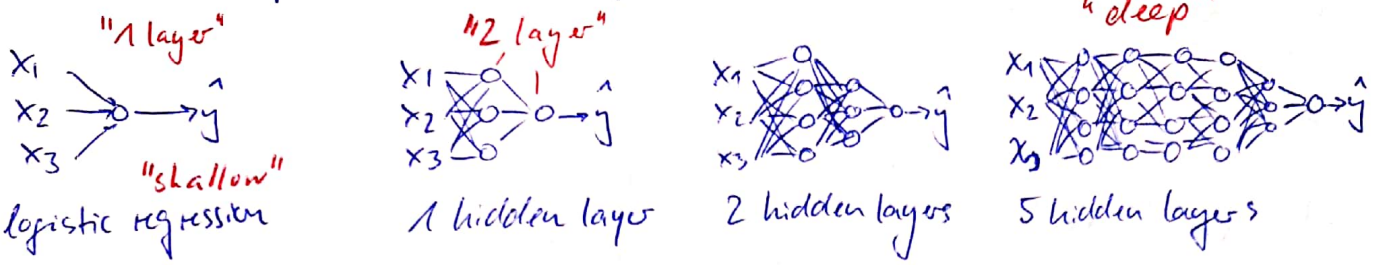
DL spec [Week 4] course 1

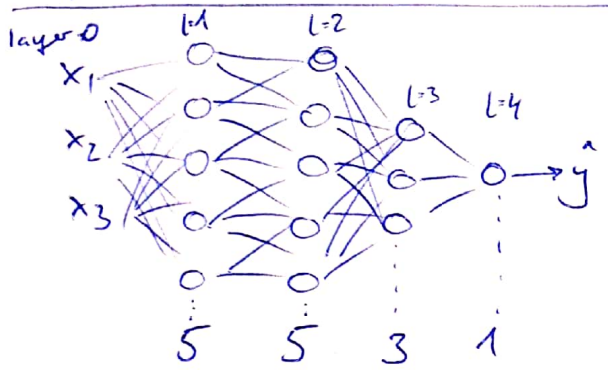
Deep L-layer neural network

What is a deep neural network?



logistic regression | 1 hidden layer | 2 hidden layers | 5 hidden layers

Deep neural network notation



$L = 4$ (# layers)

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

$n^{[1]} = 5$
$n^{[2]} = 5$
$n^{[3]} = 3$
$n^{[4]} = 1 = n^{[L]}$
$n^{[0]} = n_x = 3$

$X = a^{[0]}$ input

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

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

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

$b^{[l]}$

Forward Propagation in a deep NN

x (single example) $= a^{[0]}$

1st layer activations:
$z^{[1]} = w^{[1]}x + b^{[1]}$
$a^{[1]} = g^{[1]}(z^{[1]})$

2nd layer:
$z^{[2]} = w^{[2]}a^{[1]} + b^{[2]}$
$a^{[2]} = g^{[2]}(z^{[2]})$

...

$z^{[4]} = w^{[4]} \cdot a^{[3]} + b^{[4]}$
$a^{[4]} = g^{[4]}(z^{[4]})$

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

Vectorized:

$$Z^{[1]} = w^{[1]}X + b^{[1]} \quad (X = A^{[0]})$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$

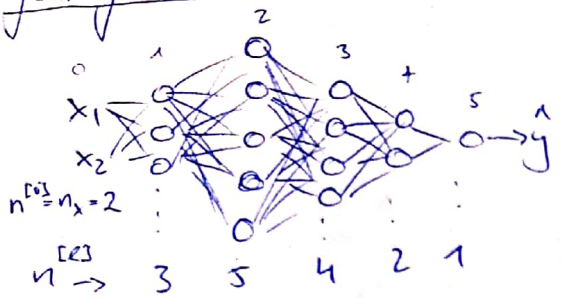
$Z^{[l]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$

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

↳ still need explicit for loop for $l = 1 \ldots 4$

Getting the matrix dimensions right

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



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

$(3,1) \quad (3,2) \quad (2,1) \quad (3,1)$

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

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

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

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

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

$(5,1) = (5,3) \quad (3,1) \quad (5,1)$

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

$w^{[3]} = (4,5)$ or $(n^{[3]}, n^{[2]})$

$w^{[4]} = (2,4) \ldots$

$w^{[5]} = (1,2) \ldots$

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

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

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

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

Vectorized Implementation

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

$\begin{bmatrix} z^{[1](1)}, z^{[1](2)}, \ldots, z^{[1](m)} \end{bmatrix}$ $(n^{[1]}, m)$

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

m by broadcasting!

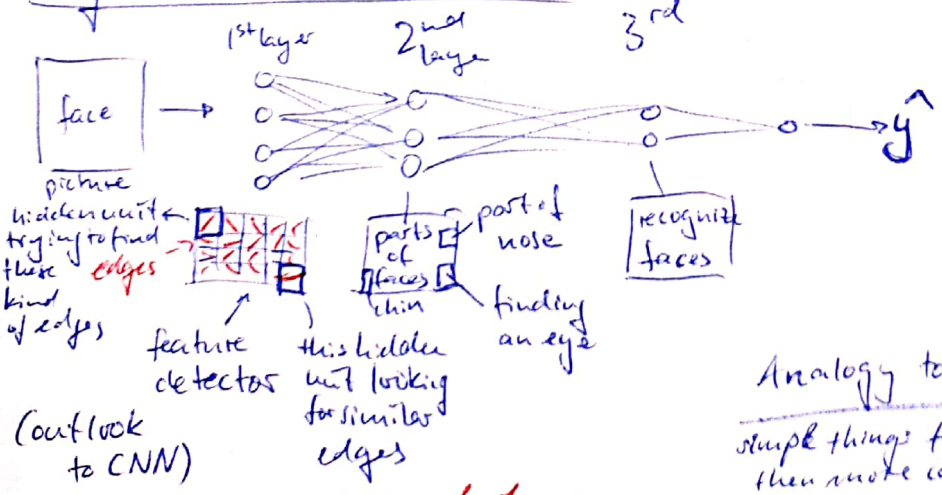
$z^{[l]}, a^{[l]} : (n^{[l]}, 1)$

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

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

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

Why deep representations?



(Outlook to CNN)

complexity of functions rises with depth

e.g. speech recognition (RNN)

from Audio clip → 1st layer low level acoustic wave form features (tone going up or down, white noise?) → 2nd layer phonemes, basic units of sound → 3rd layer words → 4th layer phrases

Analogy to brain

simple things first, then more complex things

DLspec | Week 4 | course I

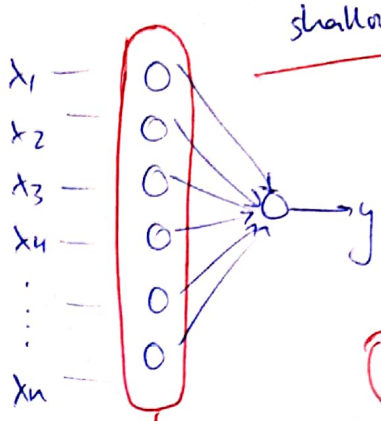
Circuit theory and deep learning

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

example: x_1 XOR x_2 XOR x_3 XOR .. XOR x_n → NN order of $\log(n)$

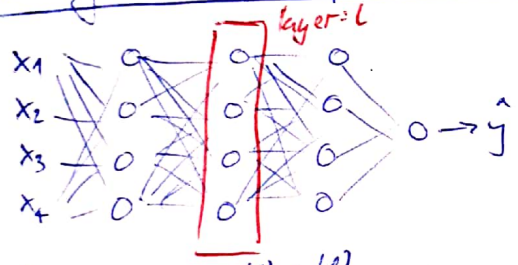
deep NN

shallow NN



# of hidden units will be exponentially large 2^{n-1} - possible configurations

Building Blocks of Deep Neural Networks



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

Forward Propagation $a^{[l-1]}$, output $a^{[l]}$

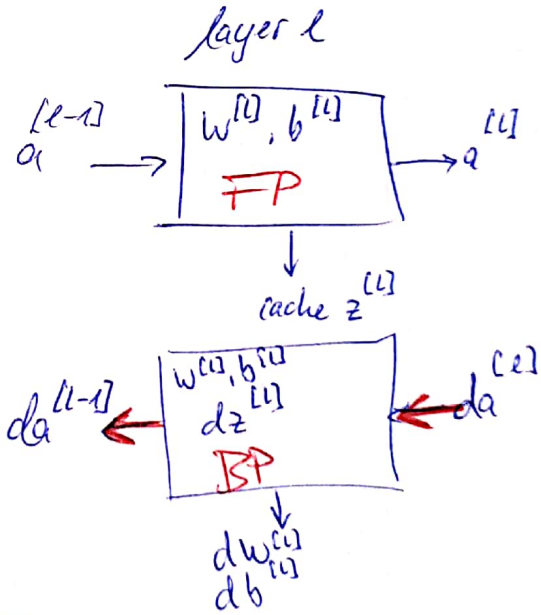
$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ cache $z^{[l]}$

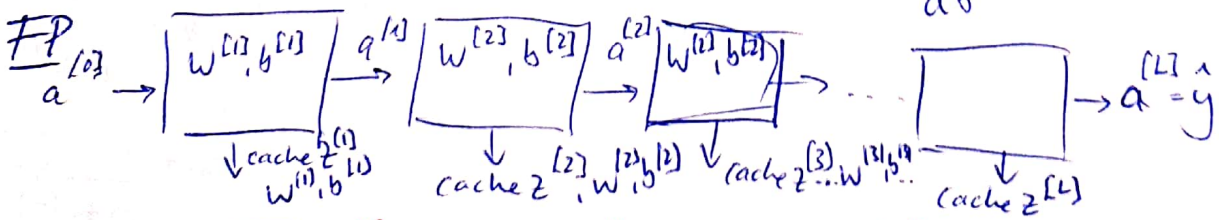
$a^{[l]} = g^{[l]}(z^{[l]})$

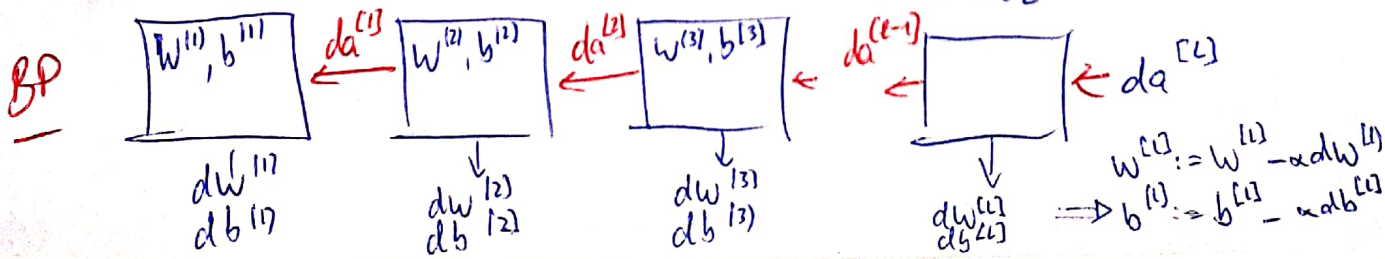
Backward Propagation:

input $da^{[l]}$, output $da^{[l-1]}$

cache($z^{[l]}$) $dW^{[l]}$ $db^{[l]}$







$W^{[l]} := W^{[l]} - \alpha dW^{[l]}$

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

Forward and Backward Propagation

FP

→ Input $a^{[l-1]}$

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

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

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

Vectorized FP

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

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

BP:

→ Input $da^{[l]}$

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

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

$dW^{[l]} = dz^{[l]} \cdot a^{[l-1]T}$ (transpose)

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

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

Vectorized BP

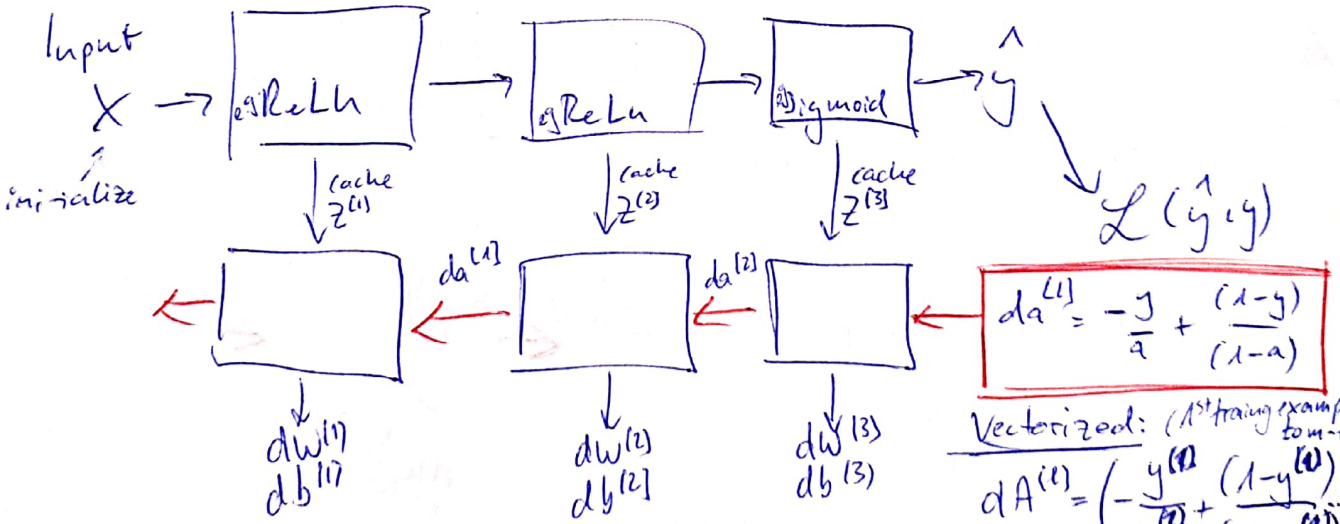
$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$

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

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

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

Summary



Vectorized: (1^{st} training example to m-th)

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

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

DL spec | Week 4 | cont'd I

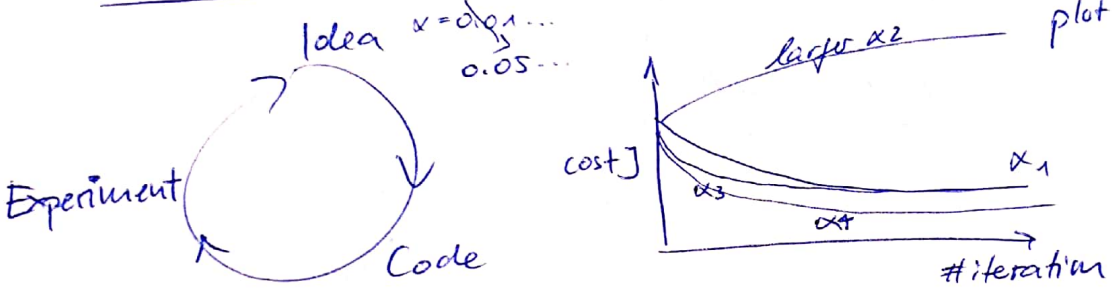
Parameters vs. Hyperparameters

- Parameters: $w^{[1]}, b^{[1]}, \dots$
- Hyperparameters: learning rate α
  - # iterations
  - NN architecture { # hidden layers L; # hidden units $n^{[1]}, n^{[2]}, \dots$; Choice of activation function }

These control the ultimate parameters $w^{[l]}$ and $b^{[l]}$

Later: Momentum, mini-batch size, regularizations...

Applied deep learning is a very empirical process

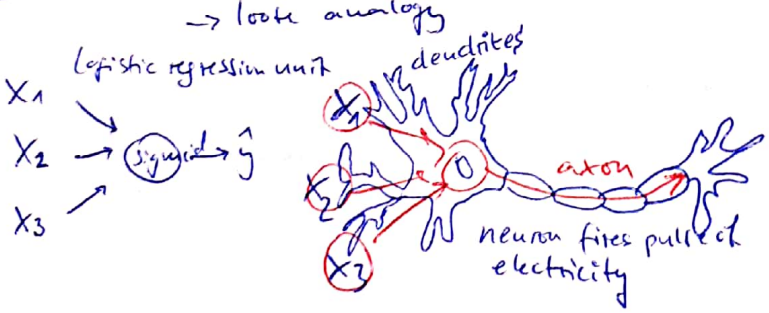


Vision, Speech, NLP, Ads, Web Search, Product recommendations

structured ~~data~~

What does DL have to do with the human brain?

→ loose analogy



$x \longrightarrow y$ mapping

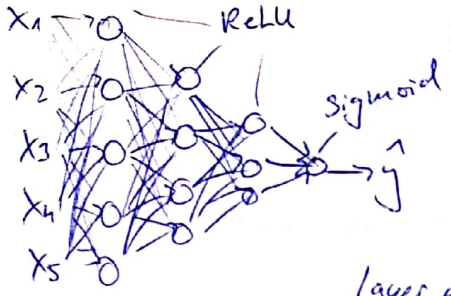
Programming Assignment week 4 – Part 1

Deep Neural Network Architecture

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

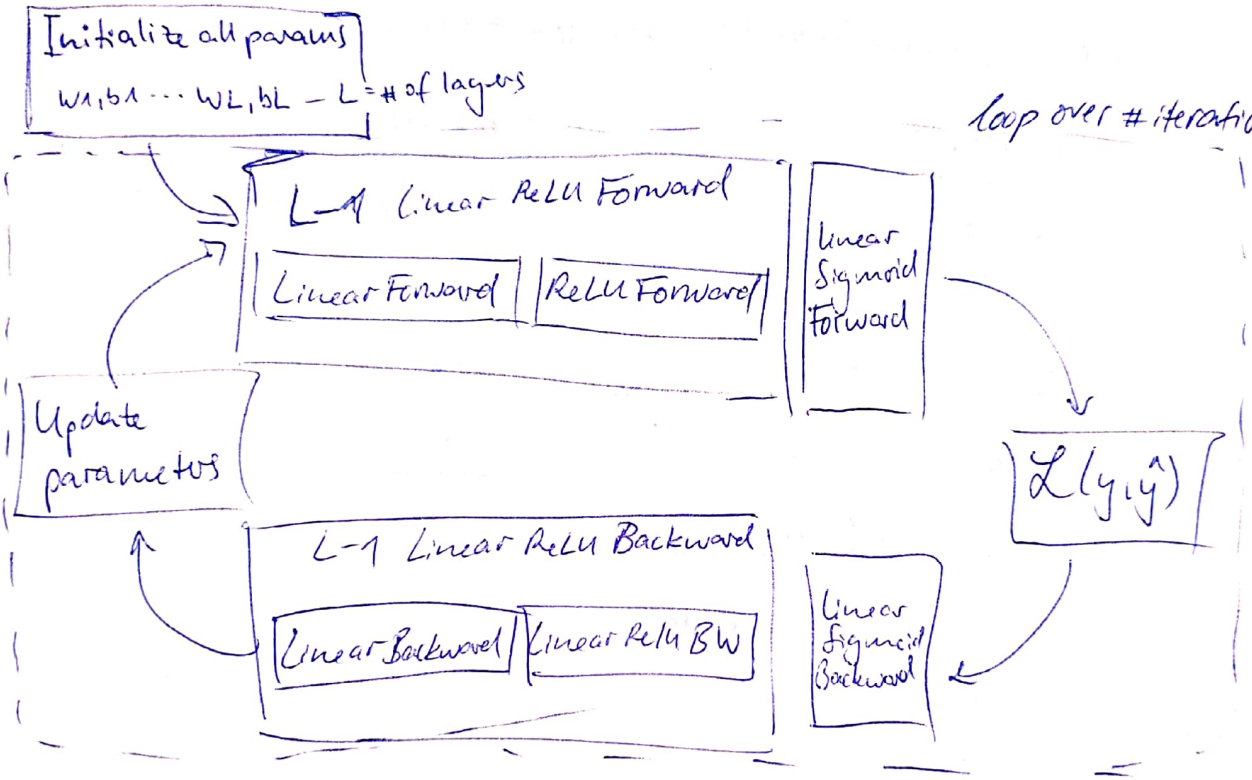
(1,3) (3,2)

[, ,] [; ; ;] + [ ]



layer_dims = [5,4,3]

L0 L1 L2 L3



- initialize_parameters (n_x, n_h, n_y) → dict. parameters (W1 b1 W2 b2)
- initialize_parameters_deep (layer_dims) → dict. parameters layer_dims = [5,4,3] ...
  L = len (layer_dims)
- linear_forward (A, W, b) → Z, cache (A, W, b)
- linear_activation_forward (A_prev, W, b, activation) → A, cache
  activation: string "sigmoid" "relu"; cache: linear cache, activation cache
- L_model_forward (X, parameters) → $A^{[L]}$, caches

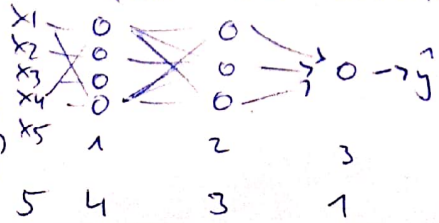
X → [relu: Z → a(Z), l → L-1] → [linear sigmoid]

b1 W1 X b2 W2 W3 b3

(4,1) (4,5) (5,4) (3,1) (3,4) (1,3) (1,1)

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

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



- compute_cost ($A^{[L]}$, Y) → cost
- linear_backward (dZ, cache) → dA_prev, dW, db
- linear_activation_backward (dA, cache, activ.) → dA_prev, dW, db
- L_model_backward
- update_parameters