PAGE NO.
DATE: / /

I Binary Classification

* Image is processed in the form of matrices for 3 colours red, green and blue.

* Notations:

m - no. of training vectors
n_x - size of input vector
n_y - size of output vector
$x(1)$ - size of input vector
$y(1)$ - size of output vector

$x \in R^{n(x)}$

$y \in (0,1)$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$
$$\longleftarrow m \longrightarrow$$

$X \in R^{n_x \times m}$

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

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

PAGE NO.
DATE: / /

II Logistic Regression -

* Algorithm used to classify algorithm of 2 classes

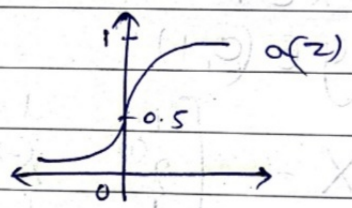
* $x \in R^{n_x}$

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

* $y = wx + b$

* If x is a vector,
$y = w(\text{transpose})x + b$
$= w^T x + b$
between (0 & 1)

* $\hat{y} = \text{output} = \sigma(\underbrace{w^T x + b}_{z})$



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

III) Logistic Regression Cost Function

loss function - calculates error for single training example
cost function - average of loss function over entire training set

$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$
$y = 1, \quad L = -\log \hat{y} \quad \therefore \hat{y}$ should be large
not possible as $\hat{y}$ (0,1)

$y = 0, \quad L = -\log(1-\hat{y}) \therefore \log(1-\hat{y}) \Rightarrow$ large
$\therefore \hat{y} \Rightarrow$ small

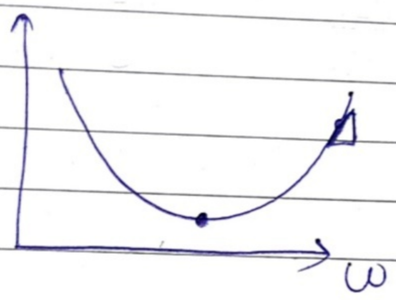
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{-1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right)$$

(IV) Gradient Descent

* Goal: To find ω, b that minimise $J(\omega, b)$

* Our cost function is convex & we initialise ω & b to 0 (0,0) ← preferable and move steeply downwards

*



$$\omega = \omega - \alpha \frac{dJ(\omega)}{d\omega} \leftarrow \text{'}d\omega\text{'}$$

learning rate

$\omega = \omega - \alpha d\omega$ — measure of how much we slope towards ω

$$\omega = \omega - \alpha \boxed{\frac{dJ(\omega,b)}{d\omega}} = \omega - \alpha \frac{\partial J(\omega,b)}{\partial \omega}$$

$$b = b - \alpha \boxed{\frac{dJ(\omega,b)}{db}} = \omega - \alpha \frac{\partial J(\omega,b)}{\partial b}$$

(V) Derivatives ✓

(VI) More Derivative Examples ✓

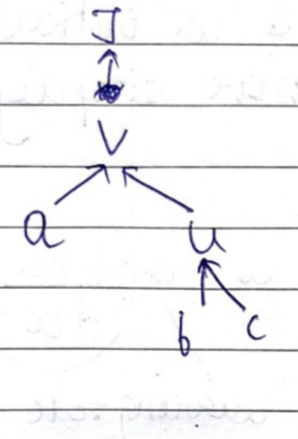
(VII) Computation Graph

It organises computation from left to right.

(VIII) Derivatives with a Computation Graph

dvar - derivative of final output w.r.t intermediate quantities

computation : R→L



$x \rightarrow y \rightarrow z$

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

(IX) Logistic regression gradient descent

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

$x_1, w_1, x_2, w_2, b \rightarrow z = w_1 x_1 + w_2 x_2 + b \rightarrow a = a(z) \rightarrow L(a,y)$

$$dz = \frac{dL}{dz} = \frac{dL}{da} \cdot \frac{da}{dz}$$

$$= a - y$$

$$da = \frac{d L(a,y)}{da}$$

$$= \frac{-y}{a} + \frac{1-y}{1-a}$$

PAGE NO.
DATE: / /

(X) Logistic regression on m example

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

$$= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial w_1} \mathcal{L}\left[a^{(i)}, y^{(i)}\right]$$

x_1 - feature
x_2 - feature
w_1 - weight of first feature
w_2 - weight of second feature
b - logistic regression parameter
m - number of training examples
$y(i)$ - expected output of i

Algo:

$J = 0$; $dw_1 = 0$; $dw_2 = 0$; $db = 0$

For loop ①:

for $i = 1$ to m
$z^{(i)} = w^T x^{(i)} + b$
$a^{(i)} = \sigma(z^{(i)})$
$J += \left[y^{(i)} \log a^{(i)} + 1 - y^{(i)} \log(1 - a^{(i)})\right]$
$dz^{(i)} = a^{(i)} - y^{(i)}$

for loop ② $\left\{ \begin{array}{l} dw_1 += x_1^{(i)} dz^{(i)} \\ dw_2 += x_2^{(i)} dz^{(i)} \end{array} \right.$ $n = 2$

$db += dz^{(i)}$

$J /= m$

$dw_1 /= m$ $= \frac{\partial J}{\partial m}$ (accumulator)

PAGE NO.
DATE: / /

$w_1 = w_1 - \alpha dw_1$
$w_2 = w_2 - \alpha dw_2$
$b = b - \alpha db$

for loops - makes code less efficient - code takes long time

vectorisation - speeds up code - gets rid of for loops

Vectorisation

* ~~Num~~ Numpy library uses vectorisation by default.
* Vectorisation can be done on CPU/ GPU ← faster

Logistic Regression pseudo code

$J = 0;$ ~~$w_1 = 0$~~
$dw1 = 0$ ~~$w_2 = 0$~~
$dw_2 = 0$ ~~$b = 0$~~
$db = 0$

$\boxed{dw = np.zeroes(n-x, 1)}$

$w_1 = 0 \; ; \; w_2 = 0 \; ; \; b = 0$

forward pass

for $i = 1$ to m
$z(i) = w1 * x1(i) + w_2 * x_2(i) + b$
$a(i) = \sigma z(i)$
$J += (y(i) * \log a(i) + 1 - y(i) \log(1 - a(i)))$

PAGE NO.
DATE: / /

backward pass

dw = np.zeros((n_x, 1))

$$dz(i) = a(i) - y(i)$$
$$dw1 \mathrel{+}= dz(i) * x1(i)$$
$$dw2 \mathrel{+}= dz(i) * x2(i)$$
$$db \mathrel{+}= dz(i)$$

$$\Bigg\}\ \boxed{dw \mathrel{+}= x^{(i)} z^{(i)}}$$

$$J \mathrel{/}= m$$
$$dw1 \mathrel{/}= m$$
$$dw2 \mathrel{/}= m$$
$$db \mathrel{/}= m$$

$$\Bigg\} \rightarrow \boxed{dw \mathrel{/}= m}$$

Gradient descent

$$w1 = w1 - alpha * dw1$$
$$w2 = w2 - alpha * dw2$$
$$b = b - alpha * db$$

⇒ Vectorising logistic regression

$$z^{(1)} = w^T x^{(1)} + b \qquad z^{(2)} = w^T x^{(2)} + b \qquad z^{(3)} = w^T x^{(3)} + b$$
$$a^{(1)} = \sigma(z^{(1)}) \qquad a^{(2)} = \sigma(z^{(2)}) \qquad a^{(3)} = \sigma(z^{(3)})$$

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

$$w^T \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$Z = [z^{(1)} \quad z^{(2)} \quad \cdots \quad z^{(m)}]$$
$$= \boxed{w^T X + [b \; b \; \cdots \; b]}$$
$$= [w^T X^{(1)} + b \quad w^T X^{(2)} + b \quad \cdots \quad w^T X^{(m)} + b]$$

PAGE NO.
DATE: / /

$$Z = np.dot(\omega^T, X + b)$$

$$A = [a^{(1)} \quad a^{(2)} \quad \ldots \quad a^{(m)}] = \boxed{\sigma(Z)}$$

$$dz^{(1)} = a^{(1)} - y^{(1)}$$
$$dz^{(2)} = a^{(2)} - y^{(2)}$$

$$dZ = [dz^{(1)} \quad dz^{(2)} \quad \ldots \quad dz^{m}]$$

$$A = [a^{(1)} \quad \ldots \quad a^{(m)}]$$

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

$$\boxed{dZ = A - Y}$$
$$= [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \ldots]$$

~~dw = 0~~ $db = 0$
$db \mathrel{+}= dz^{(1)}$
$db \mathrel{+}= dz^{(2)}$
$\vdots$
$db \mathrel{+}= dz^{(m)}$
$db \mathrel{/}= m$

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

$dw = 0$
$dw \mathrel{+}= x^{(1)} dz^{(1)}$
$\vdots$
$dw \mathrel{+}= x^{m} dz^{(m)}$
$dw \mathrel{/}= m$

$$dw = \frac{1}{m} X \, dz^T$$
$$= \frac{1}{m} \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$
$$= \frac{1}{m} \left(x^{(1)} dz^{(1)} + \ldots + x^{(m)} dz^{(m)} \right)$$

$$\omega = \omega - \alpha \, d\omega$$
$$b = b - \alpha \, db$$