

**计算机与信息 学院实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验课程： | 人工智能实验 | | | | |
| 实验编号： | 期末作业 | | | | |
| 实验名称： | 识别手写数字 | | | | |
| 实验人员： | 学号 | 18111207248 | | | |
| 姓名 | 吴钰 | | | |
| 班级 | 18级计算机科学与技术（创新班） | | | |
| 实验日期： | 2020.1.8 | | | | |
| 实验室： | 2060402 | | | | |
|  |  | | | | |
| 实验评价： |  | | | | |
| 实验成绩： | |  | 评价日期： |  |
|  | 指导教师： | |  | | |

# 实验目的

## 1、 了解掌握感知机算法思想

## 2、 学习识别手写数字的算法

## 3、 使用Python实现

# 二、实验要求

## 完成实验后请填写实验报告并上交。要求在实验报告中必须记录实验中遇到的问题及其问题解决方案。

## 研究如何利用计算机自动辨认手写在纸张上的数字。手写体数字的随意性很大,字体大小、倾斜、笔画的粗细等都会对识别结果造成影响。

## 要求：1. 给出设计的思路和原理，使用的方法和构建的过程，提交设计的word文档和编写的代码；2.禁止直接从网上下载复制别人代码，和禁止直接调用已有的接口，否则实验课记为0分。

# 三、实验内容

基本思路：

1.定义神经网络：

定义输入层

定义隐藏层

定义输出层

找到权重和偏差

设置学习率

设置激活函数

2.定义训练网络：

根据训练样本给出输出

根据输出结果

计算误差

根据误差反向更新隐藏层权重

3.定义查询网络：

预测新样本的种类，即通过神经网络最终的输出值。

4.定义主函数：

各层数据的输入

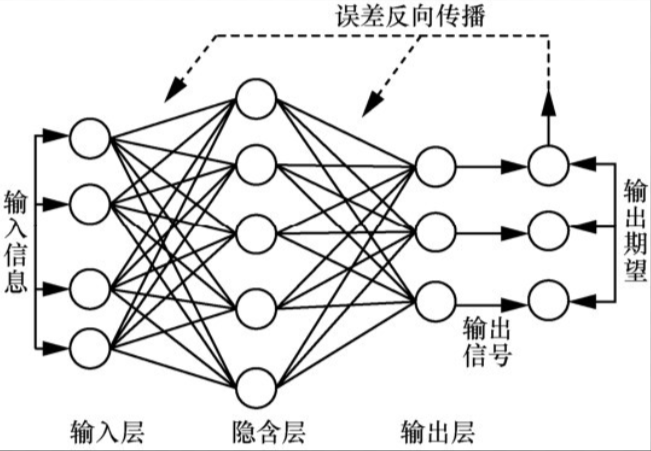
学习率的输入

生成神经网络

5.计算查询结果的正确率：

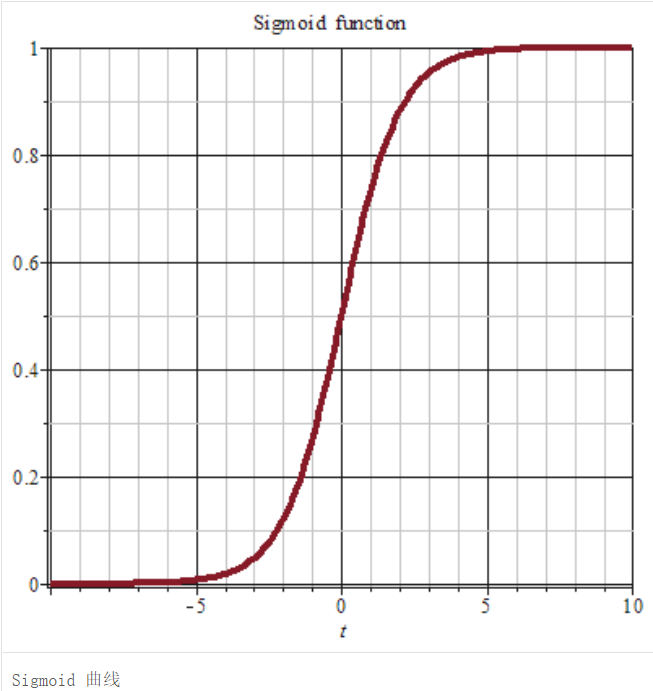
每个数字得出的输出值与正确值进行比较，得出正确率。

代码主要分为四大板块，首先是数据读取，第二个部分是神经网络的配置，第三部分是神经网络的训练，第四部分是神经网络的测试,BP神经网络是一个有监督学习模型，是神经网络类算法中非常重要和典型的算法，三层神经网络的基本结构如下：



  BP神经网络运行机理就是：一个特征向量的各个分量按不同权重加权，再加一个常数项偏置，生成隐藏层各节点的值。隐藏层节点的值经过一个激活函数激活后，获得隐藏层激活值，隐藏层激活值仿照输入层到隐藏层的过程，加权再加偏置，获得输出层值，通过一个激活函数得到最后的输出。

    激活函数可使用值域为（0，1）的sigmoid函数，也可使用值域为（-1，1）的tanh函数，激活函数图像如下：



  如果我们试图确定手写图像是否描绘为“9”。 设计网络的一种自然方式是将图像像素的强度编码到输入神经元中。 如果图像是64×64灰度图像，那么我们将有4,096 = 64×64 个输入神经元（像素点），强度在0和1之间适当缩放。输出层将只包含一个神经元，输出小于0.5的值表示“输入图像不是9”，大于0.5的值表示“输入图像是9”。

输入层：输入层包含编码输入像素值的神经元。我们的网络训练数据将由扫描的手写数字的28×28像素图像组成，因此输入层包含784 = 28×28 个神经元。

隐藏层：我们用n表示这个隐藏层中的神经元数量，我们将试验n的不同值。

输出层：如果第一个神经元激活，即输出 ≈ 1，那么这将表明网络认为该数字是0。如此类推。更准确地说，我们将输出神经元从0到9编号，并找出哪个神经元来匹配数字的最大激活值。 因为输出层只有10种结果，所以为10。

设计好神经网络，要实现数字的识别，首先是要学习的数据集即训练数据集。

代码运用MNIST数据。它包含60,000张图像用作训练数据，尺寸为28 x 28像素；10,000个图像，用作测试数据，尺寸为28 x 28像素。输入像素是灰度，值为0.0表示白色，值1.0表示黑色，在0到1之间灰度由浅变深。我们将使用测试数据来评估我们的神经网络学会识别数字的程度。

过程：

训练网络：

1.正向传播：针对训练样本给出输出，定义输入层与隐藏层之间的初始权重参数。

2.误差反向传播：针对输出，计算误差，根据误差反向更新隐藏层神经的误差，更新初始权重。

3.更新权重。

**源代码：**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jan 8 20:35:02 2021

@author: wuyu

"""

import numpy

import scipy.special

import matplotlib.pyplot

'''

python搭建BP神经网络，

并实现手写数字识别

'''

class neuralNetwork:#创建神经网络类

def \_\_init\_\_(self,inputnodes,hidennodes,outputnodes,learningrate):

'''初始化输入层、隐藏层、输出层结点的个数及学习率'''

self.inodes = inputnodes

self.hnodes = hidennodes

self.onodes = outputnodes

self.lr = learningrate

#输入层与隐藏层之间的初始权重参数

self.wih = numpy.random.normal(0.0,pow(self.hnodes,-0.5),(self.hnodes,self.inodes))

#隐藏层、输出层之间的初始权重参数

self.who = numpy.random.normal(0.0,pow(self.onodes,-0.5),(self.onodes,self.hnodes))

#sigmoid：激活函数

self.activation\_function = lambda x:scipy.special.expit(x)

def train(self,inputs\_list,target\_list):

inputs = numpy.array(inputs\_list, ndmin=2).T

targets = numpy.array(target\_list, ndmin=2).T

hidden\_inputs = numpy.dot(self.wih, inputs)#隐藏层输入

hidden\_outputs = self.activation\_function(hidden\_inputs)#输出

final\_inputs = numpy.dot(self.who, hidden\_outputs)#最终输入

final\_outputs = self.activation\_function(final\_inputs)#输出

#=========================计算输出误差==============================

output\_errors = targets - final\_outputs

hidden\_errors = numpy.dot(self.who.T,output\_errors)

self.who += self.lr \* numpy.dot((output\_errors \* final\_outputs \* (1.0-final\_outputs)),numpy.transpose(hidden\_outputs))

self.wih += self.lr \* numpy.dot((hidden\_errors \* hidden\_outputs \* (1.0-hidden\_outputs)),numpy.transpose(inputs))

#=====================查询网络，用于预测新样本的种类==================

def query(self, inputs\_list):

inputs = numpy.array(inputs\_list,ndmin=2).T

hidden\_inputs = numpy.dot(self.wih,inputs)

hidden\_outputs = self.activation\_function(hidden\_inputs)

final\_inputs = numpy.dot(self.who,hidden\_outputs)

final\_outputs = self.activation\_function(final\_inputs)

return final\_outputs

# 创建神经网络对象并用MNIST训练集训练

input\_nodes = 784

hidden\_nodes = 200

output\_nodes = 10

learning\_rate = 0.2 # 学习率

n = neuralNetwork(input\_nodes,hidden\_nodes,output\_nodes,learning\_rate)

# 读取训练集

data\_file = open("mnist\_train1.csv",'r')

data\_list = data\_file.readlines()

data\_file.close()

epochs =5

for i in range(epochs):

for record in data\_list:

all\_values = record.split(',')

inputs = (numpy.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01

targets = numpy.zeros(output\_nodes) + 0.01

targets[int(all\_values[0])] = 0.99

n.train(inputs, targets)

# ----------------------测试------------------------------

test\_data\_file = open("mnist\_test1.csv",'r')

test\_data\_list = test\_data\_file.readlines()

test\_data\_file.close()

scorecard = []

for record in test\_data\_list:

all\_values = record.split(',')

correct\_value = int(all\_values[0])

print('待识别数字：',all\_values[0])

image\_array = numpy.asfarray(all\_values [1:]).reshape((28,28))

matplotlib.pyplot.imshow( image\_array , cmap='Greys' , interpolation='None')

inputs = (numpy.asfarray(all\_values[1:])/255.0 \* 0.99) + 0.01

outputs = n.query(inputs)

label = numpy.argmax(outputs)

print('模型识别出的数字：',label)

if (label == correct\_value):

scorecard.append(1)

else:

scorecard.append(0)

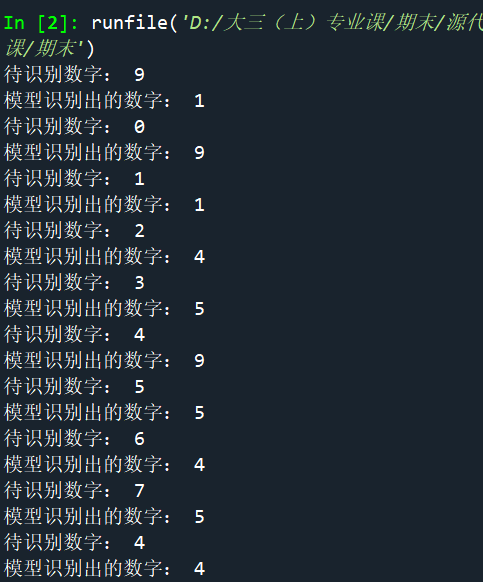
scorecard\_array = numpy.asarray(scorecard)

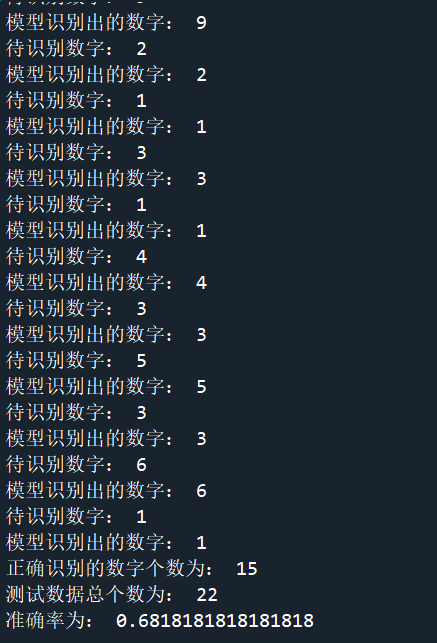
print('正确识别的数字个数为：',scorecard\_array.sum())

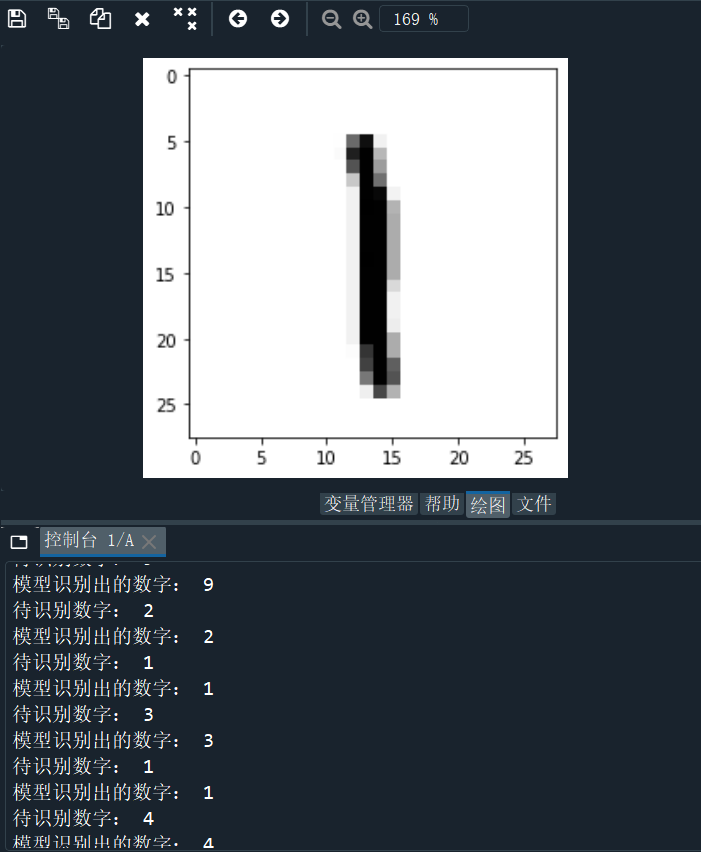
print('测试数据总个数为：',scorecard\_array.size)

print("准确率为：",scorecard\_array.sum() / scorecard\_array.size)

运行截图：

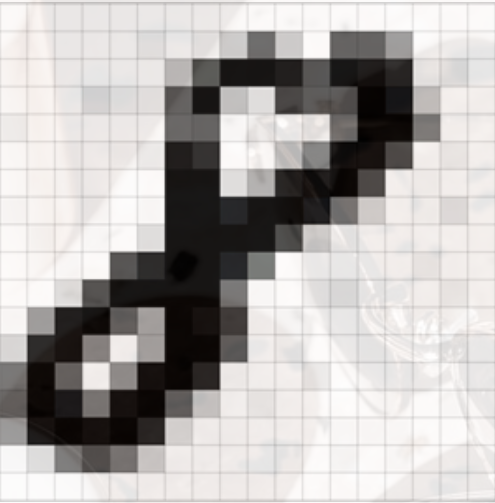








放大后



# 四、实验思考

通过此次试验，我学习了如何使用BP神经网络进行手写数字的识别，BP神经网络运行机理就是：一个特征向量的各个分量按不同权重加权，再加一个常数项偏置，生成隐藏层各节点的值。隐藏层节点的值经过一个激活函数激活后，获得隐藏层激活值，隐藏层激活值仿照输入层到隐藏层的过程，加权再加偏置，获得输出层值，通过一个激活函数得到最后的输出。由于人手写的数字图片和数据集的图片信息存在一定的差别，人工写出的数字可能并不规范，会出现图片规格的不同、清晰度不同，有污损等，所以手写图片需要进行处理。由于考虑到程序运行时间的原因，在训练时对数据做出适当删减，为了提高识别正确率，我们可以再适当增加训练集数据，但也要避免出现“过拟合”现象。