吴恩达—神经网络和深度学习 第四章 深层神经网络

4.1 深层神经网络

什么是深层神经网络：

![前面学到过的一些神经网络模型](<http://upload-images.jianshu.io/upload_images/5355764-d82524fbde679ff5.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

Logistic回归是一个浅层模型，而最后一个5隐藏层神经网络则是一个深的多的神经网络，浅层和深层是一个程度的问题，要记住当我们数神经网络有几层的时候，我们不能把输入层输进去，应该只是算上隐藏层的数量再加上输出层。单隐层神经网络其实是一个双层神经网络。Logistic回归是一个单层神经网络。

但是前几年在人工智能或者机器学习社区中，大家发觉有些函数只有非常深层的神经网络能够学习，而浅的一些模型通常无法学习，虽然处理任何具体问题的时候，预先准确地判断需要多深的神经网络都会很难，所以先试试看logistic回归是非常合理的做法，试一下单层然后两层，然后把隐层数量当成另一个可以自由选择数值大小的超参数，然后在保留交叉验证数据上评估，或者用你自己的开发集评估。

关于深层神经网络用到的参数说明：

L：用来表示神经网络的层数

：来表示第层上的单元数量，即节点的数量

：表示第中的激活函数

：在中计算值的权重

所以当我们把输入层标为第0层的话，刚好也符合我们判断一个神经网络有几层的定义。，即输入层的节点数。那么一会我们会看到在前向传播中你最后要计算的是激活函数，激活函数也会用层数来标注。最后总结一下符号约定，输入特征用x表示，但是x也是第0层的激活函数，即，最后一层的激活函数，也就是说等于预测输出，也就是这个神经网络预测出来的。

4.2 深层神经网络中的前向传播

这一节讲解如何在深度网络中灵活应用正向和反向传播，跟浅层神经网络研究的方法一样，我们先看一下对于其中一个训练样本x，该如何应用正向传播，后面再讨论向量化的版本，即所有训练样本同时参与运算，对整个训练集应用正向传播的时候。

首先，输入x后，计算第一层，，那么和就是会影响在第一层的激活单元的参数，如果你要计算这一层的激活函数，，那么激活函数g的指标取决于所在的层数，计算好第一层之后，第一层后面依次类推。注意输入特征向量x也是第0层的激活单元，最后得到的。根据上面，对于单个样本，总结的来说，

现在我们已经过了一遍针对一个训练样本的过程，接下来看看怎么用向量化的方法训练整个训练集，公式其实都差不多，第一层是, ，其中X= ，接下来后面的层都是一次类推，我们要做的只是在把所有的z或者a向量堆叠起来，如每一列对应一个训练样本，构成矩阵A，矩阵Z是一个道理。堆叠完成后，随后我们得到，这就是把所有的训练样本的预测值水平的叠在一起。

总结的说，将对于一个样本的公式，小写字母修改为大小字母，表示整个训练集即可，就得到了针对一个训练集的向量化的正向传播算法步骤，其中。

![深层神经网络正向传播的一个例子](<http://upload-images.jianshu.io/upload_images/5355764-930b6eaf9a14b14e.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

其实深层神经网络对于前面的4或5层神经网络，区别在于，深层神经网络是不知道有多少层，我们用L表示，后面的计算过程中，我们无法写清楚到底运用了多少组计算Z以及A的公式，所以我们会用到for循环，i等于从1到L，然后去计算每一层的激活函数，接着第二层，第三层，，，所以看起来是个for循环，在实现代码的时候我们一般不喜欢用显式for循环，但是在这个情况下，除了显式for循环，没有其他更好的办法。所以在我们实现正向传播时，使用for循环也是ok的。

事实上，我们在实现深度神经网络的过程中，想增加得到没有bug的程序的概率，其中一个方法就是需要仔细和系统化地去思考矩阵的维数，自己在debug的时候，通常需要拿出一张纸，一遍很仔细的过一遍，在操作的矩阵的维数。下一节将讲解如何操作矩阵的维数。

4.3 核对矩阵的维数

Debug时候核对代码是否有错的一个方法就是拿出纸，过一遍算法中矩阵的维数。目前我们只学习了只有一个输出单元的神经网络，后面我们也会说到有多个输出单元的神经网络。

总结来说，的维度应该为，也就是说线上的参数矩阵维度应该为（下一层节点数，上一层节点数），关于偏置向量的维度则是也就是（下一层的节点数，1），可以记忆为对每一节点都应该有一个偏执数，纵向堆叠在一起，行一列。

这两个式子可以帮你检查你的矩阵W的维度以及你的向量b的维度，当然如果实现反向传播的话，那么dw的维度应该和W的维度相同，db也会和b的维度一样。

我们还需要检查的就是z，x和的维度，但是很明显a的维度应该和z的相同。

![探讨深度神经网络检查矩阵维度](<http://upload-images.jianshu.io/upload_images/5355764-301c07f38d07b411.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

依照惯例，接下来应该看看向量化的过程，这样就可以同时作用于多个样本。其实即使是有了向量化的过程，W b dW db的维度应该始终都是一样的，但是Z A X会发生变化，应该是由不同样本计算出来的水平堆叠而成，因此维度变成（，m），其中m是训练集的大小，X的维度为（），值得一提的是，虽然（），但是通过python的广播机制，最后被复制成一个（）的矩阵，然后逐个元素相加，所以

还有个情况，当L等于0是，也就是输入特征X，，当然你在实现反向传播的话，

当你在实现深度网络传播过沉重，一定要确认所有的矩阵维数是前后一致的，这会大大地帮助你排除一些bug的来源。

4.4 为什么使用深层表示

我们都知道深度神经网络能解决很多的问题， 其实并不需要很大的神经网络，但是得有深度，得有比较多的隐藏层，为什么呢，通过下来的例子来帮助理解。

首先，深度网络究竟是在计算什么，如果你在建一个人脸识别，或者是人脸检测系统，深度神经网络所做的事就是当你输入一张脸部的照片，然后你可以把深度神经网络的第一层当成一个特征探测器，或者边缘探测器，在这个例子里面，我会建一个大概有20个隐藏单元的深度神经网络，观察是怎么针对这张图计算的，隐藏单元就是左下角图里的小方块，举个例子，最左上角那个小方块就是一个隐藏单元，它会去找这张照片里边缘的方向，那么最后一排倒数第二个这个隐藏单元可能是在找水平方向上的边缘在哪里，之后的课程里会继续说到专门做这种识别的卷积神经网络。你可以先把神经网络的第一层当做看图，然后去找这张图片的各个边缘，我们可以把照片里组成边缘的像素们放在一起看，然后它可以把被探测到的边缘组合成面部的不同部分，比如说，可能有一个神经元会去找眼睛的部分，另外还有别的找鼻子的部分，然后把许多的边缘结合起来就可以开始检测人脸的不同部分，最后再把这个部分放在一起，比如鼻子眼睛下巴，就可以识别或是探测不同的人脸啦。

关于深度神经网络运用到人脸识别原理的理解：本质是模式识别，把抽象的能表示一张人脸的照片抽象成计算机能理解的数字，一张图片是有256种颜色的，图像的每一个像素点都是0到256的数，可以通过像素来表征一张图片，那么这些像素值就构成了一个矩阵，如何去识别这个矩阵中的模式呢，用一个相对来讲比较小矩阵在这个比较大的矩阵中，从左到右，从上到下的扫一遍，每一个小矩阵区块内，你可以统计0到255每种颜色出现的次数，以此来表达这一个区块的特征。这样通过这一次“扫描”，你得到了另一个由很多小矩阵区块特征组成的矩阵，这个矩阵肯定比原始的矩阵小，然后对这个小一点的矩阵，再进行一次上面的步骤，进行一次特征“浓缩”，通过不断的抽象化，最后得到个1x1的矩阵，也就是一个数，来表示这个图片，计算机也就达到了辨别的目的。对应到神经网络上，在输入层输入图片特征向量后，经过一层计算，会得到一个可以表征前面这些大量特征的一个小的矩阵，我们都知道，简单的logistic回归可以把多个特征表示成一个简单的，那么第一层隐藏层则就是对很大的矩阵进行扫描后，得到对每个小矩阵区块特征都进行表示后的一个小矩阵，经过层层表示，最后得到一个数来表示。

你可以直觉上把这种神经网络的前几层当作探测简单的函数，比如边缘，之后再跟后几层结合在一起，那么总体上就能学习到更多复杂的函数。这些图的意义在学习卷积神经网络的时候再深入了解，还有一个技术性的细节需要理解的就是边缘探测器士气相对来说都是针对照片中非常小块的面积，面部探测器呢就会针对一些大一些的区域，但是最主要的概念是一般你会从比较小的细节入手，比如边缘，然后再一步步到更大更复杂的区域，比如一只眼睛或是一个鼻子，再把眼睛鼻子装在一块，组成更复杂的部分。

![深度学习与人脸识别](<http://upload-images.jianshu.io/upload_images/5355764-b61316ed6ec3a5b2.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

这种从简单到复杂的金字塔状表示方法或者组成方法，也可以应用在图像或是人脸识别以外的其他数据上，比如当你想要建一个语音识别系统的时候，需要解决的就是如何可视化语音，比如你输入一个音频片段，那么神经网络的第一层可能就会去先开始试着探测比较低层次的音频波形的一些特征，比如音调是变高了还是变低了，分辨白噪音，嘶嘶嘶的声音等，可以选择这些相对程度比较低的波形特征，然后把这些波形组合在一起，就能去探测声音的基本单元，在语言学中有个概念叫做音位，比如说单词cat c的发音就是一个音位，a也是，有了基本的声音单元以后组合起来，你就能识别音频当中的单词，单词再组合起来就能识别词组，再到完整的句子，所以深度神经网络的这许多隐层中，较早的前几层能学习一些低层次的简单特征，等到后几层就能把简单的特征结合起来去探测更加复杂的东西，比如你录在音频里的单词，词组，或者句子，这样就能进行语音识别了。 同时我们所计算的之前的几层，也就是相对简单的输入函数，比如图像的边缘呀，到网络的深层时，你实际上就能做很多复杂的事情比如探测面部之类的。

除了关于深度神经网络与人类的大脑类似，可能都是从简单到复杂的一个认识过程，另外一个关于神经网络为何有效的理论来源于电路理论，它能够用哪些电路元件和你计算哪些函数有分不开的联系，根据不同的基本逻辑门，与或非门，在非正式的情况下，这些函数都可以用相对较小，但是很深的神经网络来计算，小的意思是说隐藏单元的数量相对比较小，但是如果你用浅一些的神经网络计算同样的函数，也就是说在我们不能用很多隐藏层时，你会需要成指数增长的单元数量才能达到同样的计算结果。简单的解释下，比如你要计算四个数的和，如果先两两计算，再将两个结果加起来，总共只需要3次运算，而直接一次依次计算则需要4次，虽然看起来差别不大，但当需要计算的内容很多很多呢，差异将会越来越明显，将两个数的一次计算当成一个隐藏单元，依次计算作为一层，那么第一种需要两层3次，第二种只需要一层4次，所以当深度越小时，需要的隐藏单元数可能呈指数增长。这也就能解释为什么深度神经网络是很有效的。

4.5 搭建神经网络块

对于一个层数较少的神经网络，我们选择其中一层来讨论，需要的参数，关于输入输出都在图片上标注，首先对于第层，有参数和,正向传播过程，输入，输出为，其中公式就是计算z再计算a的两步，接下来把的值缓存起来，因为缓存的对后面的正向反向传播的步骤都非常有用。接下来是反向传播步骤，你会需要实现一个函数，输入为以及所缓存的值，输出为以及以及，以上就是基本的在某一层的正向传播和反向传播的步骤，即正向函数和反向函数。对于某一层的传播，图示如下，红色箭头表示反向过程。

![深度神经网络某一层的两个传播函数图示](<http://upload-images.jianshu.io/upload_images/5355764-f2081ac4ba7fac27.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

如果理解了某一层，那么对于整个深度神经网络实现如下图：

![神经网络的一个梯度下降循环](<http://upload-images.jianshu.io/upload_images/5355764-bbd643f3f8629d31.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

继续下去之前再补充一个细节，概念上会非常有帮助，那就是把反向函数计算出来的z值缓存下来，当你做变成练习去实现它时，你会发现缓存可能很方便，可以在分享传播函数迅速得到和的值，在编程练习中你缓存了z，还有W和b，从实现角度上看，可以将参数复制到你实现反向传播任何需要用到这些参数的地方，这就是实现过程的细节。

现在已经学习到实现深层神经网络的基本元件，在每一层中，有一个正向传播步骤和一个对应的反向传播步骤，以及把信息从一步传递到另一步的缓存，下一节将会探讨这些元件具体实现过程。

4.6 前向和反向传播

这一节学习如何实现这些正向和反向传播步骤，如下图。正向传播的实现比较简单，只需要注意最开始输入的就是训练样本的输入特征=X，图片右半部分是向量化的公式，重复这个过程就能从左到右计算正向传播。

![正向传播实现步骤](<http://upload-images.jianshu.io/upload_images/5355764-e619d947eaf1ce55.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

![反向传播步骤](<http://upload-images.jianshu.io/upload_images/5355764-d1e375c5132b5736.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

最后总结一下，用x作为输入，第一层你可能会有个修正线性单元激活函数ReLU函数，第二层可能会用另一个ReLU，第三层可能是sigmoid函数，如果你是要做二分分类的话，输出值是，用可以计算出损失，这样你就可以向后迭代，反向传播求导。

那么关于反向传播过程的初始值呢，如果你是做二分分类，那么，用这个来初始化最后一层，如果是用向量化对整个训练集进行操作的话，那么需要用。

![深度神经网络实现图示](<http://upload-images.jianshu.io/upload_images/5355764-e696f2d1641fecd4.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240>)

4.7 参数vs超参数

想要你的神经网络起到很好的效果，你还需要规划好你的参数以及超参数。

超参数：看看模型里面的参数W和b，在学习算法过程中还有很多其他参数，需要输入到学习算法中，比如学习率，通过设置学习率来决定参数如何进化，还有梯度下降法循环的数量，在你的神经网络中还有其他你想设置的参数，比如隐层数L，以及隐藏单元数，你还可以选择激活函数，在隐层中，使用ReLU函数还是tanh，还是sigmoid函数，那么以上的这几个量都是需要自己去设置的。这些数实际上控制了最后参数W和b的值，所以它们被称为超参数。这些超参数在某种程度上决定了最终得到的W和b。实际上，深度学习有很多不同的超参数，后面也会说一下其他的超参数，比如momentum，mini batch的大小，几种不同的正则化参数等等，现在不确定这些什么意思，课程2会提到。

当你自己着手设计一个神经网络时，你会发现超参数的选择有很多可能性，所以你得尝试很多不同的值，今天的深度学习领域，还是很经验性的过程，通常你有一个想法，比如你可能大致知道一个最好的学习率值0.01最好，然后再尝试其他的0.05，根据你选择的值计算损失函数J，可能发现有不同的情况，最后你发现有一个值使学习的过程更快，并且还是损失函数值收敛在一个更小的值上，那么我们就用这个。

可能的确是深度学习比较让人不满的部分就是有很多的参数，还需要反复的尝试检验这个参数的值是不是最优，但参数设定这个领域深度学习研究还在进步中，可能过段时间会有更好的办法来决定超参数的值，也很有可能由于CPU GPU网络和数据都在变化，这样的方法可能只会在一段时间内起作用，只要你不断尝试，并且尝试保留交叉检验或类似的检验方法，然后挑一个对你的问题效果比较好的数值。