# lec13.2 k-medoids

k-Means 算法的问题：

1. 结果取决于初始随机选择：
   - k-means 的结果可能会因为初始簇中心的随机选择而有所不同。这意味着每次运行算法可能会得到不同的聚类结果
2. 可能陷入局部最小值：
   - k-means 可能会陷入局部最小值，而不是全局最优解
   - 解决方法：多次运行聚类过程，每次使用不同的初始值，并选择最优的聚类结果
3. 离群点的影响较大：
   - 离群点（outliers）对均值值的影响较大，进而影响到簇中心和整个簇的结果
   - 这种敏感性可能会导致聚类结果偏差
4. 簇中心（均值）不是簇中的实际实例：
   - k-means 算法中的簇中心（均值）通常不是数据集中实际存在的实例
   - 这可能导致在解释簇时遇到困难，因为代表点不是实际的数据点
5. 欧几里得距离不适用于分类特征：
   - k-means 使用欧几里得距离来度量数据点之间的距离，而欧几里得距离不适用于分类（categorical）特征
   - 这使得 k-means 在处理分类数据时表现不佳

# k-medoids 算法

*算法概述：*
这是一个基于代表（representative-based）的算法，目标是确定 $k$ 个代表 $Y_1, \ldots, Y_k$，使得以下目标函数最小化

$$\sum_{i=1}^{n} \left[ \min_j d(X_i, Y_j) \right]$$

与 k-means 不同，在 k-medoids 算法中：

- 距离函数 $d(\cdot, \cdot)$ 可以是任何适用于数据集的函数（不一定是欧几里得距离）
- 代表从数据集中选择

*爬山策略（hill-climbing strategy）：*

1. 从问题的任意解开始
2. 尝试通过对解进行增量更改来找到更好的解
3. 如果更改产生了更好的解，则对新解进行进一步的增量更改，如此反复，直到找不到更好的解为止

*算法步骤：*

- 初始化阶段：从数据集中随机选择 $k$ 个代表（medoids）$Y_1, \ldots, Y_k$
- 分配阶段：将数据集中所有对象分配给最近的代表，得到簇 $C_1, \ldots, C_k$
- 优化阶段（爬山步骤）：
  1. 找到一对 $X, Y$，其中 $X \in \mathscr{D}$ 且 $Y \in \{Y_1, \ldots, Y_k\}$

2. 如果用 $X$ 替换 $Y$
3. 如果改善是正的（目标函数的值更小），则保留这个替换，然后返回到第二步；否则，不进行替换并返回当前簇 $C_1, \ldots, C_k$

*优点和缺点：*

- 优点：
  - 代表从数据集中选择，便于解释簇代表
  - 比 k-means 更稳健，对噪声和离群点更具鲁棒性
  - 可以使用任意的不相似度量，因此适用于复杂数据类型（如分类、混合、时间序列等）
- 缺点：
  - 结果可能因初始随机选择而异
  - 可能陷入局部最小值，而不是全局最优解
  - 比 k-means 算法慢

*时间复杂度问题：*

*重述算法过程：*

假设数据集中有 $n$ 个对象，在每次优化阶段，我们需要计算 $k \cdot n$ 次增量目标函数变化，这是非常耗时的。具体的优化步骤如下：

1. 寻找一对 $X, Y$，其中 $X \in \mathscr{D}$ 且 $Y \in \{Y_1, \ldots, Y_k\}$
2. 用 $X$ 替换 $Y$ （如果这种替换在目标函数中带来了最大的可能改进）
3. 如果改进为正，则用 $X$ 替换 $Y$，并返回到第二阶段。否则，返回当前簇 $C_1, \ldots, C_k$

由于每次优化都需要进行 $k \cdot n$ 次计算，对于大数据集来说，这样的计算量是非常昂贵的

*解决方案：*

一种解决方案是

1. 使用随机选择的一组 $r$ 个 $(X, Y)$ 对，其中 $X \in \mathscr{D}$ 且 $Y \in \{Y_1, \ldots, Y_k\}$
2. 使用这些对中的最佳对进行替换

这样，只需要计算 $r$ 次增量目标函数变化