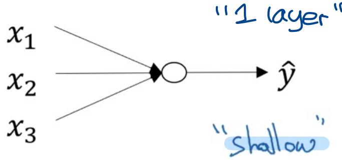
Deep L-layer Neural Network

What is a deep neural network?

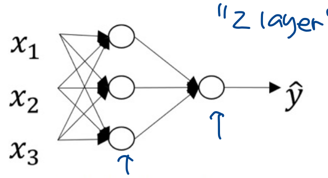
"1 layer" NN



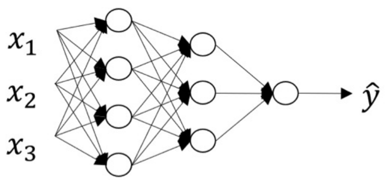
"shallow"

logistic regression

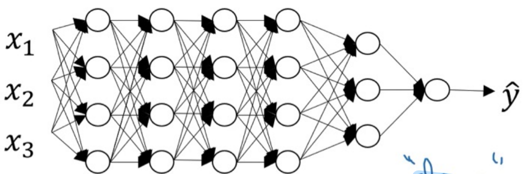
"2 layer" NN



1 hidden layer



2 hidden layers



5 hidden layers

"deep"

Andrew Ng

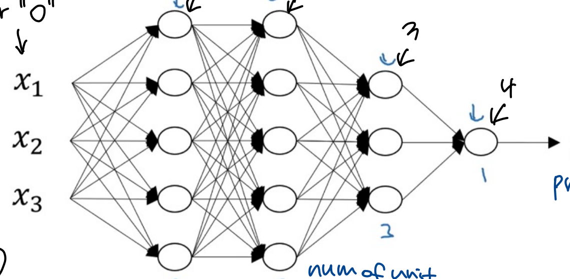
Number of Hidden Layer would be another hyper parameter.

Deep neural network notation

4 layer NN

$X = a^{[0]}$

layer "0"



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

prediction output.

num of unit

$L = 4$ (#layers)

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

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

($\because$ five hidden units)

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

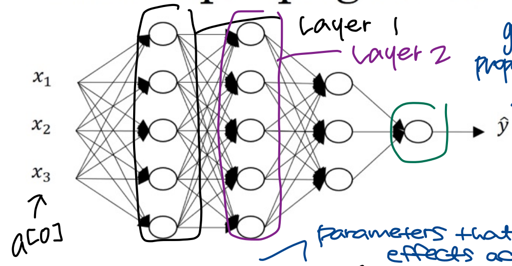
$a^{[l]}$ = activations in layer l

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

Forward Propagation in a Deep Network.

Forward propagation in a deep network

$A^{[0]} = X$



Layer 1

Layer 2

$a^{[0]}$

general propagation of forward.

$$\boxed{\begin{aligned} Z^{[l]} &= W^{[l]} A^{[l-1]} + b^{[l]} \\ A^{[l]} &= g^{[l]}(Z^{[l]}) \end{aligned}}$$

capital

change to vectorized version.

$(X = A^{[0]})$

<vectorize of whole training set>

parameters that effects activation functions.

X: $z^{[1]} = W^{[1]} \cancel{x} + b^{[1]}$ equal to $a^{[0]}$

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

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

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

$\vdots$

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

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

all of these equations seems to be same.

$Z^{[1]} = W^{[1]} \cancel{X} A^{[0]} + b^{[1]}$

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

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

$A^{[2]} = g^{[2]}(Z^{[2]})$

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

exactly same logic with $z^{[2]}$

for $l = 1..4$

(no way but only for loop here)

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

column vector stack left to right

Andrew Ng

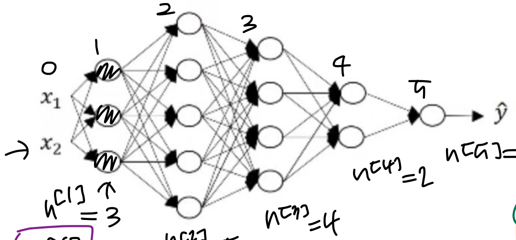
Getting Matrix Dimensions Right

Parameters $W^{[l]}$ and $b^{[l]}$

4 hidden layers

1 output layer

$L=5$ there are 5 layers (not counting input layers)



0 1 2 3 4 5

n dimensional vector.

$n^{[0]} = n_x = 2$ $n^{[1]}=3$ $n^{[2]}=5$ $n^{[3]}=4$ $n^{[4]}=2$ $n^{[5]}=1$ - single output unit.

$z^{[1]} = \boxed{W^{[1]} \cdot x} + \boxed{b^{[1]}}$

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

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

$[\,:\,] = [\,:\,:\,:\,][\,:\,]$

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

have to have same dimension

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

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

$z^{[2]} = \boxed{W^{[2]} \cdot a^{[1]}} + \boxed{b^{[2]}}$

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

$W^{[3]}: (4,5)$ $(n^{[3]}, n^{[2]})$

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

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

Ⓥ $W^{[l]}: (n^{[l]}, n^{[l-1]})$ general formula.

Ⓥ $b^{[l]}: (n^{[l]}, 1)$ help double check

$dW^{[l]}: (n^{[l]}, n^{[l-1]})$

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

Andrew Ng

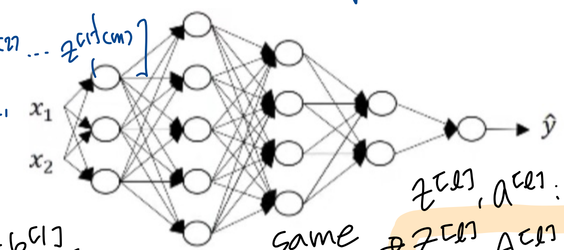
<Vectorized implementation>

even for vectorization, W, b, dW, db's dimension are same.

But the dimension of z, a, x are changed a bit.

$Z^{[1]} = [\, z^{[1](1)} \; z^{[1](2)} \cdots z^{[1](m)} \,]$

by stacking them, it be capital $Z^{[1]}$



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

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

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

same dim

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

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

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

<vectorization>

$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$ by python broadcasting

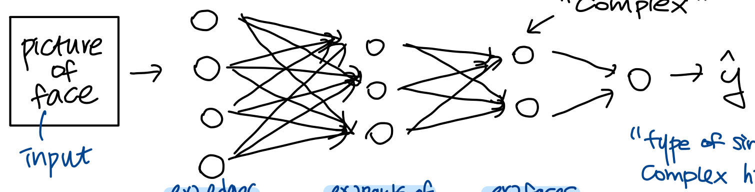
$(n^{[1]}, m)$ $(n^{[1]}, n^{[0]})(n^{[0]}, m) + (n^{[1]},1) \Rightarrow (n^{[1]}, m)$

Why Deep Representations?

Why Deep Networks Work Well?

<Intuition about deep representation>

Convolutional nets..



Small areas of an image

ex) edges first layer of Neural Network

ex) parts of faces

ex) faces

"type of simple to complex hierarchical representation or compositional representation"

→ small → large, simple → complex

Audio → low level audio waveform → phonemes C A T basic units of sound → words → Sentense/phrase

lower levels simpler features → deeper layers → then put together, the simpler things detected more complex things

Some research that Neural network and human brain where we believe on neuroscientists

→ human brain also starts on detecting simple things like edges, and put them together to form more and more complex objects

→ has served as a loose form of inspiration for some deep learning as well.

→ to show advantage of deep neural network.

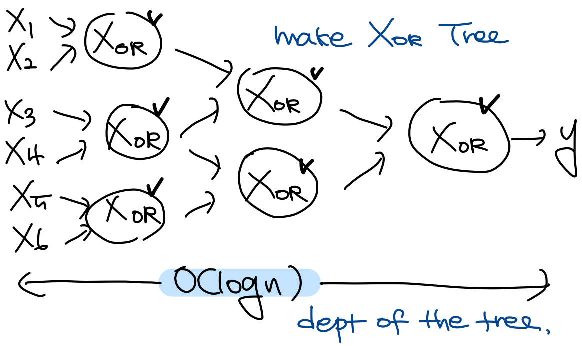
Circuit theory and deep learning

Informally: There are functions you can compute with a "small" L-layer deep neural network that shallower networks require exponentially more hidden units to compute.

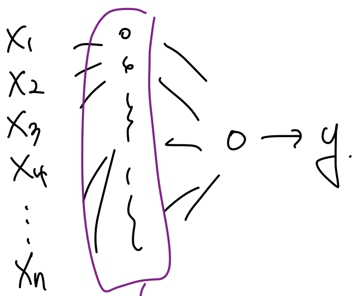
small
= hidden units are less.

example) computing X_1 XOR X_2 XOR X_3 XOR ... XOR X_n

Number of Nodes
= Number of gates
on these network.



If
not allowed to make neural network with few hidden layers,
O(log n) is hidden layers.

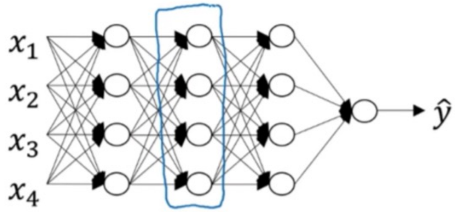


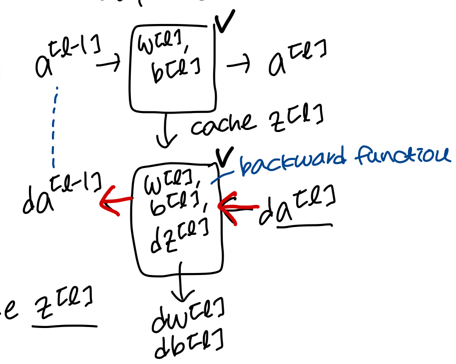
∴ $O(2^n)$
$\sim 2^{n-1}$
→ exponentially large
∵ you need to exhaustively enumerate all 2^n possible configurations of the input bits

cf)
Prof: Let number of hidden units in hyperparameter or parameter.

Building Blocks of a Deep Neural Network.

Forward and backward functions





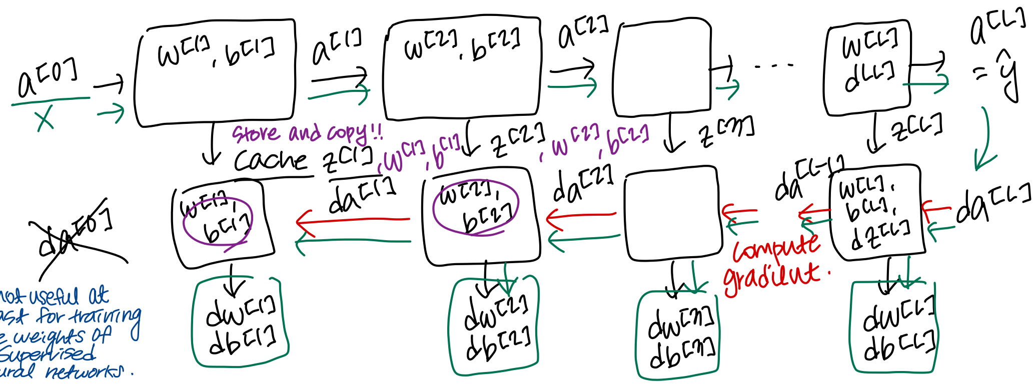
Layer l: $w^{[l]}, b^{[l]}$

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

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

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

→ Backward: Input $da^{[l]}$, cache$(z^{[l]})$ Output $da^{[l-1]}$, $dw^{[l]}$, $db^{[l]}$ output gradient descent for learning.



not useful at least for training the weights of supervised neural networks.

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

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

Forward and Backward Propagation.

<Forward Propagation>

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

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

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

$\Rightarrow$ <vectorization>

$$Z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$
$$A^{[l]} = g^{[l]}(Z^{[l]})$$

$X = A^{[0]} \rightarrow \square \rightarrow \square \rightarrow \square \rightarrow$ set left to right forward propagation.

<Backward Propagation>

Input $da^{[l]}$

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

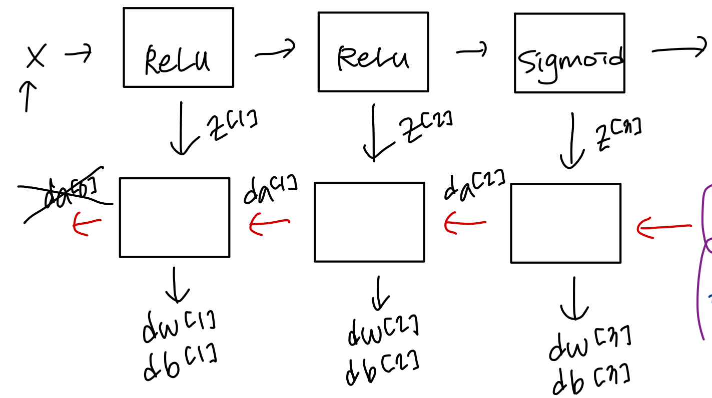
$$dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]})$$
$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$$
$$db^{[l]} = dz^{[l]}$$
$$da^{[l-1]} = w^{[l]T} \cdot dz^{[l]}$$
$$dz^{[l]} = w^{[l+1]T} dz^{[l+1]} * g^{[l]}(z^{[l]})$$

the only 4 formula to get backward propagation result.

$\Rightarrow$ <vectorization>

$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$
$$dW^{[l]} = \frac{1}{m} dZ^{[l]} \cdot A^{[l-1]T}$$
$$db^{[l]} = \frac{1}{m} \text{np.sum}(dZ^{[l]}, \text{axis}=1, \text{keepdims}=\text{True})$$
$$dA^{[l-1]} = W^{[l]T} \cdot dZ^{[l]}$$

<Summary>



$\searrow L(\hat{y}, y)$ 미분 값

$$da^{[l]} = -\frac{y}{a} + \frac{(1-y)}{(1-a)}$$

binary classification in logistic regression.

$dA^{[L]}$ in vectorization

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

implement of vectorization.

Parameter vs Hyperparameters.

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

Hyperparameters: Learning rate α — α will determine how your parameters evolve
- #iterations — parameter (that control parameters)
- #hidden layers L
- #hidden units $n^{[1]}, n^{[2]}, \ldots$
- Choice of activation function — ex) sigmoid, tanh, ReLu.. especially hidden layers

all of those things is what you need to tell to learn your algorithm

→ these parameters controls the ultimate parameters W & B.
∴ call hyperparameters. (to determine the final value of the parameter W & B)

Later: Momentum, mini batch size, regularizations, …

There are lots of settings of hyperparameters.

Applied deep learning is a very empirical process "그냥 해보렴"

Just try a lot of things and see how it works.

⊕ it could be possible to change hyperparameter later. (CPU, GPU..)

$\alpha = \cancel{0.01}$ try → 0.05 change my mind

to know the best value of hyperparameter at first is very difficult.

try many different values and go around that cycle.