

11 chapter

聚类

卿来云

聚类是一种非监督学习任务，其目的是发现数据中隐含的结构。聚类算法按照某种标准，把数据集划分成不同簇，使得同一个簇内的数据尽可能相似，不同簇中的数据差异尽可能大。

本章我们将讨论聚类算法性能的评价指标和常用的样本间相似性度量/距离度量，然后学习一些常用的聚类算法，包括K均值、混合高斯模型、层次聚类、均值漂移、基于密度的噪声鲁棒的聚类方法和基于密度峰值等算法。最后我们通过案例，介绍Scilit-Learn中各种聚类算法的API。

“物以类聚、人以群分”，机器学习中的聚类（Clustering）算法发现数据中隐含的结构，按照某种标准（如距离）把数据集划分成不同的类或簇，使得同一个簇内的数据尽可能相似，不同簇中的数据差异尽可能大。簇内相似性越大，簇间差距越大，说明聚类效果越好。本章我们先讨论聚类算法性能的评价指标和常用的样本间相似性度量/距离度量，然后学习一些常用的聚类算法。

11.1 聚类算法的评价指标

监督学习中，由于训练集中的样本/校验集的样本有标签，所以评价指标容易理解。聚类是非监督学习，相对而言，评价指标没那么直观。根据评价数据集是否有类别标注，聚类效果评价有两类指标衡量：一类是有参考分类结果作基准的外部评价指标，一类是无参考结果的内部评价指标。

11.1.1 外部评价指标

对数据集 $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ，假设通过聚类算法将样本聚为 K 类： $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$ ，参考结果给出的簇划分为 $\mathcal{C}^* = \{\mathcal{C}_1^*, \mathcal{C}_2^*, \dots, \mathcal{C}_K^*\}$ ，令 λ 和 λ^* 分别表示 \mathcal{C} 与 \mathcal{C}^* 对应的簇标记向量。我们计算聚类结果 \mathcal{C} 和参考类别结果 \mathcal{C}^* 的样本两两配对，定义如下：

$$\begin{aligned} a &= |S_1|, S_1 = \{(\mathbf{x}_i, \mathbf{x}_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \\ b &= |S_2|, S_2 = \{(\mathbf{x}_i, \mathbf{x}_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \\ c &= |S_3|, S_3 = \{(\mathbf{x}_i, \mathbf{x}_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \\ d &= |S_4|, S_4 = \{(\mathbf{x}_i, \mathbf{x}_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \end{aligned} \quad (11-1)$$

其中集合 S_1 包含在聚类结果 \mathcal{C} 中属于相同的簇并且在参考结果 \mathcal{C}^* 中也属于相同的簇的样本对， S_2 包含在 \mathcal{C} 中属于相同的簇但在 \mathcal{C}^* 中属于不同的簇的样本对， S_3 包含在 \mathcal{C} 中属于不同的簇但在 \mathcal{C}^* 中属于相同的簇的样本对， S_4 包含在 \mathcal{C} 中属于不同的簇，且在 \mathcal{C}^* 中属于相同的簇的样本对。每个样本对 $(\mathbf{x}_i, \mathbf{x}_j) (i < j)$ 仅能出现在一个集合中，因此有 $a + b + c + d = N(N - 1)/2$ 。

		参考结果	
		相同簇	不同簇
聚类结果	相同簇	a	b
	不同簇	c	d

基于以上定义，对聚类结果，我们定义如下的性能度量指标：

Jaccard 系数 (Jaccard Coefficent, JC)

$$JC = \frac{a}{a + b + c}。 \quad (11-2)$$

JC刻画了所有属于同一簇的样本对（要么在 \mathcal{C} 中属于同一簇，要么在 \mathcal{C}^* 中属于同一簇），同时在 $\mathcal{C}, \mathcal{C}^*$ 中属于同一簇的样本对的比例。 $JC \in [0, 1]$ ，JC越大，聚类效果与参考结果的一致性越高。

FM 指数 (Fowlkes and Mallows Index, FMI)

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}。 \quad (11-3)$$

令在 \mathcal{C} 中属于同一簇的样本对中，同时属于 \mathcal{C}^* 的样本对的比例为精度 (precision) P ；

在 \mathcal{C}^* 中属于同一簇的样本对中，同时属于 \mathcal{C} 的样本对的比例为召回率召回 (recall) R ，则 FMI 指数是精度和召回率的几何均值。FMI $\in [0,1]$ ，FMI越大，聚类效果与参考结果的一致性越高。

兰德指数 (Rand Index, RI)

$$RI = \frac{(a + d)}{C_N^2} = \frac{2(a + d)}{N(N - 1)}, \quad (11-4)$$

其中 $C_N^2 = N(N - 1)/2$ 为所有可能的样本对的数目， $a + d$ 表示聚类结果和参考结果吻合的样本对的数目 (在两种结果中两个样本都同属于一个簇，或者都属于不同簇)。 $RI \in [0,1]$ ，RI越大，聚类效果与参考结果的一致性越高。

调整兰德指数 (Adjusted rand index, ARI)

使用RI时有个问题，对于随机聚类，RI不保证接近 0 (可能还很大)。而ARI利用随机聚类情况下的RI ($\mathbb{E}[RI]$) 来解决这个问题，使得在聚类结果随机产生的情况下指标应近零：

$$ARI = \frac{RI - \mathbb{E}[RI]}{\max(RI) - \mathbb{E}[RI]}, \quad (11-5)$$

$ARI \in [-1,1]$ ，值越大意味着聚类结果与参考结果越吻合。

V 度量 (V-measure)

介绍 V-measure 之前，先介绍两个指标：同质性 (homogeneity) 和完整性 (completeness)。

基于下面表格，可以使用条件熵来定义同质性和完整性。

$\mathcal{C}/\mathcal{C}^*$	\mathcal{C}_1^*	...	\mathcal{C}_K^*	\sum
\mathcal{C}_1	$N_{1,1}$...	$N_{1,K}$	a_1
...
\mathcal{C}_K	$N_{K,1}$...	$N_{K,K}$	a_K
\sum	b_1	...	b_K	N

聚类结果 \mathcal{C} 的熵为：

$$H(\mathcal{C}) = - \sum_{k=1}^K p(\mathcal{C}_k) \log(p(\mathcal{C}_k)),$$

其中

$$p(\mathcal{C}_k) = \frac{a_k}{N}.$$

在给定分类结果 \mathcal{C}^* 条件下，聚类结果 \mathcal{C} 的条件熵为：

$$H(\mathcal{C}|\mathcal{C}^*) = - \sum_{k=1}^K \sum_{l=1}^K p(\mathcal{C}_k, \mathcal{C}_l^*) \log \frac{p(\mathcal{C}_k, \mathcal{C}_l^*)}{p(\mathcal{C}_l^*)},$$

其中

$$p(\mathcal{C}_k, \mathcal{C}_l^*) = \frac{N_{k,l}}{N}, \quad p(\mathcal{C}_l^*) = \frac{b_k}{N}.$$

在给定聚类结果 \mathcal{C} 条件下，分类结果 \mathcal{C}^* 的条件熵为：

$$H(\mathcal{C}^*|\mathcal{C}) = -\sum_{k=1}^K \sum_{l=1}^K p(\mathcal{C}_k, \mathcal{C}_l^*) \log \frac{p(\mathcal{C}_k, \mathcal{C}_l^*)}{p(\mathcal{C}_k)}.$$

那么我们定义同质性为：

$$h = 1 - \frac{H(\mathcal{C}|\mathcal{C}^*)}{H(\mathcal{C})}. \quad (11-6)$$

如果一个簇中只包含一个类别的样本，则同质性好。

完整性定义为：

$$c = 1 - \frac{H(\mathcal{C}^*|\mathcal{C})}{H(\mathcal{C}^*)}. \quad (11-7)$$

如果一个类的所有元素都分配给同一个簇，则完整性好。

V 度量 (V-measure) 是同质性和完整性的调和平均：

$$\nu = 2 \times \frac{h \times c}{h + c}. \quad (11-8)$$

$\nu \in [0,1]$ ，值越大，聚类结果和参考结果越一致，聚类效果越好。

11.1.2 内部评价指标

绝大多数情况下，我们不知道每个样本真实类别标签，没有参考结果，这时我们只能用内部评价法评估聚类的性能。

对数据集 $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ，假设通过聚类算法将样本聚为 K 类： $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$ ，我们定义如下指标：

$$\begin{aligned} \text{avg}(\mathcal{C}_k) &= \frac{2}{|\mathcal{C}_k|(|\mathcal{C}_k| - 1)} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C}_k} \text{dist}(\mathbf{x}_i, \mathbf{x}_j), \\ \text{diam}(\mathcal{C}_k) &= \max_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C}_k} \text{dist}(\mathbf{x}_i, \mathbf{x}_j), \\ \text{d}_{\min}(\mathcal{C}_k, \mathcal{C}_l) &= \min_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C}_k} \text{dist}(\mathbf{x}_i, \mathbf{x}_j), \\ d_{cen}(\mathcal{C}_k, \mathcal{C}_l) &= \text{dist}(\boldsymbol{\mu}_k, \boldsymbol{\mu}_l), \end{aligned} \quad (11-9)$$

其中 $|\mathcal{C}_k|$ 表示第 k 个簇 \mathcal{C}_k 中的样本数目， $\text{dist}(\mathbf{x}_i, \mathbf{x}_j)$ 表示两个样本 $\mathbf{x}_i, \mathbf{x}_j$ 之间的距离， $\boldsymbol{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_i \in \mathcal{C}_k} \mathbf{x}_i$ 表示簇 \mathcal{C}_k 的中心。所以 $\text{avg}(\mathcal{C}_k)$ 表示簇 \mathcal{C}_k 中所有样本对之间的平均距离， $\text{diam}(\mathcal{C}_k)$ 表示 \mathcal{C}_k 中距离最远的两个样本之间的距离， $\text{d}_{\min}(\mathcal{C}_k, \mathcal{C}_l)$ 为两个簇的最短距离， $d_{cen}(\mathcal{C}_k, \mathcal{C}_l)$ 则是两个簇中心之间的距离。因此前两个指标表示一个簇内样本之间的距离，这个距离越小越好；后两个指标表示两个簇之间的距离，这个距离越大越好。

基于上述定义式，可有以下的内部指标。

戴维森堡丁指数 (Davies-Bouldin Index, DBI)

$$\text{DBI} = \frac{1}{K} \sum_{k=1}^K \max_{l \neq k} \left(\frac{\text{avg}(\mathcal{C}_k) + \text{avg}(\mathcal{C}_l)}{\text{d}_{\text{cen}}(\boldsymbol{\mu}_k, \boldsymbol{\mu}_l)} \right). \quad (11-10)$$

DBI为簇内距离除以簇间距离，所以DBI越小，聚类效果越好。

邓恩指数 (Dunn Validity Index, DVI)

$$\text{DVI} = \frac{\min_{k \neq l} \text{d}_{\text{min}}(\mathcal{C}_k, \mathcal{C}_l)}{\max_m \text{diam}(\mathcal{C}_m)}. \quad (11-11)$$

DVI计算任意两个簇的最短距离（类间距离）除以任意簇中的最大距离（类内距离），因此DVI越大聚类效果越好。

Calinski-Harabaz 指数 (Calinski-Harabaz Index, CHI)

$$\text{CHI} = \frac{\text{tr}(\mathbf{B})}{\text{tr}(\mathbf{W})} \times \frac{N - K}{K - 1}, \quad (11-12)$$

其中 $\mathbf{B} = \sum_{k=1}^K |\mathcal{C}_k| (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T$ 为簇间散度矩阵， $\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$ 为所有样本的中心， $\text{tr}(\mathbf{B}) = \sum_{k=1}^K |\mathcal{C}_k| \text{dist}(\boldsymbol{\mu}_k, \boldsymbol{\mu})$ 为簇间散度矩阵的迹； $\mathbf{W} = \sum_{k=1}^K \sum_{x_i \in \mathcal{C}_k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T$ 为簇内散度矩阵， $\text{tr}(\mathbf{W}) = \sum_{k=1}^K \sum_{x_i \in \mathcal{C}_k} \text{dist}(\mathbf{x}_i - \boldsymbol{\mu}_k)$ 簇内散度矩阵的迹。CHI为簇内散度和与簇间散度和的比值，CHI越大，代表着簇自身越紧密，簇与簇之间越分散，聚类结果越好。

轮廓系数 (Silhouette coefficient, SC)

对于每个样本点 i ：

- (1) 计算 $a(i)$ ：样本点 i 到与其所属簇中其它点的平均距离，所以 $a(i)$ 与簇内散度有关；
- (2) 计算 $b(i)$ ：样本点 i 到其他类内所有点的平均距离， $b(i)$ 与簇间的距离有关。
- (3) 样本点 i 的轮廓系数为：

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}. \quad (11-13)$$

$s(i) \in [-1, 1]$ 。 $s(i) \rightarrow 1$, 表示该样本点离邻近的簇很远; $s(i) \approx 0$ 表示样本非常靠近相邻的簇，样本在两个簇的边界上； $s(i) \rightarrow -1$, 表示将该样本分配给错误的簇。

将所有点的轮廓系数求平均，就是该聚类结果总的轮廓系数：

$$s = \frac{1}{N} \sum_{i=0}^N s(i). \quad (11-14)$$

需要注意的是轮廓系数的计算复杂度高 ($O(N^2)$)，当样本数 N 很大时计算慢。

11.2 相似性度量

聚类的过程是将相似的样本聚成一簇，从而使得同一簇中的样本越相似越好，不同簇之间的样本越不相似越好。所以样本之间的相似性度量对聚类结果很关键。因为距离和相似度相反，有些聚类算法基于距离度量进行聚类（如 K 均值聚类）。一个合法的距离度量函数满足下

列条件：

$$\begin{aligned} \text{dist}(\mathbf{x}_i, \mathbf{x}_j) &\geq 0 ; & & \text{(非负性)} \\ \text{dist}(\mathbf{x}_i, \mathbf{x}_j) = 0 &\quad \text{iff } \mathbf{x}_i = \mathbf{x}_j ; & & \text{(可辨识性)} \\ \text{dist}(\mathbf{x}_i, \mathbf{x}_j) &= \text{dist}(\mathbf{x}_j, \mathbf{x}_i) ; & & \text{(对称性)} \\ \text{dist}(\mathbf{x}_i, \mathbf{x}_j) &\leq \text{dist}(\mathbf{x}_i, \mathbf{x}_k) + \text{dist}(\mathbf{x}_k, \mathbf{x}_j) . & & \text{(三角不等式)} \end{aligned} \quad (11-15)$$

样本间的相似性度量和具体应用有关，常用的相似性/聚类度量有：

欧氏距离

欧氏距离是最易于理解的一种距离计算方法，源自欧氏空间中两点间的直线距离公式：

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)} = \sqrt{\sum_{d=1}^D (x_{i,d} - x_{j,d})^2} . \quad (11-16)$$

曼哈顿距离

曼哈顿距离是一种很形象的命名。想象我们在曼哈顿要从一个十字路口开车到另外一个十字路口，驾驶距离不是两点间的直线距离，因为我们不能穿越大楼，只能沿着街区路线驾驶，驾驶距离就是“曼哈顿距离”：

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^D |x_{i,d} - x_{j,d}| . \quad (11-17)$$

切比雪夫距离

在二维棋盘格上，假设走一步能够移动到相邻的 8 个方格中的任意一个，那么从格子 (x_1, y_1) 走到格子 (x_2, y_2) 最少需要的步数是 $\max(|x_2 - x_1|, |y_2 - y_1|)$ 步，这个距离称为切比雪夫距离：

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \max_d |x_{i,d} - x_{j,d}| . \quad (11-18)$$

闵可夫斯基距离

闵可夫斯基距离不是一种距离，而是一组距离的定义：

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{d=1}^D (x_{i,d} - x_{j,d})^p \right)^{\frac{1}{p}} , \quad (11-19)$$

其中 p 是一个变参数。当 $p = 1$ 时，为曼哈顿距离；当 $p = 2$ 时，是欧氏距离。当 $p \rightarrow \infty$ 时，是切比雪夫距离（可用放缩法和夹逼法则证明）。

闵可夫斯基距离函数对特征的旋转和平移变换不敏感，但对数值的尺度敏感。如果样本不同特征的量纲不一致，需要将数据标准化。

马氏距离 (Mahalanobis Distance)

N 个样本向量 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ ，协方差矩阵记为 \mathbf{S} ，均值记为 $\boldsymbol{\mu}$ ，则 $\mathbf{x}_i, \mathbf{x}_j$ 之间的马氏距离定义为：

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)} . \quad (11-20)$$

若协方差矩阵是单位矩阵（各个特征之间独立同分布），则马氏距离就是欧氏距离了。若

协方差矩阵是对角矩阵，相当于对每维特征做标准化，然后再计算欧氏距离（标准化的欧氏距离）。因此马氏距离的优缺点是与特征的量纲无关，排除了变量之间的相关性的干扰。

汉明距离 (Hamming Distance)

在一个码组集合中，任意两个码字之间的汉明距离定义为对应位上码元取值不同的位的数目：

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^D x_{i,d} \oplus x_{j,d}, \quad (11-21)$$

其中 \oplus 表示异或操作。

两个等长字符串 s_1 与 s_2 之间的汉明距离定义为将其中一个变为另外一个所需要作的最小替换次数（字符串“1111”与“1001”之间的汉明距离为2）。

夹角余弦

若将两个样本 $\mathbf{x}_i, \mathbf{x}_j$ 看作 D 维空间的两个向量，则这两个向量间的夹角余弦可以表示这两个样本之间的相似度（不是距离）：

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|} = \frac{\sum_{d=1}^D x_{i,d} x_{j,d}}{\sqrt{\sum_{d=1}^D x_{i,d}^2} \sqrt{\sum_{d=1}^D x_{j,d}^2}}. \quad (11-22)$$

夹角余弦取值范围为 $[-1, 1]$ 。夹角余弦越大表示两个向量的夹角越小，夹角余弦越小表示两向量的夹角越大。当两个向量的方向重合时夹角余弦取最大值1，当两个向量的方向完全相反夹角余弦取最小值-1。

相关系数

相关系数亦被称为 Pearson 系数，定义为：

$$\begin{aligned} r(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\text{cov}(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_j}} = \frac{\mathbb{E}[(\mathbf{x}_i - \mu_i)(\mathbf{x}_j - \mu_j)]}{\sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_j}} \\ &= \frac{\sum_{d=1}^D (x_{i,d} - \mu_{i,d})(x_{j,d} - \mu_{j,d})}{\sqrt{\sum_{d=1}^D (x_{i,d} - \mu_{i,d})^2} \sqrt{\sum_{d=1}^D (x_{j,d} - \mu_{j,d})^2}}. \end{aligned} \quad (11-23)$$

当对数据做中心化后， $\mu_i = \mathbf{0}$, $\mu_j = \mathbf{0}$ ，此时相关系数等于夹角余弦，即 $r(\mathbf{x}_i, \mathbf{x}_j) = s(\mathbf{x}_i, \mathbf{x}_j)$ 。

杰卡德相似系数 (Jaccard similarity coefficient)

杰卡德相似系数是衡量两个集合的相似度一种指标。将一个样本 \mathbf{x}_i 看作是包含 D 个元素的集合，则两个样本 \mathcal{A}, \mathcal{B} 之间的杰卡德相似系数为这两个集合的杰卡德相似系数，即两个集合交集的元素在并集中两个集合并集元素所占的比例：

$$J(\mathcal{A}, \mathcal{B}) = \frac{\mathcal{A} \cap \mathcal{B}}{\mathcal{A} \cup \mathcal{B}}.$$

假设 $\mathbf{x}_i, \mathbf{x}_j$ 是两个 D 维向量，且所有维度的取值都是0或1。例如， $\mathbf{x}_i = (0, 1, 1, 0)^T, \mathbf{x}_j = (1, 0, 1, 1)^T$ 。我们将 $\mathbf{x}_i, \mathbf{x}_j$ 样本看成集合，1表示集合包含该元素，0表示集合不包含该元素。

p ：两个向量中对应维度都是1的维度的数目，

q ： \mathbf{x}_i 对应维度为1，而 \mathbf{x}_j 对应维度是0的维度的数目，

r : \mathbf{x}_i 对应维度为 0, 而 \mathbf{x}_j 对应维度是 1 的维度的数目,

s : 两个向量中对应维度都是 0 的维度的数目,

那么样本 $\mathbf{x}_i, \mathbf{x}_j$ 的杰卡德相似系数可以表示为 :

$$J(\mathbf{x}_i, \mathbf{x}_j) = \frac{p}{p + q + r}。 \quad (11-24)$$

此处分母之所以不加 s 的原因在于 : 杰卡德相似系数处理的是非对称二元变量。非对称的意思是指状态的两个输出不是同等重要的, 例如, 疾病检查的阳性和阴性结果。按照惯例, 我们将比较重要的输出结果, 通常也是出现几率较小的结果编码为 1 (例如 HIV 阳性), 而将另一种结果编码为 0 (例如 HIV 阴性)。给定两个非对称二元变量, 两个都取 1 的情况 (正匹配) 认为比两个都取 0 的情况 (负匹配) 更有意义。负匹配的数量 s 认为是不重要的, 因此在计算时忽略。

杰卡德相似度算法没有考虑向量中数值的大小, 而是简单的处理为 0 和 1, 因此计算效率高, 但也损失了很多信息。

11.3 K 均值 (**KMeans**) 聚类

K 均值聚类是最常用的聚类算法。虽然其性能不一定好, 但速度快, 因为只需计算样本点和簇中心之间的距离, 具有线性复杂度 $O(N)$ 。

K 均值聚类的基本思想是将样本划分到离其最近的簇中, 以迭代方式实现, 如算法 11.1 所示。

算法 11-1: K 均值聚类

输入 : 训练样本 : $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$;

簇的数目 : K

输出 : 每个簇中心 $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$, 每个样本所属的簇标记矩阵 \mathbf{R} 、每个簇的样本集合 $\mathcal{C}_1, \dots, \mathcal{C}_K$

1. 选择 K 个点作为初始质心 : $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$;

2. Repeat

2.1 将每个点指派到离其最近的质心, 形成 K 个簇 :

$$\lambda_i = \operatorname{argmin}_{k'} \operatorname{dist}(\mathbf{x}_i, \boldsymbol{\mu}_{k'}),$$

$$\begin{cases} r_{i,k} = 1, k = \lambda_i \\ r_{i,k} = 0, k \neq \lambda_i \end{cases}$$

$$2.2 \text{ 重新计算每个簇的质心} : \boldsymbol{\mu}_k = \frac{\sum_{i=1}^N r_{i,k} \mathbf{x}_i}{\sum_{i=1}^N r_{i,k}};$$

Until 簇不发生变化或达到最大迭代次数。

上述过程也可以看成是对下述目标函数取极小值 :

$$J(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \mathbf{R}) = \sum_{i=1}^N \sum_{k=1}^K r_{i,k} \operatorname{dist}(\mathbf{x}_i, \boldsymbol{\mu}_{\lambda_i}) = r_{i,k} \|\mathbf{x}_i - \boldsymbol{\mu}_{\lambda_i}\|_2^2, \quad (11-25)$$

即样本到其所属簇中心的距离和最小, 其中 λ_i 为样本 i 所属簇索引, $r_{i,k} = 1$ 表示第 i 个样本属于第 k 个簇, 否则 $r_{i,k} = 0$ 。

目标函数的参数包含两部分 : 每个簇的中心 $\boldsymbol{\mu}_k$ 和每个样本所属簇归属指示矩阵 \mathbf{R} 。在优化时我们采用坐标轴下降法, 即先固定其他参数(如簇中心)优化一个参数(如簇归属指示矩阵) ;

然后再交换参数进行迭代优化。

在给定簇的中心 μ_k 的情况下，将每个点指派到离其最近的质心会使得目标函数最小。在给定每个样本所属簇 $r_{i,k}$ 的情况下，目标函数对参数 μ_k 计算偏导；

$$\frac{\partial J(\mu_1, \dots, \mu_K, R)}{\partial \mu_k} = 2 \sum_{i=1}^N r_{i,k} (\mathbf{x}_i - \mu_{\lambda_i}) = 0.$$

从而得到：

$$\mu_k = \sum_{i=1}^N r_{i,k} \mathbf{x}_i. \quad (11-26)$$

在给定每个样本所属簇的情况下，当距离度量dist()取欧氏距离时，簇中心为该簇样本集合的均值，这也是K均值聚类算法名称的由来。由于均值计算对噪声比较敏感，可以将簇的质心由均值换成中值，得到K中值(KMedians)聚类算法。换成中值的另一个好处是中值是一个真实存在的样本，而均值通常不是一个真实样本/原型。但对于较大的数据集，K中值聚类要慢得多，因为在计算中值时，每次迭代都需要进行排序。

K均值聚类是在目标函数进行坐标轴下降优化，所以目标函数会单调下降，算法会收敛。但由于目标函数非凸，K均值不能保证收敛到全局最小值。不同的初始值可能会产生不同的聚类结果。一种常见的做法是以不同的初始值，运行K均值算法多次，再从中选择最好的结果。Scikit-Learn中KMeans构造函数中可设置参数n_init，默认值为10，即取不同的初始值运行10次K均值，取目标函数值最小那次的结果作为最终结果。

K均值聚类中，不同初始值得到最后的聚类结果可能不同，因此我们在选择初始质心时需要仔细。一个解决方案是随机确定第一个质心，其他质心的位置尽量远离已有质心。Scikit-Learn中KMeans构造函数中可设置参数init='k-means+'实现。

K均值聚类中，我们必须选择K的值，即聚成多少个类，K的选择不能以上述优化的目标函数为准，因为K越大，目标函数的值越小。可能的解决方案有：

(1) 画出训练集上目标函数值随K的变化曲线，“肘部”位置的K为最佳的K，即随着K的增加，目标函数的值不再显著下降；一个示例如图11-1所示。

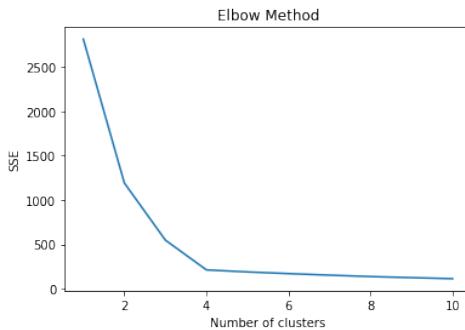


图 11-1 用“肘部”法选择K均值聚类中最佳的K，这里最佳的K = 4

- (2) 根据11.1节中的评价指标，如CHI，做准则选择即最佳的K的值；
- (3) 若聚类是一个有监督学习任务的一部分，可以以监督学习任务的评价指标为准则选取。

K 均值聚类简单快速，但也有很多缺点：

(1) K 均值聚类需要指定簇的数目 K 。如图 11-2 左上角图所示， K 不正确时，聚类结果不好。

(2) K 均值聚类根据样本点到簇中心的欧氏距离指派样本所属簇，相当于假设簇的形状为球形高斯分布。所以当数据集不满足这些假设时，K 均值聚类的效果不好（如图 11-2 右上角图所示），我们需要考虑其他方法。

(3) K 均值聚类直接用欧氏距离作为指标，相当于假设考虑各个簇的方差/散布程度相同。

当数据不符合假设时，聚类效果不好（如图 11-2 左下角图所示）。

(4) 在 K 均值聚类中，每个簇的地位相同，相当于假设每个簇的概率相等（每个簇的大小相等，密度相等），所以不能处理每个簇样本数差异很大的情况（如图 11-2 右下角图所示）。

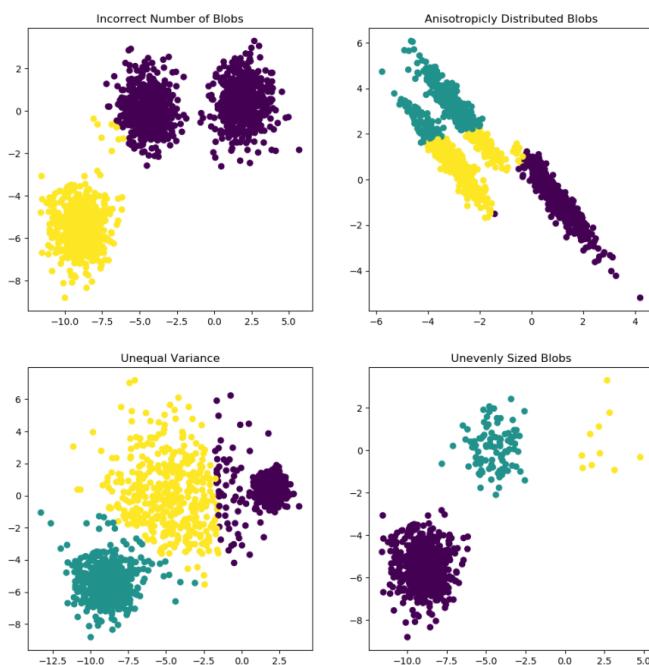


图 11-2 K 均值聚类算法结果示例

11.4 高斯混合模型

K 均值聚类的缺点是它对于簇假设过于简单：每个样本只能属于一个簇，每个簇为一个球形高斯分布。高斯混合模型（Gaussian Mixed Models, GMM）是一种比 K 均值更灵活的聚类算法。

GMM 中，我们假设每个簇的数据点服从高斯分布，用两个参数来描述簇：均值向量 μ_k 和协方差矩阵 Σ_k 。由于有协方差参数，簇可以呈椭球形状，而不是被限制为球形。另外，我们不再要求每个样本只能属于一个簇，而是以一定的概率从属于每个簇。这样样本的归属度值 $r_{i,k} \in [0,1]$ ，而不是只能取 0 或 1 两个值。

所以 GMM 模型对数据产生的假设为：假设有 K 个簇，每一个簇服从高斯分布 $N(\mathbf{x}, \mu_k, \Sigma_k), k = 1, \dots, K$ 。首先以概率 π_k 随机选择一个簇 k ，用独热编码向量 \mathbf{z} 表示样本所属的簇 ($z_k = 1$ 表示选择第 k 个簇)，并从该簇的分布中采样出一个样本点 \mathbf{x} 。所以 GMM 的概率

密度函数为

$$p(\mathbf{x}) = \sum_{k=1}^K P(z_k = 1)p(\mathbf{x}|z_k = 1) = \sum_{k=1}^K \pi_k N(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \quad (11-27)$$

对给定的样本 \mathbf{x} , 根据贝叶斯公式, 计算样本 \mathbf{x} 从属簇 k 的条件概率 $P(Z_k = 1|\mathbf{x})$:

$$\begin{aligned} r_k = \mathbb{E}[z_k] &= P((z_k = 1|\mathbf{x})) = \frac{P(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{k'=1}^K P(z_{k'} = 1)p(\mathbf{x}|z_{k'} = 1)} \\ &= \frac{\pi_k N(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} N(\mathbf{x}, \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}. \end{aligned} \quad (11-28)$$

有了概率表示, 我们可以采用极大似然法求解模型参数。log 似然函数为

$$\begin{aligned} l(\boldsymbol{\theta}) &= \sum_{i=1}^N \log(\mathbf{x}_i|\boldsymbol{\theta}) \\ &= \sum_{i=1}^N \log \left(\sum_{k=1}^K \pi_k N(\mathbf{x}_i, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right), \end{aligned} \quad (11-29)$$

其中参数向量 $\boldsymbol{\theta} = (\pi_1, \dots, \pi_K, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K)^T$ 。原则上计算目标函数 $l(\boldsymbol{\theta})$ 对各个参数的偏导数并等于 0, 可以使得似然函数最大的参数值。但目标函数中 log 函数的输入是里有求和, 所有参数耦合在一起, 导致计算困难。

令目标函数 $l(\boldsymbol{\theta})$ 对参数 $\boldsymbol{\mu}_k$ 求偏导数并等于 0, 得到

$$\begin{aligned} \frac{\partial l(\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} &= \sum_{i=1}^N \frac{1}{\sum_{k'=1}^K \pi_{k'} N(\mathbf{x}_i, \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \times \frac{\partial [\pi_k N(\mathbf{x}_i, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]}{\partial \boldsymbol{\mu}_k} \\ &= \underbrace{\sum_{i=1}^N \frac{1}{\sum_{k'=1}^K \pi_{k'} N(\mathbf{x}_i, \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \times \pi_k N(\mathbf{x}_i, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) (-\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k))}_{r_{i,k}} \\ &= - \sum_{i=1}^N r_{i,k} \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) = 0. \end{aligned}$$

得到:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^N r_{i,k} \mathbf{x}_i}{\sum_{i=1}^N r_{i,k}}. \quad (11-30)$$

类似的, 可得到:

$$\begin{aligned} \boldsymbol{\Sigma}_k &= \frac{\sum_{i=1}^N r_{i,k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^N r_{i,k}}, \\ \pi_k &= \frac{\sum_{i=1}^N r_{i,k}}{N}. \end{aligned} \quad (11-31)$$

需要注意上面的结果并不是封闭解, 因为 $r_{i,k}$ 依赖于参数 $\boldsymbol{\theta}$ 。不过上述结论也提示了我们求解问题的方案, 我们可以采用类似 K 均值聚类的迭代求解过程: 先初始化参数 $\boldsymbol{\theta}$, 计算 $r_{i,k}$; 然后再根据上述结论, 更新参数 $\boldsymbol{\theta}$ 。

上述迭代求解过程是期望最大化 (Expectation Maximization, EM) 的一个特例：

算法 11-2：GMM 聚类

1 . 初始化参数 $\boldsymbol{\theta} = (\pi_1, \dots, \pi_K, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K)^T$;

2 . Repeat

 2.1 E 步 : 给定当前参数的估计值计算后验概率/从属度 :

$$r_{i,k} = \mathbb{E}[Z_{i,k}] = \frac{\pi_k N(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} N(\mathbf{x}, \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})};$$

 2.2 M 步 : 基于当前从属度更新参数 :

$$\begin{aligned}\pi_k &= \frac{\sum_{i=1}^N r_{i,k}}{N}, \\ \boldsymbol{\mu}_k &= \frac{\sum_{i=1}^N r_{i,k} \mathbf{x}_i}{\sum_{i=1}^N r_{i,k}}, \\ \boldsymbol{\Sigma}_k &= \frac{\sum_{i=1}^N r_{i,k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^N r_{i,k}};\end{aligned}$$

Until 似然函数收敛或达到最大迭代次数。

EM 是一种通用的求解带隐含变量的目标函数的方法。

对数似然函数 $\log(P(X|\boldsymbol{\theta})) = \log(\sum_Z \log P(X, Z|\boldsymbol{\theta}))$, 即对所有的隐含变量求和。如果我们知道完整数据 (X, Z) , 通常完整数据的对数似然函数 $\log P(X, Z)$ 很直接。然而隐变量 Z 不可见, 其信息只能从后验分布 $P(Z|X, \boldsymbol{\theta}^{(t)})$ 得到。因此, 我们不妨考虑其期望值, 得到 E 步计算公式 :

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \mathbb{E}_z[\log P(X, Z|\boldsymbol{\theta})] = \sum_Z P(Z|X, \boldsymbol{\theta}^{(t)}) \log P(X, Z|\boldsymbol{\theta}). \quad (11-32)$$

在 M 步, 我们最大化 $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$:

$$\boldsymbol{\theta}^{(t+1)} = \operatorname{argmin}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}). \quad (11-33)$$

GMM 中, 协方差矩阵的参数数目为 $D(D - 1)/2$, 其中 D 为特征的维度。对协方差矩阵施加限制, 会使得模型从简单到复杂 :

- 球形 (对角线上元素值为 1, 其余元素为 0) ;
- 对角形 (只有对角线上元素值非 0) ;
- 并列 (所有簇的协方差矩阵相同) ;
- 完全协方差。

模型越复杂, 通常性能越好, 但在小数据集上容易过拟合。在 Scikit-Learn 中, 可以通过设置类 GaussianMixture 的参数 covariances 实现。

GMM 虽然比 K 均值聚类更加灵活, 簇形状可以是椭球 (如图 11-3 左图); 但 GMM 仍然假设簇的分布为高斯分布, 所以当簇的形状不满足高斯分布时, 聚类效果不佳 (如图 11-3 右图)。后续我们将讨论能发现任意形状簇的聚类算法。

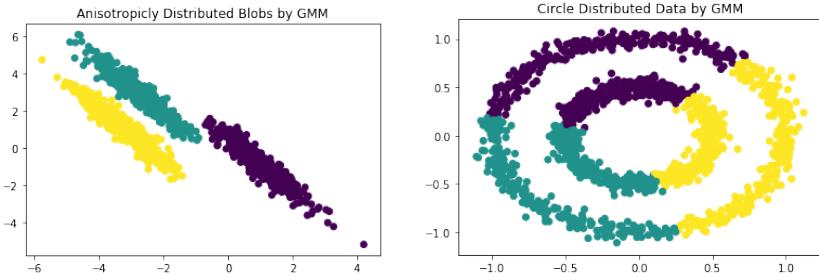


图 11-3 GMM 聚类算法结果示例

11.5 层次聚类

层次聚类算法可分为两类：自上而下（分裂式）或自下而上（凝聚式）。自下而上的算法首先将每个数据点视为一个簇，然后连续聚合两个簇，直到所有的簇都聚合成一个包含所有数据点的簇。因此，自下而上层次聚类被称为凝聚式层次聚类。类似的，分裂式从一个包含所有样本的簇开始，然后不断分裂已有簇，直到每个簇只包含一个样本。层次聚类的结果可用用树状图表示，树的根结点是收集所有样本的大簇，叶子结点仅包含一个样本的簇。由于凝聚式层次聚类计算量更小，一般层次聚类采用自下而上方式进行，Scikit-Learn 中也只支持凝聚式层次聚类。

11.5.1 凝聚式层次聚类

凝聚式层次聚类最开始每个样本视为一个簇，而后计算各簇之间的距离，将两个最相近的簇合并成一个新的簇。如此往复，最后就只剩下一个簇。

算法 11-3：凝聚式层次聚类

1. 初始化：每个样本为一个簇；
2. 计算簇与簇之间的相似度，可存为相似度矩阵；
3. Repeat
 - 3.1 合并最相似的两个簇；
 - 3.2 更新簇之间的相似度矩阵；

Until 只剩下一个簇。

层次聚类不需要指定簇的数量，甚至可以根据树状图选择一个合适簇的数量。与 K 均值和 GMM 的线性复杂度不同，层次聚类的效率较低，时间复杂度为 $O(N^3)$ 。

虽然相对其他聚类算法而言，层次聚类对样本点之间距离度量标准的选择并不敏感，但簇之间的相似度/距离度量对层次聚类还是很重要。常用的簇之间的距离度量有：

- 最小距离(MIN/Single Linkage)：两个簇中距离最近的样本对之间的距离

$$\text{dist}(\mathcal{C}_k, \mathcal{C}_l) = \min_{x_i \in \mathcal{C}_k, x_j \in \mathcal{C}_l} \text{dist}(x_i, x_j)。$$

- 最大距离(MAX, Complete Linkage)：

$$\text{dist}(\mathcal{C}_k, \mathcal{C}_l) = \max_{x_i \in \mathcal{C}_k, x_j \in \mathcal{C}_l} \text{dist}(x_i, x_j)。$$

- 平均距离(Group Average Linkage)：

$$\text{dist}(\mathcal{C}_k, \mathcal{C}_l) = \frac{1}{|\mathcal{C}_k||\mathcal{C}_l|} \sum_{x_i \in \mathcal{C}_k} \sum_{x_j \in \mathcal{C}_l} \text{dist}(x_i, x_j).$$

- 中心点距离(Distance Between Centroids):

$$\text{dist}(\mathcal{C}_k, \mathcal{C}_l) = \text{dist}(\boldsymbol{\mu}_k, \boldsymbol{\mu}_l),$$

其中 $\boldsymbol{\mu}_k, \boldsymbol{\mu}_l$ 分别为簇 $\mathcal{C}_k, \mathcal{C}_l$ 的质心。

- 沃德 (Wald) 距离

Wald 方法采用误差平方和 (Sum of the Squared Error, SSE) 采用衡量簇的质量。SSE 计算公式如下：

$$\text{SSE}(\mathcal{C}_k) = \sum_{x_i \in \mathcal{C}_k} \|x_i - \boldsymbol{\mu}_k\|_2^2.$$

Wald 距离定义为如果合并两个簇，带来 SSE 的增加量：

$$\text{dist}(\mathcal{C}_k, \mathcal{C}_l) = \text{SSE}(\mathcal{C}_k \cup \mathcal{C}_l) - \text{SSE}(\mathcal{C}_k) - \text{SSE}(\mathcal{C}_l) = \frac{|\mathcal{C}_k||\mathcal{C}_l|}{|\mathcal{C}_k| + |\mathcal{C}_l|} \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_l\|_2^2.$$

我们可以看到在沃德方法中，既考虑了簇间的距离，同时也考虑了每个簇中的样本数目。所以当两簇间的距离相等时，沃德方法会选择簇更小的那组进行合并。不同簇距离度量的聚类结果如图 11-4 所示。

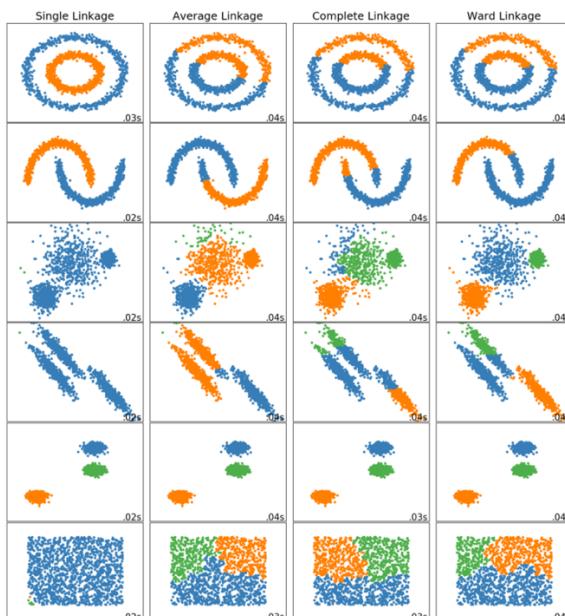


图 11-4 凝聚式层次聚类算法示例

在层次聚类中，我们可以合并到最后只剩下一个簇，也可以考虑在合并的费用增加很多时，合并聚类停止。

层次聚类算法较新的算法有：BRICH (Blanced Iterative Reducing and Clustering Using Hierarchies) 适合数值特征大数据情况；ROCK (RObust Clustering using linKs) 用于对离散型特征的样本进行聚类；变色龙 (Chameleon) 算法根据 k 近邻构造邻接图定义簇间相似性度量能自适应地合并簇。

11.5.2 分裂式层次聚类

二分 K 均值聚类算法是一种常用的分裂式层次聚类算法，是 K 均值聚类算法的一个变体，主要是为了改进 K 均值算法随机选择初始质心的随机性造成聚类结果不确定性的问题。

在二分 K 均值聚类中，最开始时所有的样本都属于同一个簇，这个簇为树状图的根结点。接下来我们要挑选一个质量最不好的簇，将其分裂成两个新的簇。选择哪个簇进行二分的原则是能否使得SSE尽可能小。同 K 均值聚类算法类似，二分 K 均值聚类算法也不适用于非球形簇的聚类，和不同尺寸和密度的类型的簇的聚类。

11.6 均值漂移聚类

均值漂移聚类试图找到数据点的最密集区域。我们采用核密度估计来估计数据的概率密度：

$$p(\mathbf{x}) = \frac{1}{Nh^D} \sum_{i=1}^N K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \quad (11-34)$$

其中 $K(\mathbf{x})$ 为核函数； h 为窗口大小，对模型性能影响大。当 h 较大时，得到的核密度估计比较平滑；当 h 较小时， h 较小时，得到的核密度变化剧烈。

我们通常采用径向基核函数，满足

$$K(\mathbf{x}) = c_{k,D} k(\|\mathbf{x}\|^2).$$

在分布的密度的极大值 \mathbf{x} 处， $p(\mathbf{x})$ 取极大值，所以 $\nabla_{\mathbf{x}} p(\mathbf{x}) = 0$ 。对公式(11-34)计算导数，得到

$$\begin{aligned} \nabla_{\mathbf{x}} p(\mathbf{x}) &= \frac{2c_{k,D}}{Nh^{D+2}} \sum_{i=1}^N (\mathbf{x}_i - \mathbf{x}) g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right) \\ &= \frac{2c_{k,D}}{Nh^{D+2}} \left[\sum_{i=1}^N g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right) \right] \left[\frac{\sum_{i=1}^N \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^N g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)} - \mathbf{x} \right], \end{aligned} \quad (11-35)$$

其中 $g(s) = -k'(s)$ ，第一项正比与核函数 $G(\mathbf{x}) = c_{g,D} g(\|\mathbf{x}\|^2)$ ，第二项

$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^N \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^N g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right)} - \mathbf{x} \quad (11-36)$$

为均值漂移向量（mean shift）。均值漂移向量总是指向密度增长最大的方向。因此通过迭代地

- 计算均值漂移向量 $\mathbf{m}_h(\mathbf{x}^{(t)})$ ，
- 移动窗口： $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \mathbf{m}_h(\mathbf{x}^{(t)})$ ，

可以收敛到概率密度的梯度为0之处。

算法 11-4：均值漂移聚类

1. 在未被分类的数据点中，随机选择一个点作为初始中心点 \mathbf{x} ；
2. 找出离该中心点距离在带宽 h 之内的所有点，记为集合 M ，这些点属于簇 C ；
3. 计算集合 M 中的所有点到中心点的距离向量，这些距离向量之和为偏移向量：

$$M_h = \sum_{i=1}^N \frac{k(\mathbf{x}_i, \mathbf{x}) \mathbf{x}_i}{k(\mathbf{x}_i, \mathbf{x})} - \mathbf{x}_o$$

4. 将中心点移动偏移向量（会移向更高密度区域）；
5. 重复步骤2、3、4，直到偏移均值向量的模的小于设定阈值，此时的中心点为一个簇中心。
6. 重复1、2、3、4、5直到所有的点都被归类。
7. 分类：根据每个簇，对每个点的访问频率，取访问频率最大的那个簇，作为当前点集的所属簇。

均值漂移聚类可以自动确定聚类数目，对不同初始值，聚类结果也相对稳定。但均值漂移需要指定窗口大小。窗口大小对核概率密度估计影响大，从而影响聚类结果。图 11-5 给出了两个不同窗口大小对应的聚类结果。当窗口较小时，会得到更多的簇。Scikit-Learn 中通过将类MeanShift构造函数的参数“bandwidth = None”，会调用estimate_bandwidth函数，根据数据集中近邻情况自动确定带宽。

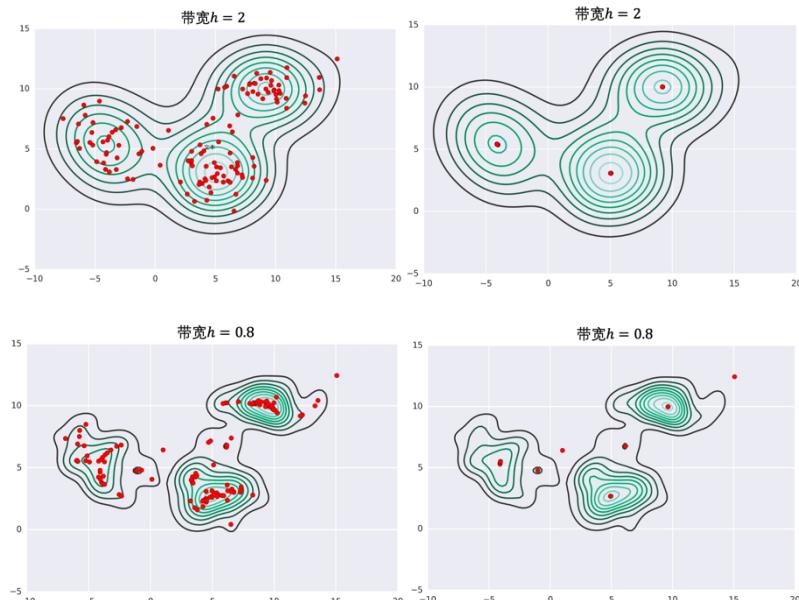


图 11-5 均值漂移聚类窗口宽度 h 的影响。第一行为高斯核函数 $h = 2$ 开始聚类和聚类收敛的结果，能正确找到 3 个簇；第二行为高斯核函数 $h = 0.8$ 开始聚类和聚类收敛的结果，此时得到 7 个簇。

11.7 DBSCAN

基于密度的噪声鲁棒的聚类（Density-Based Spatial Clustering of Applications with Noise, DBSCAN）是一种基于密度的聚类算法，它类似于均值漂移，可以处理不规则形状的簇，且对噪声数据的处理比较好。其核心思想是在数据空间中找到分散开的密集区域。

DBSCAN 定义密度为给定半径 ε 圆圈内的样本数目，然后根据密度将样本分成不同类别（如图 11-6 所示）：

- 核心点（Core point）：指定半径 ε 内多于指定数量（MinPts）个点；
- 边界点（Border point）：半径 ε 内有少于 MinPts 个点，但在某个核心点的领域内；
- 噪声点（Outliers）：核心点和边界点之外的点。

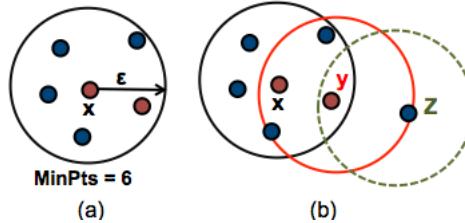


图 11-6 DBSCAN 中的基本概念。(b)中红色的点为核心点，圆圈中其他点为边界点，非圆圈中的点为噪声点。

在介绍 DBSCAN 的聚类过程之前，我们先做如下定义：

密度可达：如果连接两个点 q 和点 p 两个点的路径上所有的点都是核心点，则称点 q 到点 p 密度可达。如果 p 是核心点，那么由它密度可达的点形成一个簇。

密度相连：如果存在点 o ，从其密度可达点 q 和点 p ，则称点 q 到点 p 密度相连。

基于上述定义，簇满足以下两个性质：

1. 连接性：簇内任意两点是密度相连的；
2. 最大性：如果一个点从一个簇中的任意一点密度可达，则该点属于该簇。

算法 11-5：

DBSCAN 聚类

```

1. 初始化核心点集合 :  $\Omega = \emptyset$  ;
2. #确定核心点
   for  $i = 1, 2, \dots, N$ 
     2.1 确定样本点  $x_i$  的领域内的样本  $N_\epsilon(x_i)$ 
     2.2 if  $|N_\epsilon(x_i)| \geq \text{MinPts}$ ,
         则将样本  $x_i$  将加入核心点集合 :  $\Omega = \Omega \cup \{x_i\}$ 
     end if
   end for
3. 初始化簇的数目  $K = 0$ , 未访问过的点集合  $\Gamma = \mathcal{D}$  ;
4. #对所有核心点
   while  $\Omega \neq \emptyset$  do
     4.1 记录当前未访问过的点的集合  $\Gamma_{old} = \Gamma$  ;
     4.2 随机选择一个核心点  $o \in \Omega$ , 初始化队列  $Q = < o >$  ;
     4.3  $\Gamma = \Gamma \setminus \{o\}$ 
     4.4 while  $Q \neq \emptyset$  do
         4.4.1 取出队列  $Q$  中的首个样本点  $q$  ;
         4.4.2 if  $|N_\epsilon(q)| \geq \text{MinPts}$ ,
              $\Delta = N_\epsilon(q) \cap \Gamma$  为邻域中未访问过的点 ;
             将  $\Delta$  中的点加入队列  $Q$  ;
              $\Gamma = \Gamma \setminus \Delta$ ;
         end if
       end while
     4.5  $K = K + 1$ ,  $C_K = \Gamma_{old} \setminus \Gamma$  。

```

聚类结束后，不在任何簇内的点为噪声点（不是核心点，且不在任何核心点的邻域内），所以 DBSCAN 算法可以识别噪声，对噪声不敏感。DBSCAN 算法也能很好地找到任意大小和任意形状的簇。

DBSCAN 算法无需指定簇的数目 K ，但需要设置两个邻域参数： ε ，MinPts。如果 MinPts 不变， ε 取得值过大，会导致大多数点都聚到同一个簇中； ε 过小，会导致一个簇的分裂。如果 ε 不变，MinPts 的值取得过大，会导致更多样本点被标记为噪声点；MinPts 过小，会导致发现大量的核心点。参数可以根据 k -距离曲线和经验知识设置。

k -距离曲线的绘制过程如下：给定数据集 $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ ，对于任意点 x_i ，计算该点到集合中其他所有点的距离，这些距离按照从小到大排序，假设排序后的距离集合为 $D = d_{(1)}, d_{(2)}, \dots, d_{(k-1)}, d_{(k)}, d_{(k+1)}, \dots, d_{(N-1)}$ ，则 $d_{(k)}$ 就被称为 k -距离，即点 x_i 到其 k 近邻的距离。对所有点的 k -距离集合 E 进行升序排序，得到 k -距离集合 E' ，然后根据 E' 绘出 k -距离变化曲线。曲线发生急剧变化的位置所对应的 k -距离的值为半径 ε 的值（图 11-7 中 $\varepsilon = 0.15$ ），MinPts 的值为 k （图 11-7 中 MinPts = 5）。

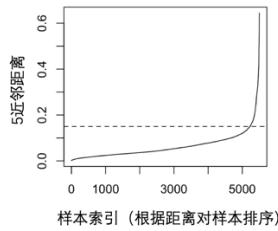


图 11-7 k -距离曲线

由于这两个参数是全局的，当簇的密度不同时，DBSCAN 的表现不如其他聚类算法。因为当密度变化时，用于识别邻域点的距离阈值 ε ，MinPts 的设置将会随着簇而变化。这个缺点在非常高维度的数据尤为突出，因为距离阈值 ε 变得难以估计。DBSCAN 的扩展 OPTICS (Ordering Points To Identify Clustering Structure) 通过优先对高密度进行搜索，然后根据高密度的特点设置参数，改进了 DBSCAN。

11.8 基于密度峰值的聚类

基于密度峰值的聚类算法[17]假设簇中心周围都是局部密度低的点，并且与任何一个局部密度较高的点保持相对较远的距离。

对于每一个数据点 i ，要计算两个量：点的局部密度 ρ_i 、和该点到具有更高局部密度的点的距离 δ_i ，而这两个值都取决于数据点间的距离 $d_{i,j}$ 。

数据点 i 的局部密度 ρ_i 定义为：

$$\rho_i = \sum_{j \in \mathcal{D}\{i\}} \chi(d_{i,j} - d_c), \quad (11-37)$$

其中 d_c 为截断距离，

$$\chi(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases} \quad (11-38)$$

所以 ρ_i 为到点*i*的距离小于 d_c 的点的数目。可以将所有点对相互距离从小到大排序，2%的位置距离数值设置为 d_c 。

上述局部密度的定义是一种硬邻域定义（是邻域点或不是邻域点），我们也可以换成用高斯核定义的软邻域定义，根据距离来表示数据点与中心点的权重，离的越近，权重越高，离得越远，权重就越低：

$$\rho_i = \sum_{j \in \mathcal{D} \setminus \{i\}} e^{-\left(\frac{d_{i,j}}{d_c}\right)^2} \quad (11-39)$$

点*i*到高局部密度的点的距离 δ_i 定义为该点到其他有更高局部密度的点之间的最小距离：

$$\delta_i = \min_{j: \rho_j > \rho_i} d_{i,j} \quad (11-40)$$

所以那些具有较大距离 δ_i 且同时具有较大局部密度 ρ_i 的点定义为聚类中心。同时具有较高的距离 δ_i 但密度 ρ_i 较小的数据点称为异常点。我们构造决策图和乘积曲线来寻找簇的数目和簇的中心。聚类中心确定之后，剩余点被分配给与其具有较高密度的最近邻居相同的簇。首先为每个簇定义一个边界区域，即划分给该簇但是距离其他簇的点的距离小于 d_c 的点。然后为每个簇找到其边界区域的局部密度最大的点，令该最大局部密度为 ρ_d ，则该簇中所有局部密度大于 ρ_d 的点被认为是簇核心部分，其余点被认为是该簇的光晕（噪声点）。图 11-8 给出了示例数据集上的聚类结果。

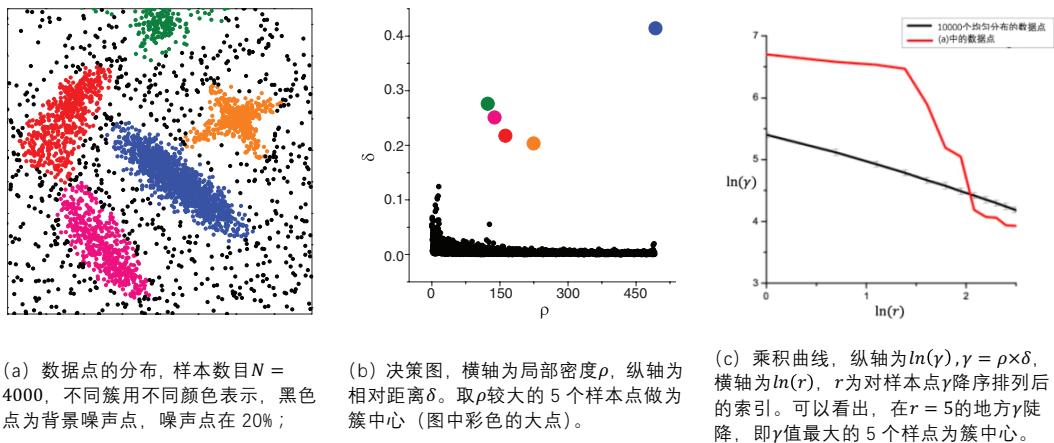


图 11-8 基于密度峰值的聚类

11.9 基于深度学习的聚类

基于深度学习的聚类将深度网络的学习和聚类结合起来，同时学习网络参数和对网络输出的特征进行聚类。我们亦可将深度网络部分视为从原始数据空间提取特征，然后在该特征空间中优化聚类。基于深度学习的聚类包含 3 部分：深度网络、网络学习目标函数和聚类目标函数。其中深度网络用于学习数据的低维非线性表示，自编码器、CNN、变分自编码器和 GAN 等模型均可用于聚类。深度聚类的目标函数通常是网络学习目标函数和聚类目标函数的线性组合：

$$J = \lambda J_c + (1 - \lambda) J_N, \quad (11-41)$$

其中 J_c 表示聚类损失，表示将样本归到某簇的代价，如 K 均值聚类中的 SSE； J_N 表示深度模型的学习的目标函数，如自编码器的重构误差和稀疏约束，超参数 λ 控制二者之间的折中。有些模型在深度网络学习中再加入局部保持约束。一种典型的模型如图 11-9 所示。

网络目标函数对于深度神经网络的初始化至关重要。通常在训练几轮之后，通过改变 λ 参数的值引入聚类损失。也有一些模型完全放弃网络学习目标函数，而只使用聚类目标函数来指导表示网络学习和聚类，此时需要仔细设计聚类目标函数，同时网络参数的初始化也很重要。

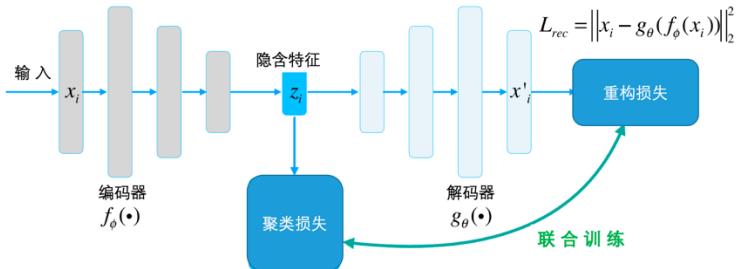


图 11-9 基于深度学习的聚类

11.10 聚类案例分析—MNIST 数据集聚类

我们在 MNIST 数据集进行聚类练习。MNIST 数据集是一个手写数字图像数据集，数据集介绍请见 5.3.3 节。数据集本身带有标签，方便我们比较各种聚类技术。我们从训练集中随机选择了 20%（共 8400 个样本）参与聚类试验。

我们在 Scikit-Learn 框架下实现了 K 均值聚类、GMM、均值漂移、DBSCAN。由于数字的图像中大部分为黑色的背景，且偏白色的数字笔划大多连续且灰度值相似，所以 MNIST 数据集中，样本的各个维度之间的相关性很强，因此我们首先采用 PCA 降维，保留 85% 方差，得到主成分数目为 59。后续聚类我们用手写数字图像 PCA 降维后的特征表示。这里我们用每个簇中出现次数最多的数字作为该簇样本的预测标签，计算预测的正确率。各聚类算法的性能如表 11-1 所示。

对 K 均值聚类，我们采用“肘部法”选择最佳的 K 值。不同 K 对应的 SSE 如图 11-10 (a) 所示。不过图中没有明显的肘部位置，相对而言 $K = 100$ 稍好些。GMM 中也取 $K = 100$ 。对层次聚类，无需指定簇的数目，不过可以通过观察不同簇数目对应的 SSE，确定最终的簇的数目。在 MNIST 数据集上，最后 200 个簇合并对应的 SSE 如图 11-10 (b) 所示。可以看出在欧氏距离为 100 左右时，再合并簇会带来较大的距离的增长，此时对应的簇的数目大约 100，因此最后簇的数目也设为 100。

均值漂移中的超参数为核函数的宽度，相当于 K 近邻中的 K。不同带宽对应的 CHI 值如图 11-10 (c) 所示。从图中我们可以发现，带宽越大，CHI 分数越大，并没有出现预期中的“U”型曲线，且当带宽继续增大时，CHI 分数继续增加，直到簇的数目只有 2 个。所以 CHI 分数并不适合用在这里做为聚类结果的评价。最终我们挑选簇的数目为 96 个（接近 100）对应的核函数的宽度。但聚类效果很不好。

DBSCAN 算法的超参数为邻域大小 eps 和邻域内最小样本数 MinPts 。超参数可以通过观察 k -距离曲线确定，如图 11-10 (c) 所示。不同 MinPts 对应的 k -距离曲线形状大致相同，图中没有看出距离显著增大的地方，因此不好确定 DBSCAN 的超参数 eps ，这也意味着 DBSCAN 聚类效果不会太好。在 750-800 之间近邻距离有较大的变化，因此取 $\text{eps} = \text{distanceDec}[780]$ ， $\text{MinPts} = 10$ ，对应的聚类效果也非常差，其他 MinPts 对应的聚类效果也很差。再适当减少 MinPts ，以发现更多核心点。 $\text{MinPts} = 5$ 的结果如表 11-1 所示。需要注意的是虽然预测正确率还可以，但还有 6851 个样本点没有参加聚类（被认为是噪声）。

表 11-1 聚类算法在 MNIST 数据集上的结果

聚类算法	簇的数目	预测正确率
K 均值聚类	100	0.832619
GMM	100	0.848929
层次聚类	100	0.845357
均值漂移	96	0.261548
DBSCAN	24	0.728212

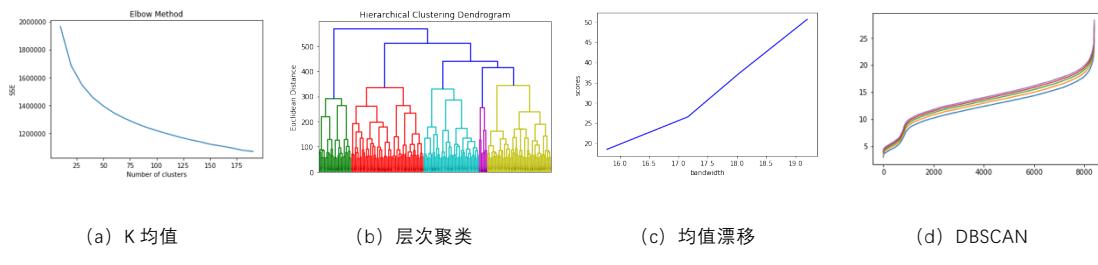


图 11-10 MNIST 数据集上聚类算法超参数确定

11.11 聚类算法小结

本章介绍了一些常用的聚类算法，其中 K 均值聚类及其改进算法属于给予划分的聚类算法， K 均值聚类简单快速被经常使用，虽然在很多情况下性能并不是很好。层次聚类我们讨论了凝聚和分裂两种形式。DBSCAN、均值漂移和基于密度峰值的聚类都属于基于密度的聚类算法。常用的聚类算法还有基于网格的聚类算法 STING、基于图的聚类算法（谱聚类）等。自组织网络（Self-Organized Maps, SMO）亦可实现聚类，深度学习也可以用于聚类任务，感兴趣的读者请自行查阅相关文献。

我们在选择聚类算法需要考虑的因素包括：数据规模和维度、簇的形状、数据的噪声水平，以及是否需要预先知道簇的数目等。

11.12 练习

- 假设数据集由 5 个簇高斯簇组成，采用 EM 算法，下面哪个模型的 log 似然值最小？
(A) 2 个簇的混合高斯模型

- (B) 5 个簇的混合高斯模型
(C) 10 个簇的混合高斯模型
2. 下面关于 K 均值和 GMM 关系的说法，哪些是正确的？
(A) K 均值可能会陷入局部最小值，GMM 的 EM 算法求解不会；
(B) GMM 能更好地表示不同方向和大小的簇；
(C) GMM 等价于无限小值的对角协方差的 GMM。
3. 现有一批 3 维的数据，我们采用 4 个簇的 GMM 模型来建模，假设协方差矩阵为全矩阵，则该模型有多少个参数？如果数据是 4 维，用 5 个簇的 GMM，假设协方差矩阵为对角矩阵，则模型有多少个参数？
4. 采用本章学过的聚类算法，对 10.10 节第 1 题中的数据，注意各算法中超参数的选择。

输入：训练样本： $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ；
邻域参数： ε , MinPts
输出：每个簇的样本集合 C_1, \dots, C_K

1. 初始化簇的数目 $K = 0$ ，标记所有的样本为未访问过；
2. for $i = 1, 2, \dots, N$ #每个样本点
 if \mathbf{x}_i 已经归入某个簇或标记为噪声
 continue;
 确定 \mathbf{x}_i 的邻域内的样本 $N_\varepsilon(\mathbf{x}_i)$
 if $|N_\varepsilon(\mathbf{x}_i)| < \text{MinPts}$
 将 \mathbf{x}_i 标记为边界点或噪声点；
 if $|N_\varepsilon(\mathbf{x}_i)| \geq \text{MinPts}$
 将 \mathbf{x}_i 标记为核心点
 建立新的簇 K ，将 \mathbf{x}_i 及其邻域内所有点 $N_\varepsilon(\mathbf{x}_i)$ 加入簇 K ；
 for $N_\varepsilon(\mathbf{x}_i)$ 尚未访问过的元素 q
 确定 q 的邻域内的样本 $N_\varepsilon(q)$
 if $|N_\varepsilon(q)| \geq \text{MinPts}$
 将 $N_\varepsilon(q)$ 未归入簇 K
 $K = K + 1$;