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，您将实现一对多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。

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。如果需要，在开始本练习之前使用 Octave/MATLAB 中的 cd 命令切换到该目录。

您还可以在课程网站的“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 - 引导您完成第 1 部分的 Octave/MATLAB 脚本 ex3 nn.m - 引导
您完成第 2 部分的 Octave/MATLAB 脚本 ex3data1.mat - 训练手写数字集
ex3weights.mat - 神经网络的初始权重练习 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 - 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 - 物流回归成本函数 [?] oneVsAll.m - 训练一对多的多类
分类器 [?] predictOneVsAll.m - 使用一对多的多类分类器进行预测 [?]
predict.m - 神经网络预测功能

?表示您需要完成的文件

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您将编写的函数。您不需要修改这些脚本。您只需按照本作业中的说明修改其他文件中的函数。

从哪里获得帮助本课程的练

习使用 Octave1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,请参考课程网站“环境设置说明”中的安装说明。

在 Octave/MATLAB 命令行中,键入 help 后跟一个函数名称会显示内置函数的文档。例如,help plot 会调出绘图的帮助信息。 Octave 函数的更多文档可以在Octave 文档页面找到。 MATLAB 文档可以在MATLAB 文档页面中找到。

我们也强烈鼓励使用在线讨论与其他学生讨论练习。但是,请勿查看他人编写的任何源代码或与他人共享您的源代码。

---

1 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）。自动手写数字识别如今已广泛使用 从识别邮件信封上的邮政编码（postal code)到识别银行支票上的金额。本练习将向您展示您所学的方法如何用于此分类任务。

在练习的第一部分,您将扩展您之前的实现
逻辑回归并将其应用于一对多分类。

1Octave 是 MATLAB 的免费替代品。对于编程练习,您可以自由使用 Octave 或 MATLAB。

1.1 数据集

```
% 从文件中加载保存的矩阵
加载（ ex3data1.mat ）； % 矩阵 X
和 y 现在将在您的 Octave 环境中
```

$$\begin{matrix} (x^{(1)}) \\ T \quad (x^{(2)}) \end{matrix}$$

吨

1.2 数据可视化

2这是 MNIST 手写数字数据集(http://yann.lecun.com/exdb/mnist/) 的子集。

displayData 函数,我们鼓励您检查代码以了解它是如何工作的。运行此步骤后,您应该会看到如图1 所示的图像。

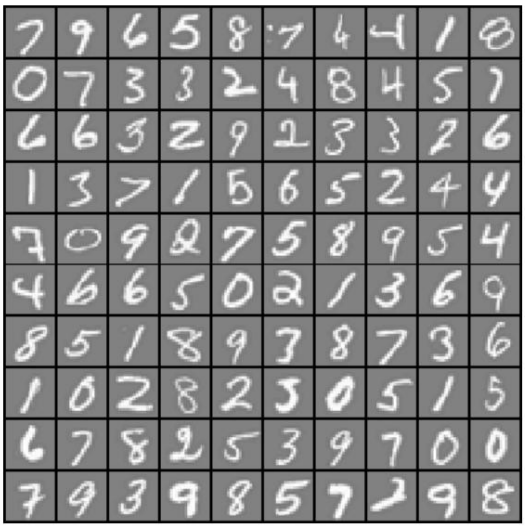

图 1:数据集中的示例

1.3 向量化逻辑回归

您将使用多个一对多逻辑回归模型来构建多类分类器。由于有 10 个类,您将需要训练 10 个单独的逻辑回归分类器。为了使这种训练有效,确保您的代码被很好地矢量化很重要。在本节中,您将实现不使用任何 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。

1.3.1 向量化成本函数

我们将从编写成本函数的矢量化版本开始。回想一下,在（非正则化)逻辑回归中,成本函数是

$$J(\theta) = \frac{1}{米} X米 \quad -y \quad (i) \log(h\theta(x\ (i)\ )) - (1 - y\ (i)\ ) \log(1 - h\theta(x\ (i)\ ))。$$

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(θ 发送x (i)) 和 g(z) = ~~1 1+e −z~~ 是个

(x (1)) T θ0

(x (2)) θ1

吨

θn

(x (1)) T

吨

(x (2))

吨 吨

吨

(一世)

th

—— — X米 (hθ(x (i)) - y (i))x j (一世)

我=1

所有θj的显式偏导数，

$$\begin{bmatrix} \frac{\partial J}{\partial \theta 0} \\ \frac{\partial J}{\partial \theta 1} \\ \frac{\partial J}{\partial \theta 2} \\ \vdots \\ \frac{\partial J}{\partial \theta n} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \sum_{i=1}^{m} (h\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \\ \sum_{i=1}^{m} (h\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \\ \sum_{i=1}^{m} (h\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \\ \vdots \\ \sum_{i=1}^{m} (h\theta(x^{(i)}) - y^{(i)}) x_n^{(i)} \end{bmatrix}$$

$$= \frac{1}{m} \sum_{i=1}^{m} (h\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$= \frac{1}{m} X^{T} (h\theta(x) - y). \qquad (1)$$

在哪里

$$h\theta(x) - y = \begin{bmatrix} h\theta(x^{(1)}) - y^{(1)} \\ h\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h\theta(x^{(m)}) - y^{(m)} \end{bmatrix}.$$

请注意，$x^{(i)}$ 和 $y^{(i)}$是向量，而$(h\theta(x^{(i)}) - y^{(i)})$ 是标量（单个数）。为了理解推导的最后一步，让$\beta_i = (h\theta(x^{(i)}) - y^{(i)})$ 和请注意：

$$\sum_i \beta_i x^{(i)} = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} = X^{T} \beta,$$

其中值$\beta_i = (h\theta(x^{(i)}) - y^{(i)})$。

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，我们鼓励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。您现在应该实现公式1来计算正确的矢量化梯度。完成后，通过实现渐变来完成函数 lrCostFunction.m。

调试提示:向量化代码有时会很棘手。一种常见的调试策略是使用 size 函数打印出您正在使用的矩阵的大小。例如,给定大小为 100 × 20 的数据矩阵 X（100 个示例,20 个特征)和 θ,一个维度为 20×1 的向量,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 向量化正则化逻辑回归

为逻辑回归实现矢量化后,您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

$$J(\theta) = \frac{1}{米} \sum_{我=1}^{米} -y \ (i) \log(h\theta(x\ (i))) - (1 - y\ (i)) \log(1 - h\theta(x\ (i))) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 。$$

请注意,您不应该正则化用于偏差的θ0
学期。

相应地， θj的正则化逻辑回归成本的偏导数定义为

$$\frac{\partial J(\theta)}{\partial \theta 0} = \frac{1}{米} \sum_{我=1}^{米} (h\theta(x\ (i)) - y\ (i))x_j^{(一世)} \qquad 对于 j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta j} = \frac{1}{米} \sum_{我=1}^{米} (h\theta(x\ (i)) - y\ (i))x^{(一世)} ! \_ + \frac{\lambda}{米} \theta j 对于 j \geqslant 1$$

现在修改 lrCostFunction 中的代码以考虑正则化。
再一次,你不应该在你的代码中加入任何循环。

Octave/MATLAB 提示:在为正则化逻辑回归实现矢量化时,您可能经常只想求和和更新 θ 的某些元素。在 Octave/MATLAB 中,您可以对矩阵进行索引以仅访问和更新某些元素。例如,A(:, 3:5)

= B(:, 1:3) 将用 B 中的第 1 到 3 列替换 A 的第 3 到 5 列。您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) 将只返回A 的第 2列到最后一列的元素。因此,您可以将其与 sum 和 .^ 操作一起使用,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,我们还提供了有关计算正则化梯度的另一种可能方法的提示。

您现在应该提交您的解决方案。

1.4 一对多分类

在这部分练习中,您将通过训练多个正则化逻辑回归分类器来实现一对多分类,每个分类器对应于我们数据集中的 K 个类别（图1）。在手写数字数据集中,K = 10,但您的代码应该适用于任何 K 值。

您现在应该完成 oneVsAll.m 中的代码,为每个类训练一个分类器。特别是,您的代码应返回矩阵 Θ ∈ R K×(N+1)中的所有分类器参数,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”循环来完成此操作,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,我们将数字“0”映射到标签 10（以避免与索引混淆）。

在为类 k ∈ {1, ..., K} 训练分类器时,您将需要标签 y 的维向量,其中yj ∈ 0, 1 表示第 j 个训练实例是否属于类 k (yj = 1),或者如果它属于不同的类 (yj = 0)。您可能会发现逻辑数组对这项任务很有帮助。

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,对向量 a（大小为 m×1)和标量 b 计算表达式 a == b 将返回一个与 a 大小相同的向量,其中 a 的元素等于 b 的位置和它们不同的地方为零。要了解它是如何为您自己工作的,请在 Octave/MATLAB 中尝试以下代码：

```
a = 1:10； % 创建 a 和 bb = 3;
a == b          % 你应该在这里尝试不同的 b 值
```

此外,您将在本练习中使用 fmincg（而不是 fminunc）。
fmincg 的工作方式与 fminunc 类似,但在处理大量参数时效率更高。

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

您现在应该提交您的解决方案。

1.4.1 一对多预测

在训练完一对多分类器之后,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”。您的一对多预测函数将选择相应逻辑回归分类器为其输出最高概率的类,并返回类标签（1、2、...或 K)作为输入示例的预测。

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

完成后,ex3.m 将使用 Θ 的学习值调用您的 predictOneVsAll 函数。您应该看到训练集的准确率约为 94.9%（即,它正确分类了训练集中 94.9% 的示例）。

您现在应该提交您的解决方案。

---

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

在这部分练习中,您将使用与之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,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。在下周的练习中,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 将帮助您逐步完成此练习。

2.1 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。由于图像大小为 20×20,这给了我们 400 个输入层单元(不包括始终输出 +1 的额外偏置单元)。和以前一样,训练数据将被加载到变量 X 和 y 中。

我们已经为您提供了一组网络参数(Θ(1) , Θ(2)) 。这些存储在 ex3weights.mat 中,并将由 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 个单元和 10 个输出单元(对应于 10 位类) .

```
% 从文件中加载保存的矩阵
负载 ( ex3weights.mat ) ;

% 矩阵 Theta1 和 Theta2 现在将在您的 Octave % 环境中

% Theta1 的大小为 25 x 401
% Theta2 的大小为 10 x 26
```

3您可以在逻辑回归中添加更多特征(例如多项式特征),但是训练起来可能非常昂贵。

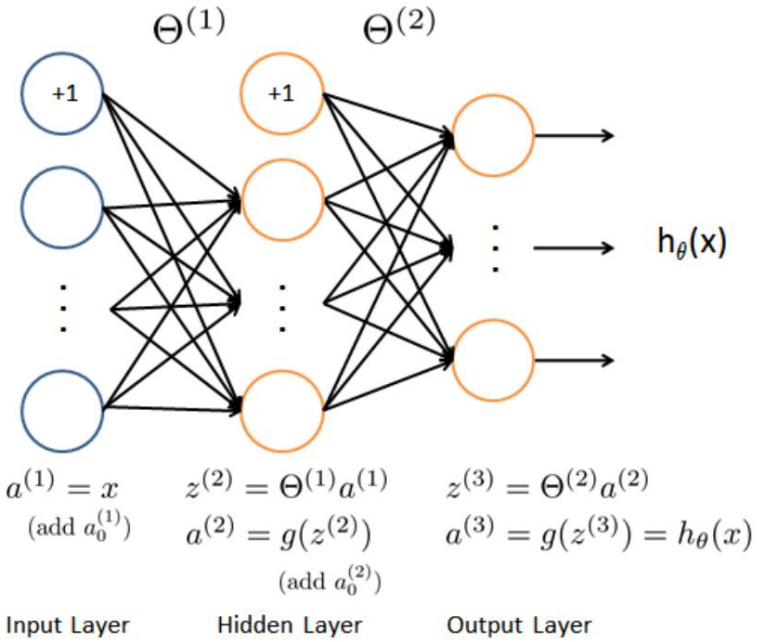

图 2:神经网络模型。

2.2 前馈传播与预测

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

您应该实现为每个示例 i计算hθ(x (i)) 并返回相关预测的前馈计算。与一对多分类策略类似,神经网络的预测将是具有最大输出(hθ(x))k 的标签。

实施说明:矩阵 X 包含行中的示例。
完成 predict.m 中的代码后,您需要将 1 的列添加到矩阵中。矩阵 Theta1 和 Theta2 包含行中每个单元的参数。具体来说,Theta1 的第一行对应于第二层的第一个隐藏单元。八度/MAT (2)

LAB,当您计算必要的 z (l)时,正确转 = Θ(1)a (1),一定要索引（如果
置 X,以便您得到 作为列向量。

完成后,ex3 nn.m 将使用已加载的 Theta1 和 Theta2 参数集调用您的预测函数。你应该看到

准确率约为 97.5%。之后，一个交互式序列将启动一次显示训练集中的图像，而控制台打印

输出显示图像的预测标签。要停止图像序列，
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到我们的服务器。以下是每个方法的细分

这个练习的一部分是计分的。

部分	提交文件	积分
正则化逻辑回归 一对多分类器训练 一对	lrCostFunction.m 30 点	
多分类器预测 神经网络预测函数 predict.m 总分	oneVsAll.m 20 分 predictOneVsAll.m 20 分	
		30 分
		100 分

您可以多次提交您的解决方案，我们将采取
只考虑最高分。