

模式识别与机器学习

实验报告（二）

班级：计科1505

姓名：陶婉莹

学号：0902150521

指导教师：梁毅雄

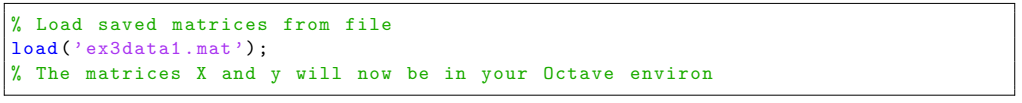
**Part I**

**Neural Networks:Feedforward Propagation**

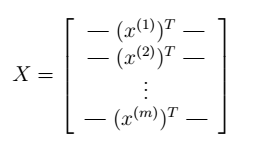
**1.dataset**

在ex3data1.mat给出了一个包含5000个手写数字训练样本的数据集。.mat格式意味着数据已经被保存在本地Matlab矩阵格式，而不是像csv文件一样的文本格式。

这些矩阵可通过使用load命令直接被读入工程中。



在ex3data1.mat文件中有5000个长度为400的向量，每个向量可以展示为一个20\*20的图像。这样每个训练样本都相当于矩阵X中的单独的一行。于是有了一个5000\*400大小的矩阵X，其中每一行都是手写数字图像的样本。

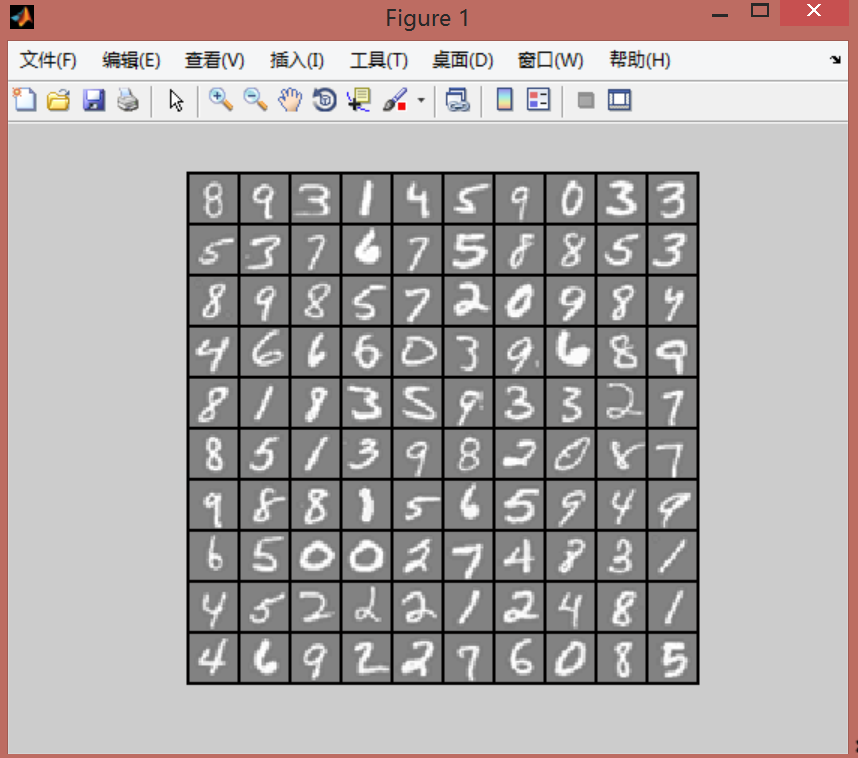


Ex3data1.mat中还有一个长度为5000的向量y，是图像所对应的数字（0被标记为“10”，1-9被标记为“1-9”）。

**2.visualizing the data**

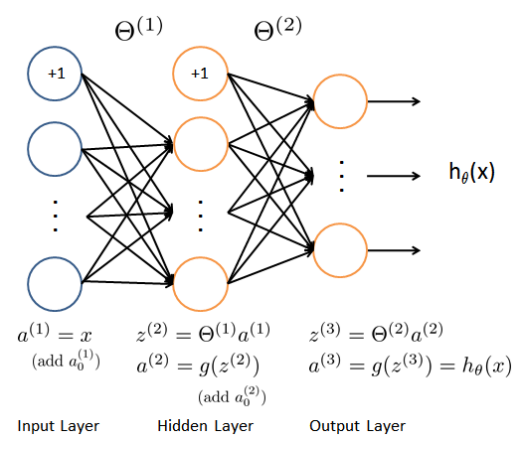
ex3.m中的第一部分代码从X中随机选择100行并传递给displayData函数中。这个函数把每一行映射为一个20像素\*20像素的图像并且一并将这些图像显示出来。

运行结果：

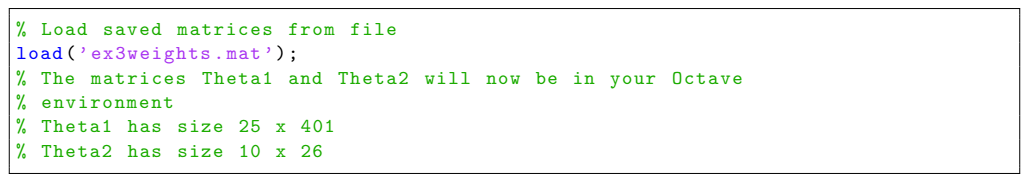


**3.model representation**

神经网络如下图所示



一共有三层：输入层、隐藏层、输出层。所有的输入都是数字图像的像素值。因为图像是20\*20，所以有400个输入层单元。训练集会被加载到变量X和y中。网络参数已经训练好并存储在ex3weights.mat中，且在ex3\_nn.m中加载到Theta1和Theta2。参数维度对应于一个第二层中有25个单元和10个输出单元的神经网络。注意到每个“神经元”的输入-输出映射关系其实就是一个逻辑回归。可采用sigmoid函数，也可选择双曲正切函数（sigmoid函数的一种变体，取值范围为[-1.1]，而不是sigmoid函数的[0,1]）。



**4.feedforward propagation and prediction**

现在对神经网络应用反馈传播。完成predict.m中的代码返回神经网络的预测。对每个样例i用反馈计算算出并且返回相应的预测值。近似于一对多分类策略，神经网络的预测是最大输出结果。

Note：矩阵X的每行中包含样例。补全代码时要向矩阵加一列1。Theta1和Theta2的行中包含每一个单元的参数。Theta1的第一行对应第二层中第一个隐藏单元。

ex3\_nn.m会调用预测函数，预期的结果应该是97.5%。然后每次会显示训练集中的一个图像。同时控制台打印显示图像的预测标签。

补充代码：X = [ones(m, 1) X];

z = X\*Theta1';

X2 = sigmoid(z);

X2 = [ones(m, 1) X2];

temp = X2\*Theta2';

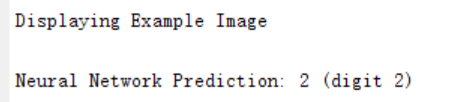
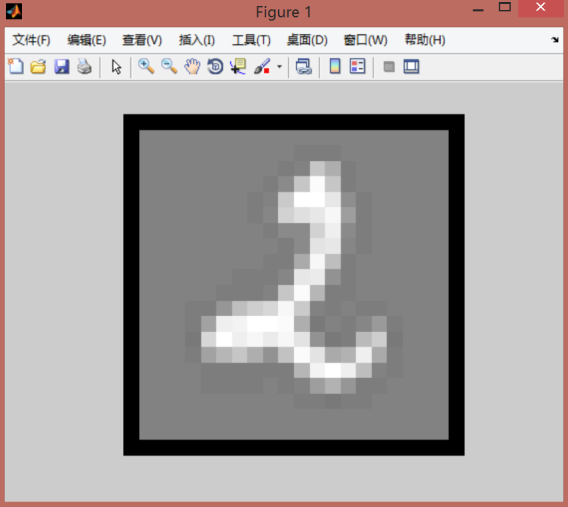
[M,I] = max(temp,[],2);

p = I(:);

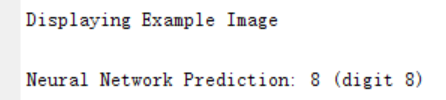
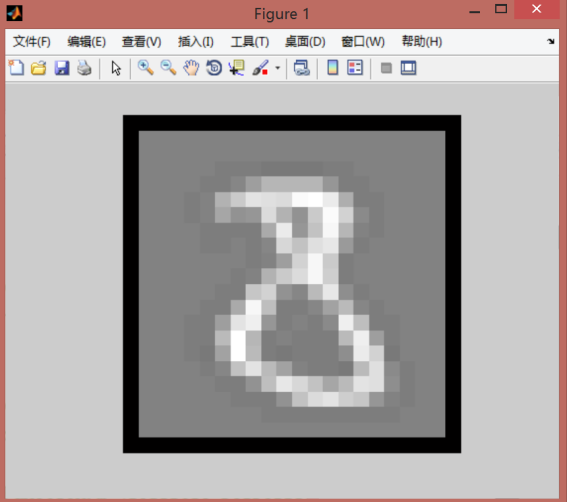
运行结果：



**样例一**



**样例二**

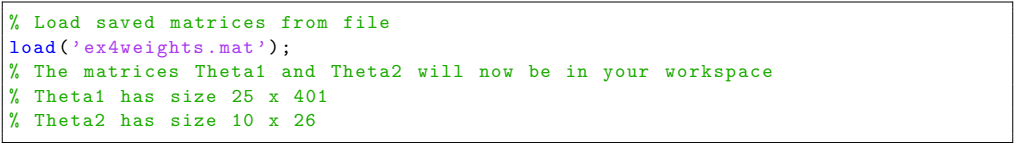


**Part II**

**Neural Networks:Backpropagation**

之前已经完成了神经网络的前馈传播，并用它来根据提供的权值预测手写数字。现在用反向传播的算法来为神经网络学习参数。

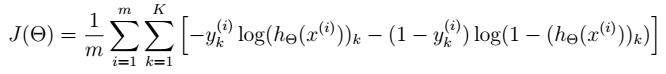
训练集数据已经在ex4.m中被导入至变量X和y中。网络参数训练好存在ex4weights.mat中，并且被导入至ex4.m的Theata1和Theta2中。参数的维度与一个第二层有着25个神经元、有10个输出元的神经网络相符。



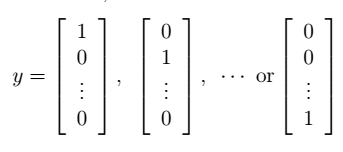
**5 Feedforward and cost function**

**5.1 Unregularized cost function**

首先完成nnCostFunction.m中的代码。神经网络的损失函数（没有正则化）是



其中的计算如图所示，是所有可能的标签的总和。是第k层输出元的激活（输出值）。鉴于初始标签是1,2,...10，为了训练一个神经网络，需要重新将标签写为只含有0,1的向量，因此



例如，如果是数字5的图像，那么对应的y应该是一个十维的、其他为0的向量。对每个样例i计算并把所有样例的损失加起来。矩阵X在行中存储样例（例如是第i个训练样本，用一个nx1的向量表示）。完成nnCostFunction.m中的代码，需要向X矩阵中加入一列1。神经网络中每个神经元的参数在矩阵Theta1和Theta2中以行的形式表示出来。Theta1的第一行对应第二层的第一个隐藏神经元。可用for循环来计算样本的损失。运行ex4.m调用nnCostFunction。预期损失大约为0.287629。

补充代码：X = [ones(m, 1) X];

s = 0;

for i = 1:m

yt = zeros(num\_labels,1);

yt(y(i)) = 1;

z2 = Theta1\*X(i,:)';

a2 = sigmoid(z2);

a2 = [1;a2];

z3 = Theta2\*a2;

a3 = sigmoid(z3);

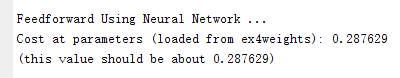
temp = -yt.\*log(a3) - (ones(num\_labels,1)-yt).\*log(ones(num\_labels,1)-a3);

s = s + sum(temp);

end

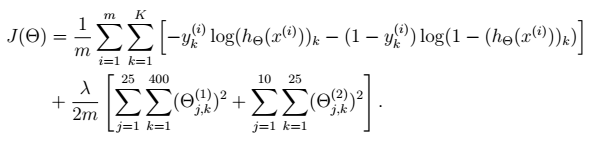
J = s/m;

运行结果：



**5.2 Regularized cost function**

正则化后的神经网络损失函数为



把正则项加入到损失函数中。然后运行ex4.m调用nnCostFunction使用Theta1和Theta2的参数集，且。预期结果应大约为0.383770。

补充代码：temp1 = Theta1;

temp1(:,1) = 0;

temp2 = Theta2;

temp2(:,1) = 0;

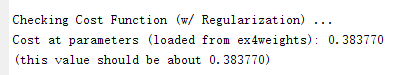
T1 = temp1.^2;

T2 = temp2.^2;

s2 = (0.5\*lambda\*(sum(T1(:)) + sum(T2(:))))/m;

J = J + s2;

运行结果：



**6 Backpropagation**

完成nnCostFunction来返回梯度的合适值。一旦计算出梯度，就可以通过最小化损失函数来训练神经网络。先完成未正则化的神经网络，结果正确后再完成正则化神经网络的梯度下降计算。

**6.1 Sigmoid gradient**

先完成sigmoid梯度下降函数。计算公式为 先输入一些值在命令行调用sigmoidGradient(z)来验证正确性。例如数很大的时候梯度应接近于0，为0时梯度应为0.25。

运行结果：



**6.2 Random initialization**

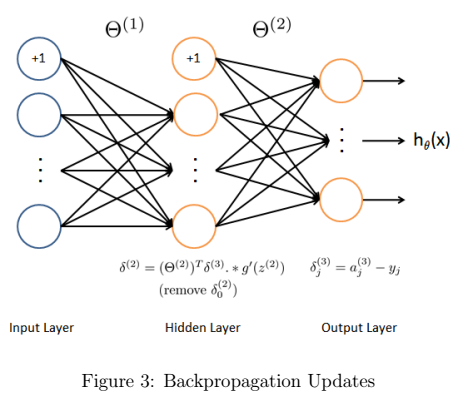
随机初始化的一个有效策略是在间均匀地随机选择Theta的值。这个范围的值能确保参数很小并且学习地更有效。完成randInitializeWeights.m来初始化。

补充代码：epsilon\_init = 0.12;

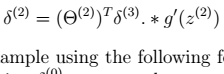
W = rand(L\_out, 1 + L\_in) \* 2 \* epsilon\_init - epsilon\_init;

**6.3 Backpropagation**

反向传播算法直观表示如下。给定一个训练样本，首先计算网络中所有的激活，包括假设函数的输出值。然后对l层的每个节点j计算去衡量节点对输出中的错误有多大的影响。对于一个输出节点，可以直接计算网络激活和真实目标值间的差异，然后用它来决定。对于隐藏神经元，用l+1层的节点的错误项平均权值来计算。



循环进行步骤1-4，每次处理一个样本。用一个t = 1:m的for循环，第t次迭代计算第t个训练样本。步骤5将累积的梯度除以m来获得神经网络的损失函数。

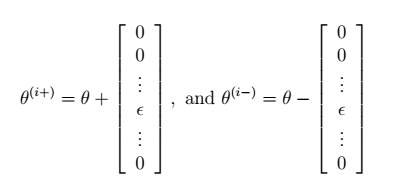
1. 设置第t个训练样本输入层的值。计算第2、3层的激活。注意要加上“+1”项去确保层和层的激活向量也含有偏差元。在Matlab中，如果a\_1是列向量，加上。
2. 对第3层的每个输出元k，设置，其中表示当前训练样本是属于k类还是属于另一个类。
3. 对l=2层隐藏层，设置。
4. 用公式累计这个样本的梯度。注意要跳过或去除。在Matlab中用来去除。
5. 将累计得到的梯度除以m得到神经网络损失函数的未正则化梯度。

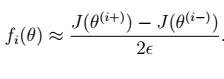


完成反向传播算法后，ex4.m将会执行gradient checking，来验证代码计算梯度的正确性。

**6.4 Gradient checking**

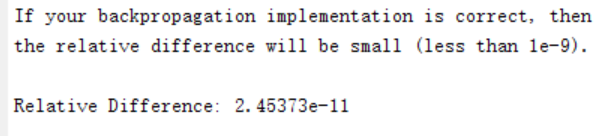
想象把参数展开成为一个长向量。假设函数可以计算，验证 是否输出了正确的导数值。



和一样，除了它的第i个元素被加上了。相似地对应向量的第i个元素被减去了。现在可以用公式来判断的正确性。两个值的相近度依赖于J。但是当 时左右两边的值至少有四位是相符的。computeNumericalG-

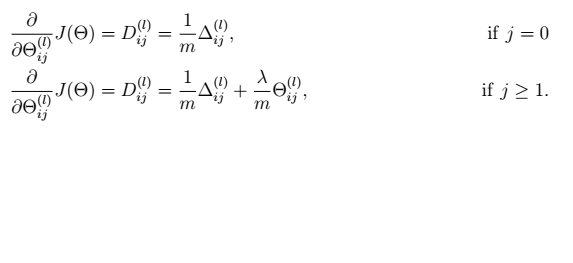
Radient.m中已经给出了相关代码。ex4.m中的下一步会运行checkNNGradients.m，创建一个小的神经网络和数据集去验证梯度。如果反向传播算法实现正确，预期结果应小于1e-9。

运行结果：

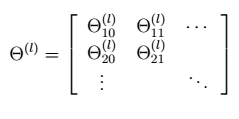


**6.5 Regularized Neural Networks**

把正则项加到梯度当中。用反向传播计算出后 ，用下列公式加入正则项。



第一列用于偏差项的不用正则化。中i是从1开始而j是从0开始，因此



调试nnCostFunction中的代码计算正则化后的梯度。然后运行ex4.m进行gradient checking。如果代码正确，预期结果应小于1e-9。

补充代码：s = 0;

TG1=0;TG2=0;

for i = 1:m

yt = zeros(num\_labels,1);

yt(y(i)) = 1;

z2 = Theta1\*X(i,:)';

a2 = sigmoid(z2);

a2 = [1;a2];

z3 = Theta2\*a2;

a3 = sigmoid(z3);

delta3 = a3 - yt;

delta2 = (Theta2(:,2:end))'\*delta3.\*sigmoidGradient(z2);

TG1 = TG1 + delta2\*X(i,:);

TG2 = TG2 + delta3\*a2';

end

Theta1\_grad = TG1/m;

Theta2\_grad = TG2/m;

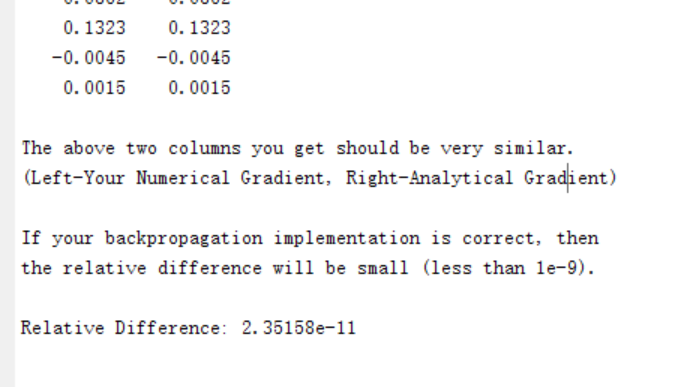
temp1 = Theta1;temp2 = Theta2;

temp1(:,1) = 0;temp2(:,1) = 0;

Theta1\_grad = Theta1\_grad + lambda\*temp1/m;

Theta2\_grad = Theta2\_grad + lambda\*temp2/m;

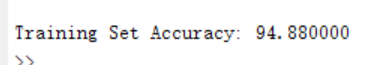
运行结果：



**6.6 Learning parameters using fmincg**

ex4.m的下一步使用fmincg学习一组好的参数。训练结束后，ex4.m会通过计算样本正确性来显示分类的训练准确度。如果正确地实现了，预期结果应该大约为95.3%（随机初始化可能会导致1%的误差）。如果神经网络训练次数加多那么可能会得到更高的训练准确度。

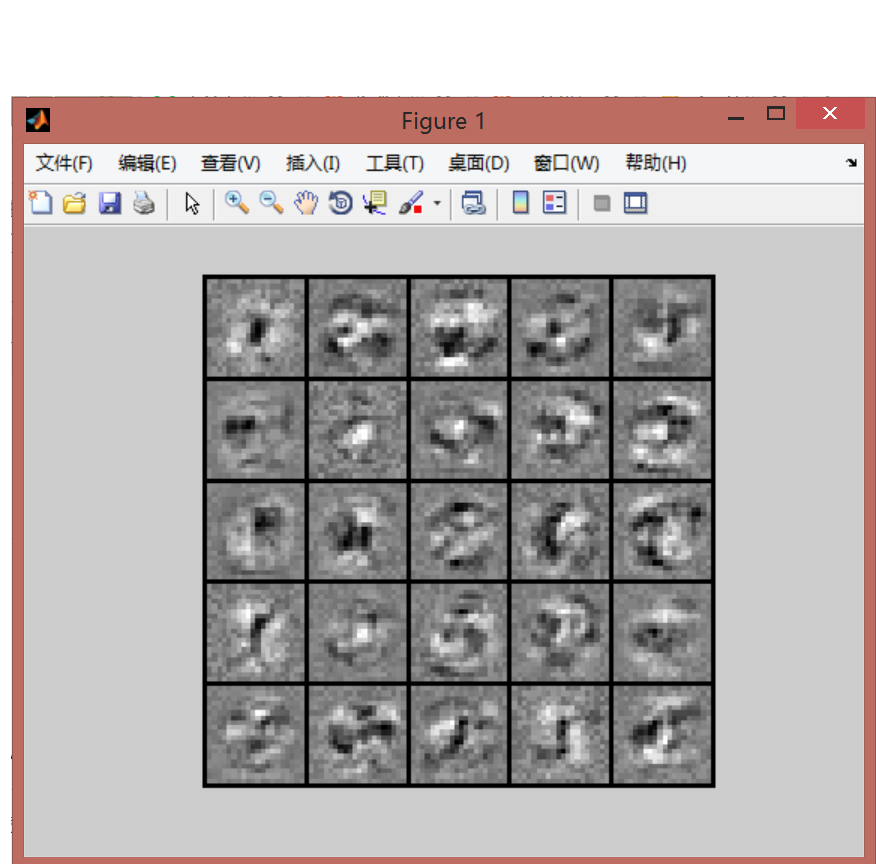
运行结果：



1. **Visualizing the hidden layer**

弄清神经网络正在学习什么东西的一个方法是可视化隐藏神经元捕捉的事物。给定一个特定的隐藏神经元，可视化其计算的一个方式是需要一个输入x去激活它。注意训练的神经网络中的第i行是一个401维的表示第i个隐藏神经元的参数的向量。如果抛弃偏差项，则是一个400维的表示每个隐藏神经元输入的权值的向量。因此可视化隐藏神经元捕捉到的事物的一个方法是将这个400为的向量重塑为一个20x20的图像并展现出来。ex4.m的下一步通过使用displayData函数展示一个有25个单元、每个对应一个隐藏神经元的图像。

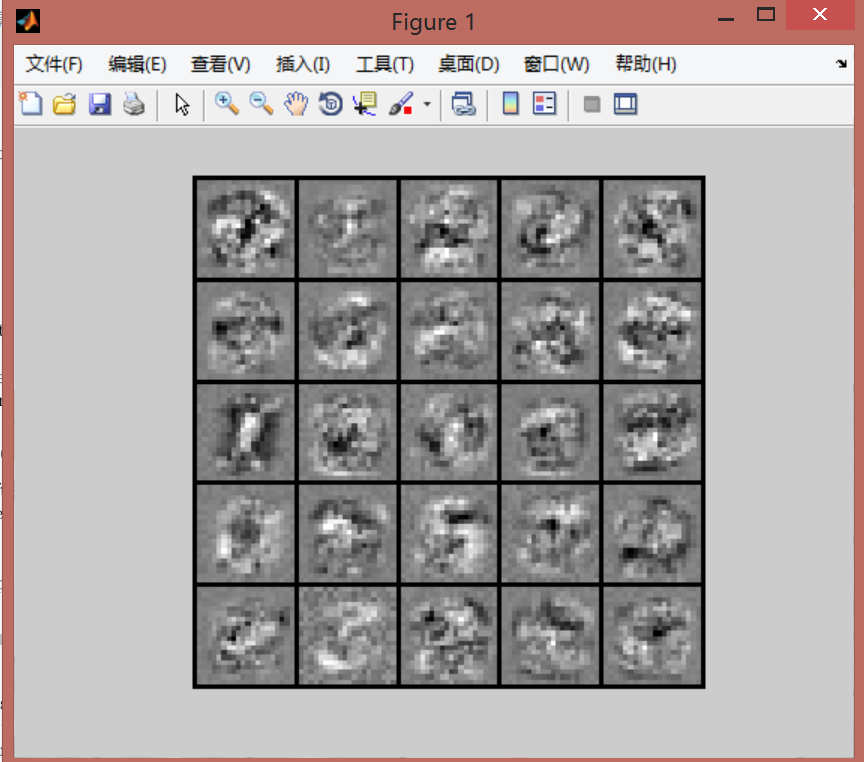
运行结果：



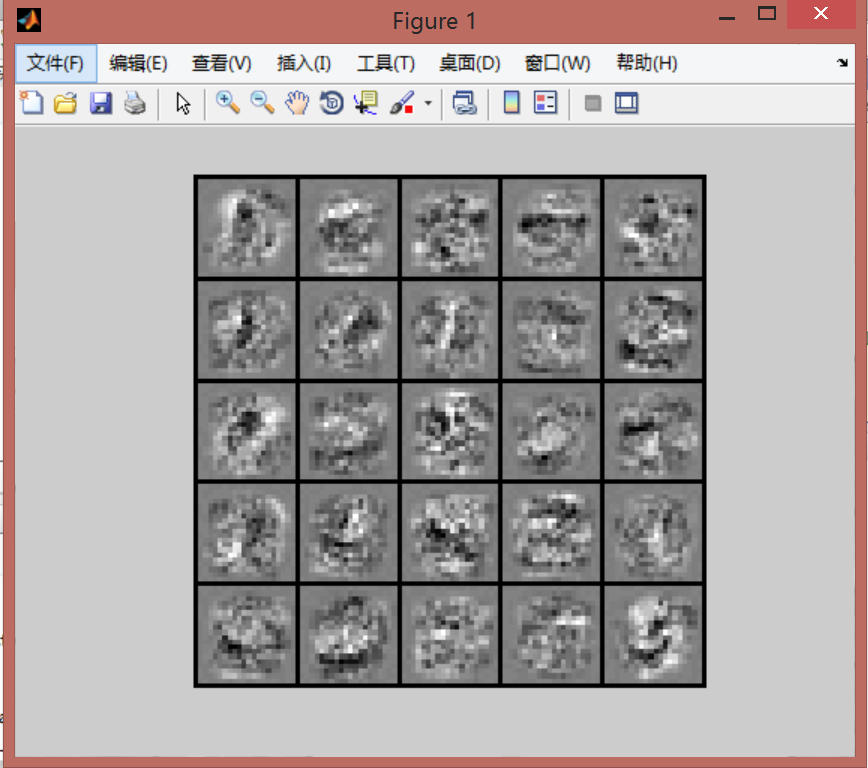
**7.1 Choosing the regularization parameter λ**

尝试为神经网络进行不同的学习设置来观察正则化参数λ和训练步骤的次数对神经网络表现的影响。神经网络是可以形成非常复杂的决定边界的模型。没有正则化，一个神经网络可能会过拟合一个训练集，所以它可能在训练集上有着100%的正确但在新的样本上表现地就不那么好了。将λ设置得更小，同时将MaxIter参数设置得更大。可以看到调整参数后展示的图像也有所变化。

将λ设置为2，MaxIter设置为100后的图像：



将λ设置为1，MaxIter设置为400后的图像：



**实验心得**：这是本课程的第二次实验——训练一个神经网络来识别手写数字0到9。神经网络算法的来源可以从字面意思看出来，也就是模拟大脑的工作机制，实现学习、认知、决策等复杂功能。这是我比较感兴趣的一章内容，不过同时也比较难理解。比如说神经网络中的hidden layer和隐藏单元的相关概念对我来说有些困难，而这些同时也是神经网络的关键，一开始刚上完课并不能够准地理解，但是经过这次实验，通过自己一步步补充代码、分析调试到最后运行成功，对这些知识有了更好的掌握。之前也不是很理解激活函数的作用，但通过实验和上网搜索相关资料了解到引入激活函数是神经网络具有优异性能的关键所在，多层级联的结构加上激活函数，令多层神经网络可以逼近任意函数，从而可以学习出非常复杂的假设函数。 这次实验中，我能够发现问题，通过多渠道搜集资料并逐一解决问题，加深了对机器学习中神经网络这一节内容的掌握，虽然学习的还很浅显，但是今后我会继续深入学习，了解更多相关知识。