

# Contents

<b>1</b>	<b>1. deep neural network</b>	<b>1</b>
1.1	1.1. deep L-layer neural network	1
1.2	1.2. forward propagation in a deep network	1
1.3	1.3. getting your matrix dimensions right	1
1.4	1.4. why deep representations?	1
1.5	1.5. building blocks of deep neural networks	1
1.6	1.6. forward and backward propagation	5
1.7	1.7. parameters vs hyperparameters	5
1.8	1.8. what does this have to do with the brain?	5
1.9	1.9. others	5

contents

- 1. deep neural network
- 1.1. deep L-layer neural network
- 1.2. forward propagation in a deep network
- 1.3. getting your matrix dimensions right
- 1.4. why deep representations?
- 1.5. building blocks of deep neural networks
- 1.6. forward and backward propagation
- 1.7. parameters vs hyperparameters
- 1.8. what does this have to do with the brain?
- 1.9. others

## 1 1. deep neural network

### 1.1 1.1. deep L-layer neural network

### 1.2 1.2. forward propagation in a deep network

$z^{[l](i)}$  表示第  $l$  层的第  $i$  个训练样本 (列向量),  $Z^{[l]}$  表示将第  $l$  层的这  $m$  个训练样本全部水平地放在一起形成的矩阵。以此 vectorization 的方法, 可以避免从  $1 \rightarrow m$  的这个 for-loop。

另外, 从第 1 层到第 4 层这个 for-loop 是无法避免的。

### 1.3 1.3. getting your matrix dimensions right

vectorized 之后,  $b^{[l]}$  仍然是  $(n^{[l]}, 1)$  维, 只是因为 broadcasting, 才复制  $m$  遍, 变成了  $(n^{[l]}, m)$  维。

### 1.4 1.4. why deep representations?

对于  $(x_1, x_2, \dots, x_n)$  的异或 (XOR) 操作, 如果用树型结构,  $n$  个叶子节点, 则树深度是  $\log_2 n + 1$  (深度为  $k$  的满二叉树的第  $i$  层上有  $2^{i-1}$  个结点, 总共至多有  $2^k - 1$  个结点), 即只需要  $O(\log_2 n)$  层的树就能完成。

而如果采用单隐层, 则需要  $2^{(n-1)}$  个节点

### 1.5 1.5. building blocks of deep neural networks

forward 时, 需要 cache  $Z^{[l]}$  以供 backward 使用。

# Deep neural network notation

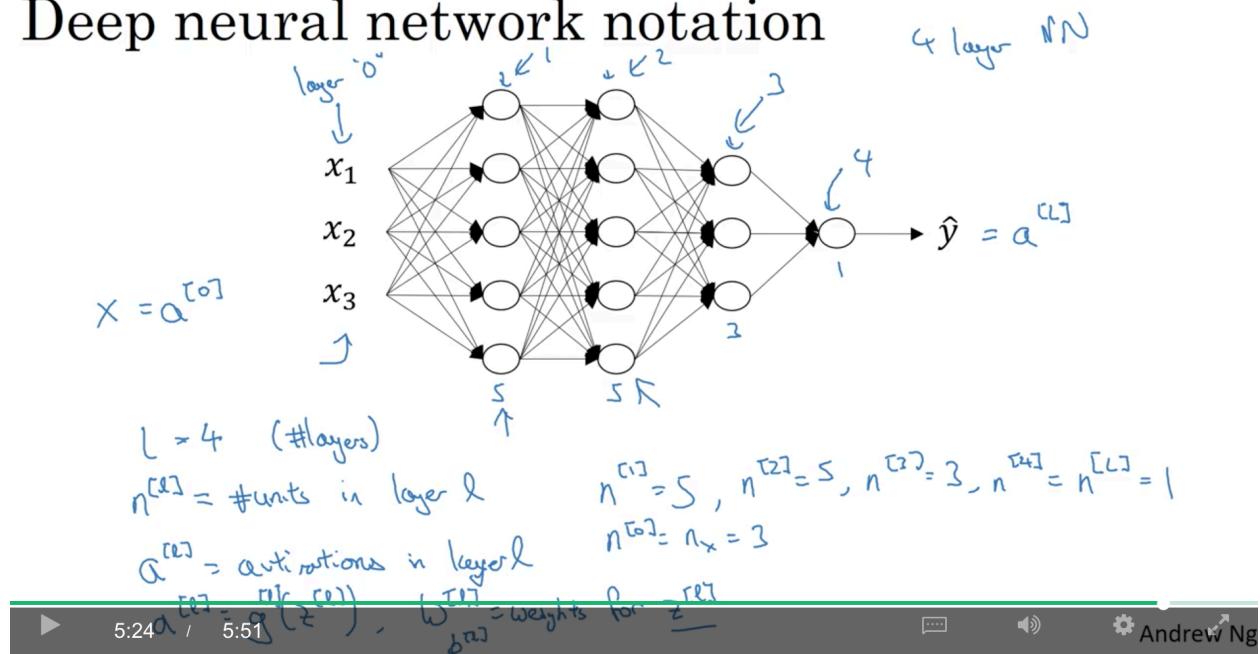


Figure 1: deep-neural-network-notation.png

## Forward propagation in a deep network

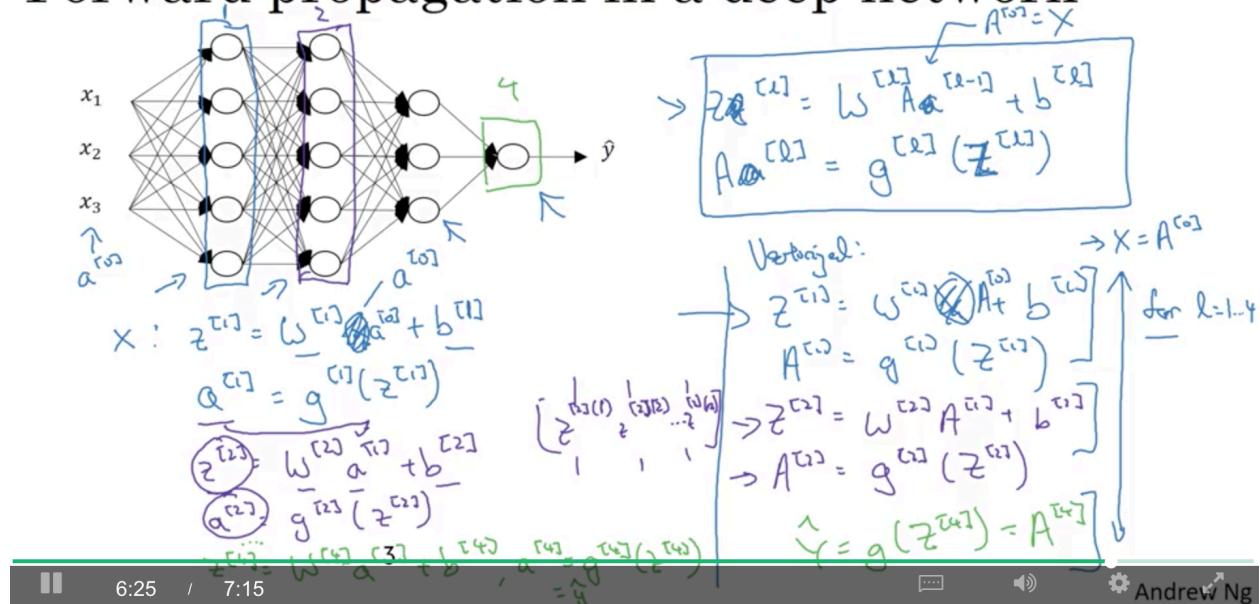


Figure 2: forward-propagation-in-a-deep-network.png

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

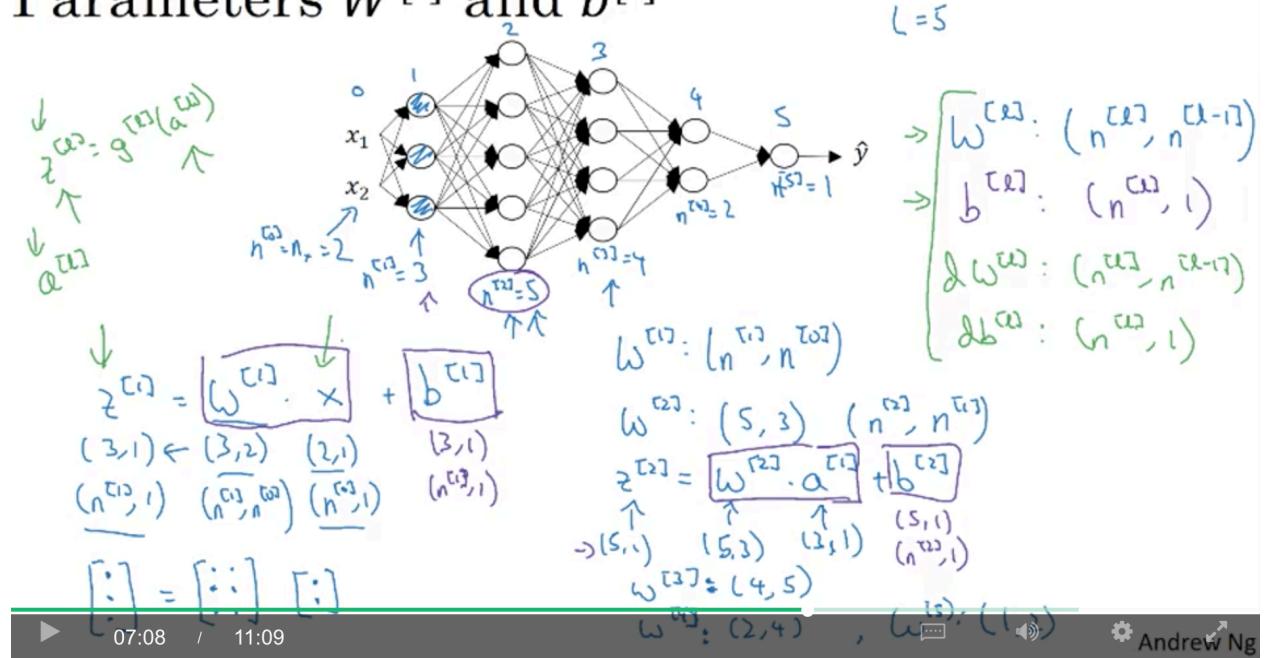


Figure 3: w-and-b-dimension.png

## Vectorized implementation

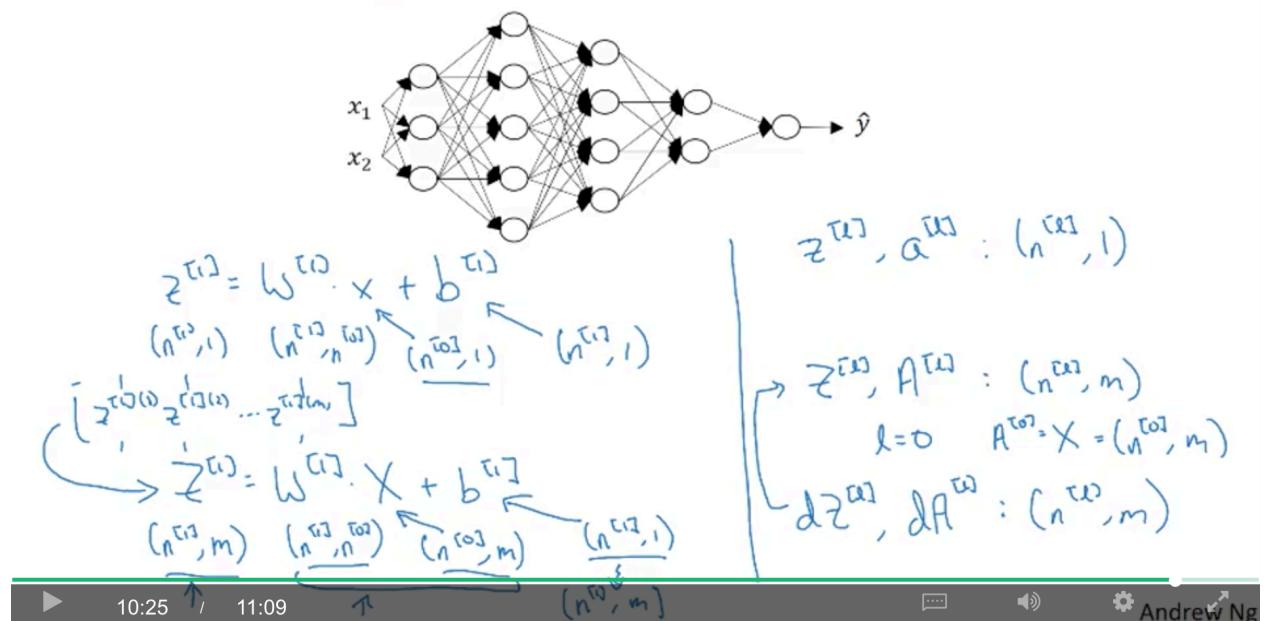


Figure 4: vectorized-implementation.png

# Intuition about deep representation

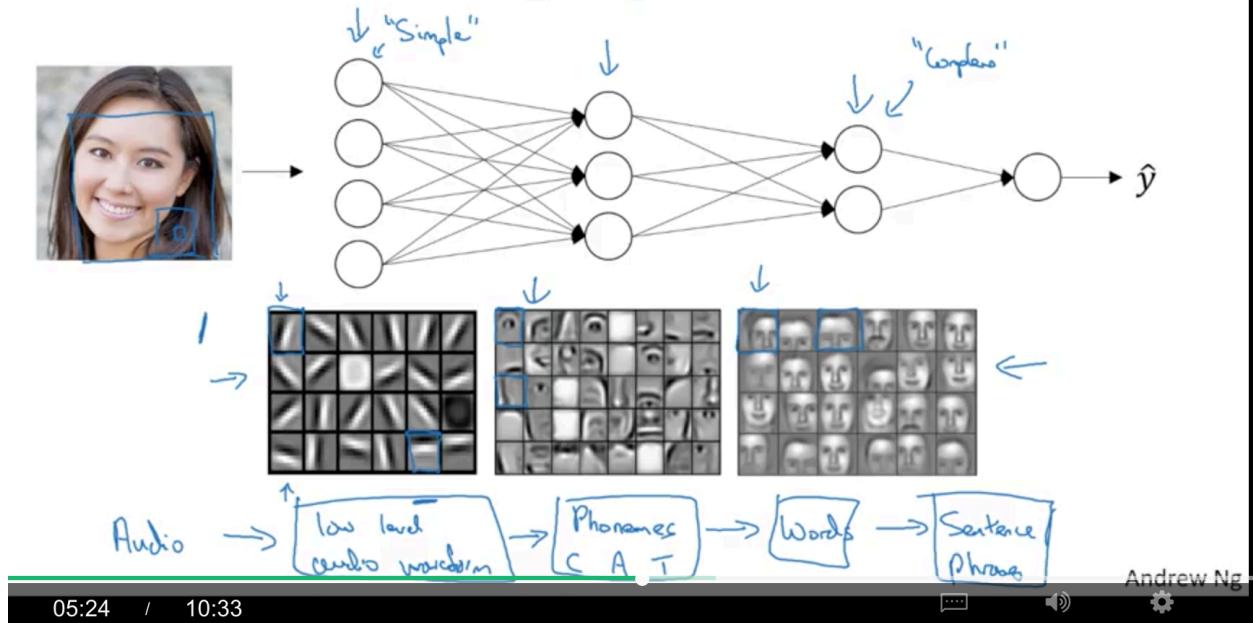


Figure 5: intuition-of-deep-representation.png

# 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.

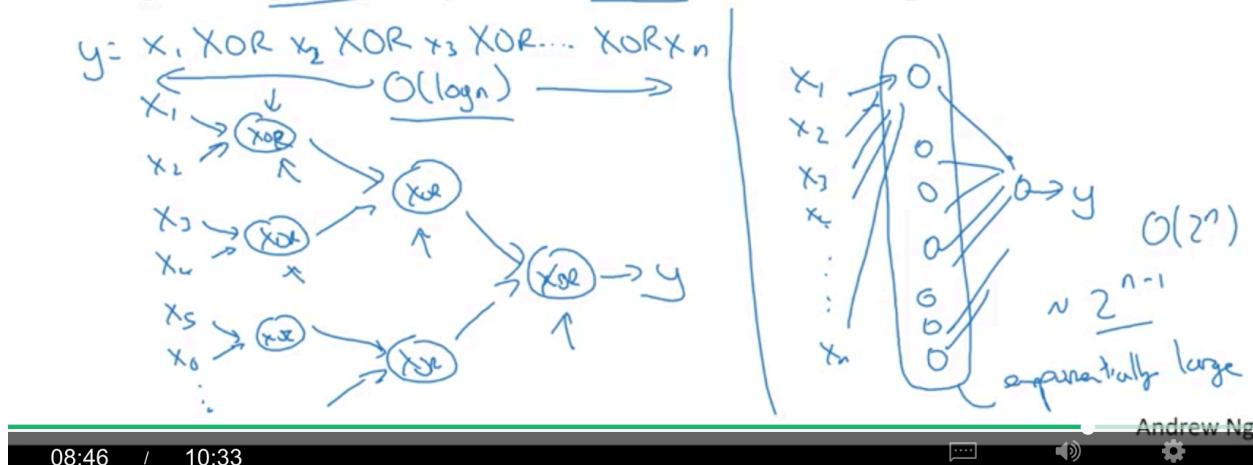
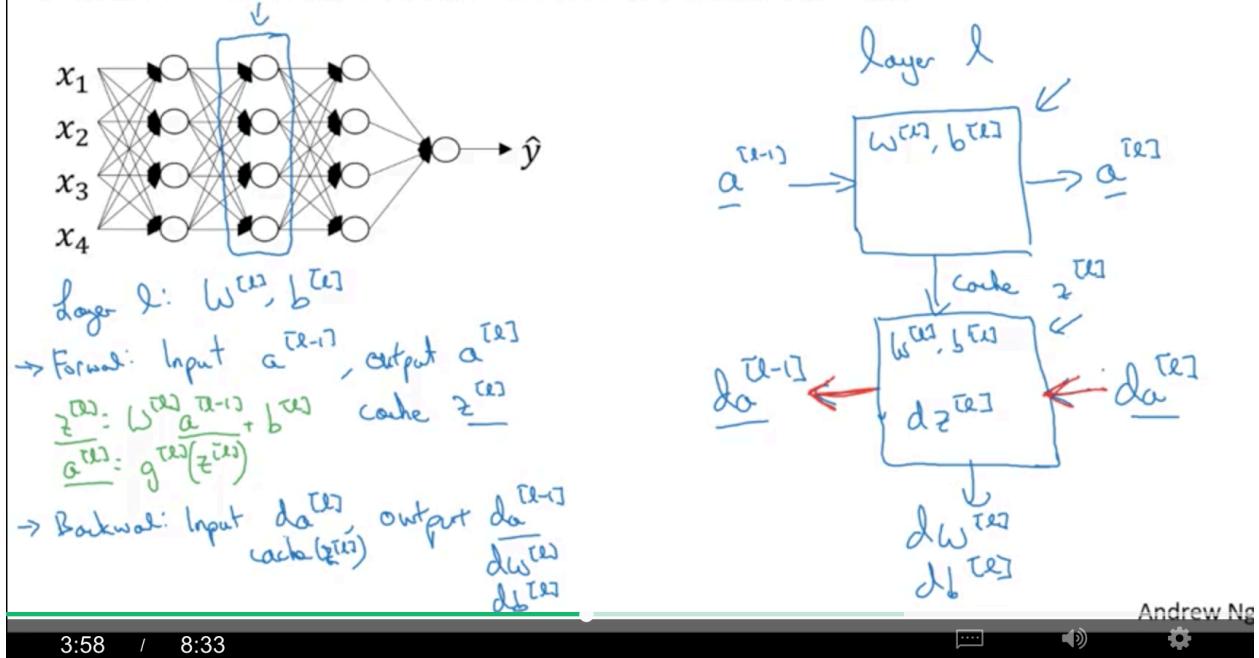


Figure 6: circuit-theory-and-deep-learning.png

# Forward and backward functions



Andrew Ng

3:58 / 8:33

Figure 7: forward-and-backward-functions.png

为了计算 backward，其实需要 cache 的有  $Z^{[l]}$ 、 $W^{[l]}$  以及  $b^{[l]}$ ：

## 1.6 1.6. forward and backward propagation

forward propagation for layer l:

backward propagation for layer l:

参考C1W2 的 backward propagation intuition 部分：注意： $da^{[l]} * g^{[l]}'(z^{[l]})$  是 element-wise product。

对于最后一层 L，如果是 sigmoid 并采用 logloss，那么：

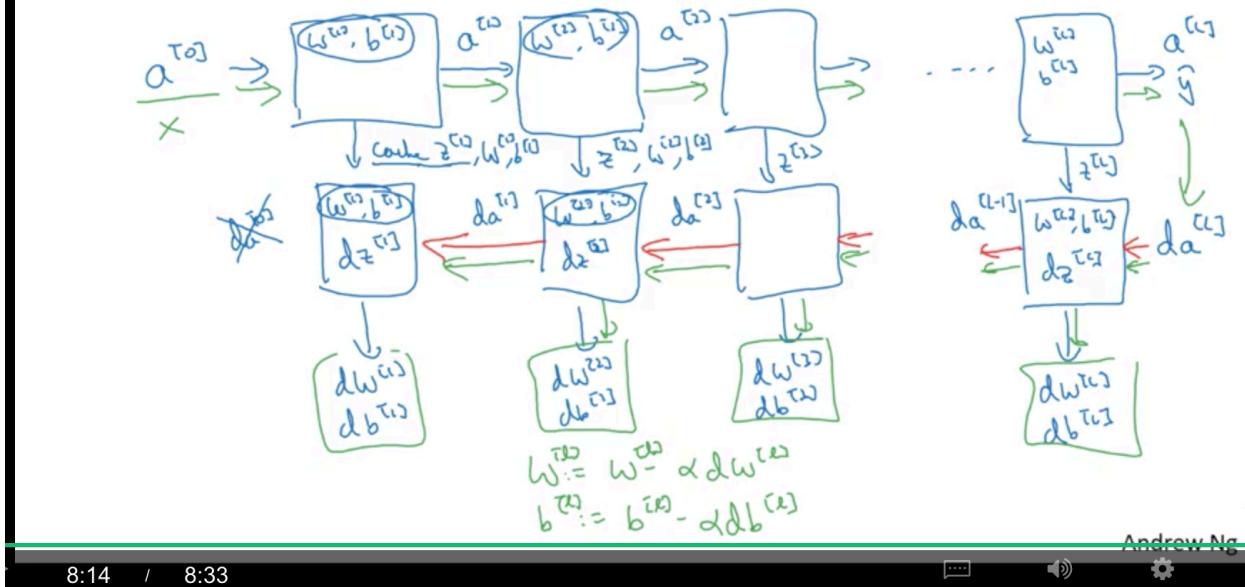
## 1.7 1.7. parameters vs hyperparameters

图中下方是：momentum, mini-batch size, regularization

## 1.8 1.8. what does this have to do with the brain?

## 1.9 1.9. others

# Forward and backward functions



8:14 / 8:33

Andrew Ng

Figure 8: forward-backward-functions.png

## Forward propagation for layer $l$

$$\begin{aligned} &\rightarrow \text{Input } a^{[l-1]} \leftarrow w^{[l]}, b^{[l]} \\ &\rightarrow \text{Output } a^{[l]}, \text{cache } (z^{[l]}) \\ z^{[l]} &= w^{[l]} \cdot a^{[l-1]} + b^{[l]} \\ a^{[l]} &= g^{[l]}(z^{[l]}) \end{aligned}$$

Vertwijf:  

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

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

$X = A^{[0]} \xrightarrow{\quad} \square \xrightarrow{\quad} \square \xrightarrow{\quad} \square \xrightarrow{\quad}$

01:58 / 10:29

Andrew Ng

Figure 9: forward-propagation-for-layer-l.png

## Backward propagation for layer $l$

→ Input  $\underline{da^{[l]}}$

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

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

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

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

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

$$dz^{[l-1]} = W^{[l+1]T} \cdot dz^{[l]} * g'(z^{[l]})$$

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

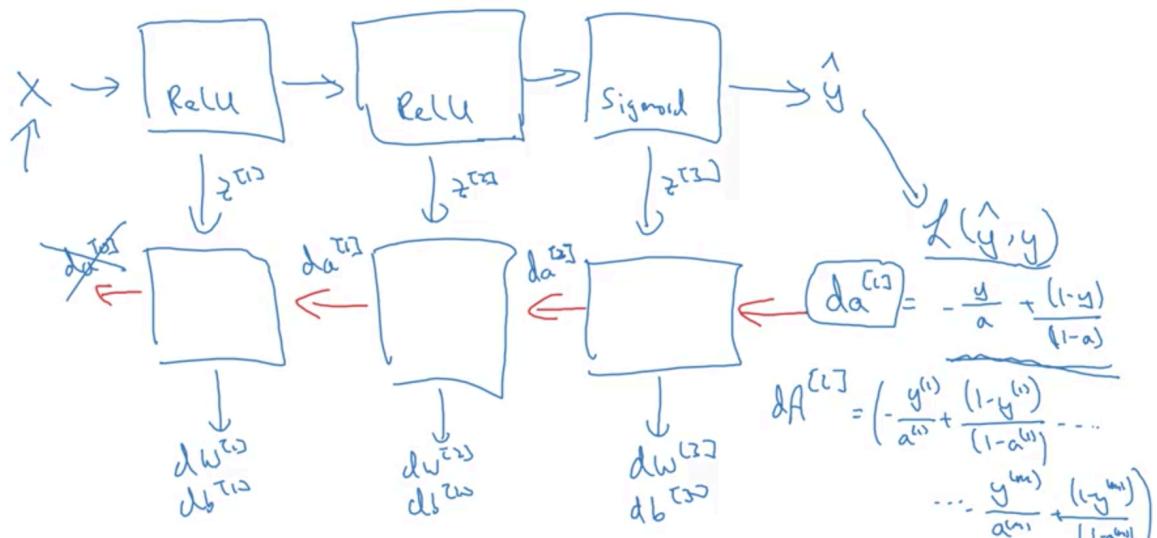
$$dW^{[l]} = \frac{1}{m} dA^{[l-1]} \cdot A^{[l-1]T}$$

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

05:18 / 10:29      Andrew Ng

Figure 10: backward-propagation-for-layer-l.png

## Summary



08:12 / 10:29      Andrew Ng

Figure 11: summary-forward-backward-propagation.png

## What are hyperparameters?

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

Hyperparameters:

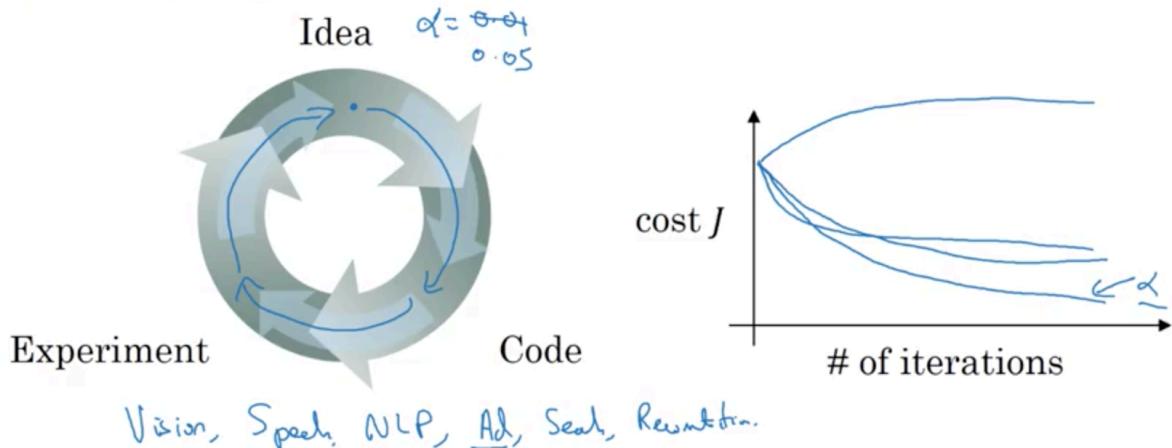
- learning rate  $\frac{\alpha}{n}$
- #iterations
- #hidden layers  $L$
- #hidden units  $n^{[1]}, n^{[2]}, \dots$
- choice of activation function

Costs: Momentum, minibatch size, regularizations, ...

Andrew Ng

Figure 12: summary-forward-backward-propagation.png

## Applied deep learning is a very empirical process



Andrew Ng

6:49 / 7:16

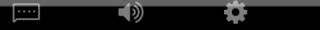


Figure 13: apply-deep-learning-is-a-very-empirical-process.png

# Forward and backward propagation

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

"It's like the brain"

$$\begin{aligned} dZ^{[L]} &= A^{[L]} - Y \\ dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]T} \\ db^{[L]} &= \frac{1}{m} np.\text{sum}(dZ^{[L]}, axis = 1, keepdims = True) \\ dZ^{[L-1]} &= dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\ \vdots \\ dZ^{[1]} &= dW^{[L]T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\ dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]T} \\ db^{[1]} &= \frac{1}{m} np.\text{sum}(dZ^{[1]}, axis = 1, keepdims = True) \end{aligned}$$

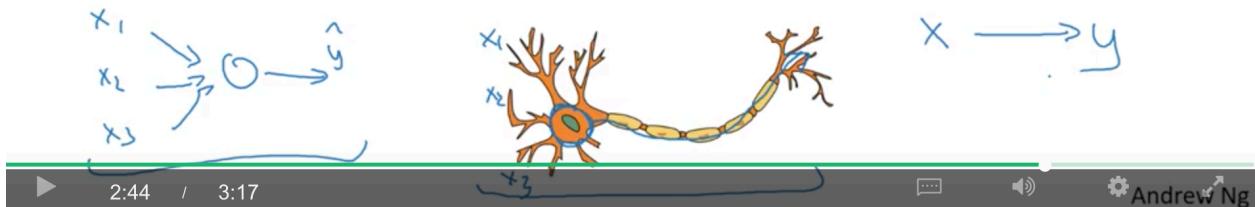


Figure 14: summary-and-brain.png

Thus for example if the size of our input  $X$  is (12288, 209) (with  $m = 209$  examples) then:

	<b>Shape of W</b>	<b>Shape of b</b>	<b>Activation</b>	<b>Shape of Activation</b>
<b>Layer 1</b>	$(n^{[1]}, 12288)$	$(n^{[1]}, 1)$	$Z^{[1]} = W^{[1]}X + b^{[1]}$	$(n^{[1]}, 209)$
<b>Layer 2</b>	$(n^{[2]}, n^{[1]})$	$(n^{[2]}, 1)$	$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$	$(n^{[2]}, 209)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
<b>Layer L-1</b>	$(n^{[L-1]}, n^{[L-2]})$	$(n^{[L-1]}, 1)$	$Z^{[L-1]} = W^{[L-1]}A^{[L-2]} + b^{[L-1]}$	$(n^{[L-1]}, 209)$
<b>Layer L</b>	$(n^{[L]}, n^{[L-1]})$	$(n^{[L]}, 1)$	$Z^{[L]} = W^{[L]}A^{[L-1]} + b^{[L]}$	$(n^{[L]}, 209)$

Figure 15: shape-of-L-layer-nn.png