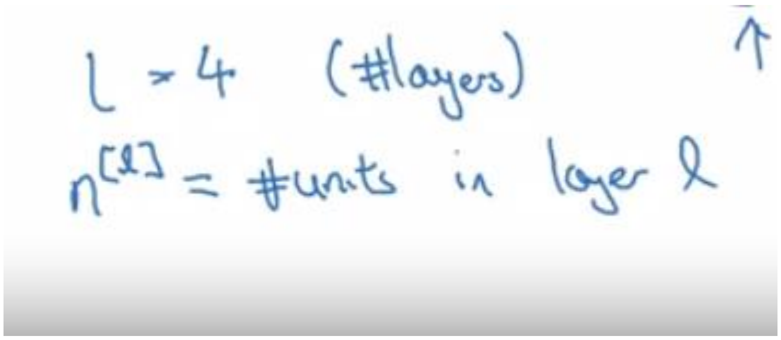
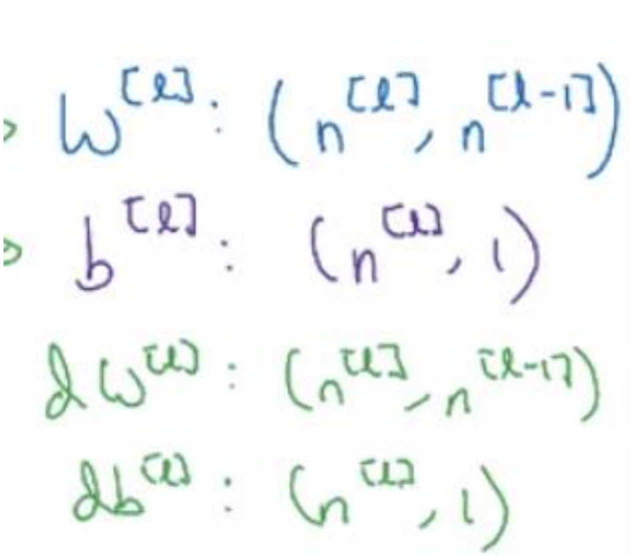
Week 4



Z[l] = W[l] * A[l – 1] + b[l] => Forward propagation



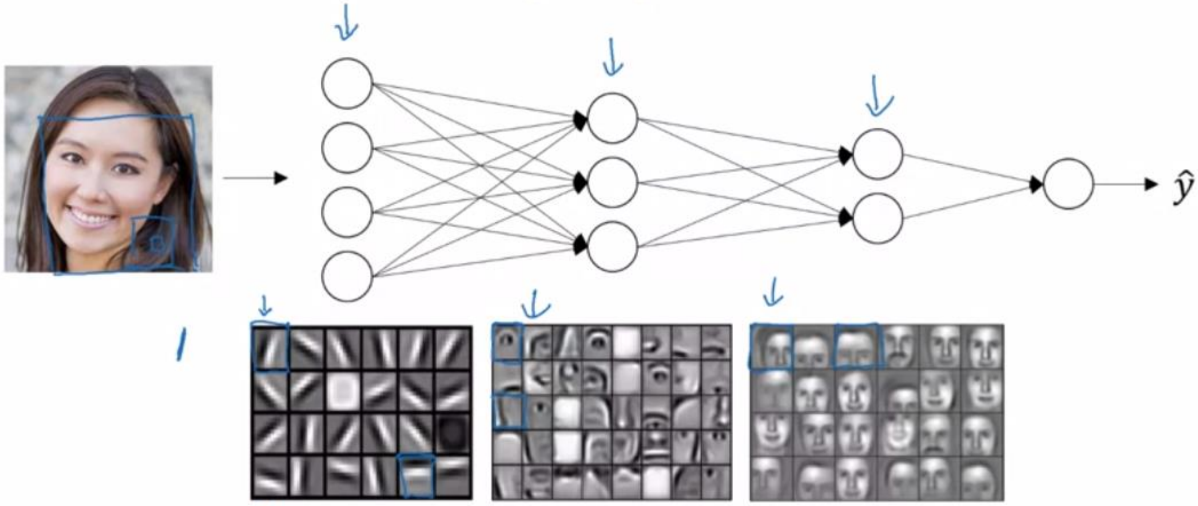
$$z^{[l]}, a^{[l]} : (n^{[l]}, 1)$$

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

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

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

Intuition about deep representation



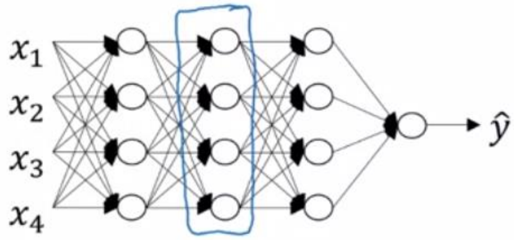
Audio → low level audio waveform → Phonemes C A T → Words → Sentence / Phrase

Andr

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.

Forward and backward functions



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

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

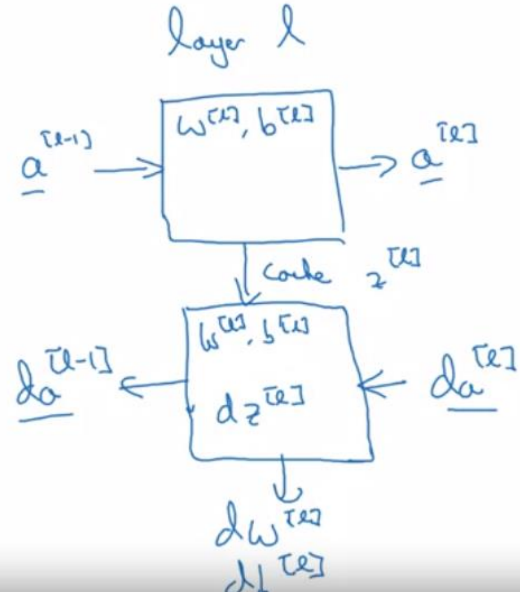
$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ cache $z^{[l]}$

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

→ Backward: Input $da^{[l]}$, cache($z^{[l]}$) output $da^{[l-1]}$

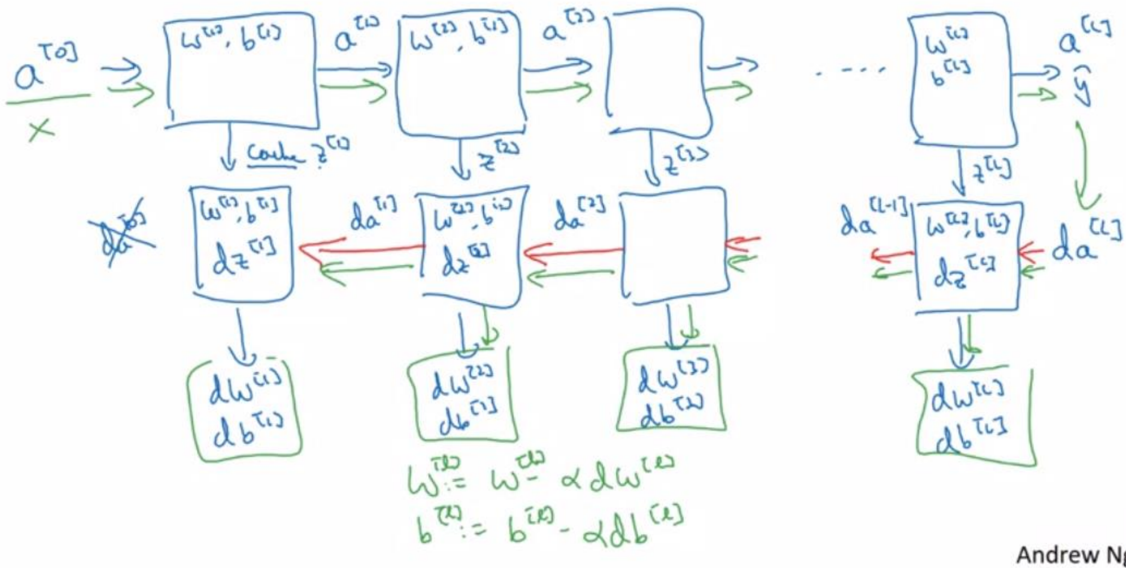
$dW^{[l]}$

$db^{[l]}$



Andrew

Forward and backward functions



Andrew Ng

Forward propagation for layer l

→ Input $a^{[l-1]}$ ←

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

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

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

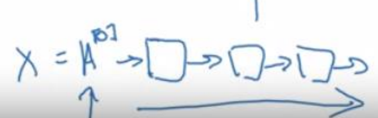
Vectorized:

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

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

$a^{[0]}$

$A^{[0]}$



Backward propagation for layer l

→ Input $da^{[l]}$

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

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

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

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

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

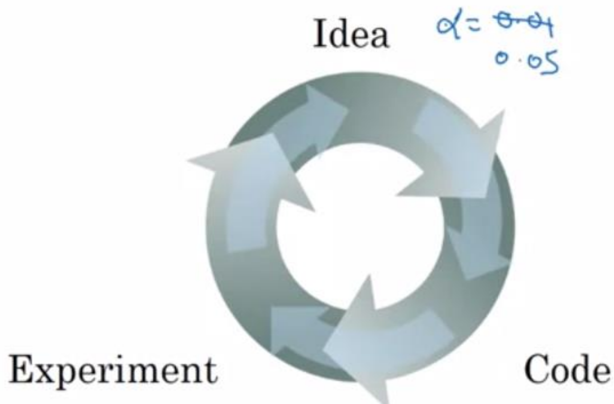
What are hyperparameters?

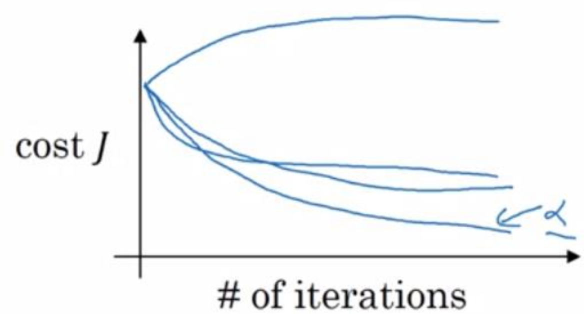
Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}$ ...

Hyperparameters: learning rate α
#iterations
#hidden layers L
#hidden units $n^{[1]}, n^{[2]}, \ldots$
choice of activation function

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

Applied deep learning is a very empirical process





Note that the formulas shown in the next video have a few typos. Here is the correct set of formulas.

$dZ^{[L]} = A^{[L]} - Y$

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

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

$dZ^{[L-1]} = W^{[L]^T} dZ^{[L]} * g'^{[L-1]}(Z^{[L-1]})$

Note that * denotes element-wise multiplication)

$\vdots$

$dZ^{[1]} = W^{[2]} dZ^{[2]} * g'^{[1]}(Z^{[1]})$

$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]^T}$

Note that $A^{[0]^T}$ is another way to denote the input features, which is also written as X^T

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

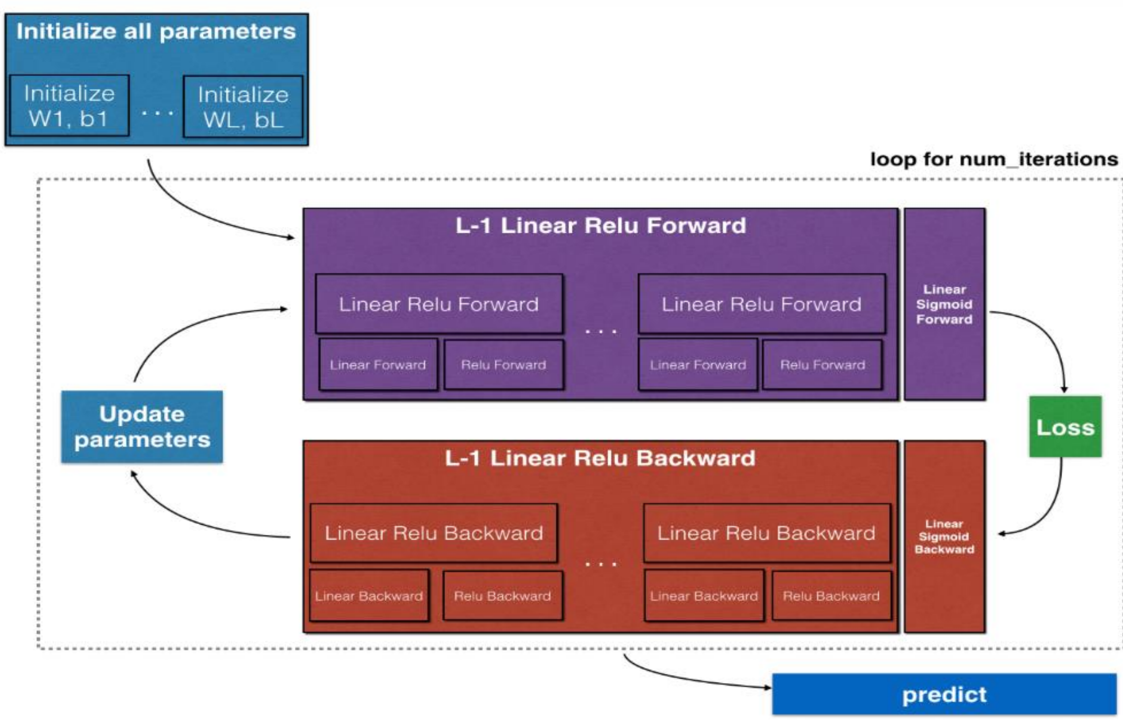


Figure 1

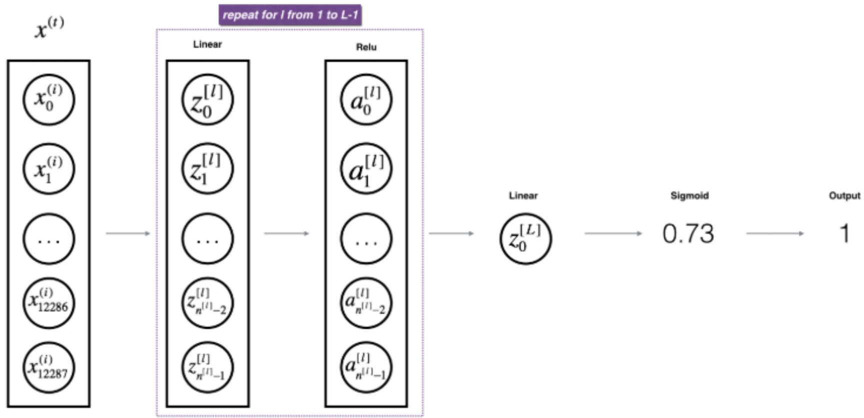


Figure 2 : *[LINEAR -> RELU]* × *(L-1)* -> *LINEAR* -> *SIGMOID* model

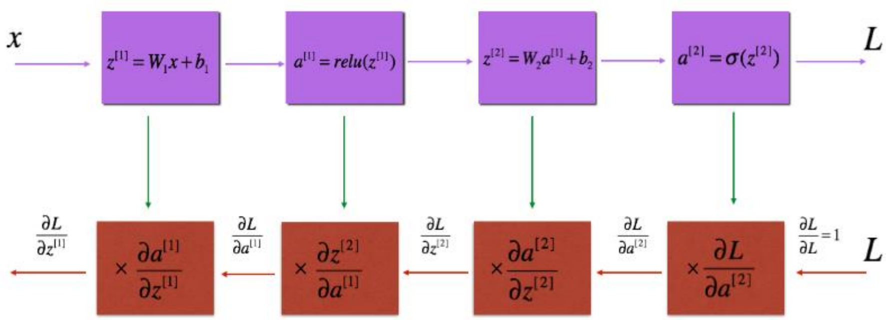


Figure 3 : Forward and Backward propagation for *LINEAR->RELU->LINEAR->SIGMOID*
The purple blocks represent the forward propagation, and the red blocks represent the backward propagation.

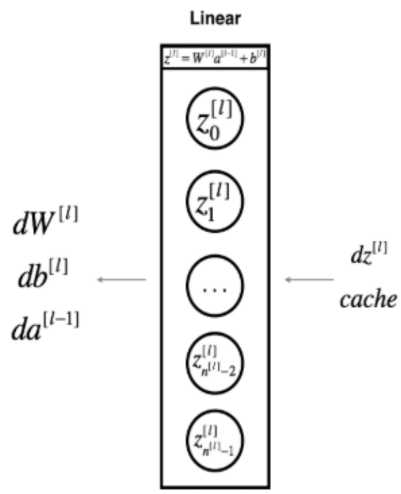


Figure 4

The three outputs $(dW^{[l]}, db^{[l]}, dA^{[l-1]})$ are computed using the input $dZ^{[l]}$.Here are the formulas you need:

$$dW^{[l]} = \frac{\partial \mathcal{J}}{\partial W^{[l]}} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$$

$$db^{[l]} = \frac{\partial \mathcal{J}}{\partial b^{[l]}} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[l](i)}$$

$$dA^{[l-1]} = \frac{\partial \mathcal{L}}{\partial A^{[l-1]}} = W^{[l]T} dZ^{[l]}$$

If $g(.)$ is the activation function, `sigmoid_backward` and `relu_backward` compute

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

.

As usual you will follow the Deep Learning methodology to build the model:

```
1. Initialize parameters / Define hyperparameters
2. Loop for num_iterations:
    a. Forward propagation
    b. Compute cost function
    c. Backward propagation
    d. Update parameters (using parameters, and grads from backprop)
4. Use trained parameters to predict labels
```