第10章　聚类分析

刘家锋

哈尔滨工业大学

第10章　聚类分析

10.1 无监督学习与聚类

聚类任务

- **无监督学习**
  - 无监督学习中，训练样本的标记信息是未知的
  - 学习的目标是要揭示训练数据的内在性质和规律
  - 例如在线性成分分析中，PCA属于无监督学习，LDA属于有监督学习
- **聚类任务**
  - 聚类是无监督学习中研究最多，应用最广的一类任务
  - 聚类试图将数据集中的样本划分为若干个不相交的子集，称为簇(cluster)
  - 每个簇对应一些潜在的概念，这些概念对聚类算法来说也是未知的，是在聚类过程中自动形成的

聚类任务

- **形式化描述**
  - 给定样本集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，均为无监督样本
  - 将D划分为k个不相交的簇$\{C_l | l = 1, \cdots, k\}$，其中

$$C_{l'} \bigcap_{l' \neq l} C_l = \varnothing, \qquad D = \bigcup_{l=1}^{k} C_l$$

  - 聚类结果可以表示为：$y_1, \cdots, y_n$
  - $y_j \in \{1, \cdots, k\}$，表示样本$\mathbf{x}_j$的簇标记

聚类与混合密度估计

- **混合密度**
  - 样本$\mathbf{x}$来自于k个聚类$\{\omega_1, \cdots, \omega_k\}$
  - 每个聚类的先验概率为$P(\omega_j)$，样本的分布为$p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$
  - 样本$\mathbf{x}$服从混合密度

$$p(\mathbf{x}) = \sum_{j=1}^{k} P(\omega_j) p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$$

- **聚类与混合密度估计**
  - 给定服从混合密度分布的样本集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$
  - 混合密度估计的目标是估计未知参数$P(\omega_j)$和$\boldsymbol{\theta}_j$
  - 聚类分析的目标是估计样本的聚类标记$\{y_1, \cdots, y_n\}$

高斯混合聚类

Algorithm 1 高斯混合聚类算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，聚类数$k$

Output: 簇划分$\mathcal{C} = \{C_1, \cdots, C_k\}$

1: 用数据集D及EM算法学习GMM的参数：$\{(\alpha_j, \boldsymbol{\mu}_j, \Sigma_j)\}_{j=1,\cdots,k}$

2: $C_j = \varnothing, \quad j = 1, \cdots, k$;

3: **for** $i = 1, \cdots, n$ **do**

4: 计算样本$\mathbf{x}_i$由各个高斯生成的后验概率

$$\gamma_{ij} = P(\omega_j|\mathbf{x}_i) = \frac{\alpha_j \cdot p(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{l=1}^{k} \alpha_l \cdot p(\mathbf{x}_i|\boldsymbol{\mu}_l, \Sigma_l)}$$

5: 标记$\mathbf{x}_i$，并划入相应的簇

$$y_i = \arg\max_{1 \le j \le k} \gamma_{ij}, \quad C_{y_i} \leftarrow C_{y_i} \cup \{\mathbf{x}_i\}$$

6: **end for**

10.2 k-均值聚类

k-means聚类算法

Algorithm 2 k-means聚类算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，聚类数$k$

Output: 数据集的聚类标签$\mathcal{Y} = \{y_1, \cdots, y_n\}$

1: 随机初始化聚类中心$\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k\}$

2: **repeat**

3: 更新聚类标签

$$y_i = \arg\min_{1 \le j \le k} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$

4: 更新聚类中心

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{n} I(y_i = j)\mathbf{x}_i}{\sum_{i=1}^{n} I(y_i = j)}$$

5: **until** $\mathcal{Y}$不再改变

k-均值算法的目标

- **k-means算法是对平方误差函数的迭代优化**
  - 给定聚类样本集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$
  - 将样本集划分为k个簇$\mathcal{C} = \{C_1, \cdots, C_k\}$
  - 聚类的目标是用每个簇的均值$\boldsymbol{\mu}_j$代替簇内样本的平方误差最小：

$$\min_{\mathcal{C}} \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \boldsymbol{\mu}_j\|^2$$

其中，每个簇的均值：

$$\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

  - k-means的优化目标是要使得每个簇内样本的相似度最大

模糊k-均值聚类

- **硬聚类与软聚类**
  - k-均值算法每一轮迭代，认为每个样本完全属于某个类别，而不属于其它类别，称为“硬聚类”
  - “软聚类”认为样本属于任意类别，只是属于的程度不同
- **隶属度**
  - 元素$\mathbf{x}$与集合A之间的关系可以用示性函数描述

$$\chi_A(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in A \\ 0, & \mathbf{x} \notin A \end{cases}$$

  - 模糊数学中，元素与模糊集合A之间的关系用隶属度函数描述：$\chi_A(\mathbf{x}) \in [0, 1]$
  - 隶属度的大小描述了元素$\mathbf{x}$属于集合A的程度

模糊k-均值聚类

- **聚类的隶属度定义**
  - “软聚类”借鉴模糊数学的概念，聚类过程中将每个簇看作一个模糊集合
  - 样本$\mathbf{x}$属于第j个簇的隶属度定义为

$$\chi_j(\mathbf{x}) = \frac{\left(1/\|\mathbf{x}-\boldsymbol{\mu}_j\|^2\right)^{\frac{1}{b-1}}}{\sum_{t=1}^{k}\left(1/\|\mathbf{x}-\boldsymbol{\mu}_t\|^2\right)^{\frac{1}{b-1}}}$$

    其中，$\boldsymbol{\mu}_j$为第j个簇的均值，自由参数$b>1$控制簇之间的混合程度

  - 簇的均值计算

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{n}\chi_j^b(\mathbf{x}_i)\cdot\mathbf{x}_i}{\sum_{i=1}^{n}\chi_j^b(\mathbf{x}_i)}$$

模糊k-均值聚类算法

Algorithm 3 模糊k-均值聚类算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，聚类数$k$

Output: 数据集的聚类标签$\mathcal{Y} = \{y_1, \cdots, y_n\}$

1: 随机初始化簇均值$\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k\}$

2: **repeat**

3: 更新样本的隶属度

$$\chi_j(\mathbf{x}_i) = \frac{\left(1/\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2\right)^{\frac{1}{b-1}}}{\sum_{t=1}^{k}\left(1/\|\mathbf{x}_i - \boldsymbol{\mu}_t\|^2\right)^{\frac{1}{b-1}}}$$

4: 更新簇均值

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{n}\chi_j^b(\mathbf{x}_i)\cdot\mathbf{x}_i}{\sum_{i=1}^{n}\chi_j^b(\mathbf{x}_i)}$$

5: **until** $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k\}$改变很小

6: **return** $y_i = \arg\max_j \chi_j(\mathbf{x}_i), \quad i = 1, \cdots, n$

10.3 层次聚类

层次聚类

- **Hierarchical clustering**
  - 层次聚类在不同层次对数据划分，形成树形的聚类结构
  - 数据集的划分可以采用“自底向上”的聚合策略，也可以采用“自顶向下”的分拆策略
- **AGNES(AGglomerative NESting)**
  - AGNES算法采用的是“自底向上”的聚合策略
  - 初始时，将数据集中的每一个样本作为一个簇
  - 每一轮迭代，选择距离最近的两个簇合并
  - 直到达到预设的聚类簇个数为止

AGNES算法

Algorithm 4 AGNES算法

Input: 数据集$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$，簇距离度量函数$d$，聚类簇数$k$

Output: 簇划分$\mathcal{C} = \{C_1, \cdots, C_k\}$

初始化簇：$C_j = \{\mathbf{x}_j\},\ j = 1, \cdots, n$；簇个数：$q = n$

计算簇距离矩阵：$M(i,j) = M(j,i) = d(C_i, C_j),\ i, j = 1, \cdots, n$

while $q > k$ **do**

　找出距离最近的两个聚类簇C_{i^*}和C_{j^*}

　合并C_{i^*}和C_{j^*}：$C_{i^*} \leftarrow C_{i^*} \cup C_{j^*}$

　删除M的第j^*行和列，重编号j^*之后的聚类簇

　重新计算M的第i^*行和列

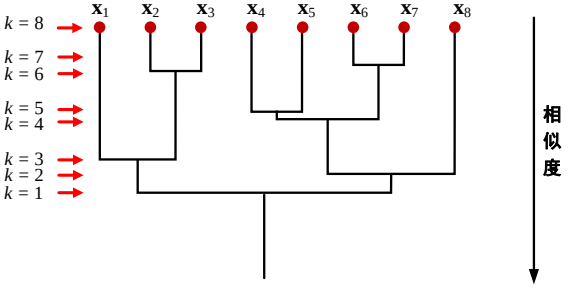
$$M(i^*, j) = M(j, i^*) = d(C_{i^*}, C_j), \quad j = 1, \cdots, q-1$$

　$q \leftarrow q - 1$

end while

层次聚类的树型图

- **层次聚类的终止条件**
  - 聚类数达到预设值k
  - 簇之间的最近距离大于设定的阈值



簇的距离度量：Hausdorff距离

- **最小距离(single-linkage)**
  - 以两个簇中距离最近的两个样本的距离作为簇之间的距离

$$d_{min}(C_i, C_j) = \min_{\mathbf{x}\in C_i, \mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$$

- **最大距离(complet-linkage)**
  - 以两个簇中距离最远的两个样本的距离作为簇之间的距离

$$d_{min}(C_i, C_j) = \max_{\mathbf{x}\in C_i, \mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$$

- **平均距离(average-linkage)**
  - 以两个簇之间样本对距离的平均值作为簇之间的距离

$$d_{min}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x}\in C_i} \sum_{\mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$$

例10.1 层次聚类

- **西瓜数据聚类**
  - 聚类数据：西瓜的密度和含糖量数据
  - 采用层次聚类算法，将样例聚类为7个聚类
  - 簇的距离度量采用“最大距离”

聚类数据

无标记西瓜数据集

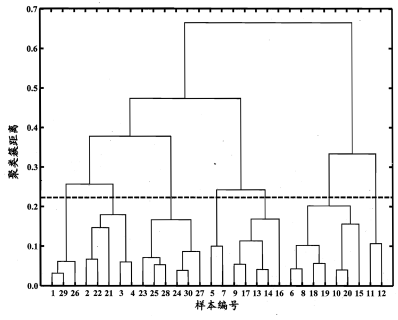
编号	密度	含糖量	编号	密度	含糖量	编号	密度	含糖量
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

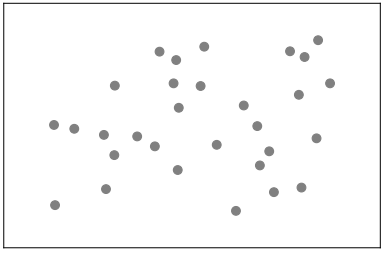
例10.1 层次聚类

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering

X = np.array([[0.697,0.460],[0.774,0.376],[0.634,0.264],[0.608,0.318],[0.556,0.215],
              [0.403,0.237],[0.481,0.149],[0.437,0.211],[0.666,0.091],[0.243,0.267],
              [0.245,0.057],[0.343,0.099],[0.639,0.161],[0.657,0.198],[0.360,0.370],
              [0.593,0.042],[0.719,0.103],[0.359,0.188],[0.339,0.241],[0.282,0.257],
              [0.748,0.232],[0.714,0.346],[0.483,0.312],[0.478,0.437],[0.525,0.369],
              [0.751,0.489],[0.532,0.472],[0.473,0.376],[0.725,0.445],[0.446,0.459]])

'''省略画图显示部分'''

agg=AgglomerativeClustering(linkage='complete',compute_full_tree=True,n_clusters=7)
for i, ax in zip(range(24),axes.reshape(24,1)):
    agg.n_clusters = X.shape[0] - i
    Labels = agg.fit_predict(X)

'''省略画图显示部分'''
```

层次聚类

AGNES算法聚类的过程，最大距离度量，聚类数设置$k = 7$





10.4 聚类模型选择

聚类模型的适用性

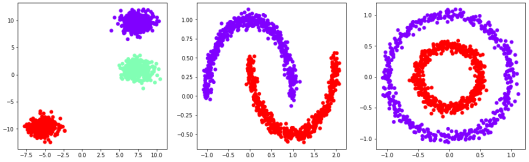
- **聚类算法的适用性**
  - 聚类算法是否有效与数据集的分布结构有关
  - 不同的聚类算法对数据集的分布做出了不同的假设
    - 原型聚类(Prototype-based Clustering)：每个簇的数据接近与球形分布，不同簇的数据数量相差不大，例如k-means算法
    - 层次聚类(Hierarchical Clustering)：数据集存在层次化的簇结构，例如AGNES算法
    - 密度聚类(Density-based Clustering)：同一簇的数据分布在一个高密度区域，不同簇之间存在数据分布的低密度区域，例如DBSCAN算法，谱聚类算法
  - 聚类算法只适用于满足相应分布结构假设的数据集

例10.2 不同的聚类算法

- **不同的数据分布**
  - 分别生成3组仿真数据，每组1000个样本
    - Blob数据：3个簇，每个簇呈“球形分布”
    - Moon数据：2个簇，每个簇呈“月形分布”
    - Circle数据：2簇，每个簇呈“环形分布”
  - 使用k-means，AGNES，DBSCAN和谱聚类算法聚类数据

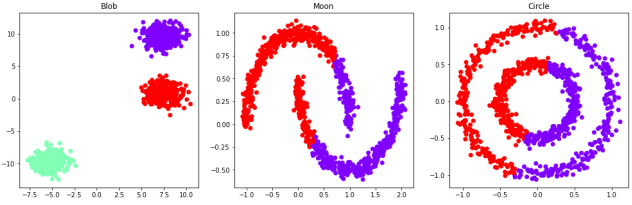
生成仿真数据

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons, make_blobs, make_circles

X_blob, y_blob = make_blobs(n_samples=1000, random_state=8)
X_moon, y_moon = make_moons(n_samples=1000, noise=0.05, random_state=0)
X_circle, y_circle = make_circles(n_samples=1000, factor=.5,noise=.05)

fig, axes = plt.subplots(1, 3, figsize=(17, 5))
axes[0].scatter(X_blob[:,0], X_blob[:,1], c=y_blob, cmap='rainbow')
axes[1].scatter(X_moon[:,0], X_moon[:,1], c=y_moon, cmap='rainbow')
axes[2].scatter(X_circle[:,0], X_circle[:,1], c=y_circle, cmap='rainbow')
plt.show()
```



k-means聚类

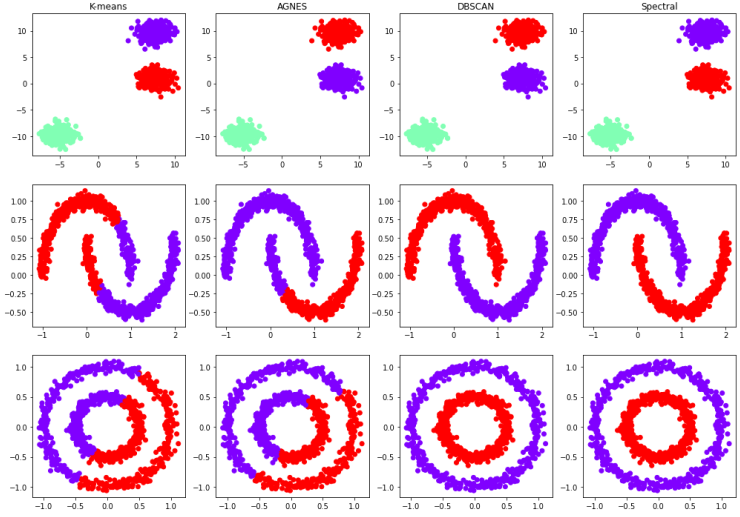
```
from sklearn.cluster import KMeans

fig, axes = plt.subplots(1, 3, figsize=(17, 5))

kmeans = KMeans(n_clusters=3)
y_predict = kmeans.fit_predict(X_blob)
axes[0].scatter(X_blob[:,0], X_blob[:,1], c=y_predict, cmap='rainbow')
axes[0].set_title("Blob")

'''部分省略'''
plt.show()
```



不同聚类算法

```
from sklearn.cluster import AgglomerativeClustering, DBSCAN, SpectralClustering

fig, axes = plt.subplots(3, 4, figsize=(17, 12))
algorithms = [ KMeans(n_clusters=3),
               AgglomerativeClustering(n_clusters=3),
               DBSCAN(min_samples=5, eps=2),
               SpectralClustering(n_clusters=3)]

names = ["K-means","AGNES","DBSCAN","Spectral"]

for ax,algorithm,name in zip(axes[0],algorithms,names):
    y_predict = algorithm.fit_predict(X_blob)
    ax.scatter(X_blob[:,0], X_blob[:,1], c=y_predict, cmap='rainbow')
    ax.set_title(name)

'''部分省略'''
plt.show()
```

不同聚类算法



聚类参数的选择问题

- **如何设置聚类算法的参数？**
  - 使用每一种聚类算法，都需要设置相应的参数
    - k-means算法：聚类数k，簇的初始均值$\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$
    - AGNES算法：聚类数k，簇之间的距离度量
    - DBSCAN算法：邻域距离ϵ，核心点最小邻域点数MinPts
    - 谱聚类算法：聚类数k，近邻方式，近邻数量
  - 使用不同的聚类算法，设置不同的聚类算法参数，都会得到不同的聚类结果
  - 聚类属于无监督学习问题，无法使用验证集来选择算法和参数
  - 一般需要采用某种指标来度量聚类结果的优劣

性能度量

- **外部指标**
  - 将聚类结果与参考模型(答案)比较，例如Jaccard系数，FM指数，Rand指数等
  - 外部指标无法用于选择聚类算法和参数
- **内部指标**
  - 直接考察聚类结果，不需要参考模型
  - 簇内样本的平方误差和

$$\text{SSD} = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \boldsymbol{\mu}_j\|_2^2$$

  - SSD只能度量度量簇内相似度，无法度量簇间的相似度

性能度量

- **DB指标(Davies-Bouldin Index)**
  - DB指标既可以度量簇内样本的相似度，也可以度量簇间样本的相异程度
  - Davies-Bouldin Index：

$$\mathsf{DBI} = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\mathsf{avg}(C_i)+\mathsf{avg}(C_j)}{d_{\mathsf{cen}}(C_i,C_j)}\right)$$

其中，簇C中样本的平均距离：

$$\mathsf{avg}(C) = \frac{2}{|C|(|C|-1)}\sum_{1\leq i<j\leq |C|}\mathsf{dist}(\mathbf{x}_i,\mathbf{x}_j)$$

簇C_i和C_j的中心距离：

$$d_{\mathsf{cen}}(C_i,C_j) = \mathsf{dist}(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)$$

$\boldsymbol{\mu}_i$为簇C_i的均值，$|C|$为簇C中的样本数

例10.3 聚类算法参数选择

- **聚类算法参数选择**
  - 仿真数据：随机生成100个2维数据，来自于3个簇
  - 随机指派每个数据的簇标记，计算Davies-Bouldin Index
  - 采用AGNES算法聚类，分别设置聚类数$k = 2, 3, 4$
  - 采用k-means算法聚类，分别设置聚类数$k = 2, 3, 4, 5$
  - 计算每个聚类结果的Davies-Bouldin Index

例10.3 聚类算法参数选择

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.datasets import make_blobs
from sklearn.metrics.cluster import davies_bouldin_score

X, y = make_blobs(random_state=1)

fig,axes = plt.subplots(2,4,figsize=(16,8),subplot_kw={'xticks':(),'yticks':()})
algorithms = [ AgglomerativeClustering(n_clusters=2),
               AgglomerativeClustering(n_clusters=3),
               AgglomerativeClustering(n_clusters=4),
               KMeans(n_clusters=2), KMeans(n_clusters=3),
               KMeans(n_clusters=4), KMeans(n_clusters=5) ]
```

例10.3 聚类算法参数选择

```
random_state = np.random.RandomState(seed=0)
random_clusters = random_state.randint(low=0, high=2, size=len(X))

axes[0,0].scatter(X[:,0], X[:,1], c=random_clusters,cmap='rainbow', s=60)
axes[0,0].set_title("Random assignment - DBI: %3.2f" \
                    %(davies_bouldin_score(X, random_clusters)))

for ax, algorithm in zip(axes.reshape(8,1)[1:8], algorithms):
    clusters = algorithm.fit_predict(X)
    ax[0].scatter(X[:,0], X[:,1], c=clusters,cmap='rainbow', s=60)
    ax[0].set_title("%s - DBI: %3.2f" \
        %(algorithm.__class__.__name__,davies_bouldin_score(X, clusters)))

plt.show()
```

例10.3 聚类算法参数选择