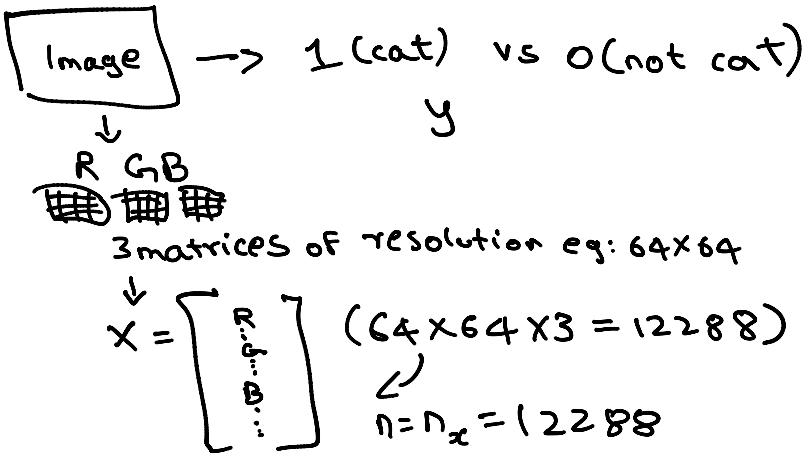
Week 2a - Logistic regression as a Neural Network

Thursday, July 30, 2020 10:23 PM

① Binary classification



→ Notation

(x, y) single training set represented by where $x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$

m training examples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

$m = m_{train}$ ↓ training set $\quad m_{test}$ ↓ test set

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad n_x \text{ (rows)}, \quad m \text{ (columns)}$$

$X \in \mathbb{R}^{n_x \times m}$ $\quad X.shape = (n_x, m)$

$Y = [y^{(1)}\ y^{(2)} \ldots y^{(m)}]$

$Y \in \mathbb{R}^{1 \times m}$

$Y.shape = (1, m)$

Stacking as colums is easier to implement than rows

② Logistic Regression

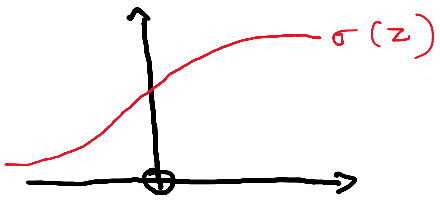
Given x, we want $\hat{y} = P(y=1 \mid x)$

$x \in \mathbb{R}^{n_x}$ — $\hat{y}$ is the probability that y is 1 when x is given

Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

Output $\hat{y} = w^T x + b$ (in case of linear regression)

$\therefore \hat{y} = \sigma(w^T x + b)$ [sigmoid of $w^T x + b$] (Logistic regression)



$\sigma(z) = \frac{1}{1+e^{-z}}$

If z is large, $\sigma(z) \approx 1$

If z is small, $\sigma(z) \approx 0$

Alt. convention $x_0 = 1$, $x \in \mathbb{R}^{n_x+1}$

$\hat{y} = \sigma(\theta^T x)$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{matrix} \rightarrow b \\ \} w \end{matrix}$

③ Logistic Regression cost Function

→ Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$

$z^{(i)}$ can also be defined → $z^{(i)} = w^T x^{(i)} + b$

→ Loss (error) function: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

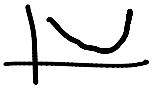
↳ mean squared error loss fn.

↓ results in non-convex graph

~~~ → multiple minima for optimization pblms

→ $\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$

↳ If $y=1$, $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$ ↳ as


~~~

- If $y=1$, $\mathcal{L}(\hat{y},y) = -\log \hat{y}$
  - as small as possible $\Rightarrow \log \hat{y}$ is large
  - $\Rightarrow \hat{y}$ is large
  - $\Rightarrow \hat{y} = 1$ (max. of $\hat{y}$)
- If $y=0$, $\mathcal{L}(\hat{y},y) = -\log(1-\hat{y})$
  - as small as possible $\Rightarrow \log(1-\hat{y})$ large
  - $\Rightarrow 1-\hat{y}$ is large
  - $\Rightarrow \hat{y}$ is small
  - $\Rightarrow \hat{y} = 0$ (min of $\hat{y}$)

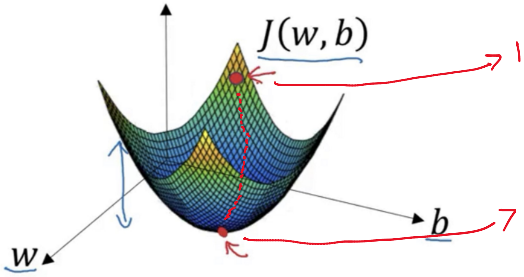
-> Cost function: $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

↳ average loss across all sets

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

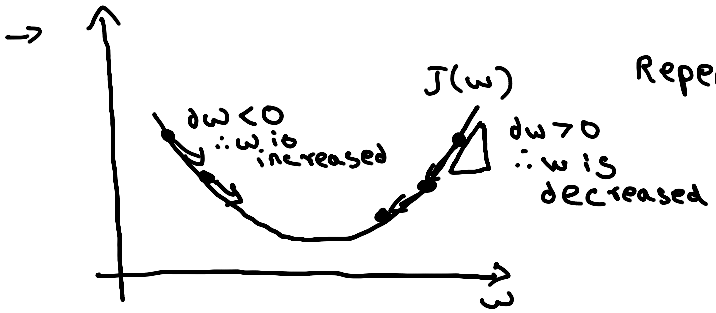
④ Gradient Descent

-> want to find w, b that minimize (w,b)



Initialisation

Global optimum



Repeat {

update symbol

Learning rate

$w := w - \alpha \frac{dJ(w)}{dw}$

"dw"

$w := w - \alpha\, dw$

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

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

⑤ Computation graph

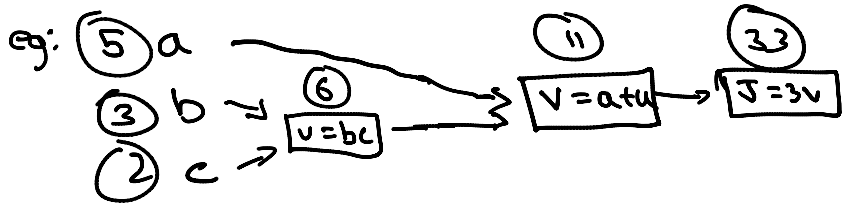
Let's say we need to compute

$$J(a,b,c) = 3(a+bc)$$

Steps:

i. Compute $u = bc$
ii. Compute $v = a+u$
iii. Compute $J = 3v$

eg:



⑥ Derivatives with a Computation graph

$\frac{dJ}{dv} = ?$

$J = 3v$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

$J = 3v$
$\frac{dJ}{dv} = 3$

$\frac{dJ}{da} = ?$

$\frac{dJ}{da} = \frac{dJ}{dv} \cdot \frac{dv}{da}$

$a = 5 \rightarrow 5.001$
$v = 11 \rightarrow 11.001$ $(dv/da = 1)$
$J = 33 \rightarrow 33.003$

$\frac{dJ}{da} = 3$

$\frac{dJ}{du}$

$\frac{dJ}{du} = \frac{dJ}{dv} \cdot \frac{dv}{du}$

$u = 6 \rightarrow 6.001$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

$\frac{dJ}{db}$

$\frac{dJ}{db} = \frac{dJ}{du} \cdot \frac{du}{db}$

$b = 3 \rightarrow 3.001$
$u = bc = 6 \rightarrow 6.002$ $(\because c = 2)$

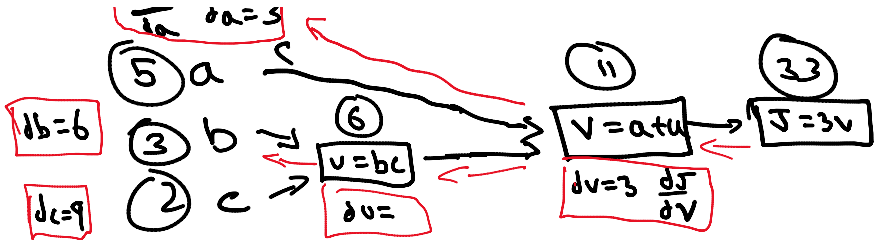
$\frac{du}{db} = 2 \Rightarrow \frac{dJ}{db} = 3 \cdot 2 = 6$

$\frac{dJ}{dc}$

$\frac{dJ}{dc} = 9$

$\frac{d(\text{Final Output var})}{d(\text{var})}$ → represented as "dvar" in code

$\frac{dJ}{da}\, da = 3$



⑦ Logistic regression gradient descent

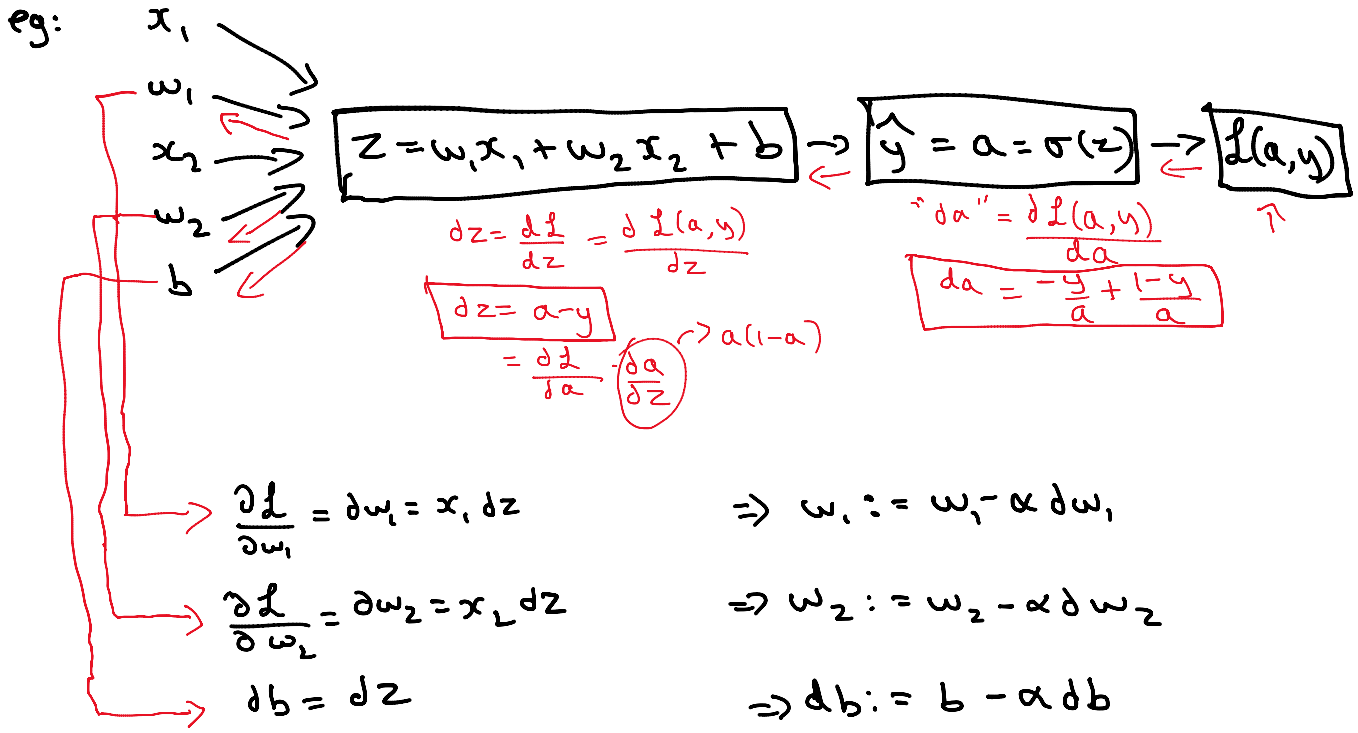
(For 1 training example)

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

$$\hat{y} = a = \sigma(z)$$

$$\mathcal{L}(a,y) = -(y \log a + (1-y)\log(1-a))$$

eg:



$$\frac{\partial \mathcal{L}}{\partial w_1} = dw_1 = x_1 dz \quad \Rightarrow w_1 := w_1 - \alpha\, dw_1$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = dw_2 = x_2 dz \quad \Rightarrow w_2 := w_2 - \alpha\, dw_2$$

$$db = dz \quad \Rightarrow db := b - \alpha\, db$$

⑧ Gradient descent on 'm' examples

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(a^{(i)}, y)$$

$$a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

$(x^{(i)}, y^{(i)})$

$dw_1^{(i)}\ dw_2^{(i)}, db^{(i)}$

$n = 2$

$$\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1} \mathcal{L}(a^{(i)}, y^{(i)})}_{dw_1^{(i)} \text{ — } (x^{(i)}, y^{(i)})}$$

- Algorithm

→ Initialise $J = 0;\ dw_1 = 0;\ dw_2 = 0;\ db = 0$

(dw_1, dw_2: accumulated $dw_1^{(i)}$ & $dw_2^{(i)}$)

→ For $i = 1$ to m

→ for i=1 to m (accumulated $dw_1^{(i)}$ & $dw_2^{(i)}$)

$$z^{(i)} = w^T x^{(i)} + b$$
$$a^{(i)} = \sigma(z^{(i)})$$
$$J+= -\left[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)}) \right]$$
$$dz^{(i)} = a^{(i)} - y^{(i)}$$
$$dw_1 += x_1^{(i)} dz^{(i)}$$
$$dw_2 += x_2^{(i)} dz^{(i)}$$
$$db += dz^{(i)}$$

↑↓ assuming n = 2 for this eg. else has to be looped n times

→ J/m;
dw1/m;
dw2/m;
db/m;

→ $w_1 := w_1 - \alpha\, dw_1$
$w_2 := w_2 - \alpha\, dw_2$
$b := b - \alpha\, db$

- vectorization can be done to eliminate multiple loops.