数据挖掘与应用

聚类

授课教师：周晟

浙江大学　软件学院

2022.10

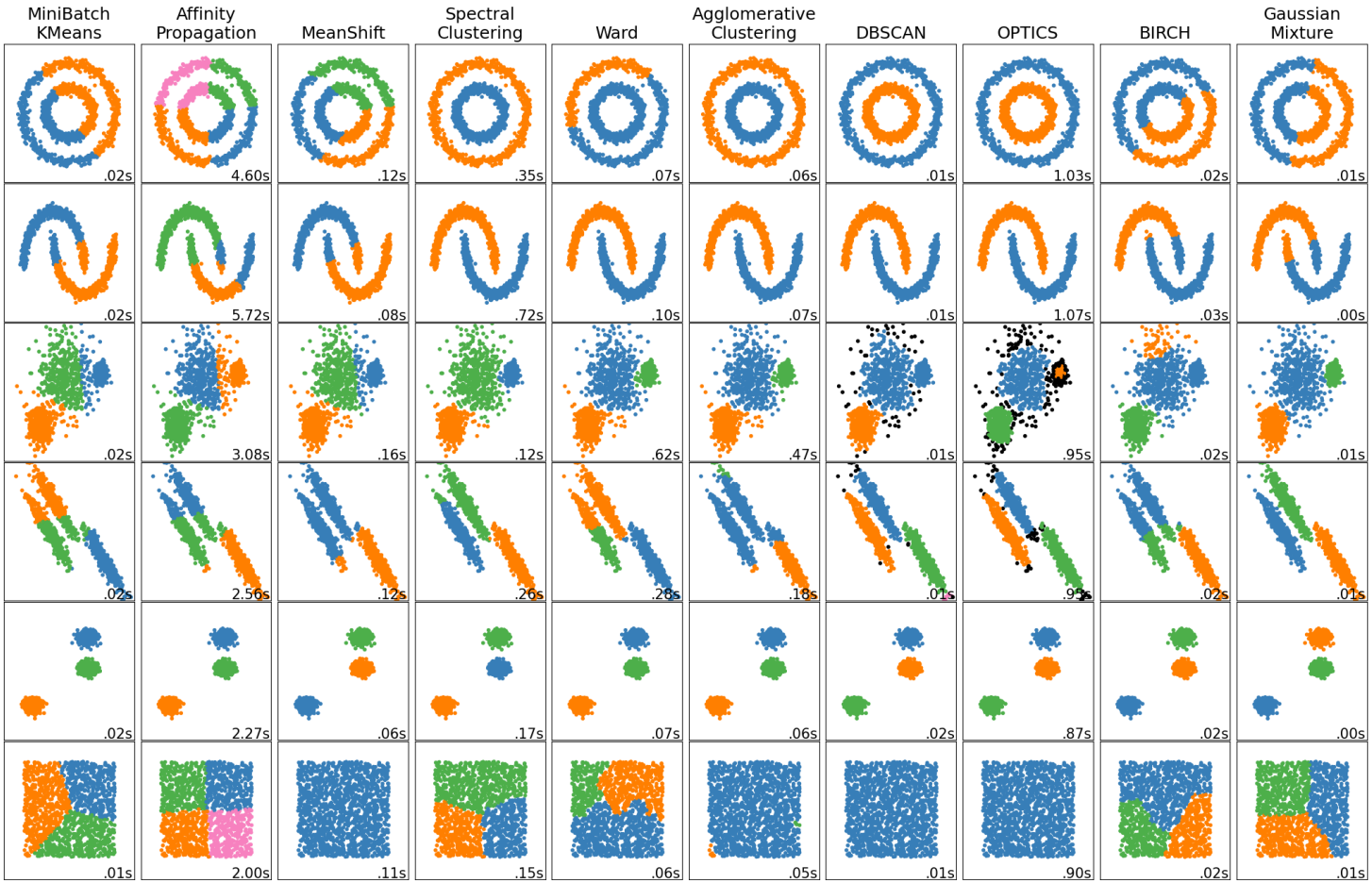
教学内容

1. 认识聚类
2. 聚类的性能指标
3. K-means 聚类
4. 谱聚类
5. 层次聚类
6. 高斯混合模型聚类
7. DBSCAN 聚类

教学内容

聚类的定义

聚类（clustering）是一个把数据对象集划分成多个组或簇（cluster）的过程，使得簇内的数据具有很高的相似性，不同簇的数据相似性很低。[1]



聚类的形式化定义

输入数据

假定样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ 包含 m 个无标记样本，每个样本 $\boldsymbol{x}_i = (x_{i1}; x_{i2}; \ldots; x_{in})$ 是一个 n 维特征向量

聚类过程

聚类算法将样本集 D 划分为 k 个不相交的簇 $\{C_l \mid l = 1, 2; \ldots, k\}$，其中 $C_{l'} \bigcap_{l' \neq l} C_l = \varnothing$ 且 $D = \bigcup_{l=1}^{k} C_l$.

聚类结果

用 $\lambda_j \in \{1, 2, \ldots, k\}$ 表示样本 $\boldsymbol{x}_j$ 的 “簇标记” (cluster label)，即 $\boldsymbol{x}_j \in C_{\lambda_j}$. 于是，聚类的结果可用包含 m 个元素的簇标记向量 $\boldsymbol{\lambda} = (\lambda_1; \lambda_2; \ldots; \lambda_m)$ 表示.

聚类的意义

聚类在数据挖掘与机器学习中具有重要意义：

1. 洞察数据分布
2. 缩小数据挖掘范围
3. 数据预处理

聚类的常见应用包括：

- 消费者群体聚类
- 图像聚类
- 产品定位
- 离群点检测

聚类的主要挑战

虽然聚类已经被研究多年，但是仍然需要面对如下问题：

1. 大规模数据聚类 - 不确定聚类个数，聚类效率问题
2. 数据特征类型多样 - 不同类型的特征融合
3. 数据特征高维 - 数据降维
4. 聚类大小不均衡 - 先验假设错误
5. 类别个数依赖人工先验 - 实际场景难以应用
6. 易受噪声数据影响 - 模型鲁棒性
7. 聚类结果可解释性 - 实际场景难以应用

外部指标

假设聚类算法给出的簇划分为 $C = \{C_1, C_2, ..., C_k\}$，而真实的簇划分为 $Y = \{Y_1, Y_2, ..., Y_s\}$。$k$ 和 s 不一定相同。令 λ^C 与 λ^Y 分别表示与 C 和 Y 对应的簇标签。定义样本对两两之间的关系：

$$
\begin{aligned}
a &= |SS|, \quad SS = \{(x_i, x_j) | \lambda_i^C = \lambda_j^C, \lambda_i^Y = \lambda_j^Y, i < j\} \\
b &= |SD|, \quad SD = \{(x_i, x_j) | \lambda_i^C = \lambda_j^C, \lambda_i^Y \neq \lambda_j^Y, i < j\} \\
c &= |DS|, \quad DS = \{(x_i, x_j) | \lambda_i^C \neq \lambda_j^C, \lambda_i^Y = \lambda_j^Y, i < j\} \\
d &= |DD|, \quad DD = \{(x_i, x_j) | \lambda_i^C \neq \lambda_j^C, \lambda_i^Y \neq \lambda_j^Y, i < j\}
\end{aligned}
$$

上述变量满足：$a + b + c + d = N(N-1)/2$

外部指标

基于上述变量可以定义如下聚类性能度量外部指标：

- Jaccard 系数（Jaccard Coefficient，简称 JC）

$$JC = \frac{a}{a+b+c}$$

- FM 指数（Fowlkes and Mallows Index，简称 FMI）

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$$

- 随机指数（Rand Index，简称 RI）

$$RI = \frac{2(a+d)}{a+b+c+d} = \frac{2(a+d)}{N(N-1)}$$

上述性能度量的结果值均在 $[0,1]$ 区间，值越大越好。

外部指标

- 归一化随机指数（Adjusted Rand Index，ARI）

$$ARI = \frac{RI - E[RI]}{\max RI - E[RI]}$$

ARI 取值范围在 $[-1, 1]$，值越大表示聚类效果越好。

$X \backslash Y$	Y_1	Y_2	$\dots$	Y_s	Sums
X_1	n_{11}	n_{12}	$\dots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\dots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\dots$	n_{rs}	a_r
Sums	b_1	b_2	$\dots$	b_s	

$$\overbrace{ARI}^{\text{Adjusted Index}} = \frac{\overbrace{\sum_{ij}\binom{n_{ij}}{2}}^{\text{Index}} - \overbrace{[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}^{\text{Expected Index}}}{\underbrace{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}]}_{\text{Max Index}} - \underbrace{[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}_{\text{Expected Index}}}$$

外部指标

上述指标是基于样本对的角度来计算，实际操作中，我们还可以直接从单个样本的角度更直观地来评估聚类的性能。

聚类精度（ACC）

对于数据集 $D = \{x_1, x_2, ..., x_N\}$，假设通过聚类给出的各样本标签为 $C = \{c_1, c_2, ..., c_N\}$，相对应的 ground truth 为 $Y = \{y_1, y_2, ..., y_N\}$

$$ACC = \max_m \frac{\sum_{i=1}^{N} \mathbf{1}\{y_i = m(c_i)\}}{N}$$

聚类划分得到的簇 id 可能与 ground truth 的 id 不同。因此，我们需要找到聚类标签与 ground truth 之间的最优映射。

外部指标

```python
def best_mapping(labels_true, labels_pred):

    # 构造 k*k 的对应数量矩阵（收益矩阵）
    D = max(max(labels_true), max(labels_pred)) + 1
    w = np.zeros((D, D), dtype=np.int64)
    for i in range(len(labels_pred)):
        w[labels_pred[i], labels_true[i]] += 1

    # 该API是求最少代价，而我们想要的是最大收益，因此需要将收益矩阵转化为代价矩阵传入
    mapping = scipy.optimize.linear_sum_assignment(w.max() - w)

    # 根据找到的映射对预测的标签进行转换
    old_pred, new_pred = mapping
    label_map = dict(zip(old_pred, new_pred))
    labels_pred = [label_map[x] for x in labels_pred]
    labels_pred = np.array(labels_pred)

    return labels_true, labels_pred
```

匈牙利算法寻找最优映射

外部指标

除了上述基于样本对的聚类效果评估方式，也可以从样本分布的角度来评估聚类效果。

标准化互信息（Normalized Mutual Information，NMI）：

$$NMI(Y,C) = \frac{I(Y,C)}{\frac{1}{2}[H(Y)+H(C)]}$$

其中 Y 为真实标签，C 为预测标签，I 是互信息度量，H 是熵。
取值范围为 $[0,1]$，值越大效果越好。

目前较为常用的外部指标是归一化随机指数 ARI，标准化互信息 NMI 和聚类精度 ACC。

内部指标

外部指标依赖给定真实数据标签（往往难以获得），因此在没有真实标签的情况下可以通过内部指标度量聚类效果。

假设聚类结果的簇划分为 $C = \{C_1, c_2, ..., c_k\}$，定义

- 簇内样本平均距离

$$avg(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \le i < j \le |C|} dist(x_i, x_j)$$

- 簇内样本间最远距离

$$diam(C) = \max_{1 \le i < j \le |C|} dist(x_i, x_j)$$

内部指标

- 两个簇的最近样本之间距离

$$d_{min}(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} dist(x_i, x_j)$$

- 两个簇的中心点之间距离

$$d_{cen}(C_i, C_j) = dist(\mu_i, \mu_j)$$

其中，$dist(\cdot, \cdot)$ 用于计算两个样本之间的距离；μ 代表簇 C 的中心点 $\mu = \frac{1}{|C|} \sum_{1 \le i \le |C|} x_i$。

内部指标

基于以上参数，我们可以导出下面这些常用的聚类性能度量内部指标：

- DB 指数（Davies-Bouldin Index，简称 DBI）

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

- Dunn 指数（Dunn Index，简称 DI）

$$DI = \min_{1 \leq i \leq k} \left\{ \min_{j \neq i} \left(\frac{d_{min}(C_i, C_j)}{\max_{1 \leq l \leq k} diam(C_l)} \right) \right\}$$

显然，DBI 的值越小越好，而 DI 则相反，值越大越好。

经典聚类方法分类

根据经典聚类方法的核心思想，可以对其进行分类：

- 基于划分——K-means, K-medoids
- 基于层次——BIRCH, CURE
- 基于密度——DBSCAN, OPTICS
- 基于模型——GMM, COBWEB
- 基于图论——CLICK, MST
- 基于模糊理论——FCM, FCS
- ……

K-means 聚类

基本假设

对于每一个聚类，我们可以选出一个中心点（center），使得该类中的所有点到该中心点的距离小于到其他聚类中心的距离。

符合（左）与不符合（右）K-means 假设

K-means 聚类

K-means 的目标函数

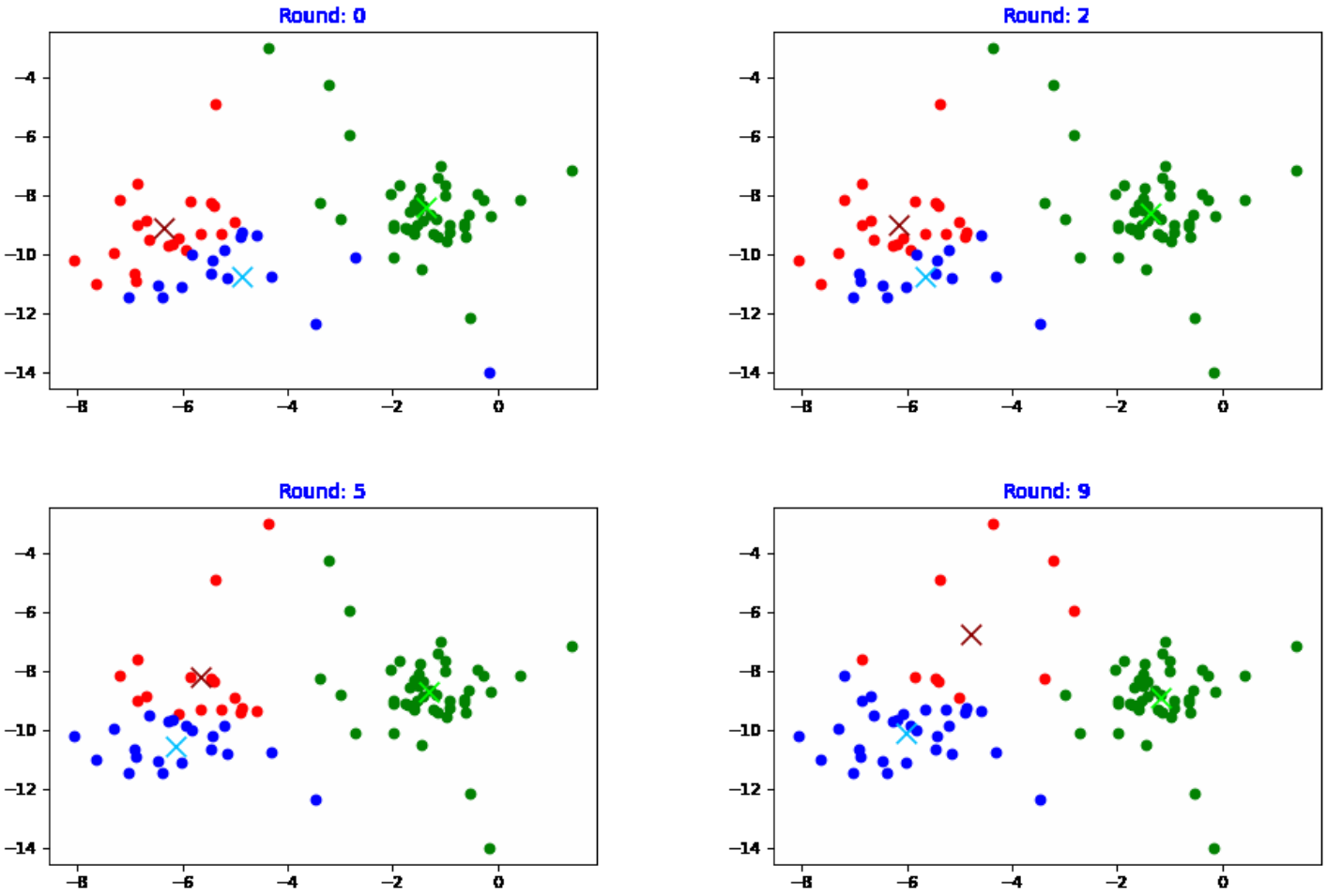
给定 N 个数据点，K-means 的优化目标可以描述为：

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| x_n - \mu_k \right\|^2$$

其中 $r_{nk} \in \{0, 1\}$ 表示第 n 个样本是否被分到第 k 个类，x_n 是第 n 个样本的表示，μ_k 是第 k 个类的中心。

直观理解：使每个样本到其对应的聚类中心的距离尽量近，即让最终的聚类结果尽量“紧凑”

K-means 聚类的优化



K-means 的优化过程

K-means 聚类的优化

K-means 算法伪代码

输入： N 个样本数据，聚类个数 k

输出： k 个聚类簇

1: 从数据集中随机选取 k 个样本作为初始聚类中心
2: **repeat**
3: 　　计算样本到每个聚类中心的距离，将样本分配到最近的那个聚类
4: 　　计算每个类的样本均值，作为新的聚类中心
5: **until** 聚类结果不再变化

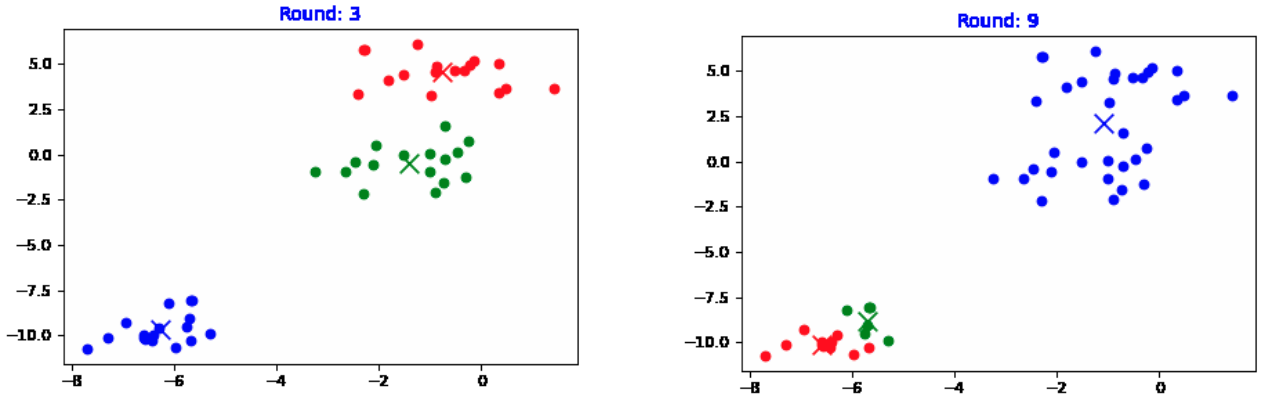
K 均值的平均复杂度是 $O(nkt)$，有一定的处理大数据的能力

思考

K-means 算法非常经典，是否存在缺陷？

K-means 聚类的缺陷

- 依赖初始化聚类中心（局部最优解）——一般可以用不同的初始化聚类中心



自然界的层次聚类

- 难以处理类别类型的特征，异常点影响——K-medoids
- 当数据量过大时，由于计算复杂度过大、迭代次数过多，会导致收敛速度非常慢——mini-batch & Kmeans++
- 依赖原始数据特征的质量——Deep Clustering

K-medoids 聚类

K-medoids 聚类算法

K-medoids 算法挑选实际的样本作为聚类的中心点，即将每个类用该类中的一个样本代表，其优化目标可以描述为：

$$E = \sum_{i=1}^{k} \sum_{p \in C_j} \operatorname{dist}(\boldsymbol{p}, \boldsymbol{o}_i)$$

优点

1. 当存在噪声和离群点时，K-medoids 算法更为鲁棒，不容易受到离群点或其他极端值的影响。
2. 不需要样本的自身特征，只需要样本之间的相似度

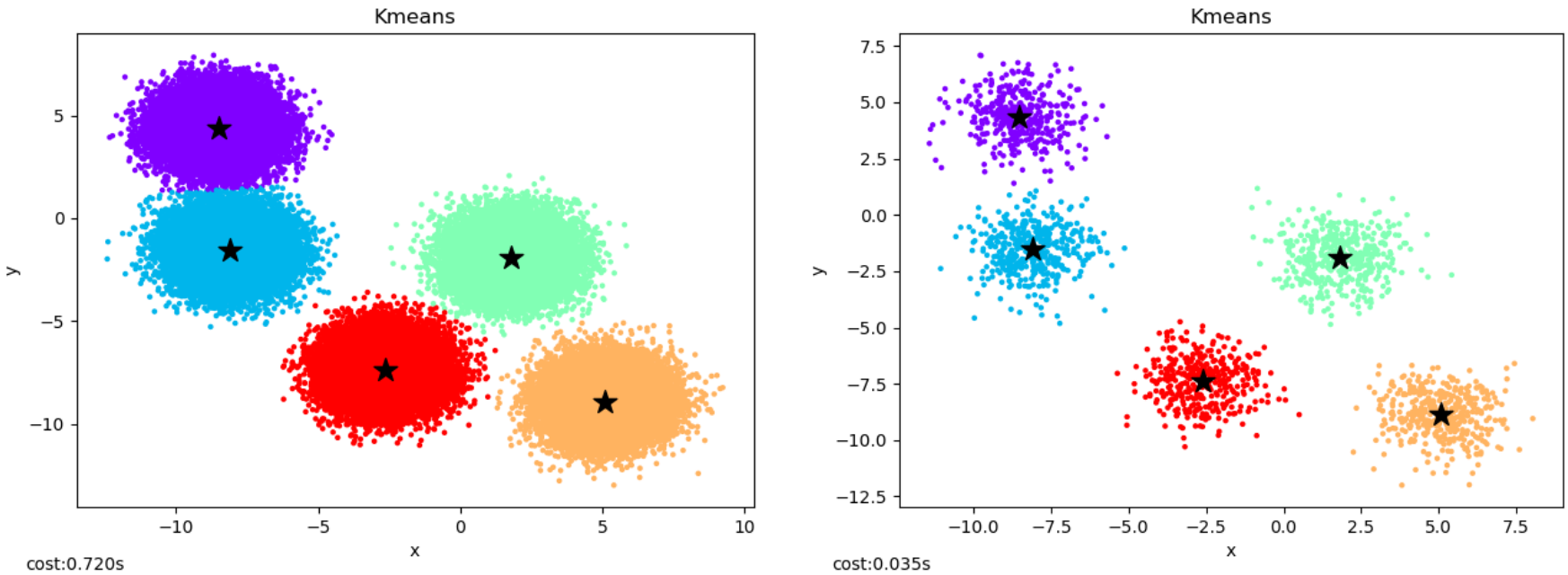
缺点

1. K-medoids 算法的复杂度是 $O(tk(n-k)^2)$，难以应用于大数据集。

Mini-batch K-means

Mini-batch K-means

Mini-batch K-means 通过从整体中进行抽样，随机选取出一小部分数据来代替整体，从而缩小数据规模提升训练效率。



从 100000 样本中抽取大小为 2000 的 mini-batch

Mini-batch Kmeans

优点

虽然 mini-batch 的原理非常简单，但它的实用性非常强，在机器学习领域广为使用。在大数据的场景下，几乎所有模型都需要做 mini-batch 优化。

缺点

随机选取也可能会导致一些问题。比如选取出的样本都在一个簇中，从理论上看这确实是有可能的。为了谨慎起见，我们可以重复多次采样，再对得到的多轮簇中心计算均值，直到簇中心趋于稳定为止。

K-means++

K-means++

K-means++ 通过选择更优的迭代起始位置，来减少收敛所需要的迭代次数。

Mini-batch Kmeans 针对的是样本数量，而 K-means++ 则是从迭代次数的角度入手。
根据探索性实验，我们发现：

- 随机选择 K 个样本点作为起始的簇中心效果比随机生成 K 个坐标点更好
- 两个离得远的点属于不同簇的可能性比离得近的大。

K-means++

基于以上两点，我们可以整理出 K-means++ 的算法原理：

- 先随机选取一个样本点作为簇中心。
- 从剩下来的点中再随机出一个点作为下一个簇中心。距离当前所有簇中心越远的点被选中的概率越大，离得越近被随机到的概率越小。

$$P(x_i) = \frac{d(x_i)}{\sum_{j=1}^{n} d(x_j)}$$

其中 $d(x_i)$ 为样本点 x_i 到当前所有簇中心的最短距离。

- 重复上述过程，直到一共选出了 K 个簇中心为止。

非负矩阵分解

非负矩阵分解（Nonnegative Matrix Factorization，NMF）是一种常见的降维方法。
相比于传统的矩阵分解方法，非负约束在许多场景中非常重要（使得分解的结果有意义）

NMF 优化目标

$$\min_{W,H} f_k(W,H) \equiv \frac{1}{2}\left\|A - WH^T\right\|_F^2 \qquad \text{s.t.} \qquad W,H \geq 0$$

非负矩阵分解

K-means 优化目标

$$J_k = \sum_{j=1}^{k} \sum_{a_i \in C_j} \|a_i - c_j\|^2 = \left\|A - CB^T\right\|_F^2$$

其中 $A \in \mathcal{R}^{m \times n}$ 是原始特征矩阵，$C \in \mathcal{R}^{m \times k}$ 是聚类中心矩阵，$B \in \mathcal{R}^{n \times k}$ 是聚类分配矩阵（每一行只有一个元素为 1，其他为 0）。

定义新的变量：

$$D^{-1} = \operatorname{diag}\left(\frac{1}{|C_1|}, \frac{1}{|C_2|}, \cdots, \frac{1}{|C_k|}\right) \in \mathbf{R}^{k \times k}$$

K-means 的优化目标转化为：

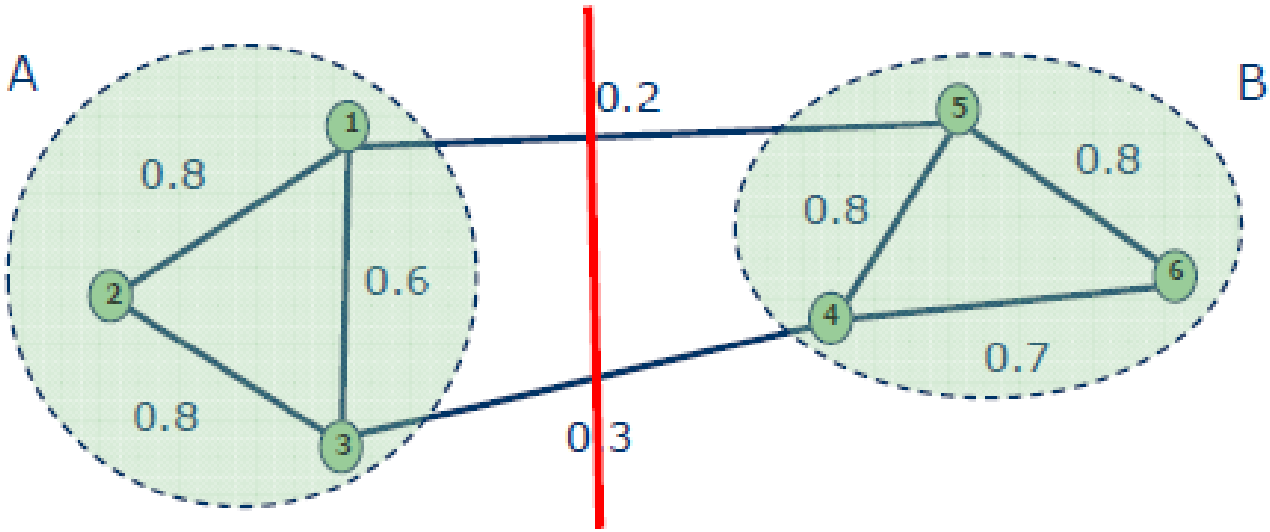
$$J_k = \left\|A - ABD^{-1}B^T\right\|_F^2$$

谱聚类 (Spectral Clustering)

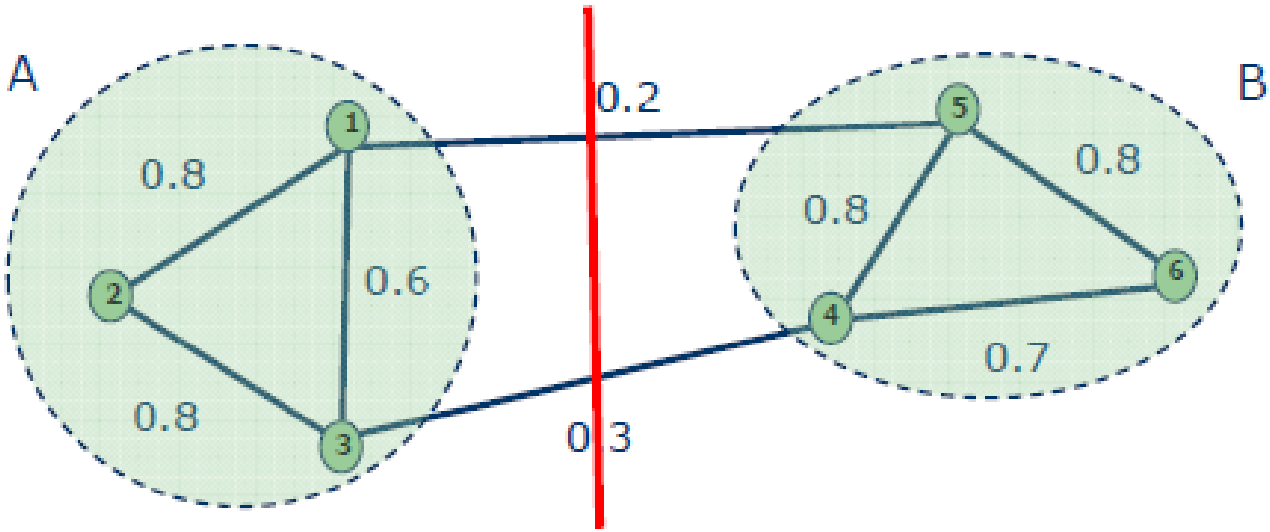
给定一个无向有权图：

$$G(V, E)$$

如下图所示，谱聚类就是一个 bi-partition 任务，希望划分成两个群体。



谱聚类 (Spectral Clustering)

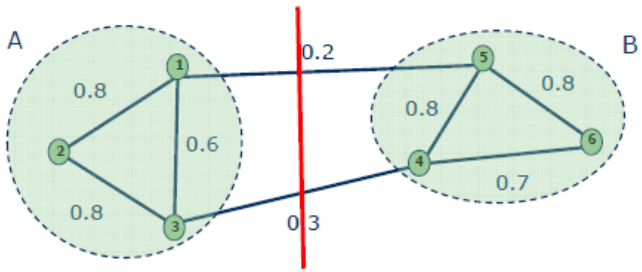


回顾聚类的评价标准：

1. 类内的样本要尽可能地相似
2. 不同类的样本要尽可能地不相似

谱聚类 (Spectral Clustering)

用邻接矩阵来表示这个图：



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.2	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.3	0	0
x_4	0	0	0.3	0	0.8	0.7
x_5	0.2	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

谱聚类 (Spectral Clustering)

前文提到聚类的优化目标是：

1. 类内的样本要尽可能地相似（子图之间的连边权重之和越少越好）
2. 不同类的样本要尽可能地不相似（子图内的权重越大越好）

于是我们可以在图上定义如下两个指标，A、B 分别是两个子图：

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

$$\mathrm{assoc}(A,A) = \sum_{i\in A, j\in A} w_{ij}$$

谱聚类 (Spectral Clustering)

自然，我们有了以下目标：

1. $$\min \operatorname{cut}(A, B)$$
2. $$\max(\operatorname{assoc}(A, A) + \operatorname{assoc}(B, B))$$

基于这两个目标，(Shi & Malik,'97) 构建了 Normalized-cut 目标：

$$\operatorname{Ncut}(A, B) = \frac{\operatorname{cut}(A, B)}{\operatorname{assoc}(A, V)} + \frac{\operatorname{cut}(A, B)}{\operatorname{assoc}(B, V)}$$

- Normalize the association between groups.

$$\operatorname{assoc}(A, V) = \sum_{i \in A, j \in V} w_{ij}$$

N-cut 算法的目标函数如下

$$\min \operatorname{Ncut}(A, B)$$

谱聚类 (Spectral Clustering)

N-cut 的形式化描述: [1]

$$x \in [1,-1]^n, x_i = \begin{cases} 1 & i \in A \\ -1 & i \in B \end{cases} \quad d_i = \sum_j w_{ij}$$

$$\begin{aligned} \mathrm{Ncut}(A,B) &= \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(B,V)} \\ &= \frac{\mathrm{assoc}(A,V) - \mathrm{assoc}(A,A)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{assoc}(B,V) - \mathrm{assoc}(B,B)}{\mathrm{assoc}(B,V)} \\ &= \frac{\sum_{x_i>0, x_j<0} -w_{ij}x_i x_j}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0, x_j>0} -w_{ij}x_i x_j}{\sum_{x_i<0} d_i} \end{aligned}$$

[1] https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf

谱聚类 (Spectral Clustering)

定义如下矩阵：

$$W \in R^{n\times n} \quad D \in R^{n\times n} \quad x \in [1,-1]^n \quad \mathbf{1} \in [1]^n \quad k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$$

N-cut 的矩阵形式描述：

$$\mathrm{Ncut}(A,B) = \frac{(\mathbf{1}+x)^T(D-W)(\mathbf{1}+x)}{k\mathbf{1}^T D\mathbf{1}} + \frac{(\mathbf{1}-\boldsymbol{x})^T(D-W)(\mathbf{1}-\boldsymbol{x})}{(1-k)\mathbf{1}^T D\mathbf{1}}$$

$$b = \frac{k}{1-k}$$

$$= \frac{[(\mathbf{1}+\boldsymbol{x})-b(1-\boldsymbol{x})]^T(D-W)[(\mathbf{1}+\boldsymbol{x})-b(\mathbf{1}-\boldsymbol{x})]}{b\mathbf{1}^T D\mathbf{1}}$$

- 我们成功地把 Ncut 用图的邻接表和度矩阵表示了出来

谱聚类 (Spectral Clustering)-图拉普拉斯

引入中间变量进一步化简:

$$y = (\mathbf{1} + x) - b(\mathbf{1} - x) \quad k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i} \quad b = \frac{k}{1-k} = \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i}$$

得:

$$\boldsymbol{y}^T D\mathbf{1} = 2\sum_{x_i>0} d_i - 2b\sum_{x_i<0} d_i = 0$$

$$\boldsymbol{y}^T D\boldsymbol{y} = 4\sum_{x_i>0} d_i + 4b^2\sum_{x_i<0} d_i = 4\left(b\sum_{x_i<0} d_i + b^2\sum_{x_i<0} d_i\right)$$

$$= 4b\left(\sum_{x_i<0} d_i + b\sum_{x_i<0} d_i\right) = 4b\mathbf{1}^T D\mathbf{1}$$

谱聚类 (Spectral Clustering)-图拉普拉斯

最终我们得到了 Ncut 目标约束优化的形式化表达：

$$\min_{\boldsymbol{x}} \mathrm{Ncut}(\boldsymbol{x}) = \min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(D-W)\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}$$

$$\text{s.t.} \quad \boldsymbol{y} \in [2, -2b]^n, \quad \boldsymbol{y}^T D\mathbf{1} = 0$$

上式难以进行直接优化，对约束条件做如下松弛：

$$\min_{y} \frac{\boldsymbol{y}^T(D-W)\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}, \boldsymbol{y} \in \mathcal{R}^n, \boldsymbol{y}^T D\mathbf{1} = 0$$

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T L\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}, \boldsymbol{y} \in \mathcal{R}^n, \boldsymbol{y}^T D\mathbf{1} = 0$$

谱聚类 (Spectral Clustering)-图拉普拉斯

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T L \boldsymbol{y}}{\boldsymbol{y}^T D \boldsymbol{y}}, \boldsymbol{y} \in \mathcal{R}^n, \boldsymbol{y}^T D \mathbf{1} = 0$$

转化为拉格朗日函数:

$$J(y) = y^T L y - \lambda \left(y^T D y - c \right)$$

$$\frac{\partial J(y)}{\partial y} = 0 \Rightarrow Ly = \lambda D y$$

上述放缩后的最小化问题可以通过解特征值特征向量

$$(\mathbf{D} - \mathbf{W})\boldsymbol{y} = \lambda \mathbf{D} y$$

获得解析解

谱聚类 (Spectral Clustering)-图拉普拉斯

适当变形：这里的 f 是一个向量

$$Ly = \lambda Dy$$

令：$y = D^{-\frac{1}{2}} f$

$$\Leftrightarrow LD^{-\frac{1}{2}} f = \lambda D^{\frac{1}{2}} f$$

$$\Leftrightarrow D^{-\frac{1}{2}} LD^{-\frac{1}{2}} f = \lambda f$$

这个时候表达式就是:

$$R(y) = \frac{f^T \left[D^{-\frac{1}{2}} LD^{-\frac{1}{2}} \right] f}{f^T f}, \quad y = D^{-\frac{1}{2}} f$$

这个目标函数也有个名字，叫做 Rayleigh quotient（瑞丽商）[2]

[2] https://www.planetmath.org/RayleighRitzTheorem

谱聚类 (Spectral Clustering)-图拉普拉斯

别忘了刚才的约束优化式还有约束：

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T L \boldsymbol{y}}{\boldsymbol{y}^T D \boldsymbol{y}}, \boldsymbol{y} \in \mathcal{R}^n, \boldsymbol{y}^T D \mathbf{1} = 0$$

全 1 向量 1 是 0（最小的特征值）特征值对应的特征向量。

$$(D - W)y = \lambda D^{\frac{1}{2}} D^{\frac{1}{2}} y$$

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} D^{\frac{1}{2}} y = \lambda D^{\frac{1}{2}} y$$

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} f = \lambda f$$

由于特征向量两两正交，所以当 y 不是 1 的时候该约束总成立

$$f_1^T f_2 = 0 \Rightarrow \left(D^{\frac{1}{2}} y_1\right)^T \left(D^{\frac{1}{2}} \boldsymbol{y}_2\right) = 0 \Rightarrow \boldsymbol{y}_1^T D \boldsymbol{y}_2 = 0$$

谱聚类 (Spectral Clustering)-图拉普拉斯

1. 为了满足第二个约束，所以 y 只能取除 0 以外的特征值对应的特征向量，于是为了最小化目标函数，我们取第二小的特征值对应的特征向量。
2. $$y = (\mathbf{1} + x) - b(\mathbf{1} - x) \quad k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i} \quad b = \frac{k}{1-k} = \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i}$$

   把实数向量 y 按照正负离散化，根据前面的这个等式，我们可以反推出 X，也就是 A,B 的子图划分。至此，我们在多项式时间内得到了 Ncut 问题的近似解（我们进行了一次约束宽松）
3. 如果 K 大于 2 呢？

谱聚类 (Spectral Clustering)-图拉普拉斯

如果 K 大于 2 呢？

1. 递归地进行二分，直到分出 K 个子图
2. 按特征值排序，从第二小开始，从小到大选取多个特征向量，组成：

$$Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k] \in R^{n \times k}$$

将这个矩阵视为新的样本矩阵，然后用 Kmeans 对新的低维样本进行聚类。

谱聚类 (Spectral Clustering)-图拉普拉斯

1. 这一不可思议的步骤叫做：拉普拉斯降维（Laplacian Eigenmaps），可以归类到流形学习。其思想是相互之间有关系的点（在图中相连的点），降维之后足够接近；没有关系的点，降维之后足够远。
2. 形式化一下

$$\min \sum_{i,j} W_{ij} \left(y_i - y_j\right)^2$$

其中 y 是降维后的样本特征向量，W 是根据降维前的样本构建的邻接矩阵。

谱聚类 (Spectral Clustering)-图拉普拉斯

1. 把刚才的方程展开

$$-2\sum_{i,j} W_{ij}y_i y_j + \sum_{i,j} W_{ij}y_i^2 + \sum_{i,j} W_{ij}y_j^2$$

按照基本的线性代数知识，

$$\sum_{i,j} W_{ij}y_i y_j = y^T W y \quad \sum_{i,j} W_{ij}y_i^2 = y^T D y$$

D 只有对角线上有值，代表数据点所连的 weight 之和，也就是度矩阵。

2. 目标变为最小化: $y^T L y, L = D - W$

谱聚类 (Spectral Clustering)-图拉普拉斯

总结一下谱聚类的流程：

1. 根据输入的相似矩阵的生成方式构建样本的相似矩阵 S
2. 根据相似矩阵 S 构建邻接矩阵 W，构建度矩阵 D
3. 计算出拉普拉斯矩阵 L
4. 构建标准化后的拉普拉斯矩阵 $D^{-1/2}LD^{-1/2}$
5. 计算其最小的 k1 个特征值所各自对应的特征向量 f，最终组成 n×k1 维的特征矩阵 F
6. 对 F 中的每一行作为一个 k1 维的样本，共 n 个样本，用其他聚类方法进行聚类，聚类维数为 k

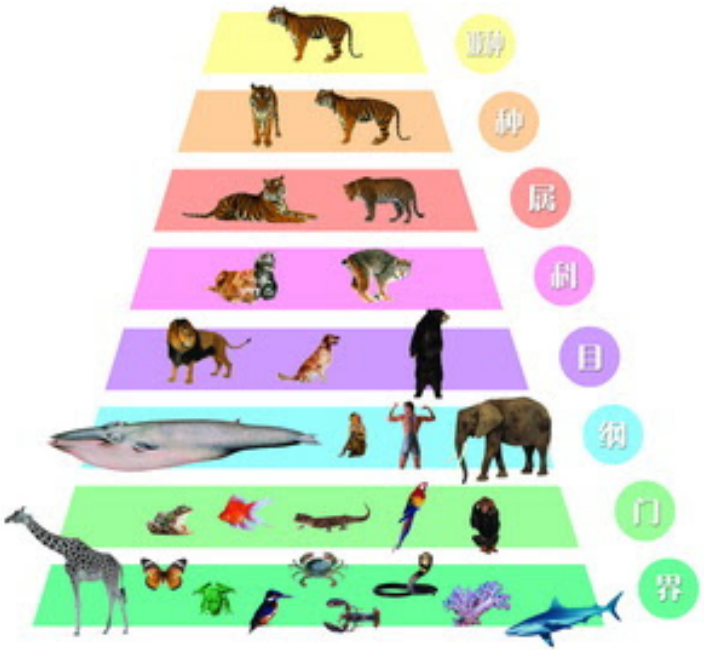
谱聚类 (Spectral Clustering)

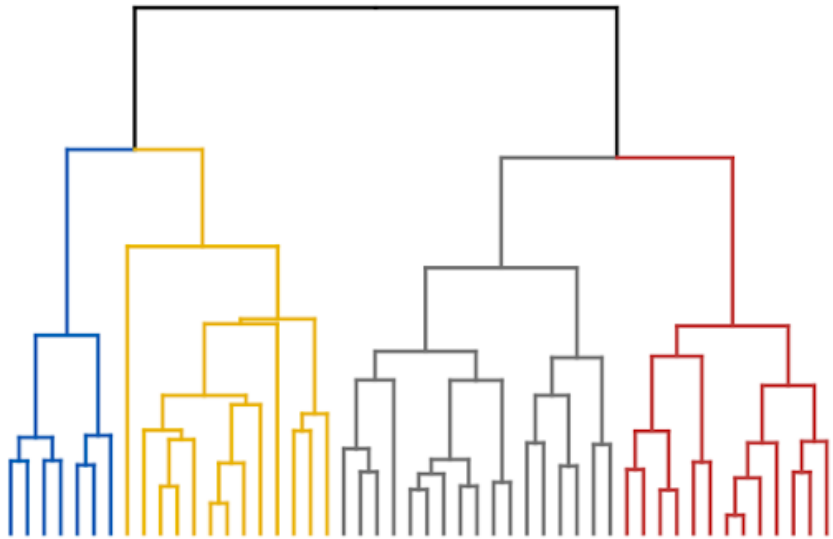
```python
import numpy as np
def calLaplacianMatrix(adjacentMatrix):

    # compute the Degree Matrix: D=sum(A)
    degreeMatrix = np.sum(adjacentMatrix, axis=1)

    # compute the Laplacian Matrix: L=D-A
    laplacianMatrix = np.diag(degreeMatrix) - adjacentMatrix

    # normailze
    # D^(-1/2) L D^(-1/2)
    sqrtDegreeMatrix = np.diag(1.0 / (degreeMatrix ** (0.5)))
    #return laplacianMatrix
    return np.dot(np.dot(sqrtDegreeMatrix, laplacianMatrix), sqrtDegreeMatrix)
W=[[0. , 0.8, 0.6, 0. , 0.2, 0. ],
       [0.8, 0. , 0.8, 0. , 0. , 0. ],
       [0.6, 0.8, 0. , 0.3, 0. , 0. ],
       [0. , 0. , 0.3, 0. , 0.8, 0.7],
       [0.2, 0. , 0. , 0.8, 0. , 0.8],
       [0. , 0. , 0. , 0.7, 0.8, 0. ]]
V,Q=np.linalg.eig(calLaplacianMatrix(W))
print(V)
print(Q)
```

[5] ✓ 3.6s Python

```
[1.11022302e-16 1.85262778e-01 1.24679950e+00 1.44192115e+00
 1.55873672e+00 1.56727985e+00]
[[-0.4        -0.3932796  -0.51994042 -0.3420326  -0.41294659  0.35704613]
 [-0.4        -0.46407639 -0.07710571  0.36787718  0.02096506 -0.69492061]
 [-0.41231056 -0.37409043  0.57517294  0.0174069   0.40150668  0.44465922]
 [-0.42426407  0.36968928  0.49333048 -0.35980757 -0.49140162 -0.26271245]
 [-0.42426407  0.40486716 -0.35957234 -0.33194288  0.62940748 -0.14300516]
 [-0.38729833  0.43523905 -0.14221648  0.71255141 -0.17377756  0.32002075]]
```

层次聚类





自然界的层次聚类

层次聚类

K-means 和 K-medoids 等基于划分的聚类方法与人类直观聚类过程不同，人类的直觉划分往往是自底向上的。

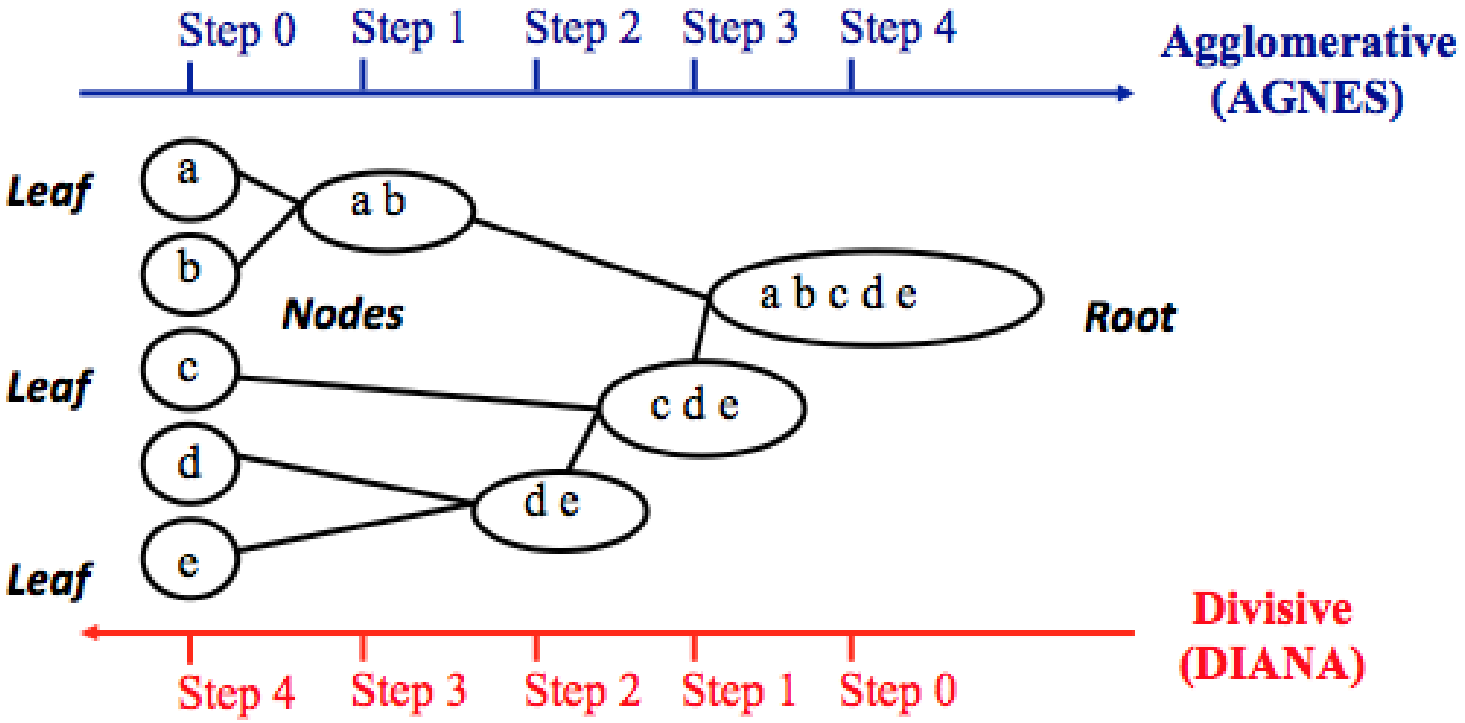
层次聚类

层次聚类（Hierarchical Clustering）的一类将数据对象组成层次结构或簇的“树”的方法的总称。

层次聚类的两大类主要方法：

1. 凝聚 (Agglomerative) 层次聚类：从每个对象都作为一个簇开始，迭代地合并，形成更大的簇。
2. 分裂 (Divisive) 层次聚类：从所有对象作为一个簇开始，迭代地分裂，形成较小的簇。

层次聚类


Step 0
Step 1
Step 2
Step 3
Step 4
Agglomerative
(AGNES)
Leaf
Leaf
Leaf
a
b
c
d
e
a b
c d e
d e
a b c d e
Nodes
Root
Divisive
(DIANA)
Step 4
Step 3
Step 2
Step 1
Step 0

层次聚类

层次聚类的核心问题是度量两个簇之间的距离，常用的距离有如下四种：

1. **最小距离**：$\mathrm{dist}_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{|\boldsymbol{p} - \boldsymbol{p}'|\}$
2. **最大距离**：$\mathrm{dist}_{\max}(C_i, C_j) = \max_{p \in C_i, \boldsymbol{p}' \in C_j} \{|\boldsymbol{p} - \boldsymbol{p}'|\}$
3. **均值距离**：$\mathrm{dist}_{\mathsf{mean}}(C_i, C_j) = |m_i - m_j|$
4. **平均距离**：$\mathrm{dist}_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'|$

最近邻层次聚类算法

最近邻（层次）聚类算法

使用最小距离 $\mathrm{dist}_{\min}(C_i, C_j)$ 作为距离度量的层次聚类算法称为最近邻层次聚类算法（Nearest-neighbor Clustering Algorithm）。

使用最小距离度量的凝聚层次聚类算法也被称为最小生成树算法（minimal spanning tree algorithm），即生成一棵连接所有节点的树且这课树具有最小的边权和。

常见的算法包括 Prim 算法和 Kruskal 算法。

经典层次聚类算法的缺陷

1. 计算复杂度高，难以应用于大规模数据集
2. 节点凝聚/拆分后难以回撤
3. 对于新数据难以快速计算聚类结果

聚类特征 CF

聚类特征 CF

给定 n 个 d 维样本组成的聚类簇，聚类特征 Clustering Feature 定义为一个三维向量：

$$CF = \langle n, LS, SS \rangle$$

其中 LS 是 n 个样本的线性和 $LS = \sum_{i=1}^{n} x_i$，SS 是数据点的平方和 $SS = \sum_{i=1}^{n} x_i^2$。

聚类特征 CF 的优点：

1. 使用聚类特征 CF 来描述簇可以避免存储个体样本的详细特征。
2. 只需要固定大小的空间来存放聚类特征。
3. 聚类特征满足可加性。

聚类特征 CF

聚类特征 CF 可以推导出簇相关的统计量：

1. 聚类的中心

$$x_0 = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{LS}{n}$$

2. 聚类的半径

$$R = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_0)^2}{n}} = \sqrt{\frac{nSS - 2LS^2 + nLS}{n^2}}$$

3. 聚类的直径

$$D = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}$$

CF Tree

CF Tree 的三个参数：

1. 每个内部节点的最大 CF 数 B
2. 每个叶子节点的最大 CF 数 L
3. 叶节点每个 CF 的最大样本半径阈值 T

CF Tree 的构造步骤：

1. 从训练集读入首个样本点并作为根节点
2. 对于数据集中的其他节点，每次判断是否能找到满足最大样本半径均值的 CF。如能，则并入对应的 CF，否则构建新的 CF
3. 如果叶子节点的 CF 数超过 L，则进行叶子节点分裂
4. 如果内部节点的 CF 数超过 B，则进行内部节点分裂

BIRCH 聚类算法

BIRCH 聚类算法

利用层次结构的平衡迭代归约和聚类（Balanced Iterative Reducing and Clustering using Hierarchies）是一种融合了层次聚类和迭代划分聚类的算法。它使用聚类特征树（Clustering Feature Tree，CF-树）来概括一个簇，在速度和效率上具有显著的优势

BIRCH 聚类算法的优点：

1. BIRCH 能够识别出数据集中数据分布的不均衡性，将分布在稠密区域中的点聚类，将分布在稀疏区域中的点视作异常点而移除
2. BIRCH 只需要单遍扫描数据集就能建立 CF Tree 并进行聚类，效率高，复杂度是 $O(n)$
3. 对异常数据不敏感，可以直接将 CF Tree 中包含数据少的 MinCluster 作为异常点
4. CF Tree 高度平衡，因此可以在树上快速进行插入或查找操作

高斯混合模型

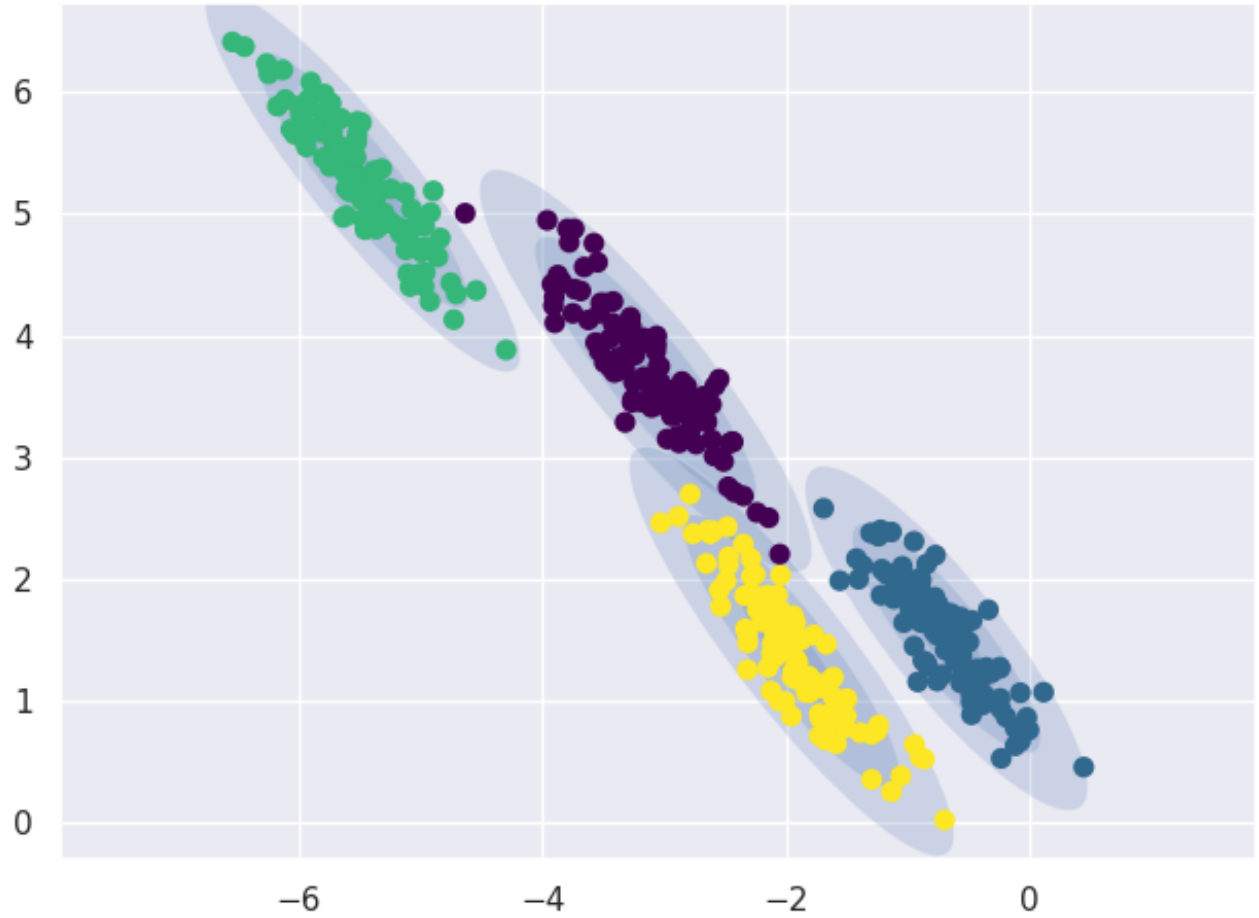
高斯混合模型（Gaussian Mixture Model）

高斯混合模型（Gaussian Mixture Model）聚类是一种基于模型的聚类方法，其核心思想为：假设当前的样本集都是由一个概率生成模型产生的，而这个概率模型则是由多个基础的高斯分布通过线性组合混合所得到。

GMM 的一般过程可以概括为：

- 通过线性组合分布的方式构造概率模型
- 通过概率生成过程来学习参数
- 通过数据点属于哪一个组件分布来决定聚类的标签

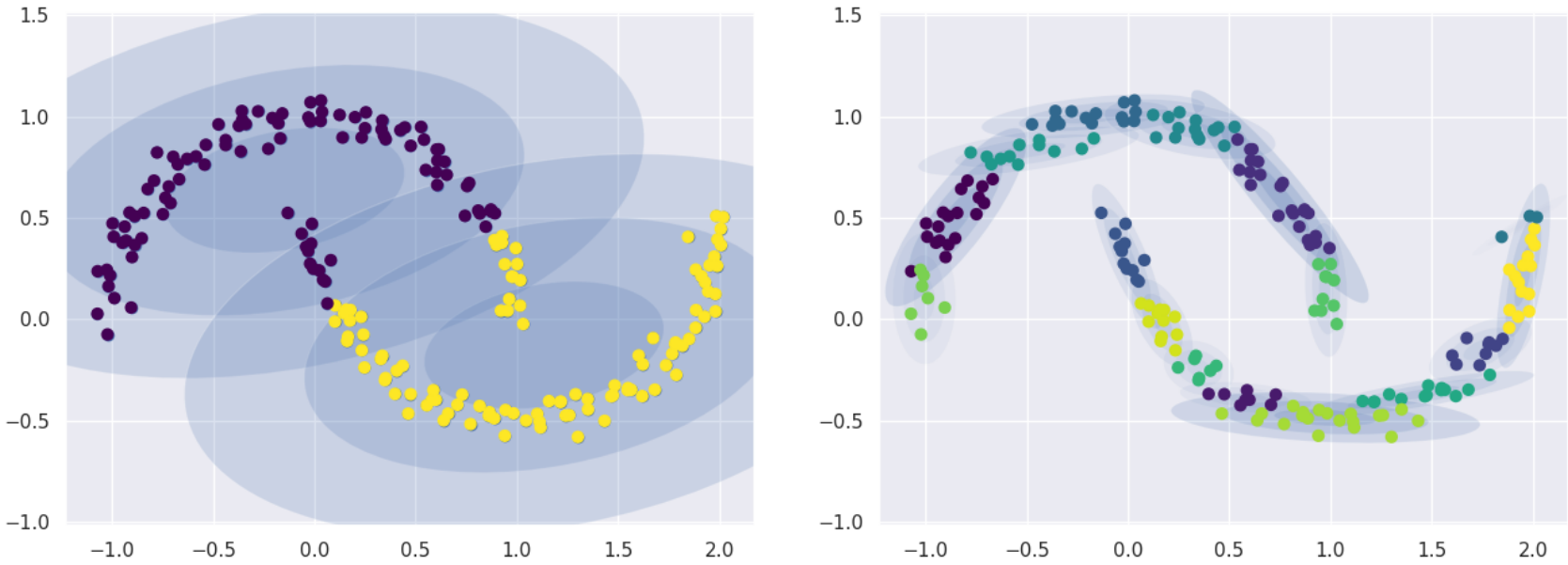
高斯混合模型



通过由多个高斯分布混合而得的模型来拟合样本分布

高斯混合模型

对于一些特殊的分布，使用 K 个高斯分布混合可能会得到较差的效果。而使用更多数量的分布可以找到一个更接近样本数据的拟合结果。



2 个与 16 个高斯分布混合模型的拟合结果

高斯混合模型

高斯分布

高斯分布（正态分布）：对于 n 维样本空间 χ 中的随机向量 x，若 x 服从高斯分布，其概率密度函数为：

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{1}$$

其中 μ 是 n 维均值向量，Σ 是 $n \times n$ 的协方差矩阵。

高斯混合分布

$$p_{\mathcal{M}} = \sum_{i=1}^{k} \alpha_i \cdot p(x|\mu_i, \Sigma_i) \tag{2}$$

该分布共由 k 个高斯分布混合组成，其中 μ_i 与 Σ_i 是第 i 个高斯混合成分的参数，而 $\alpha_i > 0$ 为相应的**混合系数**，$\sum_{i=1}^{k} \alpha_i = 1$。

高斯混合模型

若训练集 $D = \{x_1, x_2, ..., x_n\}$ 由上述过程生成，令随机变量 $z_j \in \{1, 2, ..., k\}$ 表示生成样本 x_j 的高斯混合成分，其取值未知。z_j 的先验概率 $P(z_j = i)$ 对应于 $\alpha_i (i = z, 2, ..., k)$。

根据贝叶斯定理，z_j 的后验分布对应于：

$$
\begin{aligned}
p_{\mathcal{M}}(z_j = i|x_j) &= \frac{P(z_j = i) \cdot p_{\mathcal{M}}(x_j|z_j = i)}{p_{\mathcal{M}}(x_j)} \\
&= \frac{\alpha_i \cdot p(x_j|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(x_j|\mu_l, \Sigma_l)}
\end{aligned} \tag{3}
$$

换言之，$p_{\mathcal{M}}(z_j = i|x_j)$ 给出了样本 x_j 由第 i 个高斯混合成分生成的后验概率。简记为 $\gamma_{ji} (i = 1, 2, ..., k)$。

高斯混合模型

高斯混合模型优化目标

给定样本集 D，我们可采用极大似然估计，即最大化对数似然

$$\begin{aligned} LL(D) &= \ln\left(\prod_{j=1}^{n} p_{\mathcal{M}}(x_j)\right) \\ &= \sum_{j=1}^{n}\left(\sum_{i=1}^{k} \alpha_i \cdot p(x_j|\mu_j, \Sigma_i)\right) \end{aligned} \tag{5}$$

模型参数 $\{(\alpha_i, \mu_i, \Sigma_i)|1 \leq i \leq k\}$ 如何求解？

高斯混合模型的 EM 优化

若参数 $\{(\alpha_i, \mu_i, \Sigma_i)|1 \leq i \leq k\}$ 能使式 (5) 最大化，则由 $\frac{\partial LL(D)}{\partial \mu_i} = 0$ 有

$$\sum_{j=1}^{n} \frac{\alpha_i \cdot p(x_j|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(x_j|\mu_l, \Sigma_l)}(x_j - \mu_i) = 0 \tag{6}$$

由式 (3) 以及 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i|x_j)$，有

$$\mu_i = \frac{\sum_{j=1}^{n} \gamma_{ji} x_j}{\sum_{j=1}^{n} \gamma_{ji}} \tag{7}$$

即各混合成分的均值可通过样本加权平均来估计，样本权重是每个样本属于该成分的后验概率。

GMM

类似的，由 $\frac{\partial LL(D)}{\partial \Sigma_i} = 0$ 可得

$$\Sigma_i = \frac{\sum_{j=1}^{n} \gamma_{ji}(x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^{n} \gamma_{ji}} \tag{8}$$

GMM

对于混合系数 α_i，除了要最大化 $LL(D)$，还需满足 $\alpha_i \geq 0, \sum_{i=1}^{k} = 1$，即

$$\underset{\alpha_i}{max}\, LL(D) \quad s.t. \quad \alpha_i \geq 0, \sum_{i=1}^{k} = 1$$

转化为拉格朗日函数：

$$LL(D) + \lambda \left(\sum_{i=1}^{k} \alpha_i - 1 \right) \tag{9}$$

其中 λ 为拉格朗日乘子。

GMM

由上式对 α_i 的导数为 0，有

$$\sum_{j=1}^{n} \frac{p(x_j|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(x_j|\mu_l, \Sigma_l)} + \lambda = 0 \tag{10}$$

两边同乘以 α_i，对所有样本求和可得 $\lambda = -n$，有

$$\alpha_i = \frac{1}{n} \sum_{j=1}^{n} \gamma_{ji} \tag{11}$$

即每个高斯成分的混合系数由样本属于该成分的平均后验概率确定。

GMM

由上述推导，我们可以获得高斯混合模型的 EM 算法：

- E 步：根据当前参数来计算每个样本属于每个高斯成分的后验概率 γ_{ji}。
- M 步：根据式 (7)(8) 和 (11) 更新模型参数 $\{(\alpha_i, \mu_i, \Sigma_i)|1 \leq i \leq k\}$。

GMM

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\mathsf{n}}\}$;
高斯混合成分个数 k.

过程:

初始化高斯混合分布的模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$
repeat
  for $j = 1, 2, \ldots, \mathsf{n}$ **do**
    根据式(3)计算 $\boldsymbol{x}_j$ 由各混合成分生成的后验概率, 即 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \boldsymbol{x}_j)\ (1 \leqslant i \leqslant k)$
  end for
  for $i = 1, 2, \ldots, k$ **do**
    计算新均值向量: $\boldsymbol{\mu}_i' = \frac{\sum_{j=1}^{\mathsf{n}} \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^{\mathsf{n}} \gamma_{ji}}$;
    计算新协方差矩阵: $\boldsymbol{\Sigma}_i' = \frac{\sum_{j=1}^{\mathsf{n}} \gamma_{ji} (\boldsymbol{x}_j - \boldsymbol{\mu}_i')(\boldsymbol{x}_j - \boldsymbol{\mu}_i')^{\mathrm{T}}}{\sum_{j=1}^{\mathsf{n}} \gamma_{ji}}$;
    计算新混合系数: $\alpha_i' = \frac{\sum_{j=1}^{\mathsf{n}} \gamma_{ji}}{\mathsf{n}}$;
  end for
  将模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$ 更新为 $\{(\alpha_i', \boldsymbol{\mu}_i', \boldsymbol{\Sigma}_i') \mid 1 \leqslant i \leqslant k\}$
until 满足停止条件
$C_i = \varnothing\ (1 \leqslant i \leqslant k)$
for $j = 1, 2, \ldots, \mathsf{n}$ **do**
  根据式(4)确定 $\boldsymbol{x}_j$ 的簇标记 λ_j;
  将 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \bigcup \{\boldsymbol{x}_j\}$
end for

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

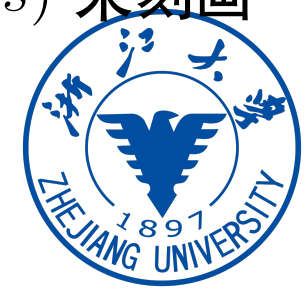
高斯混合聚类算法

DBSCAN 聚类

基于密度的聚类

由于聚类的类内相对稠密，而类间相对稀疏，因此可以将每个类视作一个密度相对较高的区域。基于密度的聚类方法从样本密度的角度来考察样本之间的可连接性，并基于可连接样本不断扩展聚类簇以获得最终的聚类结果。

Density-Based Spatial Clustering of Applications with Noise，DBSCAN，是一种著名的密度聚类算法，它基于一组邻域参数 $(\epsilon, MinPts)$ 来刻画样本分布的紧密程度。



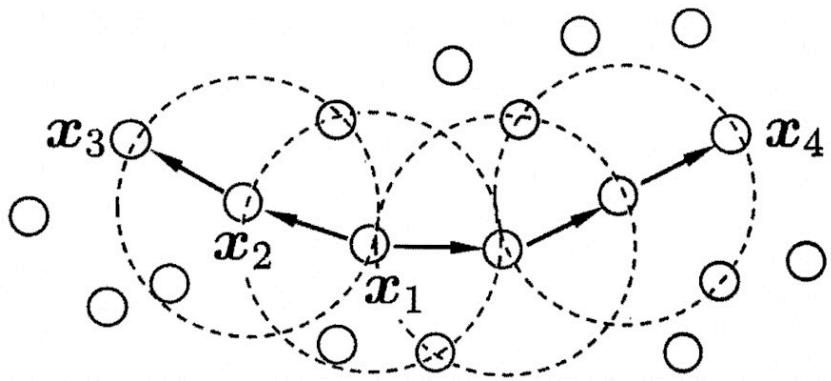
DBSCAN 聚类

给定数据集 $D = \{x_1, x_2, ..., x_N\}$，定义以下几个概念：

- ϵ-邻域：对于 $x_j \in D$，其 ϵ-邻域包含样本集 D 中与 x_j 的距离不大于 ϵ 的样本，即 $N_\epsilon(x_j) = \{x_i \in D | dist(x_i, x_j) \leq \epsilon\}$；
- 核心对象：若 x_j 的 ϵ-邻域至少包含 $MinPts$ 个样本，即 $|N_\epsilon(x_j)| \geq MinPts$，则 x_j 是一个核心对象；
- 密度直达：若 x_j 位于 x_i 的 ϵ-邻域中，且 x_i 是核心对象，则称 x_j 由 x_i 密度直达；

DBSCAN 聚类

- 密度可达：对 x_i 与 x_j，若存在样本序列 $p_1, p_2, ..., p_n$，其中 $p_1 = x_i, p_n = x_j$ 且 p_{i+1} 由 p_i 密度直达，则称 x_j 由 x_i 密度可达；
- 密度相连：对 x_i 与 x_j，若存在 x_k 使得 x_i 与 x_j 均由 x_k 密度可达，则称 x_i 与 x_j 密度相连。



虚线为 ϵ-邻域，x_1 是核心对象，x_2 由 x_1 密度直达，x_3 由 x_4 密度相连。

DBSCAN 聚类

DBSCAN 将“簇”定义为：由密度可达关系导出的最大的密度相连样本集合。

给定邻域参数 $(\epsilon, MinPts)$，簇 $C \subseteq D$ 是满足以下性质的非空样本子集：

- 连接性：$x_i \in C$，$x_j \in C \Rightarrow x_i$ 与 x_j 密度相连
- 最大性：$x_i \in C$，x_j 由 x_i 密度可达 $\Rightarrow x_j \in C$

若 x 为核心对象，由 x 密度可达的所有样本组成的集合记为 $X = \{x' \in D | x'$由x密度可达$\}$。

因此，DBSCAN 算法先任选数据集中的一个核心对象作为”种子”，再由此出发确定相应的聚类簇。

DBSCAN 聚类

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
邻域参数 $(\epsilon, MinPts)$.

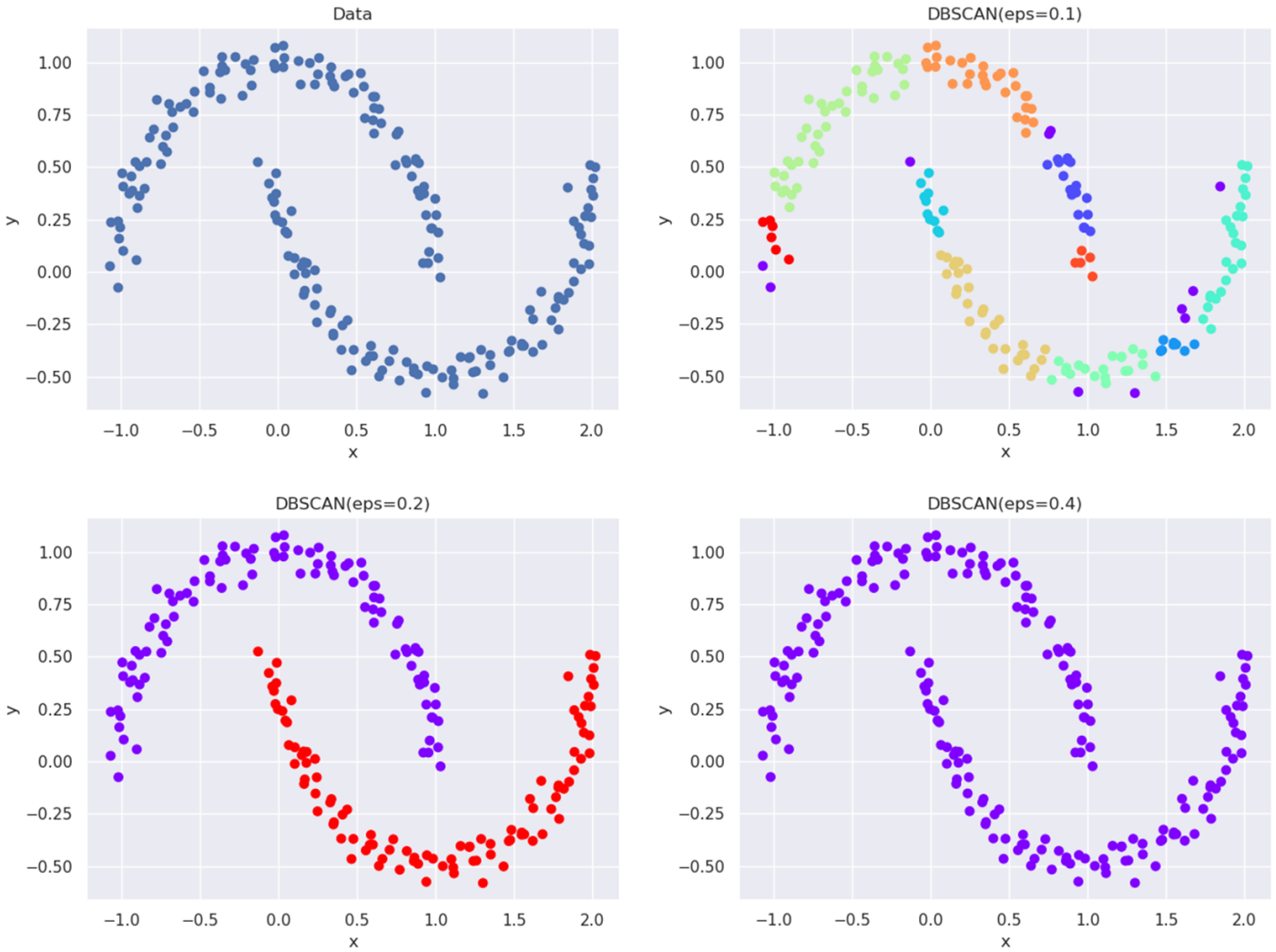
过程:

初始化核心对象集合: $\Omega = \varnothing$
for $j = 1, 2, \ldots, m$ **do**
  确定样本 $\boldsymbol{x}_j$ 的 ϵ-邻域 $N_\epsilon(\boldsymbol{x}_j)$;
  if $|N_\epsilon(\boldsymbol{x}_j)| \geqslant MinPts$ **then**
    将样本 $\boldsymbol{x}_j$ 加入核心对象集合: $\Omega = \Omega \bigcup \{\boldsymbol{x}_j\}$
  end if
end for
初始化聚类簇数: $k = 0$
初始化未访问样本集合: $\Gamma = D$
while $\Omega \neq \varnothing$ **do**
  记录当前未访问样本集合: $\Gamma_{\text{old}} = \Gamma$;
  随机选取一个核心对象 $\boldsymbol{o} \in \Omega$, 初始化队列 $Q =< \boldsymbol{o} >$;
  $\Gamma = \Gamma \setminus \{\boldsymbol{o}\}$;
  while $Q \neq \varnothing$ **do**
    取出队列 Q 中的首个样本 $\boldsymbol{q}$;
    if $|N_\epsilon(\boldsymbol{q})| \geqslant MinPts$ **then**
      令 $\Delta = N_\epsilon(\boldsymbol{q}) \bigcap \Gamma$;
      将 Δ 中的样本加入队列 Q;
      $\Gamma = \Gamma \setminus \Delta$;
    end if
  end while
  $k = k + 1$, 生成聚类簇 $C_k = \Gamma_{\text{old}} \setminus \Gamma$;
  $\Omega = \Omega \setminus C_k$
end while

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

DBSCAN 的算法流程

DBSCAN 聚类



不同参数设置下的 DBSCAN 聚类结果

本节课总结

1. 认识聚类
2. 聚类的性能指标
3. K-means 聚类
4. 谱聚类
5. 层次聚类
6. 高斯混合模型聚类
7. DBSCAN 聚类

参考文献

Xu, R., and Wunsch, D.
Survey of clustering algorithms.
IEEE Transactions on neural networks 16, 3 (2005), 645–678.