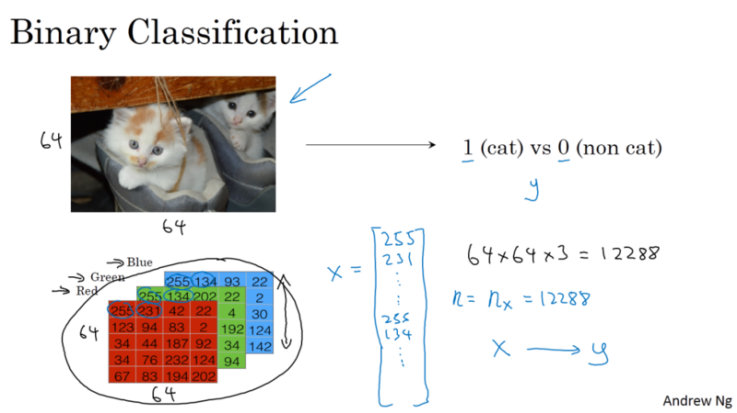
二分类问题

二分类就是输出 y 只有离散值 { 0, 1 }或者 { -1, 1 }。

以一个图像识别问题为例，判断图片中是否有猫存在，0 代表 non cat，1 代表 cat。



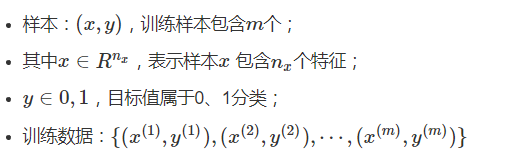
一般来说，彩色图片包含RGB三个通道。我们首先要将图片输入x（维度是（64，64，3））转化为一维的特征向量。方法是每个通道逐行提取，最后连接起来，转化后的输入特征向量维度为（64x64x3=12288）。此特征向量x是列向量，维度一般记为nx。

如果训练样本共有m张图片，那么整个训练样本X组成了矩阵，维度是（，m）。

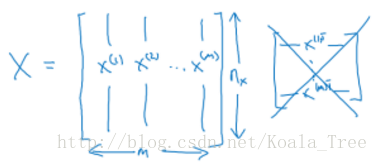
注意，这里矩阵X的行代表了每个样本特征个数，列m代表了样本个数。

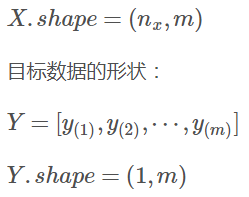
所有训练样本的输出Y也组成了一维的行向量，写成矩阵的形式后，它的维度就是（1，m）。

对于二分类问题，大牛给出了一个小的Notation。



输入神经网络时样本数据的形状：



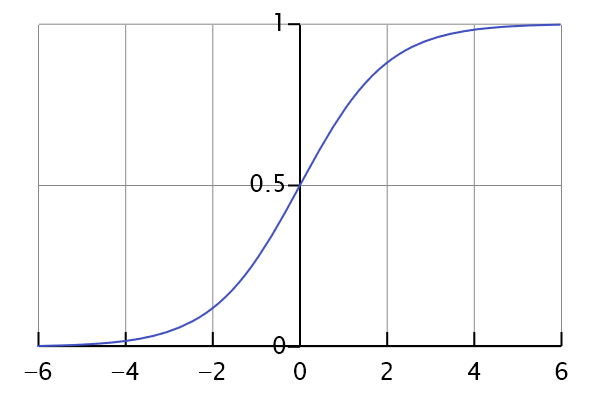


logistic回归(逻辑回归)

Logistic 回归虽然名字叫回归，但是它是用来做分类的。其主要思想是: 根据现有数据对分类边界线建立回归公式，以此进行分类。

如何使用逻辑回归来解决二分类问题？

逻辑回归中，预测值表示为1的概率，与二分类不同，取值范围在[0,1]之间。



使用线性模型，引入权重参数w和偏置参数b。权重w的维度是（，1），b是一个常数项。这样，逻辑回归的线性预测可以写成： y=wx+b

上式的线性输出区间为整个实数范围，而逻辑回归要求输出范围在[0,1]之间，所以需要引入Sigmoid函数对输出进行处理.

引入Sigmoid函数，预测值:

图片

Sigmoid为：图片

其对x的导数可以用自身表示:

图片

使用logistic回归训练的成本函数

图片

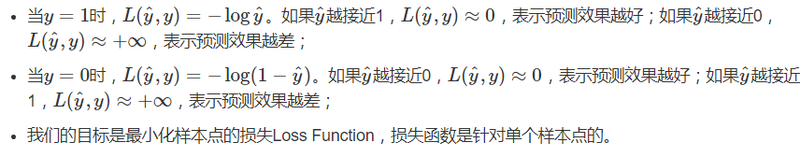
logistic回归的损失函数（Loss Function）

现在来看一下损失函数或者叫做误差函数

1. 他们可以用来衡量算法的运行情况
2. 可以定义损失函数为y^和y的差，或者他们差的平方的一半，结果表明你可能这样做，但是实际当中，大家都不会这么做，因为当你学习这些参数的时候，你会发现之后讨论的优化问题，会变成非凸的，最后会得到很多的局部最优解，梯度下降算法可能找不到最优的全局最优值，
3. 直观理解就是我们定义这个损失函数L，来衡量你的预测输出值y^和y的实际值有多接近，误差平方看起来是一个合理的选择，但是如果用这个的话，梯度下降法就不会很好用，
4. 在logistic回归中，我们会定义一个不同的损失函数，它起着于误差平方相似的作用，这会给我们一个凸的优化问题，他很容易去做优化，

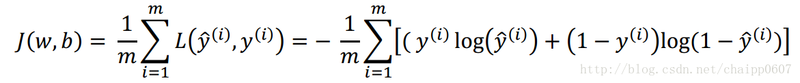
**Loss function:**

图片



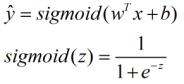
**cost function:**

而对于全体样本集的成本函数，就可以表示为：



下面证明下这个损失函数是怎么来的：

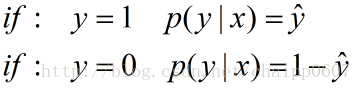
Logistic回归模型如下：



那么令y\_hat为给定x的情况下y=1的概率：

图片

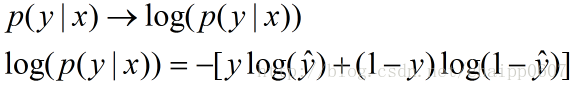
那么则有：



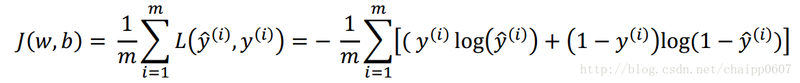
由于是个二分类问题，y的值非1即0，那么合并上式就可得到：

图片

同时由于log函数是严格单调递增的函数，在机器学习中，我们往往不太关注log的底数到底是什么，甚至直接省略掉，所以出现了log的写法，但是在数学中这样写是错的。所以，为了后续求解方便，我们可以取对数：



而对于成本函数来说，他对于整个训练集优化w和b，所以就有了这个上面出现过的式子：



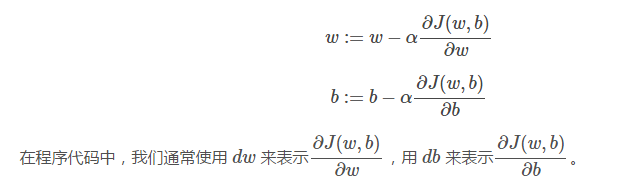
在这里其实是可以用最大似然估计的方法来求这个解的，但是在实际的优化中，我们往往直接使用梯度下降法

* Cost function是待求系数w和b的函数；
* 我们的目标就是迭代计算出最佳的w和b的值，最小化Cost function，让其尽可能地接近于0。

梯度下降

用梯度下降法（Gradient Descent）算法来最小化Cost function，以计算出合适的w和b的值。

由于J(w,b)是凸函数，梯度下降算法是先随机选择一组参数w和b，然后迭代的过程中分别沿着w和b的梯度的反方向前进一小步，不断修正w和b。梯度下降算法每次迭代更新，w和b的更新表达式为：



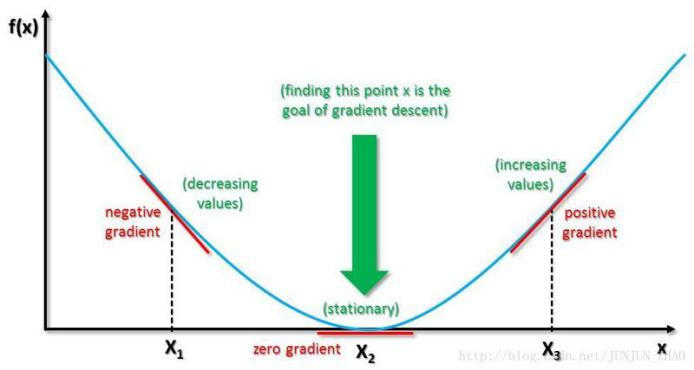
**偏导数符号** 使用 ∂还是小写字母 d：**取决于你的函数 J 是否含有两个以上的变量** :

**1. 变量超过两个就用偏导数符号** ∂，

**2.如果函数只有一个变量就用小写字母 d。**

上式中，α是学习率（learning rate），表示梯度下降的步伐大小。α越大，w和b每次更新的“步伐”更大一些。

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。

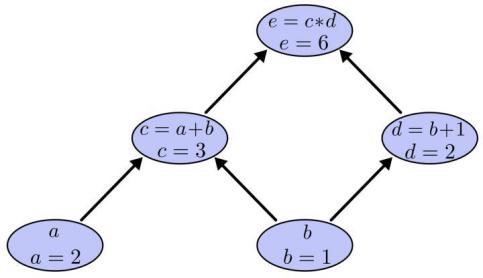


计算图

(来自知乎：https://zhuanlan.zhihu.com/p/21612460)

神经网络的训练过程包含了正向传播（Forward Propagation）和反向传播（Back Propagation）。

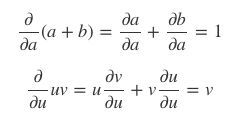
我们用计算图（Computation graph）的形式来解释这两个过程，例如，考虑表达式：e=(a+b)\*(b+1)，这里有三个操作，两次加法和一次乘法。



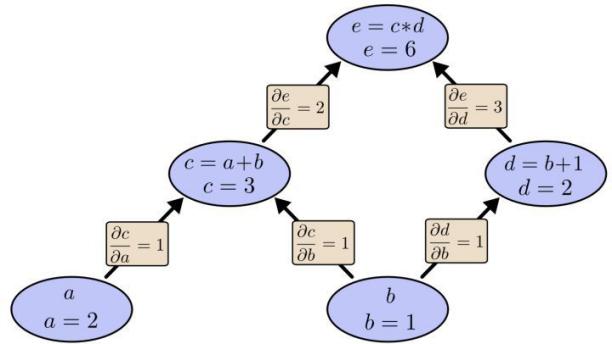
**可计算图形求导**

要理解可计算图形的求导，关键是要理解各条边上的导数。如果a直接影响到c，那么我们希望知道a是如何影响到c。如果a改变了一小点，c将发生多大的改变？我们将其称之为c对于a的偏导数。

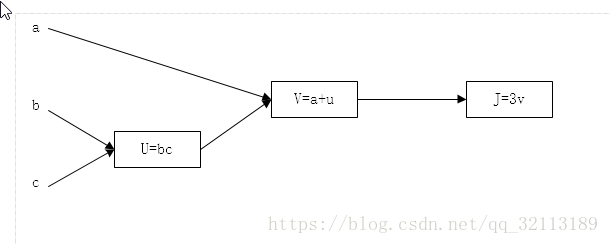
为了计算图中的偏导数，我们需要加法和乘法规则：



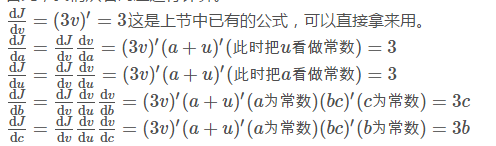
下图的各条边加上了偏导数标签。



再看看NG课程中的列子。

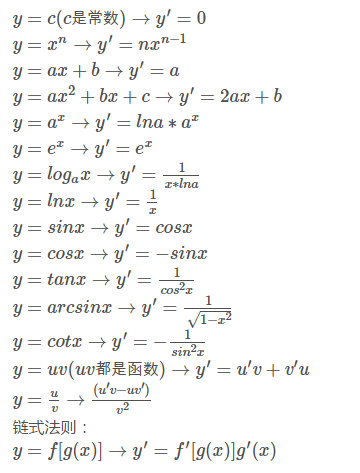


首先，我们从右向左进行计算。



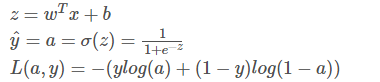
由此，我们看到，函数的反向传播主要用到的求导的链式法则，熟悉导数的猿们应该对链式法则会比较清楚，不清楚的要再去复习一下链式法则的精要概念，千万不要把常量和变量弄混，否则，会变的更加复杂。

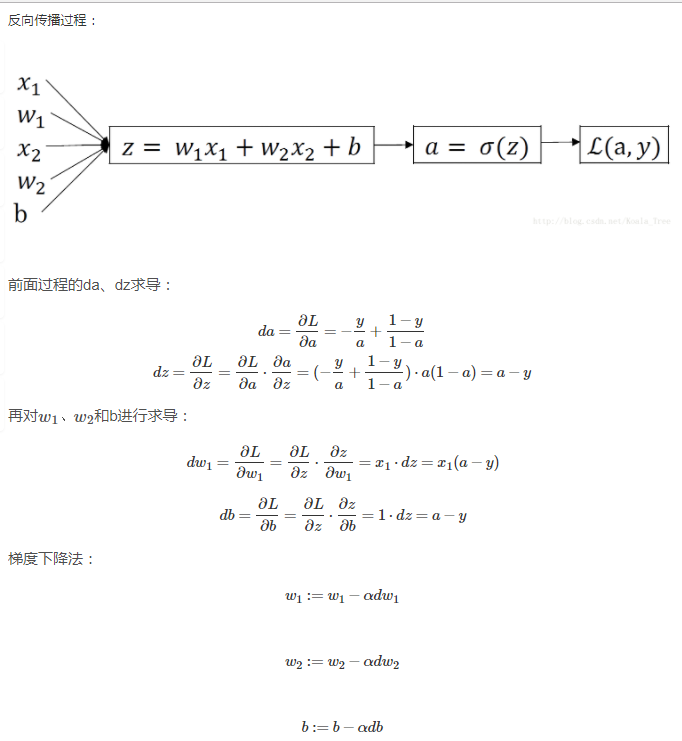
另外，向大家列出一些经常用到的函数求导公式：

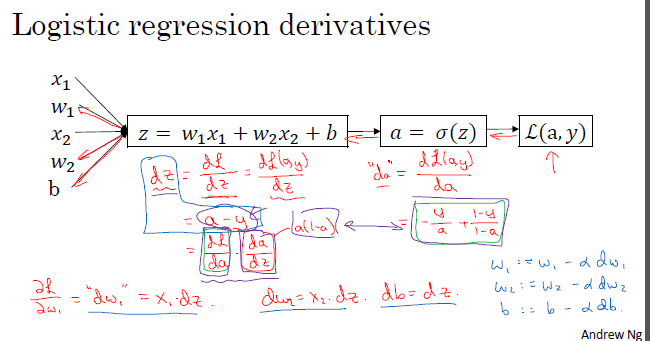


logistic回归的梯度下降法

本节课是按照三个函数来进行的，由3个函数进行梯度下降算法的计算。



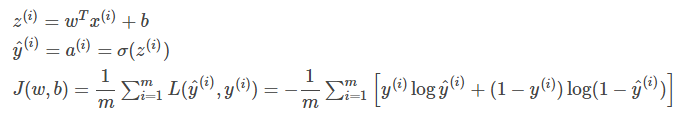




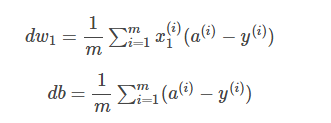
由此，我们可以知道单个样本的梯度下降算法的计算过程是怎样进行的，那么下节课我们将看多种样本的梯度下降算法的计算过程。

m个样本的梯度下降法

对 m 个样本来说，其 Cost function 表达式如下：



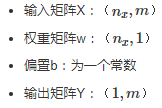
Cost function 关于w和b的偏导数可以写成所有样本点偏导数和的平均形式：



向量化的logistic回归

在深度学习的算法中，我们通常拥有大量的数据，在程序的编写过程中，应该尽最大可能的少使用loop循环语句，利用python可以实现矩阵运算，进而来提高程序的运行速度，避免for循环的使用。

逻辑回归向量化



所有m个样本的线性输出Z可以用矩阵表示：

图片

在 numpy 中可以表示为：

Z = np.dot(w.T,X) + b

A = sigmoid(Z)

dZ对于m个样本，维度为（1,m）（1,m），表示为：

图片

db可以表示为：

图片

python代码：

db = 1/m\*np.sum(dZ)

dw可表示为：

图片

python代码：

dw = 1/m\*np.dot(X,dZ.T)

单次迭代梯度下降算法流程

Z = np.dot(w.T,X) + b

A = sigmoid(Z)

dZ = A-Y

dw = 1/m\*np.dot(X,dZ.T)

db = 1/m\*np.sum(dZ)

w = w - alpha\*dw

b = b - alpha\*db

其中，alpha是学习率，决定w和b的更新速度。

python的notation

* 虽然在Python有广播的机制，但是在Python程序中，为了保证矩阵运算的正确性，可以使用reshape()函数来对矩阵设定所需要进行计算的维度，这是个好的习惯；
* 如果用下列语句来定义一个向量，则这条语句生成的a的维度为（5，），既不是行向量也不是列向量，称为秩（rank）为1的array，如果对a进行转置，则会得到a本身，这在计算中会给我们带来一些问题。

a = np.random.randn(5)

* 如果需要定义（5，1）或者（1，5）向量，要使用下面标准的语句：

a = np.random.randn(5,1)

b = np.random.randn(1,5)

可以使用assert语句对向量或数组的维度进行判断。assert会对内嵌语句进行判断，即判断a的维度是不是（5，1），如果不是，则程序在此处停止。使用assert语句也是一种很好的习惯，能够帮助我们及时检查、发现语句是否正确。

assert(a.shape == (5,1))

* 可以使用reshape函数对数组设定所需的维度

a.reshape((5,1))

logistic regression代价函数的解释

Cost function的由来：

预测输出y^的表达式：

图片

其中

图片

y^可以看作预测输出为正类（+1）的概率：

图片

