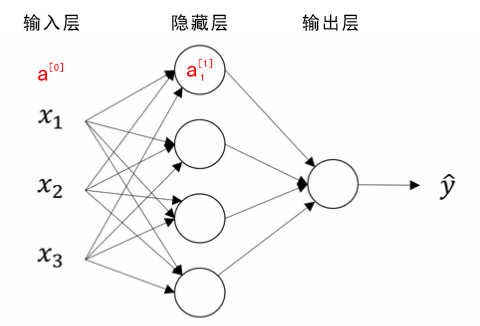
## 浅层神经网络

**1.神经网络表示**



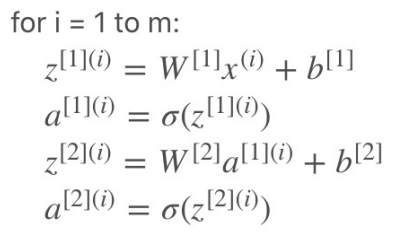
**2.计算神经网络的输出**

神经网络只不过是将Logistic 回归的计算步骤重复很多次

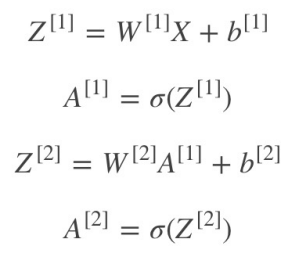
将多个等式向量化进行计算

**3.多样本向量化**

对于 m 个训练样本，在书写标记上用(i) 表示第 i 个样本。对于每个样本 i ，可以使用for循环来求解其正向输出

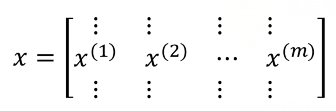


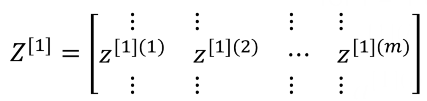
若不用for循环，而是利用矩阵运算的思想

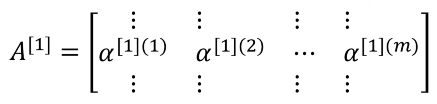


其中，行表示神经元个数n，列表示样本数目m。

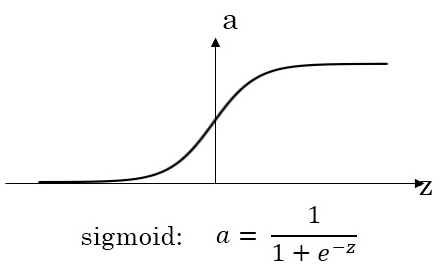
从水平上看，矩阵A代表了各个训练样本。从竖直上看，矩阵A的不同索引对应于不同的隐藏层单元



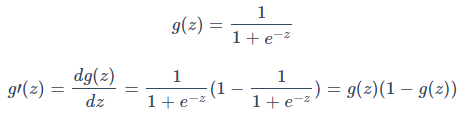




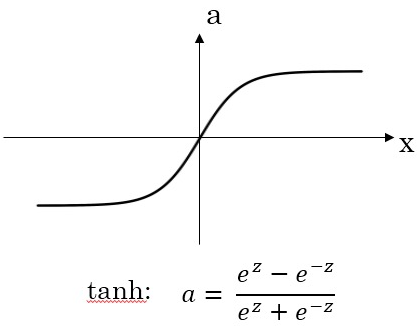
**4.激活函数与激活函数的导数**



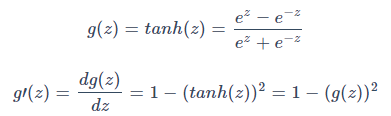
其导数为：



**注意：**在讨论优化算法时，一般情况下是不再使用sigmoid函数的，因为tanh在所有场合都由优于sigmoid函数。但是有一个例外，在二分类问题中，对于输出层，因为y的值是0或1，我们想让y\_hat的值介于0、1之间，而不是-1和+1之间，此时用sigmoid函数会更好。



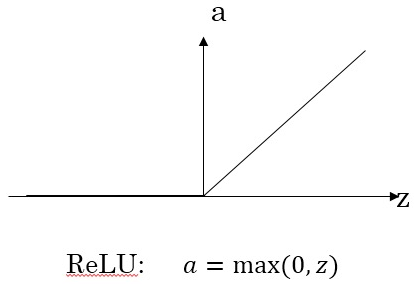
其导数为：



**注意：**

**（1）**tanh的效果总是优于sigmoid函数。因为函数值域在-1和+1的激活函数，其均值是更接近零均值的。在训练一个算法模型时，如果使用tanh函数代替sigmoid函数中心化数据，能够使得数据的平均值更接近0而不是0.5。

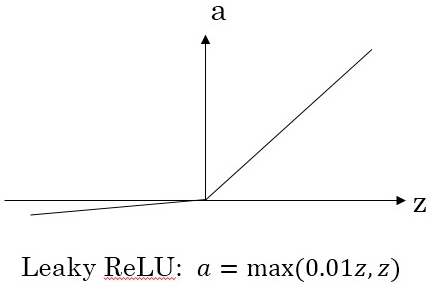
（2）sigmoid函数和tanh函数两者有一个共同的缺点：在z特别大或者特别小的情况下，导数的梯度或者函数的斜率会变得特别小，最后就会接近于0，导致梯度下降的速度减缓



**选择激活函数的经验法则：**

**如果输出的值是0和1（二分类问题），则输出层选择sigmoid函数，然后其他的所有单元都选择ReLu函数。**

**这是很多激活函数的默认选择，如果隐藏层上不确定使用哪个激活函数，那么通常会使用ReLu激活函数。有时，也会使用tanh函数，但ReLuctant的一个优点是：当z是负值时。导数等于0。**



**激活函数总结：**

**（1）在实践中，使用ReLu激活函数的神经网络通常会比使用sigmoid或者tanh激活函数学习的更快。**

**（2）sigmoid函数和tanh函数的导数在正负饱和区的梯度都会接近于0，这会造成梯度弥散。而ReLu和Leaky ReLu函数大于0的部分都为常数，不会产生梯度弥散现象。**

**（3）ReLu进入负区域时，梯度为0，神经元此时不会训练，产生所谓的稀疏性，但是Leaky ReLu不会有这个问题。**

**Tip：如果不确定哪一个激活函数效果更好，可以把它们都试试，然后在验证集或者发展集上进行评价。然后看哪一种表现得更好，就去使用它。**

**5.为什么需要使用非线性激活函数**

使用线性激活函数和不使用激活函数、直接使用 Logistic 回归没有区别，那么无论神经网络有多少层，输出都是输入的线性组合，与没有隐藏层效果相当，就成了最原始的感知器了。

**6.神经网络的梯度下降**

正向传播从输入到输出，由神经网络计算得到预测输出的过程

反向传播从输出到输入，对参数w和b计算梯度的过程。

**7.随机初始化**

如果在初始时将两个隐藏神经元的参数设置为相同的大小，那么两个隐藏神经元对输出单元的影响也是相同的，通过反向梯度下降去进行计算的时候，会得到同样的梯度大小，所以在经过多次迭代后，两个隐藏层单位仍然是对称的。无论设置多少个隐藏单元，其最终的影响都是相同的，那么多个隐藏神经元就没有了意义。

在初始化的时候，w参数要进行随机初始化，不可以设置为0。 而b因为不存在对称性的问题，可以设置为 0。

以 2 个输入，2 个隐藏神经元为例：

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

b = np.zeros((2,1))

这里将 W的值乘以 0.01（或者其他的常数值）的原因是为了使得权重 W初始化为较小的值，这是因为使用 sigmoid 函数或者 tanh 函数作为激活函数时，W比较小，则 z = Wx + bz=Wx+bz=Wx+b 所得的值趋近于0，梯度较大，能够提高算法的更新速度。而如果 W设置的太大的话，得到的梯度较小，训练过程因此会变得很慢。

ReLU 和 Leaky ReLU 作为激活函数时不存在这种问题，因为在大于 0 的时候，梯度均为 1。