****

**基于BP神经网络和MINST数据集的手写数字识别系统**

**引言**

自上世纪六十年代以来，计算机视觉与图像的处理越来越受到人们的关注，并逐渐成为一门重要的学科领域。而作为它们的研究对象的数字图像，也因为它含有研究目标的丰富信息而成为越来越重要的研究对象。图像识别的目标是用计算机自动完成某些信息的处理，用来代替人工去处理图像分类及识别的任务。

手写数字识别是图像识别学科的下一个分支，是图像处理和模式识别领域研究的课题之一，由于其具有很强的实用性一直是多年来的研究热点。由于手写体的随意性很大，例如，笔画的粗细，字体的大小，倾斜等等都直接影响到数字的正确识别，所以手写体数字识别是一个很有挑战性的课题。手写体数字识别实用性很强，在大规模数据统计（如例行车检，人口普查），财务，税务，邮件分拣等应用领域中都有广阔的应用前景。

**程序设计：**

本程序使用PyCharm编辑器，所用的Interprerer版本为：php3.0,本次使用的算法为经典BP神经网络

**程序总体结构**：包含那几个部分，例“数据处理、网络模型”，每一部分主要包含那些类、函数，并简要描述该部分主要实现的功能。

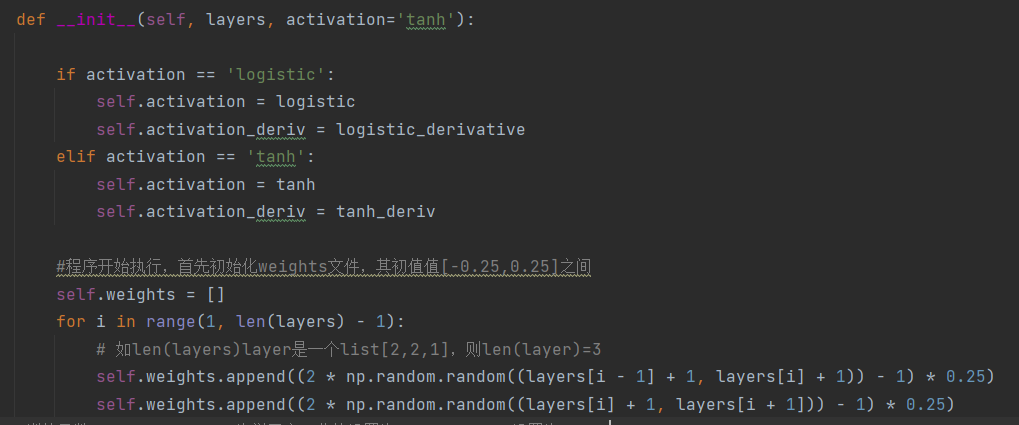
本应用设计实现了一个基于python语言的手写数字识别系统，采用了模块化设计方法，主要分为手写板输入和神经网络模型训练两个模块，实现对单个手写数字的识别。。

本应用包含四个py文件，采用模块化的方式进行编程，接下来将逐一对各个模块所实行的功能做进一步的介绍。

**NN．Py文件 主要用来实现神经网络算法**，

class NeuralNetwork:

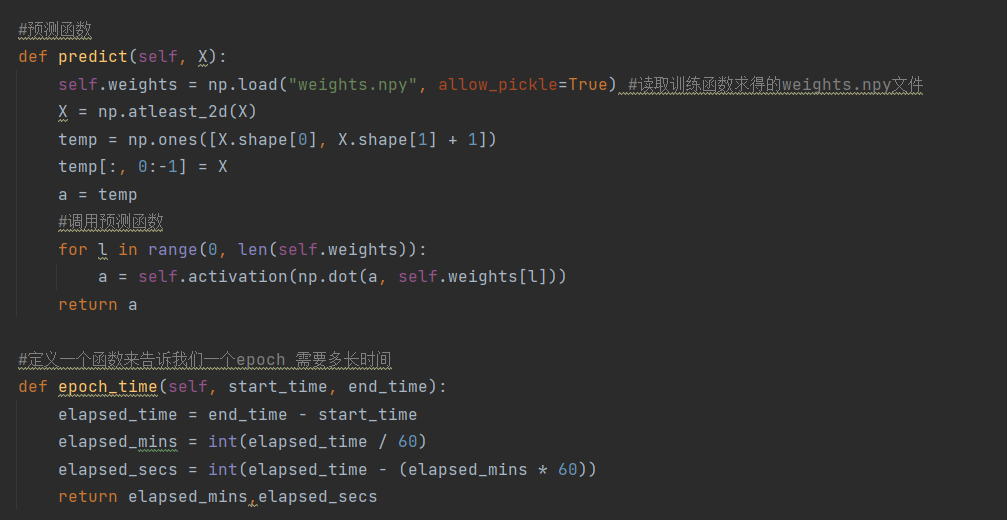
NeuralNetwork类作为神经网络的关键，里面主要包括Init函数用来初始化神经网络的输入，隐藏，输出层数目；



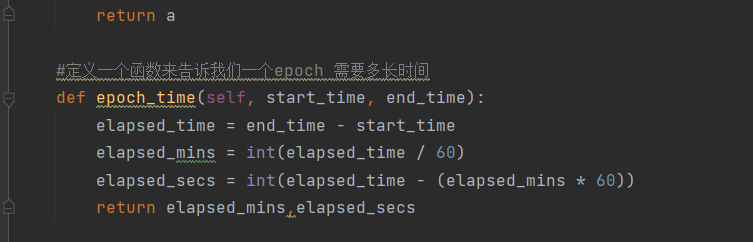
Fit函数用来训练网络模型，通过正向传播和反向传播的方式，不断更新权值和偏置来训练模型；



predic函数使用模型训练之后形成的权值文件来预测数字

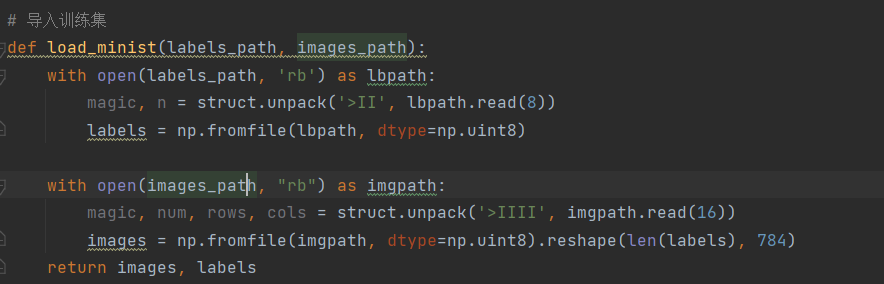


其他还有利用均分误差，方差，和计算每个epoch训练时长的函数就不一一介绍了



NN\_train.py 用于对建立的神经网络模型进行训练

这个文件只写了一个Load\_Minst函数用来从本地将Minst数据集内的数据读取出来，并对其进行标准化处理



此外就是调用fit函数将数据集的标签值和数据输入模型中，进行计算

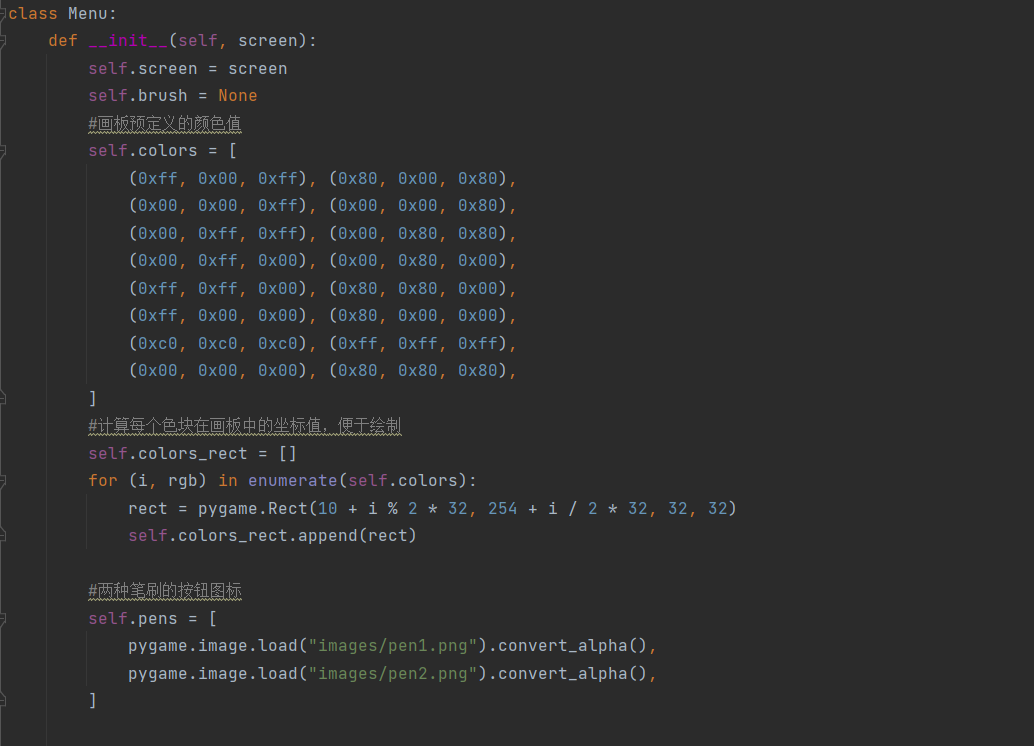
**Board.py 画板函数，主要用来设置一个画板用来进行手写数字识别系统的输入端**

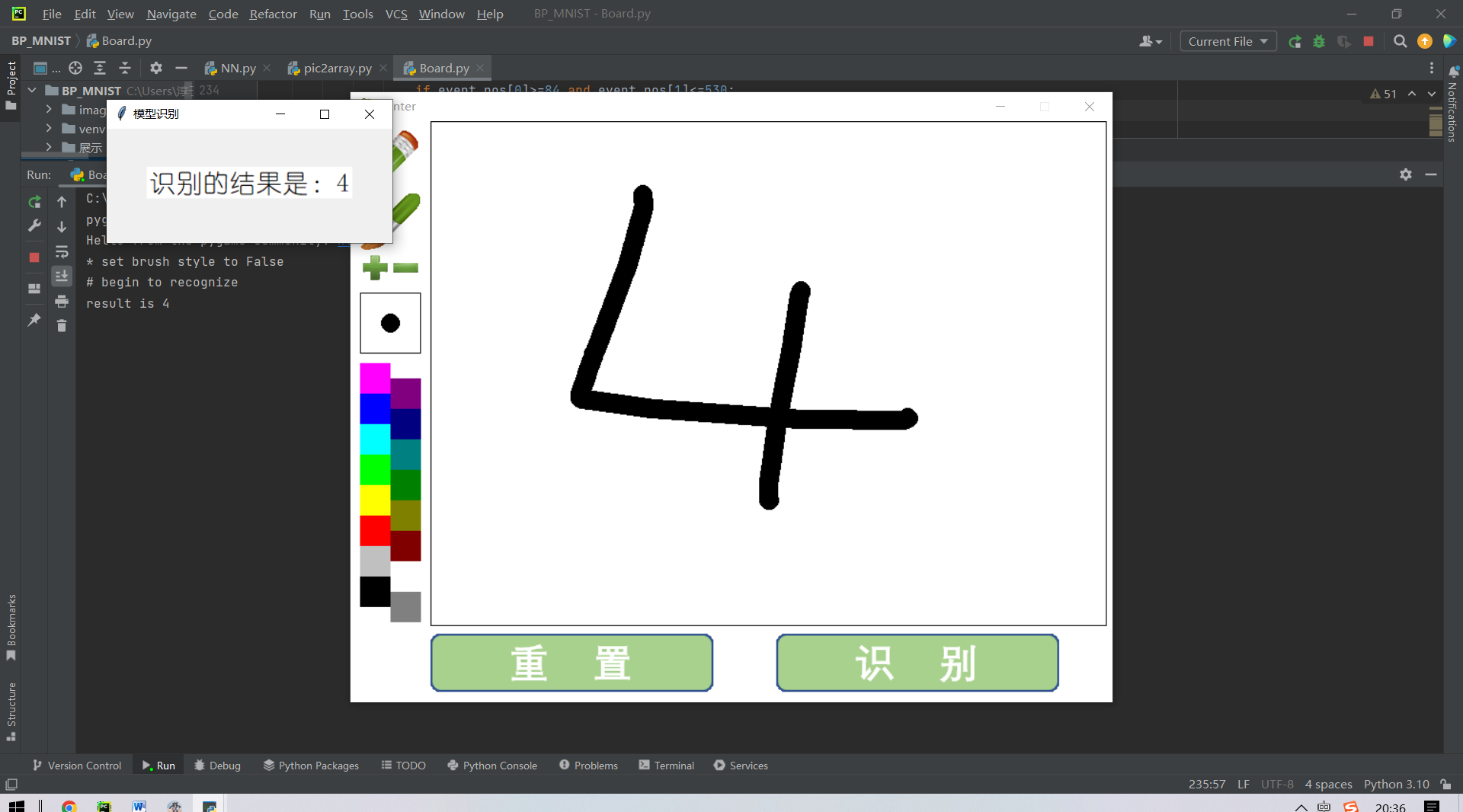
在此文件内写了二个类class Brush 和class Menu

Class Brush：里面可以设置画笔的颜色，尺寸，种类等等

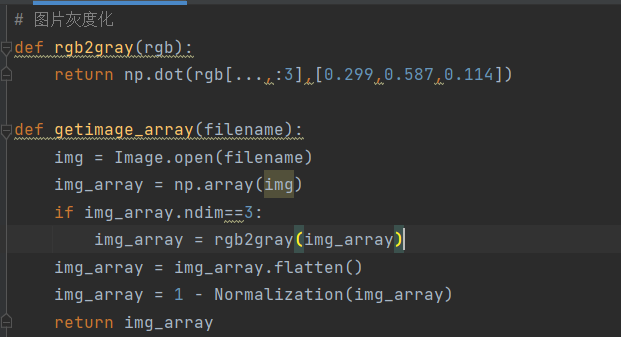


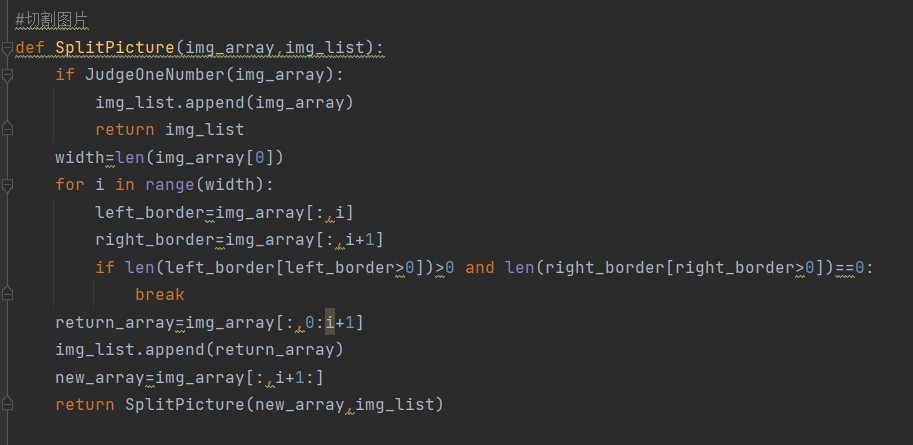
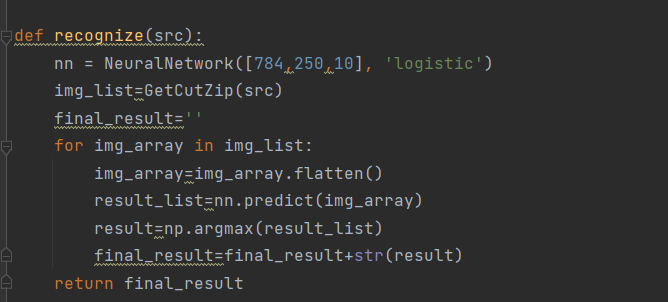
Class Menu: 里面设置了画板的大小，预定义的颜色，计算画笔此时的坐标等等



画板的样子：

Pic2array.py 分析函数，用来将普通二值图像进行预处理、分析、切割、压缩、扩展，使数字在28\*28的png格式的图片中居中显示，同时对于RGB图像要先进行灰度化、归一化和二值化，再通过与二值图像相同的操作步骤达到所需要求。图片的处理过程为：



**数据处理说明：**

所有的原始数据都来自于MINST数据库，分为四部分：训练集数据，训练集标签值，验证集数据，验证集标签值。

所有的数据都是标准的MINST数据库内数据：

1. 28\*28像素，黑底白字
2. 图像位于正中间，**且上下左右都至少留4个像素点**

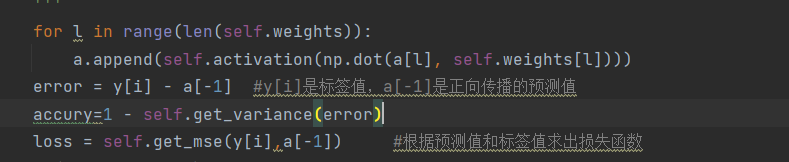
在NN\_train.py文件中先提取出本地文件夹内的MINST数据集数据，并对其进行初步处理

#读取数据集的函数，MINst数据集本身已经被下载到了本地文件夹内  
images, labels = load\_minist('train-labels.idx1-ubyte', 'train-images.idx3-ubyte')  
test\_images, test\_labels = load\_minist('t10k-labels.idx1-ubyte', 't10k-images.idx3-ubyte')  
  
#对标签值做标准化处理,例如：6转化为[0,0,0,0,0,0,1,0,0,0]  
labels = label\_binarizer(labels)  
test\_labels = label\_binarizer(test\_labels)

将原始数据输入训练集后进行进一步的处理，在训练集和验证集中，首先都是先将输入的训练数据转换为二维，在为其增加一个新的全为1的维度，用以方便进行接下来的运算

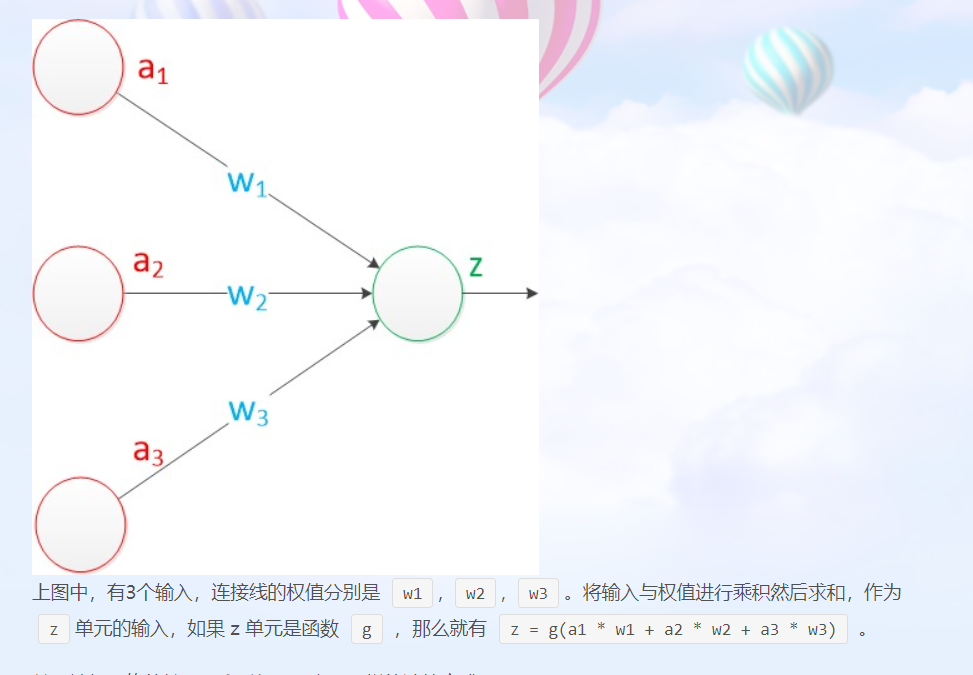
def fit(self, X, y, learning\_rate=0.15, epochs=10000):  
 X = np.atleast\_2d(X) #X：数据集,确认是二维，每行是一个实例，每个实例有一些特征值  
  
 temp = np.ones([X.shape[0], X.shape[1] + 1])  
 temp[:, 0:-1] = X  
 X = temp

在每一次epoch训练即：进行一次正向传播和反向传播：

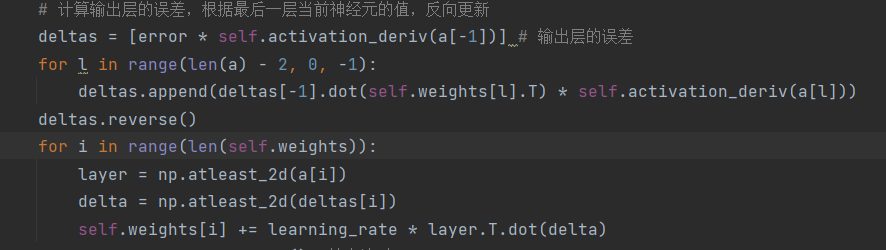


由于神经网络为三层结构：输入层，隐藏层，输出层

故此len(self.weights)为2，通过for循环来遍历输入层将输入层的数据和权值相卷积，与此图类似



进行反向传播的时候，根据前向传播计算的的误差，更新权值和偏置



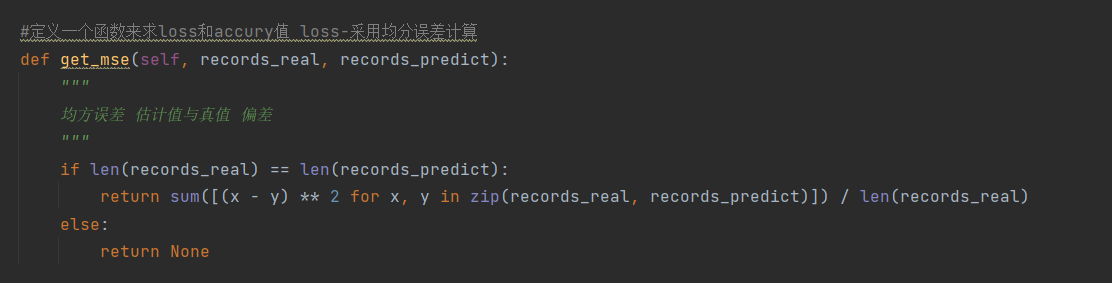
**网络模型（算法）说明：以 CNN 算法为例。**（报告中需包含该部分的代码截图）简要描述所使用的网络模型，该网络拥有多少个卷积层、池化层、全连接层，并介绍这些层的概况。

这个手写数字识别系统，采用的是最简单的三层模型，即输入层：隐藏层：输出层；[784:250:10]

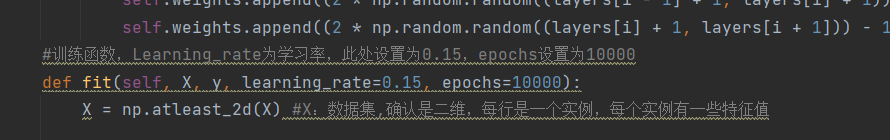
**训练模型说明：**

**由于本程序使用的是非常简单的网络模型，处理的数据量并不大，故此没有使用优化器。**

**损失函数通过计算fit（）函数内，预测值和标签值的均方误差来计算**



一次训练的数据量在10000，运行完需要1分钟左右



**程序训练结果说明分析：**简述所使用网络模型（算法）的优缺点；识别系统的最佳损失率和准确率；根据输出的混淆矩阵，分别计算各数字类别的准确率，并将结果汇总到表格中。

bp神经算法的优缺点：1）能够自适应、自主学习。这是BP算法的根本以及其优势所在，BP算法根据预设的参数更新规则，不断地调整神经网络中的参数，以达到最符合期望的输出。

（2）拥有较强的非线性映射能力。

（3）严谨的推导过程。误差的反向传播过程，采用的是已经非常成熟的链式法测，其推导过程严谨且科学。

（4）较强的泛化能力，即在BP算法训练结束之后，BP算法可以利用从原来知识中学到的知识解决新的问题。

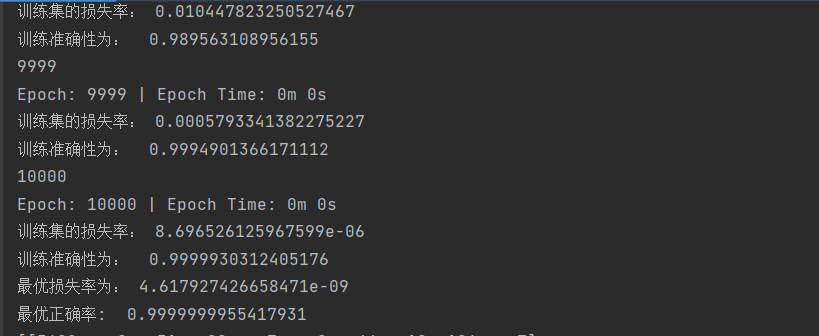
缺点：

（1）由于BP神经网络中的参数众多，每次都需要更新数量较多的阈值和权值，故会导致收敛速度过慢。

（2）网络中隐含层节点个数尚无明确的公式，传统方法需要不断地设置隐含层节点数进行试凑，根据网络误差结果确定最终隐含层节点个数。

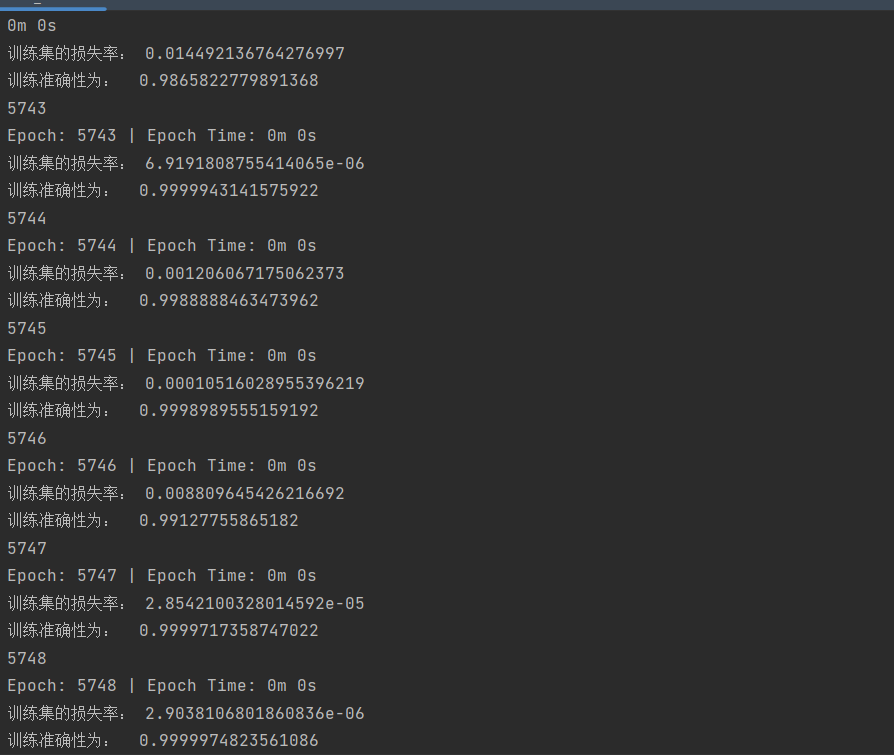
（3）从数学角度看，BP算法是一种速度较快的梯度下降算法，很容易陷入局部最小值的问题。当出现局部极小时，从表面上看，误差符合要求，但这时所得到的解并不一定是问题的真正解。所以BP算法是不完备的。

识别系统的最小损失率和最优正确率：

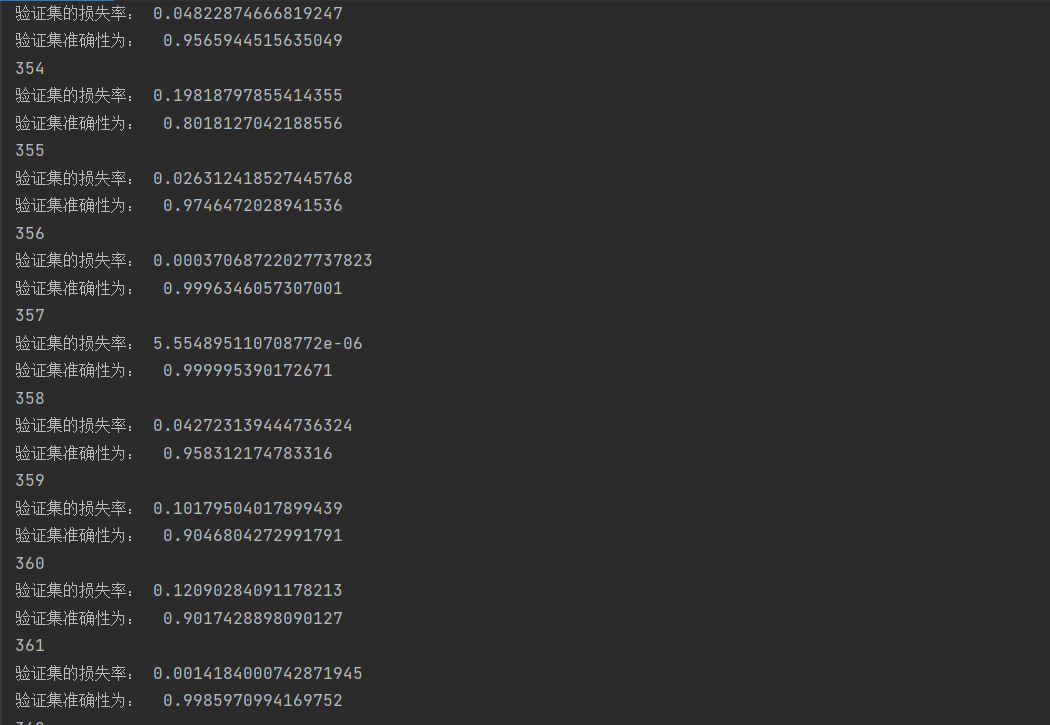


首先进行一次epoch为10000的运算，只显示一部分数据。关于epoch\_time的计算，在debug模式在能正常输出epoch\_time，但是在正常运行的时候却计算不出来

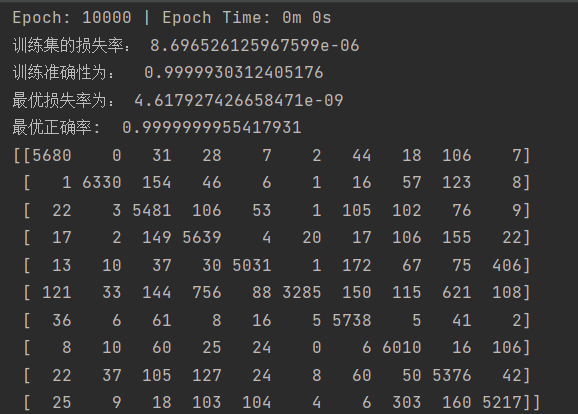
验证集的数据：



验证集的损失率和准确率：



输出的混淆矩阵为：



为了方便，只保留三位小数

此模型预测数字的成功率，从0-9如下

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0.955 | 0.982 | 0.878 | 0.821 | 0.939 | 0.987 | 0.908 | 0.879 | 0.796 | 0.880 |