人工智能实验：实验一

—— 使用 k-近邻算法分类

3190102060　黄嘉欣

一、kNN 算法概述

k-近邻算法采用测量不同特征值之间的距离方法进行分类。若存在一个样本数据集合，即训练样本集，且样本集中每个数据都存在标签，则我们可以知道样本集中每一数据与所属分类的对应关系。当输入没有标签的新数据时，我们将新数据的每个特征与样本集中数据对应的特征进行比较，此时算法会提取样本集中特征最相似数据（最近邻）的分类标签。一般来说，我们只选择样本数据集中前 k 个最相似的数据，这就是 k-近邻算法中 k 的出处，通常 k 是不大于 20 的整数。最后，选择 k 个最相似数据中出现次数最多的分类，即为新数据的分类。

二、kNN 算法一般步骤

① 定义样本数据集合函数（训练样本集），输出样本数据集特征及其标签；

② 定义 k-近邻算法函数，输入为需要测试的数据特征，样本数据集特征及其标签、k 值。通过计算比较测试的数据与样本数据特征的距离，找到 k 个最相邻的数据及其标签，k 个数据中标签次数最多的标签作为输出测试数据的分类结果。

三、AB 分类

① 实验题目

以矩阵形式建立训练样本集，矩阵的每一行为一组样本，每组样本配以对应的标签。编写 classify0 函数（k=3），测试输入样本集[0,0]、[0.8,0.7]等点的类别。

② 实验结果

设训练样本集及其标签如下：

```
group = array([[1.0,1.1],[1.0,1.0],[0,0],[0,0.1]])
labels = ['A','A','B','B']
```

当输入样本集为[0,0]、[0.8,0.7]时，程序输出为：

```
The class of [0,0] is B, and the class of [0.8,0.7] is A.
```

③ 实验结果分析

分别计算[0,0]与训练样本集中四个样本的欧式距离为：1.487、1.414、0、0.1，与其

最近的 3 个样本为：[0,0]、[0,0.1]、[1.0,1.0]，其中前两个样本的标签均为 B，故取[0,0]的类别为 B。同理，[0.8,0.7]与[1.0,1.0]、[1.0,1.1]、[0,0.1]相距最近，其中前两个样本的标签均为 A，故取[0.8,0.7]的类别为 A，算法输出正确。

四、约会网站的配对效果分类

① 实验题目

通过收集的一些约会网站的数据信息，将匹配对象归到不喜欢的人、魅力一般的人、极具魅力的人三类中，其类标签分别为 1、2、3。数据信息保存在文件 datingTestSet2.txt 中，编写程序，从文件中读取训练样本矩阵和类标签向量，在对数据进行归一化处理后利用 classify0 函数（k=3）系统性地实现 datingTestSet2.txt 中 10%数据的测试，并打印出结果。

② 实验结果及分析

将 datingTestSet2.txt 中前 10%的数据作为测试样本集，后 90%的数据作为训练样本集，得到各测试样本的预测分类如下（数据过多，可运行程序后查看输出，此处略去部分）：

```
The result that kNN predicted for the 0 th person is 3, and the right result is 3.
The result that kNN predicted for the 1 th person is 2, and the right result is 2.
The result that kNN predicted for the 2 th person is 1, and the right result is 1.
The result that kNN predicted for the 3 th person is 1, and the right result is 1.
The result that kNN predicted for the 4 th person is 1, and the right result is 1.
The result that kNN predicted for the 5 th person is 1, and the right result is 1.
The result that kNN predicted for the 6 th person is 3, and the right result is 3.
The result that kNN predicted for the 7 th person is 3, and the right result is 3.
The result that kNN predicted for the 8 th person is 1, and the right result is 1.
The result that kNN predicted for the 9 th person is 3, and the right result is 3.
```

总的预测错误数为：

```
The number of errors is 5.
```

预测正确率为：

```
The correct rate is 0.950000.
```

五、手写数字识别

① 实验题目

需要识别的数字已被处理成 32×32 的二进制图像矩阵，保存在文件夹中。创建函数，将图像矩阵转换为单个样本向量，其维度为 1×1024。编写程序，从 trainingDigits 目录中

各文件的文件名中解析出样本向量的类标签，联合样本向量作为训练样本集，再将 testDigits 目录中的文件内容存储在列表中，解析出数据及标签，利用前述 k-近邻算法 classify0 函数（k=3）系统性地实现 testDigits 目录中数据的测试，并打印出结果。

② 实验结果及分析

由于暂未提供 testDigits 目录，此处只完成了 trainingDigits 目录中各文件的数据解析，得到的训练样本矩阵如下（数据较长，显示时存在省略）：

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

每一行样本对应的类标签如下（可运行程序后查看输出，此处省略部分）：

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
```

经比对，数据及标签与 trainingDigits 目录中各文件数据完全吻合。

六、附录：实验 Python 代码（可见 src 目录）

① test.py

```python
from numpy import *
from os import *
import kNN

# 1-1
group,labels = kNN.createDataSet()
inData1 = [0,0]
inData2 = [0.8,0.7]
class1 = kNN.classify0(inData1,group,labels,3)
class2 = kNN.classify0(inData2,group,labels,3)
print("The class of [0,0] is "+class1+", and the class of [0.8,0.7] is "+class2+".\n")

# 1-2
group,labels = kNN.getDataSet("../datingTestSet2.txt")
normalGroup = kNN.normalize(group)
corRate = kNN.dateTest(normalGroup,labels)
print("The correct rate is %f.\n" % (corRate))

# 1-3
labels = kNN.getLabels("../trainingDigits")
fileList = listdir("../trainingDigits")             # 目录下所有的文件
```

```
listLen = len(fileList)                                  # 文件数
group = zeros((listLen,1024))                            # 初始化数字矩阵
for i in range(0,listLen):
    group[i,:] = kNN.getVector("../trainingDigits/" + fileList[i])       # 获取该文件的数字向量
# print(group)
# print(labels)
```

② kNN.py

```
from numpy import *
from os import listdir

def createDataSet():
    group = array([[1.0,1.1],[1.0,1.0],[0,0],[0,0.1]])
    labels = ['A','A','B','B']
    return group, labels

def classify0(inData,group,labels,k):
    """分类器
    Args:
      inData: 输入的测试样本
      group: 训练样本集
      labels: 训练样本集对应的标签
      k: 近邻数
    Returns:
      sortedClassCnt[0][0]: 测试样本的分类
    """
    groupLine = group.shape[0]              # 训练样本集大小
    inDataMat = tile(inData,(groupLine,1))  # 将输入样本转为groupSize*1的矩阵
    diffMat = inDataMat-group               # 输入样本与各个训练样本的差:[(x_1-x_2),(y_1-y_2)]
    sqDiffMat = diffMat**2                  # 平方:[(x_1-x_2)^2,(y_1-y_2)^2]
    sumSqVec = sqDiffMat.sum(axis=1)        # 按行相加:(x_1-x_2)^2+(y_1-y_2)^2
    distanceVec = sumSqVec**0.5             # 开方:sqrt((x_1-x_2)^2+(y_1-y_2)^2)
    sortedDist = distanceVec.argsort()      # 从小到大排序(下标)
    classCnt = {}                           # 字典数据类型
    for i in range(0,k):
        label = labels[sortedDist[i]]               # 距离从小到大排在第i位的训练样本的label
        classCnt[label] = classCnt.get(label,0)+1   # 使用get()读取键对应的值，若没有该键，则值初始化为0
    sortedClassCnt = sorted(classCnt.items(), key=lambda x: x[1], reverse=True) # 对字典按value排序
    # sortedClassCnt:[('B', 2), ('A', 1)]
    # print(sortedClassCnt[0][0]):B
    return sortedClassCnt[0][0]

def getDataSet(fileName):
    """从文件中读取训练样本矩阵和类标签向量
    Args:
      fileName: 文件路径
    Returns:
      group: 训练样本矩阵
      labels: 类标签向量
    """
    with open(fileName,"r") as f:
        flist = f.readlines()               # 以list的形式返回
        flines = len(flist)                 # 读取文件的行数
        group = zeros((flines,3))           # flines*3的矩阵
        labels = []                         # flines*1的向量
        # print(flist[0].strip())
        for i in range(0,flines):
            line = flist[i].strip()                 # 读取每一行的内容(去除头尾空白符)
            lineData = line.split("\t")             # 按tab分割每行内容
            group[i,:] = lineData[0:3]              # 训练样本矩阵的第i行
            labels.append(int(lineData[3]))         # 类标签向量的第i个
    return group,labels

def normalize(group):
    """归一化
    Args:
      group: 需要归一化的样本集
```

```
    Returns:
      normalGroup: 归一化后的样本集
    """
    maxData = group.max(axis=0)             # 每列的最大值
    minData = group.min(axis=0)             # 每列的最小值
    groupLine = group.shape[0]              # 矩阵的行数
    normalGroup = zeros((groupLine,3))      # groupLine*3的矩阵
    Xmin = tile(minData,(groupLine,1))      # Xmin矩阵
    Xmax = tile(maxData,(groupLine,1))      # Xmax矩阵
    normalGroup = (group-Xmin)/(Xmax-Xmin)  # 线性归一化
    return normalGroup

def dateTest(normalGroup,labels):
    """测试配对效果
    Args:
      normalGroup: 归一化后的样本集
      labels: 样本集对应的类标签向量
    Returns:
      1-errorCnt/testLen: 预测正确率
    """
    ratio = 0.1                             # 测试前10%的数据
    groupLen = normalGroup.shape[0]         # 输入的训练样本集行数
    testLen = int(groupLen*ratio)           # 测试样本集行数
    inData = zeros((testLen,3))             # 输入的测试样本集
    errorCnt = 0                            # 预测错误数
    for i in range(0,testLen):
        inData[i,:] = normalGroup[i,:]      # 测试样本矩阵的第i行

        # 前10%用作测试，后90%作为训练样本
        testLabel = classify0(inData[i,:],normalGroup[testLen:groupLen,:],labels[testLen:groupLen],3)
        print("The result that kNN predicted for the %d th person is %d, and the right result is %d." \
            % (i, testLabel, labels[i]))
        if (testLabel != labels[i]):
            errorCnt = errorCnt+1
    print("The number of errors is %d." % (errorCnt))
    return 1-errorCnt/testLen

def getVector(fileName):
    """将32*32的二进制图像矩阵转为1*1024的向量
    Args:
      fileName: 图像文件路径
    Returns:
      vec: 提取出的向量
    """
    with open(fileName,"r") as f:
        flist = f.readlines()               # 以list的形式返回
        flines = len(flist)                 # 读取文件的行数
        vec = zeros((1,1024))               # 1*1024的向量
        for i in range(0,flines):
            line = flist[i].strip()         # 读取每一行的内容(去除头尾空白符)
            for j in range(0,32):

                vec[0,32*i+j] = line[j]     # 将数字存入向量中
    return vec

def getLabels(path):
    """从文件名中获取图像矩阵对应的类标签
    Args:
      path: 文件夹路径
    Returns:
      labels: 类标签向量
    """
    labels = []                             # 保存标签
    fileList = listdir(path)                # 目录下所有的文件
    listLen = len(fileList)                 # 文件数
    for i in range(0,listLen):
        fileName = fileList[i]                      # 文件名
        prefix = fileName.split(".")[0]             # 文件名前缀
        labels.append(int(prefix.split("_")[0]))  # 标签
    return labels
```