实验报告-ex3

编程软件：MATLAB，notepad++

编程环境：Windows10

作业任务：本次实验旨在了解用神经网络算法处理one-vs-all logistic回归的问题。有ex3和ex2\_nn两个函数，函数中分别套用了相应的函数，所以需要根据这两个目录函数的指示和实验手册，完成对应的四个函数。

1 Multi-class Classification

这次实验很有趣，是利用神经网络识别0-9的手写体数字

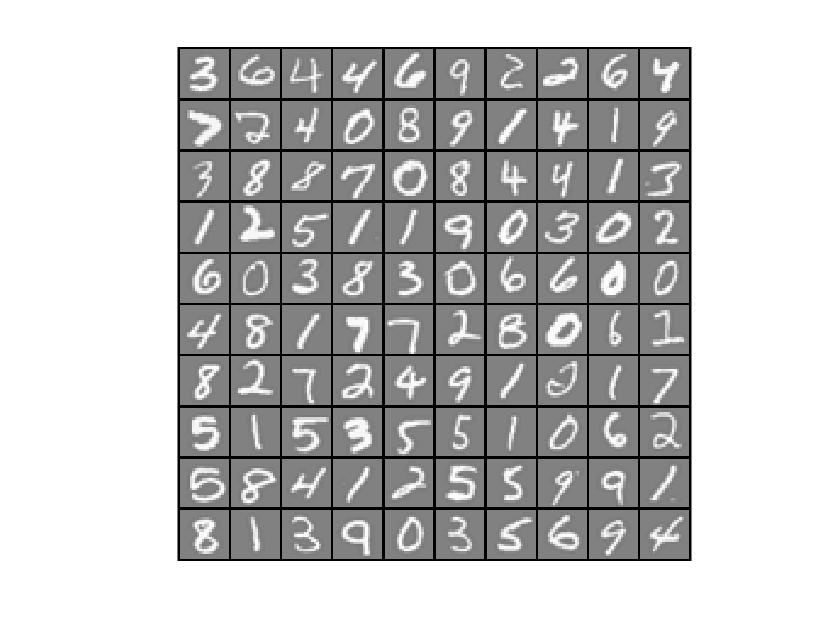
1.1 Dataset

首先是数据集。本次实验的training set和其他实验不同，每个数据不再是有特定意义的数字，而是几组数据合在一起构成一个图像的抽象表示。而且通过ex3函数可知，图像的像素是20×20，每个像素由一个浮点数来表示该位置的灰度。所以这个20×20的像素就转化为1×400的向量，而每一行代表一个training example，所以得到的X是一个5000×400的矩阵。

而结果向量y就是一个5000×1的向量，每个元素代表该手写数字的值。由于MATLAB中没有10，故将0映射到10。

1.2 Visualizing the data

这一部分随机选取了X中的100行（即5000个手写数字中任意100个），调用displayData函数，画出原来图像的样子。（该函数已给出）

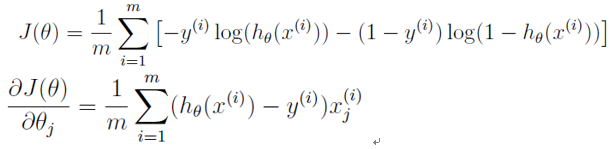


1.3 Vectorizing Logistic Regression

这一部分暂时没有用到神经网络的算法，而是利用ex2中实现的logistic回归算法来分别计算这10个结果的分类器，而为了提高效率，向量化就显得尤为重要。

1.3.1 Vectorizing the cost function

1.3.2 Vectorizing the gradient

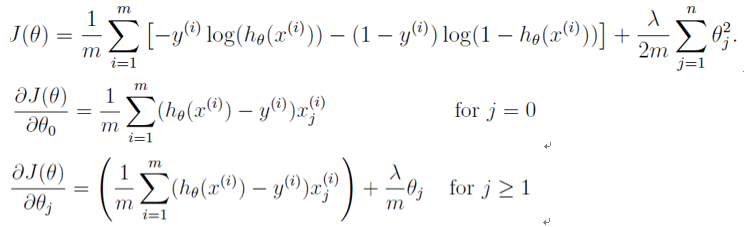


这两部分是实现没有正则化的cost function和其对θ的偏导数。

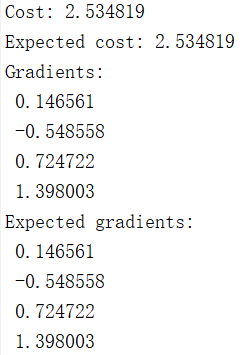
由公式可知，要想得到J和偏导数，需要先完善hθ(x)。与ex2一致，核心即实现sigmoid函数，而本次实验已经给出。则J实际上就是套用公式即可（此处应该注意向量化），而偏导数也已经在ex1中用向量化实现。

1.3.3 Vectorizing regularized logistic regression

这一部分是实现有正则化的logistic回归。



一开始出错是因为在计算temp = X' \* (predictions-y);时将X与predictions的前后位置颠倒了，导致产生的偏导数数量不正常，修改后正确。



1.4 One-vs-all Classification

这一部分要求实现多元分类（one-vs-all），需要完成的是对每一个K种类都要实现带正则化的logistic回归，在本例中K=10。

与前面一对一的分类不同的是，这里返回的Θ是K×(N+1)的矩阵，矩阵的每一行对应一种分类的参数，实现该矩阵可以对行数K进行循环，单独计算每一个向量θ。除此之外，每一个种类还需要设置一个m维的向量y，每个元素代表当前位置的training set是否在该种类中，此时需要用到逻辑数组。

实验手册中还提到要用到fmincg函数，该函数与fminunc类似，也是代替gradient descent计算最小参数的函数，并且fmincg可以用来处理更多的数据。这里要注意，fmincg函数得到的theta是列向量，而我们需要的Θ是每一行对应一个theta，故需要的是行向量，因此在每次循环最后需要加上all\_theta(i,:) = theta';来转置。

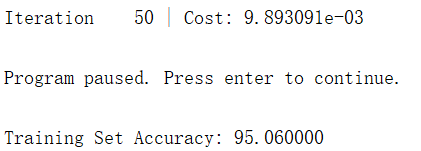
在这里发现了前面实验出现的问题。前面我是将θ0单独考虑的，然后J是减去第一个位置的值，这样子对于θ是向量时是没问题的，但是像这里面θ变成了矩阵，则需要每行的第一个都不应该考虑，此时前面那种做法就不再正确了。于是进行了修改，新设立了一个theta1，将传入theta的第一行置零，其余不变，此时后面就不需要对θ0单独考虑了。

修改后正确

1.4.1 One-vs-all Prediction

这个部分要求完善对某个手写数字图片的识别。传入Θ和待识别的X，返回一个m×1的向量，其中每一项为该图片预测的结果数字。

思路较为简单，只需要计算出所有X在当前Θ下的假设函数值，用max(temp, [], 2);选取每行假设函数值最大者，将其index返回给p即可。



2 Neural Networks

前面实现的是用logistic回归对training set进行预测，但是这种情况只有线性回归，不能实现非线性的分类，所以引入了神经网络来进行更复杂的运算。

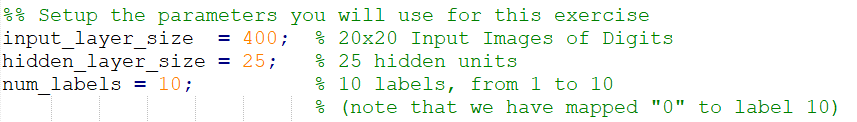
由于截止本次只学习了向前传播算法来预测，即已知weights θ，直接正向计算假设函数值，所以只实行该算法。而后面学习向后传播算法之后，就可以让机器自己学习如何获取θ值了。

2.1 Model representation

本次实验的神经网络架构只用到了3层，即input、hidden、output，结构如下：

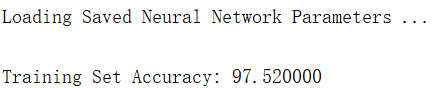


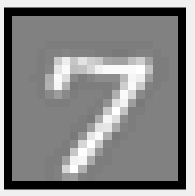
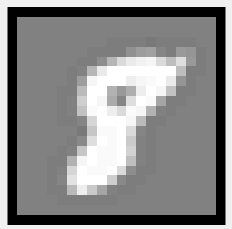
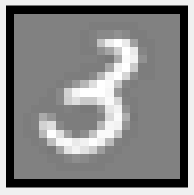
由于向前传播算法需要已知weights，所以ex2\_nn函数已经实现给出了Θ(1) Θ(2)，由代码中的注释可知，第二层的hidden是有25个单元，输出为10个单元（恰好对应0 ~ 9十个数字）



2.2 Feedforward Propagation and Prediction

这一部分需要完善predict函数来返回神经网络的预测结果，与one-vs-all的logistic回归类似，每次需要返回一个结果向量，对应X的每一行。需要注意的是，在每一层计算sigmoid前，都要先对input的矩阵进行加列处理，即在第一列前加上一列0表示x1。两次计算后，直接返回第二次计算的计算结果即可。

  
 以下是几个图片识别的例子：

○至此实验已经全部完成！