山东大学 软件 学院

模式识别技术 课程实验报告

|  |  |  |
| --- | --- | --- |
| 学号：201500301125 | 姓名： 马嫣然 | 班级： 15级1班 |
| 实验题目：KNN实验——手写数字识别 | | |
| 实验目的：  **实验内容：**   1. 实现最基本的KNN算法，使用trainingDigits文件夹下的数据，对testDigits中的数据进行预测。（K赋值为1，使用欧氏距离，多数投票决定分类结果） 2. 改变K的值，并观察对正确率的影响。 3. 更改距离度量方式，更改投票方式（距离加权），分析错误率。   **实验要求：**   1. 要求给出代码，以及运行窗口截图。 2. K对正确率的影响，最好用表格或作图说明，并做简要分析。 3. 实验内容3为选做，不做统一要求。 | | |
| 实验步骤与内容：   1. 代码来源   代码是在参考开源代码的基础上做出的改进  KNN分类算法实现手写数字识别 CSDN博客  <http://www.cnblogs.com/ahu-lichang/p/7152539.html>   1. 算法设计说明   1.实验环境：  硬件环境 个人笔记本电脑  软件环境 Python Eclipse Pydev插件  2.所用语言：Python   1. 实验数据分析：   把来自UCI数据库的手写数据集简化成32像素x32像素的黑白图像，并且以01矩阵的方式存储在txt文件中。大约有训练样本2000个，测试样本900个。  digits 目录下有两个文件夹，分别是:  trainingDigits：训练数据，1934个文件，每个数字大约200个文件。  testDigits：测试数据，946个文件，每个数字大约100个文件。  每个文件中存储一个手写的数字，文件的命名类似0\_7.txt，第一个数字0表示文件中的手写数字是0，后面的7是个序号。  我们使用目录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. 算法设计：   **（1）思路**  a.KNN的主要思想是找到与待测样本最接近的k个样本，然后把这k个样本出现次数最多的类别作为待测样本的类别。  b.下载实验要求中给的数据集digits文件夹，用trainingDigits作训练集，用testDigits作为测试集  c.将每个样本的txt文件转换为对应的一个向量  d.以欧氏距离作为度量进行KNN算法，分析样本之间的相似度   1. **具体实现** 2. **加载数据loadDataSet():**   从digits文件夹中读取训练数据和测试数据  以训练数据为例  dataSetDir = *'./digits/'*  trainingFileList = os.listdir(dataSetDir + *'trainingDigits'*)  加载同目录下/digits/trainingDigits中的训练数据文件  numSamples = len(trainingFileList)  train\_x = zeros((numSamples, 1024))  train\_y = []  train\_x用来储存txt文件转换成的向量  train\_y用来储存该文件实际代表的数字  for i in xrange(numSamples):  filename = trainingFileList[i]  对每个训练数据文件进行处理  # get train\_x  train\_x[i, :] = img2vector(dataSetDir + *'trainingDigits/%s'* % filename)  调用img2vector方法将txt文件转换为对应的一个向量  # get label from file name such as "1\_18.txt"  label = int(filename.split(*'\_'*)[0]) # return 1  train\_y.append(label)  记录该文件实际代表的数字  加载测试集数据的过程同理，用test\_x,test\_y表示  **b.将图片转换为向量img2vector(filename):**  def **img2vector**(filename):  rows = 32  cols = 32  数据的行列都是32  imgVector = zeros((1, rows \* cols))  fileIn = open(filename)  对每个训练数据文件进行处理  for row in xrange(rows):  lineStr = fileIn.readline()  for col in xrange(cols):  imgVector[0, row \* 32 + col] = int(lineStr[col])  return imgVector  转换为向量  **c.KNN分类核心方法 kNNClassify(newInput, dataSet, labels, k):**  newInput为待测试数据，dataSet是训练集，labels是分类合集  def **kNNClassify**(newInput, dataSet, labels, k):  numSamples = dataSet.shape[0] # shape[0]代表行数  已知分类的数据集（训练集）的行数  先tile函数将输入点拓展成与训练集相同维数的矩阵，再计算欧氏距离  # # step 1: 计算欧式距离  # tile(A, reps): 将A重复reps次来构造一个矩阵  # the following copy numSamples rows for dataSet  diff = tile(newInput, (numSamples, 1)) - dataSet # Subtract element-wise  样本与训练集的差值矩阵  squaredDiff = diff \*\* 2 # squared for the subtract  差值矩阵平方  squaredDist = sum(squaredDiff, axis = 1) # sum is performed by row  计算每一行上元素的和  distance = squaredDist \*\* 0.5  开方得到欧拉距离矩阵  # # step 2: 对距离排序  # argsort()返回排序后的索引  sortedDistIndices = argsort(distance)  按distances中元素进行升序排序后得到的对应下标的列表  classCount = {} # 定义一个空的字典  for i in xrange(k):  # # step 3: 选择k个最小距离  voteLabel = labels[sortedDistIndices[i]]  # # step 4: 计算类别的出现次数  # when the key voteLabel is not in dictionary classCount, get()  # will return 0  classCount[voteLabel] = classCount.get(voteLabel, 0) + 1  选择距离最小的k个点  # # step 5: 返回出现次数最多的类别作为分类结果  maxCount = 0  for key, value in classCount.items():  if value > maxCount:  maxCount = value  maxIndex = key  return maxIndex  返回出现次数最多的类别作为分类结果  **d.计算预测准确率**  # # step 3: 测试  print *"step 3: testing..."*  numTestSamples = test\_x.shape[0]  matchCount = 0  声明总待测数据和预测正确数  for i in xrange(numTestSamples):  predict = kNNClassify(test\_x[i], train\_x, train\_y, 1)  if predict == test\_y[i]:  matchCount += 1  对每一个测试样本进行测试，正确则计入  accuracy = float(matchCount) / numTestSamples  计算正确率  **（3）改进**  a.由于使用了开源代码，所以需要对程序的输出方法进行修改。  原程序运行结果仅输出几个步骤提示，界面停滞几分钟后便直接输出准确率的结果，未免有些不到位，所以应该加入更加细致的算法运行结果的显示。  需要添加：  对每个测试样本进行KNN分类算法后的结果  与该样本实际类别的比较  print *"分类结果为: %d, 实际类别为: %d"* % (predict, test\_y)  总测试数量  print *"总测试样本数: %d"* % numTestSamples  测试准确数  print *"测试正确样本数: %d"* % matchCount  分类准确率  print *'分类正确率: %.2f%%'* % (accuracy \* 100)   1. 实验结果：     要求：改变K的值，并观察对正确率的影响。   |  |  | | --- | --- | | K值 | 正确率 | | 1 | 98.63% | | 2 | 98.63% | | 3 | 98.84% | | 4 | 98.52% | | 5 | 98.20% |      1. 实验结果分析：   根据上方的图表，可以看出K的取值可以影响分类的准确率  K=1时准确率就已经高达98.63%；K=2时准确率保持不变，仍是98.63%；在K=3的时候准确率最高，上升到了98.84%；在K=4时又开始降低到98.52%；K=5时准确率仍然在降低，为98.20%  之后增加K的值，如K=6，K=7，可以看到准确率继续降低  结论是，为了保证较高的准确率，应该把K的值设得小一些。  3）实验结果截图  K=1  2  K=2  结果同K=1  K=3  1  K=4  3  K=5  4 | | |