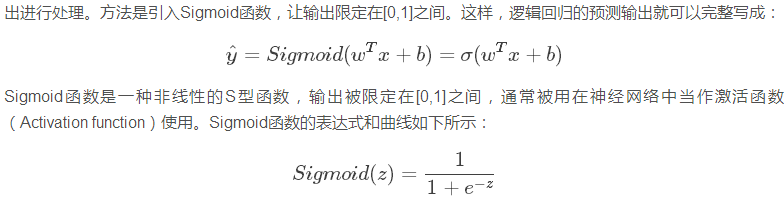
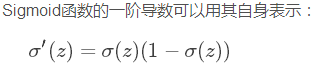
神经网络基础之逻辑回归

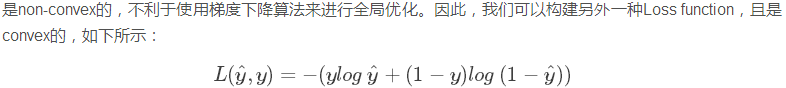
http://blog.csdn.net/red\_stone1/article/category/7143750

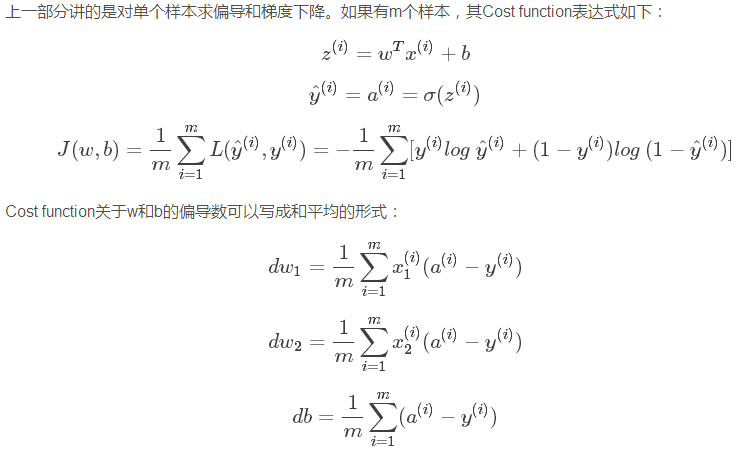
监督式学习与非监督式学习本质区别就是是否已知训练样本的输出y。在实际应用中，机器学习解决的大部分问题都属于监督式学习，神经网络模型也大都属于监督式学习





对于逻辑回归，我们一般不使用平方错误来作为Loss function。原因是这种Loss function一般是non-convex的。non-convex函数在使用梯度下降算法时，容易得到局部最小值（local minumum）





神经网络之向量化与交叉熵

http://blog.csdn.net/red\_stone1/article/details/77929889

没有向量化，则训练神经网络需要三个for。最外层for训练次数，中层for样本数量，内层for参数个数。向量化使用矩阵表示和运算多个样本与多个参数后，只需要一个for用于训练次数。

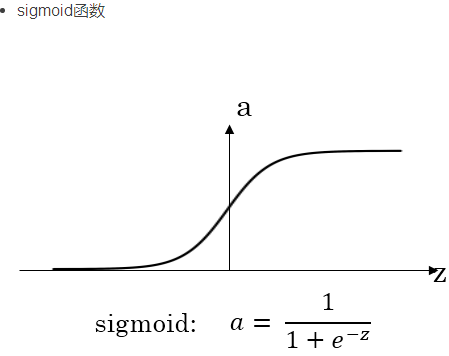
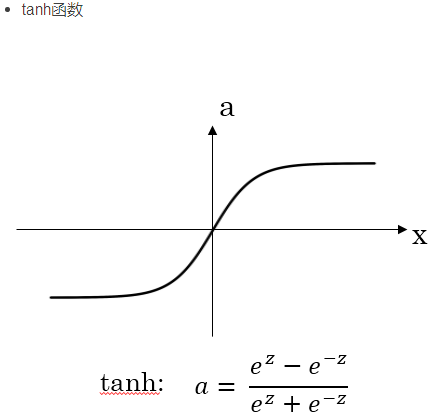
交叉熵loss fuction设计过程

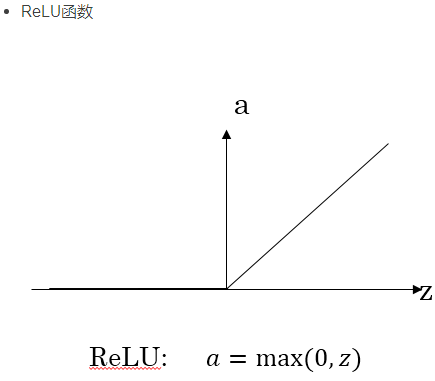
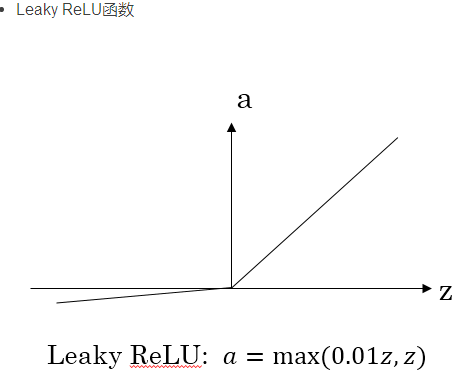
神经网络值浅层网络

http://blog.csdn.net/red\_stone1/article/details/78018269

浅层神经网络运算过程及其向量化

4种常用激活函数及其优劣

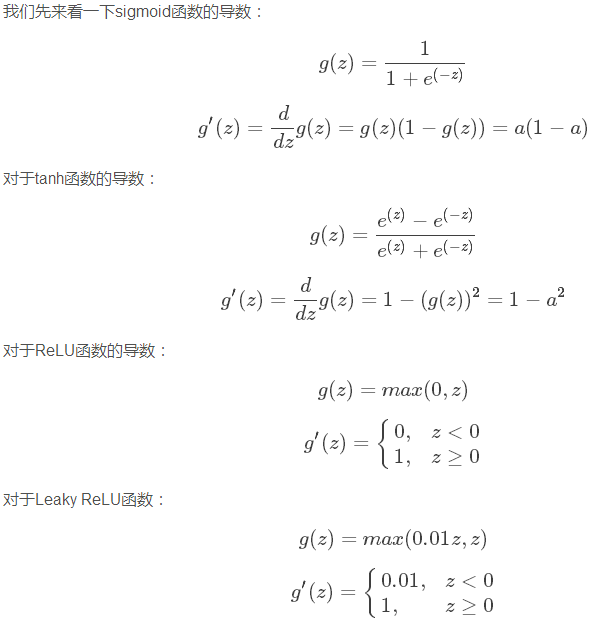
 

对于隐藏层的激活函数，tanh函数要比sigmoid函数表现更好一些。因为tanh函数的取值范围在[-1,+1]之间，均值为0，对数据起到了归一化（均值为0）的效果。 归一化的意义？？？

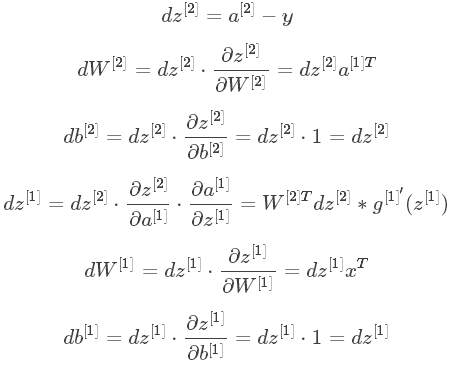
对于输出层的激活函数，因为二分类问题的输出取值为{0,+1}，所以一般会选择sigmoid作为激活函数

sigmoid函数和tanh函数，当|z|很大的时候，激活函数的斜率（梯度）很小。为了弥补sigmoid函数和tanh函数的这个缺陷，就出现了ReLU激活函数。ReLU激活函数在z大于零时梯度始终为1；在z小于零时梯度始终为0。保证了梯度下降算法的速度。

必须使用非线性激活函数。如果使用线性激活函数，则无论添加多少隐藏层，都相当于一层线性运算。

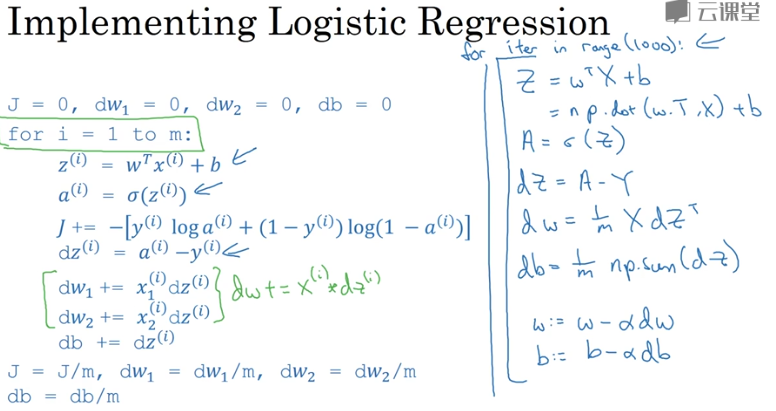


浅层神经网络反向传播计算过程，（输出层采用sigmoid激活函数，隐藏层不确定激活函数类型）



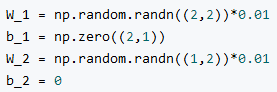
注意到前一神经层参数的梯度计算，依赖于后一神经层参数的梯度计算。（dW1和db1由dz1得出，而dz1由dz2得出。）因此存在梯度消失和梯度爆炸问题

神经网络计算全过程。左边的m为样本数量。右边向量化之后，训练所有样本不再需要显示for循环。那个1000是训练1000遍



神经网络权重参数如果全部设为0，会导致symmetry breaking problem。即每次进行梯度下降迭代后，隐藏层多个神经元的权重参数更新都完全一样，使得多个神经元的设置失去意义。偏差参数不存在此问题

解决方法也很简单，就是将W进行随机初始化（b可初始化为零）。



乘以0.01让W初始值比较小。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

神经网络之深层神经网络 弄清问题，为啥深层比浅层有效？？？