编程练习3：多元分类与神经网络

机器学习

# 介绍

在本练习中，您将实现一对多的逻辑回归和神经网络来识别手写数字。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

ex3.m-Octave/MATLAB脚本，引导您完成第1部分

ex3\_nn.m- Octave /MATLAB脚本，引导您完成第2部分

ex3data1.mat-手写数字训练集

ex3weights.mat-神经网络练习的初始权重

submit.m-将解决方案发送到我们的服务器的提交脚本

displayData.m-帮助可视化数据集的函数

fmincg.m-函数最小化例程（类似于fminunc）

sigmoid.m-sigmoid函数

* lrCostFunction.m -逻辑回归代价函数
* oneVsAll.m -训练一个一对多的多元分类器
* predictOneVsAll.m -使用一对多得多元分类器进行预测
* predict.m-利用神经网络进行预测的函数
* 表示该文件需要您来完成

在整个练习中，您将使用脚本ex3.m和ex3\_nn.m这些脚本为问题设置数据集，并调用将要编写的函数。您不需要修改这些脚本。您只需要按照本分配中的说明修改其他文件中的函数。

# 1             多元分类

对于这个练习，您将使用逻辑回归和神经网络来识别手写数字（从0到9）。自动手写数字识别在今天被广泛使用，从识别邮包上的邮政编码到识别银行支票上的金额。本练习将向您展示如何将所学方法用于此分类任务。

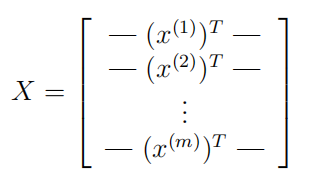
在练习的第一部分中，您将扩展先前的逻辑回归实现，并将其应用于一对多的分类。

## 1.1           数据集

在ex3data1.mat中有一个包含5000个手写数字训练样本的数据集，.mat格式意味着数据已保存为本机Octave/MATLAB矩阵格式，而不是像csv文件那样的文本（ASCII）格式。这些矩阵可以使用load命令直接读取到您的程序中。加载后，具有正确维度和值的矩阵将出现在程序内存中。矩阵将已经命名，因此您不需要为它们指定名称。

|  |
| --- |
| %从文件中加载已保存好的矩阵  Load(’ex3data1.mat’);  %矩阵x和y将会在你的Octave环境中 |

ex3data1.mat中有5000个训练样本，其中每个训练样本是数字的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样本。



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Octave/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”%也就是说用数字10表示该样本的预测值为0，而数字“1”到“9”则按其自然顺序标记为“1”到“9”。

## 1.2           可视化数据

您将从可视化训练集的一个子集开始。在ex3.m的第1部分中，代码从x中随机选择100行，并将这些行传递给displayData函数。此函数将每行映射为20像素乘20像素的灰度图像，并将这些图像显示在一起。我们已经提供了displayData函数，我们鼓励您检查代码，看看它是如何工作的。运行此步骤后，您应该看到类似的图像



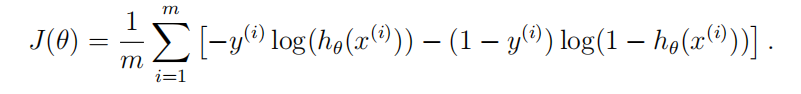
图1：来自数据集的样本

## 1.3           向量化逻辑回归

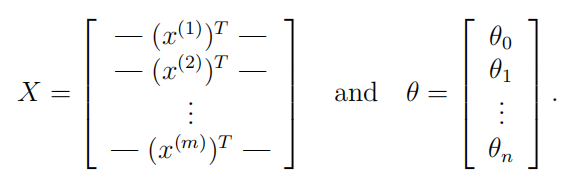
您将使用多个一对多的逻辑回归模型来构建一个多类分类器。由于有10个类，您需要培训10个单独的逻辑回归分类器。为了使这种培训有效，确保代码被很好地向量化是很重要的。在本节中，您将实现逻辑回归的矢量化版本，该版本不使用任何for循环。您可以将上一个练习中的代码用作此练习的起点。

## 1.3.1          代价函数矢量化

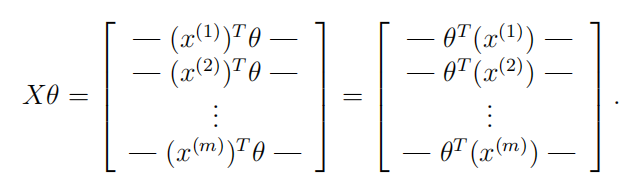
我们将从编写代价函数的矢量化版本开始。回想一下，在（非规范的）逻辑回归中，代价函数是

**

要计算求和中的每个元素，我们必须为每个示例i计算，其中并且是sigmoid函数。事实证明，我们可以用矩阵乘法快速地计算出所有的例子。让我们把x和θ定义为



然后，通过计算矩阵积*Xθ*，我们得到



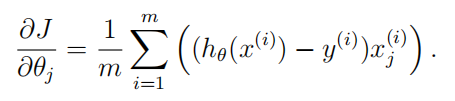
在上一个等式中，我们使用了这样一个事实：如果a和b是向量，则。这使得我们能够在一行代码中计算所有示例i的结果

您的工作就是在lrCostFunction.m文件中编写非规范化的代价函数。您的实现过程应该使用上面介绍的策略来计算。对于代价函数的其余部分，您也应该使用向量化方法。lrCostFunction.m的完整矢量化版本不应包含任何循环语句。

（提示：在编写此函数时您可能需要使用对应位置的元素相乘运算（.\*）和求和运算sum）

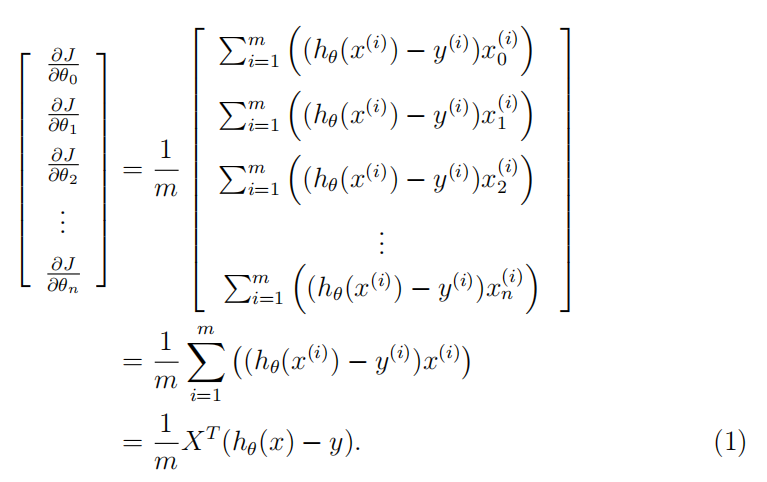
## 1.3.2         矢量化梯度

回想一下，（非规范的）逻辑回归代价函数的梯度是一个向量，其中第j个元素定义为

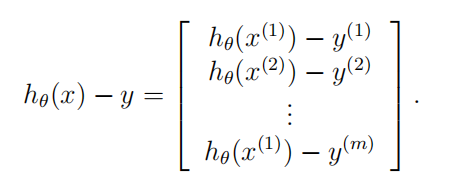
**

为了在数据集上向量化此操作，我们首先写出

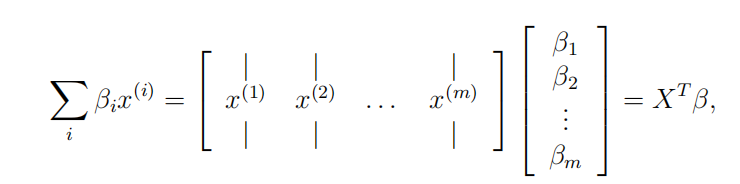
所有的偏导数，



其中

**

注意是一个向量，而是一个标量（单个数字）。为了理解推导的最后一步，令，并观察：



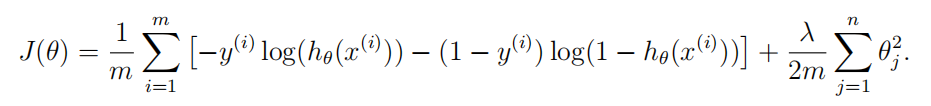
其中，值

上面的表达式允许我们计算所有的偏导数而不需要任何循环。如果您使用线性代数的方法还算顺手，我们鼓励您通过上面的矩阵乘法来说服自己，矢量化的版本可以执行相同的计算。现在应该实现公式（1）来计算正确的矢量化梯度。完成后，通过实现梯度计算来完成函数lrCostFunction.m。

**调试提示：**对代码进行矢量化有时很棘手。调试的一个常见策略是打印出使用size函数处理的矩阵的大小。例如，给定一个维度为100×20（100个例子，20个特征）的数据矩阵X和一个维度为20×1的向量θ，可以观察到是一个有效的乘法运算，而不是。此外，如果您有代码的非矢量化版本，您可以比较矢量化代码和非矢量化代码的输出，以确保它们产生相同的输出。

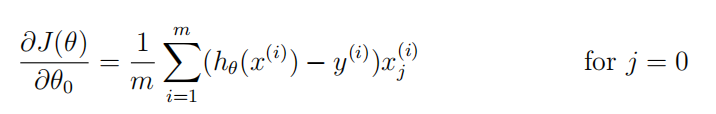
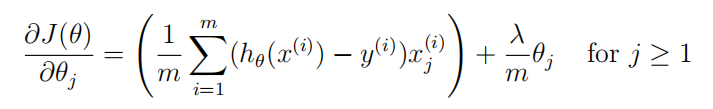
## 1.3.3          正则逻辑回归的矢量化

在为逻辑回归实现矢量化之后，现在您将向代价函数添加正则化。回想一下，对于正规化的逻辑回归，代价函数定义为



请注意，不应将作为偏置单元的正则化。

相应地，的正则化逻辑回归代价函数的偏导数定义为

现在，在lrCostFunction中修改代码，以构成正则化。

同样，您不应该在代码中放入任何循环。

**Octave/MATLAB提示：**在对正则化逻辑回归实现矢量化时，您可能常常只需要求和并更新θ的某些元素。在Octave/MATLAB中，可以索引到矩阵中，只访问和更新某些元素。例如，a（：，3:5）=b（：，1:3）将用b中的列1到3替换a中的列3到5。可以在索引中使用的一个特殊关键字是索引中的end关键字。这允许我们选择列（或行），直到矩阵结束。例如，a（：，2:end）将只返回a的第2列到最后一列的元素。因此，可以将它与sum和.^操作一起使用，只计算您感兴趣的元素的总和（例如sum（z（2:end）.^2））。在起始代码lrCostFunction.m中，我们还提供了关于计算正则化梯度的另一种可能方法的提示。

*现在您应该提交解决方案。*

## 1.4           一对多的多元分类

在本部分练习中，您将通过训练多个正则化的逻辑回归分类器来实现一个一对多的多元分类，每个分类器对应于数据集中的K类（图1）。在手写数字数据集中，K=10，但是您的代码应该适用于K的任何值（意思可能是补进去的代码应该适用于各种数量的类别，比如可能需要做100种分类）。

现在应该在oneVsAll.m中完成代码，为每个类训练一个分类器。特别是，您的代码应该返回一个矩阵（）中的所有分类器参数，其中每一行参数对应于一个类的已学习的逻辑回归参数。您可以使用从1到k的“for”循环来完成这项工作，独立地训练每个分类器。

注意，这个函数的y参数是从1到10的标签向量，我们已经将数字“0”映射到标签10（以避免索引混淆）。

当训练类的分类器时，您将需要标签y的m维向量，其中表示第j个（j-th）训练实例是属于*k*类（=1），或者如果它属于不同的类（=0）。您可能会发现逻辑数组对此任务很有帮助。

**Octave/MATLAB提示：**Octave/MATLAB中的逻辑数组是包含二进制（0或1）元素的数组。在Octave/MATLAB中，计算向量a（大小为m×1）和标量b的表达式a==b将返回与a大小相同的向量，其中a的元素等于b的位置为1，不同的位置为0。要了解这对您自己是如何工作的，请尝试在Octave/MATLAB中使用以下代码：

a=1:10；%创建a和b；

b=3；

a==b %您应该在这里尝试b的不同值

此外，您将使用fmincg进行此练习（而不是fminunc）。fmincg的工作原理与fminunc类似，但在处理大量参数时效率更高。

在正确完成oneVsAll.m的代码之后，脚本ex3.m将继续使用oneVsAll函数来训练多类分类器。

*现在您应该提交解决方案。*

### 1.4.1          一对多的预测

训练完一对多的分类器之后，您现在可以使用它来预测给定图像中包含的数字。对于每个输入，您应该使用经过训练的逻辑回归分类器计算它属于每个类的“概率”。您的一对所有预测函数将选择相应的逻辑回归分类器输出最高概率的类，并返回类标签（1、2……或K）作为输入示例的预测。

现在您应该完成predictOneVsAll.m中的代码，使用一对多分类器进行预测。

一旦你完成了，ex3.m将使用所学的值调用你的predictOneVsAll.m函数。你应该看到训练集的准确性大约为94.9%（即对培训集中94.9%的示例进行正确分类）。

*现在您应该提交解决方案。*

D:\document\convert_tasks\transweb\1731504_1743546\1731504.docx.files\image002.gif

# 2             神经网络

在本练习的前一部分中，您实现了多类逻辑回归来识别手写数字。然而，逻辑回归不能形成更复杂的假设，因为它只是一个线性分类器。[〔3〕](" \l "_ftn3" \o ")

在练习的这一部分中，您将使用与以前相同的训练集实现一个神经网络来识别手写数字。神经网络将能够表示形成非线性假设的复杂模型。本周，你将使用我们已经训练过的神经网络的参数。您的目标是实现前馈传播算法，使用我们的权重进行预测。在下周的练习中，您将编写学习神经网络参数的反向传播算法。

提供的脚本ex3D:\document\convert_tasks\transweb\1731504_1743546\1731504.docx.files\image001.gifnn.m会帮你完成这个练习。

## 2.1           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个输入层单元（不包括总是输出+1的额外偏置单元）。如前所述，培训数据将加载到变量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（）。它们存储在ex3weights.mat中，由ex3\_nn.m加载到Theta1和Theta2，参数的尺寸为神经网络的尺寸，第二层有25个单元，输出10个单元（对应于10位的类）。

|  |
| --- |
| %从文件中加载已被保存的矩阵  Load(‘ex3weights.mat’);  %矩阵Theta1和Theta2将出现在你的Octave环境中  %Theta1的尺寸为25 x 401  %Theta2的尺寸为10 x 26 |

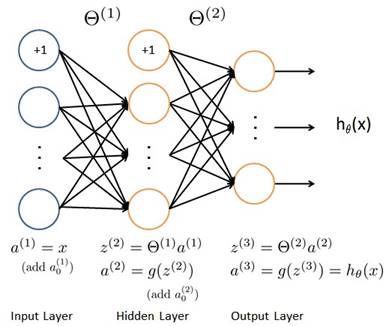


图2：神经网络模型。

## 2.2           前向传播与预测

现在，您将为神经网络实现前向传播。您需要完成predict.m中的代码以返回神经网络的预测。

您应该实现对每个示例i计算并返回相关预测的前向计算。与一对多的分类策略类似，神经网络的预测将是输出最大的标签。

**实施说明：**矩阵X以行的形式包含示例。完成predict.m中的代码后，需要将1的列添加到矩阵中（添加全是1的数作为第一列）。矩阵Theta1和Theta2以行的形式包含每个单元的参数。具体来说，Theta1的第一行对应于第二层中的第一个隐藏单元。在Octave/MATLAB中，当你计算时，确保你正确地索引（并且如果必要的话，转置）X，这样你会得到作为列向量。

完成后，ex3\_nn.m将使用为Theta1和Theta2加载的参数集调用预测函数。你应该看到准确率约为97.5%。之后，交互式序列将启动显示训练集中的图像，一次一个，同时控制台打印出显示图像的预测标签。要停止图像序列，请按ctrl-c。

*现在您应该提交解决方案。*

# 提交和评分

完成此任务后，请务必使用提交功能将您的解决方案提交到我们的服务器。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** |
| 规范化逻辑回归 | lrCostFunction.m | 30点 |
| 一对多的分类器训练 | oneVsAll.m | 20点 |
| 一对多的分类器预测 | predictOneVsAll.m | 20点 |
| 神经网络预测函数 | predict.m | 30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 ")Octave是MATLAB的免费替代品。对于编程练习，您可以自由使用Octave或MATLAB。

[〔2〕](" \l "_ftnref2" \o ")这是mnist手写数字数据集的子集（http://yann.lecun.com/exdb/mnist/）。

[〔3〕](" \l "_ftnref3" \o ")您可以在逻辑回归中添加更多的特性（如多项式特性），但这对于训练来说可能非常昂贵。