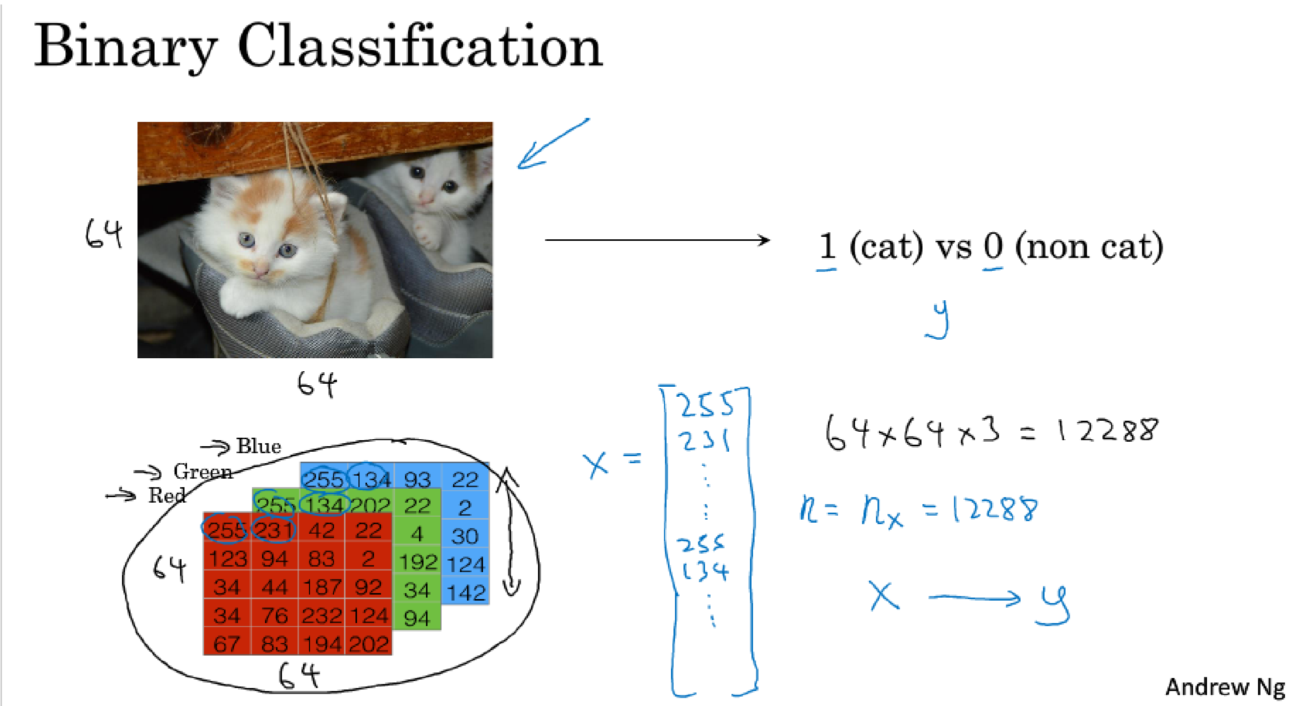
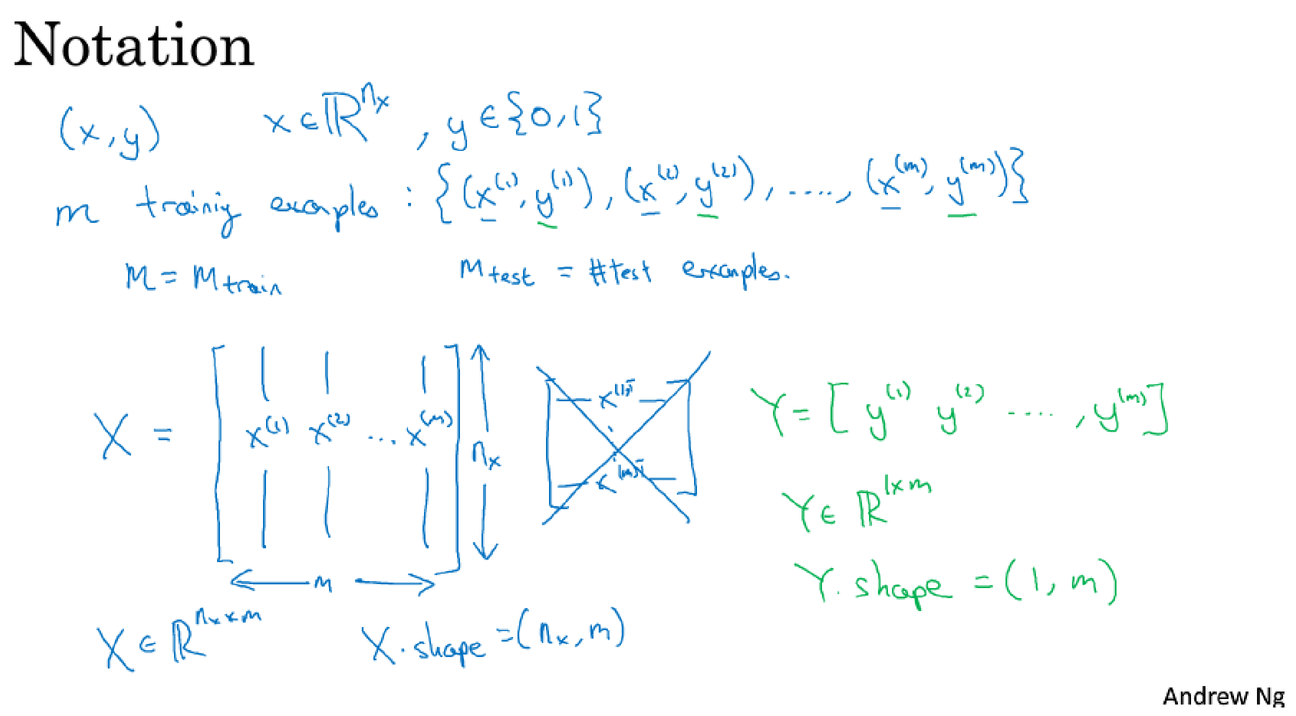
神经网络和深度学习--神经网络基础

2-1 二分分类



对于二分类问题，大牛给出了一个小的Notation。

- 样本：(x,y)，训练样本包含m个；
- 其中x∈Rnx，表示样本x 包含nx个特征；
- y∈0,1，目标值属于0、1分类；
- 训练数据：{(x(1),y(1)),(x(2),y(2)),⋯,(x(m),y(m))}



2.2 逻辑回归

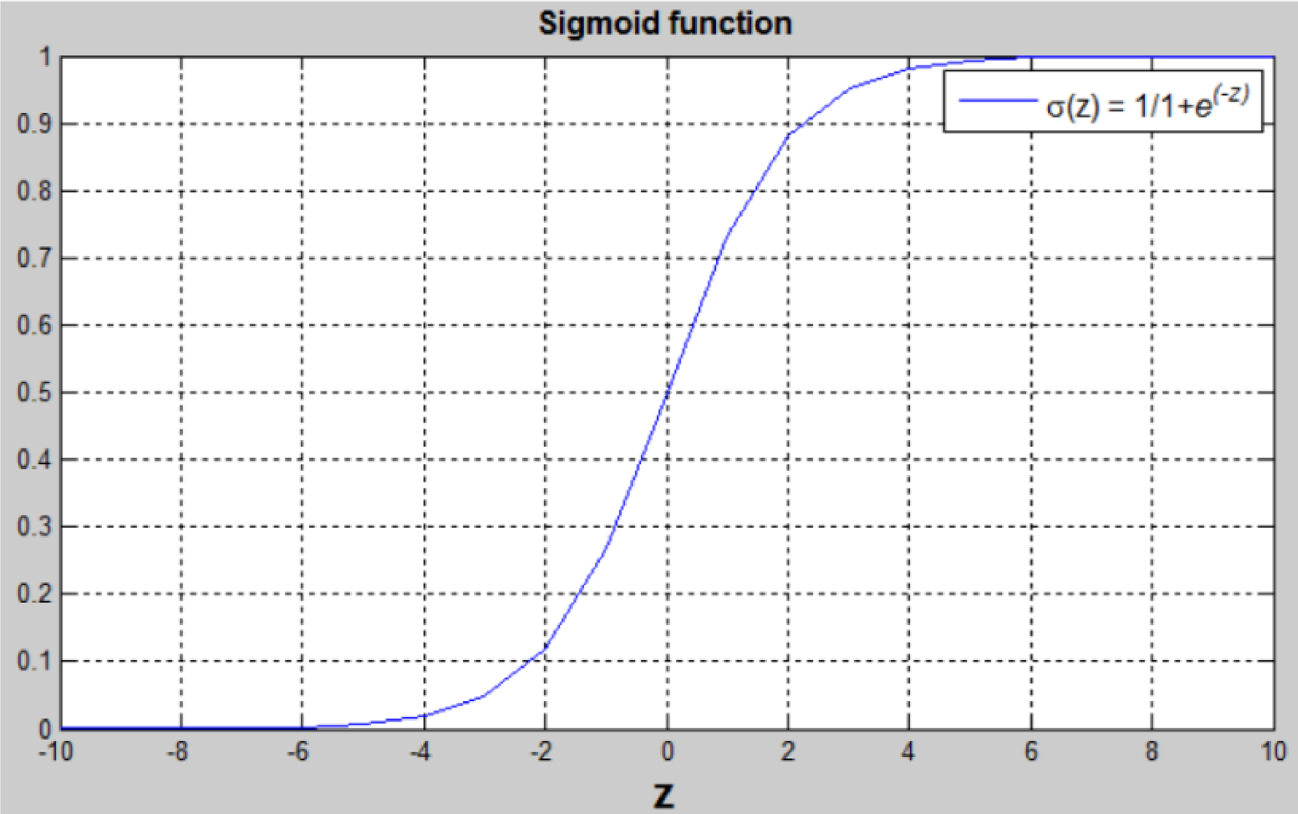
逻辑回归是一个用于监督学习问题的学习算法，输出值不是0就是1.

逻辑回归中用到的参数：

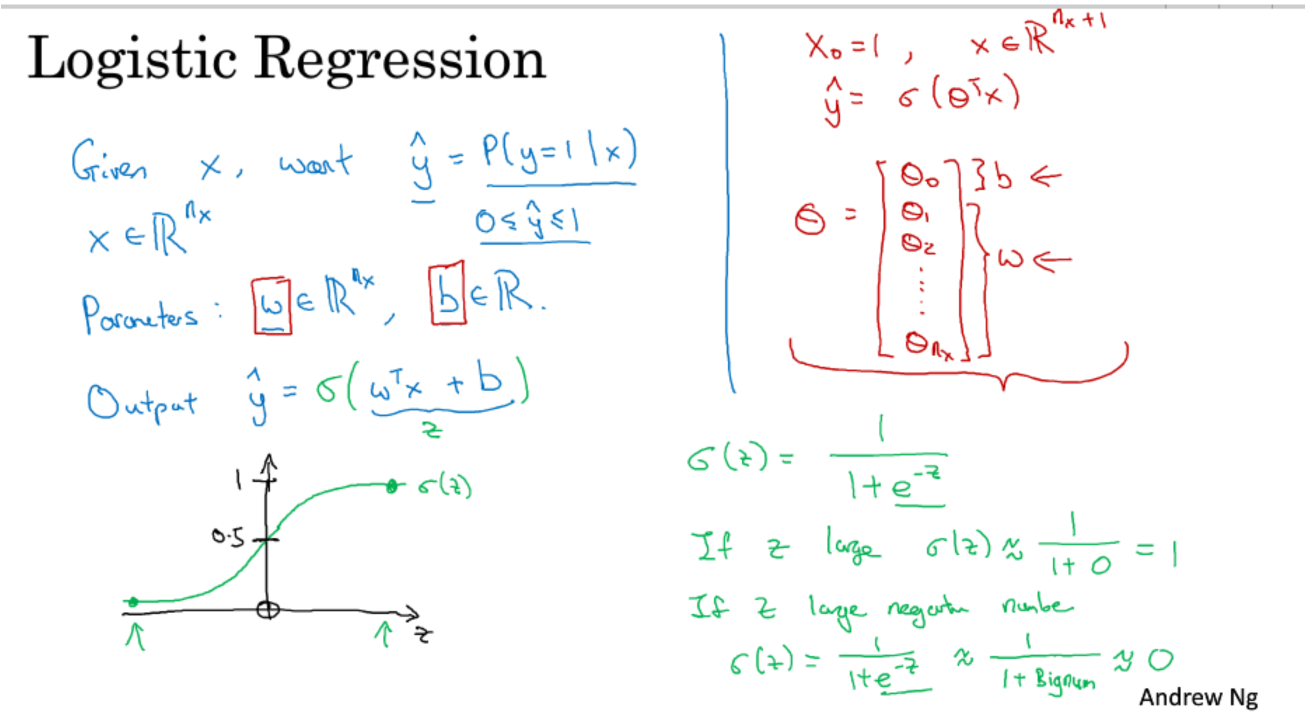
The parameters used in Logistic regression are:

- The input features vector: $x \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The training label: $y \in 0,1$
- The weights: $w \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The threshold: $b \in \mathbb{R}$
- The output: $\hat{y} = \sigma(w^T x + b)$
- Sigmoid function: s = $\sigma(w^T x + b)$ = $\sigma(z)$= $\frac{1}{1+e^{-z}}$

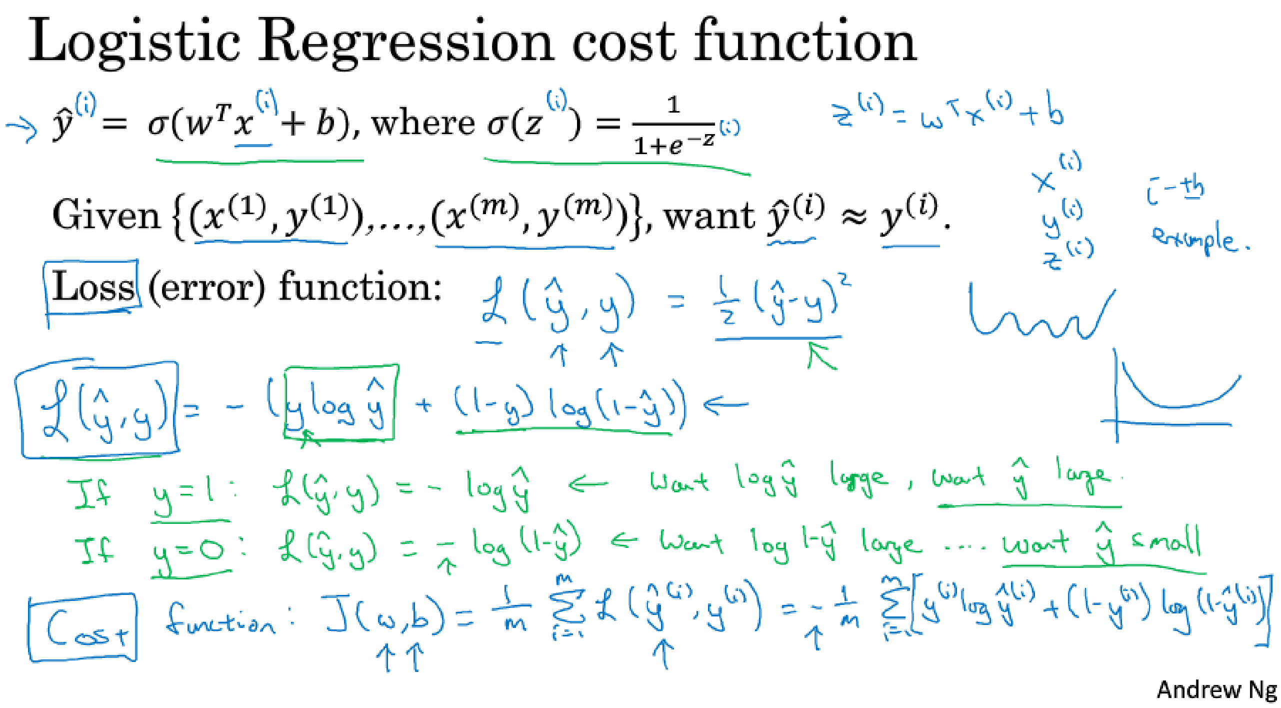
sigmoId函数的图像



- 如果z趋向于比较大的正数，那么sigmold等于1。
- 如果z趋向于比较大的负数，那么sigmold等于0。
- 如果z等于零，那么sigmold等于0.5。



逻辑回归代价函数



2.3 逻辑回归损失函数

损失函数的公式

Loss (error) function:

The loss function measures the discrepancy between the prediction ($\hat{y}^{(i)}$) and the desired output ($y^{(i)}$). In other words, the loss function computes the error for a single training example.

$$L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

$$L(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})$$

- If $y^{(i)} = 1$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\log(\hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 1
- If $y^{(i)} = 0$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)})$ where $\log(1 - \hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 0

但是，对于logistic regression 来说，一般不适用平方错误来作为Loss Function，这是因为上面的平方错误损失函数一般是非凸函数（non-convex），其在使用低度下降算法的时候，容易得到局部最优解，而不是全局最优解。因此要选择凸函数。

- 我们的目标是最小化样本点的损失Loss Function，损失函数是针对单个样本点的。

代价函数

Cost function

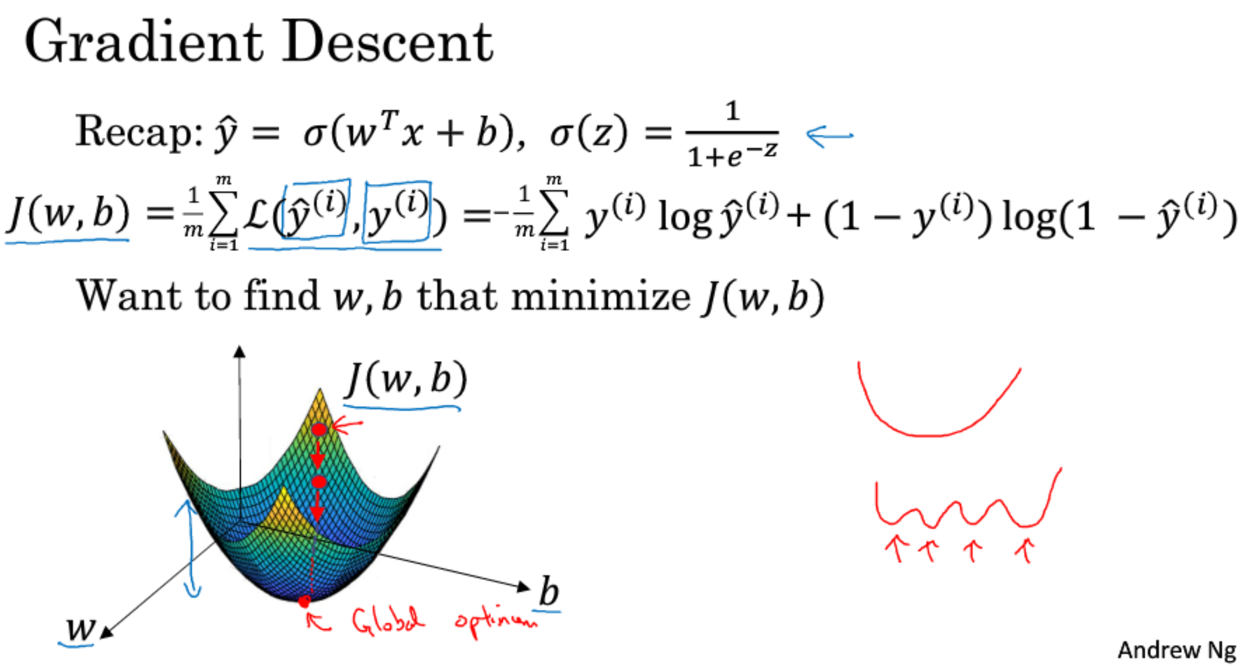
The cost function is the average of the loss function of the entire training set. We are going to find the parameters $w\ and\ b$ that minimize the overall cost function.

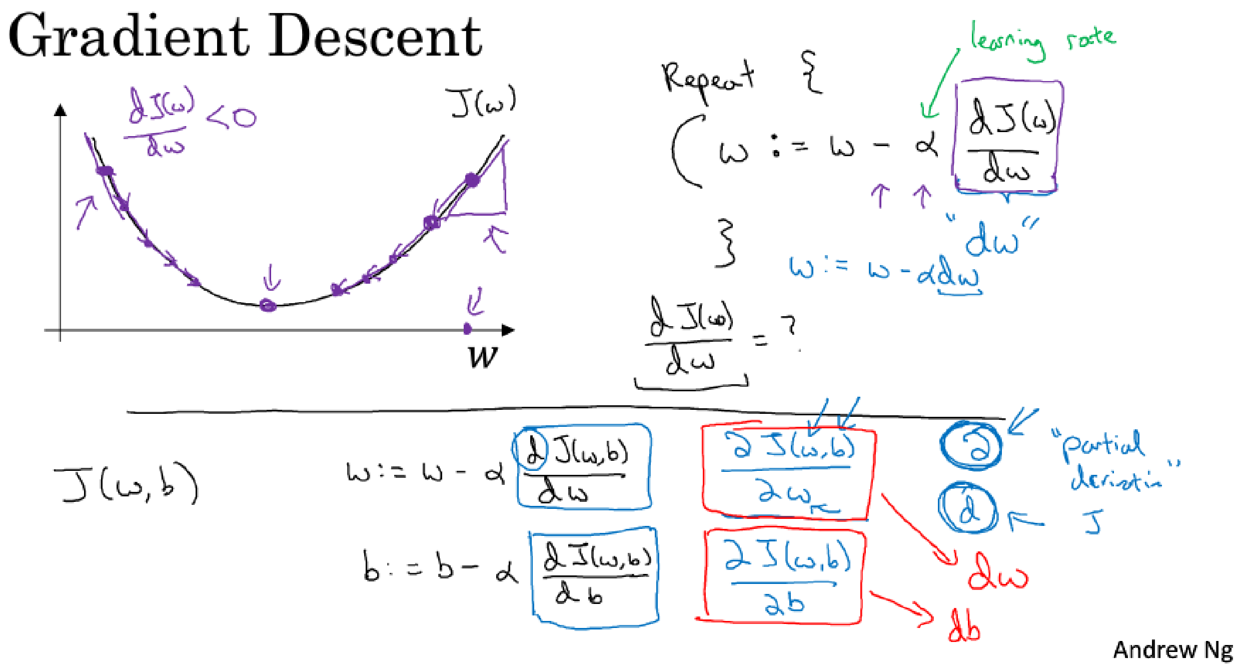
$$J(w,b)=\frac{1}{m}\sum_{i=1}^{m}L(\hat{y}^{(i)},y^{(i)})=-\frac{1}{m}\sum_{i=1}^{m}[(y^{(i)}\log(\hat{y}^{(i)})+(1-y^{(i)})\log(1-\hat{y}^{(i)})]$$

代价函数是损失函数的平均值。

2.4 梯度下降法

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。





每次迭代更新的修正表达式：

$$w := w - \alpha \frac{\partial J(w,b)}{\partial w}$$

$$b := b - \alpha \frac{\partial J(w,b)}{\partial b}$$

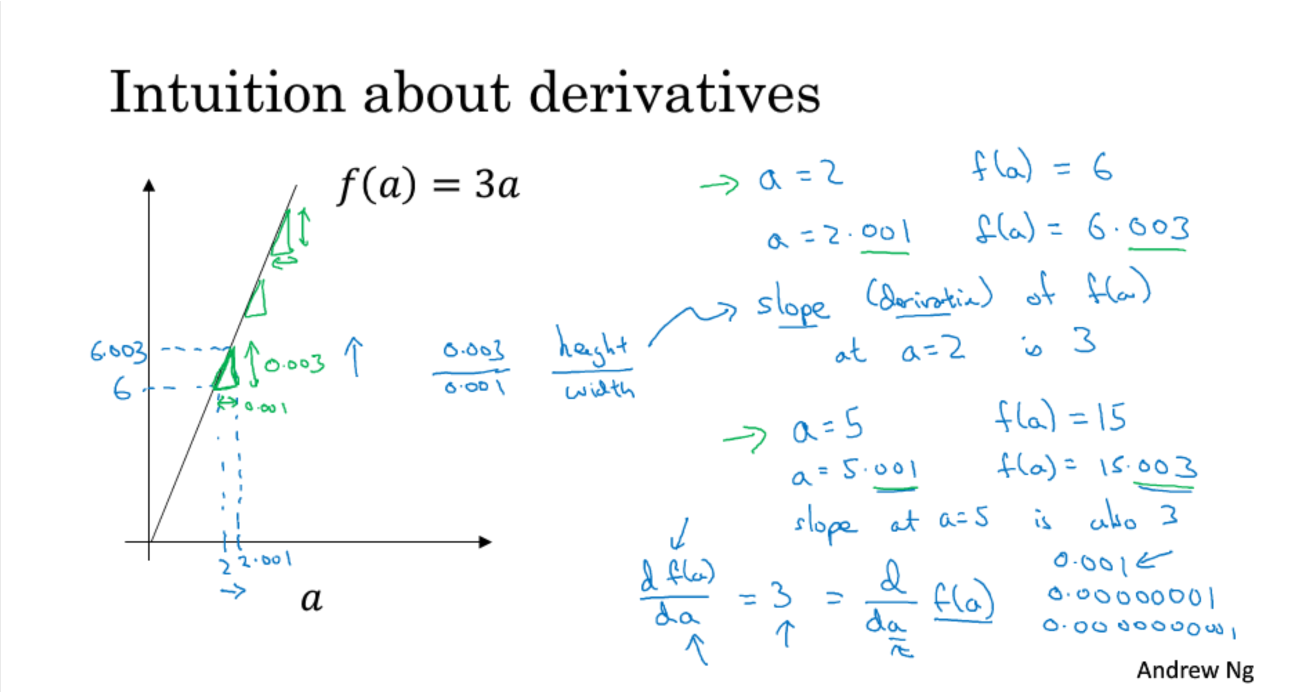
在程序代码中，我们通常使用

dw来表示$\frac{\partial J(w,b)}{\partial w}$，用db来表示$\frac{\partial J(w,b)}{\partial b}$。

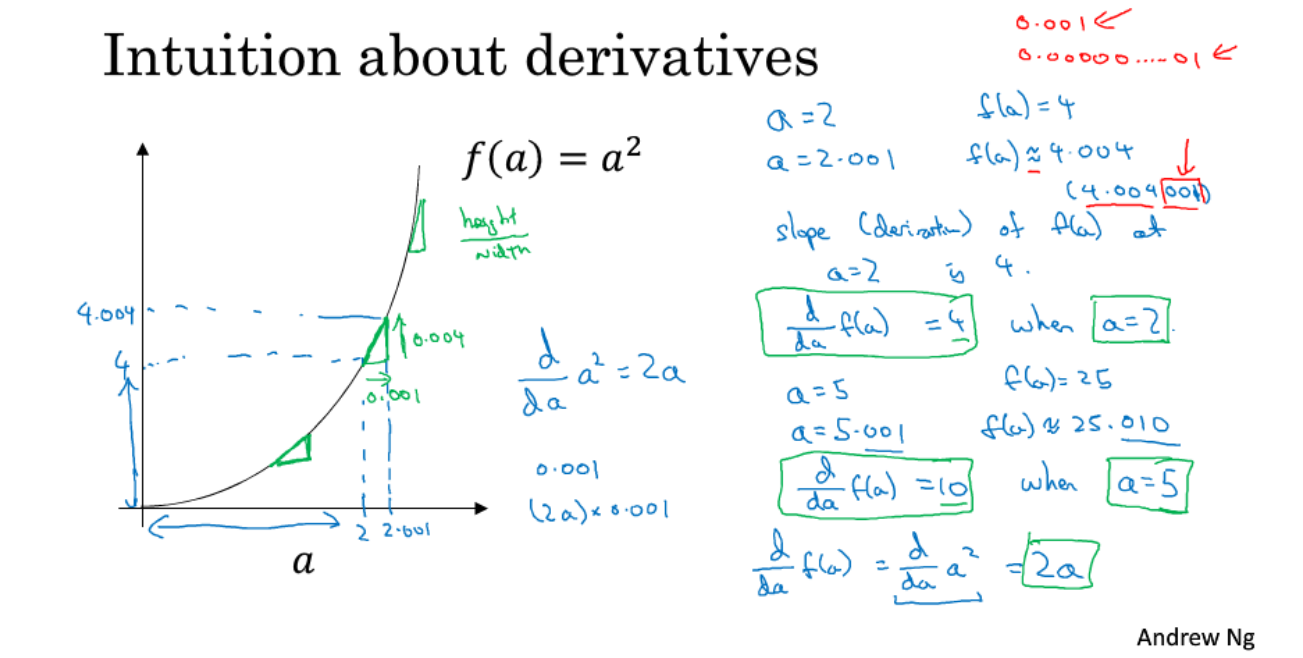
2.5 基本的神经网络编程--导数

理解导数

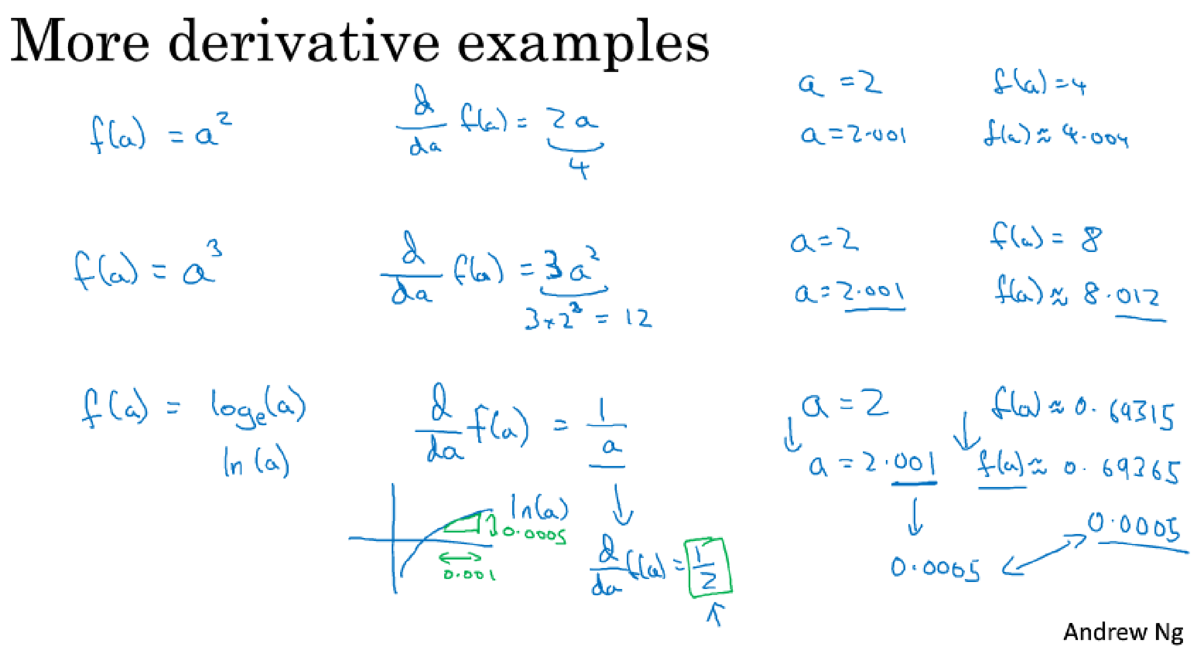
导数的直观理解就是经过某点的直线的斜率



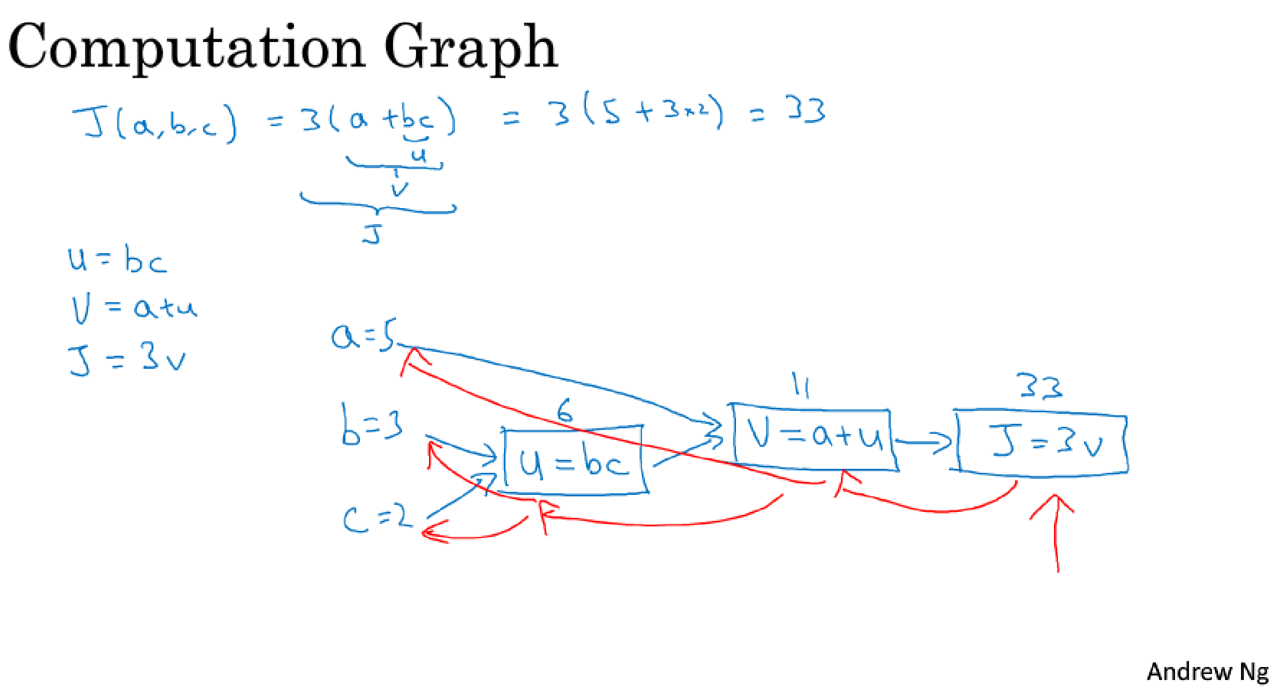
2.6 更过导数的例子



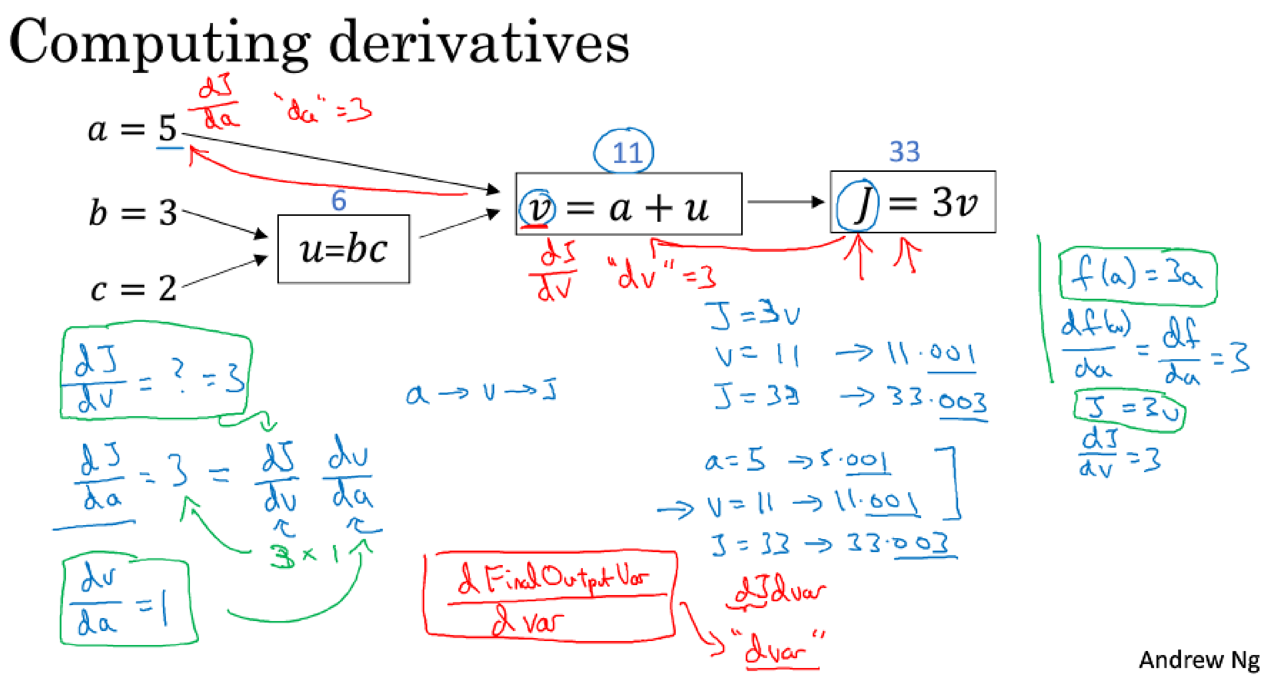
更多倒数的例子



2.7 计算图

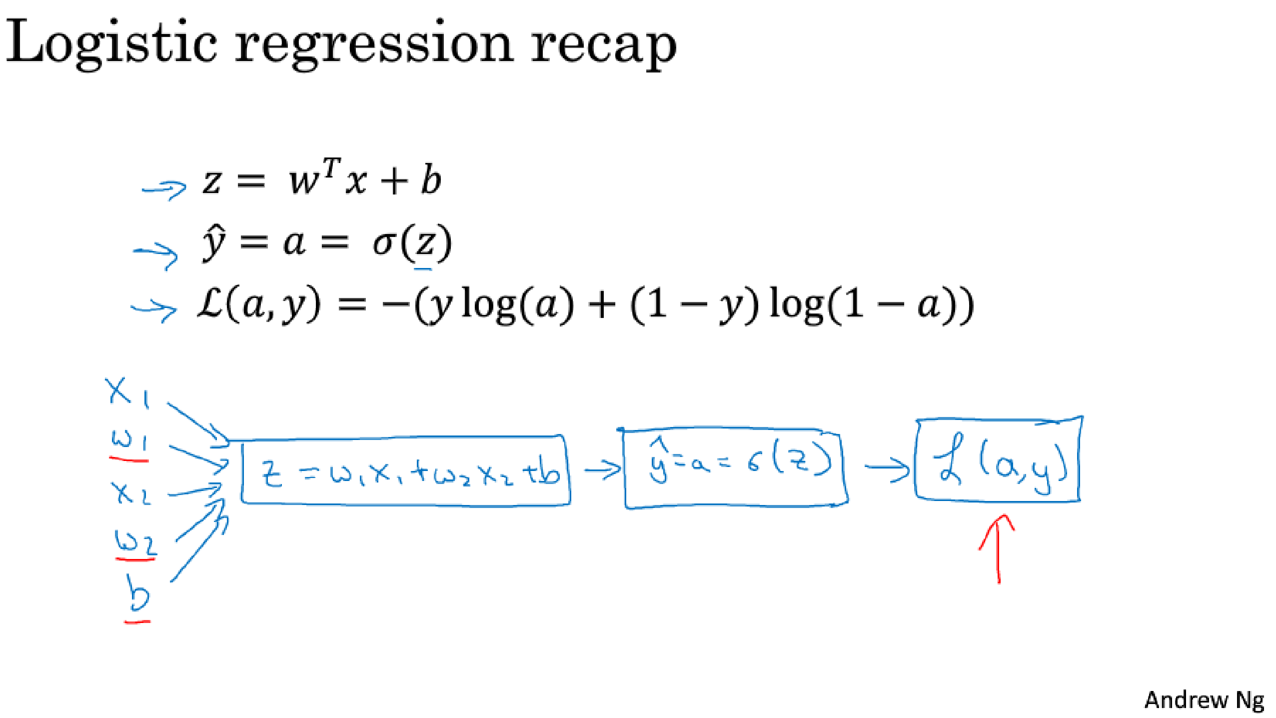


链式法则求导

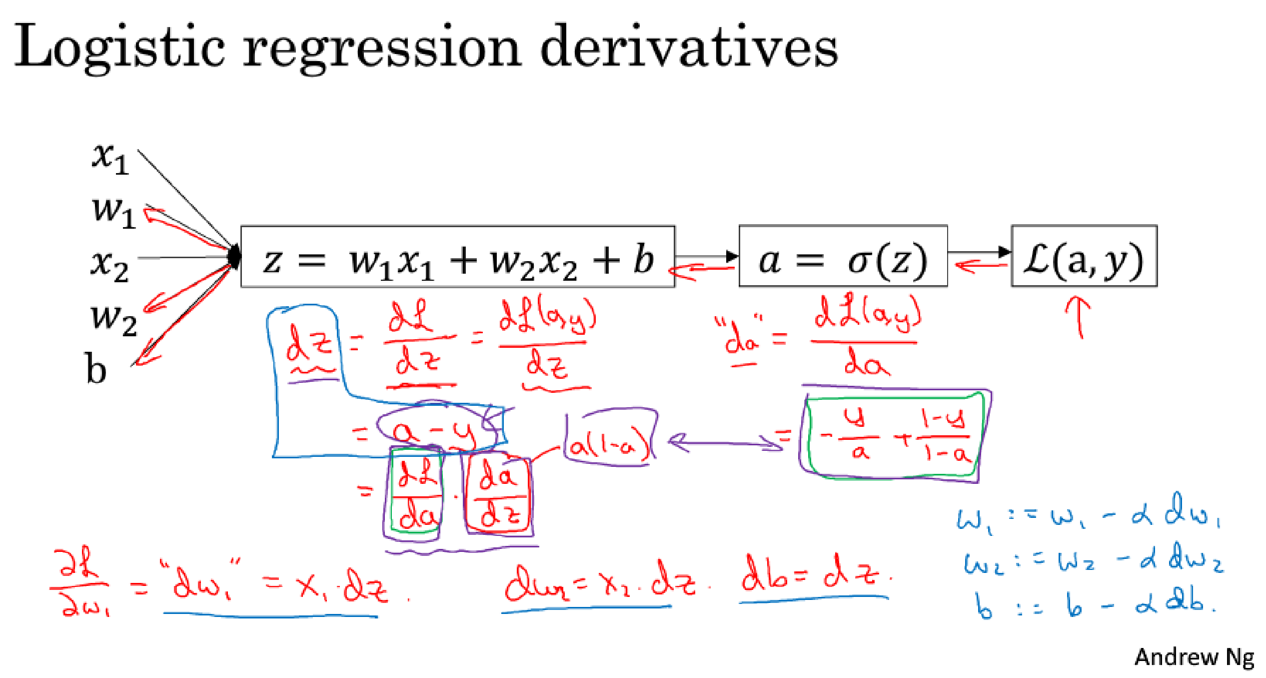


2.9 逻辑回归中的梯度下降法

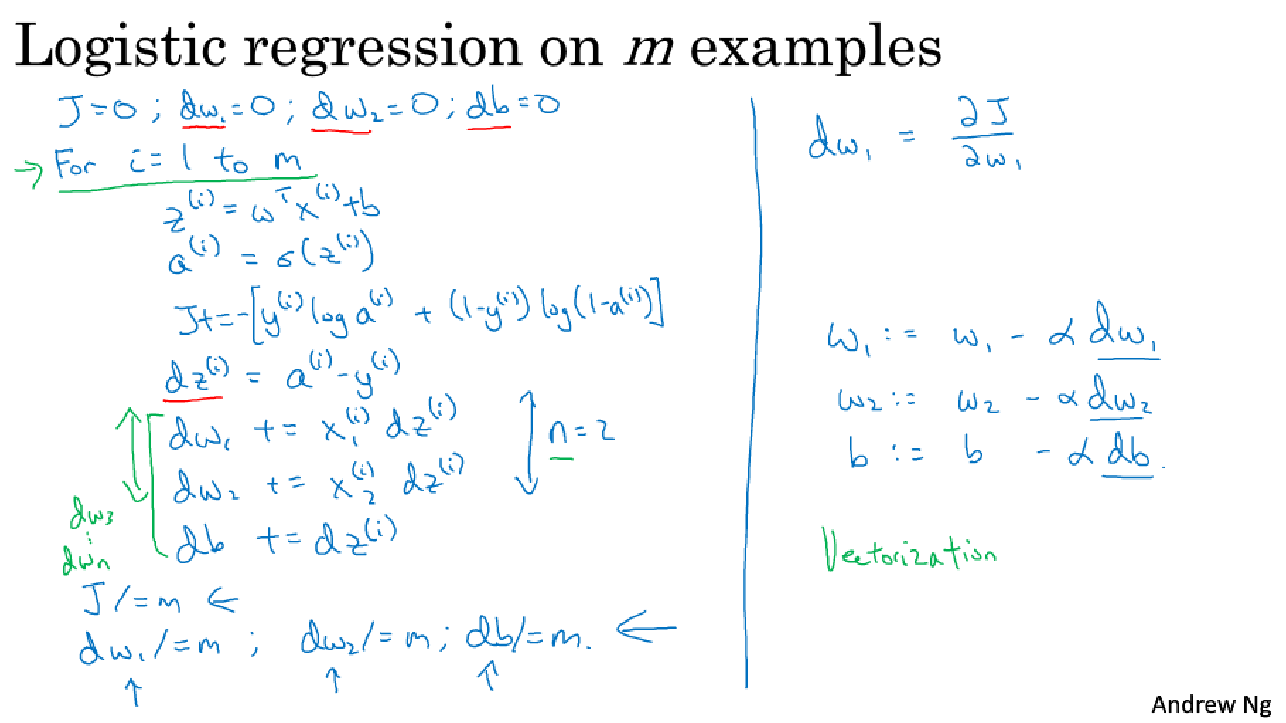
逻辑回归损失函数回顾



逻辑回归求导



2-10 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：

$$z^{(i)} = w^T x^{(i)} + b$$
$$\hat{y}^{(i)} = a^{(i)} = \sigma(z^{(i)})$$
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

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：

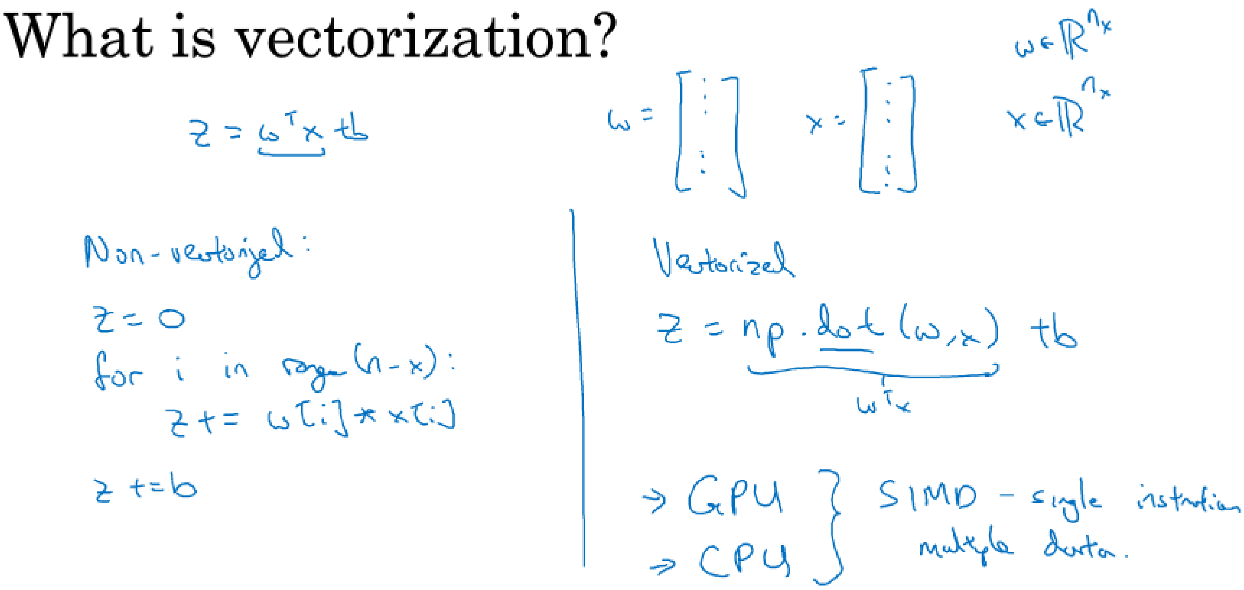
$$dw_1 = \frac{1}{m}\sum_{i=1}^{m} x_1^{(i)}(a^{(i)} - y^{(i)})$$

$$db = \frac{1}{m}\sum_{i=1}^{m}(a^{(i)} - y^{(i)})$$

2.14 向量化logistics回归的梯度输出

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。

什么是向量化？



逻辑回归向量化

输入矩阵X：（n_x, m）

权重矩阵w：（$n_x, 1$）

偏置b：为一个常数

输出矩阵Y：（$1, m$）

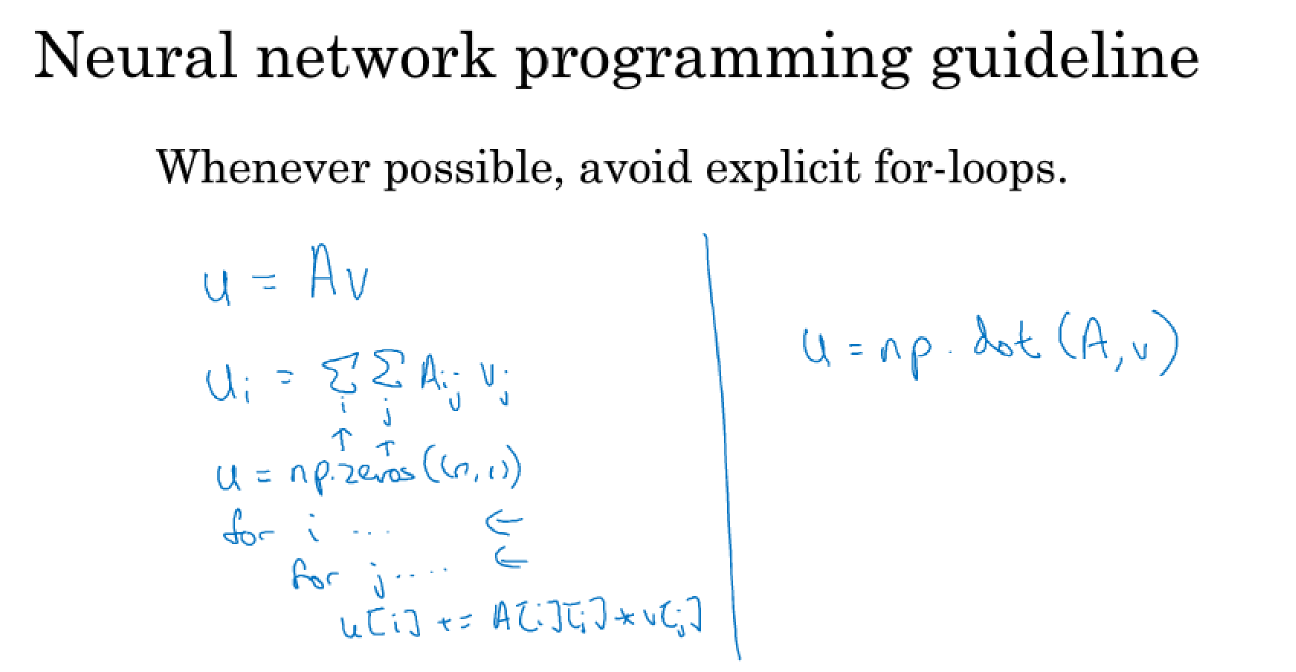
所有m个样本的线性输出Z可以用矩阵表示：

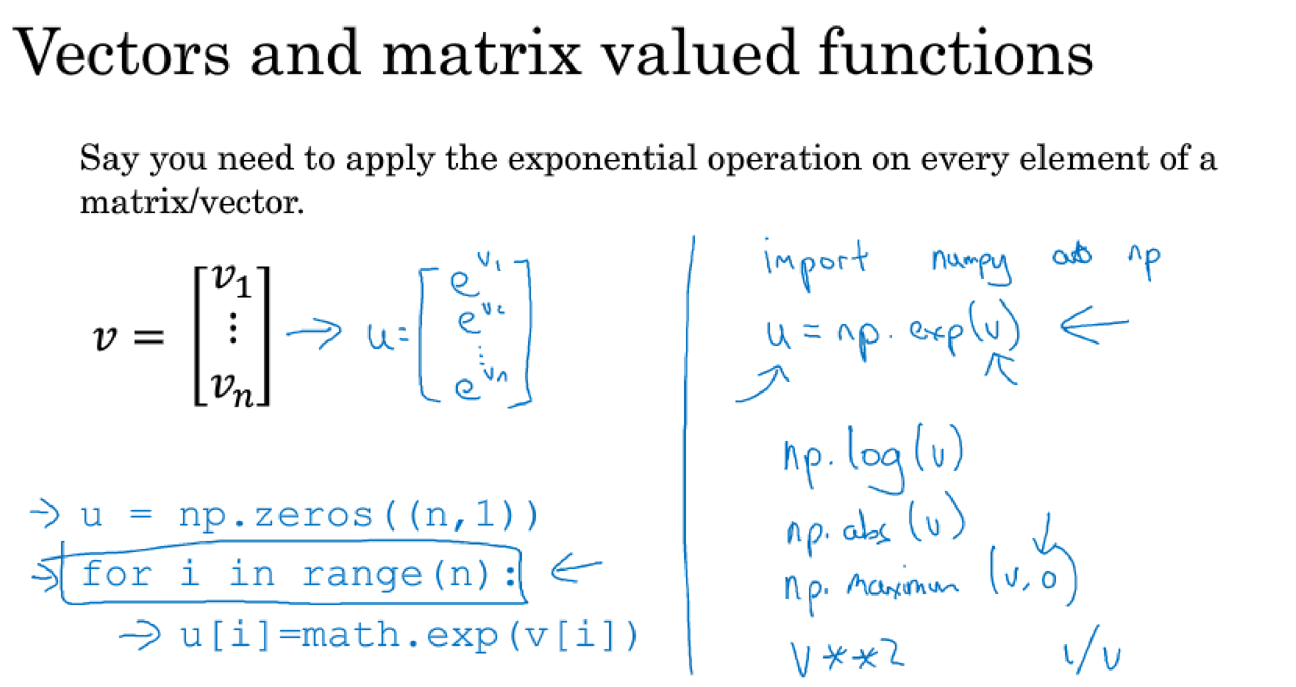
$$Z = w^T X + b$$

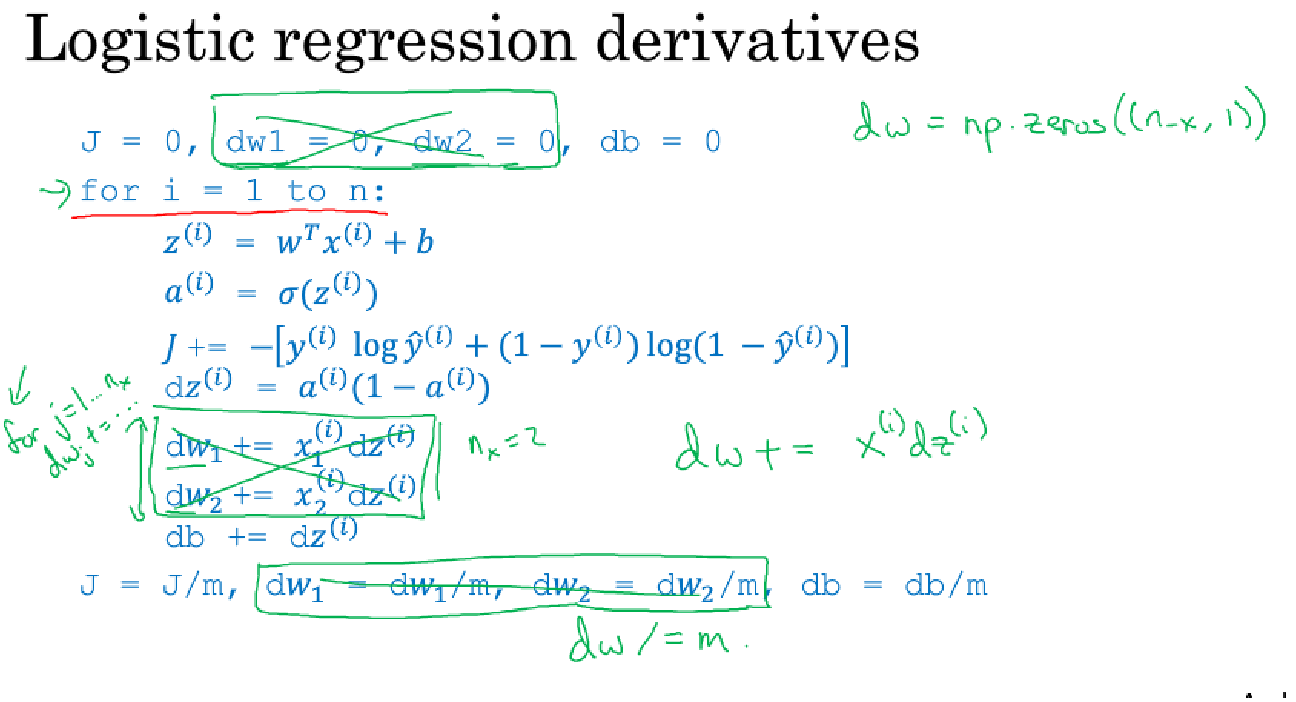
python 代码

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

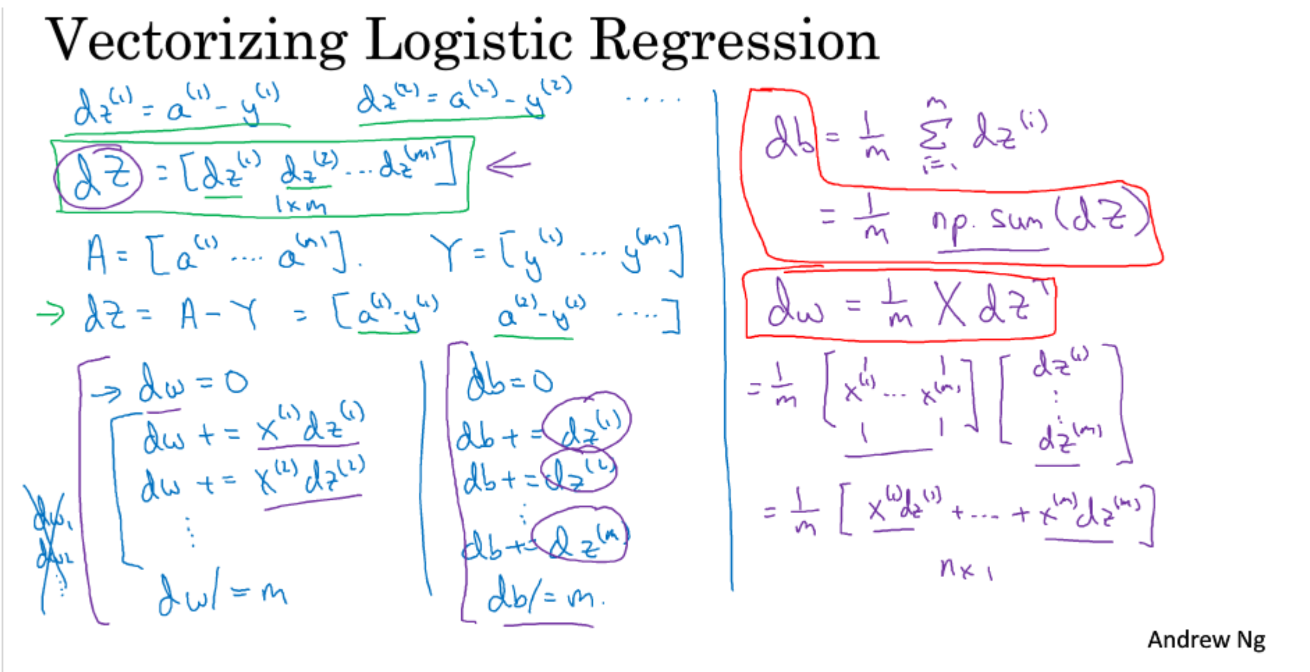
2.12 更多的向量化的例子--神经网络

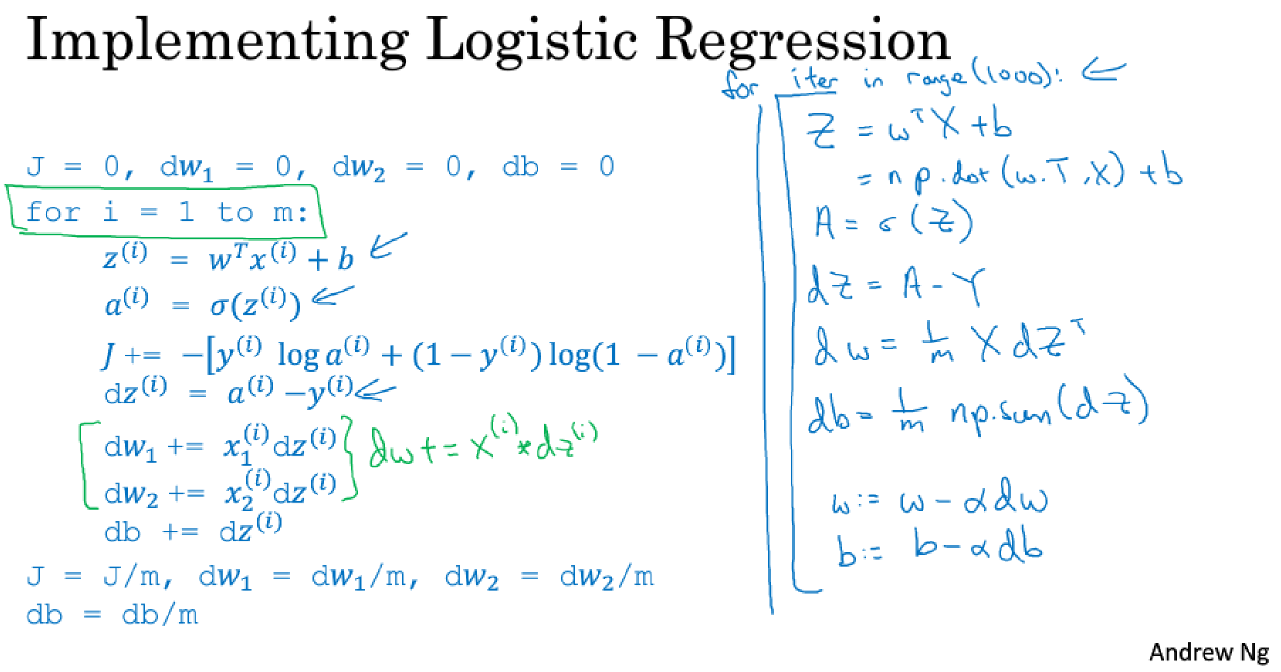






2.13 向量化逻辑回归的梯度计算






```
Z = np.dot(w,T,X) + b
A = sigmold(Z)
dZ = A-Y
dw = 1/m*np.dot(X,dZ.T)
db = i/m*np.sum(dZ)

w = w - alpha*dw
b = b - alpha*db
```

2.15 python 中的广播

Broadcasting example

Calories from Carbs, Proteins, Fats in 100g of different foods:

$$\begin{array}{l} \\ \text{Carb} \\ \text{Protein} \\ \text{Fat} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \end{array} \\ \begin{bmatrix} 56.0 & 0.0 & 4.4 & 68.0 \\ 1.2 & 104.0 & 52.0 & 8.0 \\ 1.8 & 135.0 & 99.0 & 0.9 \end{bmatrix} \end{array} = A \quad (3,4)$$

0

1

59 cal $\quad \frac{56}{59} \approx 94.9\%$

Calculate % of calories from Carb, Protein, Fat. Can you do this without explicit for-loop?

```
cal = A.sum(axis = 0)
percentage = 100*A/(cal.reshape(1,4))
```

$(3,4) \ / \ (1,4)$

Broadcasting example

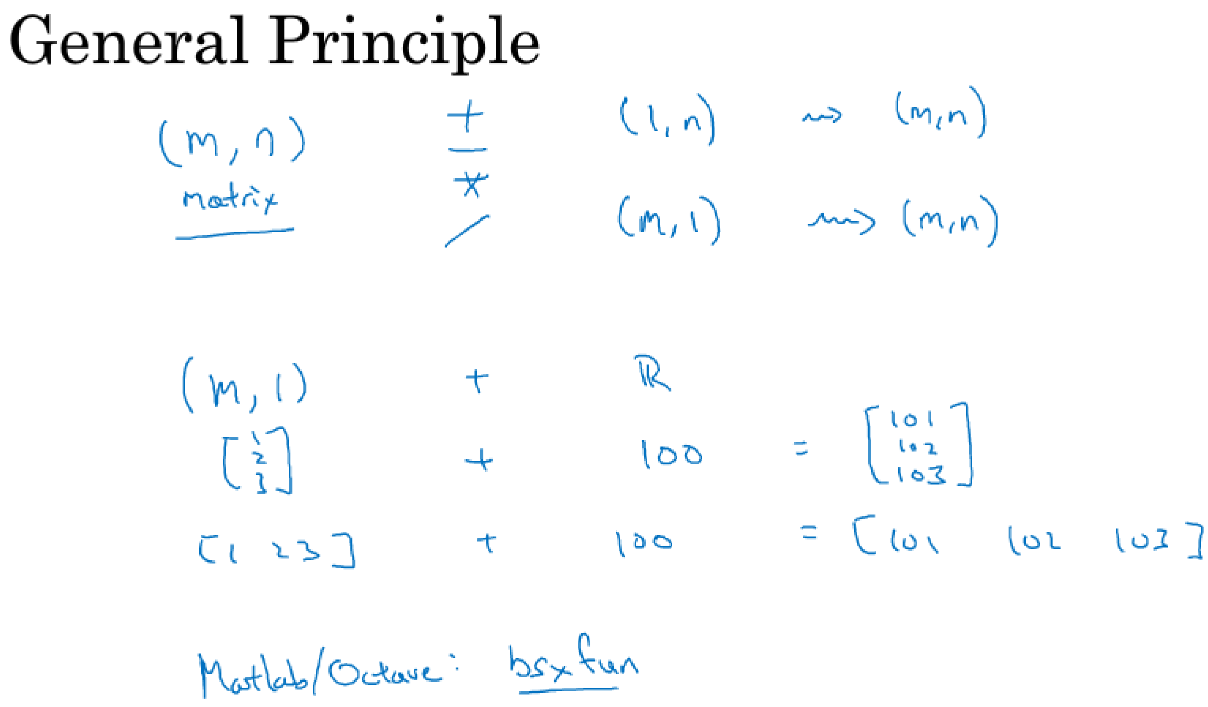
$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100}$$

$$\underset{(m,n)\ \ (2,3)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \begin{bmatrix}100 & 200 & 300\\ 100 & 200 & 300\end{bmatrix}$$

$(1,n) \rightsquigarrow (m,n) \quad (2,3)$

$$\underset{(m,n)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \begin{bmatrix}100 & 100 & 100\\200 & 200 & 200\end{bmatrix} =$$

$(m,1) \to (m,n)$



2.16 python numpy 向量的说明

- 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。

Python / numpy vectors

```
import numpy as np

a = np.random.randn(5)

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

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

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

- 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))
```