第5章

K-Means聚类算法

在未知模式识别问题中，通常需要从一堆没有标签的数据中找到其中的关联，发现数据之间的相似性，也被称为聚类，同时可以统计数据在空间上的分布，即密度估计。聚类就是将相似的事物聚集在一起，将不相似的事物划分到不同类别的过程，是无监督学习中最重要的方法之一。聚类算法的目标是将数据集分成若干簇，使得同一簇内的数据点相似度尽可能大，而不同簇间的数据点相似度尽可能小。聚类与分类的区别是其要划分的类是未知的。学术界提出了多种基于不同思想的聚类算法，主要有基于划分的聚类算法（K-Means 算法等）、基于层次的聚类算法（如 BIRCH 算法等）、基于密度的聚类算法（如 DBSCAN 算法等）、基于网格的聚类算法（如 STING 算法等）和基于模型的聚类算法等。这些算法都能取得不错的聚类效果，能适应各个方面和不同环境的需求。

聚类要求如下。

（1）可伸缩性。常规的聚类算法对于小数量（<200）的数据集聚类效果较好，但对于包含几百万对象的大数据集聚类效果较差。

（2）处理不同类型属性的能力。许多算法被设计用来聚类数值类型的数据，但是应用可能要求聚类其他类型的数据，如二元类型（binary）、分类/标称类型（categorical/nominal），或者这些数据类型的混合。

（3）发现任意形状的聚类。许多聚类算法是基于欧氏距离或者曼哈顿距离度量的。这样的距离度量算法趋向于发现具有相近尺度和密度的球状簇。但是，一个簇可能是任意形状的，提出能发现任意形状簇的算法是很重要的。

（4）用于决定输入参数的领域知识最小化。许多聚类算法在聚类分析中要求用户输入一定的参数，如希望产生的簇的数目等。聚类结果对于输入参数十分敏感，因此输入参数通常很难确定，特别是对于包含高维对象的数据集来说更为困难。

（5）处理“噪声”数据的能力。绝大多数现实中的数据库都包含孤立点、缺失或者错误的数据。有些聚类算法对于这样的数据敏感，可能导致低质量的聚类结果。

（6）对于输入记录的顺序不敏感。一些聚类算法对于输入数据的顺序是敏感的。同一个数据集合，当以不同的顺序交给同一个算法时，可能生成差别很大的聚类结果。开发对数据输入顺序不敏感的算法具有重要的意义。

（7）可解释性和可用性。聚类结果希望是可解释的、可理解的和可用的，也就是说，聚

类需要和特定的语义解释和应用相联系。

5.1 K-Means 聚类算法原理

基本 K-Means 聚类算法的思想简单，事先确定常数 K，即最终的聚类类别数，首先随机选定初始点为质心，并通过计算每一个样本与质心之间的相似度(如欧氏距离)，将样本点归到最相似的类中，接着重新计算每个类的质心(即为类中心)，重复这样的过程，直至质心不再改变，最终确定每个样本所属的类别及每个类的质心。由于每次都要计算所有的样本与每一个质心之间的相似度，因此在大规模的数据集上，K-Means 算法的收敛速度比较慢。

K-Means 算法主要分为以下 4 个步骤。

1. 初始化

首先确定要划分的簇的数量 K。这通常是根据先验知识或者通过一些经验法则来确定的。例如，如果对用户进行聚类，可以根据业务需求大致估计用户群体的类别数。然后从数据集中随机选择 K 个数据点作为初始的聚类中心。例如，在一个 2 维平面上的数据点集{(1,1),(3,4),(5,6),(7,8),(9,10)}，如果 $K=2$，随机选择 (1,1)和(5,6)作为初始聚类中心。

2. 分配数据点到簇

对于数据集中的每个数据点，计算它到各个聚类中心的距离。例如，对于数据点(x,y)和聚类中心(x_0,y_0)，欧氏距离为 $d=\sqrt{(x-x_0)^2+(y-y_0)^2}$。

将数据点分配到距离它最近的聚类中心所在的簇。例如，计算出数据点(3,4)到上述两个聚类中心(1,1)和(5,6)的距离，假设到 (1,1)的距离是$\sqrt{(3-1)^2+(4-1)^2}=\sqrt{13}$，到(5,6)的距离是$\sqrt{(3-5)^2+(4-6)^2}=\sqrt{8}$，就将(3,4)分配到距离聚类中心(1,1)最近的簇。以此类推，数据点(7,8)到上述两个聚类中心(1,1)和(5,6)的距离，分别为$\sqrt{(7-1)^2+(8-1)^2}=\sqrt{85}$，$\sqrt{(7-5)^2+(8-6)^2}=\sqrt{8}$，因此，(7,8)分配到距离聚类中心(5,6)最近的簇。同理，(9,10)也分配到距离聚类中心(5,6)最近的簇中。

3. 更新聚类中心

对于每个簇，重新计算其聚类中心。新的聚类中心是该簇中所有数据点的均值。例如，一个簇中有数据点{(1,2),(3,4)}，那么新的聚类中心 x 坐标为(1+3)/2=2，y 坐标为(2+4)/2=3。

4. 重复步骤 2 和步骤 3

不断重复分配数据点和更新聚类中心的过程，直到聚类中心不再发生显著变化(可以通过设定一个收敛阈值来判断，例如，两次迭代之间聚类中心的位置变化小于某个很小的值)或者达到预设的最大迭代次数。

5.2 K-Means 聚类算法特点及应用

K-Means 聚类算法的优点如下。

(1) 原理简单，容易理解和实现。代码实现相对简洁，对于初学者来说容易上手。

（2）计算速度快。在处理大规模数据集时，相比一些复杂的聚类算法，K-Means 的计算效率较高，因为它主要涉及距离计算和简单的均值计算。

（3）对于球状分布的数据聚类效果较好。如果数据在空间中呈现出比较明显的球状分布，K-Means 能够有效地将数据划分到不同的簇中。

K-Means 聚类算法的缺点如下。

（1）需要预先指定 K 值。如果 K 值选择不当，会导致聚类结果不理想。例如，如果 K 值过大，可能会将一个本来应该是一个整体的簇划分成多个小簇；如果 K 值过小，可能会把多个不同的簇合并成一个簇。

（2）对初始聚类中心敏感。不同的初始聚类中心选择可能会导致不同的聚类结果。例如，在一个复杂的数据分布中，随机选择的初始聚类中心可能会使算法收敛到局部最优解，而不是全局最优解。

（3）只能发现球形的簇。对于非球形（如环形、月牙形等）的数据分布，K-Means 算法的聚类效果不佳，因为它是基于距离和均值的计算来划分簇的。

K-Means 聚类算法的典型应用如下。

（1）客户细分：在市场营销中，K-Means 算法可以根据客户的购买行为、消费金额、年龄等特征将客户划分为不同的群体。例如，将客户分为高价值客户、中等价值客户和低价值客户等不同类别，以便企业针对不同群体制定个性化的营销策略。

（2）图像压缩：在图像处理领域，K-Means 算法可以用于图像颜色量化。将图像中的像素颜色划分为 K 个簇，然后用每个簇的中心颜色来代表该簇中的所有颜色。这样可以减少图像中颜色的种类，达到压缩图像的目的。

（3）文档聚类：在文本挖掘中，K-Means 算法可以根据文档的内容（如主题、词汇频率等）将文档聚类。例如，在新闻文章聚类中，将关于体育、政治、娱乐等不同主题的新闻文章分别聚成不同的簇，方便用户浏览和信息检索。

5.3 K-Means 聚类算法实例

5.3.1 自建数据集聚类

问题描述：自己创建一个大小为 500 的数据集，对这些数据进行分类，步骤如下。

（1）创建数据集。首先创建一个由 500 个样本构成的数据集合，并自定义划分为 4 类。具体代码如下。

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

#自己创建一个大小为 500 的数据集，由 2 维特征构成
X, y = make_blobs(n_samples = 500,n_features = 2,centers = 4,random_state = 1)
#具体如下
color = ["red","pink","orange","gray"]
fig, ax1 = plt.subplots(1)
for i in range(4):
    ax1.scatter(X[y == i, 0], X[y == i, 1]
```

```
                ,marker = 'o' #点的形状
                ,s = 8 #点的大小
                ,c = color[i]
                )
plt.show()
```

以上代码输出结果如图 5.1 所示。

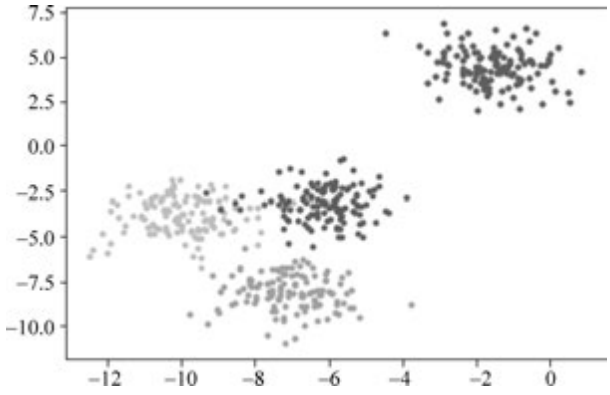

图 5.1 自建数据集的 4 类聚类结果(见彩图)

(2) 基于分布进行聚类。先假设这些数据中有 3 簇,即分成 3 类,代码如下。

```
from sklearn.cluster import KMeans
n_clusters = 3     #首先测试的超参数
cluster = KMeans(n_clusters = n_clusters, random_state = 0).fit(X)
y_pred = cluster.labels_
centroid = cluster.cluster_centers_
#其中,n_clusters 代表聚类的簇数,y_pred 代表对应元素聚类的类别,centroid 代表每个类别的中
#心点
#打印聚类结果
color = ["red","pink","orange","gray"]
fig, ax1 = plt.subplots(1)
for i in range(n_clusters):
    ax1.scatter(X[y_pred == i, 0], X[y_pred == i, 1]
                ,marker = 'o'
                ,s = 8
                ,c = color[i]
                )#循环画小样本预测点的位置
    ax1.scatter(centroid[:,0],centroid[:,1]
                ,marker = "x"
                ,s = 18
                ,c = "white")   #循环画中心点的位置
plt.show()
```

输出结果如图 5.2 所示。

从聚类结果可见,分成 3 类效果不错。但是最开始生成的是 4 类数据,同样地,以 4 类来进行聚类,即假设数据分成 4 簇,打印一下聚类效果。代码如下。

```
from sklearn.cluster import KMeans
n_clusters = 4       #首先测试的超参数
cluster = KMeans(n_clusters = n_clusters, random_state = 0).fit(X)
y_pred = cluster.labels_
```

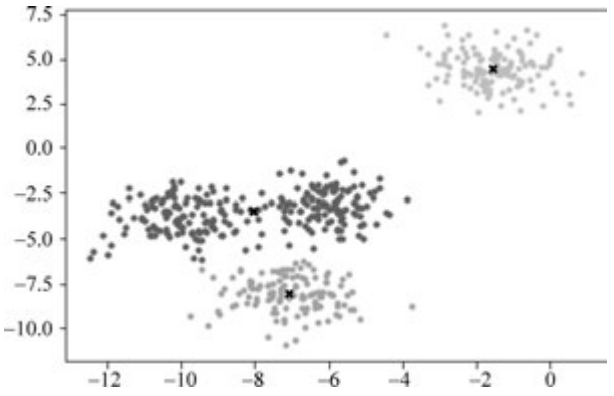

图 5.2　自建数据集的 3 类聚类结果(见彩图)

```
centroid = cluster.cluster_centers_
color = ["red","pink","orange","gray"]
fig, ax1 = plt.subplots(1)
for i in range(n_clusters):
    ax1.scatter(X[y_pred == i, 0], X[y_pred == i, 1]
                ,marker = 'o'
                ,s = 8
                ,c = color[i]
                )#循环画小样本预测点的位置
    ax1.scatter(centroid[:,0],centroid[:,1]
                ,marker = "x"
                ,s = 18
                ,c = "white")   #循环画中心点的位置
plt.show()
```

输出结果如图 5.3 所示。

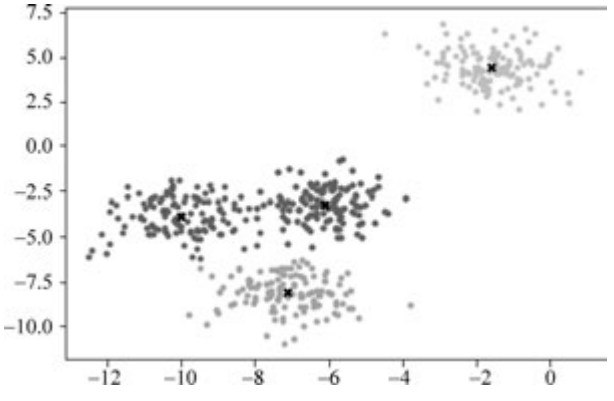

图 5.3　自建数据集的 4 类聚类结果(见彩图)

可见,分成 4 类的输出结果和原始数据更接近,效果更好。

5.3.2　客户分组画像

问题描述:银行采集了 200 位客户的信息数据,每位客户包括社保号码(Profile Id)、姓名(Name)、年龄(Age)和存款数量(Deposit)4 个特征。使用 K-Means 算法生成各类客户

的特点画像，即对客户进行分组。

素材文件见 Customer_Infor.csv，完整的程序实现代码如下。

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans

#客户存款、年龄数据集
dataset = pd.read_csv('Customer_Info.csv')
X = dataset.iloc[:, [4,3]].values

#在数据集上使用 k = 3 进行聚类
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state =
0)
y_kmeans = kmeans.fit_predict(X)

#集群可视化
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, marker = '^', c = 'red',
label = 'Not very rich')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, marker = 'o', c = 'green',
label = 'Middle')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, marker = '*', c = 'blue',
label = 'Rich')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 250, c =
'yellow', label = 'Centroids')
plt.title('Clusters of customer Info')
plt.xlabel('Deposit')
plt.ylabel('Age')
plt.legend()
plt.show()
```

输出结果如图 5.4 所示。

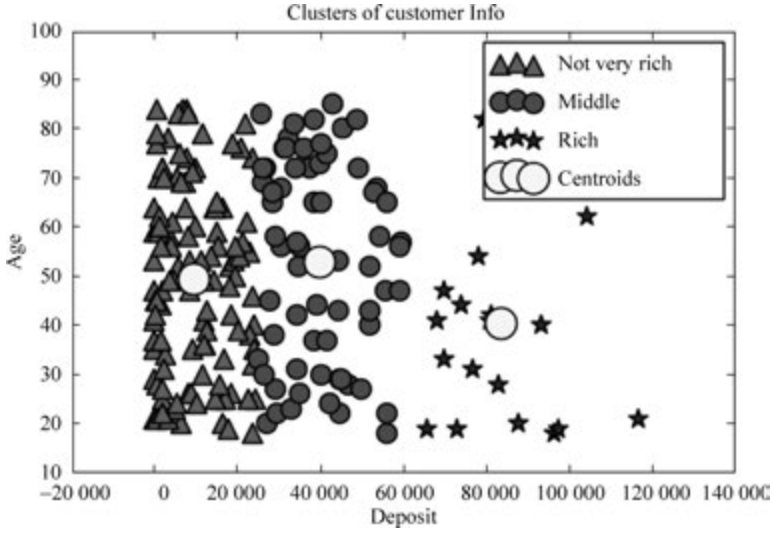

图 5.4 客户数据集聚类结果（见彩图）

小结

本章主要介绍了 K-Means 聚类算法的相关原理、概念、特点及应用，其优点是简单、速度快，缺点是聚类结果与初始中心的选择有关系，且必须提供聚类数目，如果不知道样本最适合聚成多少类别，使用 K-Means 是不适合的。最后以两个 K-Means 算法实例具体阐述 K-Means 聚类算法实现过程。

思考题

1. 如何确定 K-Means 聚类算法中的最佳 K 值？

2. K-Means 聚类算法中可以使用哪些距离度量方法？不同的距离度量方法对聚类结果有何影响？

3. K-Means 聚类算法中初始中心点的选择对聚类结果有何影响？