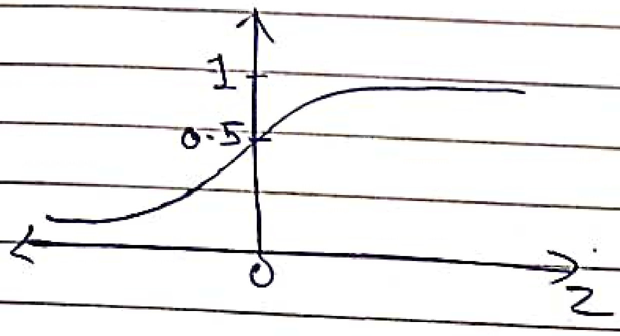
Logistic Regression

Given $x \rightarrow$ feature vector
Output $\rightarrow \hat{y} \rightarrow P(y=1|x)$
$\hat{y} \in [0,1]$, $x \in \mathbb{R}^{n_x}$

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

$\hat{y} = \sigma(w^T x + b)$ where $\sigma(z) = \dfrac{1}{1+e^{-z}}$
$\hookrightarrow$ Sigmoid



Training set $\rightarrow \{(x_1,y_1),(x_2,y_2), \ldots (x_m,y_m)\}$
want $\hat{y}_i \approx y_i$

Loss Function
$L(\hat{y},y) = -(y\log\hat{y} + (1-y)\log(1-\hat{y}))$

$y=0 \rightarrow L(\hat{y},y) = -\log(1-\hat{y})$
$y=1 \rightarrow L(\hat{y},y) = -\log\hat{y}$

This loss function makes the final optimizati convex in gradient descent with only one global minima, any other loss func creates optimization problem in gradient descent with many local minima

Cost function

$J(w,b) = \dfrac{1}{m}\sum_{i=1}^{m} L(\hat{y}_i, y_i) = \cancel{\dfrac{1}{m}\sum_{i=1}^{m}[y_i lo}$

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

$$\hat{y}_i = w_i x_i + b$$

Want to find w, b that minimise J by gradient descent

We first initialize w and b to zero or any random value then,

$$\left.\begin{aligned} w &= w - \alpha \frac{\partial J}{\partial w} \\ b &= b - \alpha \frac{\partial J}{\partial b} \end{aligned}\right\} \text{Repeat this in each iteration of gradient descent}$$

$\alpha \rightarrow$ learning rate determines how large step we take in gradient descent

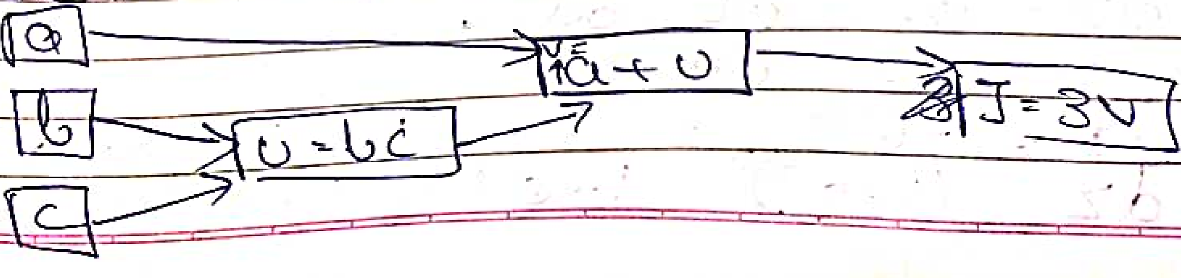
At global minima, $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$ are 0 hence no change in w and b and we get values which have minimum loss.

$$dw = \frac{\partial J}{\partial w} \qquad db = \frac{\partial J}{\partial b}$$

$$\therefore \; w = w - \alpha\, dw$$
$$b = b - \alpha\, db$$

Computation graph

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



- $\frac{dJ}{dv} = 3$
- $\frac{dJ}{da} = \frac{d(3v)}{da} = 3\frac{dv}{da} = 3 \quad \left(\frac{dv}{da} = 1\right)$

$\therefore \frac{dJ}{da} = \frac{dJ}{dv} \times \frac{dv}{da} = 3 \times 1 = 3$

- $\frac{dJ}{du} = \frac{dJ}{dv} \times \frac{dv}{du} = 3 \times 1 = 3$
- $\frac{dJ}{db} = \frac{dJ}{dv} \times \frac{dv}{du} \times \frac{du}{db} = 3 \times 1 \times c = 3c$

Gradient descent

$z = w^T x + b$

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

$L(a,y) = -y \log a - (1-y)\log(1-a)$

$$x_1, w_1, x_2, w_2, b \rightarrow \boxed{z = w_1x_1 + w_2x_2 + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{L(a,y)}$$

$\sigma'(z) = \frac{-e^{-z}}{(1+e^{-z})^2} \qquad \sigma(z) = \frac{1}{1+e^{-z}}$

- $\frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$ —— (1)

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

- $dz = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z}$

$\frac{da}{dz} = \frac{-e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = a(1-a)$

Page No.
Date

$\therefore dz = \left(\frac{-y}{a} + \frac{1-y}{1-a}\right) a(1-a)$

$dz = -y(1-a) + a(1-y)$

$dz = -y + ay + a - ay$

$dz = a - y$

$\therefore dz = \frac{\partial L}{\partial z} = a - y$ —— (2)

- $\frac{\partial L}{\partial w_1} = dw_1 = \frac{\partial L}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w_1}$

$= \frac{\partial L}{\partial z} \times \frac{\partial z}{\partial w_1}$

$= x_1(a-y)$

$\therefore dw_1 = x_1 dz$

- $\frac{\partial L}{\partial w_2} = x_2(a-y) = dw_2 = x_2 dz$

- $\frac{\partial L}{\partial b} = db = \frac{\partial L}{\partial z} \times \frac{\partial z}{\partial b}$

$= a - y$

$\therefore db = dz$

Now,

1] $w_1 = w_1 - \alpha dw_1$

2] $w_2 = w_2 - \alpha dw_2$

3] $b = b - \alpha db$

On a set of m training Egs

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(a_i, y_i)$$

$$a_i = \hat{y}_i = \sigma(z_i) = \sigma(w^T x_i + b)$$

$$\frac{\partial J}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial L}{\partial w}$$

$$\therefore \frac{\partial J}{\partial w} = \frac{1}{m}\sum dw$$

$J = 0 \quad dw_1 = 0 \quad dw_2 = 0 \quad db = 0$

for i in range m

$z_i = w^T x_i + b$

$a_i = \sigma(z_i)$

$J \mathrel{+}= -[y_i \log a_i + (1-y_i)\log(1-a_i)]$

$\cancel{dw}\, dz_i = a_i - y_i$

$dw_1 \mathrel{+}= \cancel{x_i dz_i}\ x_{1i} dz_i$

$dw_2 \mathrel{+}= \cancel{x_{2i}} dz_i\ x_{2i} dz_i$

$db \mathrel{+}= dz_i$

$J /= m \quad dw_1 /= m \quad dw_2 /= m \quad db /= m$

Now,

$w_1 = w_1 - \alpha dw_1$

$w_2 = w_2 - \alpha dw_2$

$b = b - \alpha db$

Standardizing data $= \frac{x - \mu}{\sigma}$

$(1, n_x)$ (n_x, m) n_y

In real life using Vectorization

$Z = w^T x + b$

(n_x, m)

$$X = \begin{bmatrix} | & | & | & & | \\ x_1 & x_2 & x_3 & \cdots & x_n \\ | & | & | & & | \end{bmatrix} \rightarrow \text{numpy arrays}$$

$(m, 1)$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n} \end{bmatrix} \quad \cancel{b \in R} \quad w^T = (1, m)$$

$(1, m)$

$b = [b, b, b \cdots b]$

So $Z = np.dot(w.T, X) + b$ $(1,m)$ $\cancel{(n_x, 1)}$

$X = (n_x, m)$ sigmoid(x): return $[1/(1 + np.exp(-x))]$

$Y = \cancel{(m)}\ (1, m)$ $A = \text{sigmoid}(Z)$ $(1,m)$ $\cancel{(n_x, 1)}$

$\cancel{w = (m, 1)}$ $\cancel{dw = np.dot(X, Z)}$

$w = (n_x, 1)$ $dZ = A - Y$ $(1,m)$ $\cancel{(n_x, 1)}$

$dw = np.dot(X, \cancel{dZ}\ dZ.T)/m$

$db = \cancel{dZ/m}\ np.sum(dZ)/m$

$J = -[np.dot(y, np.log(A)) + np.dot(1-y, np.log(1-A))]/m$

$w = w - \alpha dw$

$b = b - \alpha db$

$dz = [dz_1\ dz_2\ dz_3 \cdots dz_m]$

$dw = [dw_1\ dw_2\ dw_3 \cdots dw_m]$

$db =$

Page No.
Date

Dimensions

$X = (n_x, m)$ $Y = (1, m)$

$w = (n_x, 1)$ $\therefore$ $w^T = (1, n_x)$

$b = (1, 1)$

$Z = w^T X + b$

$Z = (1, m)$

$A = \sigma(Z)$

$A = (1, m)$

m = no. of training Examples

Code

```
Z = np.dot(wT, X) + b

A = sigmoid(Z)
dZ = A - Y
dw = np.dot(X, dZT)/m
db = np.sum(dZ, axis=1, keepdims=True)/m

~~cost = np.sum~~
cost = -np.sum((Y * np.log(A)) + ((1-Y) *
                          np.log(1-A)))/m

w = w - αdw
b = b - αdb
```

$dZ = (1, m)$

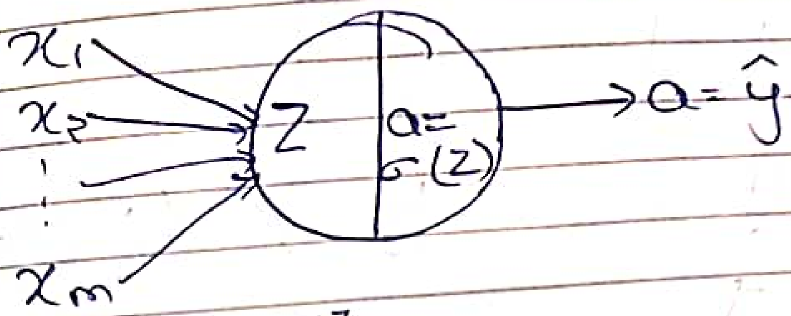
$dw = (n_x, 1)$

$db = (1, 1)$

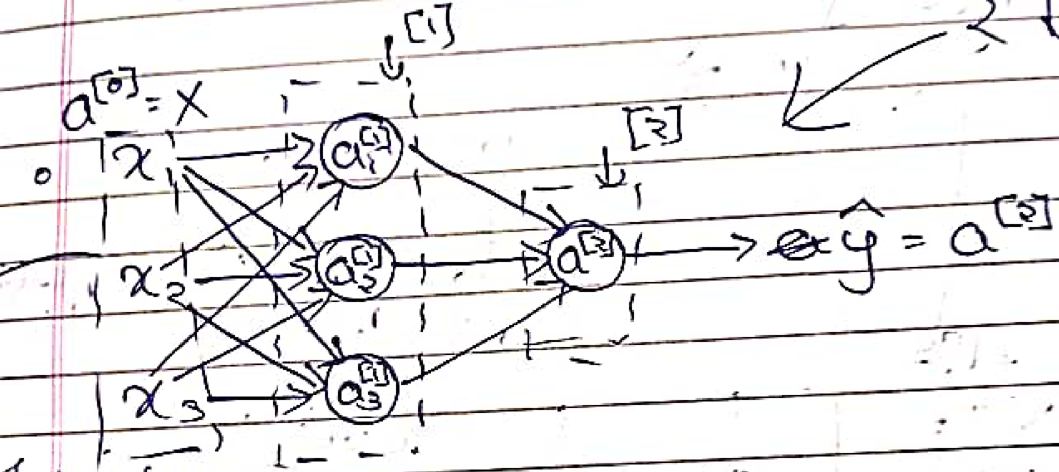
* An image of shape (~~h~~height, width, colourchannels) has to be reshaped in the shape (height*width*colorchannels, 1) for input

Neural Network

Logistic Regression Model is a single node neural network



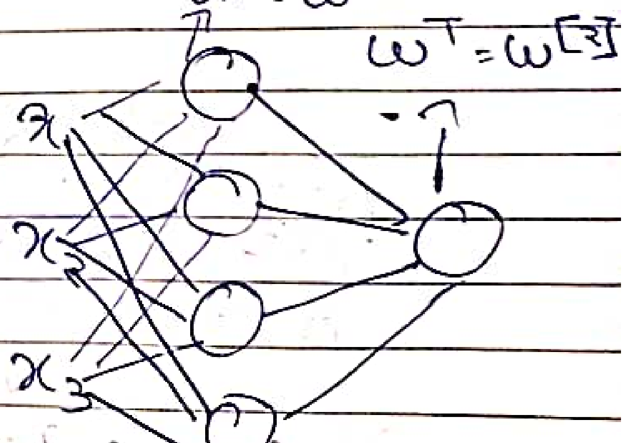
2 layer NN



x, $w^{[1]}$, $b^{[1]}$ $\rightarrow$ $\boxed{Z^{[1]} = w^{[1]}x + b^{[1]}}$ $\rightarrow$ $\boxed{a^{[1]} = \sigma(Z^{[1]})}$ $\rightarrow$ $\boxed{Z^{[2]} = w^{[2]}a^{[1]} + b^{[2]}}$ $\rightarrow$ $\boxed{a^{[2]} = \sigma(Z^{[2]})}$ $\rightarrow$ $\hat{y} = a^{[2]}$

$w^T = w^{[1]}$

$w^T = w^{[2]}$



$$\underset{(4,1)}{Z^{[1]}} = \underset{(4,3)}{w^{[1]}}\underset{(3,1)}{x} + \underset{(4,1)}{b^{[1]}}$$

$$\underset{(4,1)}{a^{[1]}} = \sigma\underset{(4,1)}{(Z^{[1]})}$$

$$\underset{(1,1)}{Z^{[2]}} = \underset{(1,4)}{w^{[2]}}\underset{(4,1)}{a^{[1]}} + \underset{(1,1)}{b^{[2]}}$$

$$\underset{(1,1)}{a^{[2]}} = \sigma\underset{(1,1)}{(Z^{[2]})}$$

For m training Examples (For 2 layer NN)

$$X = \begin{bmatrix} | & | & | & & | \\ x_1 & x_2 & x_3 & \cdots & x_m \\ | & | & | & & | \end{bmatrix} \quad (n_x, m)$$

$$Y = [y_1 \; y_2 \; \cdots \; y_m] \quad (1, m)$$

- For layer 1

$$W^{[1]} = \begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{bmatrix} \quad (4,3)$$

~~$b = [b_1 \; b_2 \; b_3$~~

~~$b = [b^{[1]}_1 \cdots b^{[1]}_4]$~~

$$b = \begin{bmatrix} b^{[1]}_1 \\ \vdots \\ b^{[1]}_4 \end{bmatrix}$$

$$W^{[1]}X + b^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix} = Z^{[1]}$$

~~$(4, m)$~~

$$A^{[1]} = \sigma(Z^{[1]}) = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = \sigma(Z^{[1]})$$

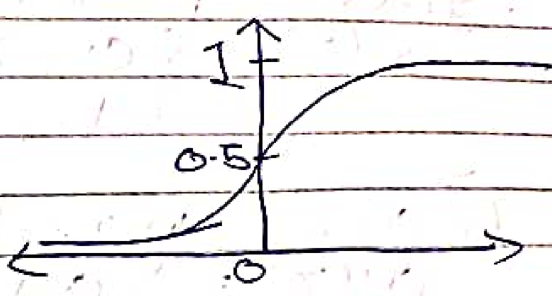
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = \hat{y} = \sigma(Z^{[2]})$$

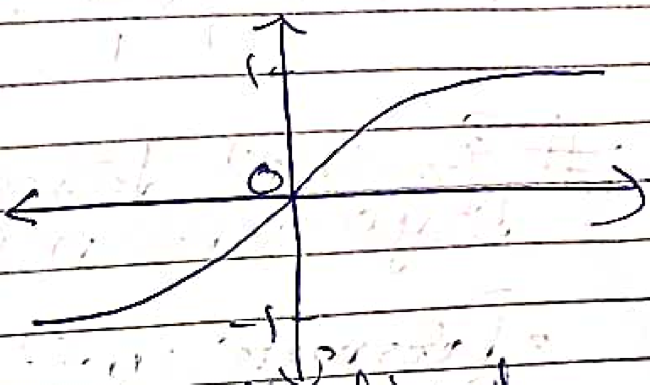
X can also be written as $A^{[0]}$

Activation Func

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

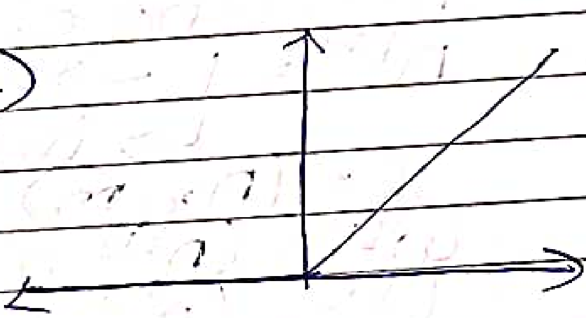


2] $\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

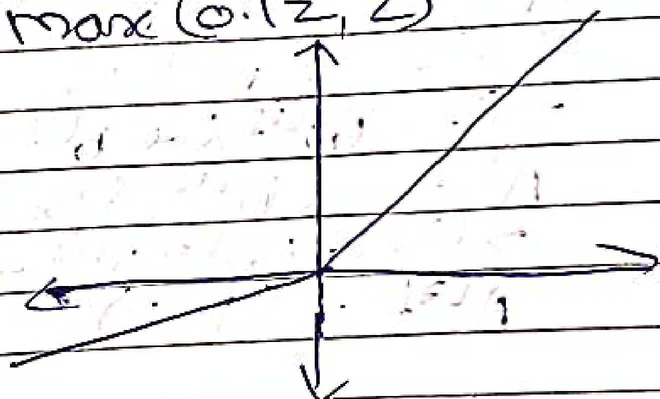


tanh better than
sigmoid as output
has mean closer to
zero hence output more centralized

3] $ReLU(z) = \max(0, z)$
Rectified Linear unit



4] Leaky $ReLU(z) = \max(0.1z, z)$



1] $a^{(z)} = \frac{1}{1+e^{-z}}$ $\quad a'(z) = \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right) = a(1-a)$

2] $a(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$ $\quad a'(z) = 1-\left(\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}\right)^2 = 1-a^2$

Page No.
Date

3] ~~a = max (0,~~
3] $a(z) = \max(0, z)$

$a'(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$

4] $a(z) = \max(0.1z, z)$

$a'(z) = \begin{cases} 0.1, & z < 0 \\ 1, & z \geq 0 \end{cases}$

Gradient Descent for a single hidden layer neural network

- Parameter: $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}$

$n_x = n^{[0]} \rightarrow$ no. of features of x

$n^{[1]} \rightarrow$ no. of hidden units (nodes)

$n^{[2]} = 1 \rightarrow$ no. of output units
↳ in our case

$X = (n_x, m)$ $Y = (1, m)$

$w^{[1]} = (n^{[1]}, n^{[0]})$

$b^{[1]} = (n^{[1]}, 1)$

$w^{[2]} = (n^{[2]}, n^{[1]})$

$b^{[2]} = (n^{[2]}, 1)$

$Z^{[1]} = w^{[1]}X + b^{[1]}$ $(n^{[1]}, n_x)$

$A^{[1]} = (n^{[1]}, n_x)$

$Z^{[2]} = w^{[2]}A^{[1]} + b^{[2]}$ $(1, n^{[2]}) = (1,1)$

$A^{[2]} = (1,1)$

$$J(w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}, y)$$

$\hat{y} = a^{[2]}$

$A^{[2]} = \sigma(Z^{[2]}) = \sigma(w^{[2]}A^{[1]} + b^{[2]})$

$= \sigma[w^{[2]}(\sigma(Z^{[1]})) + b^{[2]}]$

$\therefore A^{[2]} = \sigma\{w^{[2]}[\sigma(w^{[1]}X + b^{[1]})] + b^{[2]}\}$

Page No.
Date

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$

$$dZ^{[2]} = \frac{\partial J}{\partial Z^{[2]}} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{\partial J}{\partial W^{[2]}} = \frac{1}{m} dZ^{[2]} A^{[1]^T}$$

$$db^{[2]} = \frac{\partial J}{\partial b^{[2]}} = \frac{1}{m}$$ np.sum($dZ^{[2]}$, axis = 1, keepdims = True)

$$W^{[2]} = W^{[2]} - \alpha\, dW^{[2]}$$
$$b^{[2]} = b^{[2]} - \alpha\, db^{[2]}$$

$$dZ^{[1]} = \frac{\partial J}{\partial Z^{[1]}} = \underbrace{W^{[2]^T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]'}(Z^{[1]})}_{(n^{[1]}, m)}$$

$*$ → elementwise product

$$dW^{[1]} = \frac{\partial J}{\partial W^{[1]}} = \frac{1}{m} dZ^{[1]} X^T$$

$$db^{[1]} = \frac{\partial J}{\partial b^{[1]}} = \frac{1}{m}$$ np.sum($dZ^{[1]}$, axis = 1, keepdims = True)

$$W^{[1]} = W^{[1]} - \alpha\, dW^{[1]}$$
$$\cancel{W^{[2]} = W}$$
$$b^{[1]} = b^{[1]} - \alpha\, db^{[1]}$$

Always initialize $W^{[1]}$ and $W^{[2]}$ randomly and not zero, $b^{[1]}$, $b^{[2]}$ can be zero

$W^{[1]}$ = np.random.randn(($n^{[1]}$, $n^{[0]}$)) * 0.01
$b^{[1]}$ = np.zeros(($n^{[1]}$, 1))
$W^{[2]}$ = ← $b^{[2]} = 0.0$

In a Deep neural network having L layers (Input layer is layer 0 and output layer is layer L) and each layer having $n^{[l]}$ nodes/units where $l = 1, \dots L$

$m \rightarrow$ no. of training Examples

$n^{[0]} = a^{[0]} = n_x \rightarrow$ no. of features of a single training Example

- Dimensions

1] $X = (n_x, m) = \cancel{(n^{[0]}, m)}\ (a^{[0]}, m) = (n^{[0]}, m)$

2] $Y = (1, m)$

3] $w^{[l]} = (n^{[l]}, n^{[l-1]}) = dw^{[l]}$

~~4]~~ $\cancel{b^{[l]} = (1, n^{[l]})\ (n^{[l]}, m) = db^{[l]}}$

5] $Z^{[l]} = (n^{[l]}, m) = dZ^{[l]}$

6] $A^{[l]} = (n^{[l]}, m) = dA^{[l]}$

4] $b^{[l]} = (n^{[l]}, 1) \rightarrow$ it then gets broadcasted to $(n^{[l]}, m)$

7] $Z^{[l]} = w^{[l]} A^{[l-1]} + b^{[l]}$

8] $A^{[l]} = g^{[l]}(Z^{[l]})$

9] $dZ^{[l]} = \cancel{A^{[l]}}\ \overbrace{w^{[l+1]^T} \cdot dZ^{[l+1]}}^{da^{[l]}} * g^{[l]'}(Z^{[l]})$

$(dZ^{[L]} = A^{[L]} - Y)$

10] $dw^{[l]} = \frac{1}{m} dZ^{[l]} A^{[l-1]^T}$

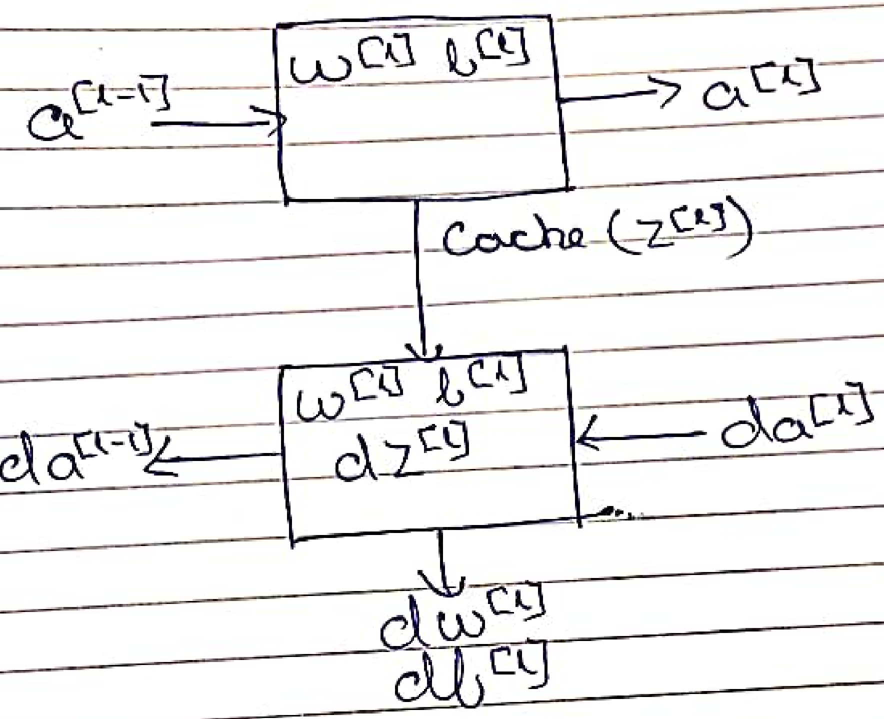
11] $db^{[l]} = \frac{1}{m}$ np.sum($dZ^{[l]}$, axis=1, keepdims=True)

12] $w^{[l]} = w^{[l]} - \alpha dw^{[l]}$

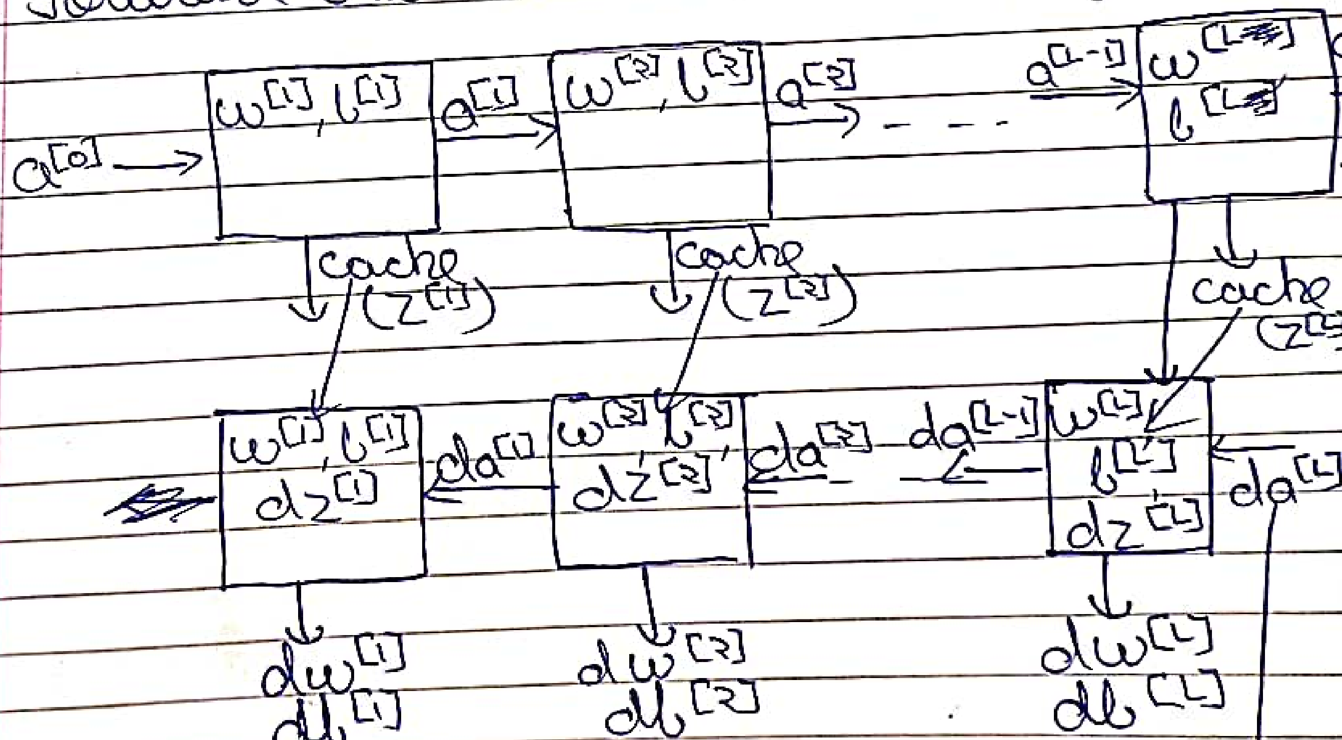
$b^{[l]} = b^{[l]} - \alpha db^{[l]}$

13] $da^{[l-1]} = w^{[l]^T} dZ^{[l]}$

Layer l



Forward and Backwards Propagation



$$w^{[l]} = w^{[l]} - \alpha\, dw^{[l]}$$
$$b^{[l]} = b^{[l]} - \alpha\, db^{[l]}$$
$$da^{[L]} = \frac{-y}{a^{[L]}} + \frac{(1-y)}{(1-a^{[L]})}$$