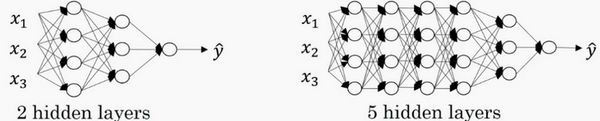
# 深层神经网络

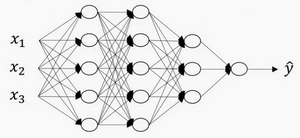
## 深层神经网络

我们在此之前已经学习完了前向传播和反向传播，以及如何搭建浅层神经网络，本周要做的就是把这些理念集合起来，就可以从头搭建一个深层的神经网络。

回顾一下关于神经网络的层数第一，如图所示。

从左到右，由0开始定义。X1,x2,x3是第0层，左边的隐藏层是第一层，由此类推。

在最近几年的研究中我们发现，有些函数只有非常深的神经网络才能够拟合，而浅层神经网络对此无用。我们很难一开始就知道要用多深的神经网络，所以比较好的方法是从逻辑回归开始，逐步增加神经网络的层次。

我们接下来看看神经网络的符号定义。如下图所示，是一个四层的神经网络，有三个隐藏层第一层和第二层都有5个神经元，第三层有三个，第四层只有1个。我们用L表示层数，L=4。表示有五个神经元节点，以此类推。

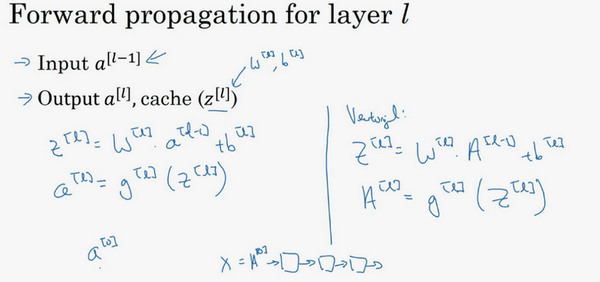
对于每层，我们都用来表示神经元激活后的数值，我们用来记作在*l*层计算值的权重。

## 前向传播和反向传播

之前我们学习了构成神经网络的基本模块，比如每一层都有前向传播和反向传播步骤，现在我们来看看是如何具体实现这些步骤的。

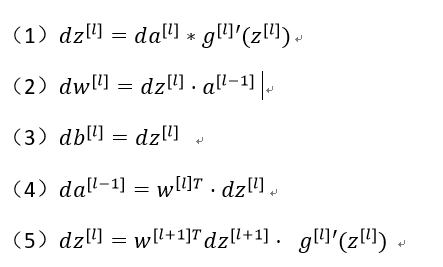
我们先从前向传播开始观察，我们输入，得到，缓存为，我们可以同时缓存下

和和，这样更容易在不同的环节调用这些参数。

如图所示，

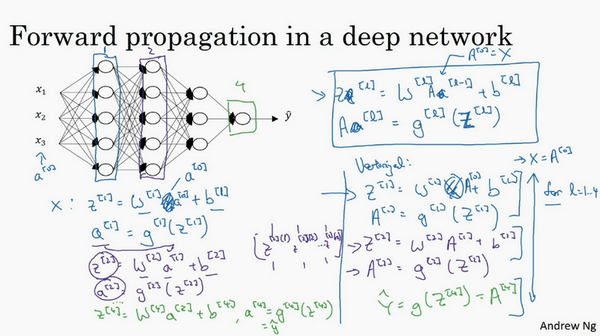
所以前向传播算法可以表示为，

前向传播算法需要喂入来进行初始化

接下来我们看反向传播算法， 我们输入为，输出为，, ，我们可以得到以下算法

## 1.3深层网络中的前向传播

和往常一样，我们先研究单个样本是如何进行前向传播的。我们有如图所示的神经网络。



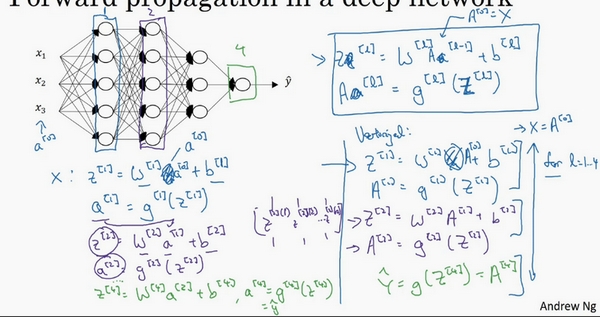
第一层我们需要计算，，

第二层我们需要计算，

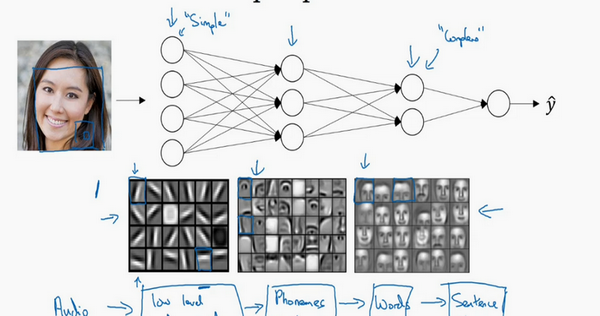
以此类推，到第四层，我们计算，，前向传播算法可以归纳为多次迭代，，。这里我们不得不用一个for循环，来执行每层的计算。

## 1.4核对矩阵的维度

在实现深度学习算法时，一个很有效的避免bug的方法就是拿出一张纸写出各个矩阵的维度。的维度是（下一层的维数，前一层的维数），即: (,)；

其他维度如图所示

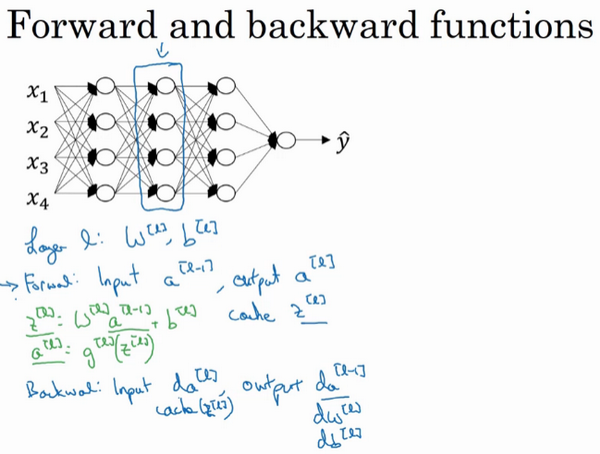
## 1.5 为什么使用深层次神经网络

神经网络能解决许多问题，而且不需要太多的神经元节点，但是需要很深的网络层次。首先，我们看看神经网络在计算什么。以人脸检测的模型为例。如图所示，

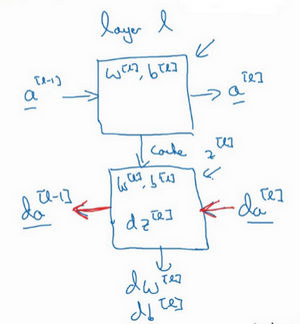
当你输入一张人脸图片，神经网络的第一层会提取出图片的边缘特征，第二层就根据提取出的面部特征检测不同的面部部分，最后识别出整张人脸。神经网络先从边缘的细节开始，之后再组合成更大的更复杂的部分。

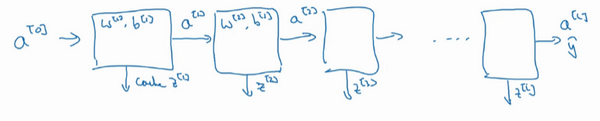
还有一个原因就是，和逻辑电路类似，当想表达一个复杂的函数时，使用深层次的网络会比浅层次占用更少的节点。

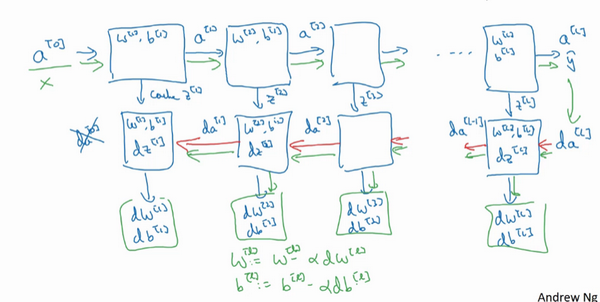
## 1.6 搭建神经网络块

如图所示，这是一个神经网络结构，我们从其中的一层计算开始研究。

在第l层我们有参数和，有激活函数，输入是，输出是，我们可以同时计算得到把，并将其缓存起来，用于反向传播的计算。在反向传播计算过程中，我们有输入，输出。我们把之前缓存的z传递到反向传播过程中，用此来计算其他参数的偏微分，并用此进行梯度下降。







## 1.7 参数和超参数

想要我们的神经网络达到很好的效果，我们还需要仔细规划参数和超参数。参数是指模拟该函数所直接用到的参数，如ω和b，超参数是决定这些参数形成的数字，如learing rate，iterations，激活函数，神经网络层数，节点等。如何寻找超参数的最优值？我们的结论是不断尝试不同的超参数，并进行更新迭代。超参数事实上并没有什么明显的经验法则。