

**课程实验报告**

**题目： KNN算法实验报告**

**课程名称： 机器学习**

**专业班级： CS1703**

**学 号： U201714600**

**姓 名： 李泽贤**

**指导教师： 李玉华**

**报告日期： 2020年5月28日**

**计算机科学与技术学院**

目录

[1实验一 3](#_Toc11046)

[1.1实验目的与要求 3](#_Toc17663)

[1.2实验内容 3](#_Toc24825)

[1.3实验方案 3](#_Toc10668)

[1.4实验结果 7](#_Toc11806)

# 1实验一KNN算法

## 1.1实验目的与要求

（1）熟悉python方法；

（2）实现KNN算法；

（3）使用MNIST数据集测试；

（4）输入若干测试图片，输出对应每张图片 k 近邻的图片。

（5）绘制 knn 算法的训练 misclassification rate 曲线，并做出分析

## 1.2实验内容

编程使用python语言实现KNN算法，并解析MNIST数据集，用于测试KNN算法。

编写相关函数输出某一测试用例的K邻近图片。

测试不同k的取值对结果的影响，并绘制misclassifification rate曲线

## 1.3实验方案

首先读取这四个数据集

输入一个k值（整数）作为kNN算法的k值

程序会计算前1000个测试数据集利用KNN算法所得分类结果，计算正确率并显示

输入一个i值（整数）来代表想要测试的数据

对测试集中第i个数据利用kNN算法进行标签分类

在终端会显示测试数据的k个近邻数据的标签，测试数据的实际标签以及利用KNN算法所得的标签

图像窗口显示第i个手写数据,标题为测试数据实际的标签和利用kNN算法所得的标签

### 1.3.1解析数据集

实验数据集采用MNIST数据集，其中包含4个文件

train-images.idx3-ubyte 训练集数据文件名

train-labels.idx1-ubyte 训练集标签文件名

t10k-images.idx3-ubyte 测试集数据文件名

t10k-labels.idx1-ubyte 测试集标签文件名

其中训练集包含60000个示例，测试机包含10000个示例，通过二进制的方式打开数据集，并以一维数组的方式存储标签，以矩阵的方式存储每个图片示例。具体代码为：

读取数据集标签：

def loadLabelSet(filename):

binfile=open(filename,'rb') #打开文件

buffers=binfile.read() #读取文件的所有内容

head=struct.unpack\_from('>II',buffers,0) #>表示大端模式，读取两个字节,第一个字节表示魔数，第二个字节表示标签数量

labelNum=head[1] #标签数量labelNum

offset=struct.calcsize('>II') #重新设置偏移地址

numString='>'+str(labelNum)+'B' #格式为>6000B

labels=struct.unpack\_from(numString,buffers,offset)

binfile.close()

labels=np.reshape(labels,(labelNum))#转化为一维数组（列向量）

return labels,labelNum

读取数据集数据示例：

def loadImageSet(filename):

binfile=open(filename,'rb') #打开文件

buffers=binfile.read() #读取文件的所有内容

head=struct.unpack\_from('>IIII',buffers,0) #读取四个字节，第一个字节表示魔数，第二个字节表示数据个数，第三四个字节分别表示行数，列数

offset=struct.calcsize('>IIII')

imgNum=head[1]

width=head[2]

height=head[3]

bits=imgNum\*width\*height

bitsString='>'+str(bits)+'B'

imgs=struct.unpack\_from(bitsString,buffers,offset)

binfile.close()

list1=[]

for i in range(bits):

if imgs[i]>127:

list1.append(1)

else:

list1.append(0)

tuple(list1)

imgs=np.reshape(list1,(imgNum,width\*height))

return imgs

### 1.3.2KNN算法

KNN算法实现首先计算训练集各示例与测试示例的距离，然后对距离进行排序，选择k个近邻，计算k个近邻中各标签出现的次数，最后返回出现次数最多的类别标签。knn算法同实战书中讲解用例一致。

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

numSample=dataSet.shape[0] #训练集数据的行数(多少个训练数据)

#step1:计算距离, P=2

#tile(A,(B,C)) 将A在行方向重复B次，在列方向重复C次,这里得到了x行28\*28列的矩阵

diff=np.tile(newInput,(numSample,1)) -dataSet #按元素求差值

squaredDiff=diff\*\*2 #将差值平方

squaredDist=np.sum(squaredDiff,axis=1) #一行的所有元素相加

distance=squaredDist\*\*0.5 #将差值平方和求开方，即为距离

#step2:对距离排序

sortDistIndices=np.argsort(distance) #argsort()返回排序后的索引值

classCount={} #定义一个空字典，方便添加元素,key=标签，value=该标签出现的次数

#print('testdata\'s',k,'near data\'s label is',end=' ')

for i in range(k): #0-(k-1)

#step3:选择k个近邻

voteLabel=labels[sortDistIndices[i]] #第i个近邻的标签

#print( voteLabel,end=',') #显示测试集数据的k个近邻的标签

#step4:计算k个最近邻中各类别出现的次数

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

#step5:返回出现次数最多的类别标签

maxCount=0 #初始化标签出现的最多次数为0

for key,value in classCount.items():

if value > maxCount:

maxIndex=key

maxCount=value

return maxIndex,sortDistIndices[0:k]

### 1.3.4测试过程

file1='C:\\Users\\asus\\Desktop\\机器智能云\\机器学习\\knnproj\\train-images.idx3-ubyte' #训练集数据文件名（包含文件路径）

file2='C:\\Users\\asus\\Desktop\\机器智能云\\机器学习\\knnproj\\train-labels.idx1-ubyte' #训练集标签文件名

file3='C:\\Users\\asus\\Desktop\\机器智能云\\机器学习\\knnproj\\t10k-images.idx3-ubyte' #测试集数据文件名

file4='C:\\Users\\asus\\Desktop\\机器智能云\\机器学习\\knnproj\\t10k-labels.idx1-ubyte' #测试集标签文件名

#生成数据集和类别标签

dataSet=loadImageSet(file1)

labels,num1=loadLabelSet(file2)

dataTest=loadImageSet(file3)

labelsTest,num2=loadLabelSet(file4)

k=int(input("input a number k as the k value of kNN algorithm:")) #输入一个整数k,作为kNN算法的k值

num=0

for a in range(200):

testX1=dataTest[a,:] #从测试数据集和类别标签中取出第i个数据

truelabelX1=labelsTest[a]

outputLabelX1,sortlist=kNNClassify(testX1,dataSet,labels,k)

if outputLabelX1==truelabelX1:

num=num+1

print("when the k is ",k,", the accuracy is ",num/200.00)

i=int(input("input a number i as the testdata:")) #输入一个整数i,表示选择第i个测试数据

testX=dataTest[i,:] #从测试数据集和类别标签中取出第i个数据和标签

truelabel=labelsTest[i]

outputLabel,sortlist1=kNNClassify(testX,dataSet,labels,k)

print( "testdata's real label:",truelabel,",")

print( "its classified label:",outputLabel)

display(testX,outputLabel) #用图像显示这个测试数据

# 用图像显示该测试数据的k邻近图片

for i in range(len(sortlist1)):

testX2=dataSet[sortlist1[i],:]

truelabelX2=labels[sortlist1[i]]

display(testX2,truelabelX2)

# 测试图片数

testNum=200

# 测试k取不同的值时的结果，并绘制misclassification rate图像

x=np.arange(1,10,1)

y=[]

for t in x:

num1=0

for b in range(testNum):

testX3=dataTest[b,:] #从测试数据集和类别标签中取出第i个数据

truelabelX3=labelsTest[b]

outputLabelX3,sortlist3=kNNClassify(testX3,dataSet,labels,t)

#print( " testdata's real label:",truelabelX1,end=',')

#print( " its classified label:",outputLabelX1)

if outputLabelX3==truelabelX3:

num1=num1+1

# print(num1)

print("when the k is ",t,", the accuracy is ",num1/float(testNum))

misclassification\_rate=1-num1/float(testNum)

y.append(misclassification\_rate)

plt.plot(x,y,label='KNN')

plt.xlabel("K")

plt.ylabel("misclassification\_rate")

plt.ylim(0,2\*max(y))

plt.legend()

plt.show()

## 1.4实验结果

### 1.4.1测试KNN算法结果

当测试用例为200个, k取3时，测试结果如图1所示。

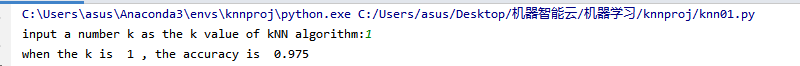


图1：KNN测试结果

KNN测试正确率为97.5%



图2：第66个测试用例图像

然后输出该测试用例的k近邻图像（k取3），如图3所示。

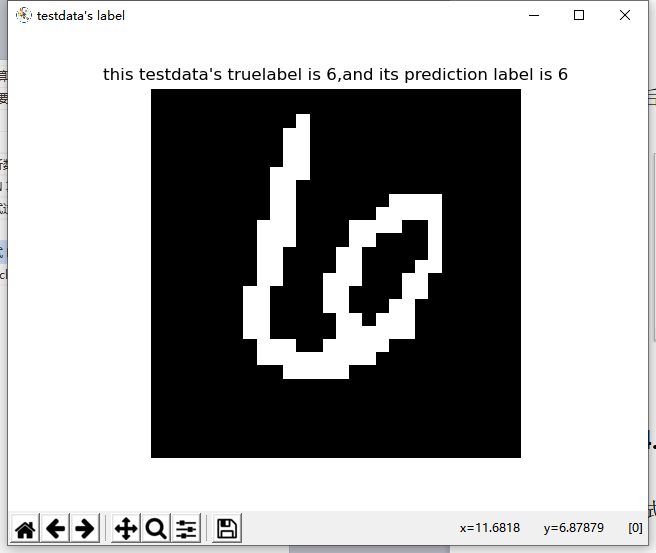
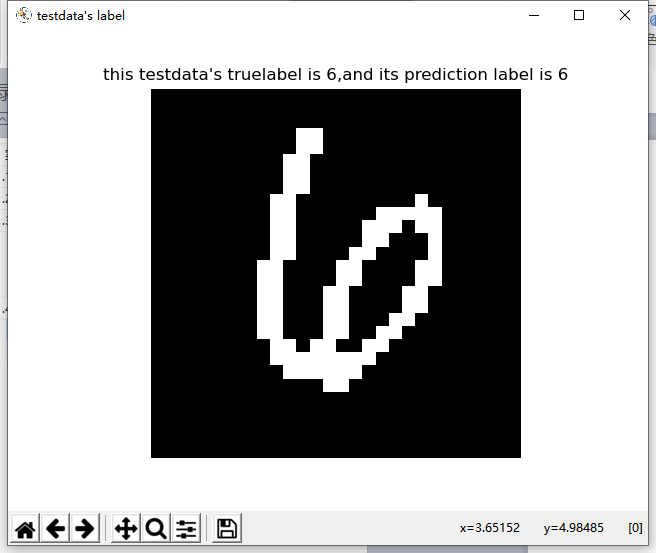


图3：k近邻图像

### 1.4.2 misclassifification rate曲线

测试k取1到9之间数值时，KNN算法所得到的不同的结果，如图4所示。

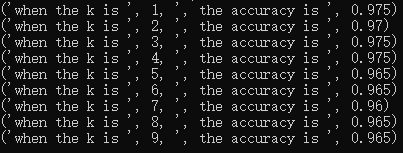


图4：k取不同值时的结果

绘制的misclassifification rate曲线如图5所示。

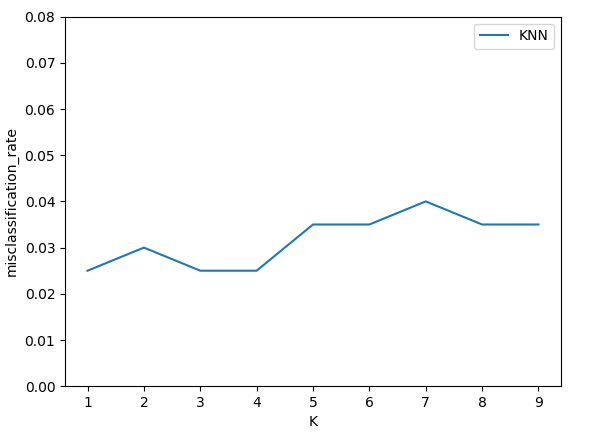


图5：misclassifification rate曲线