电子科技大学

计算机科学与工程学院

实 验 指 导 书

（实验）课程名称 人工智能综合实验 I

电子科技大学教务处制表

实验二　K 近邻法实现分类任务

一、　实验目的和任务

理解并掌握 K 近邻算法，动手基于 python 实现模型构建，并根据给出的数据使用 K 近邻算法实现数据分类。

二、　实验原理

基于 python 实现 K 近邻算法的模型构建，对给出的数据进行分类，并实现在 K 不同取值时的分类结果。

2.1 算法基本原理：

1.算法原理

K-近邻（k-Nearest Neighbour，简称 KNN），常用于有监督学习，是最常用的分类算法之一。

K 近邻算法，即给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例最邻近的 K 个实例，这 K 个实例的多数属于某个类，就把该新的输入实例分类到这个类中。

算法概述：

输入：训练样本集，训练样本集对应的类别标签，测试数据

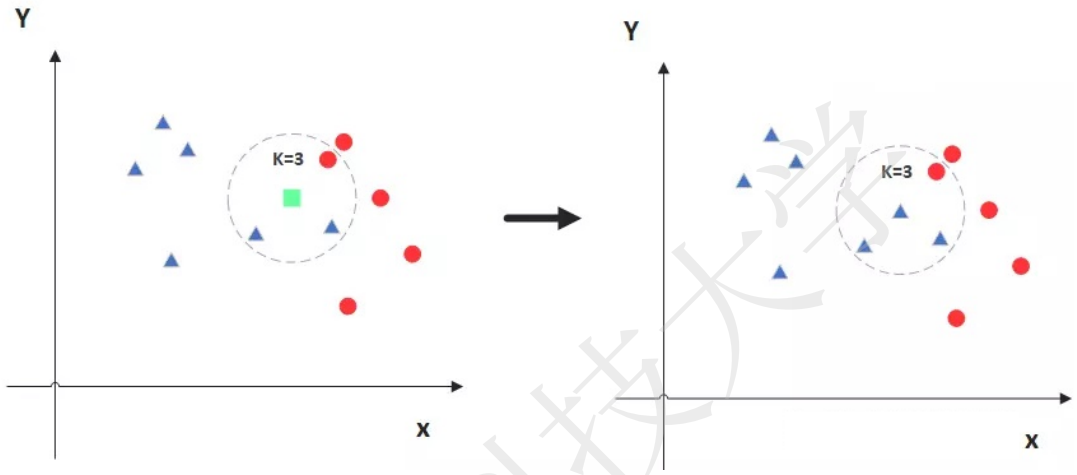
输出：测试数据对应的分类标签

步骤 1：将新数据的每个特征与样本集中的数据对应的特征进行比较

步骤 2：提取样本集中特征最相似的 K 个数据的分类标签

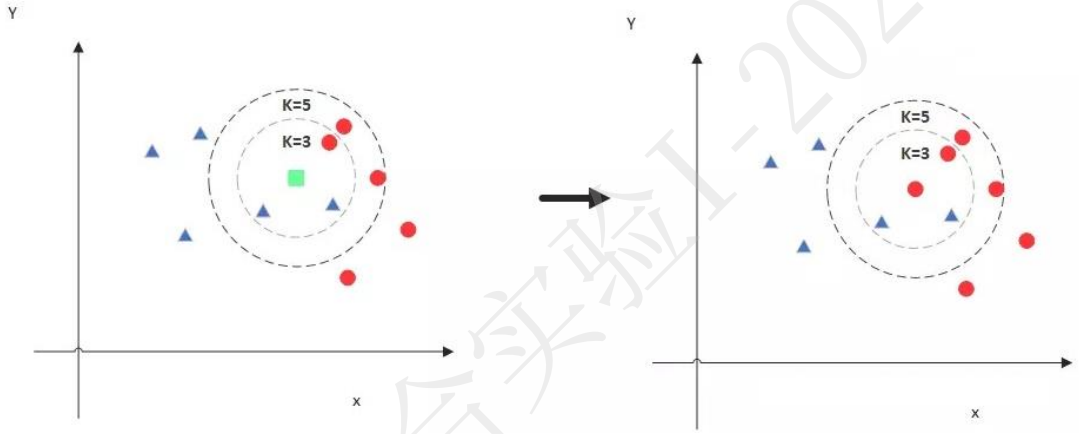
步骤 3：统计这 K 个标签中出现次数最多的类别，作为新数据的类别

2. 算法图解



如上图，假设图中绿色的点是没有标签的新数据，其余样本是有标签的数据集，现在设置 K=3，那么 KNN 算法就会找到右图中最近的三个样本（圆圈中），并且发现这三个样本中三角占多数，则将新样本归类到三角这一类别。

KNN 算法中 K 的值的选取非常重要，甚至会影响输出结果。



上图中，将 K 的值从 3 变成 5，得到的最近的 5 个样本（右图大圈中），此时红色圆圈占大多数，此时会将未知的绿色新数据分类为红色圆圈。所以大家在实现 KNN 的时候，需要设置合适的 K 值。

3. 一些常用距离

（1）欧式距离：

$$d_{ij} = \left(\sum_{k=1}^{m} |x_{ki} - x_{kj}|^2 \right)^{\frac{1}{2}}$$

（2）曼哈顿距离：

$$d_{ij} = \sum_{k=1}^{m} |x_{ki} - x_{kj}|$$

（3）LP 距离：

$$L_p(x_i, x_j) = \left(\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^p \right)^{\frac{1}{p}}$$

（4）马氏距离：

$$d = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}$$

其中 S 为协方差矩阵，下面是求协方差矩阵的方法，供参考。

```
def covariance(X, Y):
    n = np.shape(X)[0]
    X, Y = np.array(X), np.array(Y)
    meanX, meanY = np.mean(X), np.mean(Y)
    cov = sum(np.multiply(X - meanX, Y - meanY)) / (n - 1)
    return cov
def calc_covmat1(samples):

    方法 1：根据协方差公式和协方差矩阵的概念计算协方差矩阵
    S = samples   # 样本集
    na = np.shape(S)[1]   # 特征 attr 总数
    covmat1 = np.full((na, na), fill_value=0.)   # 保存协方差矩阵
    for i in range(na):
        for j in range(na):
            covmat1[i, j] = covariance(S[:, i], S[:, j])
    return covmat1

def calc_covmat2(samples):

    方法 2：先样本集中心化再求协方差矩阵
```

```
    S = samples   # 样本集
    ns = np.shape(S)[0]   # 样例总数
    mean = np.array([np.mean(attr) for attr in S.T])   # 样本集的特征均值
    print('样本集的特征均值:\n', mean)
    centrS = S - mean   ##样本集的中心化
    print('样本集的中心化(每个元素将去当前维度特征的均值):\n', centrS)
    # 求协方差矩阵
    covmat2 = np.dot(centrS.T, centrS) / (ns - 1)
    return covmat2

# samples 是数据集矩阵
```

2.2 参考 Demo

```
import numpy as np
import operator

# 数据归一化处理
def autoNorm(dataSet):
    minVals = dataSet.min(0)
    maxVals = dataSet.max(0)
    ranges = maxVals - minVals

    m = dataSet.shape[0]
    normDataSet = dataSet - np.tile(minVals, (m, 1))
    # print normDataSet
    normDataSet = normDataSet / np.tile(ranges, (m, 1))
    # print normDataSet
    return normDataSet, ranges, minVals

# 测试数据归一化
def normTest(dataSet, ranges, minVals):
    return (dataSet-minVals) / ranges

# 分类 主体部分根据实验要求自行完成
def classify(test, dataSet, labels, k):
    pass
```

三、 实验内容

实验一：编程实现 K 近邻分类器模型，在西瓜数据集 3.0α（表 4.5）上验证 K 近邻分类器。

（1）实现 K 不同取值时分类实验，分析实验结果。

（2）采用 Lp 距离、欧氏距离等不同距离计算验证 K 近邻分类结果，并分析讨论。

实验二：在 Iris 数据集上实现 KNN 分类。本实验的数据以 csv 文件的形式给出，已上传到指定文件夹。

（1）实现 K 不同取值时分类实验，分析实验结果。

（2）采用 Lp 距离、欧氏距离等不同距离计算验证 K 近邻分类结果，并分析讨论。

表 4.5 西瓜数据集 3.0α

编号	密度	含糖率	好瓜
1	0.697	0.460	是
2	0.774	0.376	是
3	0.634	0.264	是
4	0.608	0.318	是
5	0.556	0.215	是
6	0.403	0.237	是
7	0.481	0.149	是
8	0.437	0.211	是
9	0.666	0.091	否
10	0.243	0.267	否
11	0.245	0.057	否
12	0.343	0.099	否
13	0.639	0.161	否
14	0.657	0.198	否
15	0.360	0.370	否
16	0.593	0.042	否
17	0.719	0.103	否

四、 实验报告要求

根据实验要求完成实验内容，要求有实验代码，并给出每个实验的实验结果、讨论分析。

五、 实验仪器设备

机房电脑一台，编程平台为 Anaconda 下 Spyder 编辑器。