编程练习 3：

多类分类和神经网络

机器学习

# 介绍

在本练习中，您将实现一对全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，我们强烈建议您观看视频讲座并完成相关主题的复习题目。

要开始练习，您需要下载入门代码并将其内容解压缩到要完成练习的目录中。如果需要，请使用 Octave/MATLAB 中的 cd 命令更改为此目录，以便开始本练习。

您还可以在课程网站的“环境设置说明”中找到有关安装Octave / MATLAB的说明。

## 本练习中包含的文件

ex3.m - Octave/MATLAB 脚本，可引导您完成第 1 部分 ex3 nn.m - Octave/MATLAB 脚本，引导您完成第 2 部分

ex3data1.mat - 手写数字的训练集 ex3weights.mat - 神经网络练习 submit.m 的初始权重 submit.m - 将您的解决方案发送到我们的服务器的提交脚本 displayData.m - Function 以帮助可视化数据集 fmincg.m - 函数最小化例程（类似于 fminunc） sigmoid.m - Sigmoid 函数

[*?*] lrCostFunction.m - Logistic 回归成本函数

[*?*] oneVsAll.m - 训练一个一对多类分类器

[*?*] predictOneVsAll.m - 使用一对全多类分类器进行预测

[*?*] predict.m - 神经网络预测函数

*?* 表示您需要完成的文件

在整个练习过程中，您将使用 脚本 ex3.m 和 ex3 nn.m。这些脚本为问题设置数据集，并调用您将编写的函数。您不需要修改这些脚本。您只需按照此赋值中的指令离子修改其他文件中的函数。

## 从何处获取帮助

本课程中的练习使用Octave[[1]](#footnote-1) 或MATLAB，这是一种非常适合数值计算的高级编程语言。如果您没有安装Octave或MATLAB，请参阅课程网站“环境设置说明”中的安装说明。

在 Octave/MATLAB 命令行中，键入 help 后跟函数名称将显示内置函数的文档。例如，帮助绘图将显示绘图的帮助信息。有关倍频程函数的更多文档，请参阅[倍频程文档页面。](http://www.gnu.org/software/octave/doc/interpreter/) MATLAB 文档可以在 [MATLAB 文档说明页面上](http://www.mathworks.com/help/matlab/?refresh=true)找到[。](http://www.mathworks.com/help/matlab/?refresh=true)

我们还强烈建议使用在线 **讨论** 与其他学生讨论练习。但是，不要查看他人编写的任何源代码或与他人共享您的源代码。

# 多类分类

在本练习中，您将使用逻辑回归和神经网络来识别手写数字（从 0 到 9）。如今，自动手写数字识别被广泛使用 - 从识别 邮件信封上的邮政编码（邮政编码）到识别银行支票上写的金额。本练习将向您展示如何将所学方法用于此分类任务。

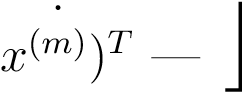
在本练习的第一部分中，您将扩展之前 实现 的 logistic 回归，并将其应用于一对一分类。

## 数据

在 ex3data1.mat 中为您提供了一个数据集，其中包含 5000 个手写数字的训练示例。[[2]](#footnote-2) .mat 格式意味着数据已以本机 Octave/MATLAB 矩阵格式保存，而不是像 csv 文件那样的文本 （ASCII） 格式。可以使用 load 命令将这些矩阵直接读入程序中 。加载后，正确尺寸和值的矩阵将出现在程序的内存中。矩阵已经命名，因此您无需为它们分配名称。

|  |
| --- |
| % 从文件加载中加载保存的矩阵 （'ex3data1.mat'）;  % 矩阵 X 和 y 现在将位于您的 Octave 环境中 |

ex3data1.mat 中有 5000 个训练示例，其中每个训练示例都是数字的 20 x 20 像素灰度图像。每个像素由一个浮点数表示，指示该位置的灰度强度。20 x 20 像素网格被“展开”成一个 400 维矢量。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5000×400矩阵X，其中每一行都是一个训练示例，用于手写数字图像。

 — （*x*（1））*T* —  — （*x*（2））*T* — 

 .. 



— (

训练集的第二部分是一个 5000 维向量 y ，其中包含训练集的标签。为了使内容与Octave / MATLAB索引更加兼容，其中没有零索引，我们将数字零映射到值十。因此，“0”数字标记为“10”，而数字“1”到“9” 按自然顺序标记为“1”到“9”。

## 可视化数据

您将首先可视化训练集的子集。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。此函数将每行映射到 20 x 20 像素的灰度图像，并一起显示这些图像。我们提供了 displayData 函数，我们鼓励您检查代码以了解其工作原理。运行此步骤后，您应该会看到一个类似于 Figure 的图像

1.



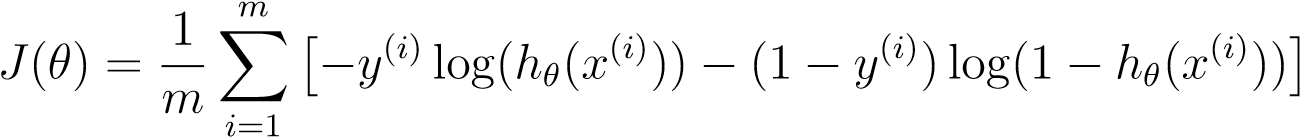
图 1：数据集中的示例

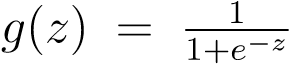
## 矢量化逻辑回归

您将使用多个一对全逻辑回归模型来构建多类分类器。由于有 10 个类，因此您需要训练 10 个单独的逻辑回归分类器。为了使这种训练高效，重要的是要确保你的代码被很好地矢量化。在本节中，您将实现不采用任何 for 循环的逻辑回归的矢量化版本 。您可以使用上一个练习中的代码作为本练习的起点。

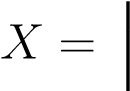
### 矢量化成本乐趣

我们将首先编写成本函数的矢量化版本。回想一下，在（未计量的）逻辑回归中，成本函数为

*.*

为了计算求和中的每个元素，我们必须为每个示例*i*计算*hθ*（*x*（*i*）），其中*hθ*（*x*（*i*）） = *g*（*θTx*（*i*）））和 是乙状结肠函数。事实证明，我们可以通过使用矩阵乘法为所有示例快速计算出这一点。让我们将 *X* 和 *θ* 定义为

 — （*x*（1））*T* —   *θ*0 

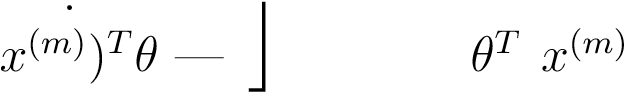
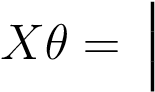
 — （*x*（2））*T* —   *θ*1 

 ...  和 *θ* =  ... *。*

— （*x*（*m*））*T* — *θn*

然后，通过计算矩阵积 *Xθ*，我们有

 — （*x*（1））*Tθ* —   — *θT*（*x*（1）） — 

— （*x*（2））*Tθ* —   — *θT*（*x*（2）） — 

..  =  ... *。*

 

— （ — （ ） —

在最后一个等式中，我们使用了*这样一*个事实，即如果 a 和 *b* 是向量，则 *a*  T b = *b Ta*。这允许我们在 一行代码中计算所有示例 *i* 的乘积 *θTx*（*i*）。

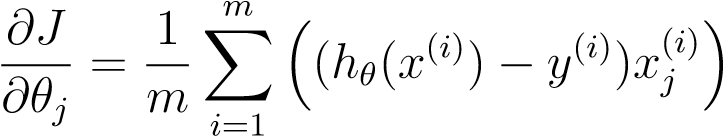
您的工作是在文件 lrCostFunction.m 中编写未本地化的成本函数，您的实现应使用我们上面介绍的策略来计算 *θTx*（*i*）。还应对成本函数的其余部分使用矢量化方法。lrCostFunction.m 的完全矢量化版本不应包含任何循环。

（提示：你可能会使用按元素的乘法运算

（.\*） 和求和运算 编写此函数时的总和）

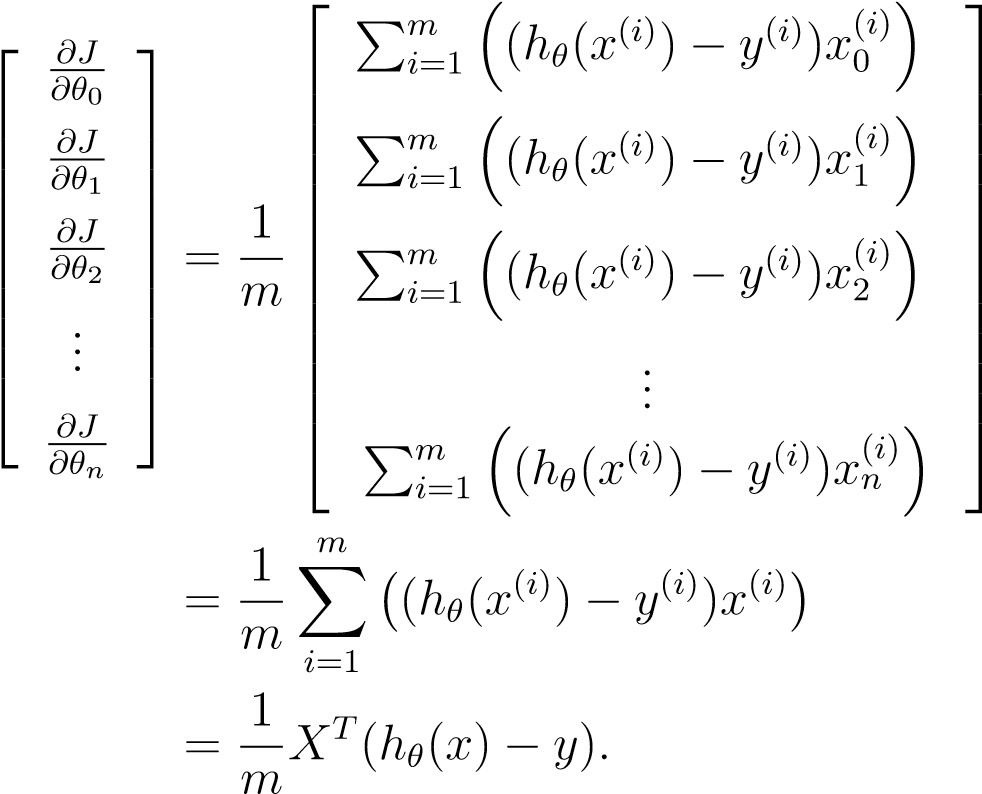
### 矢量化渐变

回想一下，（未计量的）逻辑回归成本的梯度是一个向量，其中第 *j*个 元素定义为

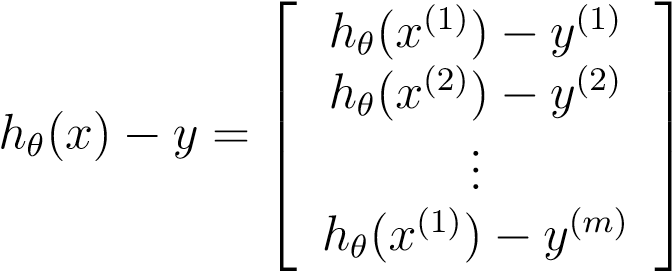
 *.*

为了在数据集上对此操作进行矢量化，我们首先写出所有

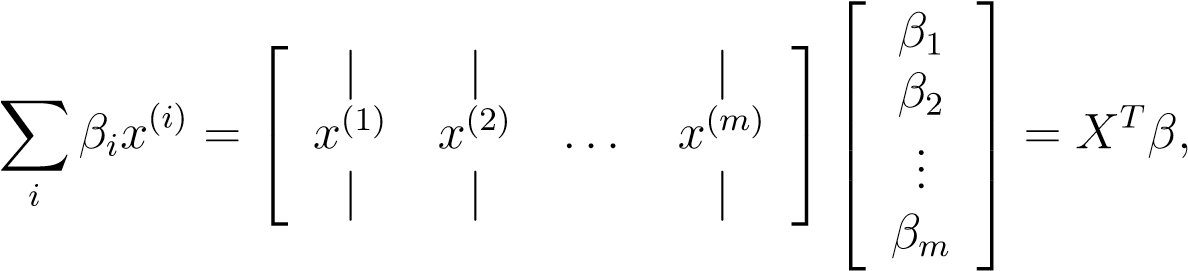
显式表示所有 *θj* 的偏导数，

 (1)

where

*.*

请注意，*x*（*i*） 是一个向量，而 （*hθ*（*x*（*i*））−*y*（*i*）） 是一个标量（单个数字）。为了理解推导的最后一步，设*βi* = （*hθ*（*x*（*i*）） − *y*（*i*））并观察：



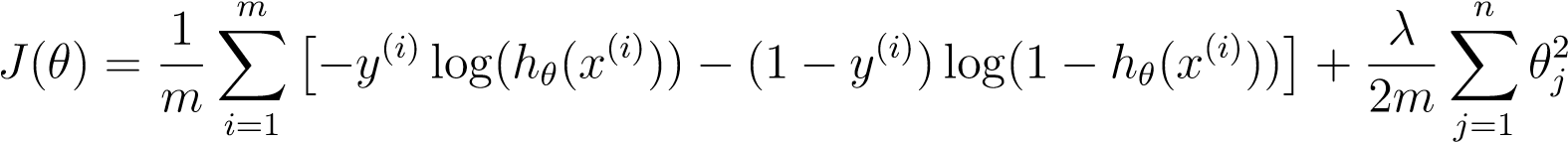
其中值 *βi* = （*hθ*（*x*（*i*）） − *y*（*i*））。

上面的表达式允许我们在没有任何循环的情况下计算所有偏导数。如果您对线性代数感到满意，我们建议您通过上面的矩阵乘法来说服自己矢量化版本执行相同的计算。您现在应该实现等式1来计算正确的矢量化梯度。完成后， 通过实现梯度来完成函数lrCostFunction.m。

**调试提示：**矢量化代码有时可能很棘手。调试的一种常见策略是使用 size 函数打印出您正在使用的矩阵的大小 。例如，给定一个大小为 100 × 20（100 个示例，20 个特征）和 *θ*（维度为 20×1 的向量）的数据矩阵 *X*，您可以观察到 *Xθ* 是有效的乘法运算，而 *θX* 不是。此外，如果您有非矢量化版本的代码，则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。

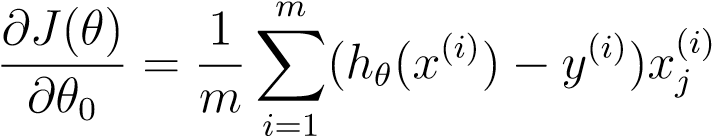
### 矢量化调节调高逻辑回归

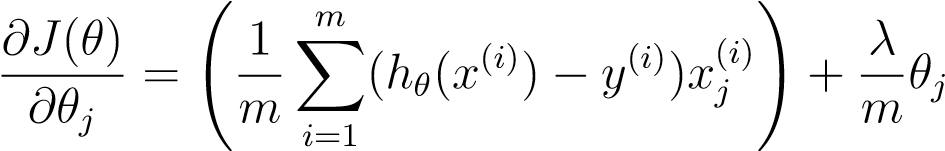
为逻辑回归实现矢量化后，现在将正则化添加到成本函数。回想一下，对于正则化逻辑回归，成本函数定义为

*.*

请注意，您*不应*正则化用于偏置项的 *θ*0。

相应地，*θj* 的正则化逻辑回归成本的偏导数定义为

 对于 *j* = 0

 对于 *j* ≥ 1

现在，在 lrCostFunction 中修改代码以考虑正则化。

同样，您不应该在代码中放入任何循环。

**Octave/MATLAB 提示：**在为正则化逻辑回归实现矢量化时，您可能通常只想对 θ 的某些元素求和和更新 。在Octave/MATLAB中，您可以索引到矩阵中以仅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 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），直到矩阵的末尾。例如，A（：， 2：end） 将仅返回 A *的第*  2 列到最后一列中的元素。因此，您可以将它与总和和一起使用。^ 运算e 仅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，我们还提供了另*一种*计算正则化梯度的方法的提示。

*您现在应该提交您的解决方案。*

## 一对一分类

在本部分练习中，您将通过训练多个正则化逻辑回归分类器来实现一对一分类，每个分类器对应于我们数据集中的每个 K 类（图 1）。在手写数字数据集中，*K* = 10，但您的代码应该适用于 K 的任何值。

现在，您应该在 oneVsAll.m 中完成代码，以便为每个类训练一个分类器。特别是，您的代码应返回 matrix Θ 中的所有分类器参数，∈ R*K*×（*N*+1），  其中 Θ 的每一行对应于一个类的学习逻辑回归参数。您可以使用从 1 到 K 的“for”循环来执行此操作，并独立训练每个分类器。

请注意，此函数的 y 参数是从 1 到 10 的标签向量，其中我们已将数字“0”映射到标签 10（以避免与索引混淆）。

训练类 *k*  的分类器时，∈ {1*,...，K*}，您将需要标签 *y* 的 *m*维向量，其中 *yj* ∈ 0*，*1 表示第 *j* 个 training 实例是否属于类 *k* （*yj* = 1），或者它是否属于其他类 （*yj* = 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，为向量a（大小为*m*×1）和标量b计算表达式a == b将返回与a大小相同的向量 ，其中a的elements 等于b，零的位置为a。要了解这如何为自己工作，请尝试Octave / MATLAB中的以下代码：

a = 1：10;% 创建 a 和 b b = 3;

a == b % 您应该在这里尝试不同的 b 值

此外，您将使用fmincg进行此练习（而不是fminunc）。 fmincg的工作方式与fminunc类似，但对于 处理大量参数更有效。

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

*您现在应该提交您的解决方案。*

### 一对一预测

训练一对全分类器后，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”。一对全预测函数将选取相应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，以使用一对全分类器进行预测。

完成后，ex3.m 将使用学习到的 Θ 值调用 predictOneVsAll 函数。您应该看到训练集的准确性约为

94.9%（即，它正确地对训练集中94.9%的示例进行了分类）。

*您现在应该提交您的解决方案。*

# 神经网络

在本练习的前一部分中，您实现了多类逻辑回归以识别手写数字。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]](#footnote-3)

在本部分练习中，您将实现一个神经网络，以使用与之前相同的训练集来识别手写数字。神经网络将能够表示形成非线性假设的比较法模型。本周，您将使用我们已经训练过的神经网络中的参数。您的目标是实现前馈传播算法，以使用我们的权重进行预测。在下周的 exercise 中，您将编写用于学习神经网络参数的反向传播算法。

提供的脚本 ex3 nn.m 将帮助您逐步完成本练习。

## 模型表示

我们的神经网络如图 2 所示。它有3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。由于图像的大小为20×20，因此为我们提供了400个输入层单位（不包括始终输出+1的e xtra偏置单位）。与之前一样，训练数据将加载到变量 X 和 y 中。

您已经为您提供了一组已由我们训练的网络参数（Θ（1），Θ（2））。这些参数存储在 ex3weights.mat 中，并由 ex3 nn.m 加载到 Theta1 和 Theta2 中 这些参数的尺寸适合神经网络，第二层有 25 个单位，10 个输出单位（对应于 10 位数字类）。

|  |
| --- |
| % 从文件加载中加载保存的矩阵 （'ex3weights.mat'）;  % 矩阵 Theta1 和 Theta2 现在将位于您的八度音阶中  % 环境  % Theta1 的尺寸为 25 x 401  % Theta2 的尺寸为 10 x 26 |

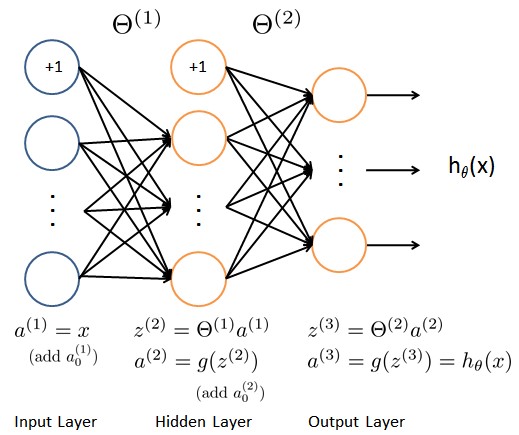


图 2：神经网络模型。

## 前馈传播和预测

现在，您将为神经网络实现前馈传播。您需要在 predict.m 中完成代码才能返回神经网络的预测。

您应该实现e 前馈计算，为每个示例 i 计算 *hθ*（*x*（*i*）） 并返回关联的预测。与一对全分类策略类似，来自神经网络的预测将是具有最大输出（*hθ*（*x*））*k*的标签。

**实现说明：**矩阵 X 包含行中的示例。在 predict.m 中完成代码时，需要将 1 列添加到矩阵中。矩阵 Theta1 和 Theta2 在行中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，当您计算*z*（2） = Θ（1）*a*（1）时，请确保正确索引（并在必要时转置）X，以便获得*a*（*l*）作为列向量。

完成后，ex3 nn.m 将使用为 Theta1 和 Theta2 加载的参数ers 集来调用您的预测函数。您应该看到准确率约为97.5%。之后，将启动一个交互式序列，一次显示一个训练集中的图像，而控制台则为显示的图像打印出预测的标签。若要停止图像序列，请按 Ctrl-C。

*您现在应该提交您的解决方案。*

# 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给我们的服务器。以下是本练习每个部分的评分方式的细分。

|  |  |  |
| --- | --- | --- |
| **部分** | **提交的文件** | **点** |
| 正则化 逻辑 回归 | lrCostFunction.m | 30 积分 |
| 一对一分类器培训 | 一个VAll.m | 20 积分 |
| 一对全分类器预测 | predictOneVsAll.m | 20 积分 |
| 神经网络预测函数 | 预测.m | 30 积分 |
| 总积分 |  | 100 积分 |

您可以多次提交解决方案，我们将仅考虑最高分。

1. 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。 [↑](#footnote-ref-1)
2. 这是 MNIST 手写数字数据集 （[http://yann.lecun.com/exdb/mnist/)](http://yann.lecun.com/exdb/mnist/). [↑](#footnote-ref-2)
3. 您可以向逻辑回归添加更多特征（如多项式特征），但在可以非常昂贵的训练。 [↑](#footnote-ref-3)