

**模式识别与机器学习**

**实验报告**

专业班级 计算机1405班

学生姓名 李强

学 号 0902140502

指导老师 梁毅雄

完成日期 2016年11月

实验二 ：神经网络模型

# 实验目的

1.利用神经网络实现对手写数字的自动化识别

2.利用已经训练好的神经网络参数，实现前馈传播算法预测

3.在已经实现的神经网络前馈传播的基础上实现饭向传播

# 实验环境

MATLAB\_R2014a(计算机系统为windows 10)

# 实验内容

part I神经网络：前馈传播

## 3.1 数据集

在ex3data1.mat中有500个手写字符的数据集，这些矩阵已经在程序中了，只需要一个载入命令即可。载入后，有正确维数和值的矩阵就出现在程序的存储中了。这5000个数据集都是20\*20的灰度字符图像。

## 3.2 显示数据

从显示数据集的子集开始，在 ex3.m的第一部分，代码实现显示功能，显示函数已被提供，运行后结果显示如下：



3.3 模型表示

神经网络有三大层次：输入层，隐藏层和输出层。任务中已经提供了一个已训练好的网络参数（θ1，θ2），参数存储在ex3weights.mat中，并且由ex3\_nn.m载入θ1和θ2中。每个神经元的“输入-输出”映射关系就是一个逻辑回归。在这里采用sigmoid函数，取值范围是[0,1]。g(z)=1/(1+exp()-z)，导数g’(z)=1-(g(z))^2。

## 3.4前馈传播和预测

现在实现神经网络的前馈传播。完成在predict.m中的代码来返回神经网络的预测。代码如下：

for i=1:m

Y(i,:) = sigmoid( [1 X(i,:)]\*Theta1');

end

for i=1:m

Z(i,:) = sigmoid( [1 Y(i,:)]\*Theta2');

end

for i=1:m

p(i) = find(Z(i,:)==max(Z(i,:)));

end

**part II 多变量线性回归**

## 3.5 前馈和代价函数

## 3.5.1非正则化代价函数

现在实现神经网络的代价函数和梯度，首先完成nnCostFunction.m中的代码以返回代价。非正则化的代价函数为：



其中k=10是标记的总数目。

在变量y中的原始标记是1,2,……,10，为了训练神经网络，将矢量改为仅含有0,1两个值的向量。比如，当是字符5的图像时，对应的应该是一个=1其他元素为0的十维向量。

## 3.5.2 正则化代价函数

正则化代价函数为：

 将正则化的代价项加到已有项上。

3.6 反向传播

在这个部分中，需要完成反向传播算法来计算神经网络代价函数的梯度。完成nnCostFunction.m来返回梯度的适当的值。一旦计算出梯度，就可以通过最小化代价函数来训练神经网络。先计算非正则化的梯度，正确后再计算正则化的梯度。

3.6.1 S梯度

首先需要实现S梯度函数。上文中提到S梯度函数为：

g’(z)=g(z)(1-g(z))

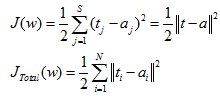
对于大量的Z值而言，梯度都应该接近于0。当z=0时，梯度值应正好等于0.25

3.6.2随机初始化

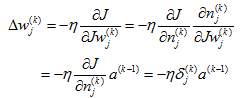
训练神经网络时，随机初始化已达到破对称是很重要的。

## 3.6.3 反向传播

现在完成反向传播算法。BP算法就是通过迭代优化网络的权值使得输出与输入之间的实际映射关系与所期望的映射关系一致，采用梯度下降算法通过调整各层权值求目标函数最小化。网络在某个或所有训练样本上的预测输出和期望输出的误差平方和：



由输出层误差逐层反向计算各层各单元的误差， 并基于梯度下降法修正各权值：



3.6.4 梯度检测

在神经网络中，最小化代价函数J(θ)。

3.7 可视化隐藏层

了解神经网络在学习什么的方法之一就是可视化隐藏层：

**for** ii**=**1**:**xSizeY

lastLayer**(**ii**,** y**(**ii**))** **=** 1**;**

**end**

jThetaMat **=** **(-**lastLayer **.\*** log**(**hThetaX**)** **-** **(**1 **-** lastLayer**)** **.\*** log**(**1 **-** hThetaX**))./**m**;**

jTheta **=** sum**(**jThetaMat**(:));**

jThetaMat **=** lambda**/**2 **.\***Theta1**(:,**2**:**input\_layer\_size**+**1**).\***Theta1**(:,**2**:**input\_layer\_size**+**1**)./**m**;**

jTheta **=** jTheta **+** sum**(**jThetaMat**(:));**

jThetaMat **=** lambda**/**2 **.\***Theta2**(:,**2**:**hidden\_layer\_size**+**1**).\***Theta2**(:,**2**:**hidden\_layer\_size**+**1**)./**m**;**

jTheta **=** jTheta **+** sum**(**jThetaMat**(:));**

delta3 **=** h2 **-** lastLayer**;**

delta2 **=** **(**Theta2**'** **\*** delta3**')'.\*** sigmoidGradient**([**ones**(**m**,**1**)** z2**]);**

Delta2 **=** zeros**(**num\_labels**,**hidden\_layer\_size**+**1**);**

Delta1 **=** zeros**(**hidden\_layer\_size**,**input\_layer\_size**+**1**);**

%Vector Method

Delta2 **=** Delta2 **+** delta3**'** **\*** **[**ones**(**m**,**1**)** a2**];**

Delta1 **=** Delta1 **+** delta2**(:,**2**:**hidden\_layer\_size**+**1**)'** **\*** **[**ones**(**m**,**1**)** X**];**

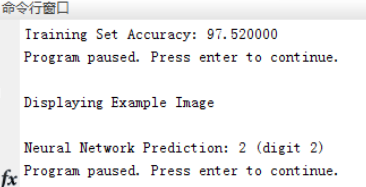
%Normalization

newTheta1 **=** **[**zeros**(**hidden\_layer\_size**,**1**)** Theta1**(:,**2**:**input\_layer\_size**+**1**)];**

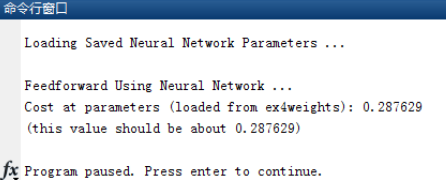
newTheta2 **=** **[**zeros**(**num\_labels**,**1**)** Theta2**(:,**2**:**hidden\_layer\_size**+**1**)];**

# 四、实验分析结果

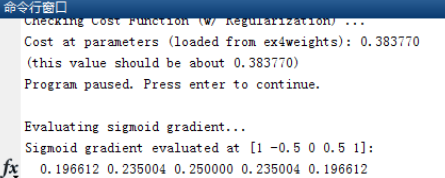
## 4.1 part1预测值



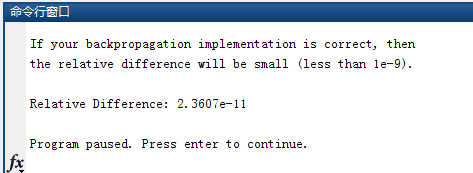
## 4.2 part2 非正则化代价值



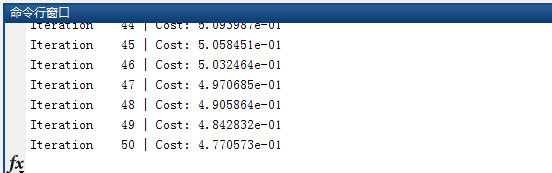
## 4.3正则化代价值：



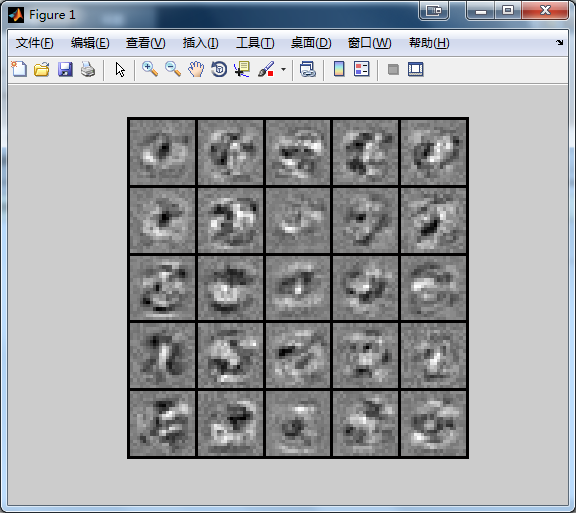
## 4.4 part2 相关性检验



## 4.5 part2 50次检验结果



## 4.6 特征图可视化



# 五、实验心得体会

本次实验在难度上比前一次实验难了很多，但是也真正地让我感受到了机器学习的魅力。神经网络模型的实现让自己对书上的知识有了更深刻，更完整的理解。