实验四　K-means 聚类

一、实验目的

1. 熟悉 Python 的基础使用
2. 理解分类和聚类的区别
3. 理解和掌握 k-均值聚类的基本原理和实现过程
4. 能够利用 k-均值聚类解决实际问题

二、实验内容

1. 导入实验所需数据集
2. 比较两个案例中数据集的不同
3. 比较两个案例解决问题的区别
4. k-均值聚类解决实际问题

三、实验原理：见课本

k-means 聚类算法描述如下：

输入：样本集

过程：从样本集中随机选择 k 个样本作为初始向量，即 k 个初始质心点

—repeat

—对样本集中的每个数据

—对于当前数据

—计算当前数据与 k 个质心之间的距离

—将当前数据加入到距离其最近的簇

—对于每个簇，计算其均值，即得到新的 k 个质心点

—util 迭代终止条件满足为止输出：簇划分

任务 1：根据随机生成的二维数据，进行聚类

任务要求：

（1）运行调试程序（复制时注意缩近问题），输出结果

（2）针对下面的指令，改变划线部分的数据取值，重新生成数据集完成聚类，保存结果，说明这四个参数对结果图形的影响。

```
arr = np.random.randint(100, size=(100, 1, 2))[:, 0, :]
```

（3）改变聚类簇数目，分别取 2，3，4，6，观察结果的不同

（4）查阅资料，不用颜色，用不同的离散点型区分簇，如方型、十字星等，并加图名和图例

任务 2：给定数据集，内含 80 个二维数据，完成聚类

首先读入 datas.txt 文件，然后数据可视化。

任务要求：

（1）查阅资料，读入 datas.txt 文件，然后把数据可视化

（2）聚类数目为 4，输出并保存聚类图形

（3）用不同的离散点型区分簇，如方型、十字星等，并加图名和图例

任务 3：实现西瓜数据集 4.0 的 k-means 聚类

任务要求：

（1）读取数据，绘制样本点

（2）完善代码，把数据集聚成三个簇

（3）聚类结果显示，用不同的离散点型区分簇，如方型、十字星等，并用 legend 指令给图加图例

（4）比较和课本的聚类结果是否相同，也可以和其他同学比较聚类结果是否相同，并分析可能的原因

（5）令迭代次数 n=5,10,50,200,观察聚类结果是否一样

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

附录 1：聚类源码

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

# 两点距离
def distance(e1, e2):
    return np.sqrt((e1[0]-e2[0])**2+(e1[1]-e2[1])**2)

# 集合中心
def means(arr):
    return np.array([np.mean([e[0] for e in arr]), np.mean([e[1] for e in arr])])

# arr 中距离 a 最远的元素，用于初始化聚类中心
def farthest(k_arr, arr):
    f = [0, 0]
    max_d = 0
    for e in arr:
        d = 0
        for i in range(k_arr.__len__()):
            d = d + np.sqrt(distance(k_arr[i], e))
        if d > max_d:
            max_d = d
            f = e
    return f

# arr 中距离 a 最近的元素，用于聚类
def closest(a, arr):
    c = arr[1]
    min_d = distance(a, arr[1])
    arr = arr[1:]
    for e in arr:
        d = distance(a, e)
        if d < min_d:
            min_d = d
            c = e
    return c


if __name__="__main__":
    ## 生成二维随机坐标，手上有数据集的朋友注意，理解 arr 改起来就很容易了
    ## arr 是一个数组，每个元素都是一个二元组，代表着一个坐标
    ## arr 形如：[ (x1, y1), (x2, y2), (x3, y3) ... ]
    arr = np.random.randint(100, size=(100, 1, 2))[:, 0, :]
```

```
    ## 初始化聚类中心和聚类容器
    m = 5
    r = np.random.randint(arr.__len__() - 1)
    k_arr = np.array([arr[r]])
    cla_arr = [[]]
    for i in range(m-1):
        k = farthest(k_arr, arr)
        k_arr = np.concatenate([k_arr, np.array([k])])
        cla_arr.append([])

    ## 迭代聚类
    n = 20
    cla_temp = cla_arr
    for i in range(n):    # 迭代 n 次
        for e in arr:    # 把集合里每一个元素聚到最近的类
            ki = 0        # 假定距离第一个中心最近
            min_d = distance(e, k_arr[ki])
            for j in range(1, k_arr.__len__()):
                if distance(e, k_arr[j]) < min_d:    # 找到更近的聚类中心
                    min_d = distance(e, k_arr[j])
                    ki = j
            cla_temp[ki].append(e)
        # 迭代更新聚类中心
        for k in range(k_arr.__len__()):
            if n - 1 == i:
                break
            k_arr[k] = means(cla_temp[k])
            cla_temp[k] = []

    ## 可视化展示
    col = ['HotPink', 'Aqua', 'Chartreuse', 'yellow', 'LightSalmon']
    for i in range(m):
        plt.scatter(k_arr[i][0], k_arr[i][1], linewidth=10, color=col[i])
        plt.scatter([e[0] for e in cla_temp[i]], [e[1] for e in cla_temp[i]],
color=col[i])
    plt.show()
```