# 作业4:神经网络

说明：本次作业会用到MATLAB工具箱以及代码，故使用MATLAB来完成本次作业会相对容易，若使用Python，R等其它语言来完成本次作业亦可，但需要提交全部代码以及题目所要求的内容。

## 1 神经网络在MNIST数据集上的应用(60%)

MNIST是著名的手写体数字图片集合，如图1所示。其中0,1,2,...,9的数字图片是由$28 \times 28 = 784$ 像素组成。利用神经网络的方法可以从这些手写体数字图片中识别出对应的数字。文件MNIST.mat包含了四个矩阵trainX(训练图片), trainY(训练图片标签),testX(测试图片), testY(测试图片标签)，对于trainX(60000,28,28)，testX(10000,28,28)，是三维矩阵,它的各个维度的含义是:(样本，像素行，像素列)，即对应为60000个和10000个$28 \times 28$的图片，每个像素取值为0到255 的整数灰度值；对于trainY(60000,10)，testY(10000,10)，是二维矩阵，它的各个维度的含义是:(样本，数字类型)，取值为0或1，总共10 列，依次分别对应于数字类别0到9，如果某样本为数字d，则对应行的第d+1列的值为1，其余为0。

图 1: MNIST手写数字图片集合

(1)Matlab中提供了该神经网络的工具箱Neural Pattern Recognition，可以进行多层神经网络的分类任务。请按照如下步骤完成3层神经网络对MNIST数据集的分类问题：

a)请先查看数据内容，在所有样本中选取出10个不同数字(0-9)的图片各一张，绘制出类似于图1的图片(可能用到的函数：subplot,imshow等)

b)预处理。按照神经网络工具包的输入要求，需要将每一副图片的所有像素都转换成1维向量，例如，对于训练集trainX 需要转换为$6000 \times 784$ 的矩阵(可能用到的函数：reshape)。

c)通常我们需要将对神经网络的输入数据进行规范化(归一化)，由于灰度值为0-255，所以各维特征除以255即可

d)使用Matlab神经网络工具箱(例如Matlab2016a→应用程序选项卡→Neural Net Pattern Recognition)，分别设置隐藏节点为5,10,20,40,100 个的时候，在训练集合(trainX,trainY) 上进行训练(70% 训练，15%验证，15%测试)，在测试集合(testX,testY) 上进行测试，保存混淆矩阵(confusion matrix)、训练误差曲线，分析错误率和收敛速度与节点数目的关系，并解释可能的原因。

说明：混淆矩阵的含义。如图2所示，横坐标代表目标类别，纵坐标代表训练预测结果类别，各坐标点(x,y)方格中的整数代表将目标类别x预测成了y的样本个数，其中的百分数代表该种情况占所有样本的比例。对角线上的数值为预测正确的情况，其余为预测错误的情况。该矩阵下面额外一行代表的是对应目标类别的分类正确率和错误

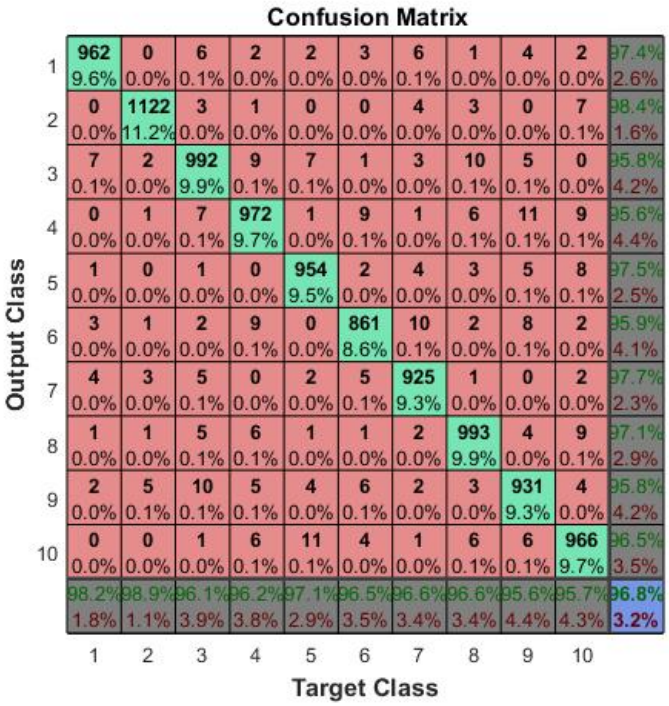


图 2: 混淆矩阵示例

率，而矩阵右边额外一列代表的是对应输出类别的正确率和错误率，右下角为所有样本的分类正确率和错误率。

提示：训练结果可能有随机因素，可训练多次重复训练，体会其中的变化，此项不做要求，仅需要提交一次训练结果即可。

要求：请提交所有的混淆矩阵，训练误差曲线(Matlab中为Performance)

(3)在(2)中所保存的混淆矩阵中(仅研究隐藏结点为100的模型)，指出混淆最严重的两个数字(只需要一组)，分别展示分错数字的例子(各3个，不足3个的按实际个数即可)。

## 2 稀疏自动编码器(40%)

本题目的是帮助各位同学掌握神经网络反向传播算法的推导，学习“矩阵-向量化”的编程方法，完成一个“自动编码器”的设计与实现。通过可视化神经网络的学习结果，大家能更深入地理解神经网络的基本原理(比如说，神经网络是如何做到同时完成特征选择和预测这两个任务的)。相信大家在完成本次作业后会有很大收获。本题作业内容摘录改编自斯坦福大学的深度学习教程[1]，感兴趣的同学可以进一步学习。

在题目中，激活函数采用sigmoid 函数：

$$f(z) = \frac{1}{1 + \exp(-z)} \tag{1}$$

在本问题中，你将实现稀疏自动编码器的算法，展示出它发掘的各种图像边沿特征。这种特殊神经网络的训练目标是使得任意输入一个样本后的输出与输入差异最小，中间层的输出可以看作是输入的一个压缩编码(希望详细了解的同学可以阅读文献[7])。其中的程序框架已经用Matlab 实现了(sparseae_exercise.zip)，你需要完成几个主要函数的实现，这些地方都标注了("YOUR CODE HERE")。你需要完成以下文件：sampleIMAGES.m, sparseAutoencoderCost.m, computeNumericalGradient.m。在启动程序文件train.m 中展示了上述函数的调用方法和次序，你只需运行这个文件即可得到最后的结果。

要求：请提交所生成的图片和所有代码(请维持原来的文件结构，包括所有提供的代码，保证无需修改即可执行)，此外，请提交2.2中练习题的详细解答过程。

## 2.1 产生训练集：将代码sampleIMAGES.m补充完整

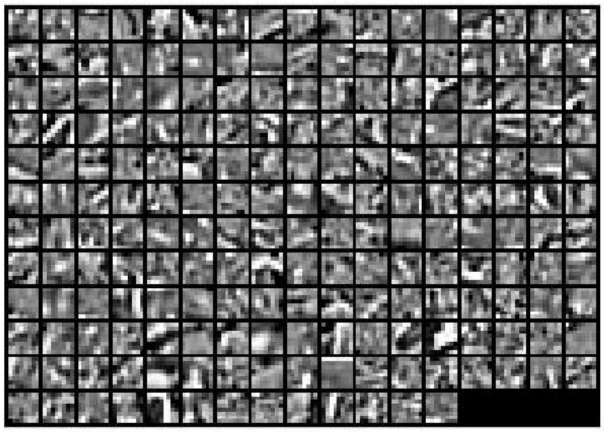

图 3: 样本生成结果实例

**(1)**在10幅图片中随机抽取一个$8 \times 8$ 像素大小的图片，并转化为$n = 64$ 维向量(行主序和列主序均可)，这样就可以得到一个训练样本$x$。总共采集$N = 10000$ 幅这样的图片,得到一个$64 \times 10000$的矩阵，即样本集合为$T = \{\boldsymbol{x}^{(i)}\}_{i=1}^{10000}$，之后我们将用$\boldsymbol{x}^{(i)} \in \Re^{64}$代表第$i$个样本对应的向量。

**(2)**运行文件train.m中标有"Step 1" 的代码(请选择Matlab中运行模块的“运行节”，后面的很多步骤也是如此)，它将画出其中200幅图片(如图3 所示)。

提示：在5秒钟以内，函数sampleImages()就能够运行完毕。如果超过了30 秒，请调整你的算法，提高算法效率。

## 2.2 计算稀疏自动编码器参数梯度：将代码sparseAutoencoderCost.m补充完整

该神经网络共三层(如图5所示)，它们的结点数目为：输入层($n = 64$)，隐层($n_1 = 25$)，输出层($m = 64$)。稀疏自动编码器的最小化目标函数$J(\boldsymbol{W}, \boldsymbol{w}_0)$ 定义如下：

$$J(\boldsymbol{W}, \boldsymbol{w_0}) = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\left\|\hat{\boldsymbol{x}}(\boldsymbol{x}^{(i)}) - \boldsymbol{x}^{(i)}\right\|^2\right)\right] + R + S, \tag{2}$$

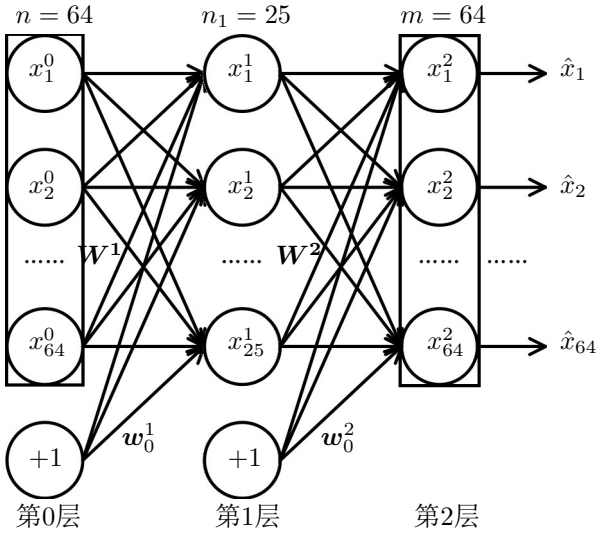

图 4: 自动编码器

其中第一项为我们上课提到的均方误差，$\hat{\boldsymbol{x}}(\boldsymbol{x}^{(i)})$为第$i$个样本的神经网络输出；第二项$R$是一个正则化项(也叫权重衰减项)，其目的是减小权重的幅度，防止过度拟合，具体可以参考文献[3,6]；第三项$S$是稀疏项(KL 距离)，目的是使得在任意样本输入网络后$\boldsymbol{x^1}$ 中仅有少数是接近1，其余均接近0，具体可以参考文献[4,7], 它们的定义如下：

$$R = \frac{\lambda}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n_1}\left(W_{ij}^1\right)^2 + \sum_{i=1}^{n_1}\sum_{j=1}^{m}\left(W_{ij}^2\right)^2\right), \tag{3}$$

$$S = \beta\sum_{j=1}^{n_1}\left(\rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right) \tag{4}$$

$$\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N}\left[x_j^1(\boldsymbol{x}^{(i)})\right] \tag{5}$$

其中，$x_j^1(\boldsymbol{x}^{(i)})$表示输入样本$\boldsymbol{x}^{(i)}$以后第$j$个中间层结点的输出值，而$\lambda = 0.0001, \beta = 3, \rho = 0.01$ 是定值。

请在反向传播算法的基础上进行一定的修改(例如，若不考虑正则项$R$，中间隐层的$\delta_i^1$需要修改为(6)式)，实现代码以求解相关参数导数(包括以下参数：权值矩阵$\boldsymbol{W}^1, \boldsymbol{W}^2$，截距向量$\boldsymbol{w}_{\boldsymbol{0}}^1, \boldsymbol{w}_{\boldsymbol{0}}^2$)，并完成下面的反向传播算法推导练习题。

$$\delta_i^1 = \left( \left( \sum_{j=1}^{n_1} W_{ij}^2 \delta_j^2 \right) + \beta \left( -\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i} \right) \right) x_i^1 (1 - x_i^1) \tag{6}$$

提示：a)请尽量让你的程序矩阵-向量化，否则会很影响效率。b)激活函数的导数很容易计算，可显式直接求解出来。c) 在目标函数中，你可以先忽略后两项，调试通过后再依次增加。

## 2.3 反向传播算法推导练习

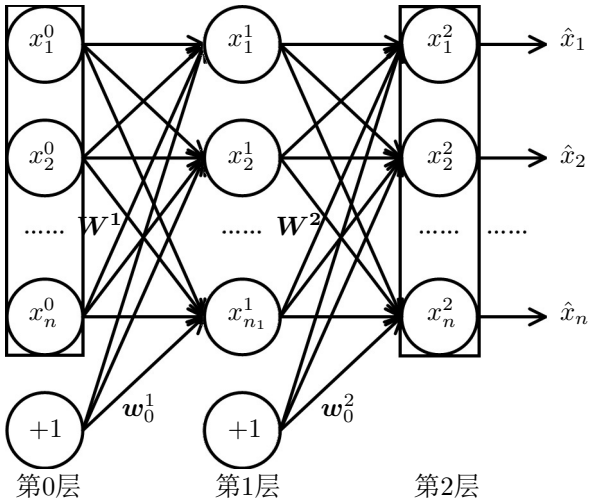

图 5: 自动编码器

已知稀疏自动编码器神经网络如图5所示，该神经网络共三层，各层的节点数目为：第0层（输入层$n$个节点，第1层（隐层）$n_1$个节点，第2层（输出层）$n$个节点。第$l$层的第$i$个神经元的输出记作$x_i^l$，第$l-1$层第$i$个神经元和第$l$层第$j$个神经元的连接权值记作$\boldsymbol{W}_{ij}^l$，第$l$层第$j$个截距项记作$\boldsymbol{w}_{0j}^l$。最后一层的输出也记作$\hat{\boldsymbol{x}}$。它是关于$\boldsymbol{x}$ 的函数：$\hat{\boldsymbol{x}}(\boldsymbol{x})$。为了方便推导和表示，在本小题中仅使用一个样本$\boldsymbol{x} \in \Re^n$来训练该神经网络（即$N=1$）目标函数定义如下：

$$J = \frac{1}{2} \left\| \hat{\boldsymbol{x}}(\boldsymbol{x}) - \boldsymbol{x} \right\|^2 + R + S, \tag{7}$$

$$R = \frac{\lambda}{2} \left( \sum_{i=1}^{n} \sum_{j=1}^{n_1} \left( W_{ij}^1 \right)^2 + \sum_{i=1}^{n_1} \sum_{j=1}^{n} \left( W_{ij}^2 \right)^2 \right), \tag{8}$$

$$S = \beta \sum_{j=1}^{n_1} \left( \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \right) \tag{9}$$

其中第一项为均方误差；第二项$R$ 是一个正则化项；第三项$S$ 是稀疏项(KL 距离),式子中 $\hat{\rho}_j = x_j^1(\boldsymbol{x})$，即为式子(5)中$N=1$的情况，而$\lambda, \beta, \rho$ 是定值。激活函数采用sigmoid 函数$f(.)$，令$J_s = \frac{1}{2}\left\| \hat{\boldsymbol{x}}(\boldsymbol{x}) - \boldsymbol{x} \right\|^2 + S$，$\delta_i^l$定义为当前输出目标值$J$中的其中两项($J_s$)对$l$层第$i$个节点值$z_i^l$ 的偏导：$\delta_i^l = \frac{\partial J_s}{\partial z_i^l}$，即有$x_i^l = f(z_i^l)$，计算结果内请不要出现变量$z$。下面请计算反向传播算法中几个关键的变量：

(1)请推导出$\delta_i^2$的表达式

(2)若所有残差$\delta$的表达式均已知，请推导出求解$\boldsymbol{W}_{ij}^1, \boldsymbol{w}_{0j}^2$的迭代关系式(关于$\delta$)

(3)当$\lambda = 0$，请证明：

$$\delta_i^1 = \left( \left( \sum_{j=1}^{n_1} W_{ij}^2 \delta_j^2 \right) + \beta \left( -\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i} \right) \right) x_i^1 (1 - x_i^1) \tag{10}$$

## 2.4 梯度检测：将代码computeNumericalGradient.m 补充完整

**(1)**完成梯度检验函数。

我们知道：

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} \tag{11}$$

所以，$\epsilon$很小时可以做如下近似(本例中取$\epsilon = 10^{-4}$ ):

$$f'(x) \approx \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} \tag{12}$$

由此我们可以通过该数值方法近似计算出目标函数$J$对各个参数的梯度，以对我们的解析解进行检验，判断程序中是否有误。请在computeNumericalGradient.m 中实现该数值计算过程。

**(2)**利用在checkNumericalGradient.m代码来测试你编写的computeNumericalGradient.m代码正确与否。

该文件中定义了一个简单的二元函数$h : \Re^2 \mapsto \Re, h(x) = x_1^2 + 3x_1x_2$，测试点为$x = (4, 10)^T$。如果你的程序是正确的，数值解和解析解将非常接近。

**(3)**使用computeNumericalGradient.m 来检测你在sparseAutoencoderCost.m 中实现的代码是否正确。关于细节，你可以看train.m 中的“Step 3”。

提示：在你调试错误的时候，你可以使用少量样本和少量隐藏节点，这样会使你节约更多时间。

### 2.5 训练稀疏自动编码器

运行train.m中的"Step 4"，等待训练完成。

提示：请注意，如果已经调试完毕，进入正式的训练阶段，请不要在每次迭代中都调用上一步的梯度检测，否则将显著降低计算速度。训练时间因计算机软硬件和实现形式的不同有差异，但一般不会超过到1小时，矩阵-向量形式实现的代码花费大约在10分钟以内。

### 2.6 可视化

在训练完稀疏自动编码器以后，使用display_network.m来可视化训练结果(详见train.m文件中的Step 5)。运行“print -djpeg weights.jpg”来保存文件“weights.jpg”。

提示：生成的结果与图6应该比较相似，如果出现类似于图7或其它情况可能是程序有问题，或者参数值有误。

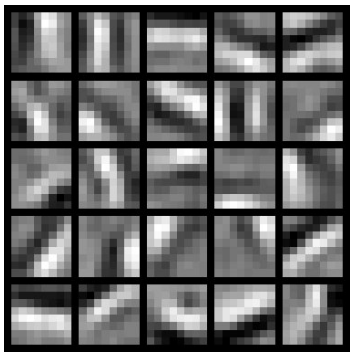

图 6: 稀疏自动编码器编码示例

## References


[1] 斯坦福大学的深度学习教程：http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial

[2] 稀疏自动编码器，wiki: https://en.wikipedia.org/wiki/Autoencoder

[3] 正则项，wiki：https://en.wikipedia.org/wiki/Norm_(mathematics)

[4] KL距离，wiki：https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

[5] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." science 313.5786 (2006): 504-507.

[6] 反向传播算法：http://ufldl.stanford.edu/wiki/index.php/Backpropagation_Algorithm

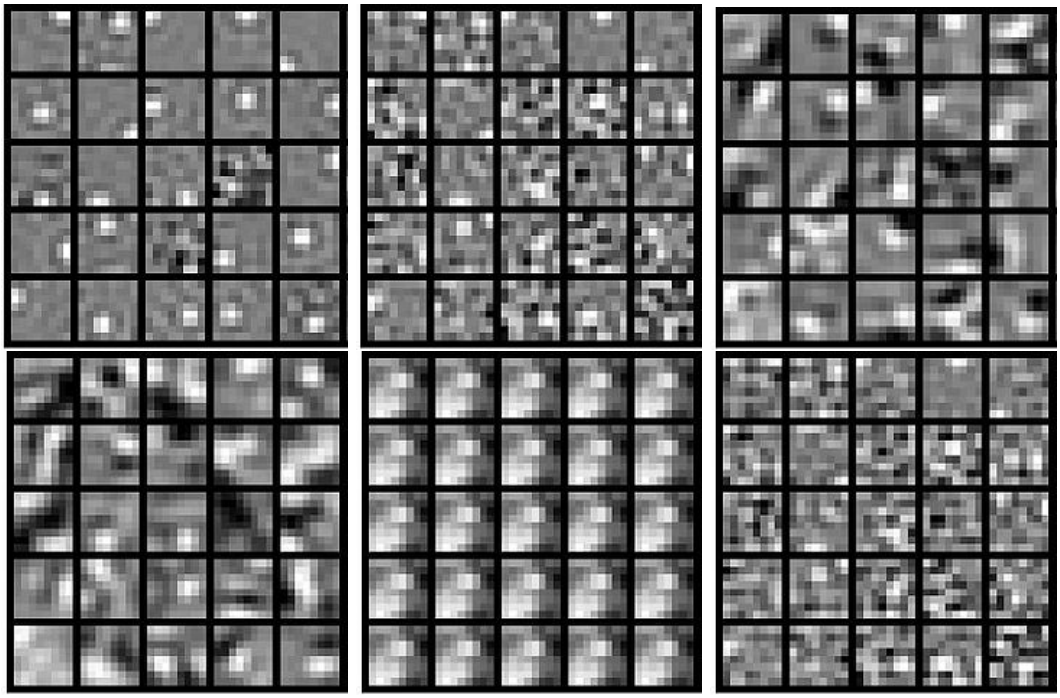

图 7: 有问题的示例

[7] 稀疏自动编码器的定义：http://ufldl.stanford.edu/wiki/index.php/Autoencoders_and_Sparsity