程序设计练习4：神经网络学习

机器学习

# 介绍

在本练习中，您将实现神经网络的反向传播算法，并将其应用于手写数字识别任务。在开始编程练习之前，我们强烈建议观看视频讲座并完成相关主题的复习问题。

要开始练习，您需要下载起始代码并将其内容解压缩到您希望完成练习的目录中。如果需要，在开始本练习之前，请使用octave/matlab中的cd命令更改到此目录。

您也可以在课程网站的“环境设置说明”中找到安装octave/matlab的说明。

## 本练习中包含的文件

ex4.m-八度/matlab脚本，引导您完成练习

ex4data1.mat-手写数字训练集

ex4weights.mat-练习4的神经网络参数

submit.m-提交脚本，将您的解决方案发送到我们的服务器displaydata.m-函数，帮助可视化数据集

fmincg.m-函数最小化例程（类似to fminunc）

sigmoid.m-sigmoid函数

computenumericalgradient.m-数值计算梯度

checknngradients.m-帮助检查梯度的函数

debuginitializewights.m-初始化权重的函数

predict.m-神经网络预测函数

[]sigmoidradiate.m-计算sigmoid函数的梯度*？*

[]randInitializeWeights.m-随机初始化权重*？*

[]nncostfunction.m-神经网络成本函数*？*

*？*指示需要完成的文件

在整个练习中，您将使用脚本ex4.m。这些脚本为问题设置数据集，并调用将要编写的函数。您不需要修改脚本。您只需要按照本分配中的说明修改其他文件中的函数。

## 在哪里寻求帮助

本课程的练习使用Octaveor Matlab，这是一种高级编程语言，非常适合数值计算。如果您没有安装八度音阶或matlab，请参考课程网站“环境设置说明”中的安装说明。[〔1〕](" \l "_ftn1" \o ")

在octave/matlab命令行中，键入帮助，然后键入函数名，显示内置函数的文档。例如，帮助图将显示用于绘制的帮助信息。有关八度函数的更多文档，请参见八度文档页面。matlab文档可以在matlab文档页面中找到。

我们还强烈鼓励使用在线讨论与其他学生讨论练习。但是，不要查看其他人编写的任何源代码，也不要与其他人共享源代码。

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image001.gif

# 1             神经网络

在前面的练习中，您实现了神经网络的前馈传播，并使用它来预测具有我们提供的权重的手写数字。在本练习中，您将实现反向传播算法来学习神经网络的参数。

提供的脚本ex4.m将帮助您逐步完成此练习。

## 1.1           可视化数据

在ex4.m的第一部分中，代码将通过调用函数display data加载数据并将其显示在二维图（图1）上。



图1：来自数据集的示例

这与上一个练习中使用的数据集相同。ex3data1.mat中有5000个培训示例，其中每个培训示例是数字的20像素乘20像素灰度图像。每个像素由一个浮点数表示，表示该位置的灰度强度。20×20的像素网格被“展开”成400维的矢量。这些训练示例中的每一个都成为数据矩阵X中的一行。这给了我们一个5000乘400的矩阵X，其中每一行都是手写数字图像的训练示例。

-（x（1））-*T*

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image003.gifD:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image004.gifD:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image005.gif-（x（2））-*T*

                                                                       ..

γ

-（

训练集的第二部分是包含训练集标签的5000维向量y。为了使事物与八度/matlab索引更兼容，在没有零索引的情况下，我们已经将数字0映射到值10。因此，“0”数字标记为“10”，而数字“1”到“9”则按其自然顺序标记为“1”到“9”。

## 1.2           模型表示

我们的神经网络如图2所示。它有三层——输入层、隐藏层和输出层。回想一下，我们的输入是数字图像的像素值。由于图像尺寸为20×20，因此我们得到了400个输入层单位（不包括总是输出+1的额外偏压单位）。培训数据将由ex4.m脚本加载到变量x和y中。

我们已经为您提供了一套我们已经培训过的网络参数（完成（1），完成（2））。它们存储在ex4weights.mat中，并将由ex4.m加载到theta1和theta2中。这些参数的尺寸适用于神经网络，第二层有25个单元，输出单元有10个（对应于10位数的类）。

|  |
| --- |
| %从文件加载中加载保存的矩阵（“ex4weights.mat”）；  %矩阵theta1和theta2现在将在您的工作区中  %Theta1的尺寸为25 x 401  %Theta2的尺寸为10 x 26 |

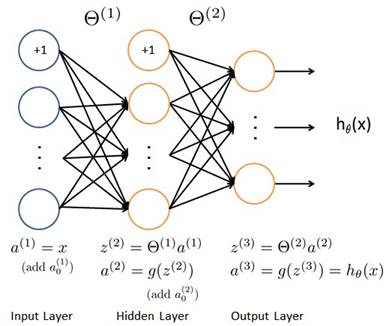


图2：神经网络模型。

## 1.3           前馈与成本函数

现在，您将实现神经网络的成本函数和梯度。首先，在nncostfunction.m中完成代码以返回成本。

回想一下，神经网络的成本函数（没有正则化）是

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image007.gif*,*

其中hθ（x（i））的计算如图2所示。k=10是可能的标签总数。注意这是第k个输出单元的激活（输出值）。另外，请记住，尽管原始标签（在变量中y）是1，2，…，10，为了训练神经网络，我们需要将标签重新编码为仅包含值0或1的向量，以便

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| γ  1  （0）            γ  *Y*＝0，  …γγ  γ  γ  0 | γ  0  （1）  γ  （0）  …γγ  γ  γ  0 | *...* | 或 | γ  0  0  （0）。  …γγ  γ  γ  1 |

例如，如果x（i）是数字5的图像，那么相应的y（i）（应该与成本函数一起使用）应该是一个10维向量，y5=1，其他元素等于0。

您应该实现对每个示例i计算hθ（x（i））的前馈计算，并对所有示例求和成本。您的代码还应该适用于任何大小的数据集，具有任意数量的标签（您可以假设始终至少有K≥3个标签）。

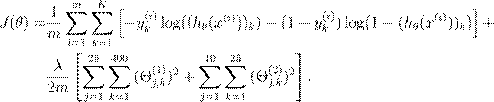
**实施说明：**矩阵x包含行中的示例（即x（i，：）&apos;是第i个训练示例x（i），表示为n×1向量。）当您在nncostfunction.m中完成代码时，需要将1的列添加到x矩阵中。神经网络中每个单元的参数在theta1和theta2中表示为一行。具体来说，theta1的第一行对应于第二层中的第一个隐藏单元。您可以使用for循环来计算成本。

完成后，ex4.m将使用为theta1和theta2加载的参数集调用nncostFunction。你应该知道费用大约是0.287629。

*现在您应该提交解决方案。*

## 1.4           规范化成本函数

正则化神经网络的代价函数由下式给出：



可以假设神经网络只有三层——输入层、隐藏层和输出层。但是，您的代码应该适用于任何数量的输入单元、隐藏单元和输出单元。虽然为了清晰起见，我们已经明确列出了上面关于完成（1）和完成（2）的索引，但是请注意，您的代码通常应该与任何大小的完成（1）和完成（2）一起工作。

请注意，您不应该调整与偏差相对应的术语。对于矩阵θ1和θ2，这对应于每个矩阵的第一列。现在您应该向成本函数添加正则化。请注意，您可以首先使用现有的nncostfunction.m计算未规范化的成本函数j，然后为规范化术语添加成本。

完成后，ex4.m将使用为theta1和theta2加载的参数集调用nncostFunction，并且λ=1。你应该看到成本大约是0.383770。

*现在您应该提交解决方案。*

# 2             反向传播

在本部分练习中，您将实现反向传播算法来计算神经网络成本函数的梯度。您需要完成nncostfunction.m，以便它返回一个适当的grad值。一旦你计算了梯度，你将能够训练神经网络，通过最小化成本函数j（完成），使用先进的优化器，如fmincg。

您将首先实现反向传播算法来计算（非规范化）神经网络参数的梯度。在您验证了非规范化情况下的梯度计算是正确的之后，您将为规范化神经网络实现梯度。

## 2.1           乙状结肠梯度

为了帮助您开始这部分练习，您将首先实现sigmoid渐变函数。sigmoid函数的梯度可以计算为

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image010.gif

哪里

乙状结肠

完成后，尝试通过在octave/matlab命令行调用sigmidgradient（z）来测试一些值。对于z的大值（正和负），梯度应该接近0。当z=0时，梯度应该正好为0.25。您的代码还应该使用向量和矩阵。对于矩阵，您的函数应该对每个元素执行sigmoid渐变函数。

*现在您应该提交解决方案。*

## 2.2           随机初始化

在训练神经网络时，随机初始化对称性破坏的参数是很重要的。随机初始化的一个有效策略是在范围内随机选择完成（）的值[

您应该使用这一范围的值，以确保参数保持较小并使学习更有效。

您的工作是完成randInitializeWeights.m以初始化完成的权重；修改文件并填写以下代码：

|  |
| --- |
| %将权重随机初始化为小值epsiloniit=0.12；  D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image014.gifW=兰特（L出，1+LD:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image015.gifin）\*[2]\*epsilonD:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image015.gif初始−epsilonD:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image015.gif初始化； |

*您不需要为此部分练习提交任何代码。*

## 2.3           反向传播

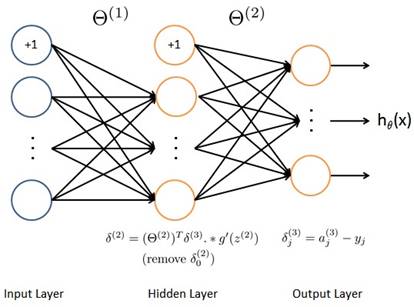


图3：反向传播更新。

现在，您将实现反向传播算法。回想一下，反向传播算法背后的直觉如下。给出一个训练示例（x（）t，y（）t），我们首先运行一个“正向传递”来计算整个网络中的所有激活，包括假设h完成（x）的输出值。然后，对于L层中的每个节点j，我们要计算一个“错误项”，它度量该节点对输出中的任何错误“负责”的程度。

对于输出节点，我们可以直接测量网络的激活与真实目标值之间的差异，并以此定义δj（3）

（因为第3层是输出层）。对于隐藏单元，您将根据层（L+1）中节点的错误项的加权平均值进行计算。

这里详细介绍了反向传播算法（也如图3所示）。您应该在一次处理一个示例的循环中实现步骤1到4。具体来说，您应该为t=1:m实现一个for循环，并将下面的步骤1-4放在for循环中，TTH迭代对TTH培训示例（x（t），y（t））执行计算。第5步将累积梯度除以m，得到神经网络成本函数的梯度。

1.      将输入层的值（a（1））设置为第t个培训示例x（t）。

执行前馈传递（图2），计算第2层和第3层的激活（Z（2）、A（2）、Z（3）、A（3））。请注意，您需要添加一个+1项，以确保层A（1）和层A（2）的激活向量也包括偏移单位。在八度/matlab中，如果1是列向量，则添加一个对应于1=[1；a 1]。

2.      对于第3层（输出层）中的每个输出单元k，设置

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image019.gif*,*

其中，yk∈0,1表示当前培训示例是属于k类（yk=1），还是属于其他类（yk=0）。您可能会发现逻辑数组对此任务很有帮助（在前面的编程练习中进行了解释）。

3.      对于隐藏层L=2，设置

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image020.gif

4.      使用下面的公式从这个例子中积累渐变。请注意，您应该跳过或删除。在八度/matlab中，删除对应于δ2＝δD:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image023.gif2（2：结束）。

∆（l）=∆（l）+δ（l+1）（a（l））*T*

5.      通过将累积梯度除以以下值，获得神经网络成本函数的（非调节）梯度：

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image025.gif

**八度/matlab提示：**只有在成功完成了前馈和成本函数之后，才能实现反向传播算法。在实现反向传播算法时，如果遇到维度不匹配错误（在Octave/Matlab中为“不符合参数”的错误），通常使用大小函数打印出正在处理的变量的大小是有用的。

在实现了反向传播算法之后，脚本ex4.m将继续对实现运行渐变检查。渐变检查将允许您增加对代码正确计算渐变的信心。

## 2.4           坡度检查

在你的神经网络中，你是在最小化成本函数j（完成）。要对参数执行渐变检查，可以想象将参数“展开”成一个长向量θ。通过这样做，您可以将成本函数想象为j（θ），并使用下面的梯度检查过程。

假设你有一个函数fi（θ），据称它可以计算；你想检查fi是否输出了正确的导数值。

                           让和

因此，θ（i+）与θ相同，只是它的第i个元素增加了。同样，θ（i−）是第i个元素减少的对应向量。现在，您可以通过检查每个i的以下内容，用数字验证fi（θ）的正确性：

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image029.gif*.*

这两个值之间的近似程度将取决于j的细节。但是假设，您通常会发现，上面的左右两侧将至少同意4个有效数字（通常更多）。

我们已经在compute numerical gradient.m中为您实现了计算数值梯度的功能。虽然您不需要修改该文件，但我们强烈建议您查看代码以了解它是如何工作的。

在ex4.m的下一步中，它将运行提供的函数checknngradients.m，它将创建一个小的神经网络和数据集，用于检查渐变。如果您的反向传播实现是正确的，

您应该看到一个小于1e-9的相对差。

**实用提示：**在进行梯度检查时，使用输入单元和隐藏单元数量相对较少的小神经网络，从而具有相对较少的参数，效率更高。θ的每个维度都需要对成本函数进行两次评估，这可能很昂贵。在函数checknnGradients中，我们的代码创建了一个小的随机模型和数据集，与ComputeNumericalGradient一起用于渐变检查。此外，在确信梯度计算正确之后，应该在运行学习算法之前关闭梯度检查。

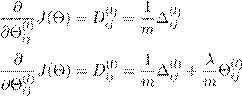
**实用提示：**渐变检查适用于计算成本和渐变的任何函数。具体来说，您可以使用相同的computeNumericalGradient.m函数来检查其他练习的渐变实现是否也正确（例如，逻辑回归的成本函数）。

*一旦您的成本函数通过（非规范）神经网络成本函数的梯度检查，您应该提交神经网络梯度函数（反向传播）。*

## 2.5           正则化神经网络

在成功地实现了反向传播算法之后，您将向渐变添加正则化。为了解释正则化，事实证明，在使用后向传播计算梯度之后，可以将其作为附加项添加。

具体来说，在使用后向传播计算∆（ijl）之后，应使用

j＝0

j＝1

请注意，您不应该将用于偏倚项的第一列“完成”（）l正规化。此外，在参数中，完成从1开始索引，而j从0开始索引。因此，

                                                                           （i）（l）

                                                                             完成1,0完成1,1*...*

                                                          （l）=\_\_（）\_2*我,*0      2*L,*1

                                                                         …………

有些令人困惑的是，八度/matlab中的索引从1开始（对于i和j），因此theta1（2，1）实际上对应于蒵（即，上面所示矩阵蒵（1）的第二行第一列中的条目）

现在修改在nncostFunction中计算渐变的代码，以说明正则化。完成后，ex4.m脚本将继续对实现运行渐变检查。如果代码正确，则应该看到小于1e-9的相对差异。现在您应该提交解决方案。

## 2.6           使用快速消费品学习参数

在成功地实现了神经网络成本函数和梯度计算之后，ex4.m脚本的下一步将使用fmincg学习一个好的集合参数。

培训完成后，ex4.m脚本将继续通过计算正确的示例百分比来报告分类器的培训精度。如果您的实现是正确的，您应该看到报告的培训精度约为95.3%（由于随机初始化，这可能会变化约1%）。通过训练神经网络进行更多的迭代，可以获得更高的训练精度。我们建议您尝试对神经网络进行更多迭代训练（例如，将maxiter设置为400），并改变正则化参数λ。有了正确的学习设置，神经网络就可以完全适应训练集。

# 3             可视化隐藏层

理解神经网络学习内容的一种方法是可视化隐藏单元捕获的表示。非正式地，给定一个特定的隐藏单元，一种可视化它计算的方法是找到一个输入x，它将使它激活（也就是说，有一个激活值

D:\document\convert_tasks\transweb\1731503_1743545\1731503.docx.files\image034.gif接近1）。对于您训练的神经网络，请注意，第i行完成（1）是一个401维向量，它表示第i个隐藏单元的参数。如果我们放弃偏压项，我们得到一个400维的向量，它表示从每个输入像素到隐藏单元的权重。

因此，将隐藏单元捕获的“表示”可视化的一种方法是将这个400维的矢量整形为20×20的图像并显示出来。ex4.m的下一步是使用displaydata函数来实现这一点，它将向您显示一个25个单元的图像（类似于图4），每个单元对应于网络中的一个隐藏单元。[〔3〕](" \l "_ftn3" \o ")

在经过训练的网络中，您应该发现隐藏的单元大致对应于在输入中查找笔画和其他模式的检测器。

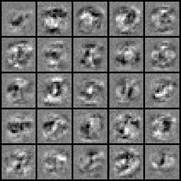


图4：隐藏单元的可视化。

## 3.1           可选（未分级）练习

在练习的这一部分中，您将尝试神经网络的不同学习设置，以了解神经网络的性能如何随正则化参数λ和训练步骤数而变化（

使用快速消费品时的最大选择）。

神经网络是非常强大的模型，可以形成高度复杂的决策边界。如果不进行正则化，神经网络可能会“过拟合”一个训练集，从而使其在训练集上获得接近100%的准确度，但在以前从未见过的新实例上却没有。您可以将正则化λ设置为较小的值，将maxiter参数设置为较高的迭代次数，以便自己看到这一点。

当你改变学习参数λ和maxiter时，你还可以自己看到隐藏单元的可视化效果的变化。

*您不需要为此可选（未分级）练习提交任何解决方案。*

# 提交和评分

完成任务的各个部分后，请务必使用提交功能系统将您的解决方案提交给我们的服务器。下面是这个练习的每个部分的评分方式的细目。

|  |  |  |
| --- | --- | --- |
| **部分** | **提交文件** | **点** |
| 前馈与成本函数 | nncost函数.m | 30点 |
| 规范化成本函数 | nncost函数.m | 15点 |
| 乙状结肠梯度 | 乙状结肠半径.m | 5点 |
| 神经网络梯度函数  （反向传播） | nncost函数.m | 40点 |
| 规则化梯度 | nncost函数.m | 10点 |
| 总分 |  | 100点 |

您可以多次提交解决方案，我们只考虑最高分数。

[〔1〕](" \l "_ftnref1" \o ")八度是matlab的免费替代品。对于编程练习，您可以自由使用八度或matlab。

[〔2〕](" \l "_ftnref2" \o ")一个有效的选择策略是根据网络中的单元数进行选择。一个很好的选择，其中lin=sl和lout=sl+1是

在层中靠近完成（l）的单元数。

[〔3〕](" \l "_ftnref3" \o ")结果表明，这相当于在给定输入“norm”约束（即kxk2≤1）的情况下，找到为隐藏单元提供最高激活的输入。