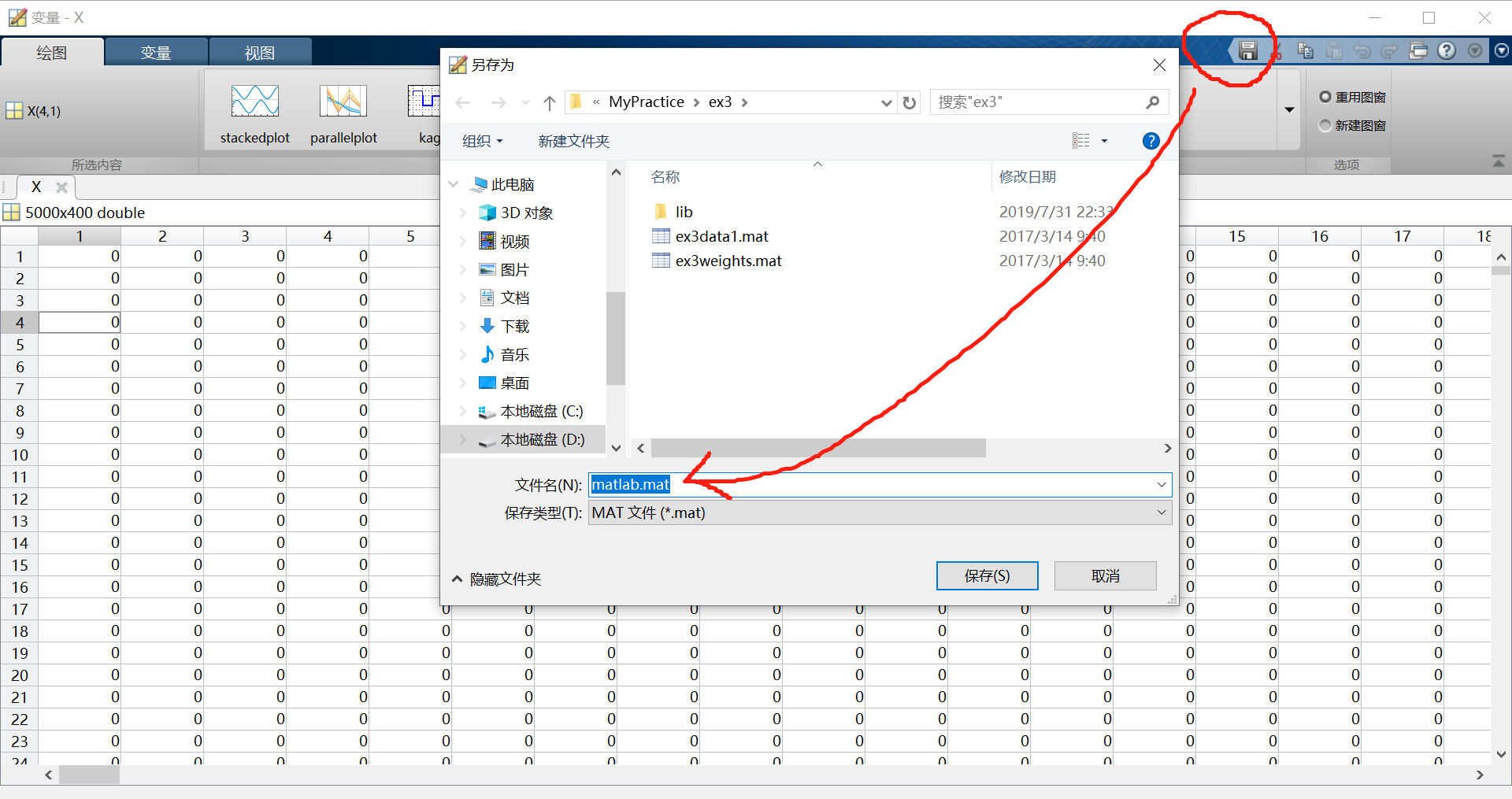
编程练习3：多元分类与神经网络

要点汇总

* fmincg函数是教授给的，它不是MATLAB的内置函数，对新手来说，应该会有很大价值
* size(X,1) %取矩阵X的行数
* size(X,2) %取矩阵X的列数
* 给定的X矩阵中包含5000个训练样本，维度为5000×400,400表示每个样本的大小为20×20像素，而这400个值表示每个像素位置上的灰度强度。这些样本已经有了正确的预测结果，应该是人工预测的，并且按顺序保存在了y向量中，X矩阵中保存的样本也是按规律存放的，1~500为数字0，用‘10’来标注，之所以不用‘0’来标注，很可能是因为要用它来作为数组索引，而数组索引没有0，只能从1开始
* 在打开MATLAB中的变量窗口后，点击右上角的保存按钮，即可将MATLAB中的矩阵保存为.mat格式的文件



* p = randperm(n) 返回行向量，其中包含从 1 到 n（包括二者）之间的整数随机置换。

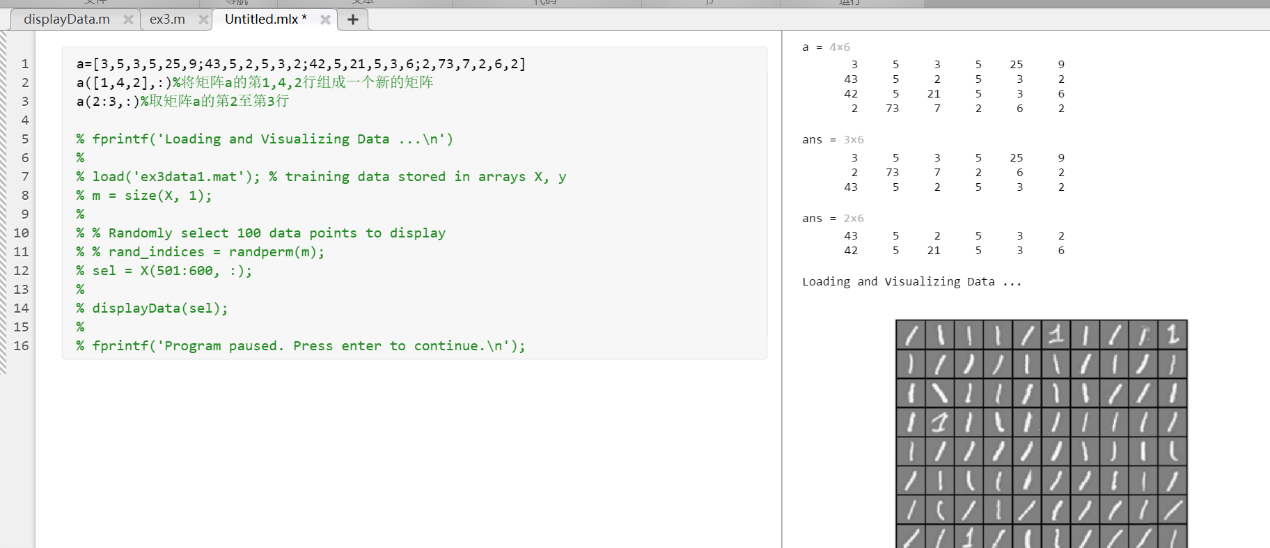


* 以下是关于选取矩阵内数据的一些测试代码。

a=[3,5,3,5,25,9;43,5,2,5,3,2;42,5,21,5,3,6;2,73,7,2,6,2]

a([1,4,2],:)%将矩阵a的第1,4,2行组成一个新的矩阵

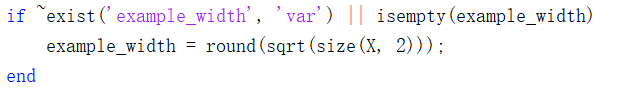
a(2:3,:)%取矩阵a的第2至第3行



* 自定义函数displayData用于将训练集中随机选取的100个样本转化为灰度图并显示出来。



这个函数有两个参数，X表示需要被转化成图片的矩阵，example\_width是每个图片的宽度（以像素为单位）。在主程序中调用这个函数时，可以不指定第二个参数，该函数的开头就是在检查调用函数时是否包含第二个参数。如果第二个参数不存在或者值为空，就取矩阵X的列数400，再取它的平方根，再取整，来作为每个图片的宽度，而400就是每个图片的像素数量



以下语句是在计算每个图片的高度（以像素为单位）。

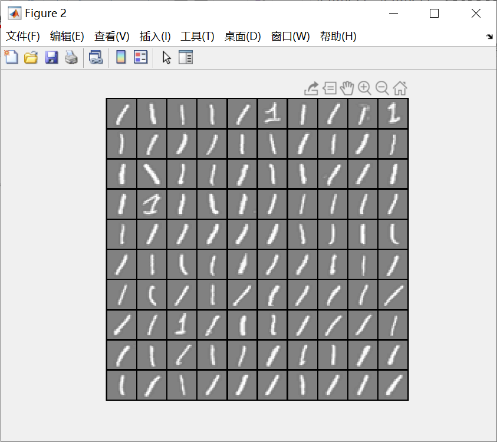
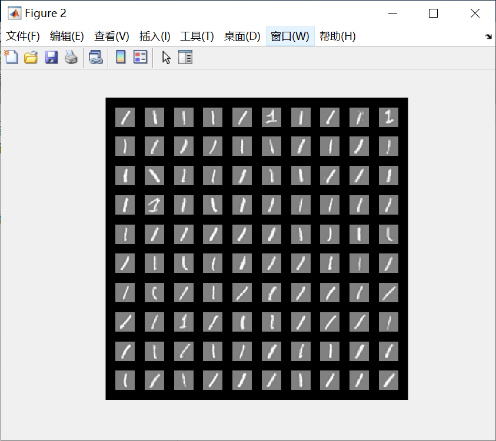


colormap(map) 将当前图窗的颜色图设置为 map 指定的颜色图。如果您为图窗设置了颜色图，图窗中的坐标区和图将使用相同的颜色图。新颜色图的长度（颜色数）与当前颜色图相同。

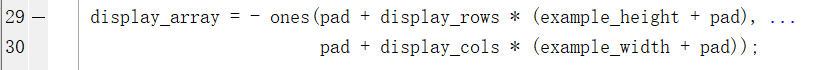
函数内第13行，将当前图窗颜色改成了灰色。



变量pad用于控制每个图片之间的间隔，以下是pad=1和pad=10时输出结果的对比。

可以明显看出pad=10时图像之间的间隔变大了。



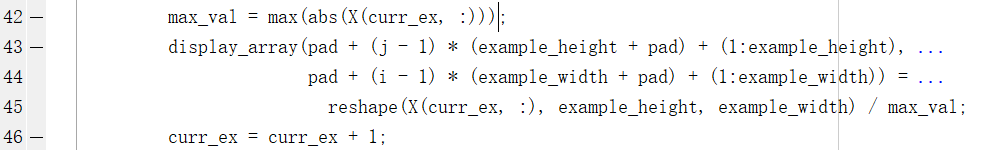
函数内第29行计算的变量display\_array用于设置空白显示，也就是控制显示的每个图片之间的间隔用什么颜色来填充。ones()函数内的表达式计算的是可视化结果图中像素的维度(211×211)。

灰度值-1显示为纯黑色，+1显示为纯白色，所以代码中ones前加上一个负号，就是将最终结果的图片之间的间隔设置为黑色。

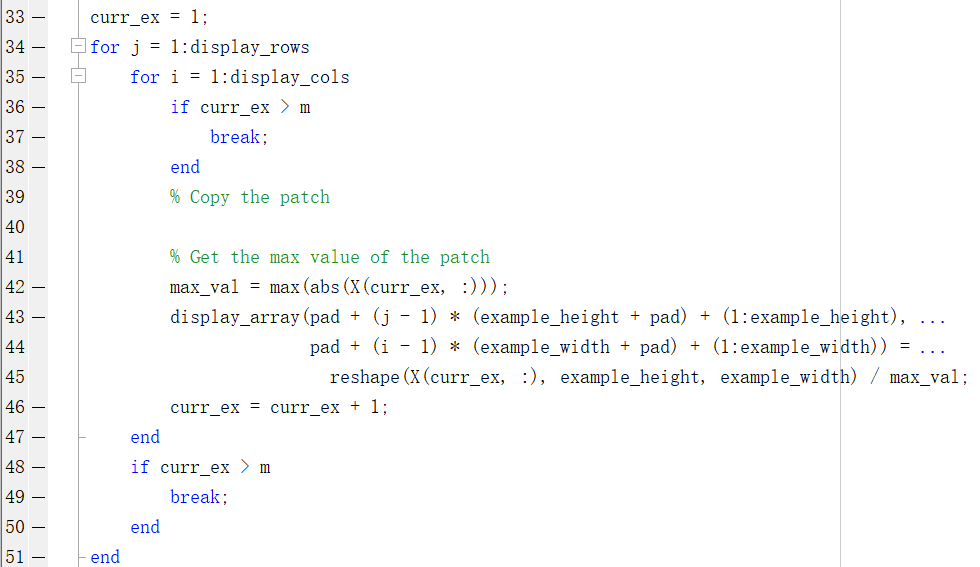
函数中第42行，先取了矩阵中的最大数，又让矩阵中每个数除以这个最大数，这应该是就是一种归一化处理，作用可能类似于特征缩放。这种处理对于本题而言没有太多的帮助，但是却具有普遍意义（放在别的程序中也能用）。

关于归一化的参考资料：

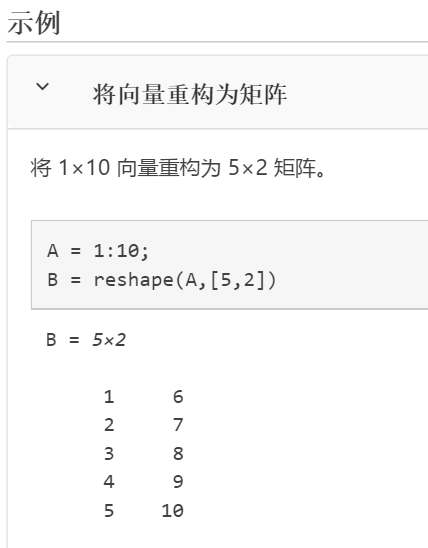
https://blog.csdn.net/tiankong\_/article/details/88173515



从函数的第33行开始，向display\_array矩阵中复制样本图。

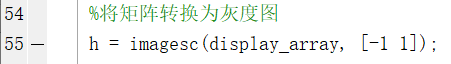
就像在一张黑布上贴一个个小图片，顺序为从左至右、从上至下，如果交换了一对i、j的话，顺序就变成了从上至下、从左至右。变量curr\_ex控制贴向黑布的样本的序号，每次循环结束贴上去一个样本，变量curr\_ex自加1，一共要贴100个。当变量curr\_ex>100时就退出循环，但是一个break只能退出一层循环，所以还要在外层循环再判断一次并加上第二个break。

上图中的reshape函数的作用是重构数组。如下图，经测试证明下图中的方括号可以省略。

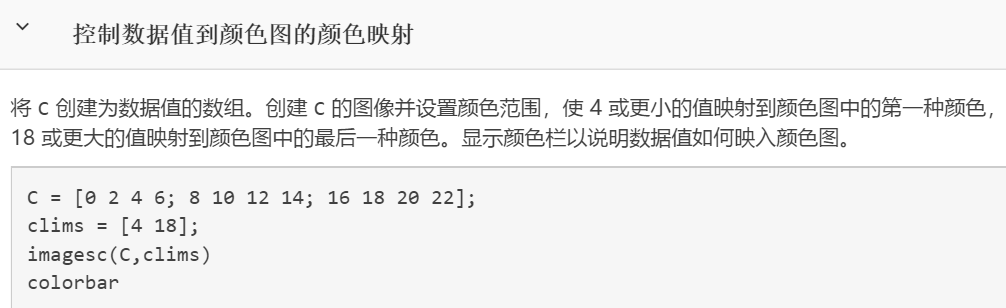


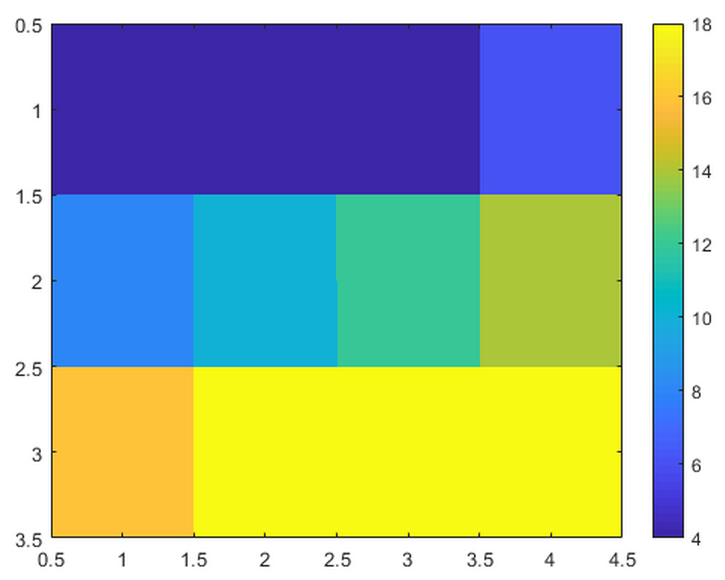
每个小图片有多高就说明它在黑布中占了多少行，所以变量example\_height会用于控制矩阵display\_array的行数选取，这乍一看可能会引起疑惑。

接下来函数在第55行将矩阵转换为灰度图。



其中imagesc函数的作用是显示使用经过标度映射的颜色的图像。



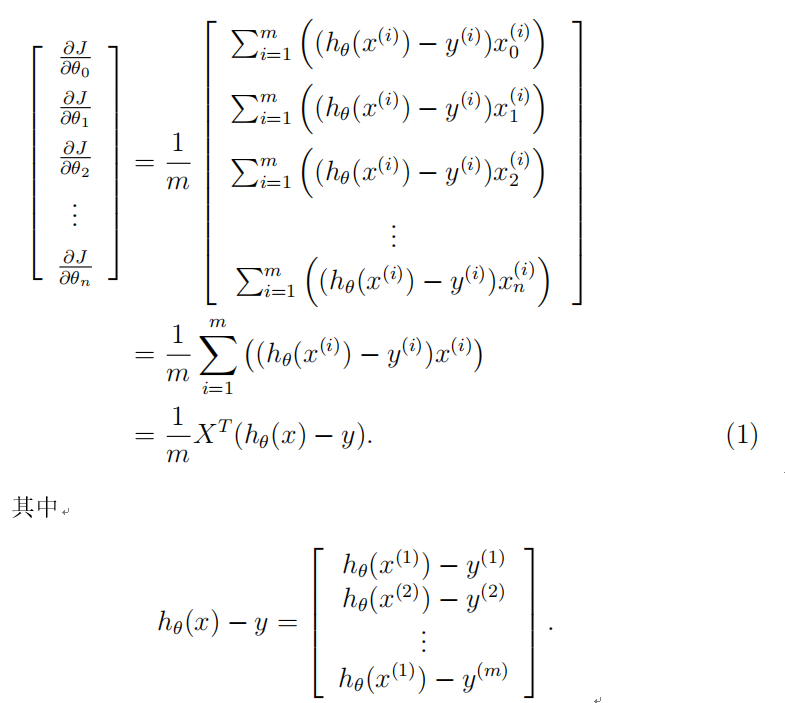


方括号内两个数值的作用是控制颜色范围。

Part 2a 向量化逻辑回归

* 这部分内容根本没有涉及到神经网络的知识，只是对之前逻辑回归的内容做了一个回顾，并检验了对这部分内容的理解，而且这些内容会用于神经网络中。
* 这里参数向量和训练集样本都已经给出来了，所以能够明确地计算出代价函数值（J）和梯度向量(grad)，所以只要补全代价函数中的代码就可以了，这一部分可以参照笔记中的7.4节和4.6节。补全代码后，程序会显示补全代码的计算结果和预期结果的对比，用以检验代码中是否有逻辑错误。

矢量化梯度的公式非常有用，具体细节参见G:\Python相关学习资料汇总\人工智能相关\吴恩达Matlab实现代码\AndrewNgMachineLearning-master\code\machine-learning-ex3\译文\_ex3.docx 文件中的1.3.2节



* 在计算J时出现重大失误中的“1-”错放进了sigmoid函数里，而正确做法应该放进log函数里（sigmoid函数外）。

## Part 2b 一对多训练

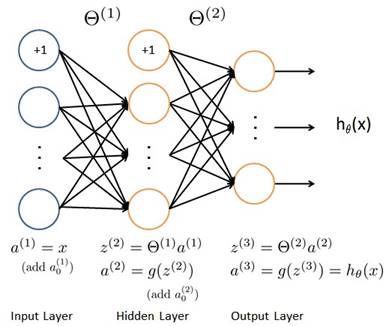
* 关于fmincg函数的使用可以参考笔记6.6节
* 在这个部分需要利用max函数返回矩阵中每行最大元素的索引



上图所示方法返回的I保存的是矩阵中每列最大元素的行索引，如果改成[M,I]=max(A’)则I保存的是矩阵中每行最大元素的列索引。

## 神经网络部分

* 重点是下图上面写个公式，每个都会用的到。



* 注意构造矩阵时需要加上偏置单元，可能是加上一列数字1再转置，也可能是加上一行数字1
* predict函数内不应出现像5000这样的具体数字，而应该使用相应的变量，否则主程序在调用随机的一个样本时会出现维度不一致的错误。