$y = w^T x + b$ $\Rightarrow$ $\hat{y} = \sigma(y) = \sigma(w^T x + b)$.

↑ 参数

损失函数: $L(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$.

(理解含义). $y=1 \Rightarrow \hat{y} \approx 1$

$y=0 \Rightarrow \hat{y} \approx 0$

代价函数 $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

梯度下降: α: 学习率; $\frac{dJ(w)}{dw}$, 即代价函数关于 w 的求导.

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

由学习率优化参数 w, b, 求函数 $J(w,b)$ 中 w/b 的偏导.

(自动计算) $L(a,y) \Rightarrow$ 计算 $da = \frac{dL(a,y)}{da} = -y/a + (1-y)/(1-a)$

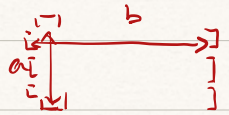
进一步自动计算, $dz = \frac{dL(a,y)}{dz} = \frac{dL}{da}\cdot\frac{da}{dz} = a - y$.

Vectorization: $z = w^T x + b$

$w = \begin{bmatrix} \\ \\ \end{bmatrix}$ $x = \begin{bmatrix} \\ \\ \end{bmatrix}$

python code: `c = np.dot(w, x)` $\Rightarrow$ 点乘 w, x.

avoid explicit for loop as much as possible!

`np.zeros(a,b)`: 

a axb matrix filled with 0.

`np.exp(a)`: e^a.

$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$ $\Rightarrow$ dw 向量化, `w = np.zeros((n(x), 1))`

$\Rightarrow dw += x^{(i)} dz$

Vectorization Logical Regression:

$Z = [z^{(1)} z^{(2)} \cdots z^{(i)}] = w^T X + [b \cdots b] = [w^T x^{(1)} + b,\ w^T x^{(2)} + b \cdots w^T x^{(i)} + b]$

python: `Z = np.dot(w.T, x) + b` ← adding b here called "broadcasting"

$A = [a^1, a^2 \cdots a^i] = \sigma(Z)$ <= forward propagation

Vectorizing Logistic Regression's Gradient:

$dZ = [dz^1, dz^2, \cdots dz^m]$, $A = [a^1, a^2 \cdots a^m]$, $Y = [y^1, y^2 \cdots y^m]$

$dZ = A - Y$ (see previous notes)

$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)} = \frac{1}{m} \cdot np.sum(dZ)$

$dw = \frac{1}{m} \cdot X \cdot dZ^T$

Broadcasting in Python