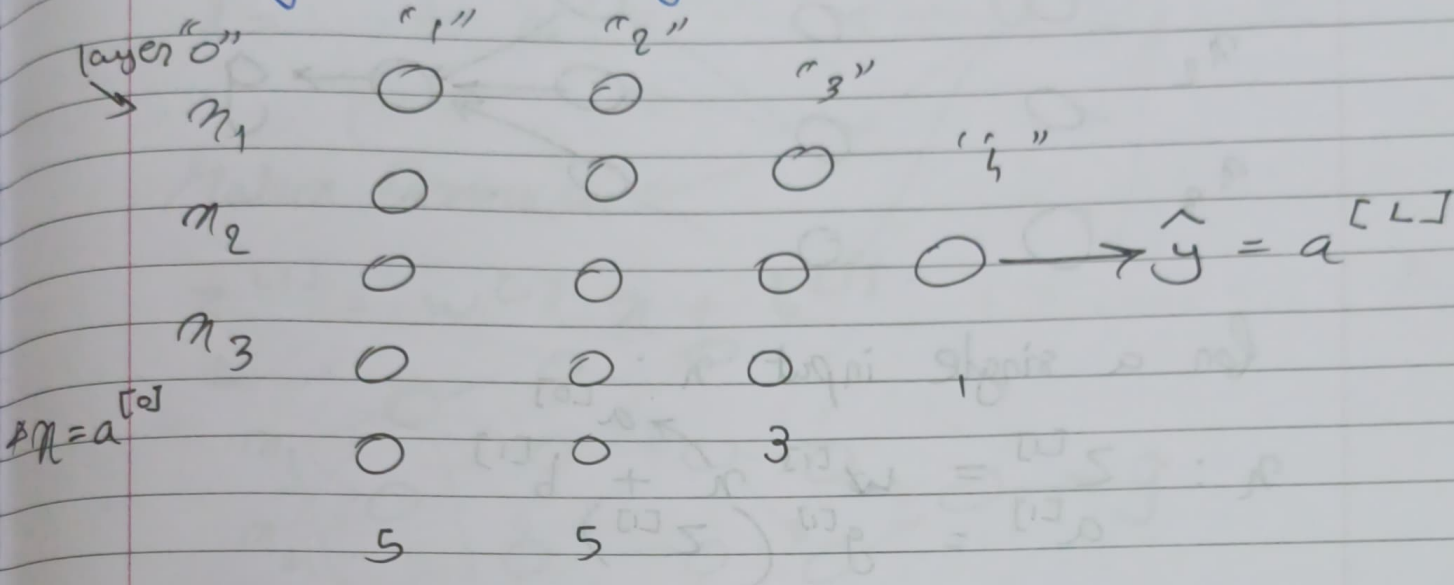
Page No.
Date

Week 4 : course 1

A deep neural network is the one with many hidden layers.



$L = 4$ (# layers)

$n^{[l]}$ = # of units in layer l

$n^{[1]} = 5,\ n^{[2]} = 5,\ n^{[3]} = 3,\ n^{[4]} = 1 = n^{[L]}$

$n^{[0]} = n_x = 3$

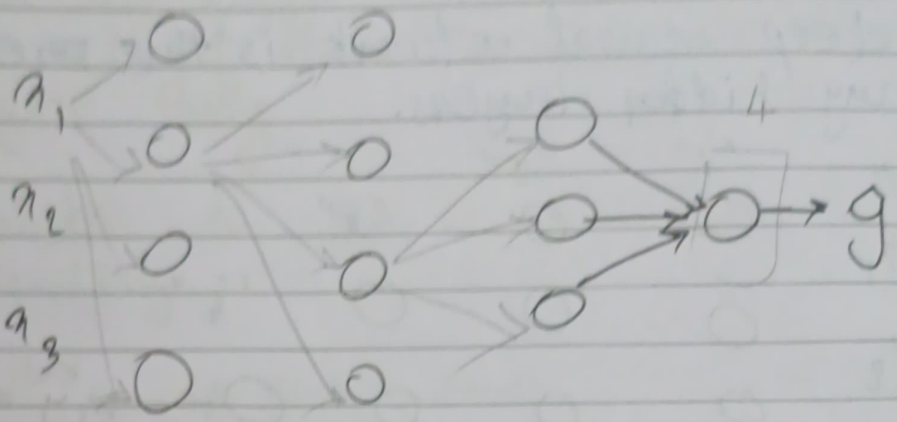
$a^{[l]}$ = activations in layer l

$a^{[l]} = g^{[l]}(z^{[l]})$

$w^{[l]}$ = weights for computing $z^{[l]}$

$b^{[l]}$ = bias for "_____"

Forward propagation in deep network:



For a single input x:

x: $z^{[1]} = w^{[1]} x + b^{[1]}$ ($x = a^{[0]}$)

$a^{[1]} = g^{[1]}(z^{[1]})$

$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$

$a^{[2]} = g^{[2]}(z^{[2]})$

$[z^{[1](1)} z^{[2](1)} \dots]$

...

$z^{[4]} = w^{[4]} a^{[3]} + b^{[4]}$

$a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

Vectorized (for 'm' training examples)

$X = A^{[0]}$

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

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

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

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

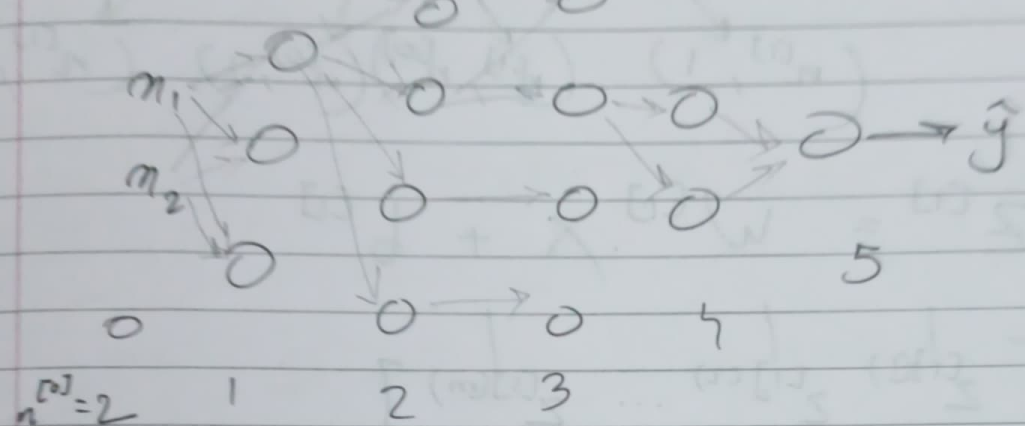
$b^{[l]} = (n^{[l]}, 1)$, $db = (n^{[l]}, 1)$

$$\hat{y} = A^{[4]} = g^{[4]}(z^{[4]})$$

we are going to compute above vectorized computations using for loop $l = 1$ to 4.

Matrix Dimensions:

$$z^{[1]} = w^{[1]} x + b^{[1]}$$



$n^{[1]} = 3$, $n^{[2]} = 4$, $n^{[3]} = 4$, $n^{[4]} = 2$, $n^{[5]} = 1$

$$z^{[1]} = w^{[1]} x + b^{[1]}$$

(3,1) (3,2) (2,1) (3,1)

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

$w^{[2]} = (n^{[2]}, n^{[1]}) = (4, 3)$

$[z]^{[2]} = w^{[2]} a^{[1]} + [b]^{[2]}$

(4,1) (4,3) (4,1) (4,1)

Note: $w^{[l]} = (n^{[l]}, n^{[l-1]})$

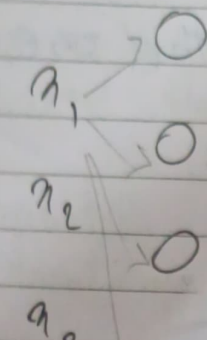
$b^{[l]} = (n^{[l]}, 1)$

$w^{[3]} = (n^{[3]}, n^{[2]}) = (4, 4)$

$w^{[4]} = (n^{[4]}, n^{[3]}) = (2, 4)$

$w^{[5]} = (n^{[5]}, n^{[4]}) = (1, 2)$

Forward propagation in



Recall, $z^{[1]} = w^{[1]} x + b^{[1]}$

$(n^{[1]}, 1) \quad (n^{[1]}, n^{[0]}) \quad (n^{[0]}, 1) \quad (n^{[1]}, 1)$

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

$$\rightarrow \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

$\Rightarrow Z^{[1]} \rightarrow (n^{[1]}, m)$

$W^{[1]}$ stays the same i.e. $(n^{[1]}, n^{[0]})$

$X \rightarrow (n^{[0]}, m)$

& $b \rightarrow$ through python broadcasting changes from $(n^{[1]}, 1)$ to $(n^{[1]}, m)$.

$$a^{[\ell]} = g^{[\ell]}(z^{[\ell]})$$

$\because z^{[\ell]} \rightarrow (n^{[\ell]}, 1)$

$\Rightarrow a^{[\ell]} \rightarrow (n^{[\ell]}, 1)$

Vectorized :

We know that, $z^{[1]} = w^{[1]} x + b^{[1]}$

$(n^{[1]}, 1) \quad (n^{[1]}, n^{[0]}) \quad (n^{[0]}, 1) \quad (n^{[1]}, 1)$

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

$$\rightarrow \begin{bmatrix} z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

$\Rightarrow Z^{[1]} \rightarrow (n^{[1]}, m)$

$W^{[1]}$ stays the same i.e. $(n^{[1]}, n^{[0]})$

$X \rightarrow (n^{[0]}, m)$

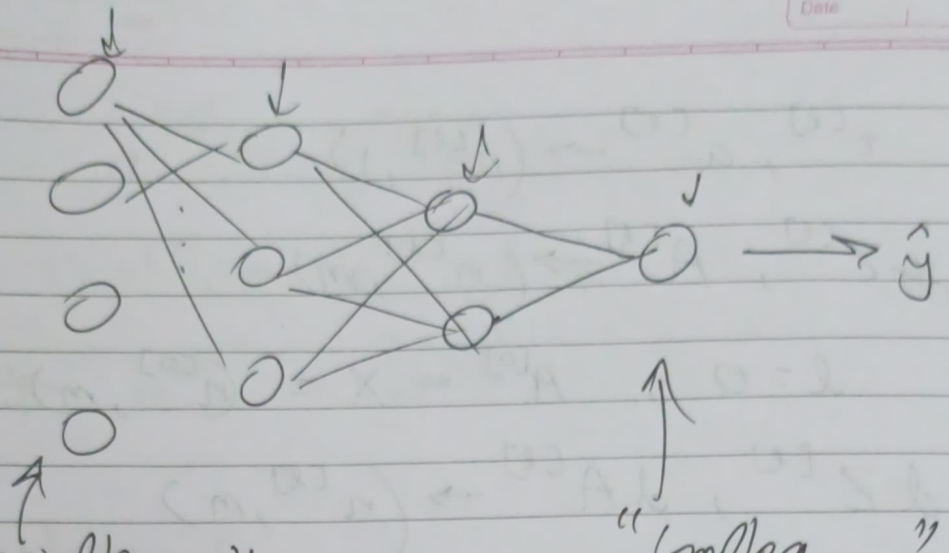
& $b \rightarrow$ through python broadcasting changes from $(n^{[1]}, 1)$ to $(n^{[1]}, m)$.

$z^{[l]}, a^{[l]} \rightarrow (n^{[l]}, 1)$

$Z^{[l]}, A^{[l]} \rightarrow (n^{[l]}, m)$

$l = 0, \quad A^{[0]} = X = (n^{[0]}, m)$

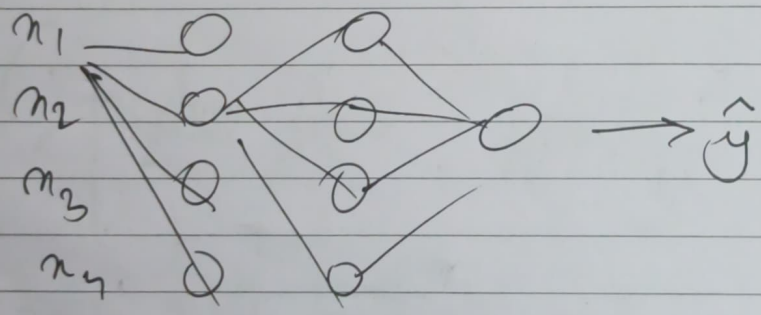
$dZ^{[l]}, dA^{[l]} \rightarrow (n^{[l]}, m)$



"simpler" computations like detecting small edges of image

"Complex" computations like detecting faces, words or phrases

Building Blocks of NN:



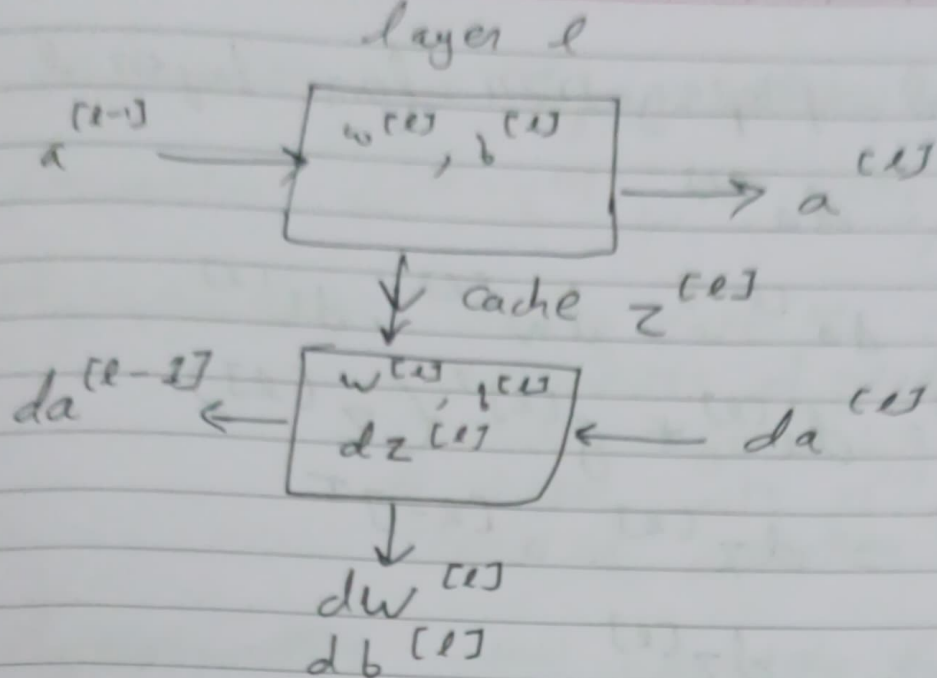
layer l: $w^{[l]}$, $b^{[l]}$

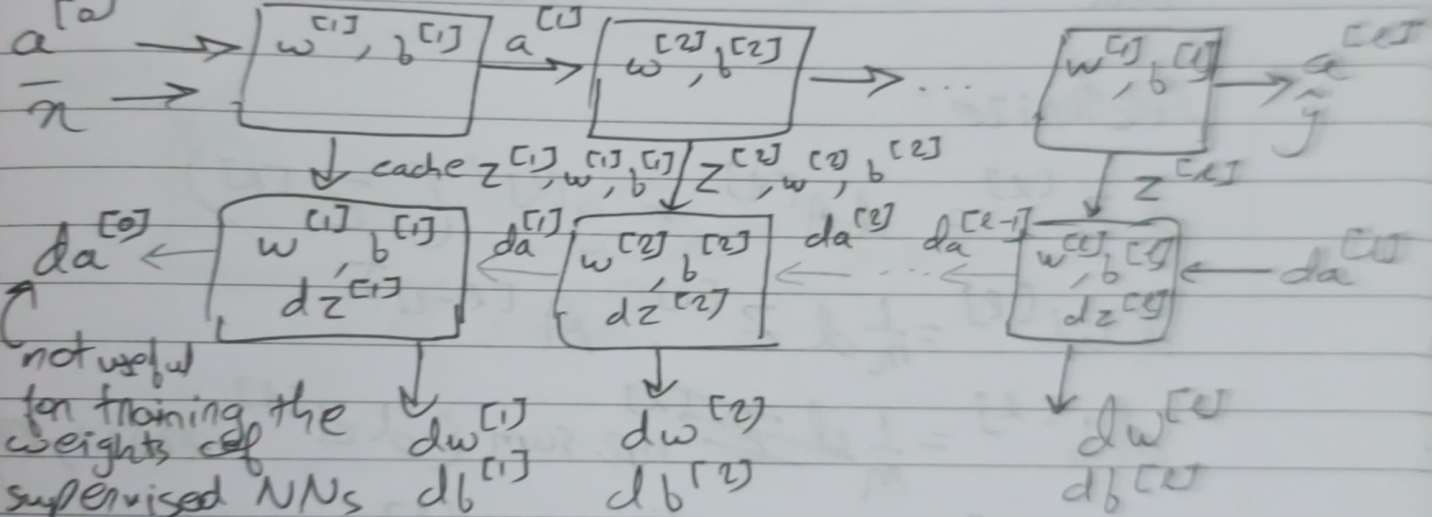
Forward: Input $a^{[l-1]}$, Output $a^{[l]}$

$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]} \rightarrow \text{cache } z^{[l]}$$
(store temporarily)

$$a^{[l]} = g^{[l]}(z^{[l]})$$

Backwards: Input $da^{[l]}$, output $da^{[l-1]}$
cache($z^{[l]}$)
$dw^{[l]}$
$db^{[l]}$





$$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}$$
$$b := b^{[l]} - \alpha\, db^{[l]}$$

Forward propagation for layer l

$$Z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(Z^{[l]})$$

Vectorized version of initializing backward propagation:

$$dA^{[l]} = \left(-\frac{y^{(1)}}{a^{(1)}} + \frac{1-y^{(1)}}{1-a^{(1)}} - \cdots + \frac{1-y^{(m)}}{1-a^{(m)}}\right)$$

Page No.
Date

Backward propagation for layer l

Input : $da^{[l]}$

Output : $da^{[l-1]}, dw^{[l]}, db^{[l]}$

$$dz^{[l]} = da^{[l]} * g^{[l]'}(z^{[l]})$$

$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]T}$$

$$db^{[l]} = dz^{[l]}$$

$$da^{[l-1]} = w^{[l]T} \cdot dz^{[l]}$$

Vectorize:

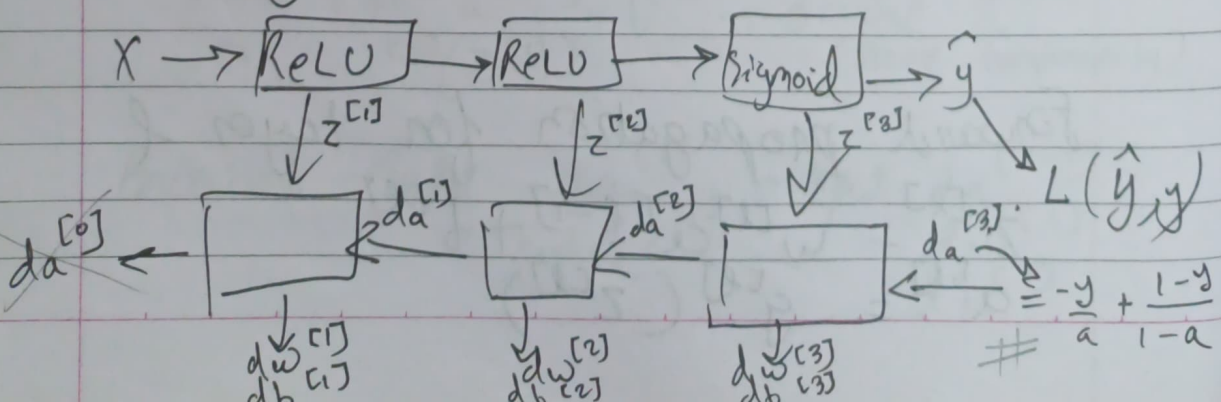
$$dZ^{[l]} = dA^{[l]} * g^{[l]'}(Z^{[l]})$$

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

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

$$dA^{[l-1]} = w^{[l]T} \cdot dZ^{[l]}$$

Summary:



Page No.
Date

Parameters vs Hyperparameter:

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \ldots$

Hyperparameters: Learning rate α
iterations of gradient descent
Hidden layers L
Hidden units $n^{[1]}, n^{[2]}, \ldots$
choice of activation function

Hyperparameters control ultimately the parameters (w, b)