实验六 k 近邻

一、实验目的

1.熟悉 Python 的基础使用

2.理解和掌握 k 近邻基本原理和实现过程

3.利用 k 近邻解决实际问题

二、实验内容

1. 绘制实验所需数据集

2. 运行 k 近邻程序

3.对测试数据进行分类判别，并显示分类结果

三、实验原理：具体分析见课本（报告中要写出 KNN 原理）

任务 1：

有以下数据点：([[2, 1],[3, 2],[4, 2],[1, 3],[1.5, 4],[1.7, 3],[2.6, 5],[3.4, 3],[3, 6],[1, 7],[4, 5],[1.2, 6],[1.8, 7],[2.2, 8],[3.7, 7],[4.8, 5]])

其类别为([0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1])

设计 KNN 算法进行分类判断，测试数据为([3.2, 5.4])

任务要求：

（1）读懂程序，尤其是排序和遍历得到前 K 个数值的过程；

（2）让 k=1，3，5，7，观察结果的不同。用 subplot 指令将四个结果用子图的形式保存成 2 行 2 列，每个子图用 k 值命名；

（3）多改变几个测试数据的值，观察结果的不同。

任务 2:思考分析（先做任务 3，最后做这个）

从图中看出,K 近邻找到的貌似不全是距离测试样本最近的点，阅读程序中关于距离计算的部分，读懂程序，分析可能的原因，并给出修改建议。

任务 3：直接调用 KNN 库

任务要求：

打开原文链接 https://www.cnblogs.com/angle6-liu/p/10416736.html

（1）运行 1. 数据蓝蝴蝶和 2. 根据身高、体重、鞋子尺码，预测性别的程序。

（2）直接调用 KNN 库，完成任务 1，比较结果。

任务 1 参考程序：

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

# 初始化模拟数据， X_train 为样本点
X_train = np.array([[2, 1],[3, 2],[4, 2],[1, 3],[1.5, 4],[1.7, 3],[2.6, 5],[3.4, 3],
                    [3, 6],[1, 7],[4, 5],[1.2, 6],[1.8, 7],[2.2, 8],[3.7, 7],[4.8, 5]])
y_train = np.array([0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1])  # y_train 为样本点标记
X_test = np.array([3.2, 5.4])  # X_test 为测试样本
k = 3   # k 为邻居数

# 这里的距离公式采用欧式距离
square_ = (X_train - X_test) ** 2
square_sum = square_.sum(axis=1) ** 0.5
# 根据距离大小排序并找到 测试样本 与 所有样本 k 个最近的样本的序列
square_sum_sort = square_sum.argsort()
small_k = square_sum_sort[:k]
# K 近邻用于分类则统计 K 个邻居分别属于哪类的个数，用于回归则计算 K 个邻居的 y 的平均值作为预测结果
# 统计距离最近的 k 个样本 分别属于哪一类的个数 别返回个数最多一类的序列 作为预测结果
y_test_sum = np.bincount(np.array([y_train[i] for i in small_k])).argsort()
# 打印预测结果
print('predict: class {}'.format(y_test_sum[-1]))

# 将数据可视化 更生动形象
# 将 class0 一类的样本点 放到 X_train_0 中
X_train_0 = np.array([X_train[i, :] for i in range(len(y_train)) if y_train[i] == 0])
# 将 class1 一类的样本点 放到 X_train_1 中
X_train_1 = np.array([X_train[i, :] for i in range(len(y_train)) if y_train[i] == 1])

# 绘制所有样本点 并采用不同的颜色 分别标记 class0 以及 class1
plt.scatter(X_train_0[:,0], X_train_0[:,1], c='g', marker='o', label='train_class0')
plt.scatter(X_train_1[:,0], X_train_1[:,1], c='m', marker='o', label='train_class1')
if y_test_sum[-1] == 0:
    test_class = 'g'
elif y_test_sum[-1] == 1:
    test_class = 'm'
plt.scatter(X_test[0], X_test[1], c=test_class, marker='*', s=100, label='test_class')
# 连接 测试样本 与 k 个近邻
for i in small_k:
    plt.plot([X_test[0], X_train[i, :][0]], [X_test[1], X_train[i, :][1]], c='c')
plt.legend(loc='best')
plt.show()
```