbf{u}=\lambda\mathbf{D}\mathbf{u}$$

4.假设矩阵 $\mathbf{U}\in\mathbb{R}^{n\times k}$ 为这些特征向量按照列构成的矩阵。对于 $\mathbf{U}\in\mathbb{R}^{n\times k}$，假设 $\mathbf{y}_i\in\mathbb{R}^k$ 为矩阵 $\mathbf{U}$ 的第 i 个行向量，对这些行向量用 k 均值算法进行聚类，得到簇 $C_1,...,C_k$。

5.输出最终的簇：$A_1,...,A_k$，其中 $A_i=\left\{j\middle|y_j\in C_i\right\}$

前面已经解释，需要使用 k 均值算法的原因是上面求解的问题是连续问题，而原始的切割问题是 NP 难的组合优化问题，为减小计算量将离散问题放宽为连续问题，因此要将连续优化问题的解变换回组合优化问题的解。

18.7 算法评价指标

和有监督学习算法一样，我们需要对聚类算法的效果进行评估。由于聚类算法要处理的样本可能没有人工标定的标签值，因此需要定义其特有的评价指标。这些指标可以分为内部指标和外部指标两种类型。

18.7.1 内部指标

内部指标只用算法对聚类样本的处理结果来评价聚类的效果，不依赖于事先由人工给出的标准聚类结果，因此它完全由聚类算法的内部结果而定。下面介绍几种常用的指标。

Davies-Bouldin 指标定义为：

$$\frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left(\frac{\sigma_i+\sigma_j}{d\left(c_i,c_j\right)}\right)$$

其中 n 为簇的数量，c_i 是第 i 个簇的质心，σ_i 是第 i 个簇的所有样本离这个簇的质心的平均距离。$d\left(c_i,c_j\right)$ 是第 i 个簇的质心与第 j 个簇的质心之间的距离。上式中的求和项是簇内差异与簇间差异的比值，因此这个指标值越大，聚类效果越差，反之则越好。

Dunn 指标是簇间距离的最小值与簇内距离的最大值之间的比值，计算公式为：

$$\frac{\min_{1\le i<j\le n} d(i,j)}{\max_{1\le k\le n} d'(k)}$$

其中 $d(i,j)$ 为第 i 个簇与第 j 个簇的簇间距离，$d'(k)$ 为第 k 个簇的簇内距离。这个比值越大，说明簇之间被分的越开，簇内越紧密，聚类效果越好；反之则聚类效果越差。

18.7.2 外部指标

外部指标用事先定义好的聚类结果来评价算法的处理效果，它的计算依赖于人工标定结果。下面介绍几种典型的指标。

纯度定义了一个簇包含某一个类的程度，即这个簇内的样本是否都属于同一个类，类似于决策树中的纯度指标。定义为：

$$\frac{1}{N}\sum_{m\in M}\max_{d\in D}|m\cap d|$$

其中 M 是人工划分的簇，D 是聚类算法划分的簇的集合，N 为样本数。这个指标反映了算法聚类的结果与人工聚类结果的重叠程度，值越大，说明聚类效果越好。Rand 测度定义了算法划分的结果与人工划分结果之间的耦合程度，定义为：

$$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}}$$

公式中的 4 个值定义与第 3.2 节相同。这个指标值越大，聚类结果与人工分类结果越相似。Jaccard 指标是两个集合的交集与并集的比值，类似于目标检测中的 IOU 指标，定义为：

$$\left|\frac{A\cap B}{A\cup B}\right|=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

这个值越大，说明两个集合的重合度越高，即算法聚类的结果与人工聚类的结果越接近，聚类效果越好。

18.8 实验程序

下面通过实验程序介绍 k 均值算法的使用。程序基于 sklearn 开源库，对随机生成的 3 类样本进行聚类，然后显示每个簇的样本以及簇中心。

程序源代码可以通过左侧二维码获取。程序运行结果如下图 18.5 所示。

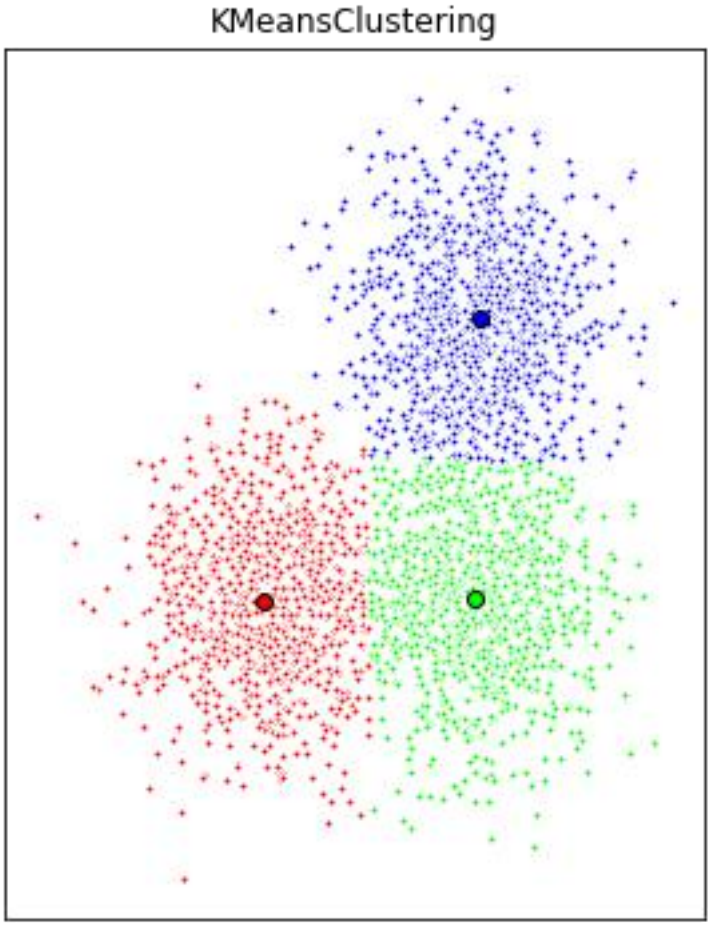

图 18.5　k 均值算法的聚类结果

18.9　应用

聚类算法有很多实际应用的案例。在自然语言处理的文本分析[9][10]，机器视觉中的图像分类[14][16]与视频分析[17]问题，基因数据分析[15]等问题中都有它的应用。图像分割问题也可以看做是聚类问题，将所有像素划分成几个类别。

参 考 文 献

[1] Rokach, Lior, Oded Maimon. Clustering methods. Data mining and knowledge discovery handbook. Springer US, 321-352, 2005.

[2] MacQueen, J. B. Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1. University of California Press. pp. 281 – 297, 1967.

[3] Martin Ester, Hanspeter Kriegel, Jorg Sander, Xu Xiaowei. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 226 – 231, 1996.

[4] Mihael Ankerst, Markus M Breunig, Hanspeter Kriegel, Jorg Sander. OPTICS: Ordering Points To Identify the Clustering Structure. ACM SIGMOD international conference on Management of data. ACM Press. pp. 49 – 60, 1999.

[5] Yizong Cheng. Mean Shift, Mode Seeking, and Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995.

[6] J C Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Cybernetics and Systems, 1973.

[7] Charles Elkan. Using the triangle inequality to accelerate k-means. international conference on machine learning, 2003.

[8] Arthur P Dempster, Nan M Laird, Donald B Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the royal statistical society series b-methodological, 1976.

[9] Inderjit S Dhillon, Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. Machine Learning. Machine Learning, 2001.

[10] Michael Steinbach, George Karypis, Vipin Kumar. A comparison of document clustering techniques. In KDD workshop on text mining (Vol. 400, No. 1, pp. 525-526), 2000.

[11] Dan Pelleg, Andrew W Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In ICML (Vol. 1), 2000.

[12] Greg Hamerly, Charles Elkan. Learning the k in k-means. Advances in neural information processing systems, 2004.

[13] D Sculley. Web-scale k-means clustering. international world wide web conferences, 2010.

[14] Gabriella Csurka, Christopher R Dance, Lixin Fan, Jutta Willamowski, Cedric Bray. Visual categorization with bags of keypoints. european conference on computer vision, 2004.

[15] Amir Bendor, Ron Shamir, Zohar Yakhini. Clustering Gene Expression Patterns. Journal of Computational Biology, 1999.

[16] Mohamed N Ahmed, Sameh M Yamany, Nevin A Mohamed, Aly A Farag, Thomas M Moriarty. A Modified Fuzzy C-Means Algorithm for Bias Field Estimation and Segmentation of MRI Data. IEEE Transactions on Medical Imaging, 2002.

[17] Tanvi Banerjee, James M Keller, Marjorie Skubic, Erik E Stone. Day or Night Activity Recognition From Video Using Fuzzy Clustering Techniques. IEEE Transactions on Fuzzy Systems, 2014.

[18] Alireza Kashanipour, Amir Reza Kashanipour, Nargess Shamshiri Milani, Peyman Akhlaghi. Robust Color Classification Using Fuzzy Reasoning and Genetic Algorithms in RoboCup Soccer Leagues. robot soccer world cup, 2008.

[19] Greg Hamerly, Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. conference on information and knowledge management, 2002.

[20] M Emre Celebi, Hassan A Kingravi, Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Systems With Applications, 2013.

[21] Andrew Y Ng, Michael I Jordan, Yair Weiss. On Spectral Clustering: Analysis and an algorithm. neural information processing systems, 2002.

[22] Ulrike Von Luxburg. A tutorial on spectral clustering. Statistics and Computing, 2007.

[23] Dorin Comaniciu, Peter Meer. Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.

[24] Shi, J. and Malik, J. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (8), 888-905. 2000.