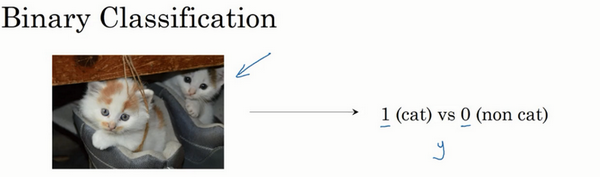
# 神经网络的编程基础

## 二分类

这周我们学习神经网络的基础知识，实现神经网络的基本技术和技巧。了解前向传播和反向传播算法。我们将使用逻辑回归来传达这些想法，这样理解起来会更容易。

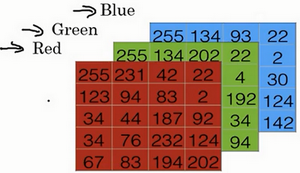
逻辑回归是一个用于二分类问题的算法，我们在机器学习课程中已经学习过，在此不进行赘述。

我们以识别猫片为例，我们将要给出一系列图片，并识别其是否是一只猫。

如图所示，

我们首先看一幅照片在计算机中是如何存储的

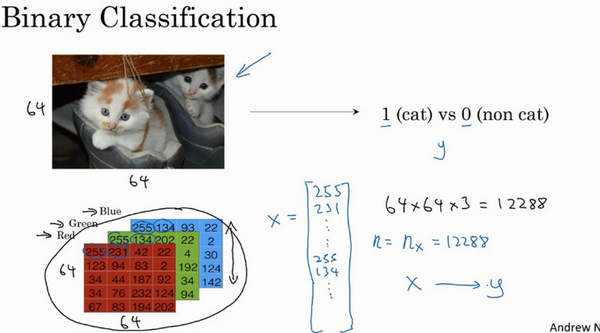
首先，为了保存一张图片，我们需要三个矩阵，分别对应于RGB三种不同颜色的通道，如果你照片的像素为n\*n，那么你就拥有三个n\*n规模的矩阵，如图所示



为了把这些像素值放到一个特征向量中，我们需要把这些像素都提取出来，然后放入特征向量x，如果图片的大小为64x64像素，那么向量 x 的总维度，将是64乘以64乘以3。

一个向量x即表示一张图片，我们的目标就是把这个特征作为输入，得到相应的输出。

如图所示



此时的特征x和我们机器学习的课程中不同，我们不需要增加额外特征x0。

若一个多样本的矩阵X，维度为n\*m，那么n为特征数量， m为样本数量。在深度学习课程中，将每个特征表示为列向量会使我们实现算法更加方便。

而Y是1\*m的行向量，表示输出特征。

## 逻辑回归

具体的逻辑回归算法在机器学习课程中已有介绍，我们将要用逻辑回归算法来实现我们的二分类器。

在这门课程中，我们不在使用θ作为参数表示，而使用ω和b，ω作为特征的系数，b作为偏置项

## 逻辑回归的代价函数

逻辑回归中代价函数采用极大似然估计的形式，而不用平方损失函数。

损失函数指当个样本的损失，而代价函数是所有样本的总和，逻辑回归的损失函数如图所示，至于为什么不用平方损失，我会在其他文章中给出说明。

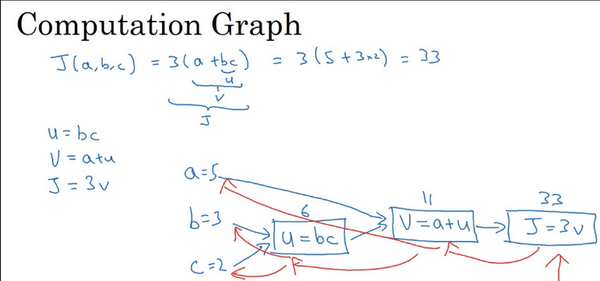
逻辑回归的损失函数便是如此

所以我们在训练模型时，要找到合适的参数，使总体的代价函数最小

## 计算图

可以说，一个神经网络的计算，都是按照前向传播和反向传播的方式组织的，首先我们计算出一个新的网络输出（前向过程），紧接着进行反向传播，用来计算出对应的梯度或者导数。

我们以如下的计算为例

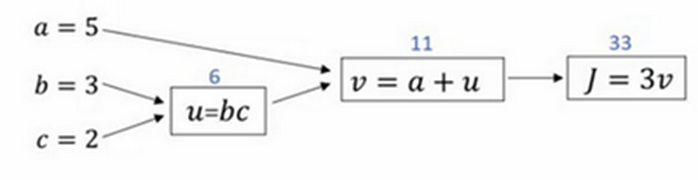


我们需要计算一个函数J，J中有三个参数，a,b,c。这个函数是J = 3\*(a+bc)，计算这个函数有三个不同的步骤，首先要计算b\*c，我们将其储存在u中，接着计算(a+u),我们将其储存在v中，最计算J = 3v，以此步骤，我们就得出了计算图。以此图为例，我们输入a=5,b=3,c=2，

此使我们计算u = 3\*2 = 6,接着计算v = 5+6 = 11，最后计算J = 3\*11 =33  
当有不同的或者一些特殊的输出变量时，例如本例中的J和逻辑回归中你想优化的代价函数J，因此计算图用来处理这些计算会很方便。可以看出，计算图实际就是从左到右的逐步计算。

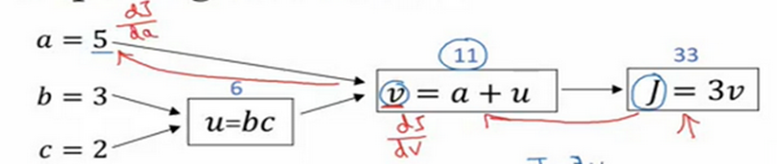
## 使用计算图求导数

经过上一小节，我们已经知道了如何通过前向传播求出最后的函数值，现在，我们将用反向传播来求出各个参数的导数值，仍旧以此图为例



假设我们要计算中各个中间量的导数值，我们要怎么计算呢？

首先我们计算出，接着我们要计算和，由于我们已经计算出了，我们可以根据微积分中的链式法则，得到，，而b和c的导数则通过链式法则由u的导数求得，如此便可很方便的计算出各个参数的偏微分，如图所示



这就是一个计算图的流程，我们进行前向传播计算出输出值，并根据输出值进行反向传播，求出各个变量的偏微分，以此进行梯度下降。

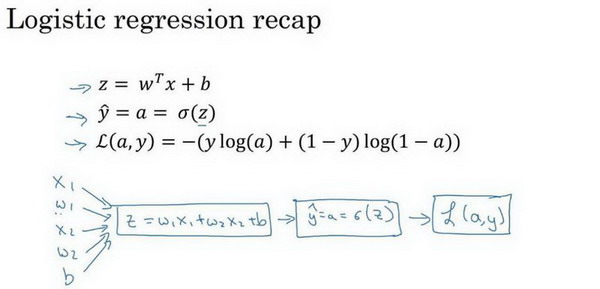
## 计算图中的逻辑回归梯度下降

为了加深对计算图的理解，我们使用反向传播算法来对逻辑回归中的参数进行梯度下降，虽然在事实上这是杀鸡用牛刀

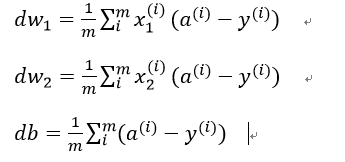
假设我们的样本只有两个特征值，x1和x2,为了计算Z（即sigmoid函数的参数），我们需要参数ω1，ω2和b，z的计算如右，

我们的代价函数为

现在我们仅考虑当个样本的情况，单个样本的代价函数为

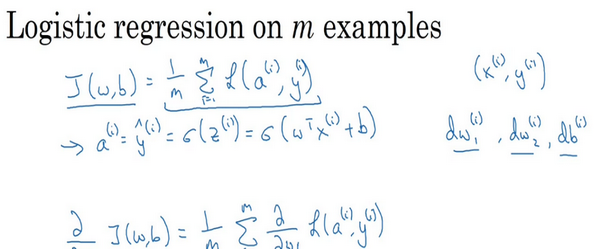
现在我们画出如图所示的计算图

为了使得代价函数能够最小化，我们需要更改参数ω和b，通过链式法则来求偏微分，我们得到 通过这个，我们可以得到接下来的偏微分

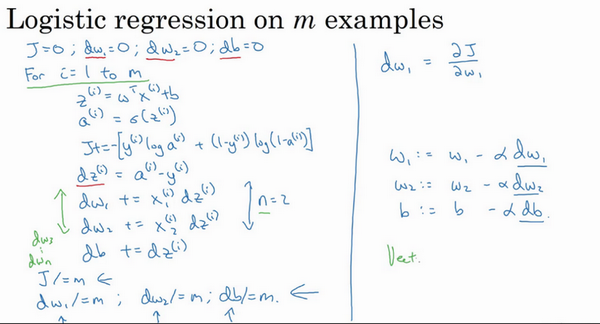


## 7.M个样本的梯度下降

首先，我们的代价函数定义为，我们需要对此进行求偏微分，并利用梯度下降公式，不管怎么求导，Σ符号都是放最前面的，而且要先把Σ后的式子运算完才开始求和。如图所示



我们的代码流程如图所示



我们需要从遍历从1到m所有样本，计算出z，a，与dw，db的总和，求出遍历一轮所有样本后的db，dw平均值，并进行梯度下降更新参数

需要很注意的一点是，这里的dz只是作为累加器使用，并不参与叠加，因为对初始参数求导后，Σ符号仍然存在，若将dz求导后再和x相乘，会得到错误的结果，造成拟合不充分。同时，dz是表示损失函数对z的求导，而dw和db都是表示整体代价函数的导数。

## 向量化的逻辑回归

对这些式子进行向量化处理后，由于CPU与gpu的并行计算特性，我们能得到更快的计算结果。