# 设计内容分析与功能描述

## 1.1功能描述

使用Python实现对手写数字的识别工作，通过使用windows上的画图软件绘制一个大小是28x28像素的数字图像，图像的背景色是黑色，数字的颜色是白色，将该绘制的图像作为输入，经过训练好的模型识别所画的数字。

## 1.2 设计内容分析

手写数字的识别可以分成两大板块：一、手写数字模型的训练；二、手写数字的识别。其中最为关键的环节是手写数字模型的训练。下图为程序设计的总体流程。



本次选取使用的模型是多元线性回归模型。手写数字有10中，分别是0~9，所以可以将该问题视为一个多分类问题。

# 算法基本原理

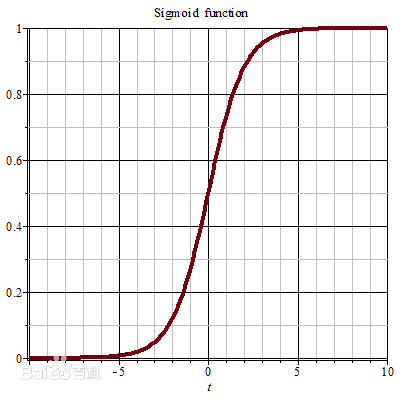
一个问题含有多个变量因素是，我们把包括两个或两个以上自变量的回归称为多元线性回归。多元线性回归的基本原理和基本计算过程与一元线性回归相同，不同的地方在于进行计算的变量个数是非常多的。由于在进行手写数字识别时，存在着10种数字，所以在进行标记训练数据集的标签时，通过热编码的方式来进行标签标记。下图展示了热编码的效果。



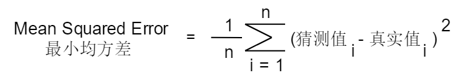
多元线性回归使用的是如下的公式进行计算的；

但是经过计算后y的值将会不在0~1内，而同时我们在进行标签标记时使用的是热编码，所以进行线性运算后需要将计算的结果转换到0~1之间，所以通过如下的公式：

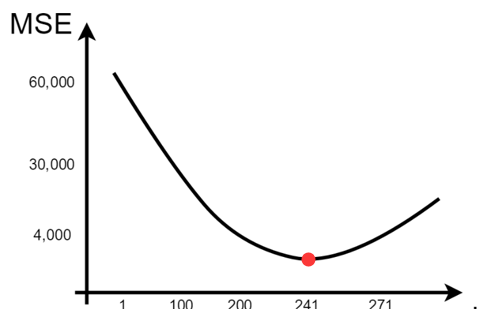
该公式被称为sigmoid函数，其曲线当y的值越大时，y1的结果越接近1，当y的值越小时，y1的结果越接近0。下图为其函数曲线图。



利用上述的原理，进行一次训练，之后不断的进行运行，但是训练到何种程度算是结束训练呢，这就需要使用最小均方误差函数，使预测值与真实值之间的误差达到最小。下面的公式为最小均方误差函数表达式：



在进行迭代运算时，使用梯度下降的方法不断的对线性回归公式中的自变量参数m1...mn和b进行迭代更新，使最小均方误差函数达到最小。梯度计算时可以将最小均方误差进行求导，从而得到当前关于m1...mn和b的变化率，利用此变化率，来不断的更新当前的m1...mn和b。如果我们将迭代的过程中最小均方误差的变化曲线如下图所示，图像红点所对应的地方的m1...mn和b值即为我们期望的。



所以根据以上的原理，进行训练的具体流程如下图所示：



得到训练后的权重后，在进行预测时通过使用公式

来计算该带预测的手写数字是0~9中的那个概率大，从而概率最大的就是预测的值。

# 各个组成模块介绍及代码

## 3.1手写数字训练代码

**import** numpy **as** np  
  
  
feature = np.loadtxt(**"train\_image.csv"**, delimiter=**","**, max\_rows=6000) / 255  
featureMatrix = np.append(feature, np.ones(shape=(len(feature), 1)), axis=1)  
  
weightm = np.ones(shape=(feature.shape[1], 10))  
weightb = np.ones(shape=(1, 10))  
weight = np.r\_[weightm, weightb]  
learningrate = 0.0001  
  
label = np.loadtxt(**"train\_label\_hotencoding.csv"**, delimiter=**","**, max\_rows=6000)  
  
  
**def** grandientDecent():  
 predict2sigmod = 1/ (1 + np.exp(-np.dot(featureMatrix, weight)))  
 temp\_slop = np.dot(featureMatrix.T, predict2sigmod - label)  
 **return** temp\_slop  
  
  
**def** train():  
 **global** weight  
 **for** i **in** range(1, 50000):  
 slopmb = grandientDecent()  
 weight -= slopmb \* learningrate  
 **return** weight  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 myweight = train()  
 np.savetxt(**"myweight1.csv"**,myweight, fmt=**"%f"**, delimiter=**","**)

当把训练程序运行完成后会得到线性回归的权重值，并将数据保存到myweight1.csv文件中去。在训练程序中train\_image.csv是训练集数据，train\_label\_hotencoding.csv是对应与训练数据集的标签。

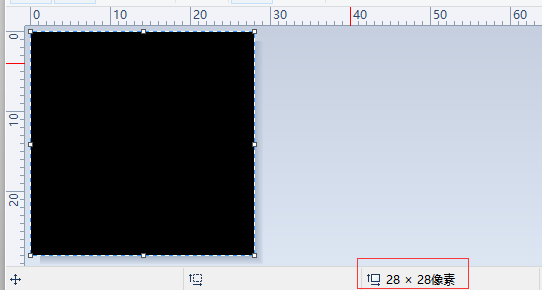
## 3.2预测程序

**import** numpy **as** np  
**from** PIL **import** Image  
  
image = Image.open(**"D:/8.bmp"**)  
t = np.array(image) / 255  
  
  
  
testfeature = t.reshape(1,784)  
  
  
np.set\_printoptions(threshold=np.inf, suppress=**True**)  
  
  
myweight = np.loadtxt(**"myweight1.csv"**, delimiter=**","**)  
  
testfeatureMatrix = np.append(testfeature, np.ones(shape=(len(testfeature), 1)), axis=1)  
mypredict = np.dot(testfeatureMatrix, myweight)  
expmpre = np.exp(mypredict)  
expsum = np.sum(expmpre, axis=1)  
  
**for** i **in** range(len(testfeature)):  
 expmpre[i, :] = expmpre[i, :] / expsum[i]  
 pre = np.argmax(expmpre[i, :])  
 print(pre)  
  
print(**"-"** \* 20)

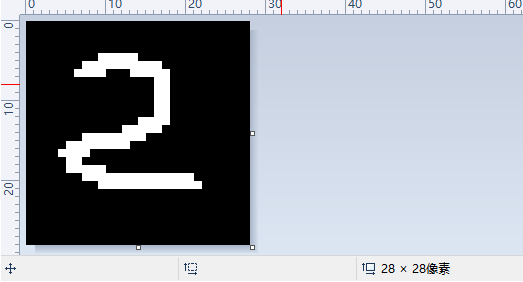
程序首先是对我们手写的数字进行处理，将其转换成与训练集相同格式的数据，之后是加载线性回归的权重模型myweight1.csv

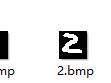
# 运行效果与分析

画图软件画图界面里的要求如下图：必须是黑色的底白色的字，并且保存的图片的大小是28x28的，这样就与训练的数据集向吻合。

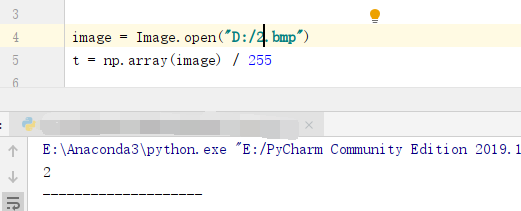


手写数字后识别界面：





其识别效果如下图：



但是在进行测试的时候仍然会存在着一定的问题，经过查询相关的资料发现是由于进行训练时出现了过拟合现象，所以仍需继续学习。

# 设计收获及心得

这次课程设计，我通过实践锻炼了编程能力，学习了numpy库的使用方法，学会了如何将图片转换成为一个一维数组。学习逻辑回归的原理和程序的实现方法，学习了热编码的方式。在编程的过程中，不可避免的会遇到错误提示，通过查找博客，论坛等其中使用较多的是CSDN，所以在此表示感谢。用别人给的解决办法进行尝试，可以逐渐解决这些错误，最终实现整个程序的正常运行与结果的正常显示。通过编写程序的过程中不断的加深了对多元线性回归的理解，该方法可以说在进行判断的时候同样是使用概率大小的方法进行判断的。

# 参考文献

[1] 代亮,许宏科,陈婷,钱超,梁殿鹏. 基于MapReduce的多元线性回归预测模型[J]. 计算机应用,2014,07:1862-1866.