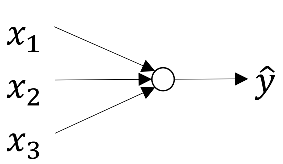
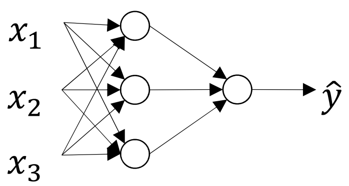
Deep Neural Network
- Multi-Layered Percepron (MLP)

Most of this material is from Prof. Andrew Ng'and Chang's slides.

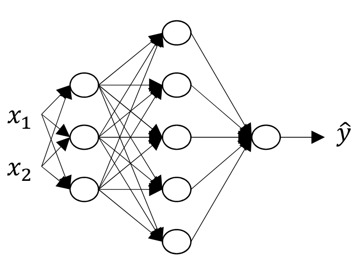
What is a deep neural network?



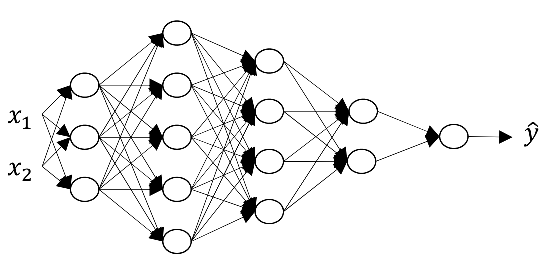
logistic regression



1 hidden layer



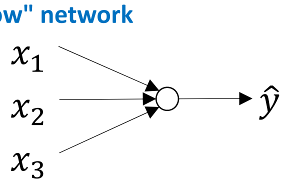
2 hidden layers



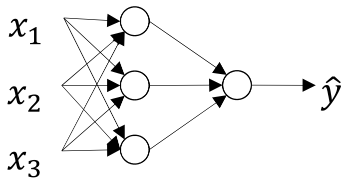
4 hidden layers

What is a deep neural network?

"shallow" network

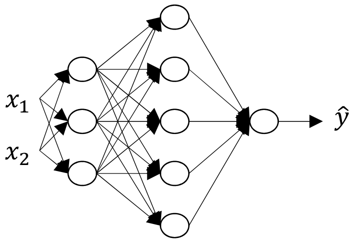


logistic regression

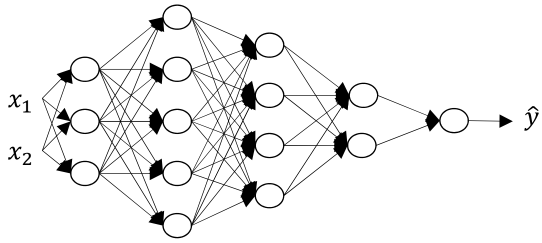


1 hidden layer

"deep" network

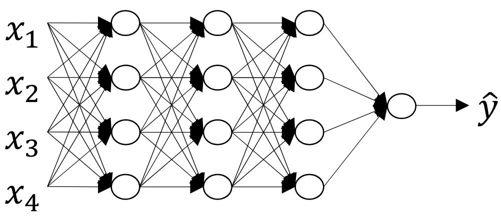


2 hidden layers



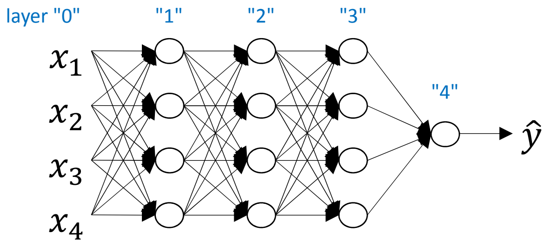
4 hidden layers

Deep neural network notation



- $L = 4$: the number of layers
- $n^{[l]}$: the number of units in layer l

Deep neural network notation

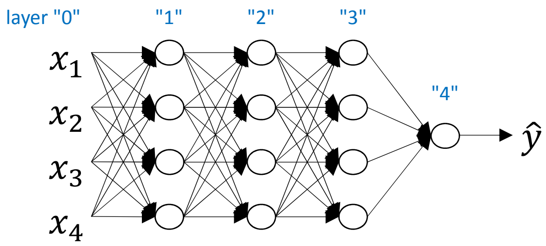


– $L = 4$: the number of layers

– $n^{[l]}$: the number of units in layer l

$$\begin{cases} n^{[0]} = n^{[1]} = n^{[2]} = n^{[3]} = 4 \\ n^{[4]} = 1 \end{cases}$$

Deep neural network notation



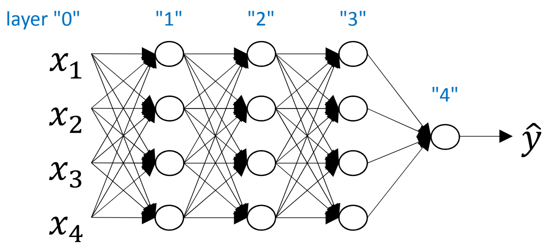
– $L = 4$: the number of layers

– $n^{[l]}$: the number of units in layer l

$$\begin{cases} n^{[0]} = n^{[1]} = n^{[2]} = n^{[3]} = 4 \\ n^{[4]} = 1 \end{cases}$$

– $a^{[l]}$: activations in layer l

Deep neural network notation



- $L = 4$: the number of layers
- $n^{[l]}$: the number of units in layer l
- $a^{[l]}$: activations in layer l

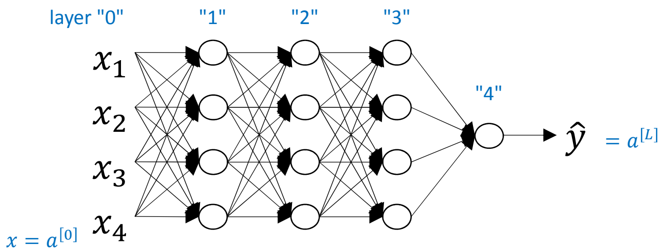
$n^{[0]} = n^{[1]} = n^{[2]} = n^{[3]} = 4$

$n^{[4]} = 1$

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

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

Deep neural network notation



- $L = 4$: the number of layers
- $n^{[l]}$: the number of units in layer l
- $a^{[l]}$: activations in layer l

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

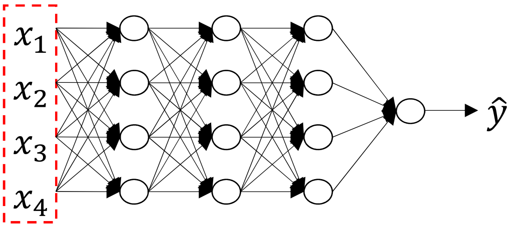
$n^{[4]} = 1$

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

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

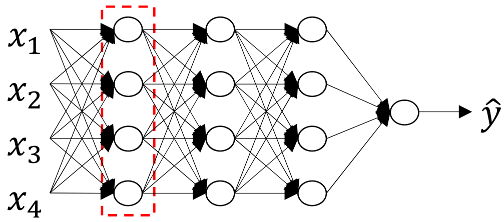
- x : input of the network
- $\hat{y}$: prediction of the network

Forward propagation in a deep network



$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Forward propagation in a deep network

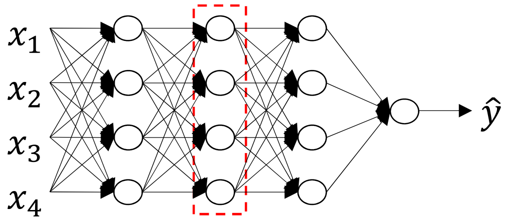


$$z^{[1]} = W^{[1]}x + b^{[1]}$$

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

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]})$$

Forward propagation in a deep network

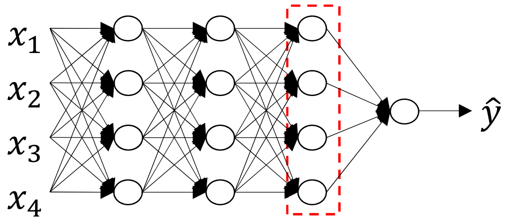


$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$

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

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]})$$

Forward propagation in a deep network

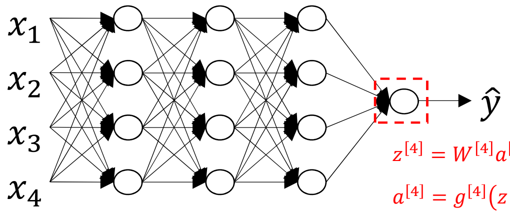


$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$

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

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]})$$
$$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$$
$$a^{[3]} = g^{[3]}(z^{[3]})$$

Forward propagation in a deep network

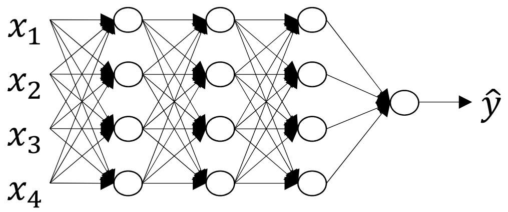


$z^{[4]} = W^{[4]}a^{[3]} + b^{[4]}$

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

$z^{[1]} = W^{[1]}x + b^{[1]}$
$a^{[1]} = g^{[1]}(z^{[1]})$
$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$
$a^{[2]} = g^{[2]}(z^{[2]})$
$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$
$a^{[3]} = g^{[3]}(z^{[3]})$
$z^{[4]} = W^{[4]}a^{[3]} + b^{[4]}$
$a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

Forward propagation in a deep network



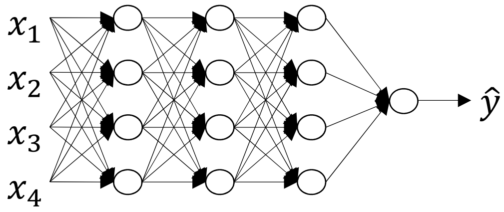
general
forward propagation
equations

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

$x = a^{[0]}$

$z^{[1]} = W^{[1]}x + b^{[1]}$
$a^{[1]} = g^{[1]}(z^{[1]})$
$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$
$a^{[2]} = g^{[2]}(z^{[2]})$
$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$
$a^{[3]} = g^{[3]}(z^{[3]})$
$z^{[4]} = W^{[4]}a^{[3]} + b^{[4]}$
$a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

Forward propagation in a deep network



general
forward propagation
equations

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

$x = a^{[0]}$

$$
\begin{aligned}
z^{[1]} &= W^{[1]}x + b^{[1]} \\
a^{[1]} &= g^{[1]}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= g^{[2]}(z^{[2]}) \\
z^{[3]} &= W^{[3]}a^{[2]} + b^{[3]} \\
a^{[3]} &= g^{[3]}(z^{[3]}) \\
z^{[4]} &= W^{[4]}a^{[3]} + b^{[4]} \\
a^{[4]} &= g^{[4]}(z^{[4]}) = \hat{y}
\end{aligned}
$$

vectorize for all training samples

$X = A^{[0]}$

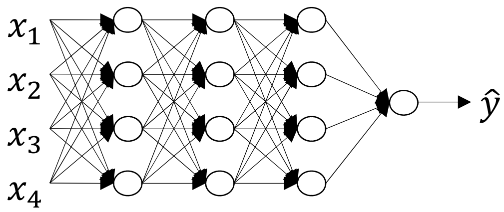
$$
\begin{aligned}
Z^{[1]} &= W^{[1]}X + b^{[1]} \\
A^{[1]} &= g^{[1]}(Z^{[1]}) \\
Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(Z^{[2]}) \\
Z^{[3]} &= W^{[3]}A^{[2]} + b^{[3]} \\
A^{[3]} &= g^{[3]}(Z^{[3]}) \\
Z^{[4]} &= W^{[4]}A^{[3]} + b^{[4]} \\
A^{[4]} &= g^{[4]}(Z^{[4]}) = \hat{Y}
\end{aligned}
$$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

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

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

Forward propagation in a deep network



general
forward propagation
equations

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

$x = a^{[0]}$

$$
\begin{aligned}
z^{[1]} &= W^{[1]}x + b^{[1]} \\
a^{[1]} &= g^{[1]}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= g^{[2]}(z^{[2]}) \\
z^{[3]} &= W^{[3]}a^{[2]} + b^{[3]} \\
a^{[3]} &= g^{[3]}(z^{[3]}) \\
z^{[4]} &= W^{[4]}a^{[3]} + b^{[4]} \\
a^{[4]} &= g^{[4]}(z^{[4]}) = \hat{y}
\end{aligned}
$$

vectorize for all training samples

$X = A^{[0]}$

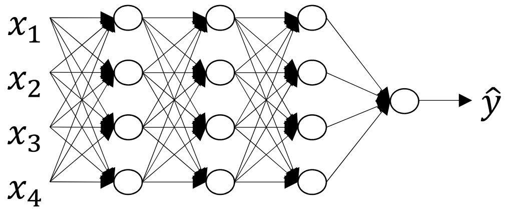
$$
\begin{aligned}
Z^{[1]} &= W^{[1]}X + b^{[1]} \\
A^{[1]} &= g^{[1]}(Z^{[1]}) \\
Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(Z^{[2]}) \\
Z^{[3]} &= W^{[3]}A^{[2]} + b^{[3]} \\
A^{[3]} &= g^{[3]}(Z^{[3]}) \\
Z^{[4]} &= W^{[4]}A^{[3]} + b^{[4]} \\
A^{[4]} &= g^{[4]}(Z^{[4]}) = \hat{Y}
\end{aligned}
$$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

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

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

Forward propagation in a deep network



general
forward propagation
equations

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

$x = a^{[0]}$

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]})$$
$$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$$
$$a^{[3]} = g^{[3]}(z^{[3]})$$
$$z^{[4]} = W^{[4]}a^{[3]} + b^{[4]}$$
$$a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$$

vectorize for all training samples

$X = A^{[0]}$

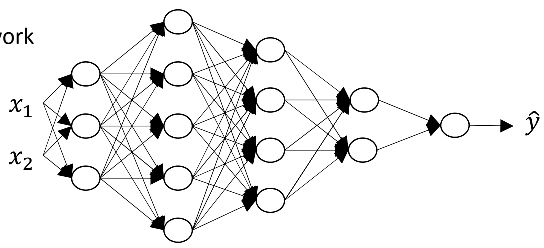
$$\left.\begin{aligned}
Z^{[1]} &= W^{[1]}X + b^{[1]}\\
A^{[1]} &= g^{[1]}(Z^{[1]})\\
Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]}\\
A^{[2]} &= g^{[2]}(Z^{[2]})\\
Z^{[3]} &= W^{[3]}A^{[2]} + b^{[3]}\\
A^{[3]} &= g^{[3]}(Z^{[3]})\\
Z^{[4]} &= W^{[4]}A^{[3]} + b^{[4]}\\
A^{[4]} &= g^{[4]}(Z^{[4]}) = \hat{Y}
\end{aligned}\right\}$$

for $l = 1 \ldots 4$

this "for" loop
cannot be removed

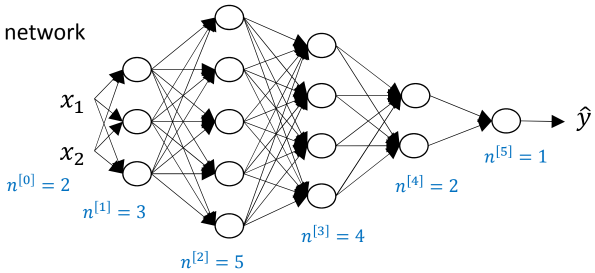
Parameters $W^{[l]}$ and $b^{[l]}$

5-layer network



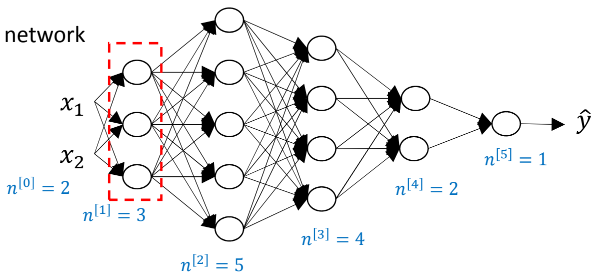
Parameters $W^{[l]}$ and $b^{[l]}$

5-layer network



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

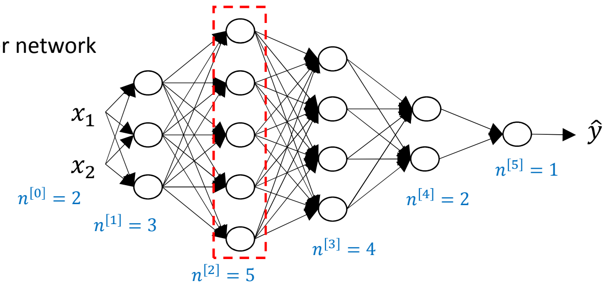
5-layer network



$$\begin{array}{ccccccc} z^{[1]} & = & W^{[1]} & \cdot & x & + & b^{[1]} \\ (3,1) & & (3,2) & & (2,1) & & (3,1) \\ (n^{[1]},1) & & (n^{[1]},n^{[0]}) & & (n^{[0]},1) & & (n^{[1]},1) \end{array}$$

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

5-layer network



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

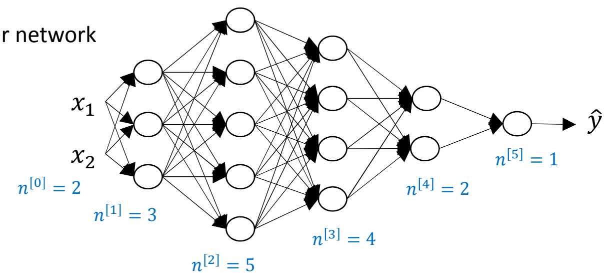
$z^{[1]}$	$W^{[1]}$	x	$b^{[1]}$
(3,1)	(3,2)	(2,1)	(3,1)
$(n^{[1]}, 1)$	$(n^{[1]}, n^{[0]})$	$(n^{[0]}, 1)$	$(n^{[1]}, 1)$

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

$z^{[2]}$	$W^{[2]}$	$a^{[1]}$	$b^{[2]}$
(5,1)	(5,3)	(3,1)	(5,1)
$(n^{[2]}, 1)$	$(n^{[2]}, n^{[1]})$	$(n^{[1]}, 1)$	$(n^{[2]}, 1)$

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

5-layer network



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

$z^{[1]}$	$W^{[1]}$	x	$b^{[1]}$
(3,1)	(3,2)	(2,1)	(3,1)
$(n^{[1]}, 1)$	$(n^{[1]}, n^{[0]})$	$(n^{[0]}, 1)$	$(n^{[1]}, 1)$

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

$z^{[2]}$	$W^{[2]}$	$a^{[1]}$	$b^{[2]}$
(5,1)	(5,3)	(3,1)	(5,1)
$(n^{[2]}, 1)$	$(n^{[2]}, n^{[1]})$	$(n^{[1]}, 1)$	$(n^{[2]}, 1)$

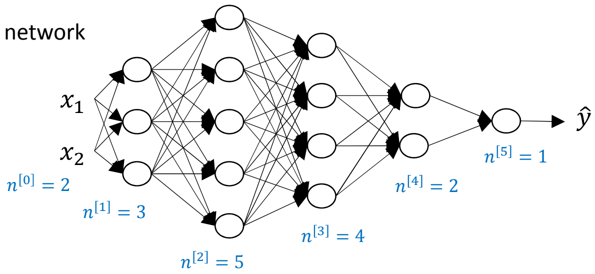
<general formula for checking dimensions>

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

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

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

5-layer network



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

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

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

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

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

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

<general formula for checking dimensions>

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

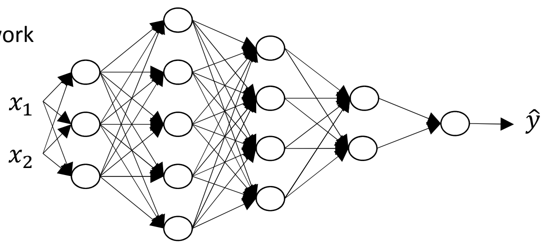
$b^{[l]} : (n^{[l]}, 1)$

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

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

Vectorized implementation

5-layer network



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

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

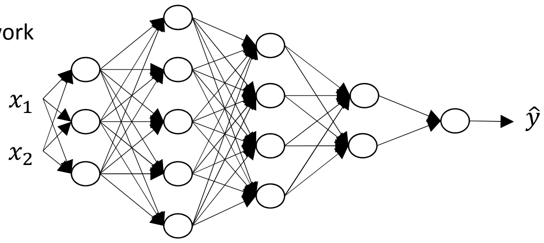
$$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix} \qquad X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

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

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

Vectorized implementation

5-layer network



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

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

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

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

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

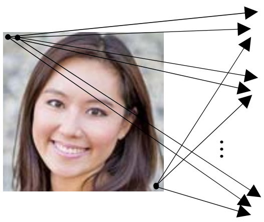
$z^{[l]}, a^{[l]}$: $(n^{[l]}, 1)$

$\Downarrow$

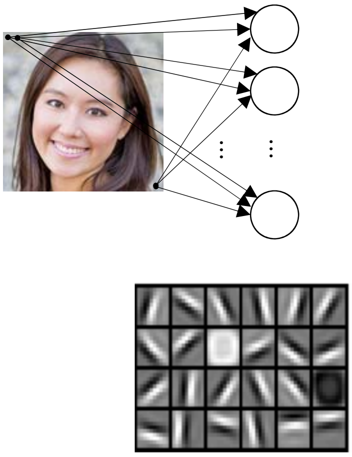
$Z^{[l]}, A^{[l]}$: $(n^{[l]}, m)$

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

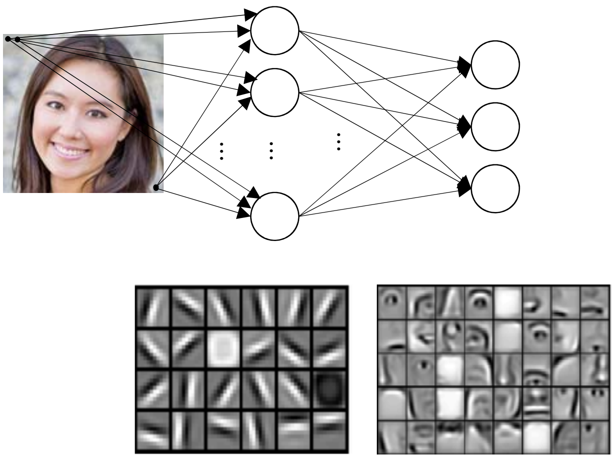
Intuition about deep representation



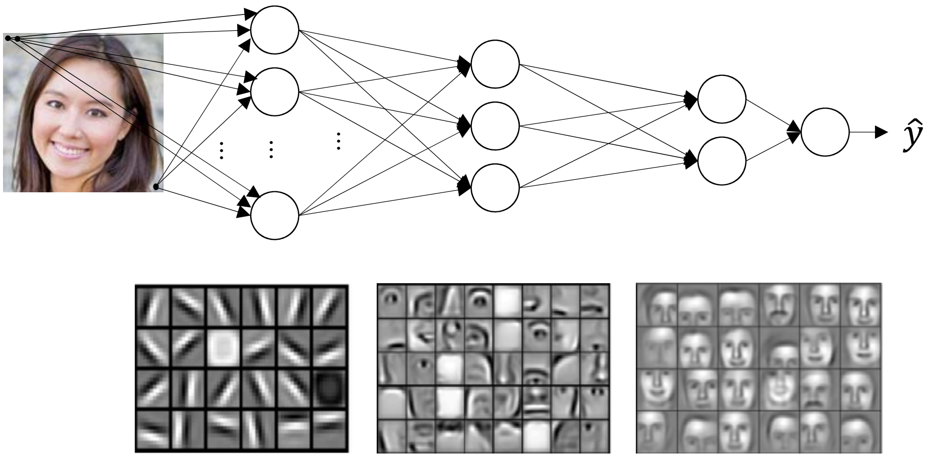
Intuition about deep representation



Intuition about deep representation



Intuition about deep representation



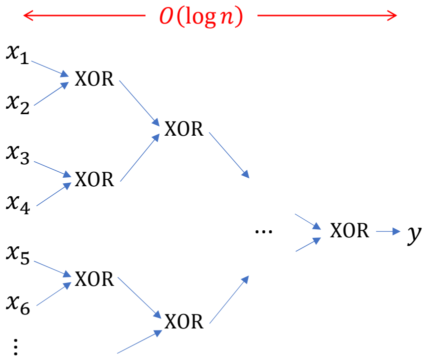
Circuit theory and deep learning

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

Circuit theory and deep learning

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

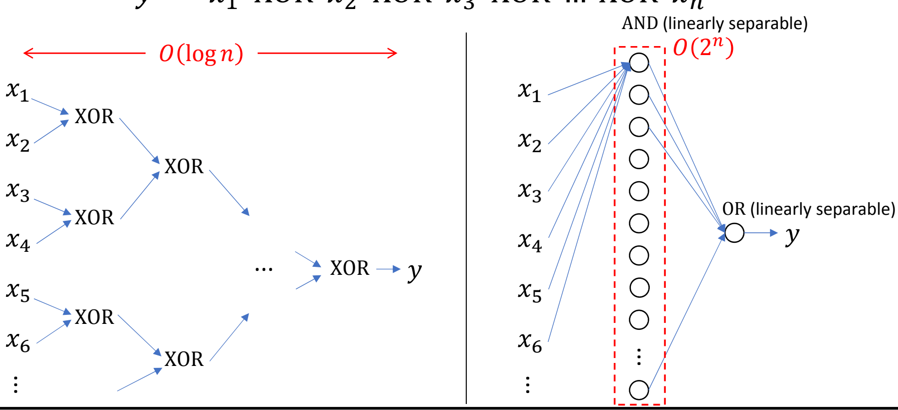
$$y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \text{ XOR } \ldots \text{ XOR } x_n$$



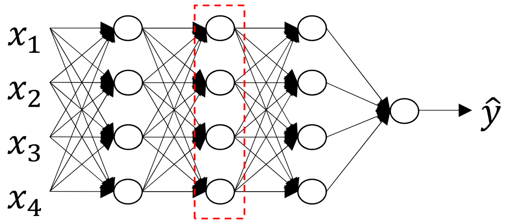
Circuit theory and deep learning

- Informally, there are funcitons you can compute with a "small" L-layer deep neural network while shallower networks require exponentially more hidden units to compute.

$$y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \text{ XOR } \ldots \text{ XOR } x_n$$



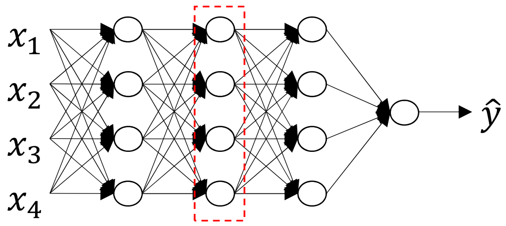
Forward and backward functions



layer l

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

Forward and backward functions



layer l

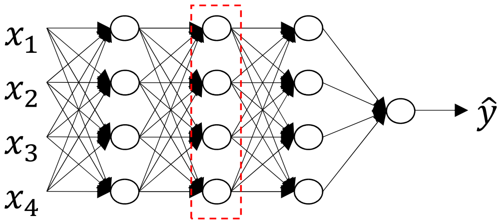
parameters: $W^{[l]}$, $b^{[l]}$

Forward: input $a^{[l-1]}$, output(caching) $a^{[l]}$, $z^{[l]}$

$z^{[l]} \coloneqq W^{[l]} a^{[l-1]} + b^{[l]}$

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

Forward and backward functions



layer l

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

Forward: input $a^{[l-1]}$, output(caching) $a^{[l]}$, $z^{[l]}$

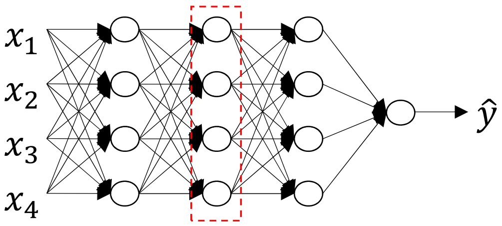
$z^{[l]} \coloneqq W^{[l]}a^{[l-1]} + b^{[l]}$

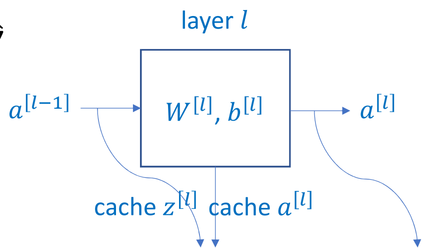
$a^{[l]} \coloneqq g^{[l]}(z^{[l]})$

Backward: input $da^{[l]}$, $a^{[l-1]}$(cached), $z^{[l]}$(cached)

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

Forward and backward functions





layer l

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

Forward: input $a^{[l-1]}$, output(caching) $a^{[l]}$, $z^{[l]}$

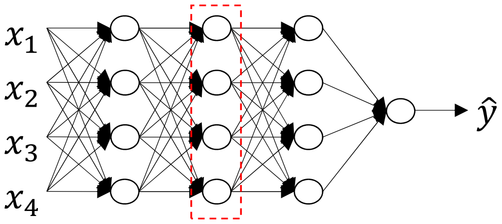
$z^{[l]} \coloneqq W^{[l]}a^{[l-1]} + b^{[l]}$

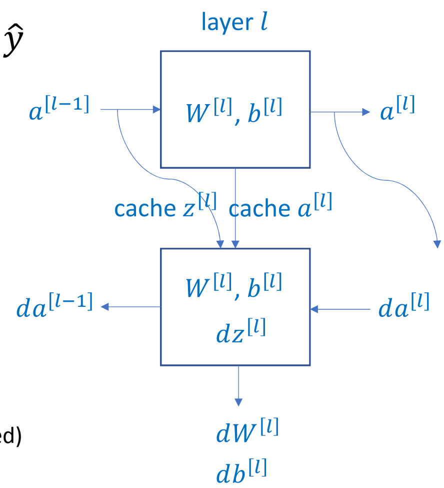
$a^{[l]} \coloneqq g^{[l]}(z^{[l]})$

Backward: input $da^{[l]}$, $a^{[l-1]}$(cached), $z^{[l]}$(cached)

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

Forward and backward functions





layer l

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

Forward: input $a^{[l-1]}$, output(caching) $a^{[l]}, z^{[l]}$

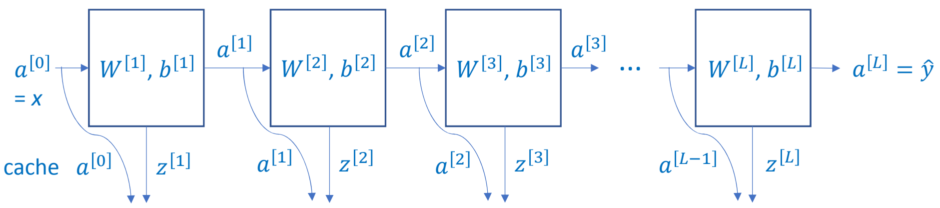
$z^{[l]} := W^{[l]}a^{[l-1]} + b^{[l]}$

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

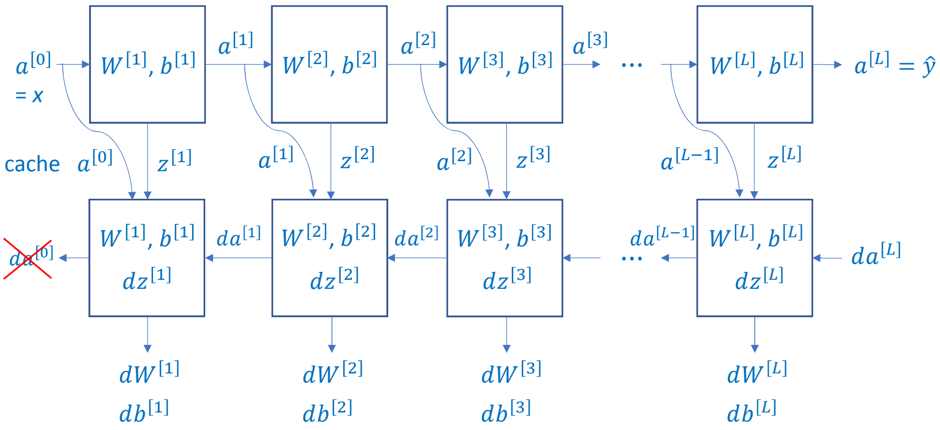
Backward: input $da^{[l]}$, $a^{[l-1]}$(cached), $z^{[l]}$(cached)

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

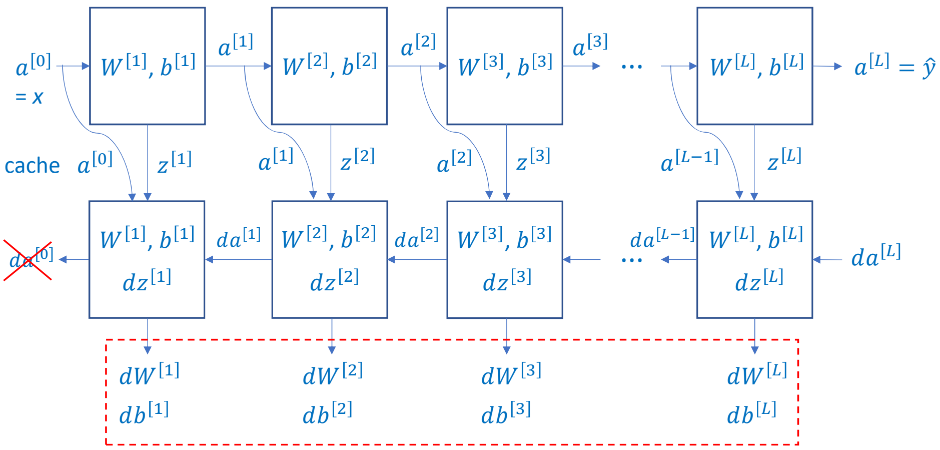
Forward and backward functions



Forward and backward functions



Forward and backward functions



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

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

Forward propagation for layer l

- Input $a^{[l-1]}$
- Output(caching) $a^{[l]}$, $z^{[l]}$

Forward propagation for layer l

- Input $a^{[l-1]}$
- Output(caching) $a^{[l]}$, $z^{[l]}$

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

$$\xrightarrow{\text{to matrix}}$$

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

Backward propagation for layer l

- Input $da^{[l]}$, $a^{[l-1]}$(cached), $z^{[l]}$(cached)
- Output $da^{[l-1]}$, $dW^{[l]}$, $db^{[l]}$

Backward propagation for layer l

- Input $da^{[l]}$, $a^{[l-1]}$(cached), $z^{[l]}$(cached)
- Output $da^{[l-1]}$, $dW^{[l]}$, $db^{[l]}$

element-wise product

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

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

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

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

Backward propagation for layer l

- Input $da^{[l]}$, $a^{[l-1]}$(cached), $z^{[l]}$(cached)
- Output $da^{[l-1]}$, $dW^{[l]}$, $db^{[l]}$

element-wise product

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

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

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

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

to matrix

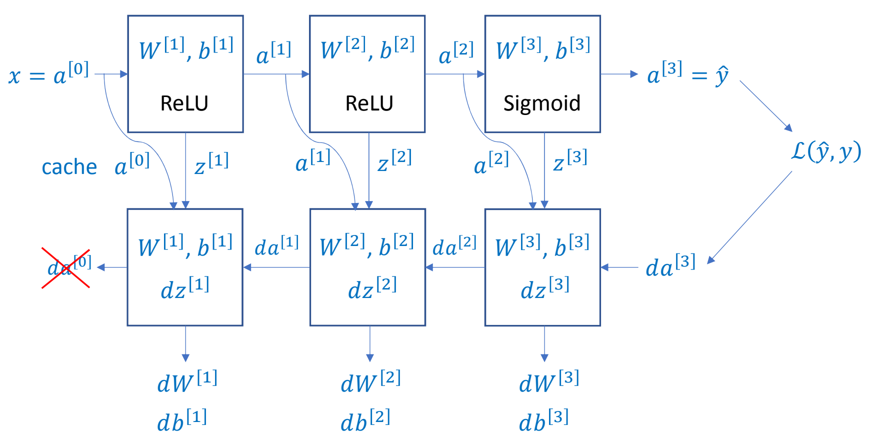
$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$

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

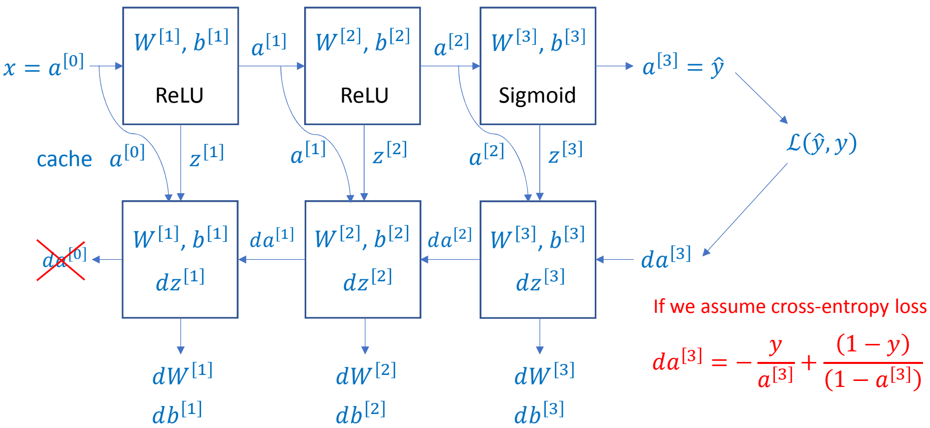
$$db^{[l]} = \frac{1}{m} dZ^{[l]} I$$

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

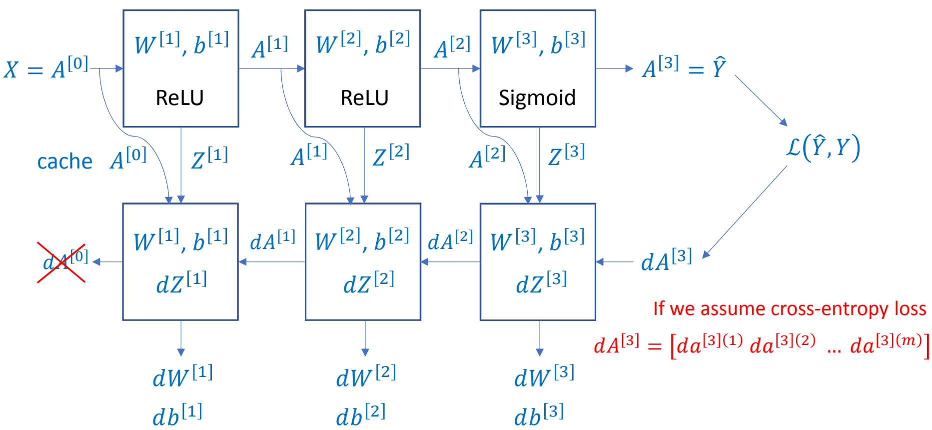
Summary of Backpropagation Learning



Summary of Backpropagation Learning



Summary of Backpropagation Learning



What are hyperparameters?

- To make our neural network effective, we need to organize not only network parameters but also hyperparameters
- Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}, \dots$

What are hyperparameters?

- To make our neural network effective, we need to organize not only network parameters but also hyperparameters
- Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}, \dots$
- Hyperparameters:
  - learning rate α : determine how your parameters evolve
  - # of iterations (of gradient descent)
  - # of hidden layers L
  - # of hidden units $n^{[1]}, n^{[2]}, n^{[3]}, \dots$
  - choice of activation functions (ReLU, tanh, sigmoid, ...)

parameters that control the ultimate parameters W and b

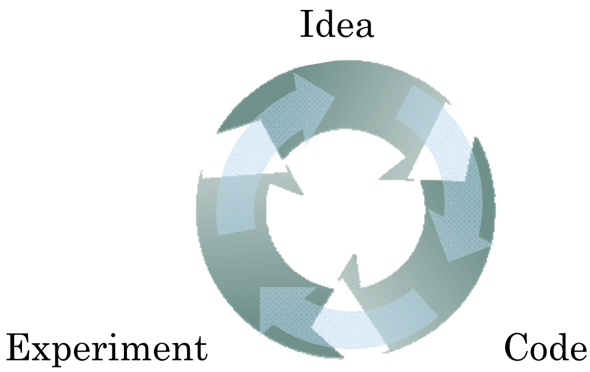
What are hyperparameters?

- To make our neural network effective, we need to organize not only network parameters but also hyperparameters
- Parameters: $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$, $W^{[3]}$, $b^{[3]}$, ...
- Hyperparameters:
  - learning rate α : determine how your parameters evolve
  - # of iterations (of gradient descent)
  - # of hidden layers L
  - # of hidden units $n^{[1]}$, $n^{[2]}$, $n^{[3]}$, ...
  - choice of activation functions (ReLU, tanh, sigmoid, ...)
  - momentum
  - mini-batch size
  - regularization parameters

parameters that control the ultimate parameters W and b

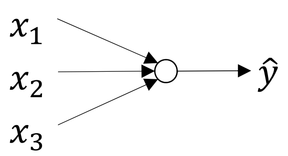
Applied deep learning is a very empirical process

- There are a lot of different hyperparameters
- When you're starting on the new application, you will find it very difficult to know in advance exactly what's the best value of the hyperparameters
- We just have to try out many different values and go around this cycle



What does this have to do with the brain?

- Why people keep making the analogy between deep learning and the human brain?
- There is a very loose analogy between logistic regression unit with a sigmoid activation function and a single neuron in the brain
- Neuron
  - receives electric signals (x_1, x_2, ...) from other neurons and then performs simple thresholded computation
  - if this neuron fires, it sends a pulse or electricity (y) down the axon to other neurons



$\approx$