

**模式识别大作业**

题 目 手写数字识别

学 院 信息科学与工程学院

专 业 控制科学与工程

组 员 徐慧

指导教师 赵海涛老师

**完成日期： 2019 年 12 月04日**

**模式识别作业报告——手写数字识别**

组员：徐慧

通过半个学期的模式识别课程的学习，我对机器学习方面的知识有了初步的了解，但这只是在理论方面。想要更好的掌握学到的知识，就要把理论与实践相结合，自己动手，去做题目，加强对模式识别课程内容的理解和掌握。经过思考，我在平台上选择了“手写数字识别”这一题，以下就是我的详细做题过程。

# 一 手写数字识别项目及其数据集介绍

手写数字识别是根据已经有的训练数据集，来判断测试数据集中的手写的数字。手写数字数据集是数据集，是来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST)。训练集和测试集均由不同人手写的数字构成，其中50%是高中学生，50%来自人口普查局的工作人员。数据集中的每张图片由28 x 28个像素点构成，每个像素点用一个灰度值表示。

本次题目所用的数据集是来自平台，包括训练数据集和测试数据集，两个数据集中都包含从零到九的手绘数字的灰度图像，每个图像的高度和宽度均为28像素，每个像素都有一个与之关联的像素值，是在0到255之间的整数，用来表示该像素的明暗程度，数字越高表示像素越暗。

训练数据集有42001行785列，第一行是列标签行，是各列的标签，其余的42000行是手写数据样本，每一行代表一个手写数字；第一列是手写数字的“标签”，其余784列是每个数据样本的像素值。测试数据集有28000行784列，和训练数据集相似，比训练数据集少了“标签”列，数据样本个数比训练集少。

# 二 整体解决方案

## 2.1 方案分析

根据在模式识别课程上所学的知识，解决手写数字识别问题的方法有多种，例如神经网络、、(K近邻算法)等。经过比较，我认为方法通俗易懂，编程上也比其他两个简单，而且判别准确率也不低，所以选择了方法。因为语言语法结构清晰，编程简单方便，函数库丰富，使用广泛，所以使用来编写程序。

## 2.2 数据读入

项目所用的数据均是格式。把数据下载好后放到项目文件夹里，这样在读取数据的时候不容易出错。使用的来编写程序。对于格式的数据，可以使用工具包中的命令导入数据，数据格式为，训练数据集导入结果如下图所示：

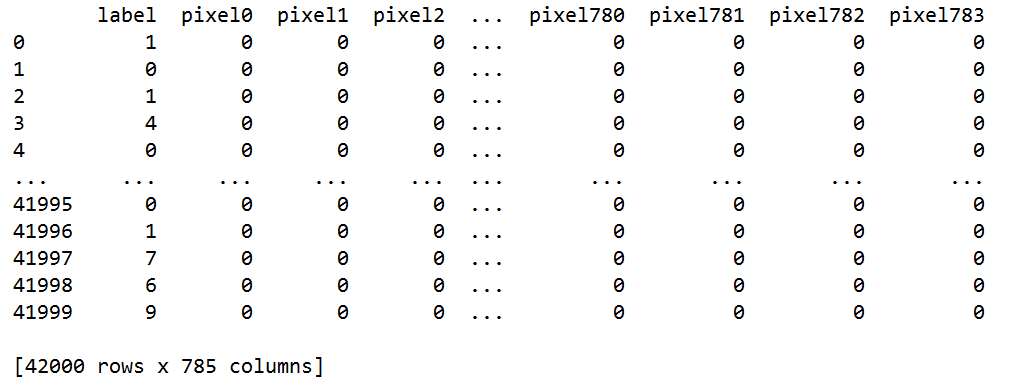


图2.1 训练集数据导入

把导入数据格式经过处理转换形式，并把数据分成标签集合和不带标签的样本集合两部分，以便后续运算，处理结果如下所示：

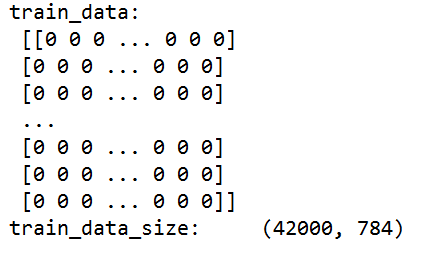


图2.2 训练数据集的不带标签的样本集合

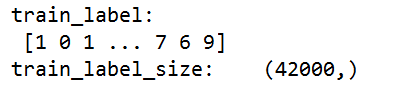


图2.3 训练数据集的标签集合

测试集数据导入如下图所示：

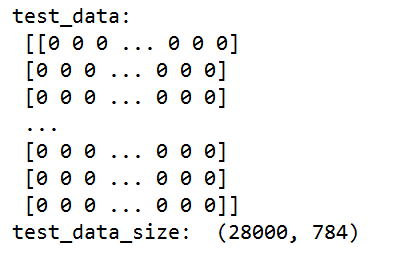


图2.4 测试数据集的样本集合

程序代码如下：

|  |
| --- |
| #导入要使用的工具包  import operator  import numpy as np  import pandas as pd |
| if \_\_name\_\_ == '\_\_main\_\_':  """读取文件信息，存放到矩阵中"""  #读取训练集数据CSV格式文件  train = pd.read\_csv('train.csv')  #转化数据格式为数组格式，以便后续运算  train\_data = train.iloc[:,1:].values  train\_label = train.iloc[:,0].values  #读取测试集数据CSV格式文件  test = pd.read\_csv('test.csv')  test\_data = test.iloc[:,:].values |

一般来说，我们选择训练数据集中前K个与测试数据最相似的数据，再这就是K近邻算法的由来。

## 2.3 算法原理分析

方法是一种常用的监督学习方法，与其他方法相比，它没有训练过程，是一种典型的“懒惰学习”。优点是通俗易懂，编程简单；缺点是计算量大。

的工作原理为：存在训练集，训练集中每个样本数据都存在对应的标签，输入没有标签的新数据即测试数据，将测试数据的每个特征与训练集中的样本数据对应的特征进行比较，然后找到训练集中与测试数据特征最相似的数据，以该数据的分类标签作为测试数据的分类标签。示意图如图2.5所示，圆形表示测试数据，三角形和矩形表示数据类型。

中有三个基本要素，分别是距离度量、K值得选择和分类决策规则。

距离度量是用来表示两个数据间的相似程度，常用的方法是用闵可夫斯基距离，叫范数，当P取得不同值时，范数类型也不同。最常用的是当时的范数，也叫欧式距离，其公式表示如下：

 (2.1)

K值得选择会对算法的结果产生重大影响。当选择的K值较小时，会对近邻的数据敏感，判断测试数据的近似误差会减小，但估计误差会增大，模型变复杂，容易发生过拟合。当选择的K值较大时，会使得模型变简单，判断测试数据的估计误差会减小，但较远处的数据也会产生影响，使得近似误差会增大。一般来说，K会取一个比较小的值。

方法中的分类决策规则一般采用多数表决，即在K个数据中选择出现次数最多的分类，作为测试数据的分类。

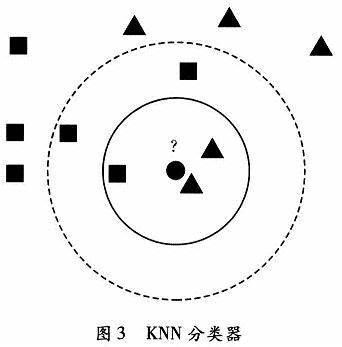


图2.5 分类器示意图

## 2.4 程序调试

方法的编程重点是在编写分类器，我的编程思路是：

1. 先导入数据
2. 计算测试数据与训练数据集中所有样本点之间的距离；
3. 把求出的距离按照递增排序；
4. 选取与当前点距离最小的K个点；
5. 返回前K个点出现频率最高的类别作为测试数据的分类。

分类器的程序为：

|  |
| --- |
| #导入要使用的工具包  import operator  import numpy as np  import pandas as pd  def classify(testing, train\_data, train\_label, k):  # numpy函数shape[0]返回train\_data的行数  train\_data\_size = train\_data.shape[0]  # 在列向量方向上重复testing共1次(横向),行向量方向上重复#train\_data\_size次(纵向)  diffMat = np.tile(testing, (train\_data\_size, 1)) - train\_data  # 二维特征相减后平方  diffMat\_sqrt = diffMat \*\* 2  # sum(1)行相加  distance = diffMat\_sqrt.sum(axis=1)  # 返回distances中元素从小到大排序后的索引值  sorted\_distance\_index = distance.argsort()  # 定一个记录类别次数的字典  classCount = {}  for i in range(k):  # 取出前k个元素的类别  voteIlabel = train\_label[sorted\_distance\_index[i]]  # dict.get(key,default=None),字典的get()方法,返回指定键的值,如果值不#在字典中返回默认值。  # 计算类别次数  classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1  # key=operator.itemgetter(1)根据字典的值进行排序  # reverse降序排序字典  sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), reverse=True)  # 返回次数最多的类别,即所要分类的类别  return sortedClassCount[0][0] |

K值的选取。考虑到如果逐个的用不同的K值去求在全部训练集下的正确率，耗时较长。所以，为选取合适的K值，我在训练集中选取10000个数据做为新的训练数据集，选取2000个数据做测试集（由于原训练集中的数据是无序的，所以我没有使用交叉验证法），使K的取值在1~9之间，程序如下（调用的分类器见以上程序）：

|  |
| --- |
| #导入要使用的工具包  import operator  import numpy as np  import pandas as pd  def handwritingClassTest(train\_data,train\_label,test\_data,test\_label,k):  # 错误检测计数  error\_num = 0.0  # 测试数据的数量  test\_lence = len(test\_label)  # 从文件中解析出测试集的类别并进行分类测试  test\_result = []  for i in range(test\_lence):  testing = test\_data[i]  classifierResult = classify(testing, train\_data,train\_label, k)  test\_result.append([i+1,classifierResult])  # result = pd.DataFrame(test\_result,columns=['ImageId','Label'])  # result.to\_csv('new\_result.csv',index=False)  if (classifierResult != test\_label[i]):  error\_num += 1.0  #print(error\_num)  print("总共错了%d个数据\n正确率为%.2f%%" % (error\_num, (test\_lence-error\_num)\*100 / test\_lence))  print("")  """读取文件信息，存放到矩阵中"""  #读取训练集数据CSV格式文件，并转换格式存放变量中  train = pd.read\_csv("train.csv")  train\_lence = len(train)  train\_data = train.iloc[:10000,1:].values  train\_label = train.iloc[:10000,0].values  test\_data = train.iloc[10000:12000,1:].values  test\_label = train.iloc[10000:12000,0].values  for i in range(1,10):  print("k=%d时:"% i)  handwritingClassTest(train\_data,train\_label,test\_data,test\_label,i) |

程序运行结果如下：



图2.6 K在1~6范围取值时的测验结果

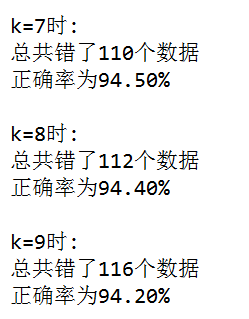


图2.7 K在7~9范围取值时的测验结果

由测试可得，当K=4时，错误率达到最小。所以用4做为方法中K的取值。

当使用整个要求的训练集数据和测试集数据时，即训练集数据个数是42000个，测试集数据个数是28000个，程序的运行时间较长。而且由于测试集中的数据没有对应的标签，需要在程序中把测试集数据的标签用方法预测出来，再提交到上，查看准确率。把测试集数据的预测结果按照上的要求以格式存放到文件夹中，程序如下所示：

|  |
| --- |
| def handwritingClassTest(train\_data,train\_label,test\_data,k):  # 测试数据的数量  test\_lence = test\_data.shape[0]  # 从文件中解析出测试集的类别并进行分类测试，并把结果保存到文件夹中  test\_result = []  for i in range(test\_lence):  testing = test\_data[i]  classifierResult = classify(testing, train\_data,train\_label, k)  test\_result.append([i+1,classifierResult])  result = pd.DataFrame(test\_result,columns=['ImageId','Label'])  result.to\_csv('new\_result.csv',index=False) |

最后得到的测试集数据预测的准确率为97.185%，如下图所示：

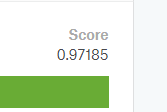


图2.8 测试所得的准确率

# 三 作业总结

在本次做题中，我也碰到一些之前做小练习时没碰到过的问题，比如本次的数据量较大，整个程序不能够在我预料的较短时间内完成。最后编程测验得到的准确率还可以，但我在上提交后得到的排名只能排带一千五百多名。这使我意识到，还有很多好的方法我可以去学习尝试。

这次作业使用编程来解决具体问题—手写数字识别，让我对模式识别有了更深入的了解，不仅使得我对于在课堂上学到的知识有了更好的掌握，增加了我的动手能力，训练了我的编程能力，还增加了我对模式识别的兴趣。非常感谢赵老师在课程中给我的教导。

**附：完整程序**

|  |
| --- |
| # 导入工具包  import numpy as np  import operator  import pandas as pd  """  函数说明:kNN算法,分类器  参数:  testing ： 用于分类的一个数据  train\_data ： 训练集数据  train\_label ： 训练集数据标签  k ： kNN算法参数,选择距离最小的k个点  返回值:  sortedClassCount[0][0] ： 返回分类结果  """  def classify(testing, train\_data, train\_label, k):  # numpy函数shape[0]返回train\_data的行数  train\_data\_size = train\_data.shape[0]  # 在列向量方向上重复testing共1次(横向),行向量方向上重复train\_data\_size次(纵向)  diffMat = np.tile(testing, (train\_data\_size, 1)) - train\_data  # 二维特征相减后平方  diffMat\_sqrt = diffMat \*\* 2  # sum(1)行相加  distance = diffMat\_sqrt.sum(axis=1)  # 返回distances中元素从小到大排序后的索引值  sorted\_distance\_index = distance.argsort()  # 定一个记录类别次数的字典  classCount = {}  for i in range(k):  # 取出前k个元素的类别  voteIlabel = train\_label[sorted\_distance\_index[i]]  # dict.get(key,default=None),字典的get()方法,返回指定键的值,如果值不在字典中返回默认值。  # 计算类别次数  classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1  # key=operator.itemgetter(1)根据字典的值进行排序  # reverse降序排序字典  sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), reverse=True)  # 返回次数最多的类别,即所要分类的类别  return sortedClassCount[0][0]  """  函数说明:手写数字识别分类  参数:  train\_data ： 训练集数据  train\_label ： 训练集数据标签  test\_data ： 测试集数据  k ： kNN算法参数,选择距离最小的k个点  返回值:  无  """  def handwritingClassTest(train\_data,train\_label,test\_data,k):  # 错误检测计数  #error\_num = 0.0  # 测试数据的数量  test\_lence = len(test\_label)  # 从文件中解析出测试集的类别并进行分类测试  test\_result = []  for i in range(test\_lence):  testing = test\_data[i]  classifierResult = classify(testing, train\_data,train\_label, k)  test\_result.append([i+1,classifierResult])  # 把分类所得的结果以特定的格式保存到new\_result.csv文件夹中  result = pd.DataFrame(test\_result,columns=['ImageId','Label'])  result.to\_csv('new\_result.csv',index=False)  # if (classifierResult != test\_label[i]):  # error\_num += 1.0  # print("总共错了%d个数据\n正确率为%.2f%%" % (error\_num, (test\_lence-error\_num)\*100 / test\_lence))  # print("")  """  函数说明:主函数  参数:  无  返回值:  无  """  if \_\_name\_\_ == '\_\_main\_\_':  """读取文件信息，存放到矩阵中"""  #读取训练集CSV格式文件  train = pd.read\_csv('train.csv')  #转化数据格式为数组格式，以便后续运算  train\_data = train.iloc[:,1:].values  train\_label = train.iloc[:,0].values  #读取测试集CSV格式文件  test = pd.read\_csv('test.csv')  test\_data = test.iloc[:,:].values  handwritingClassTest(train\_data,train\_label,test\_data,4) |