****

**编程实习**

**Python神经网络编程**

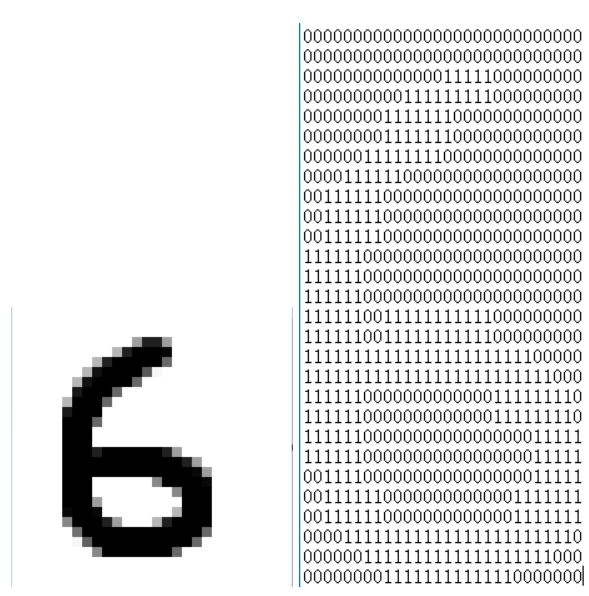
|  |  |
| --- | --- |
| 专 业 | 自动化 |
| 班 级 | 18062911 |
| 学 号 | 18061319 |
| 学生姓名 | 潘宗勇 |
| 指导教师 | 薛梦凡 |
| 完成日期 | 2019年7月13日 |

1. **项目介绍**

本项目是学习python的相关知识，以及学习神经网络的知识，利用python创建神经网络，对手写数字进行识别，并测试神经网络。

1. **算法介绍**

输入信息经过隐藏层输出层权重计算得出结果，计算和标准结果的误差,再反向传播数据更新权重，来接近正确答案。



将测试数据由图片形式转换成只有一列的0-1矩阵形式：上左图中有像素点的位置记为1，没有像素点的位置记为0，成上右图（上右图是经过我切割、拉伸后的结果）。再将上右图中所有后一行数字接到前一行末尾，形成一行，最后转置即可得一列0-1矩阵形式

将所有（L个）训练数据也都用上方法从图片形式转换成只有一列的0-1矩阵形式

把L个单列数据存入新矩阵A中——矩阵A每一列存储一个图片的所有信息

用测试数据与矩阵A中的每一列求距离，求得的L个距离存入距离数组中（距离 = 对应位差值的平方和再求平方根）

从距离数组中取出最小的K个距离所对应的训练集的索引

拥有最多索引的值就是预测值（有多个众数时，按距离和最小）

**三．框架介绍**

（1）初始化函数

设置输入层节点、隐藏层节点和输出层节点的数量，定义神经网络的形状和尺寸。

创建网络的节点和链接，使用这些权重来计算前馈信号、反向传播误差，并且在试图改进网络

时优化链接权重本身。

1. 训练——学习给定训练集样本后，优化权重。

第一部分，针对给定的训练样本计算输出。这与我们刚刚在query()函数上所做的没什么区别。

第二部分，将计算得到的输出与所需输出对比，使用差值来指导网络

权重的更新。

（3）查询——给定输入，从输出节点给出答案。

**四．代码**

第一部分 函数初始化

**（1）#导入函数库**

import numpy

import scipy.special

import matplotlib.pyplot

%matplotlib inline

class neuralNetwork :

**（2）#创建每层三个节点，学习率**

**#使用\_\_init\_\_()方法来传递参数，赋值给对象属性等。**

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

self.inodes = inputnodes

self.hnodes = hiddennodes

self.onodes = outputnodes

**（3）#使用权重来计算前馈信号、反向传播误差，并且在试图改进网络 时优化链接权重本身。**

**#创建了两个链接权重矩阵，并使用self.inodes、self. hnodes和 self. onodes**

**#为两个链接权重矩阵设置了合适的大小。**

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

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

self.lr = learningrate

self.activation\_function = lambda x: scipy.special.expit(x)**#S函数**

pass

**第二部分 训练**

**#训练神经网络的过程中有两个阶段，第一个阶段就是计算输出，第二个阶段就是反向传播误差，告知如何优化链 接权重。**

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

inputs = numpy.array(inputs\_list,ndmin = 2).T**#二维数组转制**

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

hidden\_inputs = numpy.dot(self.wih, inputs) **#计算隐藏层中的信号**

hidden\_outputs = self.activation\_function(hidden\_inputs) **#计算隐藏层中出现的信号**

final\_inputs = numpy.dot(self.who, hidden\_outputs) **#将信号计算到最终输出层**

final\_outputs = self.activation\_function(final\_inputs) **#计算从最终输出层发出的信号**

output\_errors = targets - final\_outputs **#输出层错误（训练样本所提供的预期目标输出值与 实际计算得到的输出值之差）**

hidden\_errors = numpy.dot(self.who.T, output\_errors) **#隐藏层错误是输出错误，按权重分割，在隐藏节点重新组合**

**#更新隐藏层和输出层之间链接的权重**

**#transpose 作用是改变序列**

**#学习率是self.lr**

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

**#更新输入层和隐藏层之间链接的权重**

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

pass

**第三部分 查询**

**#查询神经网络**

def query(self, inputs\_list): **#query()函数接受神经网络的输入，返回网络的输出。**

**#将输入列表转换为二维数组**

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

hidden\_inputs = numpy.dot(self.wih, inputs) **#链接权重矩阵W input\_hidden 点乘输入矩阵I，计算隐藏层中的信号**

hidden\_outputs = self.activation\_function(hidden\_inputs) **#计算隐藏层中出现的信号**

final\_inputs = numpy.dot(self.who, hidden\_outputs) **#将信号计算到最终输出层**

final\_outputs = self.activation\_function(final\_inputs) **#计算从最终输出层发出的信号**

return final\_outputs

**第四部分 准备测试**

**（1）#输入、隐藏、输出节点数**

input\_nodes = 784

hidden\_nodes = 200

output\_nodes = 10

**#学习率**

learning\_rate = 0.1

**#创建神经网络实例**

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

**（2）#将文件导入列表**

training\_data\_file = open("E:\pzy的学习文档\python 神经网络\mnist\_train.csv", 'r')

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

training\_data\_file.close()

**#epochs是训练数据集用于训练的次数**

epochs = 5

for e in range(epochs):

**（3）**  **#查看培训数据集中所有数据**

for record in training\_data\_list:

all\_values = record.split(',') #用逗号拆分记录

inputs = (numpy.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01 **#将输入颜色值范围进行缩放**

targets = numpy.zeros(output\_nodes) + 0.01 **#创建目标输出值**

**#所有值【0】都是此纪录的目标标签**

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

n.train(inputs, targets)

pass

pass

**第五部分 测试神经网络**

**（1）#测试神经网络**

test\_data\_file = open("E:\pzy的学习文档\python 神经网络\mnist\_test.csv", 'r') **#将文件导入列表**

test\_data\_list = test\_data\_file.readlines() **#将文件中的所有行读入变量**test\_data \_list

test\_data\_file.close()**#关闭文件**

#测试自己写的数字

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

print(all\_values[0])

**（2）#检查测试数据集中所有记录**

scorecard = []

for record in test\_data\_list:

correct\_label = int(all\_values[0]) **#正确答案是第一个值**

inputs = (numpy.asfarray(all\_values[1:]) / 255.0 \* 0.99) + 0.01 **#缩放和移动输入**

outputs = n.query(inputs) **#查询网络**

label = numpy.argmax(outputs) **#最大值的索引对应标签**

**#追加正确或不正确的列表**

if (label == correct\_label):

scorecard.append(1) **#正确记分卡加1**

else:

scorecard.append(0) **#错误记分卡加0**

**pass**

pass

**（3）#计算成绩分数，正确答案的分数**

scorecard\_array = numpy.asarray(scorecard)

print ("performance = ", scorecard\_array.sum() / scorecard\_array.size)