## 神经网络基础

**主要内容：**

1、介绍二分类问题，以图片为例，将多维输入（x）转化为特征向量，输出（y）离散值 { 0, 1 }；

2、介绍逻辑回归（logistic）及其对应的代价函数（Cost function）形式；

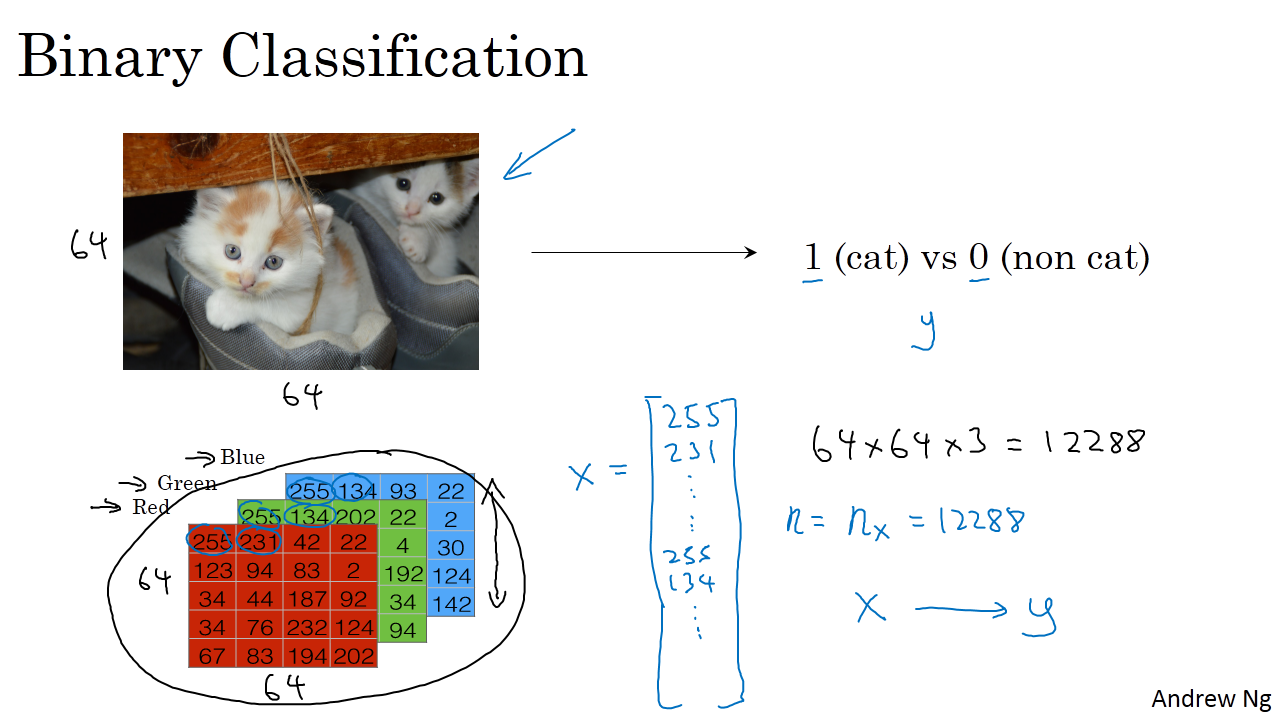
3、介绍梯度下降算法，使用计算图（compution graph）描述神经网络的正向、反向传播过程；

4、在逻辑回归中使用梯度下降（Gradient Descent）算法，总结出优化权重参数 w 和偏置参数 b 的算法流程。

### 二分类（binary classification）

输出的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 代表无猫，1代表有猫。



一般彩色图片包含RGB三个通道，首先将图片输入x（维度是（64，64，3））转化为一维的特征向量（feature vector）。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=12288，1）。此特征向量x是列向量，维度一般记为nx。

### 逻辑回归(logistic regression)

逻辑回归中，预测值表示为1的概率，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。w的维度是，b是一个常数项。则逻辑回归的线性预测为：



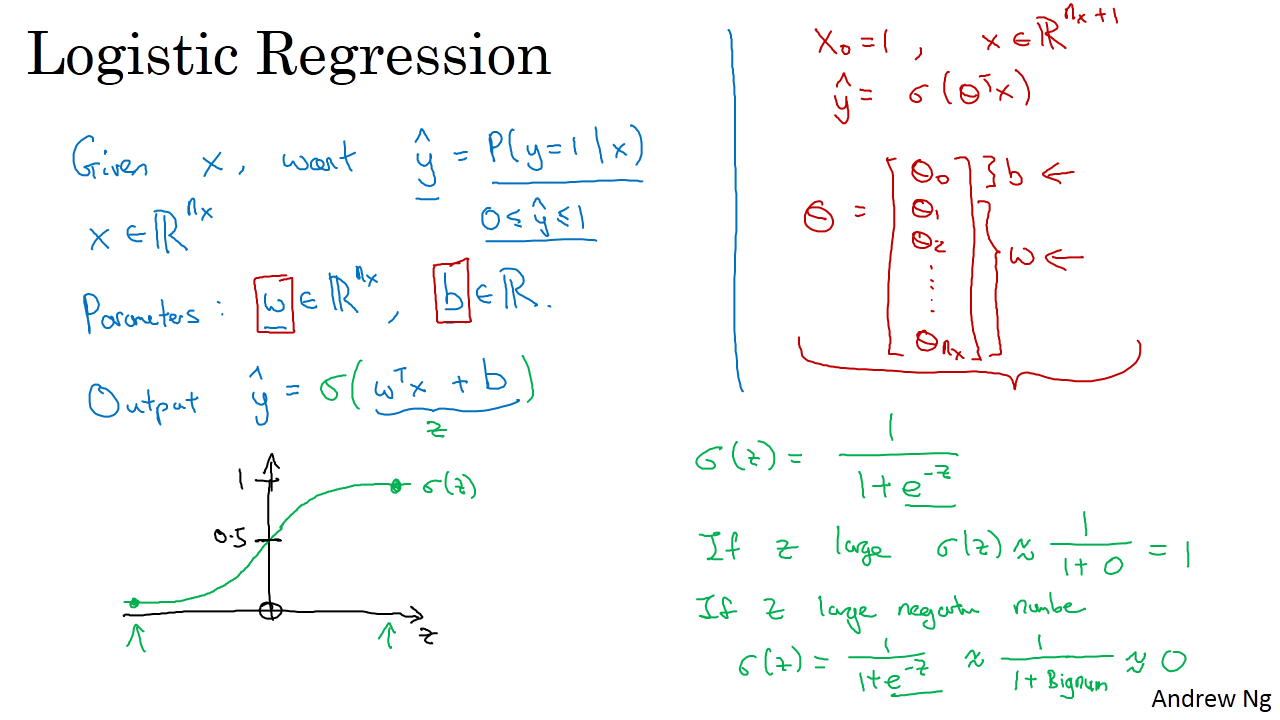
上式的输出范围是整个实数范围，为了使y属于[0,1]，引入sigmoid函数，让输出限定在[0,1]之间。则：





Sigmoid一阶导数可用其自身表示：



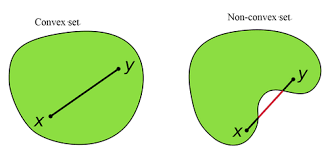


### 代价函数(cost function)

通过优化代价函数，得到对应的w和b。从单个样本的角度，希望样本的预测值和真实值越接近越好，若使用平方误差（squared error），如下：



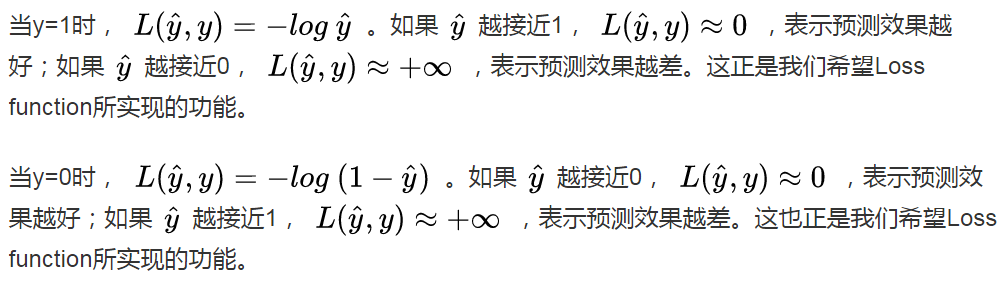
但逻辑回归，一般不用上式作为损失函数，因为上式是非凸函数，非凸函数在梯度下降时，容易得到局部最小值，所以一般选的LOSS function是凸函数。



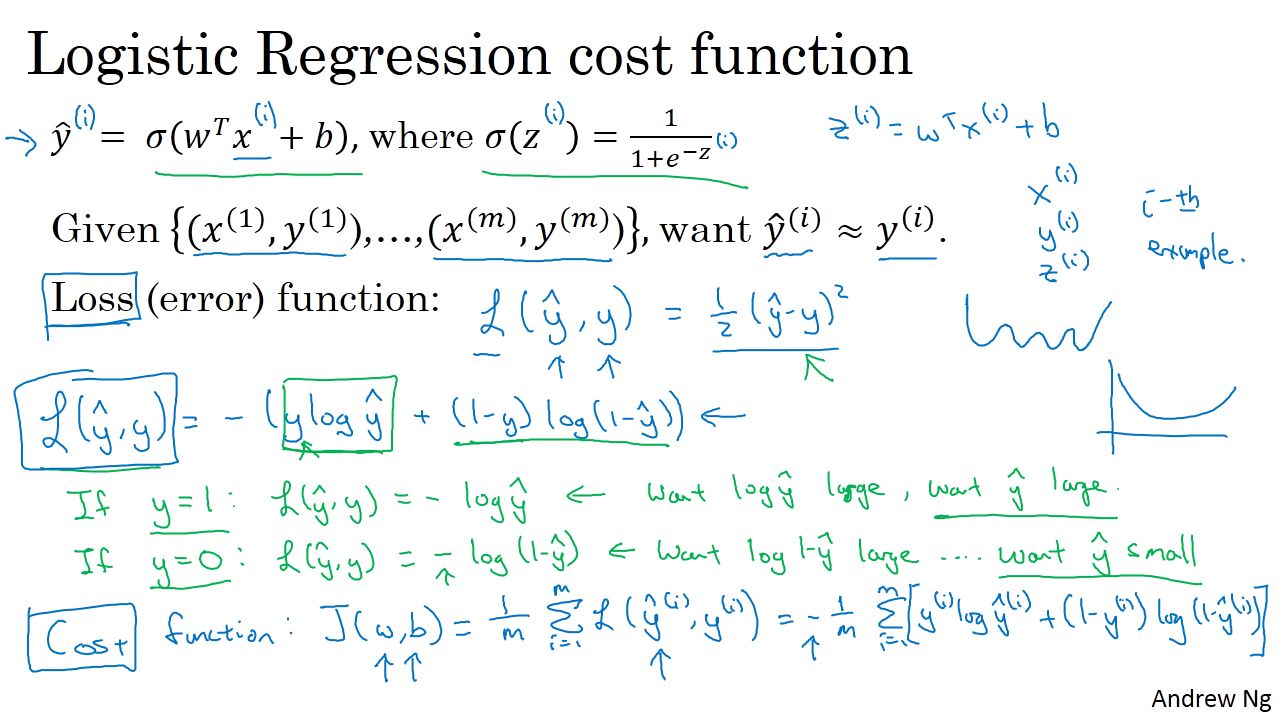
**图1 凸函数与非凸函数**

所以选用如下Loss function:



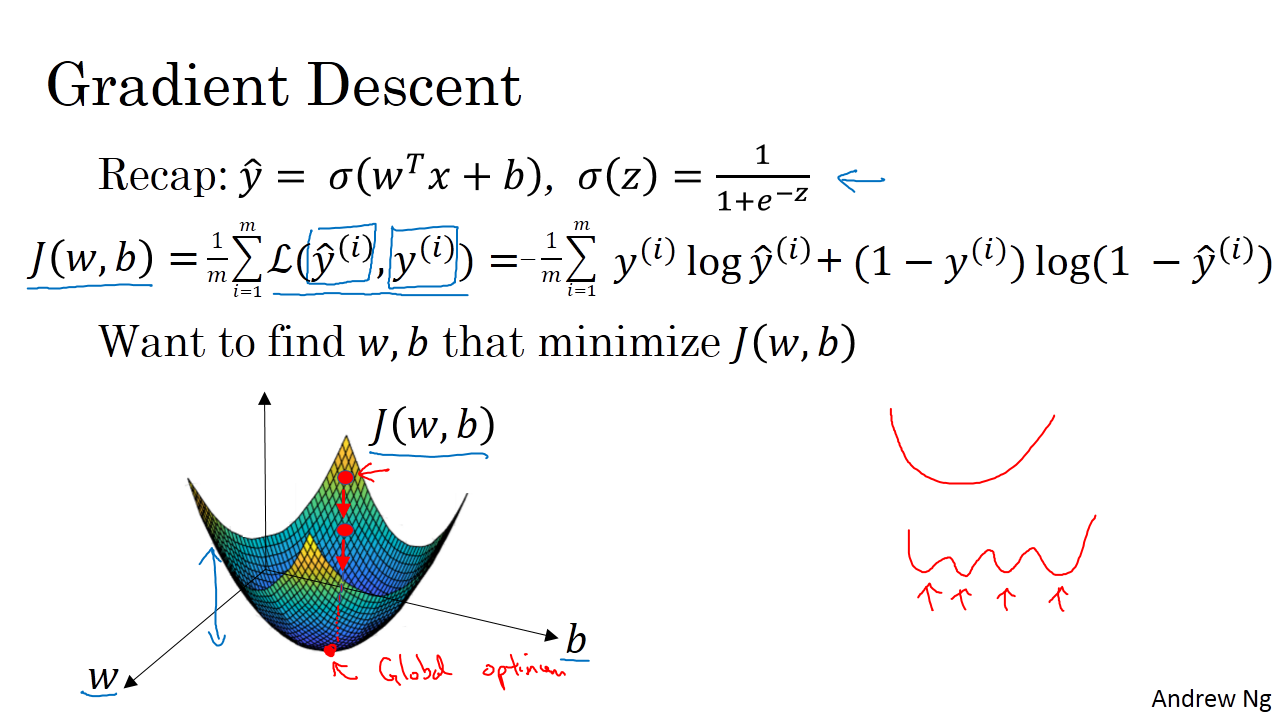


上面是单个样本，m个样本的Cost function反应的是m个样本的平均接近程度。



### 梯度下降（gradient descent）

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：

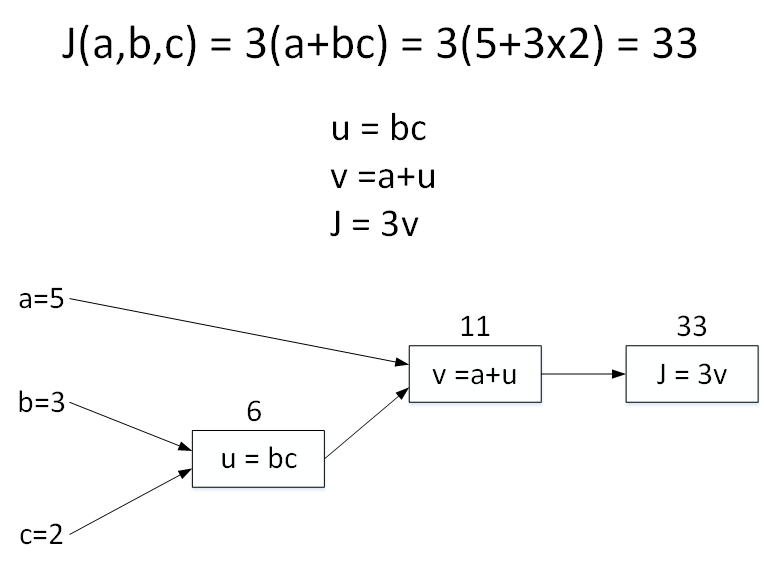




上式中是学习因子（learning rate）

### 计算图(Computation graph) 整个神经网络的训练包含两个过程：正向传播(Forward Propagation)和反向传播。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；反向传播是从输出到输入，对参数w和b计算梯度的过程。下面，我们用计算图（Computation graph）的形式来理解这两个过程。

举个简单的例子，假如Cost function为J(a,b,c)=3(a+bc)，包含a，b，c三个变量。我们用u表示bc，v表示a+u，则J=3v。它的计算图可以写成如下图所示：



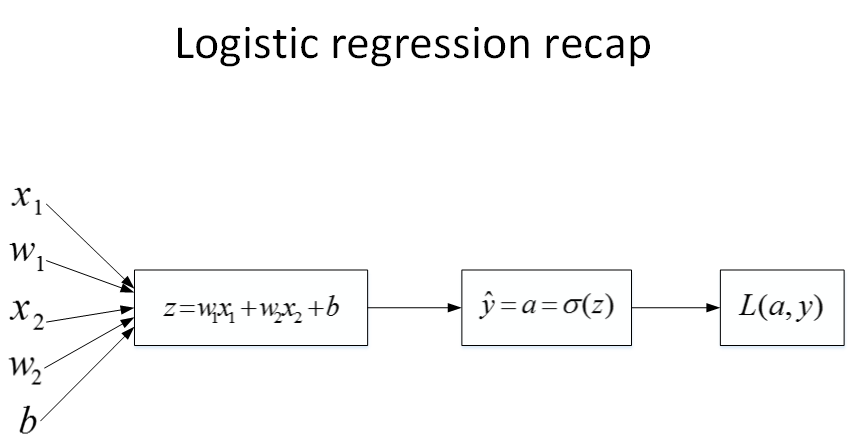
### 逻辑回归的梯度下降

对单个样本而言，逻辑回归Loss function表达式如下：

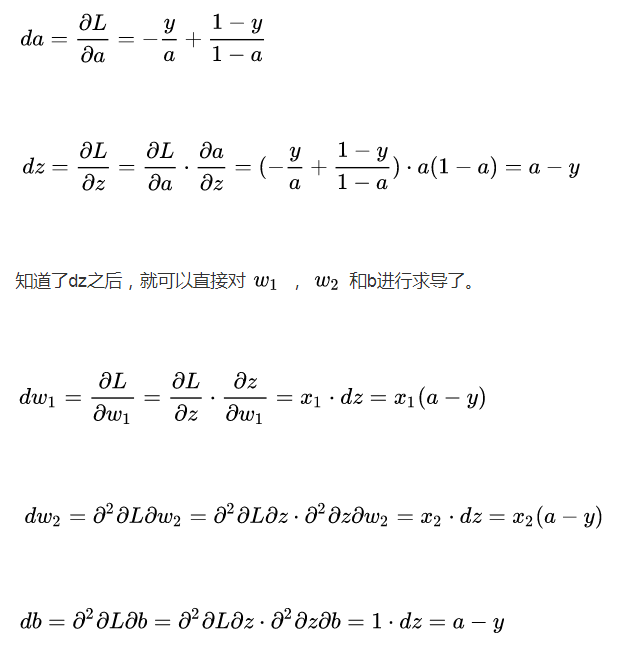




正向传播过程如下图



反向传播过程即由Loss function计算参数w和b的偏导数。推导过程如下：

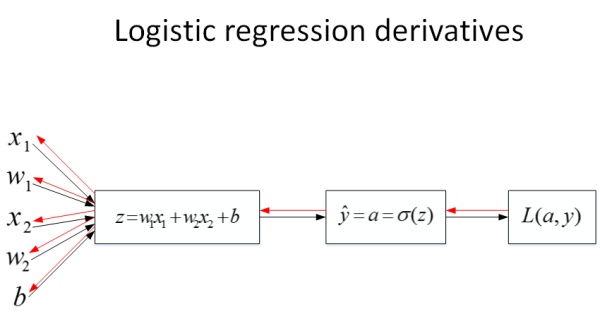


则梯度下降算法可表示为：

w1:=w1−α dw1

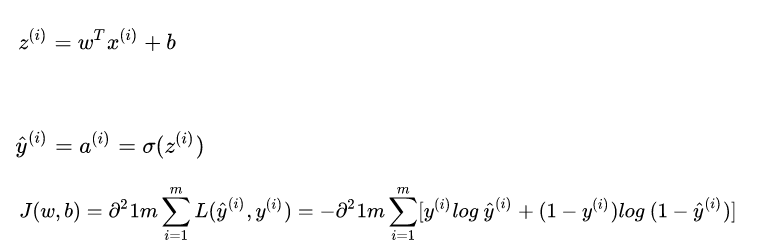
w2:=w2−α dw2

b:=b−α db

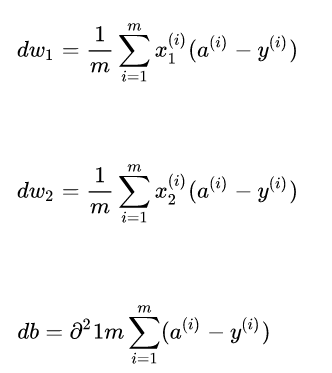


### M个样本的梯度下降

上一部分讲的是对单个样本求偏导和梯度下降。如果有m个样本，其Cost function表达式如下：



Cost function关于w和b的偏导数可以写成和平均的形式：



这样，每次迭代中w和b的梯度有m个训练样本计算平均值得到。其算法流程图如下所示：

J=0; dw1=0; dw2=0; db=0;

for i = 1 to m

z(i) = wx(i)+b;

a(i) = sigmoid(z(i));

J += -[y(i)log(a(i))+(1-y(i)）log(1-a(i));

dz(i) = a(i)-y(i);

dw1 += x1(i)dz(i);

dw2 += x2(i)dz(i);

db += dz(i);

J /= m;

dw1 /= m;

dw2 /= m;

db /= m;

