Lab2：聚类算法

林泽春　2200012868

2023 年 9 月 29 日

目录

§1 实验目的

1. 了解 1-Wire 总线并借助 1-Wire 总线学习使用温度传感器和 glob 库；
2. 了解并学习使用 Numpy 库和 Matplotlib 库进行数据处理和数据可视化；
3. 了解聚类算法原理并自己实现；

§2 实验内容

2.1 1-Wire 总线及相关内容

2.1.1 1-Wire 总线简介

1-Wire 总线 (单总线) 是 Maxim 全资子公司 Dallas 的一项专有技术。与目前多数标准串行数据通信方式 SPI/I2C/MICROWIRE 不同，它采用单根信号线，既传输时钟又传输数据，而且数据传输是双向的。1-Wire 总线接口具有节省 I/O 资源、结构简单、成本低廉、便于总线扩展和维护等诸多优点。

1-Wire 总线被定义为仅有一根数据线，数据线连接一个 4.7K 欧姆的上拉电阻，电阻再接到电源 (3V 到 5.5V)。每个设备 (主设备或从设备) 通过一个漏极开路或 3 态门引脚连接至数据线上。这就允许每个设备“释放”数据线，当设备没有传递数据的时其他设备可以有效地使用数据线。

2.1.2 温度传感器 DS18B20 的基本使用

树莓派提供了 2 个驱动模块 w1-gpio 与 w1-therm，分别用于通过 GPIO 扩展 1-Wire 总线接口及提供对温度传感器 DS18B20 的读写控制。其中，w1-gpio 提供了 1-Wire 总线的 I/O 操作方法，w1-therm 提供了对温度传感器 DS18B20 的内部操作方法。

在文件系统在/sys/bus/w1/devices 目录下会发现一个以 28-XXXX 开头的文件夹 (如果接多个 DS18B20，将会看到多个 28-xxxx 的文件，分别对应各个 DS18B20)。进入以 28-XXXX 开头的文件夹，显示文件夹会发现一个名为 w1_slave 的文件，利用 cat 命令读取文件 w1_slave 的内容会返回温度传感器当前的温度值。

2.1.3 glob 库

在不知道文件具体名称的时候如果要查看目录中的文件或者目录，可以使用 python 的标准库 glob，它的参数是要查看的文件名，一般通过使用通配符来查找多个文件。例如下面代码列出 dir 目录下的全部文件：

```
import glob
for name in glob.glob(dir/*):
    print(name)
```

常用的通配符有：

星号（*）匹配 0 个或多个字符

问号（?）匹配 1 个字符

中括号（[]）表示范围，例如 [a-k] 表示匹配这个范围的全部字符

如果需要递归的查找可以使用 recursive=True 参数，例如：

```
import glob
files  = glob.glob(**/*.txt)
```

此时 files 是当前目录下全部扩展名为 txt 的文件列表。这里使用两个星号用来匹配 0 个或多个目录，仅在递归模式下使用。

2.2 NumPy 库和 Matplotlib 库

2.2.1 NumPy 库

NumPy 是 Python 中科学计算的基础软件包。它是一个提供多了维数组对象，多种派生对象（如：掩码数组、矩阵）以及用于快速操作数组的函数及 API，它包括数学、逻辑、数组形状变换、排序、选择、I/O、离散傅立叶变换、基本线性代数、基本统计运算、随机模拟等等。

可以用它生成特定分布的随机数。例如下面代码生成了 10 个方差为 sigma，均值为 mean 的正态分布随机数。

```
import numpy as np
mean = 5
sigma = 1.3
x=mean+sigma*np.random.randn(10)
```

NumPy 包的常用数据类型是 ndarray 对象。它与标准 Python Array（数组）之间有几个重要的区别：

1. NumPy 数组在创建时具有固定的大小，与 Python 的原生数组对象（可以动态增长）不同。更改 ndarray 的大小将创建一个新数组并删除原来的数组。

2. NumPy 数组中的元素都需要具有相同的数据类型，因此在内存中的大小相同。例外情况：Python 的原生数组里包含了 NumPy 的对象的时候，这种情况下就允许不同大小元素的数组。

3. NumPy 数组有助于对大量数据进行高级数学和其他类型的操作。通常，这些操作的执行效率更高，比使用 Python 原生数组的代码更少。

NumPy 中还提供了 np.means(),np.sum(),np.append(),np,square() 等函数对数组中的元素和数组进行方便的操作。

2.2.2 Matplotlib 库

Matplotlib 是一个 Python 2D 绘图库，可以生成各种格式和跨平台交互式环境的出版物质量图片数据。它的语法格式类似于 matlab，例如下面代码画出 x 数据的直方图。

```
import matplotlib.pyplot as plt
plt.hist(x,80,histtype=bar,facecolor=yellowgreen,alpha=0.75)
plt.show()
```

同时还有可以绘制散点图的 plt.scatter() 等函数，功能强大。

2.3 聚类算法简介

聚类算法试图将数据集中的样本分成若干个通常不相交的子集，每个子集称为一个簇（cluster）。通过这样的划分，每个簇就可能对应于一个潜在的概念，例如可以把生物分成

植物和动物，把书籍分成文史类和科技类等。这种分类方式在人类认识世界的过程中不断的在进行着，但如果让计算机掌握这样的分类能力就不那么容易了。

本实验采用一种非常简单的聚类算法，即 K 均值聚类算法 (K-means clustering)。

K 均值聚类算法是一种迭代求解的聚类分析算法，其步骤是随机选取 K 个位置作为初始的聚类中心，然后计算每个样本与各个聚类中心之间的距离，把每个样本分配给距离它最近的聚类中心。聚类中心以及分配给它们的样本就代表一个聚类。每分配完成后，聚类的聚类中心会根据聚类中现有的样本被重新计算，获得新的聚类中心。

这个过程将不断重复直到满足某个终止条件。终止条件可以是没有（或最小数目）样本被重新分配给不同的聚类，或者是聚类中心不再发生变化。

2.4 实验题目的实现

2.4.1 读取温度传感器的值

使用 Python 代码获取当前温度值并显示到屏幕上：

```
import glob
import os

for name in glob.glob(/sys/bus/w1/devices/2*): # 利用通配符查找满足要求的文件夹
    folder = name + /w1_slave # 拼接字符串获得目标文件路径
    with open(folder, r) as f: #利用open()函数打开对应文件并输出
        for line in f:
            print(line)
```

2.4.2 一维数据聚类算法实现

假定某共用办公室有两个人使用，他们使用办公室的时候都会用空调遥控器设置房间的温度。但空调遥控器显示面板坏掉了，只能通过加减温度的方式盲调整。已知他们习惯的温度不同，办公室的温度计有记录功能，每天记录 12 小时的温度，每 30 分钟记录一次，

根据一个月所记录的数据，通过聚类算法，算出这两个人的喜好温度，并估算他们各使用办公室多少时间。

可以认为这两个人使用办公室的时候室内温度是不同方差和不同均值的正态分布随机数，采用 randn 函数生成模拟数据，并用 matplotlib 画出数据的直方图。使用 K 均值聚类算法将模拟数据分成两个簇，解答前面的问题。

使用温度传感器读入温度，根据温度值，判断是谁在使用办公室。

解决如上问题的代码如下：

```
import glob
import numpy as np
import matplotlib.pyplot as plt

# 随机生成两组不同的满足正态分布的数据并将它们拼接起来
t1 = np.random.randn(360)
t2 = np.random.randn(360)
x1 = np.random.randint(20, 28)
x2 = np.random.randint(20, 28)
temp = np.append(t1, t2)

# 每次添加新元素后维护聚点，返回值为新的聚点
def change(s):
    m = np.mean(s)
    return m

# 初始聚点随机选取
k1, k2 = [temp[0] + x1], [temp[361] + x2]
m1, m2 = k1[0], k2[0]

# 遍历及分类
for i in range(2, temp.size):
    if abs(temp[i] - m1 + x1) < abs(temp[i] - m2 + x2):
        k1.append(temp[i] + x1)
        m1 = change(k1)
    elif abs(temp[i] - m1 + x1) > abs(temp[i] - m2 + x2):
        k2.append(temp[i] + x2)
```

```
        m2 = change(k2)

# 数据以直方图的形式进行可视化输出，同时输出分类结果，与已知均值比较
plt.hist(k1, bins = 360, facecolor = r, alpha = 0.5)
plt.hist(k2, bins = 360, facecolor = b, alpha = 0.5)
plt.show()
print(x1:, x1,x2:, x2)
print(m1:, m1,m2:,m2)
```

2.4.3 二维数据的聚类

分类思路与一维相似，但本题中使用欧式距离为分类标准，实现代码如下：

```
import matplotlib.pyplot as plt
import numpy as np

# 进行生成数据，拼接初始数据等处理
r1 = 2
r2 = 4
p1 = []
p2 = []
p = np.array([[np.random.rand() * r2 * 2 - r2, np.random.rand() * r2 * 2 - r2]])
for i in range(10000):
    x = np.random.rand() * r2 * 2 - r2;
    y = np.random.rand() * r2 * 2 - r2;
    if abs(x * x + y * y - r1) < 0.5:
        p1.append([x, y])
    if abs(x * x + y * y - r2) < 0.5:
        p2.append([x, y])
    p = np.append(p, np.array([[x, y]]), axis = 0)
t1 = np.mean(p1, axis = 0)
t2 = np.mean(p2, axis = 0)

# 维护新聚点的函数
def change(tp):
    r = np.mean(tp, axis = 0)
```

```
    return r

m , n = np.array([p[0]]) , np.array([p[1]])
result1 , result2 = m[0], n[0]

# 二维数据聚类算法
for i in range(2, int(p.size / 2)):
    if abs(np.sqrt(np.sum(np.square(p[i] - t1))) - r1) < abs(np.sqrt(np.sum(np.square(p[i] - t2
        ))) - r2):
        if abs(np.sqrt(np.sum(np.square(p[i] - t1))) - r1) < 0.5:
            m = np.append(m, np.array([p[i]]), axis = 0)
            result1 = change(m)
    else:
        if abs(np.sqrt(np.sum(np.square(p[i] - t2))) - r2) < 0.5:
            n = np.append(n, np.array([p[i]]), axis = 0)
            result2 = change(n)

# 数据可视化及输出
plt.scatter(m[:,0], m[:,1], c=r, s=3)
plt.scatter(n[:,0], n[:,1], c=b, s=3)
plt.show()
print(t1, t2, result1, result2)
```

§3 思考题

3.1 分析实验结果，得到的聚类中心是否与模拟数据的均值相等，不等的话请解释偏差的原因

得到的聚类中心与模拟数据的均值接近但不相等，因为聚类算法初始的聚类中心是随机选取的，最终得到的聚类中心是在初始聚类中心条件下得到的局部最优解。

3.2 讨论 k 均值算法的局限性

1. 对特征较为接近的数据分类效果较差

2. 若初始聚类中心选取较差，分类结果较差。

3.k 的选取对结果影响大。

4. 凸数据集难收敛。

5. 若数据本身不平衡，则聚类效果差。