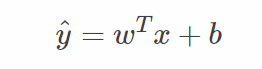
**神经网络逻辑回归**

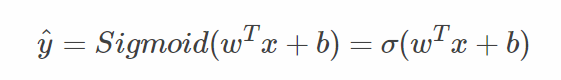
**逻辑回归模型：**一般用来解决二分类问题，二分类就是输出y只有{0,1}两个离散值（也有{-1,1}的情况）。

**逻辑回归解决二分类问题：**

1. **线性回归函数：**

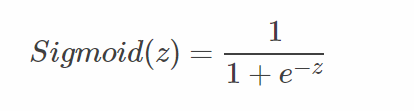


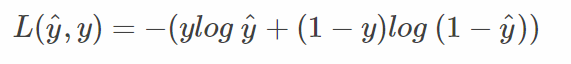
其中，由于y^不在[0,1]之间，故引入了Sigmoid函数，让输出限定在[0,1]之间。

**2.Sigmoid函数：**

其中，Sigmoid函数是一种非线性的S型函数，输出被限定在[0,1]之间，通常被用在神经网络中当作激活函数使用。

Sigmoid函数：

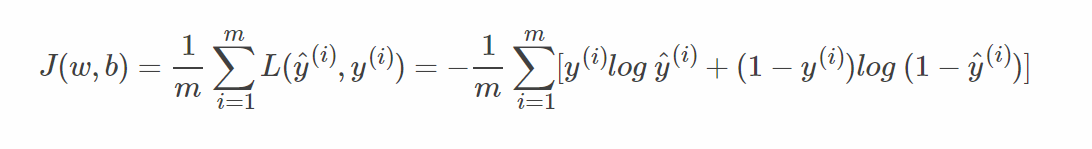


**3.损失函数：**

这个损失函数，它是衡量错误大小的，损失函数越小越好。

这是针对单个样本的函数。

1. **成本函数：**

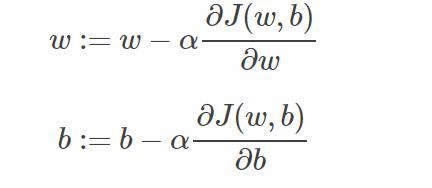


这是针对多个（m个）样本的函数。我们的目标就是迭代计算出最佳的w和b值，最小化成本函数，让成本函数尽可能地接近于零。

1. **梯度下降法：**

运用梯度下降法来操纵w和b的梯度即为偏导值，每次修正w和b，然后迭代一次就代表梯度下降一次，多次迭代后接近于全局最小值。

梯度下降算法每次迭代更新，w和b的修正表达式为：



梯度下降算法能够保证每次迭代w和b都能向着J(w,b)全局最小化的方向进行。

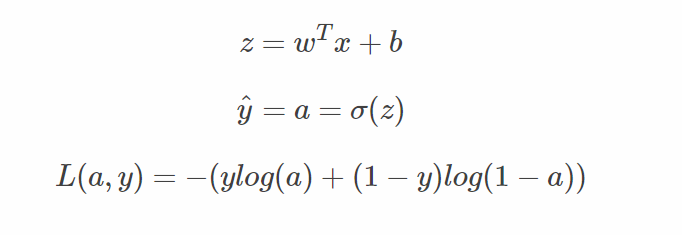
1. **计算图：**

整个神经网络的训练过程实际上包含了两个过程：正向传播和反向传播。

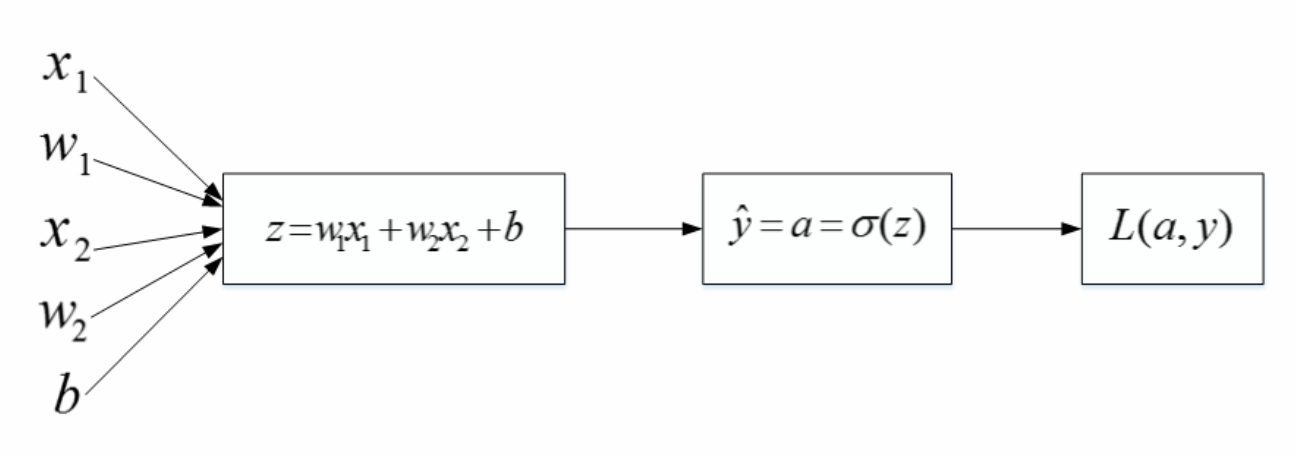
1.正向传播是从输入到输出，由神经网络计算得到预测输出的过程；

2.反向传播是从输出到输入，对参数w和b计算梯度的过程。

一．逻辑回归的正向传播：



例如输入样本x有两个特征(x1,x2)(x1,x2)，相应的权重w维度也是2，即(w1,w2)(w1,w2)。则z=w1x1+w2x2+b。其损失函数正向计算传播图如下所示：



1. 逻辑回归的反向传播：

即为由损失函数从右到左计算w和b的偏导数。

最后用梯度下降法将成本函数逐渐迭代逼近全局最小值。

这是针对单个样本来计算的正反向传播。

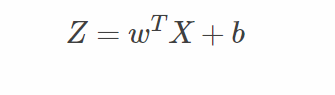
1. **向量化：**

深度学习算法中，数据量很大，在程序中应该尽量减少使用loop循环语句，而可以使用向量运算来提高程序运行速度。

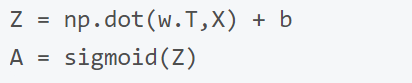
向量化运算就是并行运算，计算机中GPU和CPU都有并行指令（SIMD)，gpu并行速度快于cpu。

最后针对前面的逻辑回归的正反向传播算法，进行向量化后速度大大增加，其**正向传播向量化过程**为：

假设样本构成的输入矩阵X的维度是（nx，m），权重矩阵w的维度是（nx，1），b是一个常数值，而整个训练样本构成的输出矩阵Y的维度为（1，m）。利用向量化的思想，所有m个样本的线性输出Z可以用矩阵表示：

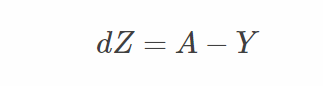


Python代码表示为：

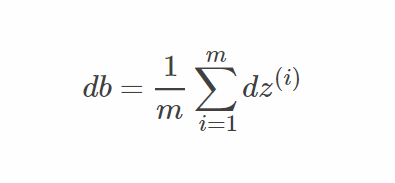


其**反向传播向量化过程**为：

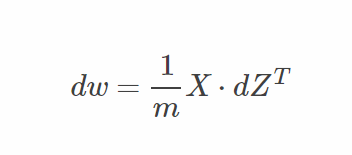
对于所有m个样本，dZ的维度是（1，m），可表示为：



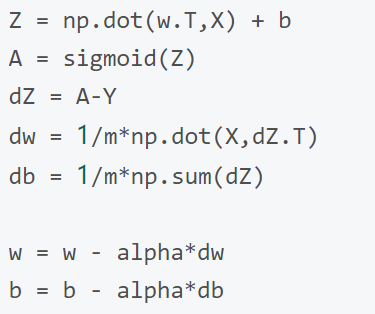
Db可表示为：



Dw可表示为：



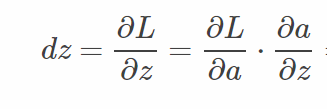
最后正反向传播向量化代码为：



这只是梯度下降的一次迭代，多次迭代还需要用到for循环。

1. **个人课程心得：**

**1.计算图反向传播求导过程中，应用到函数中间值的变化量引起目标变量的改变：如z改变引起a改变，a改变引起L改变，对应求导为：**



**相应的改变量即为对应的求导。**

**2.向量化取代for循环加速本质：将想要得到的目标转化为向量形式。**