**北京邮电大学软件学院**

**2017－2018学年第二学期实验报告**

**课程名称： Python语言与编程**

**项目名称： Python应用 (人工智能，网络通信)**

**项目完成人：**

**姓名：\_\_\_田宇\_\_\_\_\_**

**学号：\_2016212011\_**

**指导教师： 管浩**

**日 期： 2018 年 6 月 15 日**

1. **实验目的**

通过使用Python进行机器学习K近邻算法实践与网络编程实践，理解并掌握Python解决实际问题的方法，同时巩固加深对已学Python的知识的理解与掌握。**实验内容**

1. **实验环境**

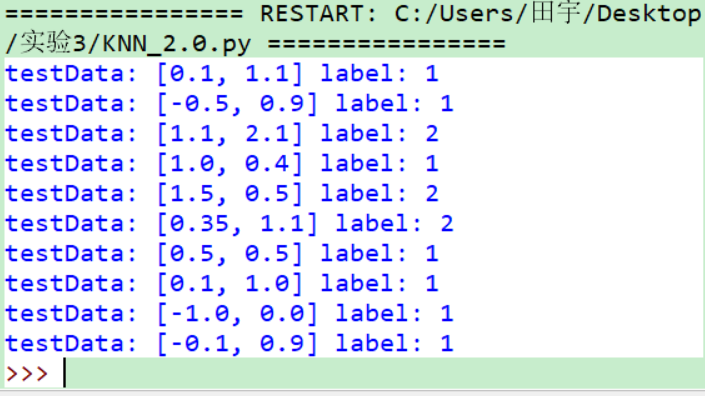
Windows10 IDLE

1. **实验结果**

见附录

1. **附录**
   1. **KNN算法**

**实验结果：**



**调试心得：**

1. 算法

KNN是通过测量不同特征值之间的距离进行分类。它的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别，其中K通常是不大于20的整数。KNN算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

1. Sting.replace()函数可以替换掉读文件产生的多余的字符，

str.strip([chars])只能取出首尾的指定字符串。

1. 文件内容读出来默认是字符串，需要类型转换。
2. numpy 的argsort()函数返回排列过后的脚标，利用前k个脚标 来判断测试数据属于哪个类型。
3. 给定l=[]以后，不能再用l[i]=...因为长度固定为0了，只能 用append或者insert。

**源代码：**

'''

KNN算法的实现，首先将数据文件转化成txt文件，比较容易处理，通过查询微软的表格文档也是可以处理的

但是还要包含很多模块，配置环境比较复杂，所以这次就直接将表格导出为txt

'''

import codecs

from math import sqrt

from numpy import \*

#将训练数据导入准备进行训练

def importData():

with codecs.open('C:\\Users\\田宇\\Desktop\\实验3\\train.txt','r') as file:

data=[]

for i in range(0,200):

d=file.readline().replace('\t',' ').replace('\n','')

data.append(d)

return data

#进行训练

def classify(testData,data,k):

dist=[]

#欧氏距离的计算

for i in range(0,200):

d0=data[i].split(' ')

distance=sqrt((testData[0]-float(d0[0]))\*\*2+(testData[1]-float(d0[1]))\*\*2)

dist.append(distance)

sortedDistIndex = argsort(dist)

#选择标签

n1,n2=0,0

for i in range(0,k):

if sortedDistIndex[i] in range(100):

n1+=1

else:

n2+=1

if n1>n2:

label0=1

else:label0=2

return label0

#将测试文件包含进去

t\_data=[]

with codecs.open('C:\\Users\\田宇\\Desktop\\实验3\\test.txt','r') as f:

for i in range(0,10):

d=f.readline().replace('\t',' ').replace('\n','')

t\_data.append(d)

test=[]

for i in range(0,10):

d0=t\_data[i].split(' ')

test.append(d0)

testData=[0,0,0,0,0,0,0,0,0,0]

for i in range(0,10):

testData[i]=[float(n) for n in test[i]]

data=importData()

for i in range(10):

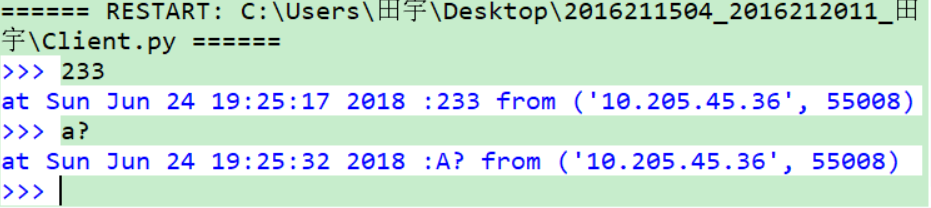
label0=classify(testData[i],data,5)

print('testData:',testData[i],'label:',label0)

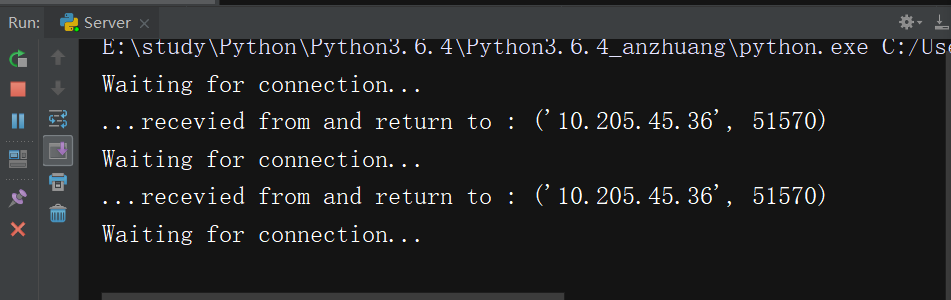
2、简单的数据收发器

**实验结果：**

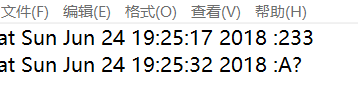
Client:



Server:



存储



**调试心得：**

1. 发送的必须是byte类型，需用bytes()函数进行类型转换。
2. 时间与信息一起打包，接收端用string.split()函数分离。
3. 获取系统时间：t=asctime(localtime(time()))

**源代码：**

**Server**

from socket import \*

from time import ctime

host = '' #监听所有的ip

port = 55008 #接口必须一致

bufsize = 1024

addr = (host,port)

udpServer = socket(AF\_INET,SOCK\_DGRAM)

udpServer.bind(addr) #开始监听

rfile = open('test\_message.txt', 'w')

while True:

print('Waiting for connection...')

data,addr = udpServer.recvfrom(bufsize)

#接收数据和返回地址

#处理数据

data = data.decode(encoding='utf-8').upper()

data = "at %s :%s"%(ctime(),data)

rfile = open('test\_message.txt', 'a')

rfile.write(data + '\n')

udpServer.sendto(data.encode(encoding='utf-8'),addr)

#发送数据

print('...recevied from and return to :',addr)

rfile.close()

os.chdir(basePath)

udpServer.close()

Client

from socket import \*

host = '10.205.45.36' #电脑的ip

port = 55008

bufsize = 1024

addr = (host,port)

udpClient = socket(AF\_INET,SOCK\_DGRAM)

while True:

#作为客户端输入，将无限循环是个好方法

data = input('>>> ')

if not data:

break

data = data.encode(encoding="utf-8")

if data=='BYE'.encode(encoding="utf-8"):

break

udpClient.sendto(data,addr) # 发送数据

data,addr = udpClient.recvfrom(bufsize) #接收数据和返回地址

print(data.decode(encoding="utf-8"),'from',addr)

udpClient.close()