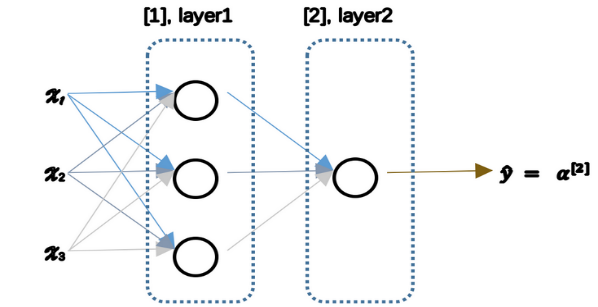
# 浅层神经网络

## 神经网络概述

本周我们将学习如何实现一个神经网络，在具体学习相关技术之前，我们会对神经网络的大体特征做一些了解。

神经网络看起来是如下图所示的样子

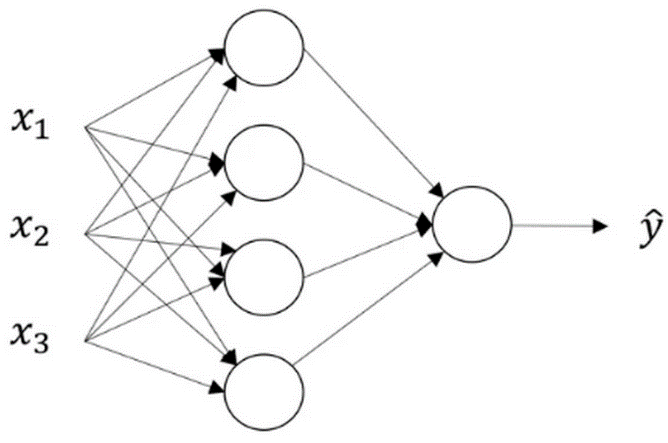


我们首先计算第一层各个节点的值，再使用前向传播算法，计算下一层各个节点的值

接着我们再使用反向传播算法，从右到左计算出各个参数的偏导数。

## 神经网络的表示

我们通过1.1已经知道了神经网络的大致样子，现在我们将讨论其中的具体含义。我们从一个简单的例子入手。如图所示。



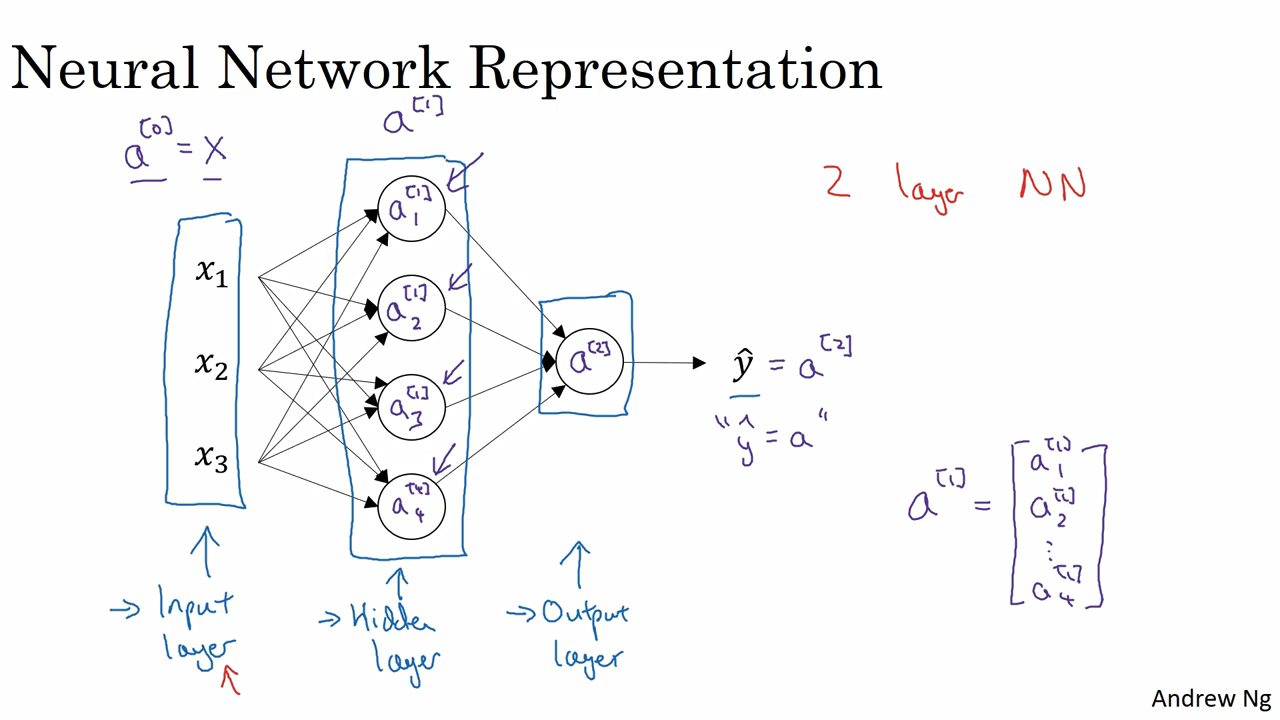
我们的样本有三个输入特征，(x1,x2,x3),它们数值的堆叠起来，成为一个列向量。这叫做神经网络的输入层，接着，我们有四个节点，叫做隐藏层，因为这四个节点在样本中并不可见。

最后，我们有个输出层，负责输出最后的预测值。

如图所示的其实是个两层神经网络，输入层作为第0层，隐藏层为第一层，输出层为第二层。

我们引入几个新的计号，a表示激活(activation),所以输入层即为，第n层产生的激活值即为.第n层的第j个节点我们记作.在本例中，第一个隐藏层可以表示为

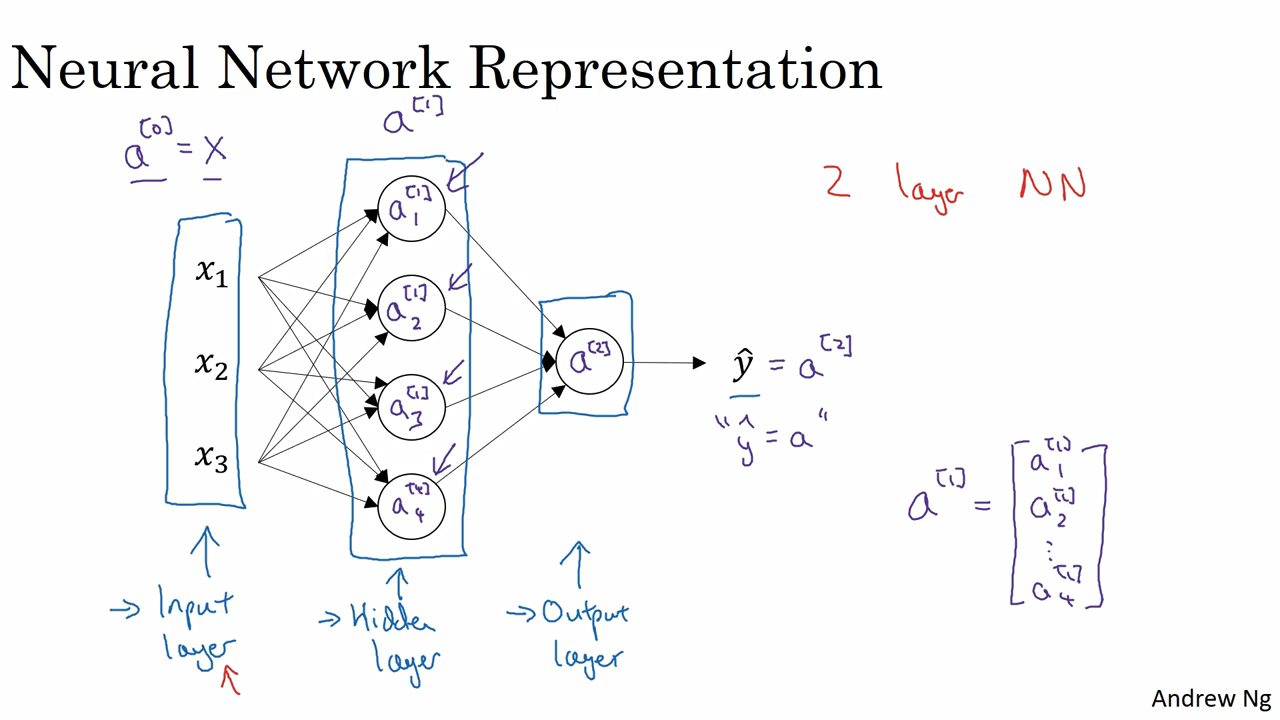
最后的输出层将产生一个实数，如图所示。



此外，我们还要注意到，神经网络的每一层都是带有参数的，这里的隐藏层将拥有两个参数w和b，即(,,

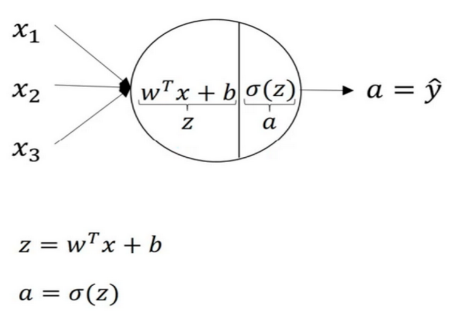
## 1.3计算一个神经网络的输出

我们已经了解了神经网络的结构和表示，接下来，我们将要了解神经网络是如何计算出结果的。回顾一下那个只有两层的神经网络结构，如图所示。



其中x表示特征，a表示各层激活后的值，w表示特征的权重。

那么，我们是如何得到神经网络的输出的呢？

我们回顾一下逻辑回归算法，如图所示

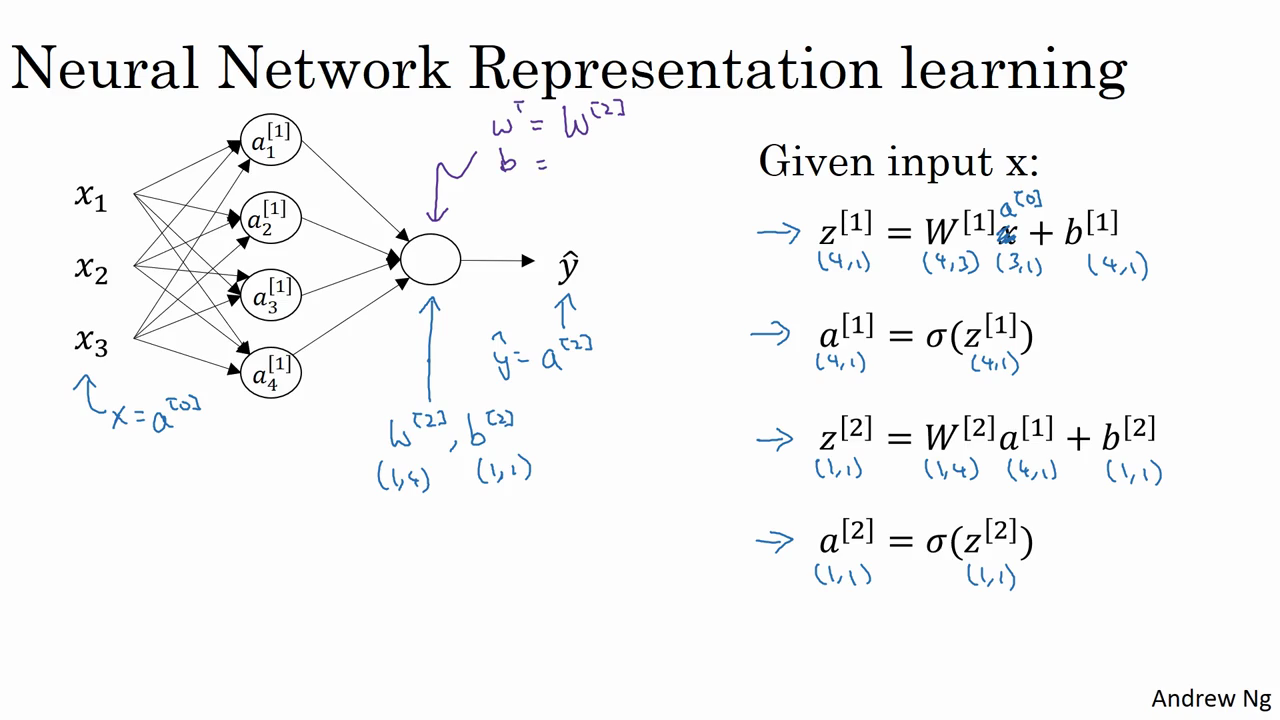
首先我们计算出z，接着经过激活函数，我们计算出a，一个神经网络只是做了很多次这样子的重复计算。

我们来体验一下上图两层神经网络的计算步骤。

我们从隐藏层的第一个神经元开始， 与逻辑回归类似，计算分为两步，首先，计算出，接着，我们经过激活函数，计算。隐藏层中的各个节点都有类似的计算，详细结果如下。

如果我们使用for循环来进行计算，那么效率真的很低，所以我们接下来要做的就是把这个等式向量化。根据线性代数中的知识，w是系数，x是变量，z可由x线性组合而成。

我们把w纵向堆积起来，并与x进行矩阵相乘，得到，

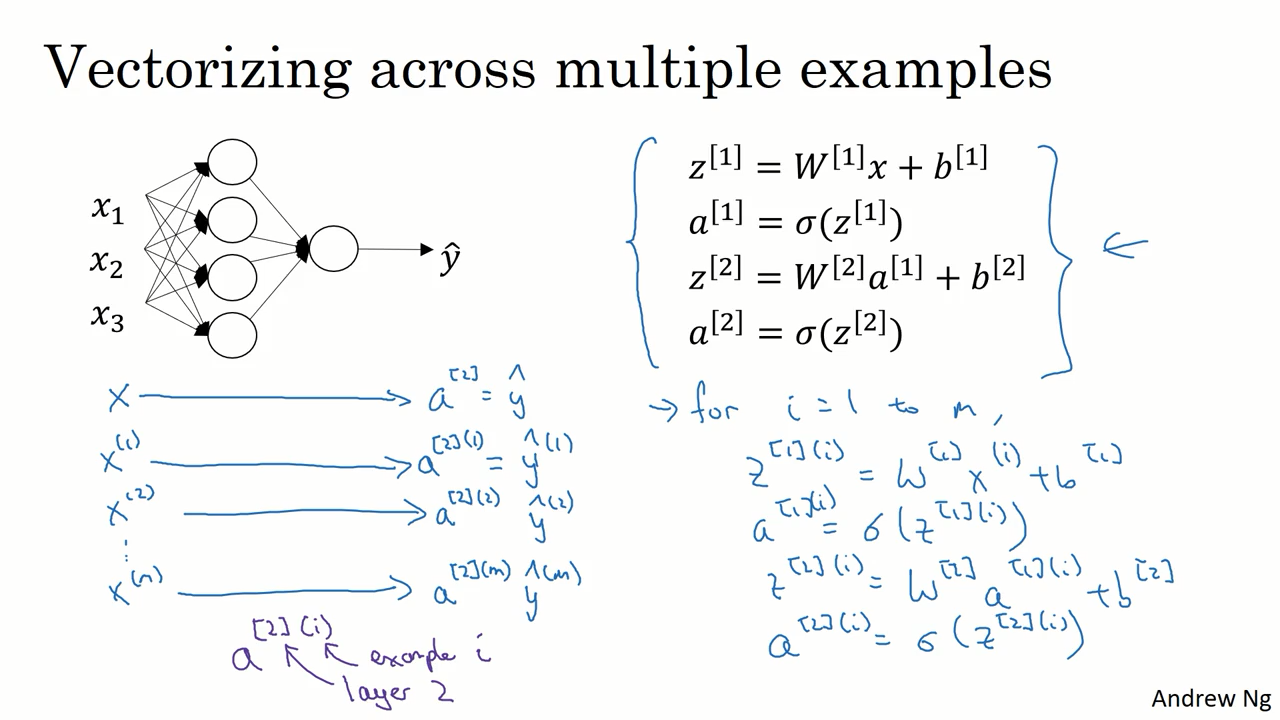


如此，我们经过向量化，就可以一次得到隐藏层中每个节点的输出。

## 1.4多样本的向量化

在上一节，我们已经学习到如何为单个样本计算输出值，现在，我们将体验使用多个样本同时计算，获得结果。

如下图所示

我们将每个样本x横向叠加，得到

接着，我们计算z1，得到

如此，我们便完成了多个样本的向量化计算

X的每一列都是一个样本

## 1.5 激活函数

到目前为止，我们只使用过sigmoid函数，实际上，如果我们更换成其他激活函数，效果可能会更好

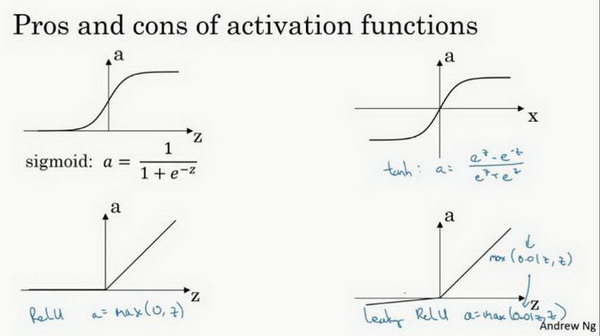
例如我们可以用tanh函数，tanh的值域位于(-1,1)，可以起到将数据中心化的效果，

事实上，tanh是sigmoid向下平移和伸缩后的结果

Tanh在任何场合下都优于sigmoid，但是在01分类中，还是需要用到sigmoid函数

Sigmoid和tanh的共同缺点是，在z特别大或者特别小的情况下，梯度都会变得很小，导致梯度下降的速度很慢

另一个流行的函数时，ReLu函数，各种不同的激活函数如下图所示



Relu在通常情况下是最快的，也是最常见最好的选择

## 1.6 神经网络的梯度下降

首先使用前向传播算法计算出预测值

(1)

(2)

(3)

(4)

接着使用反向传播算法计算参数的偏导数

公式3.32：

公式3.33：

公式3.34： 

公式3.35：

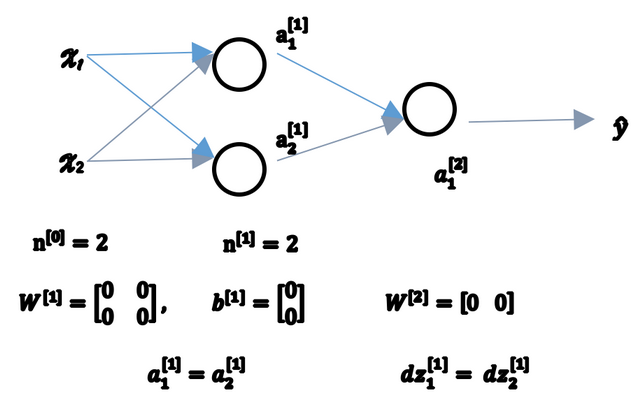


公式3.36：

公式3.37：

## 1.7 随机初始化

对于神经网络而言，对参数进行随机初始化是非常重要的，如果将参数全部初始化为相同值，则每个节点得到的函数都是完全相同，训练后的结果也是完全相同，如图所示



最后整个神经网络会变得完全对称而无法发挥作用