**实验三 k近邻分类算法**

k近邻算法，k Nearest Neighbor(KNN)，它的工作原理如下：

存在一个样本数据集合，也称作训练样本集，并且样本集中每个数据都存在标签，即我们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后，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，然后算法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，作为新数据的分类。

**实验要求：**

①编写一个简单的程序，实现k近邻算法；**(重点)**

②利用k近邻算法实现手写数字的识别。

**实验步骤：**

创建一个名为kNN.py的文件，在文件的开头位置加入如下语句：

*#-\*- coding:utf-8 -\*-*

*from numpy import \**

*import os*

本文档使用的代码均放在这个文件中。

1)利用简单数据集实现k近邻算法：

①导入数据。这里我们使用一个很简单的数据集，该数据集包含四个点，分为两类。

*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*

**②编写k近邻算法函数**。根据提示完成下面的代码。要求理解这个函数的每一行代码。

*def kNNClassify(newInput, dataSet, labels, k):*

##Step1: 计算新数据到训练样本集中各个数据的欧式距离。

# 这里可以使用**tile(A, reps)**函数，它可以把A重复reps次；

# 当reps为(m,n)时，表示垂直方向上重复m次，水平方向上重复n次；

# 当reps只有一个数时，表示的是水平方向重复的次数；

# python里直接用**－**表示两个矩阵的对应元素相减;

# **x的n次方**用**x\*\*n**表示;

# 用**sum(A, axis=0)**函数进行矩阵的按列求和或者按行求和，当axis=0

# 时表示按列求和，axis=1时表示按行求和。

**#【代码待补全】**

##Step2: 对得到的距离排序

# 可使用**argsort(A)**函数，该函数的返回值为按照升序排序的下标。

# 使用变量sortedDistIndices保存排序结果。

**# 【代码待补全】**

# 创建一个字典来统计前k个最小距离中各个类别出现的次数

*classCount={}*

*for i in range(k):*

##Step3: 获取前k个最小距离的分类标签

*voteLabel = labels[sortedDistIndices[i]]*

##Step4: 统计该分类标签在这k个最相似数据中出现的次数

# 当键值voteLabel不存在时，get()函数返回0

*classCount[voteLabel] = classCount.get(voteLabel,0)+1*

##Step5:选择k个最相似数据中出现次数最多的分类，把分类结果保存

# 在predictedClass中并返回

# 可用classCount.items()获得该字典的内容

**# 【代码待补全】**

*return predictedClass*

可在python命令行中使用如下代码测试上述函数：

*import kNN*

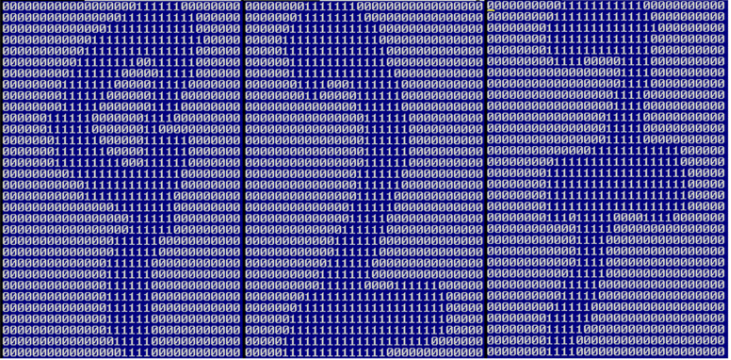
*group,labels = kNN.createDataSet()*

*kNN.kNNClassify([0,0], group, labels, 3)*

大家可以使用任意一个点来代替上面第三行代码中的[0,0]来查看分类结果。

2)使用k近邻算法识别手写数字。这里只要求给出分类的错误率。

这里需要用到的数据集中已经经过一些处理，每一个数字都被处理成32×32的黑白图像并转成了txt文件。下图是一些例子图片：



数据集包含在提供的两个文件夹中。其中trainingDigits中包含了大约2000个例子，每个数字大约有200个样本，每个txt文件的命名格式为“该文件代表的数字\_该数字的第几个样本”，如0\_25.txt就代表数字0的第25个样本，注意每一类数字的样本编号是从0开始的；testDigits中包含了大约900个测试数据。*由于机房的电脑硬件上可能会有些落后，大家可以斟酌在训练集中对每一个数字的样本数稍微减少一些，以免运行代码时间过长。*

**最好把两个数据集放到项目文件夹下。**要求使用trainingDigits中的数据训练分类器，使用testDigits中的数据测试分类器。

为了使用前面写好的分类器，我们需要将训练数据，32×32的二进制图像矩阵转换为1×1024的向量（1024=32×32）。

①编写函数将图像转换为向量。函数的参数应为文件名，打开该文件后循环读出文件的前32行，并将每行的头32个字符值存储在Numpy数组中返回。按照提示补全代码。

*def img2vector(filename):*

# 初始化返回向量returnVect

**#【代码待补全】**

# 打开文件

*fileIn = open(filename)*

*for i in range(32):*

# 读取整行数据

*lineStr = fileIn.readline()*

*for j in range(32):*

# 将头32个字符保存在返回变量中

# 注意使用int()函数将数据保存成数值

**#【代码待补全】**

# 关闭文件

*fileIn.close()*

*return returnVect*

②编写函数，读取训练集和测试集。按照提示补全代码。

*def loadDataSet():*

##Step1: 获取训练集

*print("---Getting training set...")*

# 根据实际情况设置数据集路径

*dataSetDir = "path to your dataset"*

*trainingFileList = os.listdir(dataSetDir + "trainingDigits")*

*numSamples = len(trainingFileList)*

*train\_x = zeros((numSamples,1024))*

*train\_y = []* #保存标签数据，即分类结果

*for i in range(numSamples):*

*filename = trainingFileList[i]*

*train\_x[i, :] =\*

*img2vector(dataSetDir+"trainingDigits/%s" % filename)*

*label = int(filename.split('\_')[0])*

*train\_y.append(label)*

##Step2: 获取测试集

# 仿照训练集的获取代码，编写获取测试集的逻辑

# 要求使用test\_x保存数据，test\_y保存类别

**#【代码待补全】**

*return train\_x, train\_y, test\_x, test\_y*

③编写测试函数。

def testHandWritingClass():

# 获取数据集

print("Step 1: Load data...")

# 要求使用train\_x, train\_y, test\_x, test\_y作为变量

**#【代码待补全】**

# 由于knn是不需要训练步骤的，所以这里直接使用pass跳过

print("Step 2: Training...")

pass

print("Step 3: Testing...")

numTestSamples = test\_x.shape[0]

matchCount = 0 #用以统计分类正确的数目

for i in range(numTestSamples):

# 对测试集中的数据进行分类，取k=3，将得到的结果与标签对

# 比，如果相等则分类正确的数目加一

**#【代码待补全】**

# 所有测试集数据都跑完后计算分类的正确率，保存到acuuracy变量

**#【代码待补全】**

# 显示结果

print("Step 4: Show the result...")

print("The classify accuracy is %.2f%%" % (accuracy \* 100))

可在python命令行中输入以下命令查看结果：  
 *import kNN*

*kNN.testHandWritingClass()*