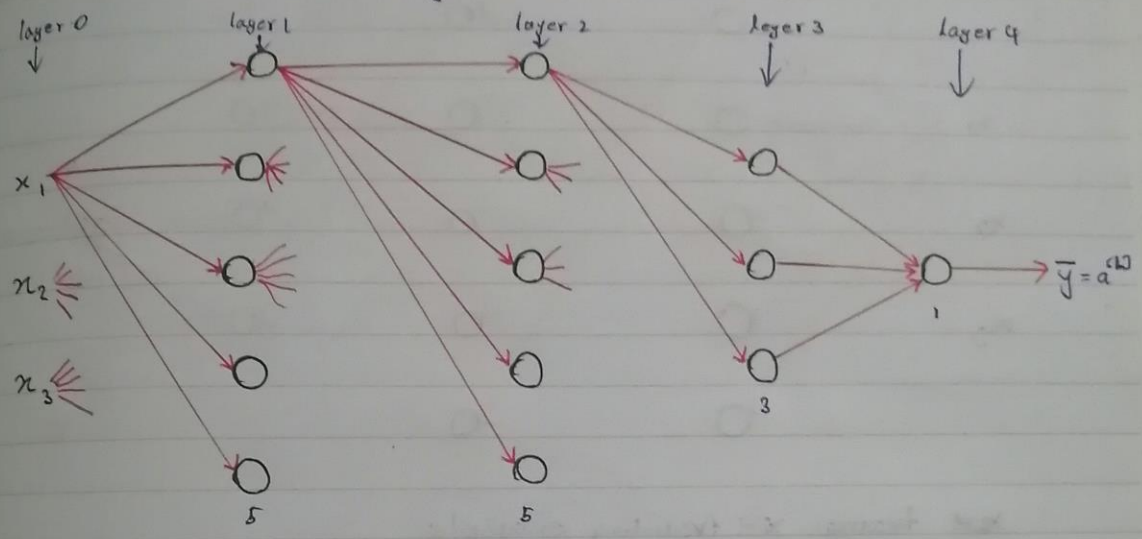
Week 4

Deep Neural Network

Deep L-layer Neural Network



$L = 4$ (Number of layers)

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

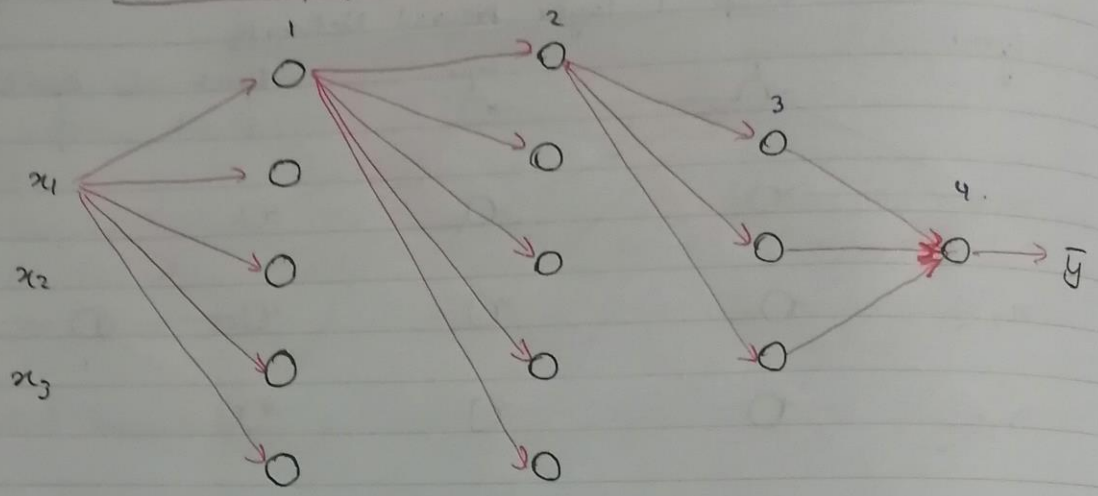
$a^{[l]}$ = auto notations in layer l

$a^{[l]} = g^{[l]}(z^{[l]})$, $w^{[l]}$ = weights for $z^{[l]}$

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

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

Forward Propagation in a Deep Network



~~$x^{(i)}$~~ ~~training~~ x : training example

layer 1

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

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

layer 2

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

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

layer 3

$z^{[3]} = w^{[3]}a^{[2]} + b^{[3]}$

$a^{[3]} = g^{[3]}(z^{[3]})$

layer 4

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

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

formula

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

* Now we look into vectorized method for above calculations.

layer 1

$$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$

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

layer 2

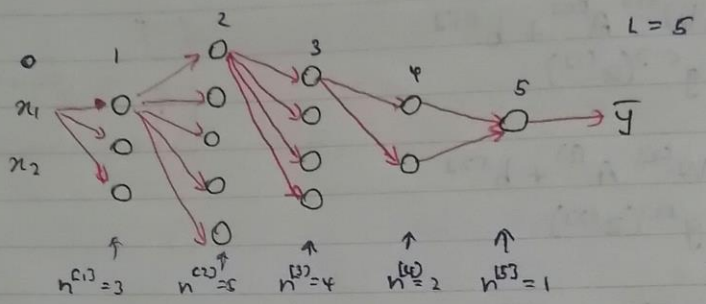
$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$

$$\hat{Y} = g(Z^{[4]}) = A^{[4]}$$

vectorized version formula

$$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$

Getting your Matrix Dimention right



layer 1

$$\underset{(n^{[1]},1)}{z^{[1]}} = \underset{(n^{[1]},n^{[0]})}{w^{[1]}} \underset{(n^{[1]},1)}{x} + b^{[1]}$$

$$\begin{bmatrix} \vdots \end{bmatrix} = \begin{bmatrix} \vdots\vdots \end{bmatrix} \begin{bmatrix} \vdots \end{bmatrix}$$

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

$w^{[2]} : (5,3)\ (n^{[2]}, n^{[1]})$

layer 2

$$\underset{(5,1)}{z^{[2]}} = \underset{(5,3)}{w^{[2]}} \underset{(3,1)}{a^{[1]}} + b^{[2]}$$

likwise → $w^{[3]} : (4,5)$

$w^{[4]} : (2,4)$

$w^{[5]} : (1,2)$

Vectorized implementation above model

$$\underset{(n^{[l]}, m)}{Z^{[l]}} = \underset{(n^{[l]}, n^{[0]})}{W^{[l]}} \cdot \underset{(n^{[0]}, m)}{X} + \underset{(n^{[l]}, m)}{b^{[l]}}$$

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

note

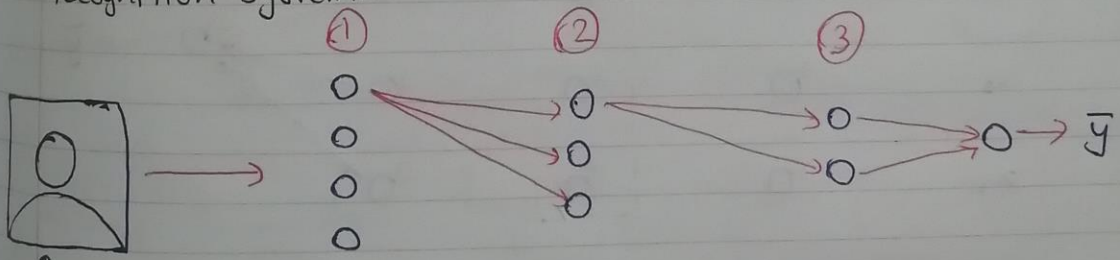
$z^{[l]}, d^{[l]} : (n^{[l]}, 1)$

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

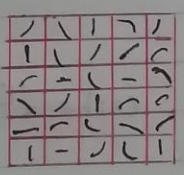
$dZ^{[l]}, dA^{[l]} : (n^{[l]}, m)$

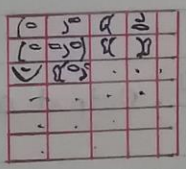
Why Deep Representations?

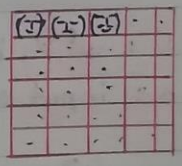
* for an example, let's assume we are going to build a face recognition system.



Picture of a face





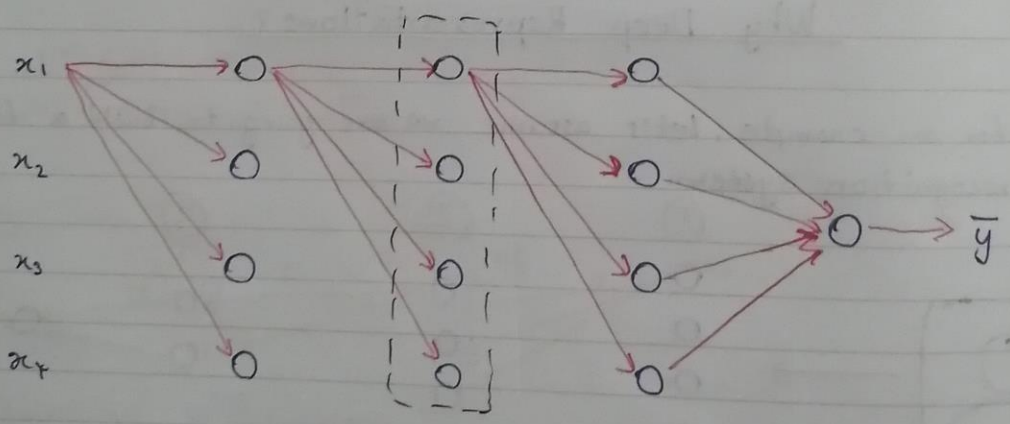


① → You enter a picture of a face and then first layer of the neural network, we can think may be use for feature detector of an edge detector. given all squre bones represent edges of the given picture.

② → In this layer, edges and group edges detected together and it represent parts of the faces (nose, eyes, ears, Etc.)

③ → In this layer, it putting together different parts of faces and try to recognize or detect different types of faces.

Building Blocks of Deep Neural Networks



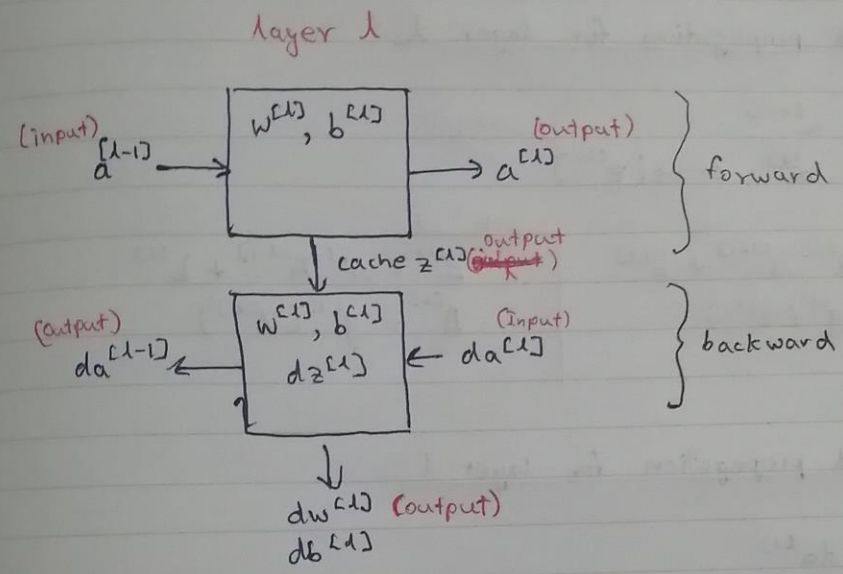
layer l : $w^{[l]}$, $b^{[l]}$

forward : input $a^{[l-1]}$, Output $a^{[l]}$, cache $z^{[l]}$

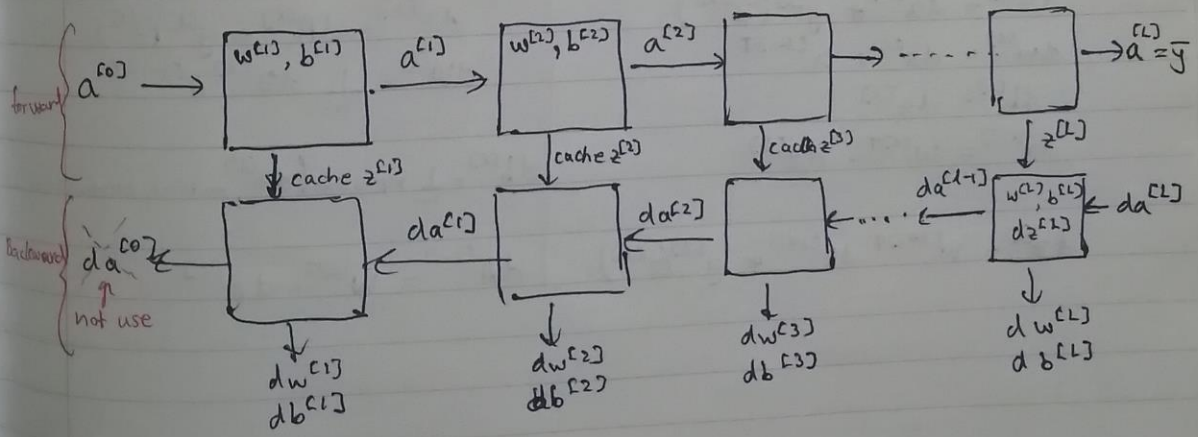
$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$$

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

Backward : input $da^{[l]}$, output $da^{[l-1]}$, cache $z^{[l]}$



* After applying above method into neural network



Forward and Backward Propagation

Forward propagation for layer l

Input $a^{[l-1]}$

Output $a^{[l]}$, cache($z^{[l]}$) ← $w^{[l]}, b^{[l]}$

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

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

Vectorized

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

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

Backward propagation for layer l

Input $da^{[l]}$

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

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

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

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

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

$dz^{[l]} = w^{[l+1]T} dz^{[l+1]} * g^{[l]}(z^{[l]})$

vectorized

$dz^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$

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

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

$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$

Parameteres vs Hyperparameteres:

What are hyperparameters?

Parameters :- $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}, \ldots$

Hyperparameteres :- learning rate α (ex:- number of iterations of gradient descent you carry out
:- number of hidden layers (L)
:- number of hidden unitr ($n^{[1]}, n^{[2]} \ldots$)

→ :- choice of activation function

* hyperparameters need to control parameters above mentioned ($W^{[1]}, b^{[1]}, \ldots$)
* hyper parameters determine final values of W and b.
* In later we will learn these hyper parameters :- momentom term, mini batch size various forms of regularization parameters, ..