Lab6　kNN 应用实践

实验目的：利用 python 实现 kNN 分类器。

实验简介：导入数据，归一化数据，距离计算，实现 kNN 分类器；实例学习 kNN 分类器如何改进约会网站，以及识别手写数字。

1. 实现 kNN 分类器

导入两个模块：科学计算包 numpy 和运算符模块 operator。在构建完整的 kNN 分类器之前，需要编写一些基本的通用函数。

```
# 使用 createDataSet()函数创建一个简单数据集合和标签，此函数包含在 knn1 模块中：
>>> import knn1
# 测试函数功能：创建变量 group 和 labels
>>> group, labels = knn1.createDataSet()
# 查看变量 group 和 labels 的值：
>>> group
>>> labels
# 通过函数 classify()实现 kNN 分类器
# 测试分类器功能：
>>> knn1.classify([0,0], group, labels, 3)
```

输出结果是 B，可以改变输入[0,0]内其他值，测试运行结果。

2. 使用 kNN 改进约会网站的配对效果

利用收集的在线约会网站的约会数据，将约会网站推荐的匹配对象归入恰当的分类（不喜欢的人，魅力一般的人，极具魅力的人）。

（1）准备数据

收集的数据存放在文本文件 datingTestSet.txt 中，每条数据占一行，总共 1000 行。主要包括 3 个特征：每年获得的飞行里程数，玩游戏视频所耗时间百分比，每周消费的冰激凌公升数。在特征数据输入分类器之前，需要将待处理数据的格式转换为分类器可以接受的格式。

file2matrix 函数解决格式输入问题，函数的输入为文件名字符串，输出为训练样本矩阵和类标签向量，该函数包含在 knn2 模块中。

```
>>> import knn2
>>> datingDataMat, datingLabels = knn2.file2matrix('datingTestSet2.txt')
>>> print(datingDataMat)
>>> datingLabels[0:20]
# 使用 Matplotlib 创建散点图
>>>import matplotlib
>>>import matplotlib.pyplot as plt
>>>from numpy import *
>>>plt.scatter(datingDataMat[:,1], datingDataMat[:,2])
>>>plt.show()
```

>>> plt.scatter(datingDataMat[:,1], datingDataMat[:,2], 15.0*array(datingLabels), 15.0*array(datingLabels))

 # 数据归一化处理

在处理不同取值范围的特征值时，通常采用的方法是将数值归一化，将取值范围处理为 0 到 1 或-1 到 1 之间。如下公式可将任意取值范围的特征值转化为 0 到 1 区间内的值：

newValue = (oldValue-min)/(max-min)，其中 max 和 min 分别是数据集中的相应维度的最大特征值和最小特征值。

函数 autoNorm()将数字特征值转化为 0 到 1 的区间。

>>> normMat, ranges, minVals = knn2.autoNorm(datingDataMat)

>>> normMat

>>> ranges

>>> minVals

（2）测试分类器

利用函数 datingClassTest()测试分类器效果：

>>> knn2.datingClassTest()

（3）使用算法：构建完整可用系统

给用户提供程序，通过该程序用户会在约会网站上找到某个人并输出它的信息。程序会给出用户对对方喜欢程度的预测值。

函数 classifyPerson()完成此功能：

>>> knn2.classifyPerson()

3. 使用 kNN 识别手写体

实验所用到的实际图像存储在两个子目录中：目录 trainingDigits 中包含了大约 2000 个例子，命名规则如 9_45.txt，表示该文件的分类是 9，是数字 9 的第 45 个实例，每个数字大概有 200 个实例。目录 testDigits 中包含了大约 900 个测试例子。将使用目录 trainingDigits 中的数据训练分类器，使用目录 testDigits 中的数据测试分类器的效果，两组数据没有重叠。

（1）准备数据

使用 kNN 分类器，首先要将图像处理为一个向量。实验中，将把一个 32*32 的二进制图像矩阵转换成 1*1024 的向量，为此首先要编写函数 img2vector，将图像转换为向量，该函数创建 1*1024 的 Numpy 数组，然后打开给定的文件，循环读出文件的前 32 行，并将每行的前 32 个字符值存储在 Numpy 数组中，最后返回数组。

（2）构建训练数据集

函数 trainingDataTest 利用目录 trainingDigits 中的文本数据构建训练集向量，以及对应的分类标签向量（标签向量可理解为对应的文件中数字的正确分类）。由于文件名的规律命名，可编写函数 classnumCut 以实现从文件名中解析分类数字，提供分类标签。注意在程序开头写上 from os import listdir 以导入 listdir 函数，它可以列出给定目录的文件名。

（3）测试算法

通过测试 testDigits 目录下的样本，计算准确率。

handwrtingTest() 函数来实现分类器测试。每个数据文件中的数字按顺序展开成一个 1024 维的向量，而向量之间的距离用欧式距离。

切换至文件 knn3.py 所在目录，并在 cmd 窗口执行：

> python knn3.py

实际运行代码时，会发现 kNN 算法分类器的执行效率并不高，因为算法需要为每个测试向量计算约 2000 次欧氏距离，每个距离计算包括 1024 个维度浮点运算，全部样本要执行 900 多次，可见算法实际耗时长。另外，kNN 算法必须保存全部数据集，每次需为测试向量准备 2MB 的存储空间（2 个 1024x1024 矩阵的空间）。

4. 操作练习

（1）对于 knn2，测试不同 k 值对错误率的影响。

（2）对于 knn2，使用曼哈顿距离，观察对错误率的影响。

（3）对于 knn2，随机选取训练样本，测试不同样本数目对错误率的影响。

（4）将 knn1, knn2, knn3 中的语句“from numpy import * ”用语句“import numpy as np”代替，修改其中对应的代码，使其能够正常执行。

（5）利用 sklearn 实现使用 kNN 改进约会网站的配对效果和使用 kNN 识别手写体应用。并分别比较应用 PCA 前后的效果。

（6）尝试利用 Tensorflow 实现 knn1, knn2, knn3。（扩展）

WBQ-2019-09-10