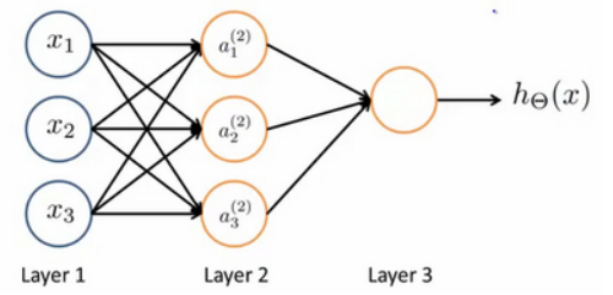
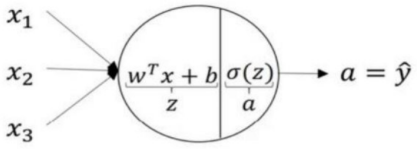
神经网络：

在了解了线性回归和逻辑回归之后，我们发现对于模型来讲，每一个维度始终只用一个参数来拟合，这样会产生一定的偶然性，相当于参加选举，只有一个人进行投票，所以现实中往往采取多数表决的方法，基于该思想，前馈神经网络被提出(多层感知机)。

神经网络模型结构：



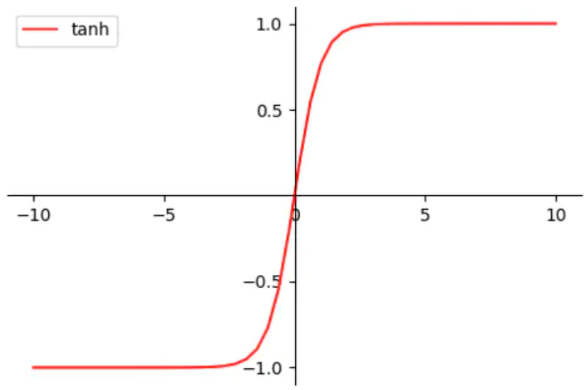
逻辑回归模型结构：



本质上来讲，前馈神经网络就是多个神经元(逻辑回归)堆叠而成，以矩阵的形式观看，就是参数的维度发生了变化，通常维度为(num_features,n)，其中n代表神经元的个数。神经网络的参数更新过程叫做反向传播，实际上也就是个迭代更新的过程。

神经网络含有多种激活函数：

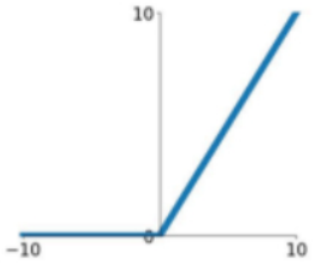
tanh:



$$a = g(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g(z)' = \frac{d}{dz} g(z) = 1 - (tanh(z))^2$$

relu:



$a = max(0, z)$

$$g(x)' = \begin{cases} 0 & if z < 0 \\ 1 & if z > 0 \\ undefined & if z = 0 \end{cases}$$

softmax:

输入的是多行值，预测概率最大的值。

激活函数的作用：

引入非线性激活函数，可以将原数据进行更丰富的变换，从而学到更多特征。

神经网络通常采用relu激活函数，因为sigmoid等函数计算梯度时，涉及到除法和指数运算，计算量大，切位于sigmoid的平缓区和变换区时容易出现梯度消失，梯度爆炸的现象，同时relu会使得部分神经元输出为0，缓解了过拟合。

损失函数：

常采用交叉熵

$$H(y^{(i)}, \hat{y}^{(i)}) = -\sum_{j=1}^{q} y_j^{(i)} log\hat{y}_j^{(i)} = -log\hat{y}_{y^{(i)}}^{(i)}$$

对于神经网络的过拟合问题通常采用Dropout方法，其本质就是使部分神经元随机失活(权重变为0)。