## 浅层神经网络

### 1. 神经网络概览

神经网络与逻辑回归类似，就是多了一层隐藏层，所以神经网络的正向传播和反向传播多了一次重复计算。

第一层是输入层到隐藏层用[1]表示：

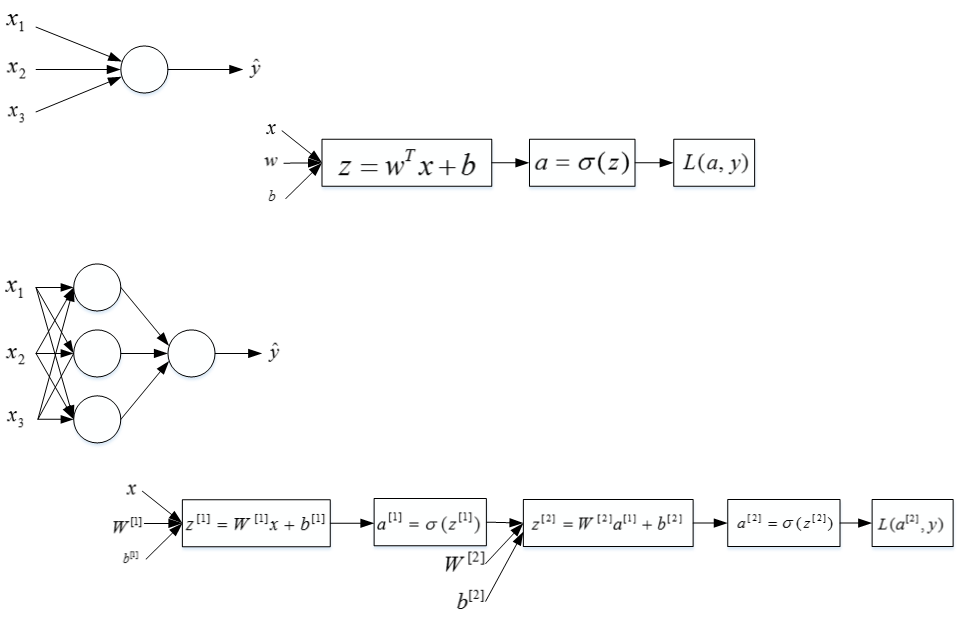


第二层是隐藏层到输出层，用上标[2]来表示：



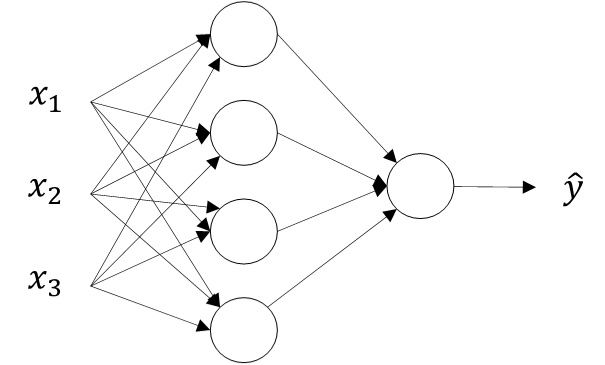
方括号上标[i]表示当前所处的层数；圆括号上标(i)表示第i个样本。

同样，反向传播过程也分成两层。第一层是输出层到隐藏层，第二层是隐藏层到输入层。



### 2. 神经网络表示

下面以图示的方式介绍一下单隐藏层神经网络



结构上，从左到右，可以分成三层：输入层（Input layer），隐藏层（Hidden layer）和输出层（Output layer）

在写法上，我们通常把输入矩阵X记为，把隐藏层输出记为，上标从0开始。下标表示神经元，从[1]开始，隐藏层第一个神经元。这样，隐藏层有4个神经元就可以将其输出写成矩阵的形式：



输出为。这种单隐藏层神经网络也被称为两层神经网络（2 layer NN）

隐藏层权重的维度是（4,3）。这里的4对应着隐藏层神经元个数，3对应着输入层x特征向量包含元素个数。常数项的维度是（4,1）。

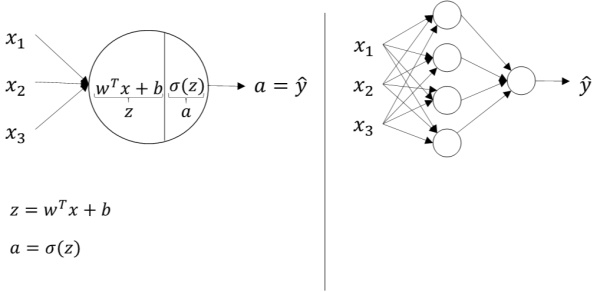
的维度是（1,4），的维度是（1,1）。

### 3. 计算神经网络的输出

神经网络推导。

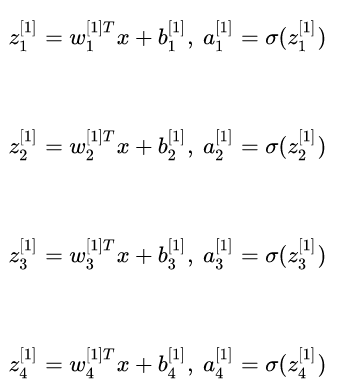
逻辑回归的正向计算可以分解成计算z和a的两部分：





对于两层神经网络，从输入层到隐藏层对应一次逻辑回归运算；从隐藏层到输出层对应一次逻辑回归运算。

从输入层到输出层的计算公式列：



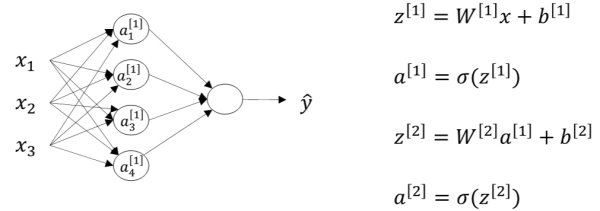
从隐藏层到输出层



上述每个节点的计算都对应着一次逻辑运算的过程，分别由计算z和a两部分组成。

为了提高程序运算速度，转换成矩阵运算

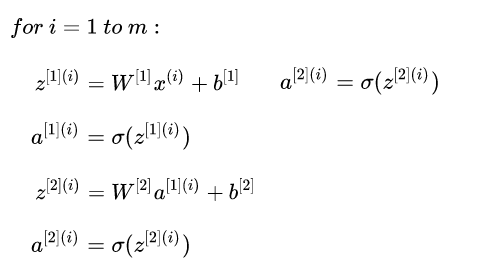


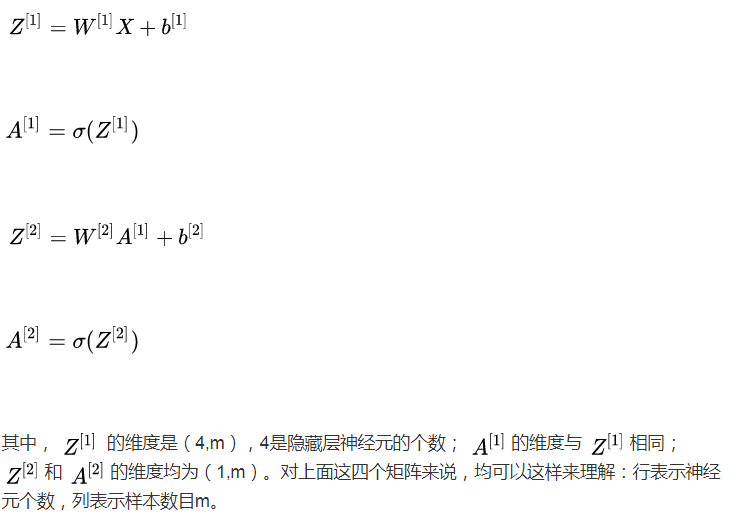
### 4. 多个例子向量化

m个训练样本的神经网络正向传播矩阵运算过程。

用上标(i)表示第i个样本

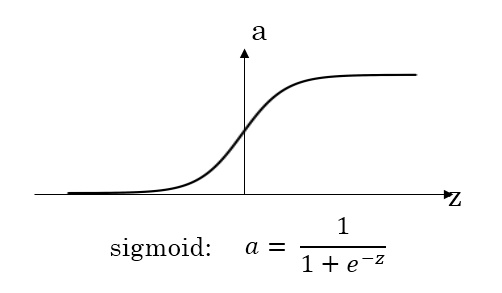
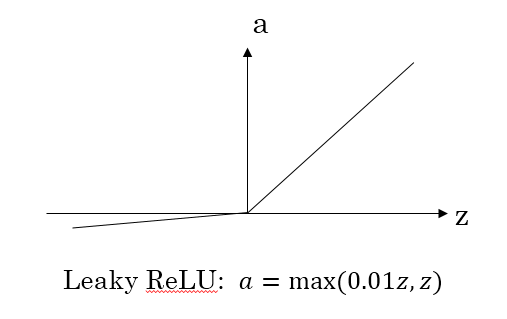
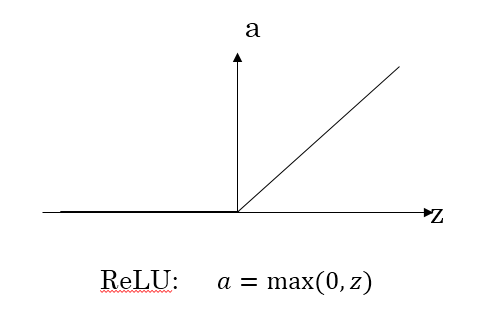
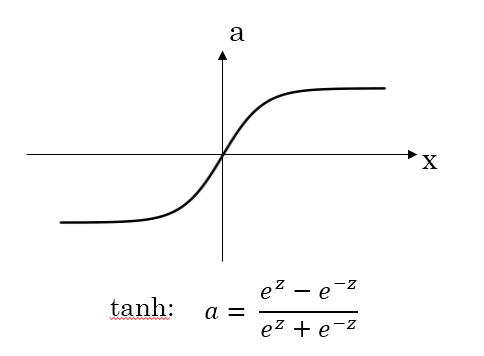


不使用for循环，利用矩阵运算的思想，输入矩阵X的维度为(nx,m)。这样，我们可以把上面的for循环写成矩阵运算的形式：



### 6. 激活函数

隐藏层还输出层都需要激活函数（activation function），下面介绍几种不同的激活函数

1. sigmoid函数
2. tanh函数
3. ReLU函数
4. Leaky ReLU函数

**sigmoid函数和tanh函数比较：**

对于隐藏层的激活函数，一般来说，tanh函数要比sigmoid函数表现更好一些。因为tanh函数的取值范围在[-1,+1]之间，隐藏层的输出被限定在[-1,+1]之间，可以看成是在0值附近分布，均值为0。这样从隐藏层到输出层，数据起到了归一化（均值为0）的效果。因此，隐藏层的激活函数，tanh比sigmoid更好一些。

对于输出层的激活函数，因为二分类问题的输出取值为{0,+1}，所以一般会选择sigmoid作为激活函数。

Sigmoid函数与tanh函数，当z很大时，激活函数的斜率很小，梯度下降算法比较慢。在实际应用中，避免z落入这个区域，使|z|尽量在零值附近。

为了弥补这个缺陷，出险了ReLU函数

**ReLU函数**

z>0时，梯度始终为1；z<0，梯度始终为0；z等于零时的梯度可以当成1也可以当成0。

对隐藏层，ReLU函数保证z大于零时梯度始终为1，提高神经网络梯度下降算法运算速度。但当z小于零时，存在梯度为0的缺点，实际应用中，这个缺点影响不是很大。

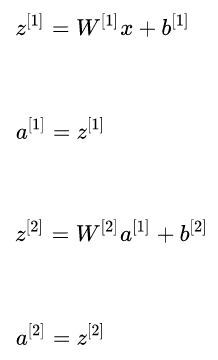
为了弥补这个缺点，出现了**Leaky ReLU激活函数**，能够保证z小于零是梯度不为0。

**总结**

如果是分类问题，输出层的激活函数一般会选择sigmoid函数。但是隐藏层的激活函数通常不会选择sigmoid函数，tanh函数的表现会比sigmoid函数好一些。实际应用中，通常会会选择使用ReLU或者Leaky ReLU函数，保证梯度下降速度不会太小。

### 7 为什么需要非线性函数

如果直接另，即，那么浅层神经网络的各层输出为：





经推导，仍然是输入变量x的线性组合。这样的话神经网络就没有任何作用了。因此，隐藏层的激活函数必须要是非线性的。

### 8. 激活函数的导数

（1）sigmoid函数的导数

****

（2）tanh函数的导数：





（3）ReLU函数的导数：





（3）Leaky ReLU函数的导数：



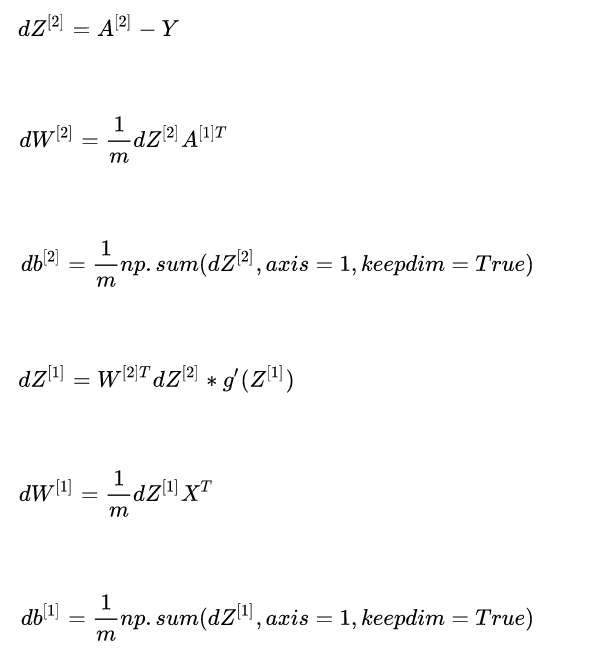


### 9. 神经网络的梯度下降

**正向传播过程：**



**反向传播是计算梯度：**



**证明：**



## 11. 随机初始化

神经网络模型中的参数权重W是不能全部初始化为零的，参数b可以全部初始化为零，并不会影响神经网络训练效果。

如果权重W全部初始化为0，那么每次迭代更新都会得到相同的结果，始终等于，完全对称。这样隐藏层设置多个神经元就没有任何意义了。

**解决方法：**

将和乘以0.01，尽量初始化为比较小的值。之所以让W比较小，是因为如果使用sigmoid函数或者tanh函数作为激活函数的话，W比较小，得到的|z|也比较小（靠近零点），而零点区域的梯度比较大，这样能大大提高梯度下降算法的更新速度，尽快找到全局最优解。

当然，如果激活函数是ReLU或者Leaky ReLU函数，则不需要考虑这个问题。但是，如果输出层是sigmoid函数，则对应的权重W最好初始化到比较小的值。

W\_1 **=** np**.**random**.**randn((2,2))**\***0.01

b\_1 **=** np**.**zero((2,1))

W\_2 **=** np**.**random**.**randn((1,2))**\***0.01

b\_2 **=** 0