

第 5 章

聚 类

聚类作为一种无监督学习方法,是数据挖掘的重要任务之一,它能够帮助企业执行客户细分、市场分析、欺诈检测和产品推荐等核心业务。本章将详细介绍多种经典的聚类算法,深入讲解基于划分的 K-Means 系列聚类,凝聚式和分裂式层次聚类,基于密度的 DBSCAN 聚类,模糊聚类和基于神经网络的聚类等,展示如何运用聚类算法解决实际问题。

5.1 聚类概述

聚类能够从无标签数据中发现潜在模式,其关键在于根据数据特性选择合适的算法、相似度度量和评估方法。

5.1.1 聚类的概念

“物以类聚,人以群分”。聚类的本质是将数据集中相似的样本进行分组。每组称为一个簇(Cluster),每簇的样本对应一个潜在的类别。聚类是进行无标签数据探索的重要工具,样本是没有类别标签的,因此聚类是一种典型的无监督学习任务。

当这些簇满足相同簇的样本之间距离较近、不同簇的样本之间距离较远这两个条件时(如图 5-1 所示),可以认为得到了很好的聚类。也就是说,划分的一般准则为同一个类中的对象尽可能“接近”或相关,不同类中的对象尽可能“远离”或不同。

现实中,聚类可能是模糊的。如图 5-2 所示,同样一堆数据,可能被聚为两个集群、4 个集群或 6 个集群,似乎都是较好的聚类结果。所以,得到最好的聚类结果有时是比较困难的。

聚类在很多应用中十分重要,例如空间数据分析,聚类可以产生 GIS(地理信息系统)的专题地图(Thematic Maps),在空间数据挖掘中检测空间聚类并解释它们;Web 日志数据分析,有助于发现类似访问模式;市场营销场景,可以发现基本顾客的不同组群,进而制订有针对性的营销计划;国土管理,可以从地球观测数据库中识别类似的国土使用区域。

扫一扫



视频讲解

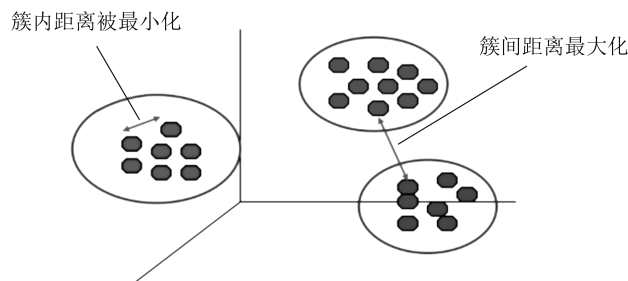


图 5-1 簇内、簇间关系图

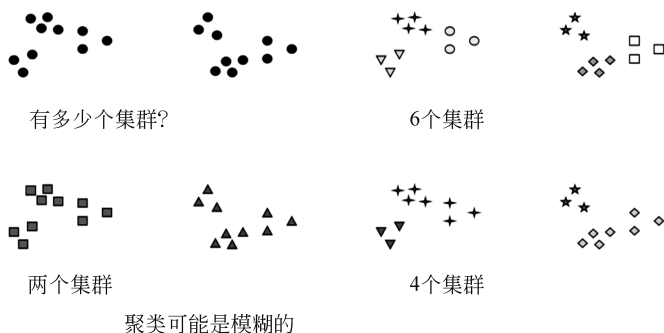


图 5-2 聚类结果图

5.1.2 聚类的分类

聚类按照结果的类型分为硬聚类与软聚类。

- 硬聚类：每个数据点只能属于一个簇。例如，K 均值聚类(K-Means)算法，每个数据点被分配到最近的簇中心，属于该簇且不属于其他簇。
- 软聚类：每个数据点可以属于多个簇，并且有隶属度表示属于每个簇的程度。例如，模糊 C 均值聚类(Fuzzy C-Means, FCM)算法，数据点在每个簇的隶属度之和为 1。

聚类按照过程的策略分为划分聚类、层次聚类、基于密度的聚类、基于网格的聚类、基于模型的聚类、基于神经网络的聚类等。

- 划分聚类：将数据集划分为若干不相交的子集，每个子集称为一个簇。划分聚类的目标是优化某个准则函数，以使簇内的相似度高，簇间的相似度低。
- 层次聚类：通过构建嵌套的聚类来组织对象，形成一个树状的聚类层次结构。
- 基于密度的聚类：根据数据的密度分布来识别簇，簇是密度相连的对象的集合。这种聚类方法能够发现任意形状的簇，并且对噪声和离群点具有鲁棒性。
- 基于网格的聚类：将数据空间量化成有限数量的单元格(网格)，形成一个网格结构，然后对网格单元进行聚类。这种方法的主要优点是处理速度快。
- 基于模型的聚类：假设数据是由某个模型生成的，通过估计模型的参数来确定聚类。
- 基于神经网络的聚类：使用神经网络的拓扑结构和自适应能力来实现聚类。

聚类按照数据的类型分为数值型数据聚类、分类型数据聚类、文本聚类等。

- 数值型数据聚类：适用于连续数值型数据，如 K 均值聚类、DBSCAN 等。
- 分类型数据聚类：适用于离散分类型数据，如 K 模式聚类(K-Modes)等。
- 文本聚类：在对文本处理的基础上采用现有的聚类方法进行聚类。



5.2 划分聚类

划分聚类是一种直观且高效的聚类方法，其核心优势在于计算简单，适用于大规模数据和凸形类簇。划分聚类尽管存在需要预设 K 值、对初始条件敏感等局限性，但其变种算法（如 K-Means++、K-Medoids）通过优化初始质心或度量方式，已在实际应用中展现出强大的实用性。在选择聚类方法时，需要结合数据特性（如分布形状、维度、噪声水平）和业务需求（如是否需要层次结构），灵活选用或组合多种算法。

5.2.1 K-Means 聚类

K-Means 算法是典型的基于距离的划分聚类算法，在最小化误差函数的基础上将数据划分为预定的类数 K，采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。

1. K-Means 聚类的基本过程

K-Means 聚类的基本过程如图 5-3 所示。

- (1) 初始化：从 N 个样本数据中选取 K 个对象作为初始质心。
- (2) 分配：将每个样本指派到最近的质心。
- (3) 更新质心：用每个簇的样本均值来更新质心。
- (4) 终止条件：重复步骤(2)和(3)，直到质心不发生明显变化时停止并输出聚类结果。

注意，这里的样本均值不一定是数据集中存在的点。

2. K-Means 聚类中的 K 值

在 K-Means 聚类过程中 K 值是事先给定的，不同的 K 得到不同的簇，而 K 值的选定非常难以估计，常见的方法如下。

- (1) 拍脑袋法：样本量除以 2 再开平方。

$$K \approx \sqrt{\frac{n}{2}}$$

(5.1)

(2) 肘部法则(Elbow Method)：当 K 小于真实聚类数时，由于 K 的增大会大幅增加每个簇的聚合程度，故误差平方和(SSE)的下降幅度会很大，而当 K 达到真实聚类数时，再增加 K 所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，然后随着 K 值的继续增大而趋于平缓，也就是说 SSE 和 K 的关系图是一个手肘的形状，如图 5-4 所示。

sklearn.cluster.KMeans 是 scikit-learn 库中用于实现 K-Means 聚类算法的类。

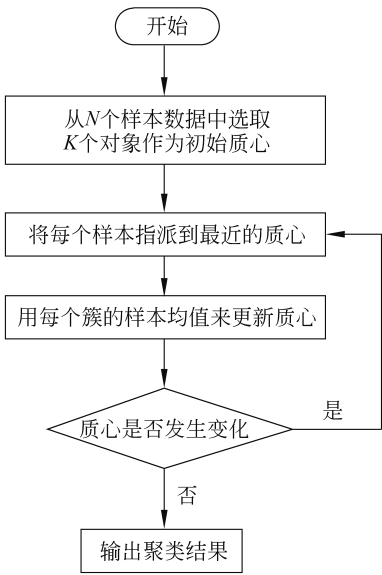


图 5-3 K-Means 聚类过程

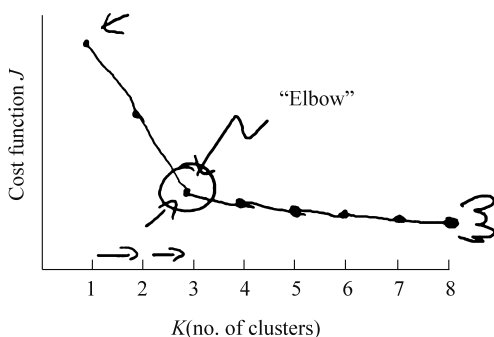


图 5-4 肘部法则示意

KMeans 类的构造函数的语法结构如下,参数介绍如表 5-1 所示。

```
sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=
300, ..., random_state=None, ...)
```

表 5-1 KMeans 类的构造函数的参数介绍

参 数	说 明
n_clusters	希望分成的簇数,默认为 8
init	设置初始质心的方法为'k-means++'(默认)、『random』或『ndarray』(自行设定目标函数变化程度的阈值,小于阈值则停止迭代,默认 1e-4 为最大迭代次数阈值;大于阈值则停止迭代,默认为 300)
n_init	随机使用不同质心来运行 K-Means 算法的次数,默认为 10
max_iter	设置随机种子
random_state	设置随机种子,用于控制初始质心的随机性

KMeans 类常见的属性与方法如下。

- cluster_centers_: 二维数组,形状为(n_clusters, n_features),用于存储每个簇的质心(中心点)坐标。其中,n_clusters 是指定的聚类数量,n_features 是数据的特征数量。例如,在二维平面上进行聚类,每个质心由(x,y)坐标表示;在三维空间中,则由(x, y, z)坐标表示。
- labels_: 一维数组,形状为(n_samples,),用于存储每个样本所属的簇标签,标签的取值范围为 0~n_clusters-1。通过该属性,可以知道每个样本被分配到的簇。
- inertia_: 浮点数,表示所有样本到其所属簇中心距离的平方和,也称为惯性或簇内误差平方和。该属性用于衡量聚类的质量,值越小表示聚类效果越好,即样本点更紧密地围绕在各自的簇中心周围。
- n_iter_: 整数,表示算法收敛所需要的迭代次数。在 K-Means 算法中会不断地更新簇中心,直到满足收敛条件(如达到最大迭代次数或簇中心的变化小于某个阈值),n_iter_记录了实际进行的迭代次数。
- fit(X[,y]): 对输入数据 X 进行聚类分析,找到最优的簇中心和样本划分。y 参数在 K-Means 算法中通常不需要使用,因为它是无监督学习算法。
- fit_predict(X[, y]): 先调用 fit 方法对数据进行聚类,然后返回每个样本所属的簇

标签。这是一个便捷方法,相当于依次调用 fit 方法和 predict 方法。

- predict(X): 根据已经训练好的 K-Means 模型预测新样本 X 所属的簇标签。注意,在调用该方法之前,模型必须已经使用 fit 方法进行了训练。
- transform(X): 将输入数据 X 转换为每个样本到各个簇中心的距离。使用这个方法可以获取样本与各个簇中心的相对位置信息。

实验说明: 肘部法则实例(5_elbowsse_consump.py)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans

# 使用 Pandas 读入数据
data_path = '../data/consumption_data.xls'
x = pd.read_excel(data_path, index_col = 'Id')
# x = 1.0 * (data - data.mean()) / data.std()
# x = 1.0 * (data - data.min()) / (data.max() - data.min())

cost = []
for k in range(2, 11):
    model = KMeans(n_clusters = k, max_iter = 500, random_state = 41)
    model.fit(x)
    cost.append(model.inertia_)

# 绘制 K 值与成本的关系图
plt.plot(range(2, 11), cost, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum of the Squared Error')
plt.title('the SSE')
plt.show()
```

3. K-Means 聚类中的初始质心

K-Means 算法对初始质心是比较敏感的,对于同样的 K 值,选取不同的质心,会影响算法的聚类结果和迭代的次数。如图 5-5 和图 5-6 所示,两个不同的初始质心,聚类的结果不同。随机初始化质心是常见的做法,但是该方法不一定能得到最优的聚类结果。

为了避免这种情况出现,进行以下步骤:

- (1) 多次运行,每次选取一组不同的随机初始质心。
- (2) 选取具有最小目标函数值的簇集,使用层次聚类对样本聚类。
- (3) 从层次聚类中提取 K 个簇,并选择这些簇的中心作为初始质心,选择的初始质心要尽可能互相远离。

4. K-Means 聚类算法的结束条件

K-Means 聚类算法的结束条件通常选择下面一种。

- (1) 质心不再发生变化: 这是最理想的结束条件,即所有的点都不会从一个簇转移到另一个簇,质心的位置也不再发生变化。在实际应用中,通常会设置一个非常小的阈值,当

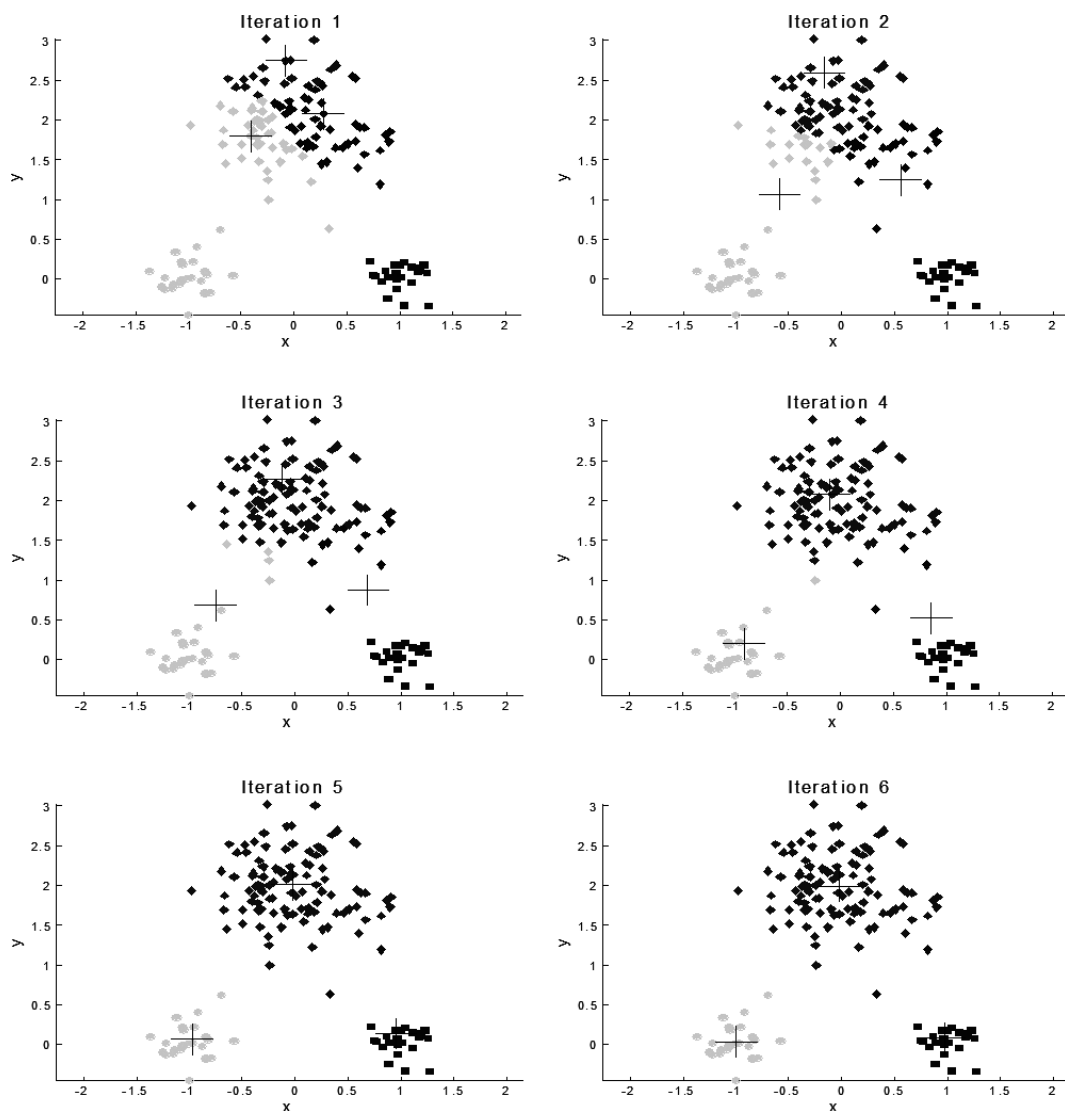


图 5-5 初始质心 1 的聚类过程

质心的变化量小于这个阈值时,可以认为聚类已经收敛。

(2) 达到最大迭代次数: 为了避免算法陷入无限循环,通常会设置一个最大迭代次数。当算法的迭代次数达到这个预设值时,无论聚类是否完全稳定,算法都会停止迭代。

(3) 簇内样本的分配不再变化: 在连续几次迭代中,如果数据点的簇分配情况不再发生变化,也可以认为聚类过程已经收敛,此时可以停止迭代。

(4) 簇内平均距离变化量: 另一种常见的结束条件是考虑簇内数据点之间的平均距离变化量,当连续几次迭代中簇内数据点之间的平均距离不再发生显著变化时,算法停止迭代。

(5) 误差平方和(SSE): K-Means 聚类的目标函数是最小化簇内误差平方和,即簇内每个点到簇中心的距离平方和。如果在某次迭代前后 SSE 没有发生显著变化,则说明簇的分配不再发生变化,算法已经收敛。

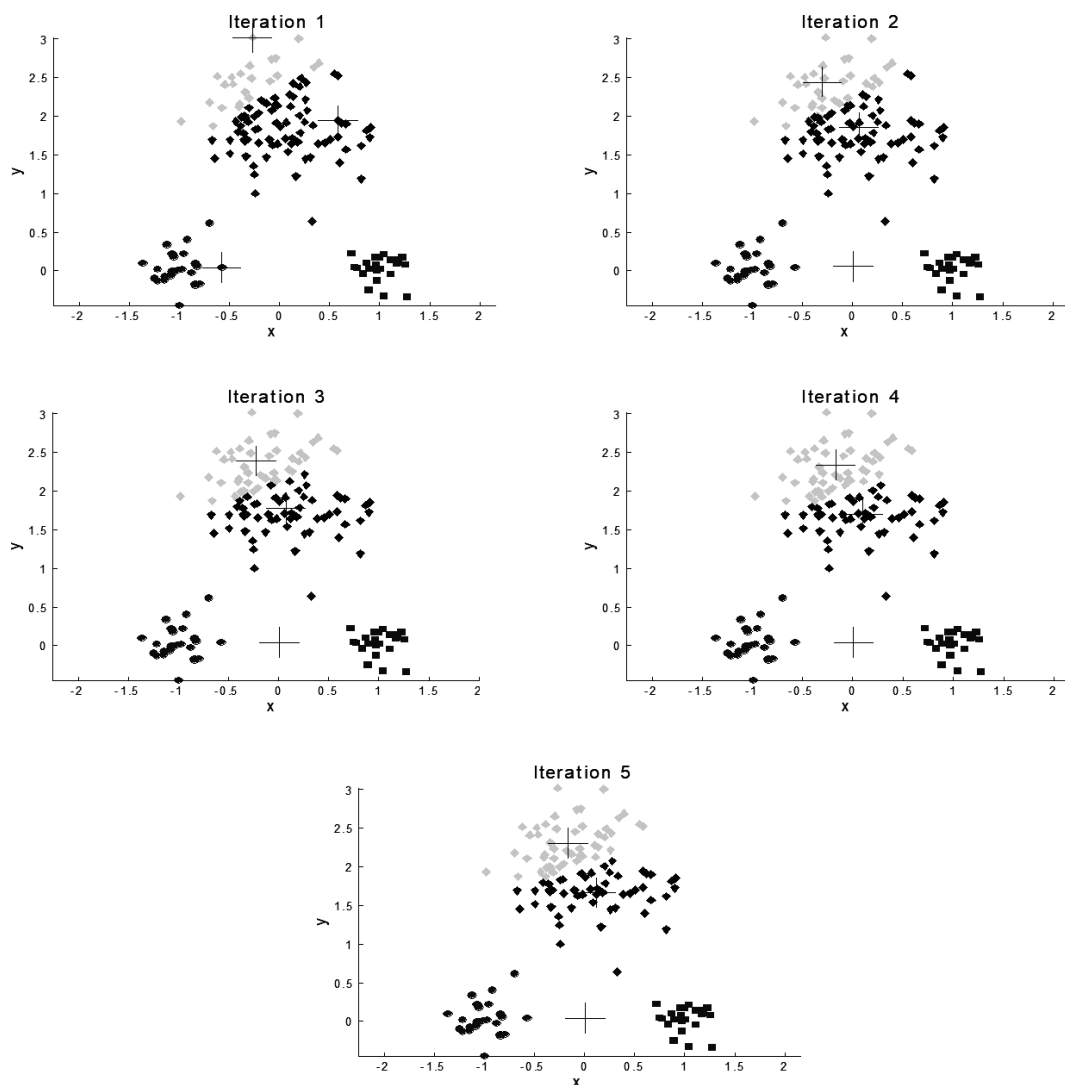


图 5-6 初始质心 2 的聚类过程

5. K-Means 聚类的目标函数

K-Means 聚类的目标函数是误差平方和,误差平方和可用于评估聚类的优劣。对于两种不同的聚类结果,选择 SSE 小的分类结果。

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} (c_i - x)^2 \quad (5.2)$$

目标函数为误差平方和,决定了聚类中质心的选择,理由如下。

$$\begin{aligned} \frac{\partial}{\partial c_k} SSE &= \frac{\partial}{\partial c_k} \sum_{i=1}^K \sum_{x \in C_i} (c_i - x)^2 \\ &= \sum_{i=1}^K \sum_{x \in C_i} \frac{\partial}{\partial c_k} (c_i - x)^2 \end{aligned} \quad (5.3)$$

$$= \sum_{x \in C_k} 2(c_k - x_k) = 0$$

误差平方和对质心求偏导,得到质心是簇内对象的平均值。

$$\sum_{x \in C_k} 2(c_k - x_k) = 0 \Rightarrow m_k c_k = \sum_{x \in C_k} x_k \Rightarrow c_k = \frac{1}{m_k} \sum_{x \in C_k} x_k \quad (5.4)$$

如果使用绝对误差和作为度量聚类质量的目标函数,则质心的选择有所不同。

$$\begin{aligned} \text{SAE} &= \sum_{i=1}^K \sum_{x \in c_i} |c_i - x| \\ \frac{\partial}{\partial c_i} \text{SAE} &= \frac{\partial}{\partial c_k} \sum_{i=1}^K \sum_{x \in c_i} |c_i - x| \\ &= \sum_{i=1}^K \sum_{x \in c_i} \frac{\partial}{\partial c_k} |c_i - x| \\ &= \sum_{x \in c_i} \frac{\partial}{\partial c_k} |c_k - x| = 0 \end{aligned}$$

$$\sum_{k \in C_1} \frac{\partial}{\partial c_k} |c_k - x| = 0 \Rightarrow \sum_{k \in C_k} \text{sign}(x - c_k) = 0$$

由此,得到质心 $c_k = \text{median}\{x \in C_k\}$,即簇中各点的中位数。

6. K-Means 聚类的优缺点

K-Means 聚类的优点是算法非常高效,且伸缩性较好;原理简单,实现容易。其缺点是聚类中心的个数 K 需要事先给定。在实际应用中,这个 K 值的选定是非常难以估计的,并且对噪声和离群点敏感。另外,聚类结果依赖于初始质心的选择,容易陷入局部最优。因此,该方法不适用于发现非凸面形状的簇(见图 5-7)或大小差别很大的簇(见图 5-8)。

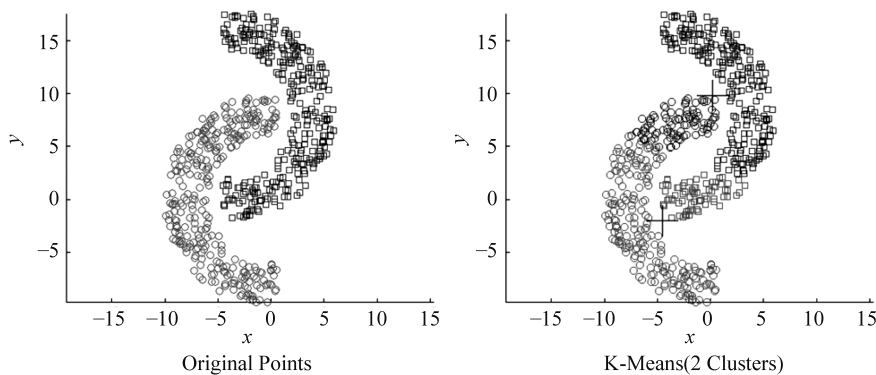


图 5-7 非凸面聚类情况

在 K-Means 的基础上提出了以下改进方法:

- (1) 针对 K-Means 对初始值设置的敏感问题,提出了 K-Means++、Intelligent K-Means、Genetic K-Means。
- (2) 针对 K-Means 对噪声和离群值非常敏感,提出了 K-Medoids 和 K-Medians。
- (3) 针对 K-Means 只适用于数值类型数据,不适用于离散类型数据,提出了 K-Modes。
- (4) 针对 K-Means 不能解决非凸(Non-Convex)数据,提出了 Kernel K-Means。

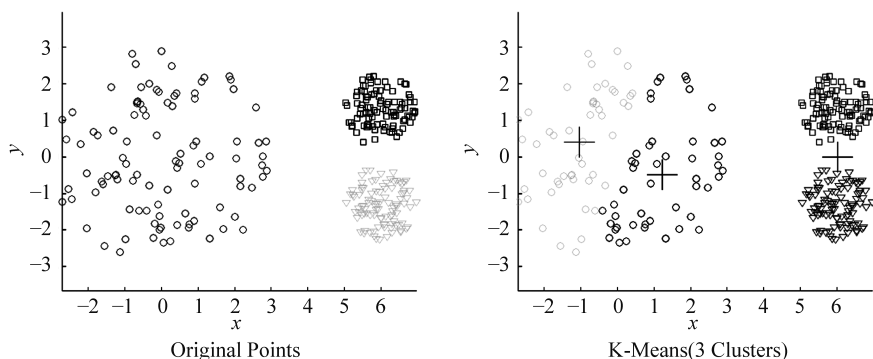


图 5-8 大小差别很大的聚类情况

7. K-Means 聚类的评价方法

1) 轮廓系数

对于每个样本,轮廓系数都需要计算 a 、 b 两个值。其中, a 是这个样本和同簇内其他样本间的平均距离, b 是这个样本和距离第二近的簇内所有样本间的平均距离。单个样本的轮廓系数计算如下。

凝聚度 $a(i)$: 样本 i 和同簇内其他样本间的平均距离。

$$a(i) = \frac{1}{|C_i| - 1} \sum_{\substack{j \in C_i \\ j \neq i}} \text{dis}(j, i) \quad (5.5)$$

分离度 $b(i)$: 样本 i 和距离第二近的簇内所有样本间的平均距离。

$$b(i) = \min \frac{1}{|C_j|} \sum_{j \in C_j} \text{dis}(j, i) \quad (5.6)$$

单个样本的轮廓系数计算:

$$s(i) = \frac{b(i) - a(i)}{\max(b(i), a(i))} \quad (5.7)$$

所有样本的轮廓系数由每个样本的轮廓系数求平均:

$$s = \frac{1}{N} \sum_i s(i) \quad (5.8)$$

平均轮廓系数的取值范围为 $-1 \sim 1$, 越接近 1 说明聚类效果越好, 平均轮廓系数最大的 k 是最佳聚类数。

`sklearn.metrics.silhouette_score` 是 `scikit-learn` 库中的一个函数, 用于计算聚类结果的轮廓系数 (Silhouette Coefficient)。轮廓系数的取值范围为 $-1 \sim 1$, 其值越大, 表示聚类效果越好。其值接近 1, 表示样本与自身所在的聚类紧密相关, 且与其他聚类分离得较好, 说明聚类效果理想; 接近 0, 表示样本可能位于两个聚类的边界上, 聚类效果不太明确; 接近 -1 , 表示样本可能被错误地分配到其他聚类中, 聚类效果较差。

`sklearn.metrics.silhouette_score` 的语法结构如下, 参数介绍如表 5-2 所示。

```
sklearn.metrics.silhouette_score(X, labels, metric='euclidean', sample_size=
None, random_state=None, ...)
```

表 5-2 sklearn.metrics.silhouette_score 的参数介绍

参 数	说 明
X	必需参数,它是一个二维数组,形状为(n_samples,n_features)。其中,n_samples 表示样本的数量,n_features 表示每个样本的特征数量。该参数用于指定聚类分析的原始数据集
labels	必需参数,它是一个一维数组,形状为(n_samples,)。数组中的每个元素对应 X 中一个样本的聚类标签,用于指定每个样本所属的聚类
metric	可选参数,字符串或可调对象,默认值为'euclidean'。该参数指定了计算样本之间的距离时所使用的度量方法,常见的取值如下。 'euclidean': 欧氏距离,最常用的距离度量方式。 'manhattan': 曼哈顿距离。 'cosine': 余弦距离。 用户也可以传入自定义的距离计算函数
sample_size	可选参数,整数,默认值为 None。如果指定了该参数,会从数据集中随机抽取 sample_size 个样本用于计算轮廓系数,这样可以在样本数量较大时减少计算量

实验说明：轮廓系数评估

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# 生成一些示例数据
X=np.array([[1, 2], [1, 4], [1, 0], [4, 2], [4, 4], [4, 0]])

# 使用 KMeans 进行聚类
kmeans =KMeans(n_clusters=2, random_state=42).fit(X)
labels =kmeans.labels_

# 计算轮廓系数
silhouette_avg=silhouette_score(X, labels)
print(f"平均轮廓系数: {silhouette_avg}")
```

2) Calinski-Harabasz 指数

Calinski-Harabasz(CH)指数的公式如下：

$$CH(k) = \frac{\text{tr}(\mathbf{B})/k - 1}{\text{tr}(\mathbf{W})/n - k} \quad (5.9)$$

其中, n 表示训练样本集数目, k 表示类别数, $\mathbf{B}(k)$ 为类别之间的协方差矩阵, $\mathbf{W}(k)$ 为类别内部数据的协方差矩阵, tr 表示矩阵的迹。通常,类别内部数据的协方差越小越好,类别之间的协方差越大越好,这样的 Calinski-Harabasz 指数会高。

sklearn.metrics.calinski_harabasz_score 是 scikit-learn 库中用于评估聚类效果的一个函数,它通过计算 Calinski-Harabasz 指数(也称为方差比准则)来衡量聚类结果的质量,返回一个浮点数,即 Calinski-Harabasz 指数。该值越大,意味着聚类效果越理想。

sklearn.metrics.calinski_harabasz_score 的语法格式如下,参数介绍如表 5-3 所示。