电子科技大学

实验报告

课程名称： 机器学习

学　　院： 电子科技大学（深圳）高等研究院

专　　业：电子信息

指导教师：张栗粽

学生姓名： 刘文晨

学　　号：202222280328

电 子 科 技 大 学

实 验 报 告

实验五

一、实验项目名称

实现 K-means 聚类

二、实验学时：4 学时

三、实验目的

1. 了解聚类的基本概念
2. 掌握 K-means 聚类算法的基本原理；

四、实验原理

1. 聚类

将物理或抽象对象的集合分成由类似的对象组成的多个类的过程被称为聚类。由聚类所生成的簇是一组数据对象的集合，这些对象与同一个簇中的对象彼此相似，与其他簇中的对象相异。“物以类聚，人以群分”，在自然科学和社会科学中，存在着大量的分类问题。聚类分析又称群分析，它是研究（样品或指标）分类问题的一种统计分析方法。聚类分析起源于分类学，但是聚类不等于分类。聚类与分类的不同在于，聚类所要求划分的类是未知的。聚类分析内容非常丰富，有系统聚类法、有序样品聚类法、动态聚类法、模糊聚类法、图论聚类法、聚类预报法等。

在机器学习中，聚类是一个将数据集中在某些方面相似的数据成员进行分类组织的过程，聚类就是一种发现这种内在结构的技术，聚类技术经常被称为无监督学习。

2. K-means 聚类算法

K-means 聚类算法是一种迭代求解的聚类分析算法。算法思想是：需要随机选择 k

个对象作为初始的聚类中心，然后计算每个对象和各个聚类中心之间的距离，然后将每个对象分配给距离它最近的聚类中心。

K-means 聚类算法的一般步骤如下所示：

输入: 样本集$D = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$;聚类簇数k.
过程:
1：从D中随机选择k个样本作为初始均值向量$\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k\}$
2：repeat
3：　令$C_i = \varnothing (1 \le i \le k)$
4：　for $j = 1, 2, ..., m$ do
5：　　计算样本$\mathbf{x}_j$与各均值向量$\boldsymbol{\mu}_i (1 \le i \le k)$的距离：$d_{ji} = ||\mathbf{x} - \boldsymbol{\mu}_i||_2$;
6：　　根据距离最近的均值向量确定x_j的簇标记：$\lambda_j = \mathbf{argmin}_{i \in \{1,2,...,k\}} d_{ji}$;
7：　　将样本$\mathbf{x}_j$划入相应的簇：$C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$;
8：　end for
9：　for $i = 1, 2, ..., k$ do
10：　　计算新均值向量：$\boldsymbol{\mu}_i' = \frac{1}{|C_i|} \sum_{x \in C_i} \mathbf{x}$;
11：　　if $\boldsymbol{\mu}_i' \neq \boldsymbol{\mu}_i$ then
12：　　将当前均值向量$\boldsymbol{\mu}_i$更新为$\boldsymbol{\mu}_i'$
13：　　else
14：　　保持当前均值向量不变
15：　　end if
16：　end for
17：until 当前均值向量均未更新
输出:簇划分$\mathcal{C} = \{C_1, C_2, ..., C_k\}$

图 1 K-means 聚类算法的一般步骤

五、实验内容与要求

1. 数据预处理；
2. 实现K-means聚类算法；
3. 计算各簇的质心，打印结果；
4. 绘制图像。

六、实验器材（设备、元器件）

处理器：Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz

Python 3.9.0

matplotlib 3.4.0

numpy 1.23.4

七、实验步骤

1. 数据预处理

给定的数据样本是 data 文件，将数据转存到 K-means.txt 中。读取这个文件，将顶点坐标转为浮点数，存储在一个列表中。代码如下：

```
    # 读取数据
    f = open('data/K-means.txt', 'r')
    for line in f:
        points.append(np.array(line.split(','), dtype=np.string_).astype(np.float64))
    # print(points)
```

2. 实现 K-means 聚类算法

首先，设置簇数 k、最大迭代次数、均值向量变化量下界等参数。代码如下：

```
    # 参数设置
    # 簇数
    k = 3
    # 迭代次数
    round = 0
    # 最大迭代次数
    ROUND_LIMIT = 50
    # 均值向量变化量下界
    THRESHOLD = 1e-10
```

接着初始化 k 个簇的质心，实现 K-means 算法，距离选择为欧式距离，当均值向量变化量足够小或达到最大迭代量时，退出 K-means 算法。代码如下：

```
    # 随机取 k 个不重复的顶点作为初始质心
    mean_vectors = random.sample(points, k)
    # K-means 算法
    while True:
        # 迭代次数自增
        round += 1
        # 初始化均值向量变化量
        change = 0
        # 清空对簇的划分
        clusters = []
        for i in range(k):
            clusters.append([])

        for point in points:
            '''''
            argmin 函数找出容器中最小的下标，在这里这个目标容器是
            list(map(lambda vec: np.linalg.norm(melon - vec, ord = 2), mean_vectors)),
            它表示 melon 与 mean_vectors 中所有向量的距离列表。
```

```
            (numpy.linalg.norm 计算向量的范数,ord = 2 即欧几里得范数，或模长)
            '''
            c = np.argmin(
                list(map(lambda vec: np.linalg.norm(point - vec, ord=2), mean_vectors))
            )
            clusters[c].append(point)

        for i in range(k):
            # 求每个簇的新均值向量
            new_vector = np.zeros((1, 2))
            for point in clusters[i]:
                new_vector += point
            new_vector /= len(clusters[i])
            # 累加改变幅度并更新均值向量
            change += np.linalg.norm(mean_vectors[i] - new_vector, ord=2)
            mean_vectors[i] = new_vector
        # 当均值向量变化量足够小或达到最大迭代量时，退出 K-means 算法
        if round > ROUND_LIMIT or change < THRESHOLD:
            break
```

3. 计算各簇的质心，打印结果

簇的聚类结果保存在列表 clusters 中，其中每一个元素都是一个列表，列表的元素就是位于同一个簇中的顶点坐标。将同簇中各顶点坐标求和后取平均值即是该簇的质心坐标。打印迭代次数 round 和各个簇的质心坐标。代码如下：

```
    # 打印结果
    print('共迭代%d 轮。' % round)

    for i, cluster in zip(range(k), clusters):
        # 求各簇的质心
        centroid = []
        x_mean = 0.0
        y_mean = 0.0
        for point in cluster:
            x_mean += point[0]
            y_mean += point[1]
        x_mean /= len(cluster)
        y_mean /= len(cluster)
        centroid.append(x_mean)
        centroid.append(y_mean)
        print('簇%d 的质心为: ' % (i + 1), centroid)
```

4. 绘制图像

绘制散点图，使得不同簇的顶点的颜色不同，同一个簇的顶点的颜色相同。代码如下：

```
    # 绘图
```

```
    colors = ['red', 'green', 'blue']
    # 每个簇换一下颜色，同时迭代簇和颜色两个列表
    for i, color in zip(range(k), colors):
        for point in clusters[i]:
            # 绘制散点图
            plt.scatter(point[0], point[1], color=color)
    plt.savefig('聚类结果.jpg')
    plt.show()
```

5. 实验结果

运行程序。

当簇的个数为 3 时，共迭代 17 轮。其中：

簇 1 的质心为 [3.423076923076923, 7.897435897435898]，

簇 2 的质心为 [11.116883116883116, 16.207792207792206]，

簇 3 的质心为 [14.878787878787879, 4.833333333333333]。

散点图如图 2 所示。

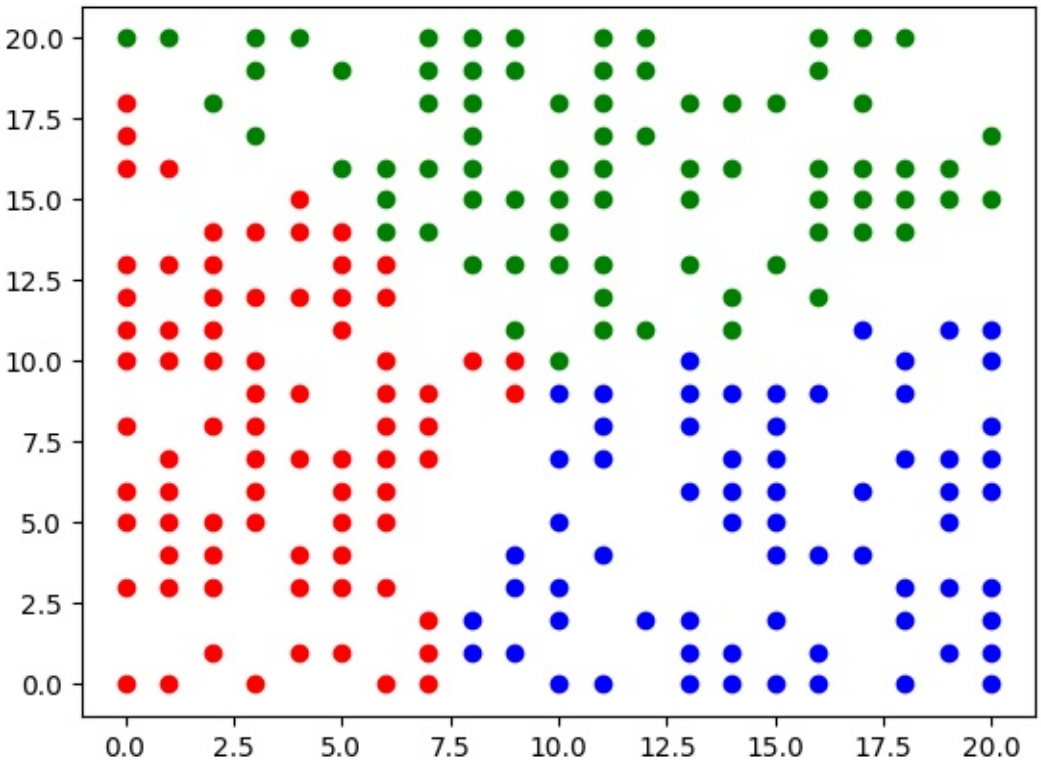

图 2 散点图（k=3）

修改簇的个数，当簇的个数为 4 时，共迭代 18 轮。其中：

簇 1 的质心为 [16.0188679245283, 4.9245283018867925]，

簇 2 的质心为 [4.344827586206897, 14.60344827586207]，

簇 3 的质心为 [4.827586206896552, 4.551724137931035]，

簇 4 的质心为 [13.923076923076923, 15.596153846153847]。

散点图如图 3 所示。

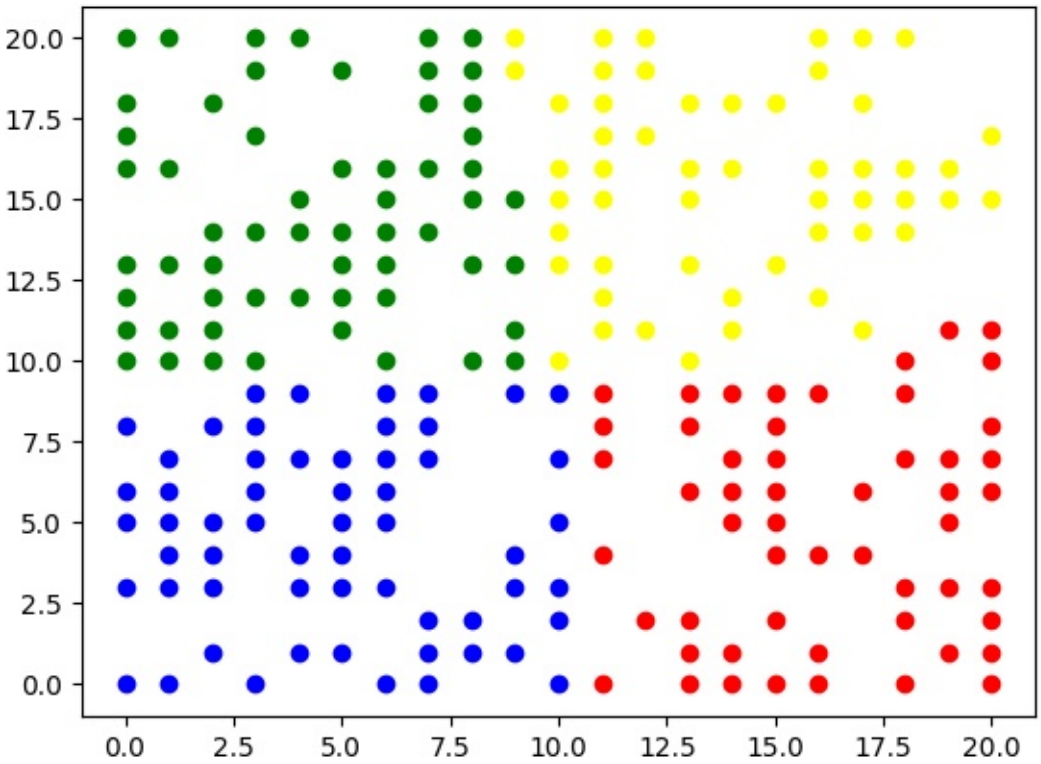

图 3 散点图（k=4）

当簇的个数为 5 时，共迭代 14 轮。其中：

簇 1 的质心为： [11.128205128205128, 17.358974358974358]，

簇 2 的质心为： [3.5106382978723403, 13.97872340425532]，

簇 3 的质心为： [15.619047619047619, 3.2857142857142856]，

簇 4 的质心为： [4.444444444444445, 4.62962962962963]，

簇 5 的质心为： [15.64102564102564, 11.820512820512821]。

散点图如图 4 所示。

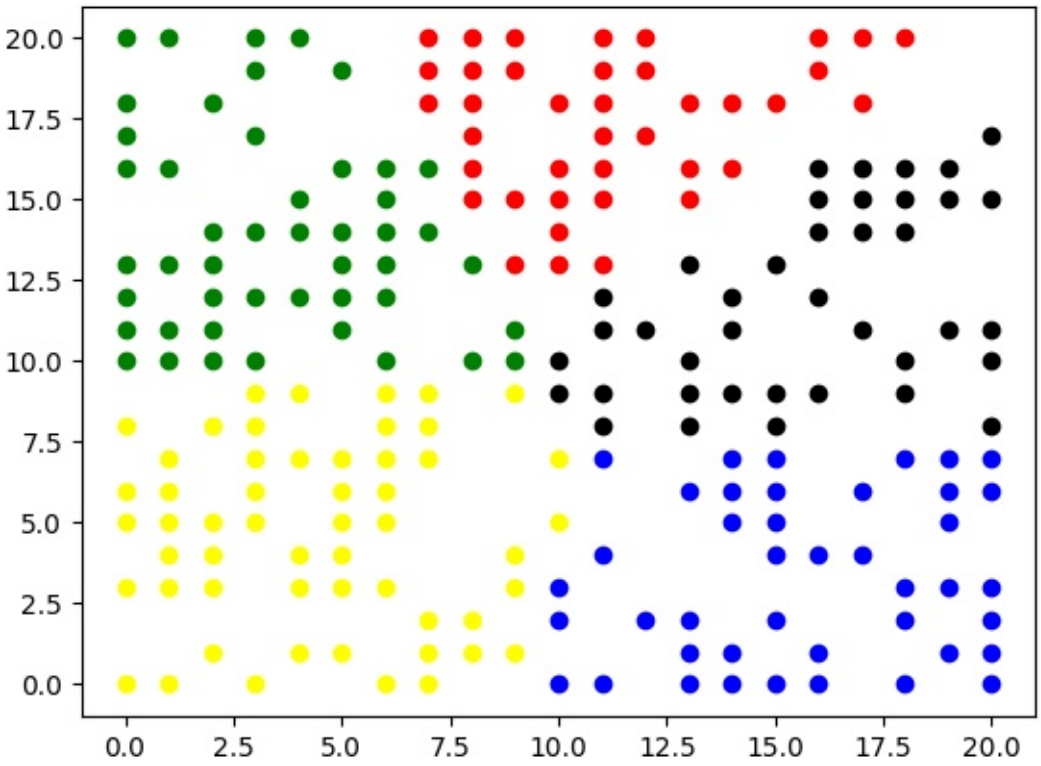

图 4 散点图（k=5）

从上述实验结果可以看出，K-means 算法能够将数据样本划分为指定个数的簇的集合。通过迭代计算，使得簇内距离尽可能小，簇间距离尽可能大。从绘制的散点图可以看出，该算法很好地达到了聚类的效果。

八、心得体会

本实验实现了 K-means 聚类，分别将簇的个数设置为 3、4、5，得到了不同的聚类结果，实验效果良好。通过此次实验，很好地掌握了 K-means 聚类算法的基本原理，熟悉了 Python 第三方库——matplotlib 和 numpy 的使用。