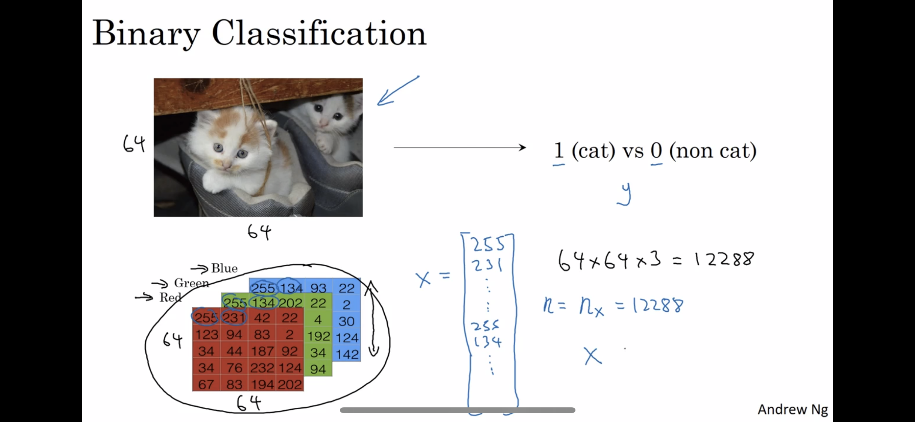
Logstic

1. **读取图片**

三通道彩图 64\*64\*3

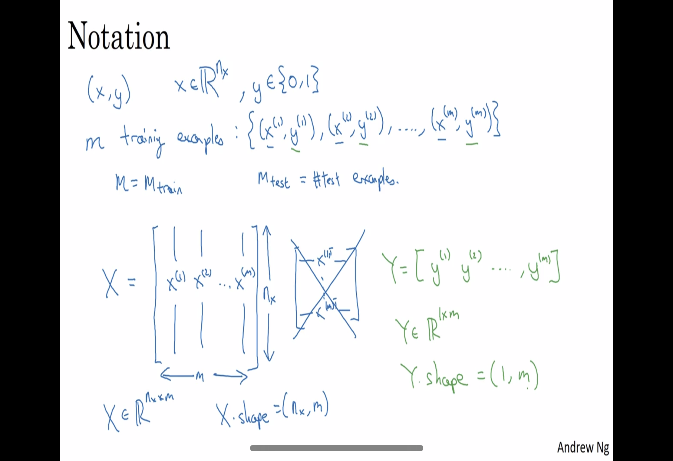
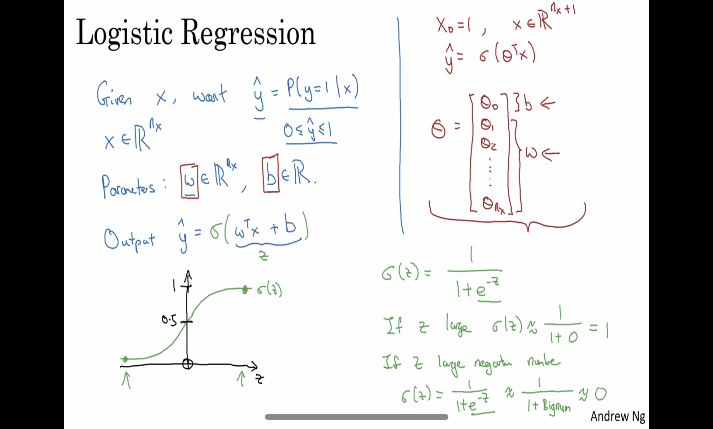
x1=[255,231,42,22……]

x1.shape=(12288,)

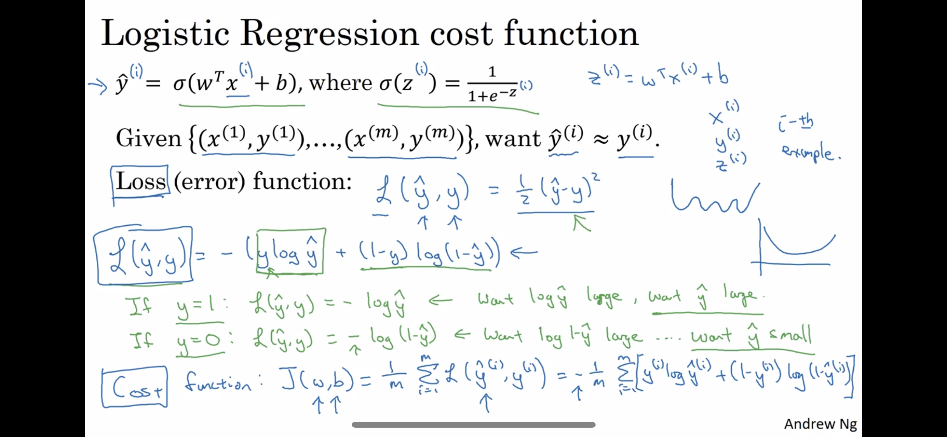
X=[x1,x2…xn]

X.shape=(m,Nx)

**2. define notation**



1. **定义损失函数和代价函数**

Input: x

Output: y’=σ(wTx+b)

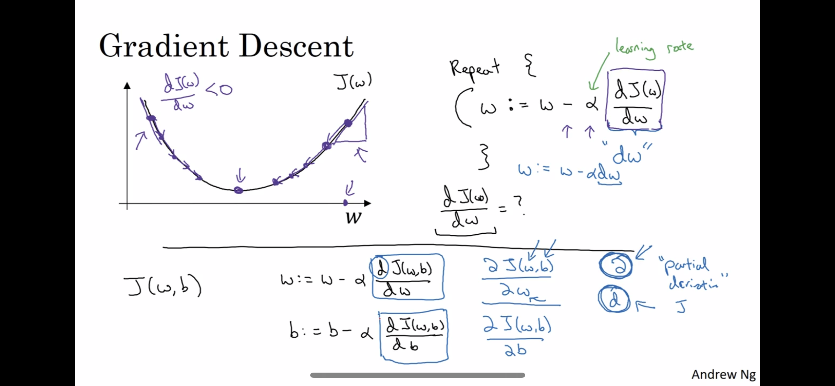
损失函数Loss：

L=-(ylogy’+(1-y)log(1-y’))

代价函数Cost：

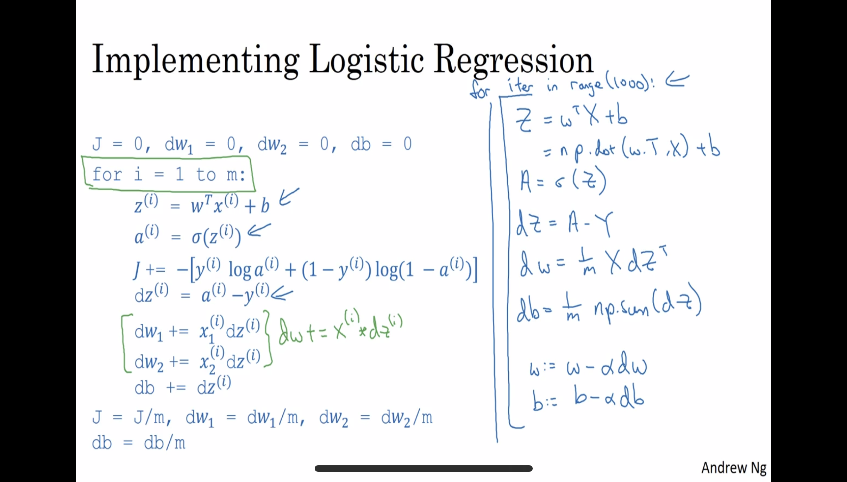
J=1/mΣL

注：为什么不用梯度下降作为损失函数，因为梯度下降会形成一个非凸函数。

1. **实现梯度下降**

Repeat: w-dw

b-db

1. **向量化**
2. 输入X∈(m[样本数]\*nx[特征数])
3. 输入Y∈(m[样本数],)
4. 初始化变量w.zeros∈(m[样本数],)
5. 初始化变量b=0
6. 初始化J,dw1,dw2,db=0
7. 定义sigmoid函数

789为正向传播

1. z(i) = w.T\*x(i)+b
2. a(i) = sigmoid(z(i))
3. J += -[y(i)loga(i)+(1-y(i))log(1-a(i))]

10为反向传播通过dw,db求更新w和b

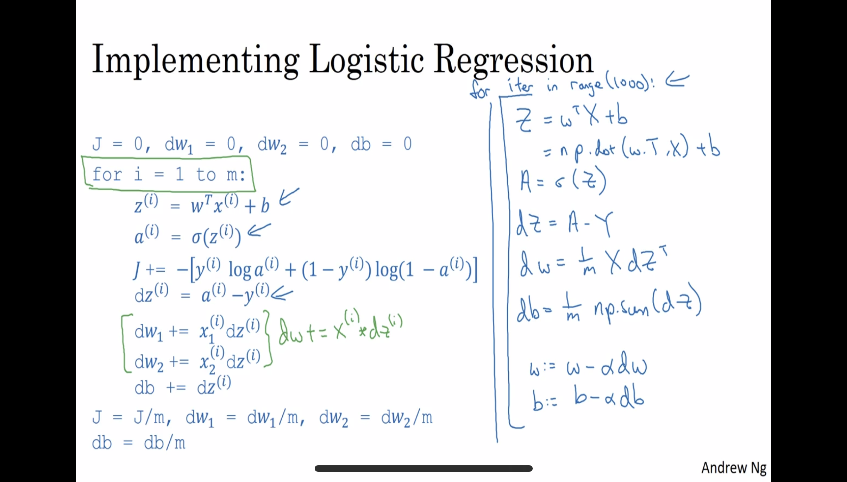
10) dz(i)=a(i)-y(i)

dw1+=x1(i)dz(i) dw2+=x2(i)dz(i)

db+=dz(i)

11）J=J/m , dw1=dw1/m , dw2=dw2/m , db=db/m

12）更新w,b 完成一轮迭代

**向量化**

1） 输入X∈(m[样本数]\*nx[特征数])

2） 输入Y∈(m[样本数],)

3） 初始化变量w.zeros∈(m[样本数],)

4） 初始化变量b=0

5） 初始化J,dw1,dw2,db=0

6） 定义sigmoid函数

78正向

7） 定义Z=w.T\*x+b=np.dot(w.T,x)+b

8) 定义A=σ（Z）

9-11反向

9）dZ=A-Y

10）dw=1/mX\*dZ.T

11）db=1/mnp.sun(dZ)

12) w:=w-αdw ；b:=b-αdb