

数据挖掘与应用

聚类

授课教师：周晟

浙江大学 软件学院

2021.10



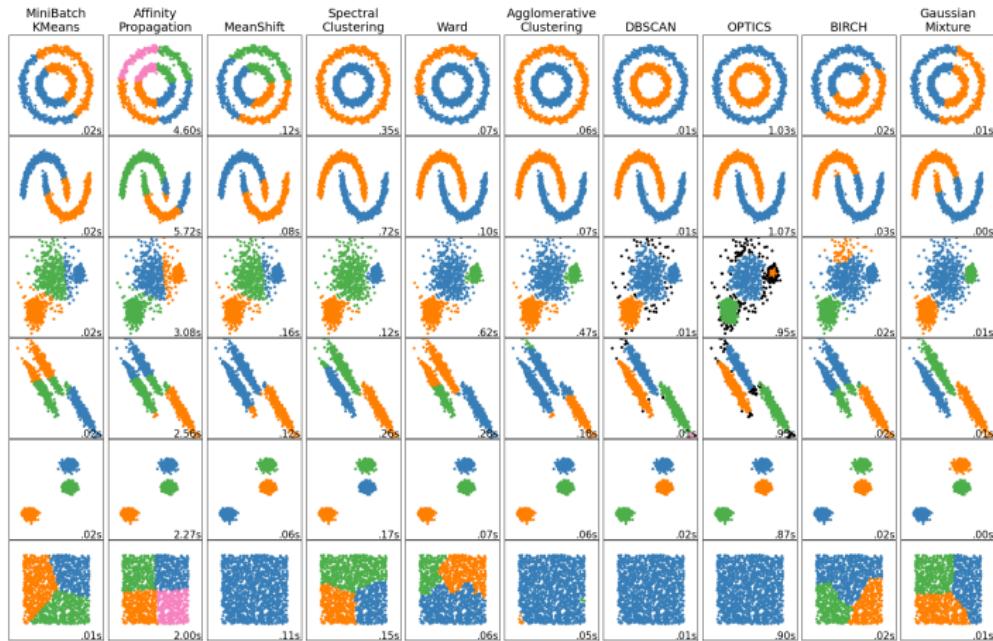
教学内容

- ① 认识聚类
- ② 聚类的性能指标
 - 外部指标
 - 内部指标
- ③ K-means 聚类
- ④ K-means 聚类的改进
- ⑤ 谱聚类
- ⑥ 层次聚类
- ⑦ 高斯混合模型聚类
- ⑧ DBSCAN 聚类



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。



聚类的形式化定义

输入数据

假定样本集 $D = \{x_1, x_2, \dots, x_m\}$ 包含 m 个无标记样本, 每个样本 $x_i = (x_{i1}; x_{i2}; \dots; x_{in})$ 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 $\{C_l \mid l = 1, 2, \dots, k\}$, 其中 $C_{l'} \cap_{l' \neq l} C_l = \emptyset$ 且 $D = \bigcup_{l=1}^k C_l$.

聚类结果

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

- 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



① 认识聚类

② 聚类的性能指标

- 外部指标
- 内部指标

③ K-means 聚类

④ K-means 聚类的改进

⑤ 谱聚类

⑥ 层次聚类

⑦ 高斯混合模型聚类

⑧ DBSCAN 聚类



外部指标

对于数据集 $D = \{x_1, x_2, \dots, x_N\}$, 假设通过聚类给出的簇划分为 $C = \{C_1, C_2, \dots, C_k\}$, 而 ground truth 的簇划分为 $Y = \{Y_1, Y_2, \dots, Y_s\}$ 。
 k 和 s 不一定相同。

相应地, 令 λ^C 与 λ^Y 分别表示与 C 和 Y 对应的簇标签。我们从样本对两两之间的关系来考虑, 定义

$$a = |SS|, \quad SS = \{(x_i, x_j) | \lambda_i^C = \lambda_j^C, \lambda_i^Y = \lambda_j^Y, i < j\}$$

$$b = |SD|, \quad SD = \{(x_i, x_j) | \lambda_i^C = \lambda_j^C, \lambda_i^Y \neq \lambda_j^Y, i < j\}$$

$$c = |DS|, \quad DS = \{(x_i, x_j) | \lambda_i^C \neq \lambda_j^C, \lambda_i^Y = \lambda_j^Y, i < j\}$$

$$d = |DD|, \quad DD = \{(x_i, x_j) | \lambda_i^C \neq \lambda_j^C, \lambda_i^Y \neq \lambda_j^Y, i < j\}$$

由于每个样本对 $(x_i, x_j)(i < j)$ 能且仅能出现在一个集合中,
因此有 $a + b + c + d = N(N - 1)/2$ 成立。



外部指标

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)

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

- Rand 指数 (Rand Index, 简称 RI)

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

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

Random Index 存在一个问题：对于两个随机的划分，其 RI 值并不是一个接近于 0 的常数。

因此，在 RI 的基础上，我们进一步引入 Adjusted Rand Index，ARI。

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

取值范围在 $[-1, 1]$ ，值越大表示聚类效果越好。



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**样本**的角度更直观地来评估聚类的性能。

然而，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**。

具体使用时，比较常用的方式是使聚类簇的数量等于 ground truth 中类别的数量，即聚类标签和 ground truth 之间一对映射。



外部指标

对于数据集 $D = \{x_1, x_2, \dots, x_N\}$, 假设通过聚类给出的各样本标签为 $C = \{c_1, c_2, \dots, c_N\}$, 相对应的 ground truth 为 $Y = \{y_1, y_2, \dots, y_N\}$ 。我们可以通过下式计算聚类精度 (ACC) :

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

其中, m 包含了所有从聚类生成标签到 ground truth 标签的一对一映射。具体实现时, 可以使用匈牙利算法来寻找最优映射。



外部指标

```
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
```



图：寻找最优映射

外部指标

当然，我们也可以从整个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 ground truth, C 为聚类标签, 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调整兰德指数 ARI, 标准化互信息 NMI 和聚类精度 ACC。



内部指标

不使用 ground truth 的指标

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

- 簇内样本平均距离

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

- 簇内样本间最远距离

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



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
 $\mu = \frac{1}{|C|} \sum_{1 \leq i \leq |C|} x_i$ 。



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)

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

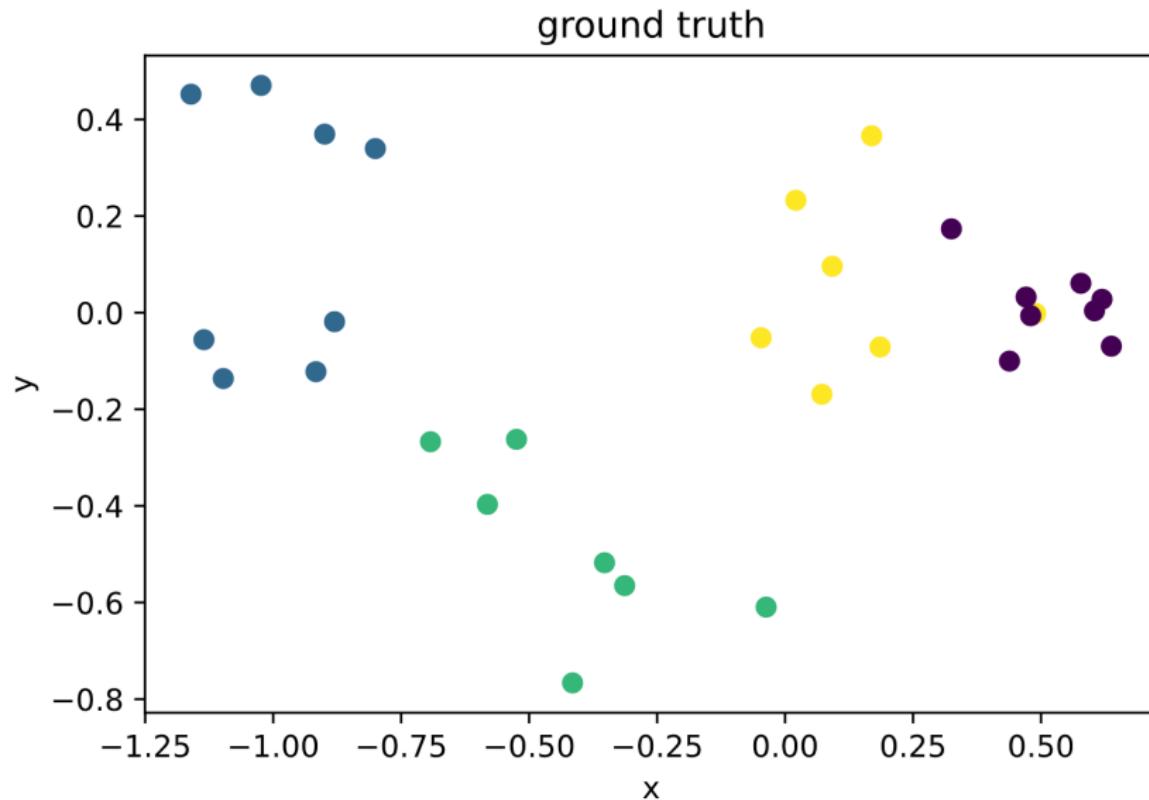
- Dunn 指数 (Dunn Index, 简称 DI)

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

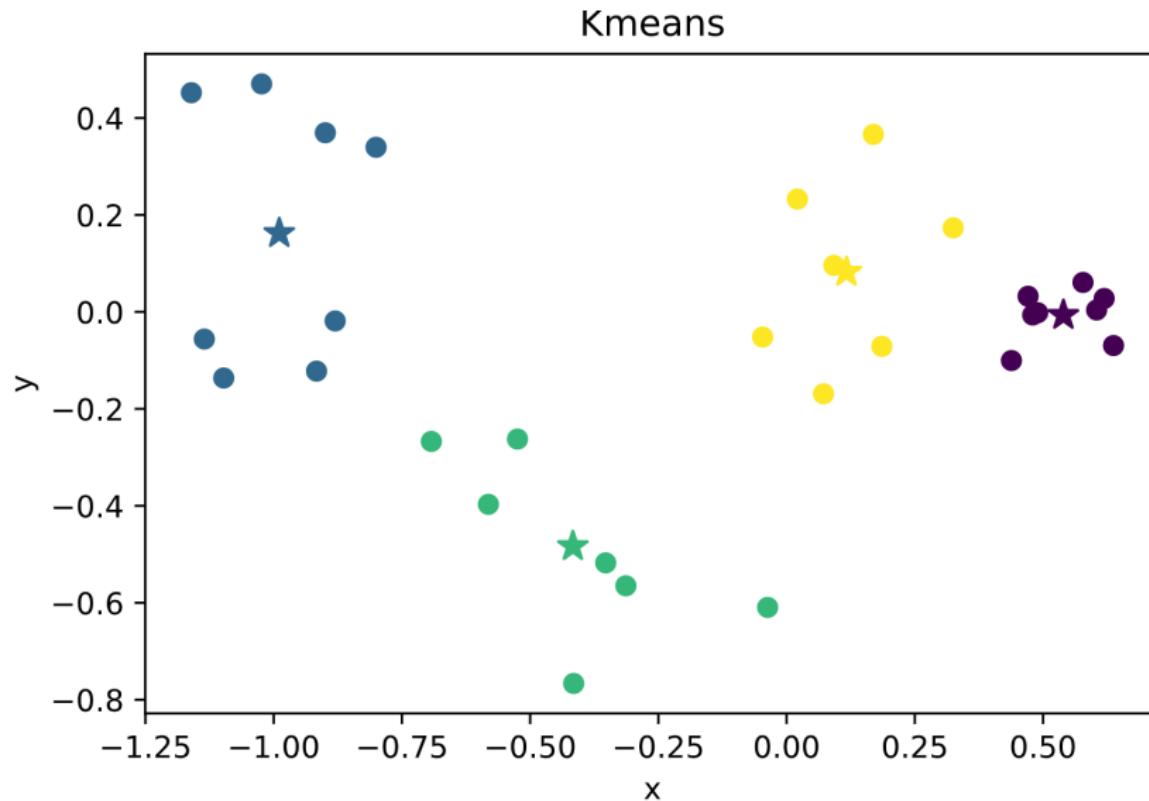
显然，DBI 的值越小越好，而 DI 则相反，值越大越好。



聚类的性能评价

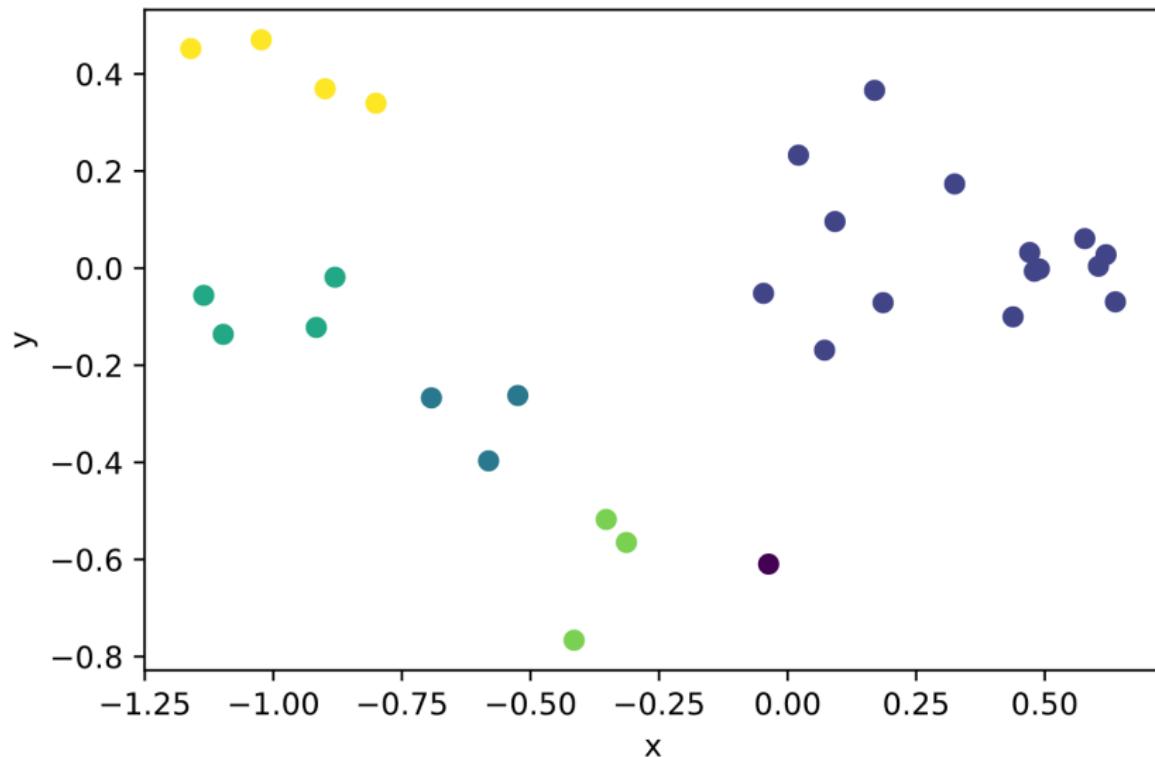


聚类的性能评价



聚类的性能评价

DBSCAN



聚类的性能评价

	x	y	truth	kmeans	dbscan
0	0.619	0.028	0	0	0
1	0.186	-0.071	3	3	0
2	-0.581	-0.397	2	2	1
3	0.021	0.233	3	3	0
4	0.637	-0.069	0	0	0
5	-1.097	-0.136	1	1	2
6	0.490	-0.002	3	0	0
7	-0.313	-0.565	2	2	3
8	-1.161	0.452	1	1	4
9	-1.024	0.470	1	1	4
10	0.072	-0.169	3	3	0
11	-0.916	-0.122	1	1	2
12	-1.135	-0.056	1	1	2
13	-0.693	-0.267	2	2	1
14	0.480	-0.006	0	0	0

	x	y	truth	kmeans	dbscan
15	0.169	0.366	3	3	0
16	-0.800	0.340	1	1	4
17	-0.880	-0.019	1	1	2
18	-0.037	-0.610	2	2	-1
19	-0.899	0.370	1	1	4
20	-0.415	-0.766	2	2	3
21	-0.525	-0.262	2	2	1
22	-0.047	-0.052	3	3	0
23	0.438	-0.100	0	0	0
24	0.471	0.032	0	0	0
25	0.325	0.173	0	3	0
26	0.578	0.061	0	0	0
27	0.092	0.096	3	3	0
28	0.604	0.004	0	0	0
29	-0.353	-0.518	2	2	3



聚类的性能评价——外部指标

- K-means

$$a = 85, \quad b = 13, \quad c = 13, \quad d = 324$$

$$JC = 0.766, \quad FMI = 0.867, \quad RI = 0.940$$

- DBSCAN

$$a = 67, \quad b = 56, \quad c = 31, \quad d = 281$$

$$JC = 0.435, \quad FMI = 0.610, \quad RI = 0.800$$



聚类的性能评价——内部指标

- K-means

$$avg = [0.106, \quad 0.337, \quad 0.311, \quad 0.256]$$

$$diam = [0.221, \quad 0.624, \quad 0.740, \quad 0.544]$$

$$DBI = 0.795, \quad DI = 0.274$$

- DBSCAN

$$avg = [0.321, \quad 0.108, \quad 0.139, \quad 0.121, \quad 0.164]$$

$$diam = [0.686, \quad 0.171, \quad 0.258, \quad 0.256, \quad 0.377]$$

$$DBI = \mathbf{0.573}, \quad DI = \mathbf{0.377}$$

注意：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，可以对齐进行分类：

- 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
-



- ① 认识聚类
- ② 聚类的性能指标
 - 外部指标
 - 内部指标
- ③ K-means 聚类
- ④ K-means 聚类的改进
- ⑤ 谱聚类
- ⑥ 层次聚类
- ⑦ 高斯混合模型聚类
- ⑧ DBSCAN 聚类



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。



图：符合（左）与不符合（右）



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:

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

其中 $r_{nk} \in \{0, 1\}$ 表示第 n 个样本是否被分到第 k 个类。

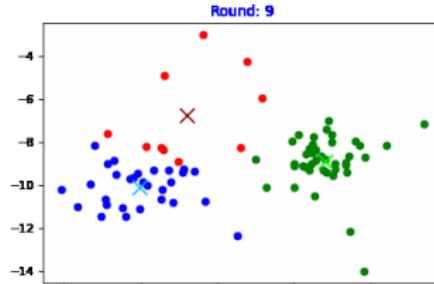
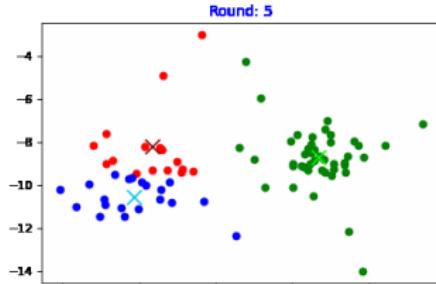
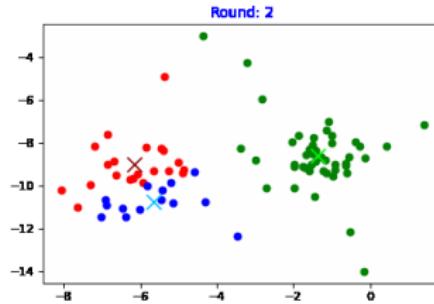
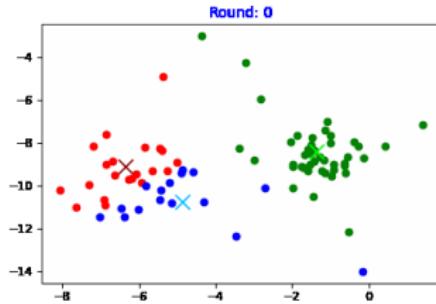
该目标函数的直观理解是使每个样本到其聚类中心的距离尽量近, 即让最终的聚类结果尽量“紧凑”



K-means 聚类的优化

一种最粗糙的做法是枚举所有可能的组合，该情况下的复杂度是
 $O(n^{dk+1} \log n)$

迭代优化，离散优化



K-means 聚类的步骤

Algorithm 1 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



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

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



非负矩阵分解

K-means 和非负矩阵分解的等价性推导

K-means 优化目标

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

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）。



非负矩阵分解

定义新的变量：

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

K-means 的优化目标转化为：

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



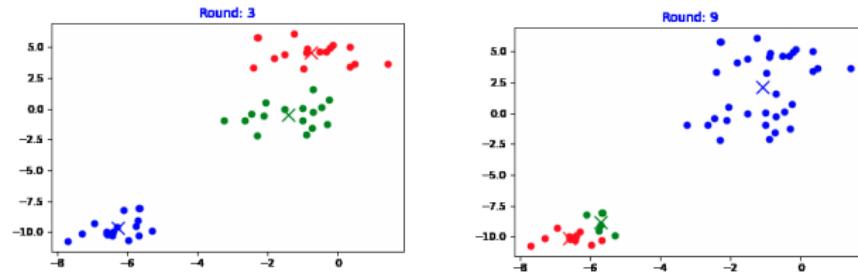
思考

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



图：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++



- ① 认识聚类
- ② 聚类的性能指标
 - 外部指标
 - 内部指标
- ③ K-means 聚类
- ④ K-means 聚类的改进
- ⑤ 谱聚类
- ⑥ 层次聚类
- ⑦ 高斯混合模型聚类
- ⑧ DBSCAN 聚类



K-medoids 聚类

K-medoids 聚类算法，主要克服 K-means 算法容易受到离群点影响的问题。

K-medoids 聚类算法

不同于 K-means 使用簇中对象的均值作为聚类中心点，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。其他与 K-means 策略相同。其优化目标可以描述为：

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



K-medoids 聚类与 K-means 算法对比

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

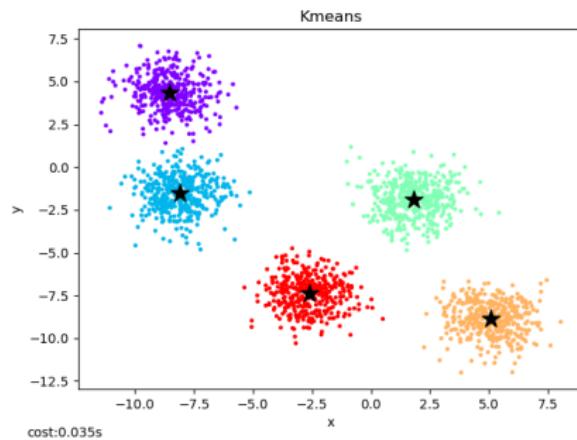
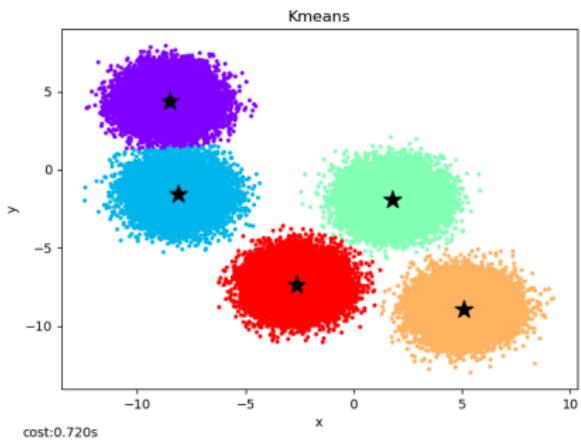
- ① K-medoids 算法的复杂度是 $O(tk(n - k)^2)$, 难以应用于大数据集。



Mini-batch K-means

mini-batch 的思想非常朴素：既然整个样本集的数据量太大，会使得我们迭代的时间过长，那么我们是否可以缩小数据规模？

最简单的想法就是从整体中进行抽样，随机选取出一小部分数据来代替整体。



图：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，从理论上讲这确实是有可能的。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++

导致 K-means 效率问题的因素有两个：样本数量太大和迭代次数过多。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。

然而，K-means 迭代过程中具体的次数是由其内部的迭代逻辑决定的，对这一部分进行优化会比较复杂。所以，我们可以挑一个更软的柿子来捏——迭代起始的位置。

直观上，如果我们的初始状态离最终的收敛状态越近，那么收敛所需要的迭代次数就越少。



K-means++

那么，如何判断起始状态离收敛结果的远近呢？

通过分析，我们可以获得以下两个结论：

- 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。



- ① 认识聚类
- ② 聚类的性能指标
 - 外部指标
 - 内部指标
- ③ K-means 聚类
- ④ K-means 聚类的改进
- ⑤ **谱聚类**
- ⑥ 层次聚类
- ⑦ 高斯混合模型聚类
- ⑧ DBSCAN 聚类



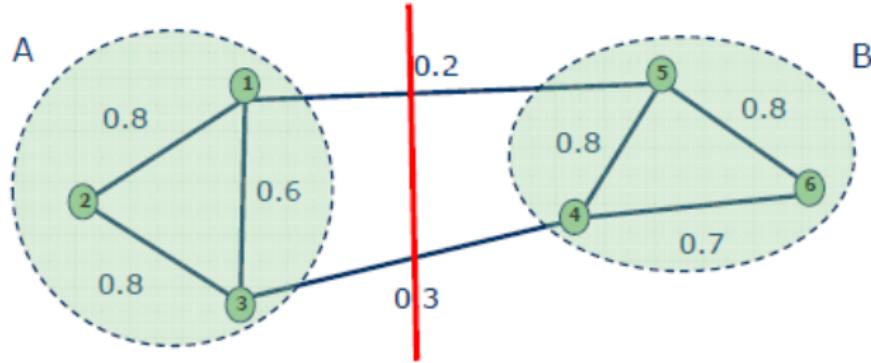
谱聚类 (Spectral Clustering)

问题定义：

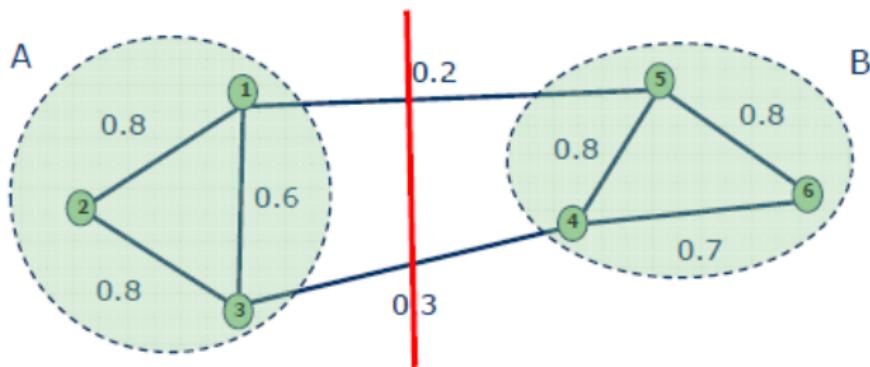
给定一个无向有权图

$$G(V, E)$$

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)



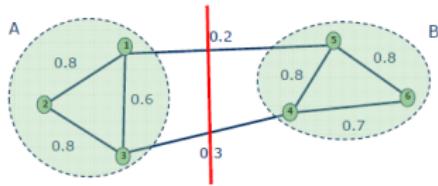
回顾聚类的评价标准：

- 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)

显然，基于图的各种图切割算法比如 N-Cut 可以适用这个问题。但我们现在来学习一个又简单又复杂的算法：**谱聚类 (Spectral Clustering)**
首先我们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)

为了更好地理解谱聚类的思想，我们还是需要从 N-Cut 开始讲起。前文提到我们的目标是：

- 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：

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

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



谱聚类 (Spectral Clustering)

自然，我们有了以下目标：

①

$$\min \text{cut}(A, B)$$

②

$$\max(\text{assoc}(A, A) + \text{assoc}(B, B))$$

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

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

- Normalize the association between groups.

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

N-cut 算法的目标函数如下

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



谱聚类 (Spectral Clustering)

接下来我们用形式化的数学语言来描述 N-cut: ¹

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

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



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

谱聚类 (Spectral Clustering)

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

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

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

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

- 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

上面的公式太难看了，我们进行化简：

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

于是有：

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

$$\begin{aligned} \mathbf{y}^T D \mathbf{y} &= 4 \sum_{x_i > 0} d_i + 4b^2 \sum_{x_i < 0} d_i = 4 \left(b \sum_{x_i < 0} d_i + b^2 \sum_{x_i < 0} d_i \right) \\ &= 4b \left(\sum_{x_i < 0} d_i + b \sum_{x_i < 0} d_i \right) = 4b \mathbf{1}^T D \mathbf{1} \end{aligned}$$



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：

$$\begin{aligned} \min_{\mathbf{x}} \text{Ncut}(\mathbf{x}) &= \min_{\mathbf{y}} \frac{\mathbf{y}^T(D - W)\mathbf{y}}{\mathbf{y}^T D \mathbf{y}} \\ \text{s.t. } \mathbf{y} &\in [2, -2b]^n, \quad \mathbf{y}^T D \mathbf{1} = 0 \end{aligned}$$

但问题来了，能写出来是一回事，能优化出来又是另一回事。当年的顶级数学家对这个式子进行了如下 Relaxation：

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

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



谱聚类 (Spectral Clustering)-图拉普拉斯

$L = D - W$ 有一个名字，叫做拉普拉斯矩阵，它是个半正定矩阵，证明如下。

$$\begin{aligned}
 x^T L x &= x^T D x - x^T W x = \sum_{i,j=1}^n (D_{ij} - A_{ij}) x_i x_j \\
 &= \sum_i D_{ii} x_i^2 - \sum_{(i,j) \in E} 2x_i x_j = \sum_{(i,j) \in E} (x_i^2 + x_j^2 - 2x_i x_j) \\
 &= \sum_{(i,j) \in E} (x_i - x_j)^2 \geq 0
 \end{aligned}$$

所以拉普拉斯矩阵是半正定矩阵。由于 $L = D - W$ 当 $x = (1, \dots, 1)$ 时， $Lx = (D - W)x = Dx - Wx = 0$ ，则 $\lambda = 0$ 为拉普拉斯矩阵的一个特征值，又由于拉普拉斯矩阵的特征值非负，所以 0 为最小的特征值，对应的特征向量为 $x = (1, \dots, 1)$

谱聚类 (Spectral Clustering)-图拉普拉斯

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

将其一般化：

$$J(\mathbf{y}) = \mathbf{y}^T L \mathbf{y} - \lambda (\mathbf{y}^T D \mathbf{y} - c)$$

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

上述放缩后的最小化问题可以通过解特征值特征向量

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

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

$$Ly = \lambda Dy$$

$$\begin{aligned} &\text{令: } y = D^{-\frac{1}{2}} f \\ &\Leftrightarrow LD^{-\frac{1}{2}} f = \lambda D^{\frac{1}{2}} f \\ &\Leftrightarrow D^{-\frac{1}{2}} LD^{-\frac{1}{2}} f = \lambda f \end{aligned}$$

这个时候表达式就是：

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

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

我们来看具体怎么做。

①

$$\min_x \frac{x^T A x}{x^T B x} = \min_x R(x) \Leftrightarrow \min_x x^T A x \quad \text{s.t. } x^T B x = C$$

这个步骤将分式优化问题转化为拉格朗日乘子问题，C 为常数。

②

$$L(x) = x^T A x + \lambda (x^T B x - C)$$

写出拉格朗日函数，令导数为零，得到驻点的解析解方程

③

$$\frac{\partial L(x)}{\partial x} = 0 \quad (A + A^T)x + \lambda (B + B^T)x = 0$$

A,B 都是对称阵

④

$$Ax = \kappa Bx, \kappa = -\lambda$$



谱聚类 (Spectral Clustering)-图拉普拉斯

根据前文提到的瑞丽商构造法，该式可以转化为：

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

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

可以看出：

- ① 约束项 $R(y)$ 的最大值就是 $D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ 最大特征值，最小值就是最小特征值
- ② y 的解，就是对应的特征向量



谱聚类 (Spectral Clustering)-图拉普拉斯

别忘了刚才的约束优化式还有另一个约束：

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

全 1 向量 $\mathbf{1}$ 是 0 (最小的特征值) 特征值对应的特征向量。

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

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

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

由于特征向量两两正交，所以当 \mathbf{y} 不是 1 的时候该约束总成立

$$f_1^T f_2 = 0 \Rightarrow \left(D^{\frac{1}{2}} y_1\right)^T \left(D^{\frac{1}{2}} y_2\right) = 0 \Rightarrow y_1^T D y_2 = 0$$



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：

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

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 显然，递归地进行二分很好理解
- ② 但是为什么突然要选多个特征向量形成新的表征再去聚类呢？难道谱聚类实际上是一种特征工程吗？
- ③ 最小特征值对应的特征向量是优化目标的最优解这个可以理解，但凭什么最小的若干个特征向量可以拿来对数据做变换呢？



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

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

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 KNN-weight 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

- ① 如果 y 是离散的 0, 1, 代表数据划分为两个子图, 最后优化目标是两个子图之间的 weight 之和, 可以看出 **y 这个目标对应了图划分的目标**, 问题是图划分是 NP-hard 的, 因此转化为拉普拉斯降维来做。
- ② 但是经过约束宽松, 我们最终得到的 y 是一个实数向量, 无法直接当作图划分来用。但我们现在把 y 视为离散的图划分 assignment (它本来的意义)
- ③ 这时我们发现, 拉普拉斯矩阵除 0 之外最小的 k_1 个特征值对应的特征向量代表了: **k_1 种最优的图二分割**, 因为前文证明了拉普拉斯矩阵的特征向量与图二分的结果直接相关。



谱聚类 (Spectral Clustering)-图拉普拉斯

- ① 举例而言，假如我们现在对人群进行 k 聚类，首先要对人群进行 KNN 构图，然后分析了拉普拉斯矩阵的特征向量，也就是该人群的若干种最优的二分策略
- ② 现在我们给这些二分策略赋予语义（只是为了更好理解，实际上不能这样做），比如戴不戴眼镜，留不留胡子等等……
- ③ 这样，我们选取的 k_1 个特征向量，实际上是该人群在 k_1 种二分标准下的全新特征。这些特征是通过拉普拉斯矩阵分析得到的，在原本的特征空间中不存在。
- ④ 把这些能把人群显著二分的特征再拿去聚类，效果会好很多。



谱聚类 (Spectral Clustering)-图拉普拉斯

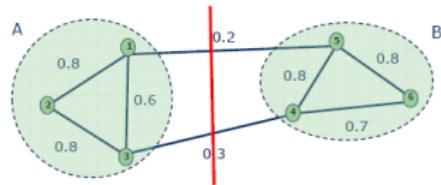
总结一下谱聚类的流程：

- 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 k_1 个特征值所各自对应的特征向量 f , 最终组成 $n \times k_1$ 维的特征矩阵 F
- ⑥ 对 F 中的每一行作为一个 k_1 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)

看看这个例子，请大家课后回去自己算一下



	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)

```

> import numpy as np
def callLaplacianMatrix(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

    # normalize
    # 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(callLaplacianMatrix(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]]



① 认识聚类

② 聚类的性能指标

- 外部指标
- 内部指标

③ K-means 聚类

④ K-means 聚类的改进

⑤ 谱聚类

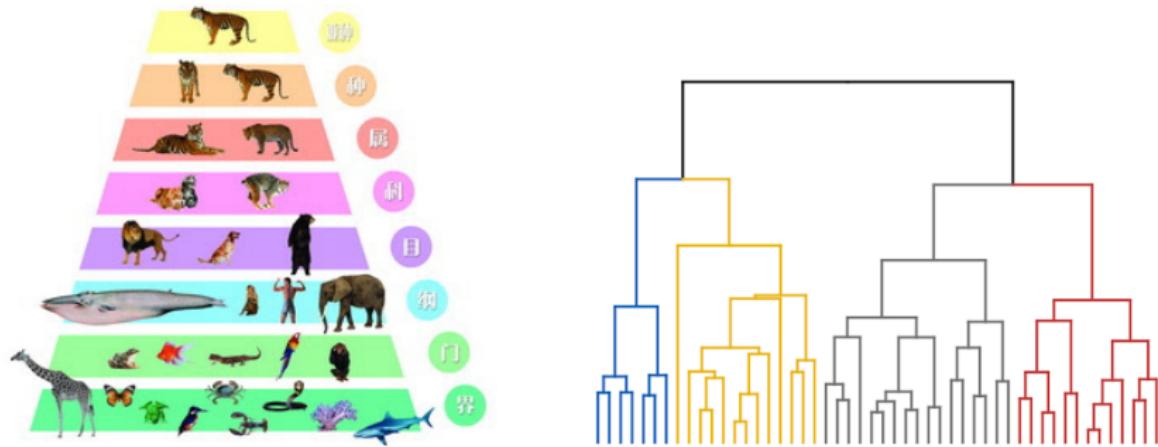
⑥ 层次聚类

⑦ 高斯混合模型聚类

⑧ DBSCAN 聚类



层次聚类



图：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

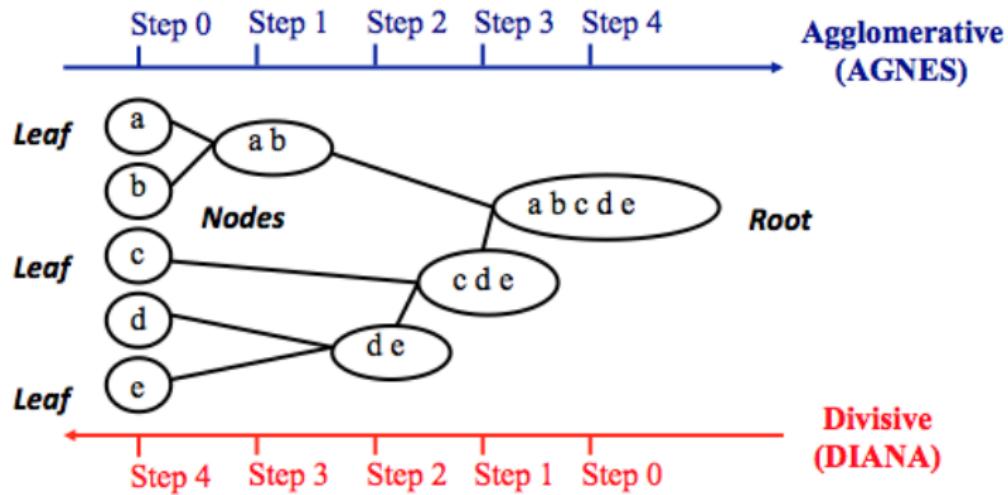
层次聚类 (Hierarchical Clustering) 的一类将数据对象组成层次结构或簇的“树”的方法的总称。

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



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



最近邻层次聚类算法

最近邻（层次）聚类算法

使用最小距离 $\text{dist}_{\min}(C_i, C_j)$ 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棵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



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



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

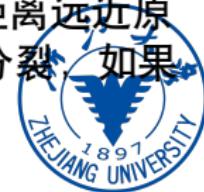
CF Tree 的构造步骤：

- 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



CF Tree 树的插入

- ① 从根节点向下寻找和新样本距离最近的叶子节点和叶子节点里最近的 CF 节点
- ② 如果新样本加入后，这个 CF 节点对应的超球体半径仍然满足小于阈值 T ，则更新路径上所有的 CF 三元组，插入结束。否则转入 3.
- ③ 如果当前叶子节点的 CF 节点个数小于阈值 L ，则创建一个新的 CF 节点，放入新样本，将新的 CF 节点放入这个叶子节点，更新路径上所有的 CF 三元组，插入结束。否则转入 4。
- ④ 将当前叶子节点划分为两个新叶子节点，选择旧叶子节点中所有 CF 元组里超球体距离最远的两个 CF 元组，分布作为两个新叶子节点的第一个 CF 节点。将其他元组和新样本元组按照距离远近原则放入对应的叶子节点。依次向上检查父节点是否也要分裂。如果需要按和叶子节点分裂方式相同。



进阶 BIRCH 聚类算法

真实的 BIRCH 算法除了构造 CF Tree 之外，往往还可以引入更多操作来提升效果

- ① 将 CF Tree 进行筛选，去除一些异常 CF 节点，这些节点一般里面的样本点很少。对于一些超球体距离非常近的元组进行合并
- ② 利用其它的一些聚类算法比如 K-Means 对所有的 CF 元组进行聚类，得到一颗比较好的 CF Tree. 这一步的主要目的是消除由于样本读入顺序导致的不合理的树结构，以及一些由于节点 CF 个数限制导致的树结构分裂。
- ③ 利用 CF Tree 的所有 CF 节点的质心，作为初始质心点，对所有的样本点按距离远近进行聚类。这样进一步减少了由于 CF Tree 的一些限制导致的聚类不合理的情况。



- ① 认识聚类
- ② 聚类的性能指标
 - 外部指标
 - 内部指标
- ③ K-means 聚类
- ④ K-means 聚类的改进
- ⑤ 谱聚类
- ⑥ 层次聚类
- ⑦ 高斯混合模型聚类
- ⑧ DBSCAN 聚类



GMM

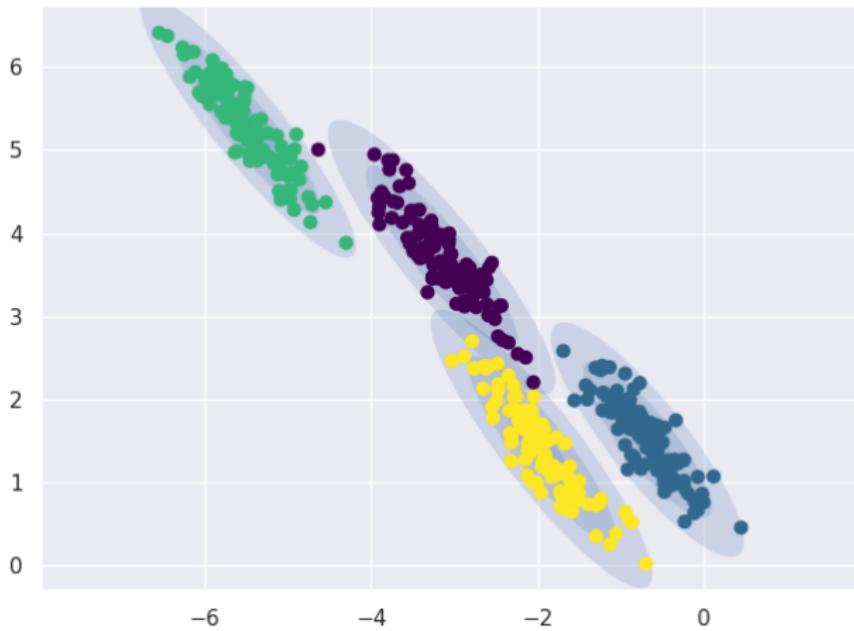
高斯混合模型（Gaussian Mixture Model）聚类是一种基于模型的聚类方法，其核心思想为：假设当前的样本集都是由一个概率生成模型产生的，而这个概率模型则是由多个基础的高斯分布通过线性组合混合所得到。

基于这个思想，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



GMM

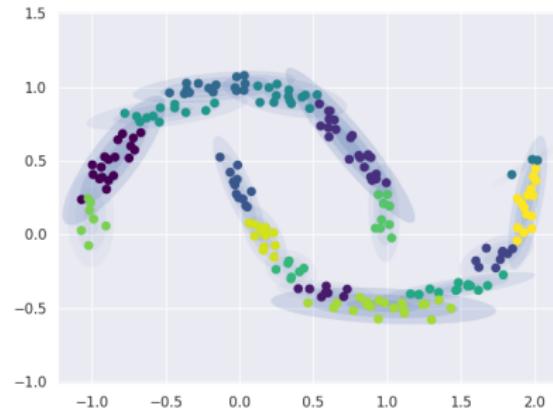
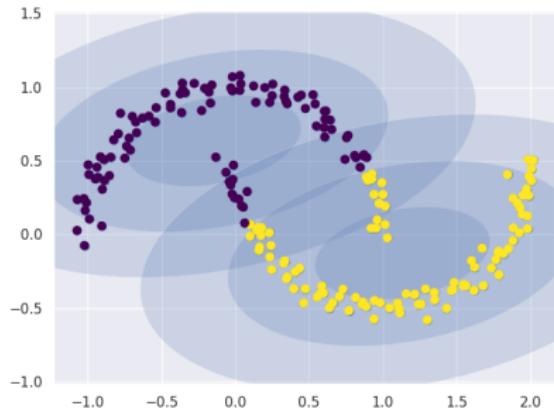


图：通过由多个高斯分布混合而得的模型来拟合样本分布



GMM

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。



图：2 个与 16 个高斯分布混合模型的拟合结果



GMM

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：

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

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

由上式可以看出，高斯分布完全由均值向量 μ 和协方差矩阵 Σ 这两个参数确定。

因此，为了明确显示高斯分布与相应参数的依赖关系，我们将概率密度函数记为 $p(x|\mu, \Sigma)$ 。



GMM

我们定义高斯混合分布为：

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

该分布共由 k 个混合成分组成，每个混合成分对应一个高斯分布。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**， $\sum_{i=1}^k \alpha_i = 1$ 。



GMM

假设样本的生成过程由高斯混合分布给出，具体过程如下：

- 首先，根据 $\alpha_1, \alpha_2, \dots, \alpha_k$ 定义的先验分布选择高斯混合成分，其中 α_i 为选择第 i 个混合成分的概率。
- 然后，根据被选择的混合成分的概率密度函数进行采样，从而生成相应的样本。



GMM

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

根据贝叶斯定理， z_j 的后验分布对应于：

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

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。为方便叙述，将其简记为 $\gamma_{ji} (i = 1, 2, \dots, k)$ 。



GMM

更直观的来说， $\gamma_{ji}(i = 1, 2, \dots, k)$ 可以理解为样本 x_j 属于第 i 类的概率，即 soft assignment。

当高斯混合分布（式 (2)）已知时，高斯混合聚类把样本集 D 划分为 k 个簇 $\mathcal{C} = C_1, C_2, \dots, C_k$ ，每个样本 x_j 的簇标记 λ_j 如下确定：

$$\lambda_j = \arg \max_{i \in \{1, 2, \dots, k\}} \gamma_{ji} \quad (4)$$



GMM

那么，剩下的问题就是对于下式，模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) | 1 \leq i \leq k\}$ 如何求解。

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

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

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



GMM

接下来，我们采用 EM 算法进行迭代优化求解，推导过程如下：

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

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



GMM

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

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

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

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



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

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

其拉格朗日函数为

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

其中 λ 为拉格朗日乘子。



GMM

由上式对 α_i 的导数为 0，有

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

两边同乘以 α_i ，对所有样本求和可得 $\lambda = -n$ ，有

$$\alpha_i = \frac{1}{n} \sum_{j=1}^n \gamma_{ji} \quad (11)$$

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 $D = \{x_1, x_2, \dots, x_n\}$;

高斯混合成分个数 k .

过程:

- 1: 初始化高斯混合分布的模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$
 - 2: **repeat**
 - 3: **for** $j = 1, 2, \dots, n$ **do**
 - 4: 根据式(3)计算 x_j 由各混合成分生成的后验概率, 即

$$\gamma_{ji} = p_M(z_j = i \mid x_j) \quad (1 \leq i \leq k)$$
 - 5: **end for**
 - 6: **for** $i = 1, 2, \dots, k$ **do**
 - 7: 计算新均值向量: $\mu'_i = \frac{\sum_{j=1}^n \gamma_{ji} x_j}{\sum_{j=1}^n \gamma_{ji}}$;
 - 8: 计算新协方差矩阵: $\Sigma'_i = \frac{\sum_{j=1}^n \gamma_{ji} (x_j - \mu'_i)(x_j - \mu'_i)^T}{\sum_{j=1}^n \gamma_{ji}}$;
 - 9: 计算新混合系数: $\alpha'_i = \frac{\sum_{j=1}^n \gamma_{ji}}{n}$;
 - 10: **end for**
 - 11: 将模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$ 更新为 $\{(\alpha'_i, \mu'_i, \Sigma'_i) \mid 1 \leq i \leq k\}$
 - 12: **until** 满足停止条件
 - 13: $C_i = \emptyset \quad (1 \leq i \leq k)$
 - 14: **for** $j = 1, 2, \dots, n$ **do**
 - 15: 根据式(4)确定 x_j 的簇标记 λ_j ;
 - 16: 将 x_j 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \cup \{x_j\}$
 - 17: **end for**
-
- 输出: 簇划分 $C = \{C_1, C_2, \dots, C_k\}$



图: 高斯混合聚类算法

- ① 认识聚类
- ② 聚类的性能指标
 - 外部指标
 - 内部指标
- ③ K-means 聚类
- ④ K-means 聚类的改进
- ⑤ 谱聚类
- ⑥ 层次聚类
- ⑦ 高斯混合模型聚类
- ⑧ DBSCAN 聚类



DBSCAN 聚类

密度聚类假设聚类结构能通过样本分布的紧密程度确定。

通常情形下，密度聚类算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

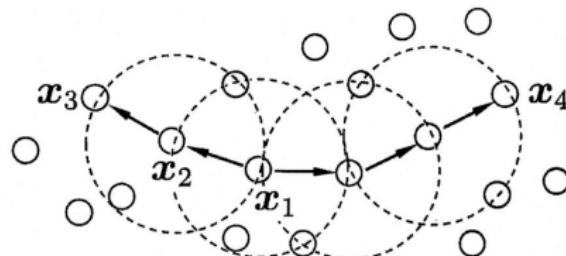
给定数据集 $D = \{x_1, x_2, \dots, x_N\}$, 定义以下几个概念:

- 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, \dots, 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

基于这些概念，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。

形式化地说，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 $X = \{x' \in D | x' \text{ 由 } x \text{ 密度可达}\}$ 。

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 $D = \{x_1, x_2, \dots, x_m\}$;
邻域参数 $(\epsilon, MinPts)$.

过程:

```

1: 初始化核心对象集合:  $\Omega = \emptyset$ 
2: for  $j = 1, 2, \dots, m$  do
3:   确定样本  $x_j$  的  $\epsilon$ -邻域  $N_\epsilon(x_j)$ ;
4:   if  $|N_\epsilon(x_j)| \geq MinPts$  then
5:     将样本  $x_j$  加入核心对象集合:  $\Omega = \Omega \cup \{x_j\}$ 
6:   end if
7: end for
8: 初始化聚类簇数:  $k = 0$ 
9: 初始化未访问样本集合:  $\Gamma = D$ 
10: while  $\Omega \neq \emptyset$  do
11:   记录当前未访问样本集合:  $\Gamma_{old} = \Gamma$ ;

```

```

12:   随机选取一个核心对象  $o \in \Omega$ , 初始化队列  $Q = < o >$ ;
13:    $\Gamma = \Gamma \setminus \{o\}$ ;
14:   while  $Q \neq \emptyset$  do
15:     取出队列  $Q$  中的首个样本  $q$ ;
16:     if  $|N_\epsilon(q)| \geq MinPts$  then
17:       令  $\Delta = N_\epsilon(q) \cap \Gamma$ ;
18:       将  $\Delta$  中的样本加入队列  $Q$ ;
19:        $\Gamma = \Gamma \setminus \Delta$ ;
20:     end if
21:   end while
22:    $k = k + 1$ , 生成聚类簇  $C_k = \Gamma_{old} \setminus \Gamma$ ;
23:    $\Omega = \Omega \setminus C_k$ 
24: end while

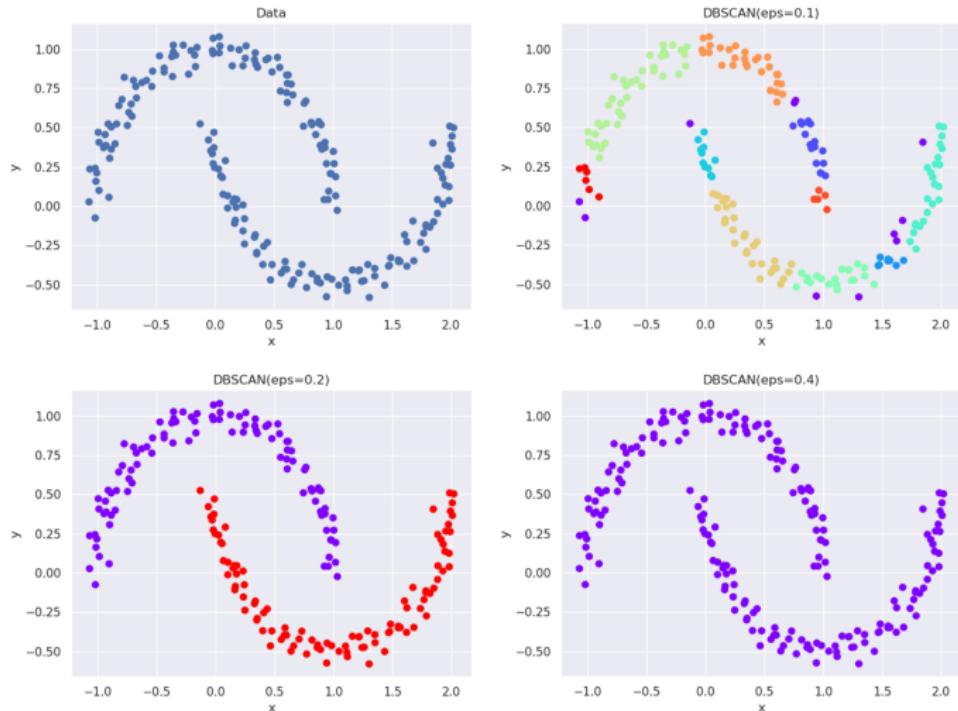
```

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

图: DBSCAN 的算法流程



DBSCAN 聚类



图：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
 - 外部指标
 - 内部指标
- ③ K-means 聚类
- ④ K-means 聚类的改进
- ⑤ 谱聚类
- ⑥ 层次聚类
- ⑦ 高斯混合模型聚类
- ⑧ DBSCAN 聚类

