

**《手写数字识别》**

题 目： **手写数字识别**

院 系： 数学与计算科学学院

专 业： 统计学

姓 名： 滕乾安

学 号： 1500720135

指导教师： 覃 义

日 期： 2018 年 01 月 06 日

目录

[1．简介 3](#_Toc29467)

[2．基础概要 3](#_Toc1815)

[2.1 MNIST数据集 3](#_Toc22826)

[2.1.1MNIST数据包 3](#_Toc24717)

[2.1.2 MNIST训练集 4](#_Toc6779)

[2.2 Tensorflow介绍 7](#_Toc24987)

[2.2.1 Tensorflow简介 7](#_Toc15857)

[2.2.2在Tensorflow中实现逻辑回归的步骤： 7](#_Toc8834)

[2.3 anaconda软件介绍 7](#_Toc7171)

[3． 算法研究 8](#_Toc15894)

[3.1 Softmax回归模型 8](#_Toc24459)

[3.2 softmax 回归算法的代价函数 9](#_Toc2514)

[4． 流程图 10](#_Toc11767)

[5． 代码解析 11](#_Toc11208)

[6．效果图 14](#_Toc14855)

[7．个人总结 16](#_Toc25825)

[8． 附录 17](#_Toc23938)

[9．参考文献： 22](#_Toc23107)

# 1．简介

对于手写数字识别任务，必然需要使用一个数据集进行处理，单纯地对形状的处理的算法正确率低并且编写复杂。经过搜索资料可以找到了MNIST手写数字数据集，该数据集包含60,000条和10,000条经过归一化并且带有标签的手写数字训练集和验证集。由于选用了其他人研究验证过的算法，于是我经过思考决定使用了TensorFlow的Python接口进行编程。程序包含界面部分与算法部分，算法部分首先采用TensorFlow自带的MNIST数据集导入功能，导入MNIST数据集。随后使用MNIST数据集对模型进行训练，训练次数可以在程序界面上调整。当然，为了解决课程设计的演示时间，我们已经调整了训练次数，并且预训练出了一个模型。这个模型对验证集的正确率可以达到0.9216。其次我实现了调用Windows画图程序的功能，使得我们的程序可以现场识别手写数字。但由于我的程序没有对现场手写的数据加上预处理（归一化、居中），故识别率并没有MNIST验证集中数据那么高，程序界面部分采用PyQt编写。

# 2．基础概要

## 2.1 MNIST数据集

### 2.1.1MNIST数据包

MNIST 数据集已经是一个被”嚼烂”了的数据集, 很多教程都会对它”下手”, 几乎成为一个 “典范”. 不过有些人可能对它还不是很了解, 下面来介绍一下.

MNIST 数据集可在MNIST官方网站获取, 它包含了四个部分:

**Training set images**: train-images-idx3-ubyte.gz (9.9 MB, 解压后 47 MB, 包含 60,000 个样本)

**Training set labels**: train-labels-idx1-ubyte.gz (29 KB, 解压后 60 KB, 包含 60,000 个标签)

**Test set images**: t10k-images-idx3-ubyte.gz (1.6 MB, 解压后 7.8 MB, 包含 10,000 个样本)

**Test set labels**: t10k-labels-idx1-ubyte.gz (5KB, 解压后 10 KB, 包含 10,000 个标签)

MNIST 数据集来自美国国家标准与技术研究所, **National Institute of Standards and Technology (NIST)**. 训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生, 50% 来自人口普查局 (the Census Bureau) 的工作人员. 测试集(test set) 也是同样比例的手写数字数据.

不妨新建一个文件夹 – mnist, 将数据集下载到 mnist 以后, 解压即可:

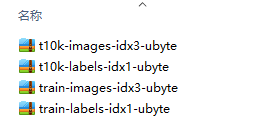


图 1-1

图片是以字节的形式进行存储, 我们需要把它们读取到 NumPy array 中, 以便训练和测试算法。

### 2.1.2 MNIST训练集

下载下来的数据集被分成两部分：60000行的训练数据集（mnist.train）和10000行的测试数据集（mnist.test）。这样的切分很重要，在机器学习模型设计时必须有一个单独的测试数据集不用于训练而是用来评估这个模型的性能，从而更加容易把设计的模型推广到其他数据集上（泛化）。

正如前面提到的一样，每一个MNIST数据单元有两部分组成：一张包含手写数字的图片和一个对应的标签。我们把这些图片设为“xs”，把这些标签设为“ys”。训练数据集和测试数据集都包含xs和ys，比如训练数据集的图片是 mnist.train.images ，训练数据集的标签是 mnist.train.labels。

每一张图片包含28X28个像素点。我们可以用一个数字数组来表示这张图片：

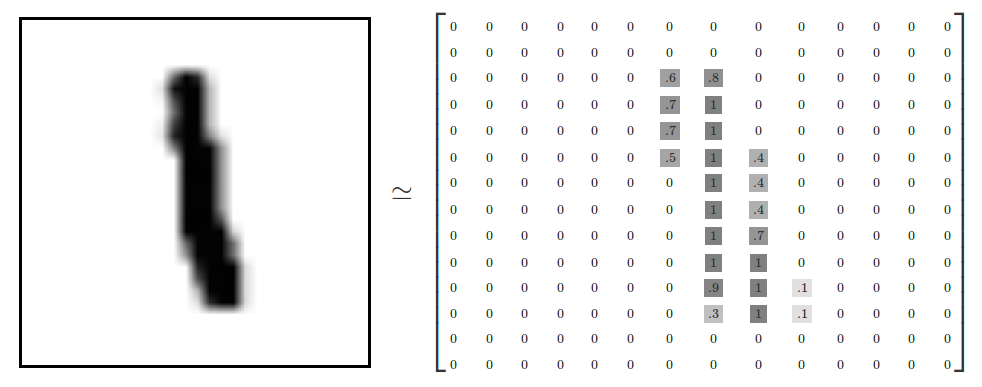


图 2-1

我们把这个数组展开成一个向量，长度是 28x28 = 784。如何展开这个数组（数字间的顺序）不重要，只要保持各个图片采用相同的方式展开。从这个角度来看，MNIST数据集的图片就是在784维向量空间里面的点, 并且拥有比较复杂的结构 (提醒: 此类数据的可视化是计算密集型的)。

展平图片的数字数组会丢失图片的二维结构信息。这显然是不理想的，最优秀的计算机视觉方法会挖掘并利用这些结构信息，我们会在后续教程中介绍。但是在这个教程中我们忽略这些结构，所介绍的简单数学模型，softmax回归(softmax regression)，不会利用这些结构信息。

因此，在MNIST训练数据集中，mnist.train.images 是一个形状为 [60000, 784] 的张量，第一个维度数字用来索引图片，第二个维度数字用来索引每张图片中的像素点。在此张量里的每一个元素，都表示某张图片里的某个像素的强度值，值介于0和1之间。

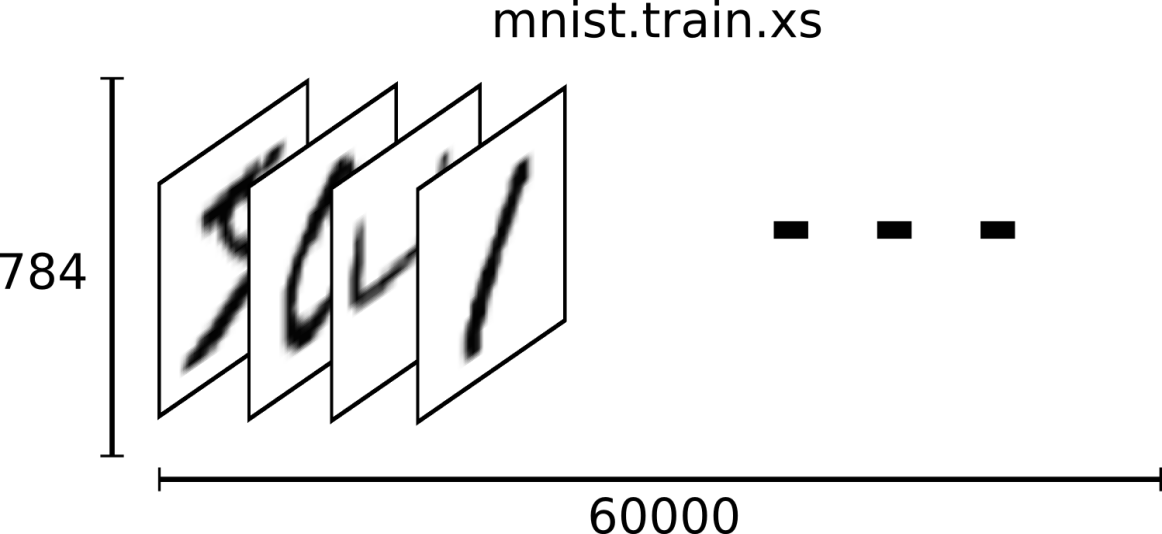


图 2-2

相对应的MNIST数据集的标签是介于0到9的数字，用来描述给定图片里表示的数字。为了用于这个教程，我们使标签数据是"one-hot vectors"。 一个one-hot向量除了某一位的数字是1以外其余各维度数字都是0。所以在此教程中，数字n将表示成一个只有在第n维度（从0开始）数字为1的10维向量。比如，标签0将表示成([1,0,0,0,0,0,0,0,0,0,0])。因此， mnist.train.labels 是一个 [60000, 10] 的数字矩阵。

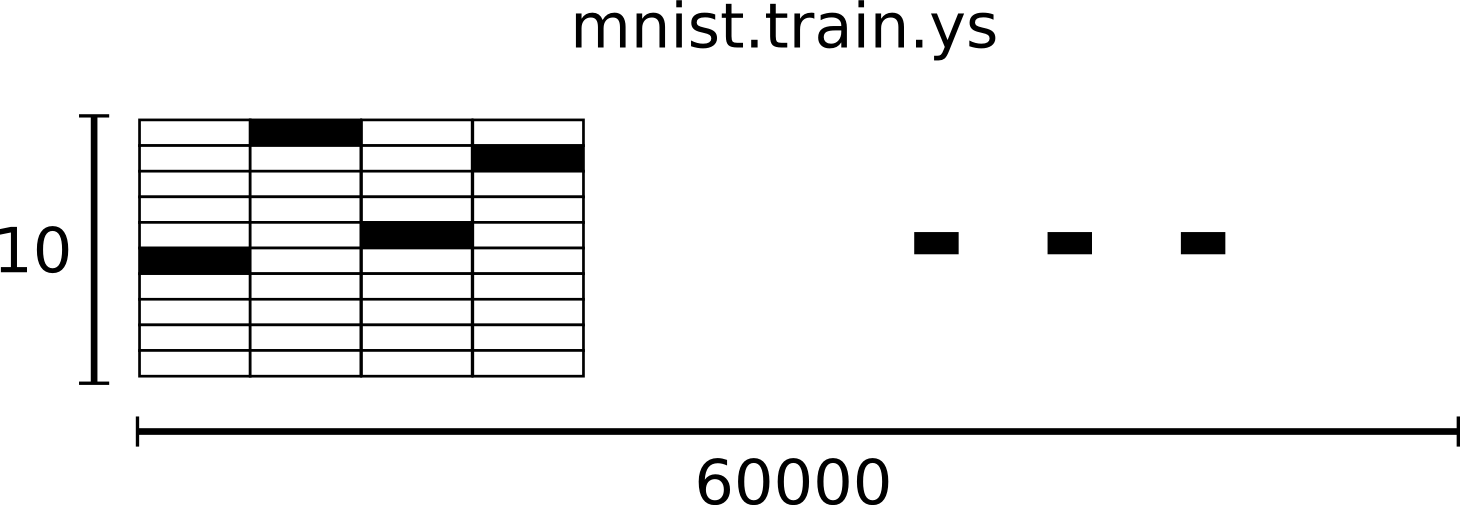


图 2-3

## 2.2 Tensorflow介绍

### 2.2.1 Tensorflow简介

TensorFlow 是一个编程系统, 使用图来表示计算任务. 图中的节点被称之为 op (operation 的缩写). 一个 op 获得 0 个或多个 Tensor, 执行计算, 产生 0 个或多个 Tensor. 每个 Tensor 是一个类型化的多维数组. 例如, 你可以将一小组图像集表示为一个四维浮点数数组, 这四个维度分别是 [batch, height, width, channels].

一个 TensorFlow 图描述了计算的过程. 为了进行计算, 图必须在 会话 里被启动. 会话 将图的 op 分发到诸如 CPU 或 GPU 之类的 设备 上, 同时提供执行 op 的方法. 这些方法执行后, 将产生的 tensor 返回. 在 Python 语言中, 返回的 tensor 是 numpy ndarray 对象。

### 2.2.2在Tensorflow中实现逻辑回归的步骤：

1）定义算法公式；

2）定义函数，选择优化器优化函数；

3）使用输入训练集数据进行迭代训练；

4）在验证集或测试集上评估模型的准确率。

## 2.3 anaconda软件介绍

anaconda是一个用于科学计算的Python发行版，支持 Linux, Mac, Windows系统，提供了包管理与环境管理的功能，可以很方便地解决多版本python并存、切换以及各种第三方包安装问题。Anaconda利用工具/命令conda来进行package和environment的管理，并且已经包含了Python和相关的配套工具。

这里先解释下conda、anaconda这些概念的差别。conda可以理解为一个工具，也是一个可执行命令，其核心功能是**包管理**与**环境管理**。包管理与pip的使用类似，环境管理则允许用户方便地安装不同版本的python并可以快速切换。Anaconda则是一个打包的集合，里面预装好了conda、某个版本的python、众多packages、科学计算工具等等，所以也称为Python的一种发行版。其实还有Miniconda，顾名思义，它只包含最基本的内容——python与conda，以及相关的必须依赖项，对于空间要求严格的用户，Miniconda是一种选择。

# 3． 算法研究

## 3.1 Softmax回归模型

Softmax回归模型是logistic回归模型在多分类问题上的推广，在多分类问题中，类标签 可以取两个以上的值（相应地，logistic回归的类标签仅有0、1两个取值）。 Softmax回归模型对于诸如手写数字分类等问题是很有用的，对于该问题，类标签可以取0-9十种不同的数字的值。Softmax回归是有监督的。（MNIST 是一个手写数字识别库，由NYU 的Yann LeCun 等人维护。http://yann.lecun.com/exdb/mnist/ ）

在类似的 logistic 回归中，训练集由 个已标记的样本构成： ，其中输入特征。（本文对符号的约定如下：特征向量 的维度为，其中 对应截距项 。） 由于 logistic 回归是针对二分类问题的，因此类标记。假设函数(hypothesis function) 如下：

 (公式 3-1)

我们将训练模型参数，使其能够最小化代价函数 ：

（公式 3-2）

在 softmax回归中，我们解决的是多分类问题（相对于 logistic 回归解决的二分类问题），类标 可以取 个不同的值（而不是2个）。因此，对于训练集，我们有 。（注意此处的类别下标从 1 开始，而不是 0）。在本次课程设计中，我们有个不同的类别。

对于给定的测试输入 ，我们想用假设函数针对每一个类别j估算出概率值 。也就是说，我们想要估计的每一种数字出现的概率。因此，我们的假设函数将要输出一个维的向量（向量元素的和为1）来表示这 个估计的概率值。 具体地说，我们的假设函数 形式如下：

（公式 3-3）

其中是模型的参数。请注意这一项对概率分布进行归一化，使得所有概率之和为1。本次课程设计使用的TensorFlow输出的矩阵就没有对其进行归一化处理，需要手工编写代码实现。

为了方便起见，我们同样使用符号来表示全部的模型参数。在实现Softmax回归时，将用一个的矩阵来表示会很方便，该矩阵是将按行罗列起来得到的，如下所示：

（公式 3-4）

## 3.2 softmax 回归算法的代价函数

在下面的公式中，是示性函数，其取值规则为：

举例来说，表达式的值为1 ，的值为 0。我们的代价函数为：

（公式 3-5）

在Softmax回归中将分类为类别的概率为：

（公式 3-6）

对于的最小化问题，我们使用梯度下降法。经过求导，我们得到梯度公式如下：

（公式 3-7）

有了上面的偏导数公式以后，我们就可以将它代入到梯度下降法等算法中，来最小化。 例如，在梯度下降法的标准实现中，每一次迭代需要进行如下更新: ()。

# 4． 流程图

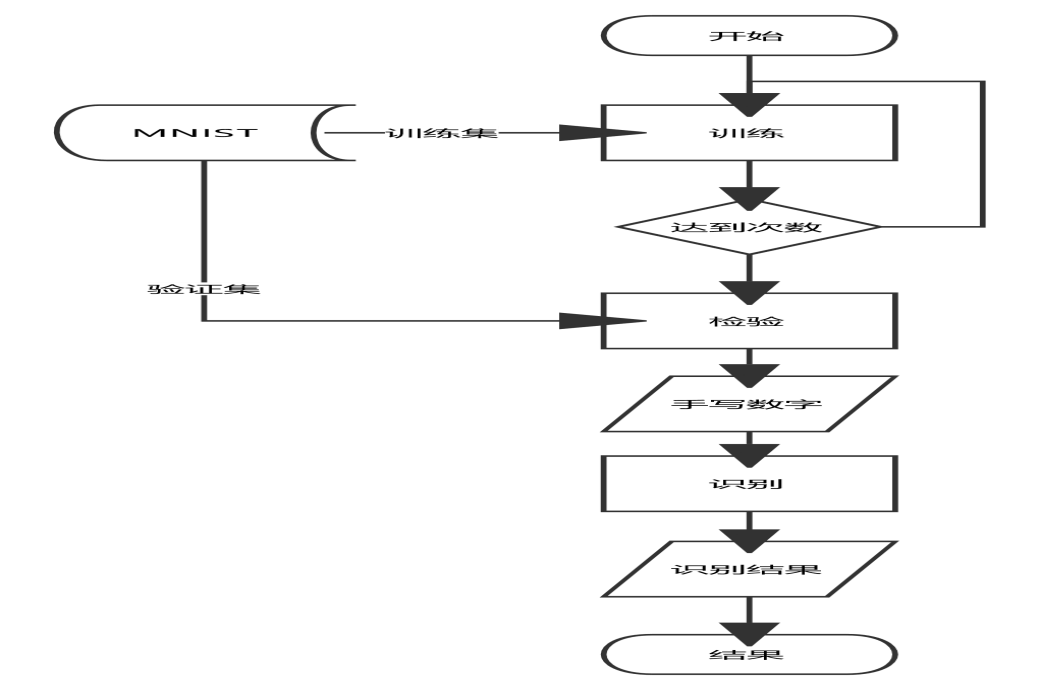


图 3-1

# 5． 代码解析

1）这里使用TensorFlow实现一个简单的神经网络，使用的是MNIST数据集

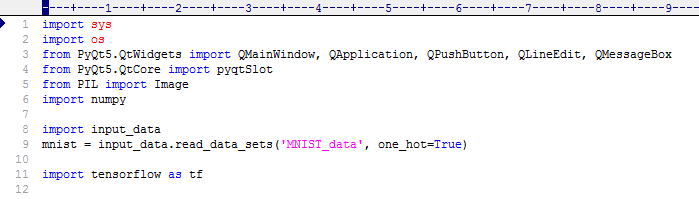


图 4-1

2）28\*28的灰度图，像素个数784 ，是10分类问题。定义符号变量 x （数据）和 y\_（标签），其中：x 是一个占位符，我们在TensorFlow实际运行计算时再输入这个值。参数[ None,784 ]中的 None 表示我们接受输入任意数量的图像，784 表示每一张图展开成 784 维的向量。定义变量 w 和 b，其中：一个变量代表一个可修改的张量，它们可以用于计算输入值，也可以在计算中被修改。

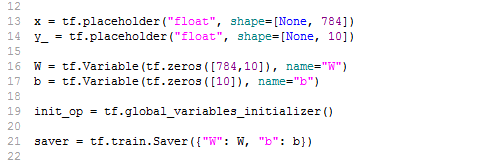


图 4-2

3）UI界面设置

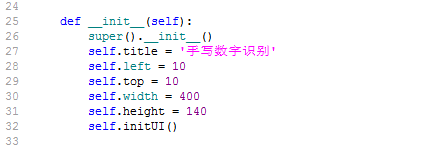


图 4-3

4）按钮设置

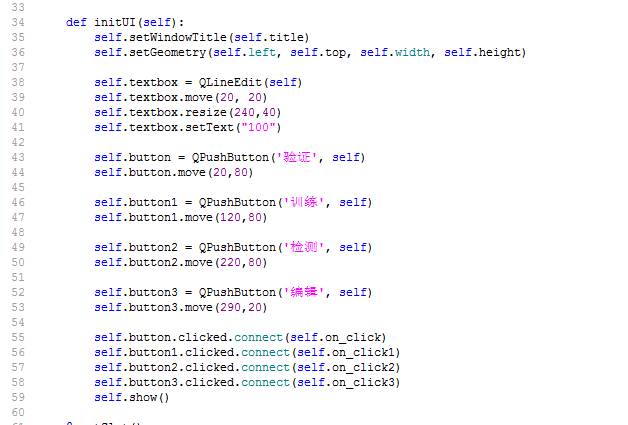


图 4-4

5）训练网络，首先要对所有变量初始化，之后，每次从训练集中随机去除100个样本训练网络，调用了训练并存储好的模型参数，以及MNIST验证集对模型的性能进行检验。

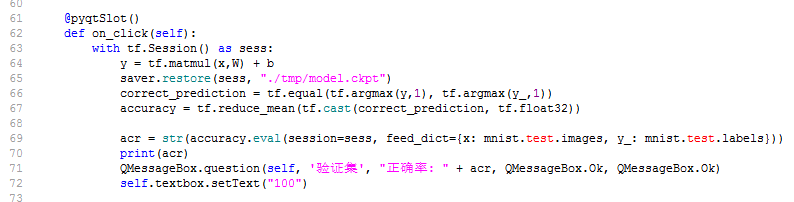


图 4-5

6）调用了MNIST的训练集，对模型参数进行训练。

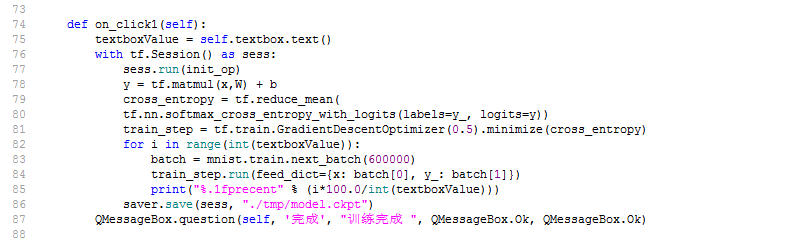


图 4-6

7）调用了Windows画图工具，对图片进行编辑并根据存储好的模型，对图片进行识别。

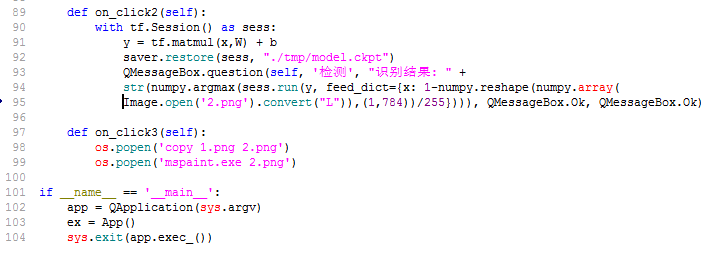


图 4-7

# 6．效果图

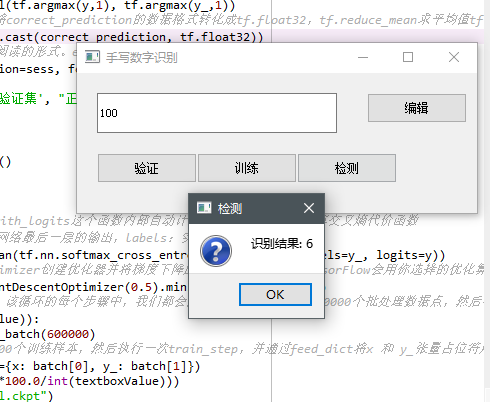


图 5-1

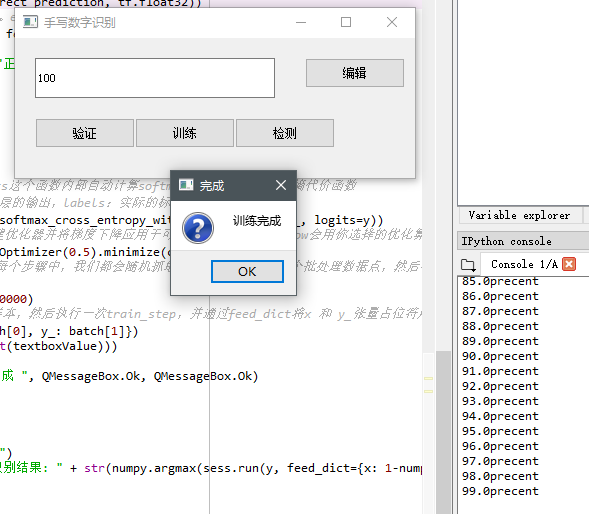


图 5-2

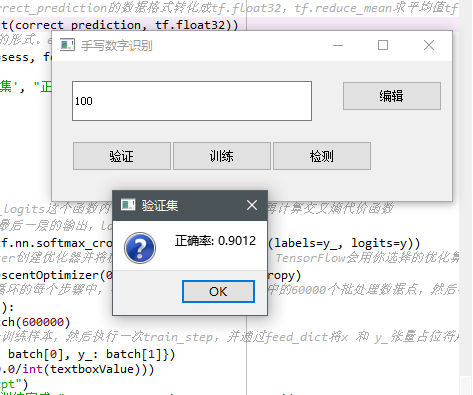


图 5-3

# 7．个人总结

当学习一门新的编程语言的时候，我们总是以输出“hello word”作为学习这门编程语言的开始，表示我们开启了这门编程语言的大门。而在机器学习的领域中，识别手写数字就像输出“hello word”一样作为开启了机器学习的大门。MNIST相当于机器学习界的Hello World。在机器学习训练模型的过程中，数据的预处理占有非常重要的作用，指从数据源获取数据之后直接处理数据。一般，在使用机器学习训练某一个模型的时候，需要将原数据经过一定的处理之后，再用于模型的训练。在手写数字识别的模型中，输入的数据是一张的图片，为了简化处理，将每张28\*28的图片装换成了一个784维的向量，很显然在转换的过程中，不得不丢弃了图片的二维结构信息，如果想要不丢弃图片的二维结构信息，可以使用卷积，在这个例子中，做了简化处理。手写数字识别的模型中，最终的目的是将一张图片与一个数字（0-9）进行对应，其实也就是将这个问题转换成了一个多分类的问题进行处理。在机器学习中，对于多分类问题，可以使用softmax回归进行处理。除此之外，Tensorflow进行深度学习和人工智能具有开发简单，建模速度快，准确度高的优点，所以使用了TensorFlow的Python接口进行编程。程序包含界面部分与算法部分，算法部分首先采用TensorFlow自带的MNIST数据集导入功能，导入MNIST数据集。随后使用MNIST数据集对模型进行训练，训练次数可以在程序界面上调整。当然，为了解决课程设计的演示时间，我们已经调整了训练次数，并且预训练出了一个模型。这个模型对验证集的正确率可以达到0.9216。其次我实现了调用Windows画图程序的功能，使得我们的程序可以现场识别手写数字。但由于我的程序没有对现场手写的数据加上预处理（归一化、居中），故识别率并没有MNIST验证集中数据那么高，程序界面部分采用PyQt编写。

# 8． 附录

import sys

import os

from PyQt5.QtWidgets import QMainWindow, QApplication, QPushButton, QLineEdit, QMessageBox

from PyQt5.QtCore import pyqtSlot

from PIL import Image

import numpy

import input\_data

mnist = input\_data.read\_data\_sets('MNIST\_data', one\_hot=True)

import tensorflow as tf

x = tf.placeholder("float", shape=[None, 784])

y\_ = tf.placeholder("float", shape=[None, 10])

W = tf.Variable(tf.zeros([784,10]), name="W")

b = tf.Variable(tf.zeros([10]), name="b")

init\_op = tf.global\_variables\_initializer()

saver = tf.train.Saver({"W": W, "b": b})

class App(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title = '手写数字识别'

self.left = 10

self.top = 10

self.width = 400

self.height = 140

self.initUI()

def initUI(self):

self.setWindowTitle(self.title)

self.setGeometry(self.left, self.top, self.width, self.height)

self.textbox = QLineEdit(self)

self.textbox.move(20, 20)

self.textbox.resize(240,40)

self.textbox.setText("100")

self.button = QPushButton('验证', self)

self.button.move(20,80)

self.button1 = QPushButton('训练', self)

self.button1.move(120,80)

self.button2 = QPushButton('检测', self)

self.button2.move(220,80)

self.button3 = QPushButton('编辑', self)

self.button3.move(290,20)

self.button.clicked.connect(self.on\_click)

self.button1.clicked.connect(self.on\_click1)

self.button2.clicked.connect(self.on\_click2)

self.button3.clicked.connect(self.on\_click3)

self.show()

@pyqtSlot()

def on\_click(self):

with tf.Session() as sess:

y = tf.matmul(x,W) + b

saver.restore(sess, "./tmp/model.ckpt")

correct\_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y\_,1))

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

acr = str(accuracy.eval(session=sess, feed\_dict={x: mnist.test.images, y\_: mnist.test.labels}))

print(acr)

QMessageBox.question(self, '验证集', "正确率: " + acr, QMessageBox.Ok, QMessageBox.Ok)

self.textbox.setText("100")

def on\_click1(self):

textboxValue = self.textbox.text()

with tf.Session() as sess:

sess.run(init\_op)

y = tf.matmul(x,W) + b

cross\_entropy = tf.reduce\_mean(

tf.nn.softmax\_cross\_entropy\_with\_logits(labels=y\_, logits=y))

train\_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross\_entropy)

for i in range(int(textboxValue)):

batch = mnist.train.next\_batch(600000)

train\_step.run(feed\_dict={x: batch[0], y\_: batch[1]})

print("%.1fprecent" % (i\*100.0/int(textboxValue)))

saver.save(sess, "./tmp/model.ckpt")

QMessageBox.question(self, '完成', "训练完成 ", QMessageBox.Ok, QMessageBox.Ok)

def on\_click2(self):

with tf.Session() as sess:

y = tf.matmul(x,W) + b

saver.restore(sess, "./tmp/model.ckpt")

QMessageBox.question(self, '检测', "识别结果: " + str(numpy.argmax(sess.run(y, feed\_dict={x: 1-numpy.reshape(numpy.array(Image.open('2.png').convert("L")),(1,784))/255}))), QMessageBox.Ok, QMessageBox.Ok)

def on\_click3(self):

os.popen('copy 1.png 2.png')

os.popen('mspaint.exe 2.png')

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = App()

sys.exit(app.exec\_())

# 9．参考文献：

[1] Yann LeCun. THE MNIST DATABASE

of handwritten digits [EB/OL]. http://yann.lecun.com/exdb/mnist/,

[2]Google Inc. Tensor Flow[CP]. http://www.tensorflow.org/

[3] Andrew Ng. UFLDL Tutorial [DB/CD]. Stanford University，2012