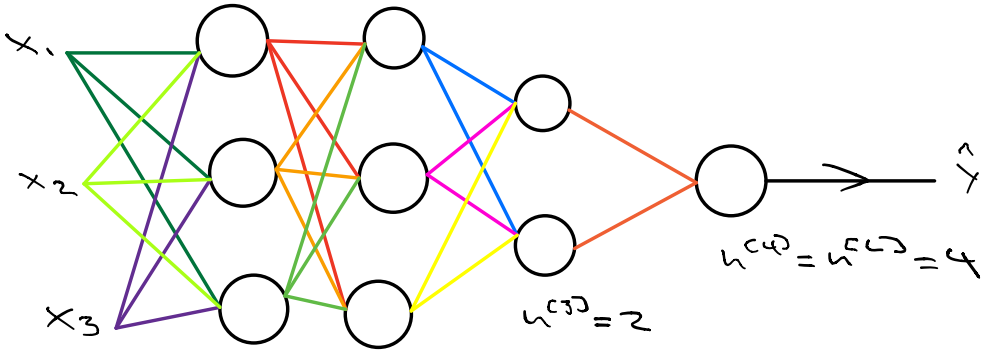
Week 4 Deep neural Network

Deep L-layer neural network

- What is a deep neural network?
  - the # of hidden layers in a neural network is another hyperparameter that can be tweaked.
  - "L = 4" denotes the # of layers in a neural network with 3 hidden layers + 1 output layer.
  - "$n^{[l]}$" describes the units in a particular layer
  - anything longer than 2-layer is categorized as deep neural net.



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

↳ # of features

Forward propagation in Deep neural networks

Computing forward propagation

↳ for: $z^{[l]} = w^{[l]} x + b^{[l]}$ $\quad x = a^{[0]}$

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

$h^{[2]}$: $z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$

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

⋮

$h^{[4]}$: $z^{[4]} = w^{[4]} a^{[3]} + b^{[4]}$

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

General rule: $z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$

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

- vectorized:
  - $z^{[1]} = w^{[1]} X + b^{[1]}$ $\quad X = A^{[0]}$
  - $A^{[1]} = g^{[1]}(z^{[1]})$

⋮

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

* once again, each example represents a column vector and they're stacked together to form a matrix of m examples.
* there's a for loop required to create the layers required in the neural net
  - for $l = 1 \ldots 4$ (in the example above) in forward propagation.

Getting the Dimensions Right

- often used as a debugging tool when writing code.
  - $W^{[l]} : (n^{[l]}, n^{[l-1]})$ — info from previous layer
    - # of neurons
  - $b^{[l]} : (n^{[l]}, 1)$
  - $dW^{[l]} : (n^{[l]}, n^{[l-1]})$
  - $db^{[l]} : (n^{[l]}, 1)$

* for single example implementation

Why Deep Representations?

- it's not only that they have to be big, they must be deep with many hidden layers.

- What is a deep network computing?
  - on an image (for facial recognition)
    - the first layer might be computing the features or detecting the edges.
    - the next stage will likely compute a more complex feature (such as an eye) by putting together lots of edges)
    - finally, by putting together the more complex features (from the previous layer), the network can try to recognize different types of faces.
  - similar functions are computed into more

complex features as we move down the neural network. (similar to computer hierarchical representation).

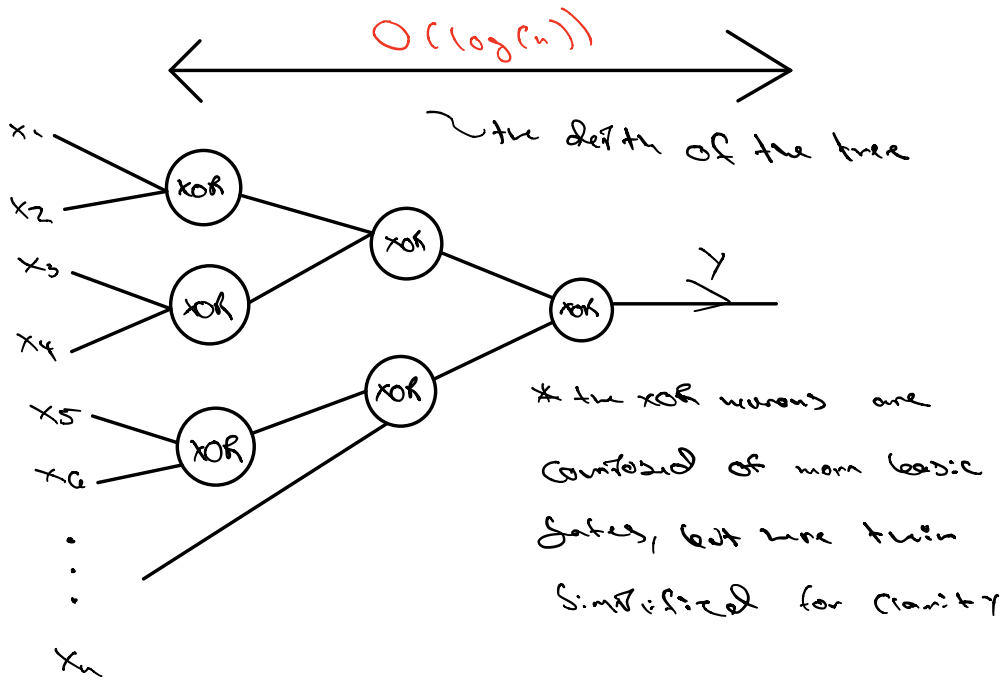
- in a speech recognition system,
  - the first layer might learn to detect low level audio waveform features
  - by composing basic low level waveforms we might then learn to detect basic units of sound (phonemes)
  - composing the phonemes we could learn to combine words
  - composing the words together we could learn to recognize phrases or sentences.

* once again a deep neural net learns in a hierarchical composition way from low level features to more abstract complex constructs.

- Circuit theory and deep learning
  - there are functions that can be computed with a "small" L-layer network that shallower networks require exponentially more units to compute (AND/OR gates and such functions)

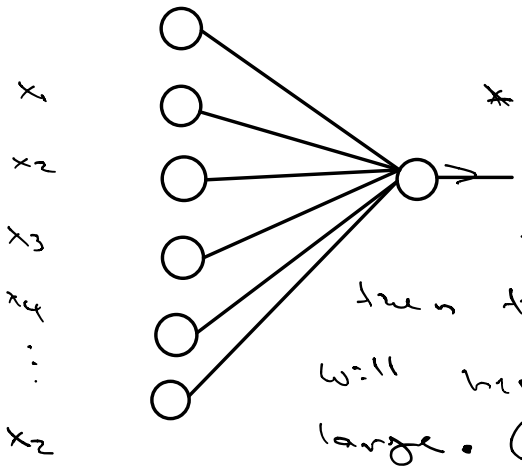
Example: $y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \text{ XOR } \ldots \text{ XOR } x_n$

n features



* the XOR neurons are composed of more basic gates, but are treated simplified for clarity

* the # of gates in this tree setup is not that large as the neurons halt every time we advance to the next layer.



* if we are forced to compute all the function in one hidden unit then the # of neurons required will need to be exponentially large. (2^{n-1}) hidden units in the hidden layer to compute it compared to $\log(n)$

Building Blocks of Deep neural networks

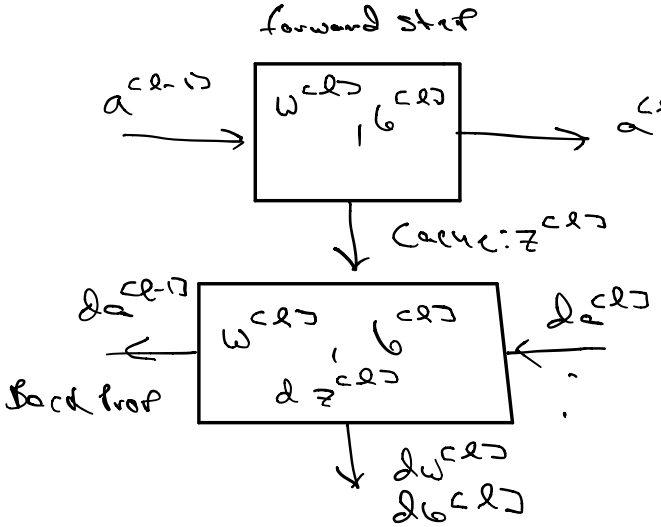
- forward and backward functions

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

- forward propagation: Input $a^{[l-1]}$ (activation from previous layer), output $a^{[l]}$
  - $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$
  - $a^{[l]} = g^{[l]}(z^{[l]})$

  * it is useful to cache the value $z^{[l]}$ for later use. (for the back prop step)

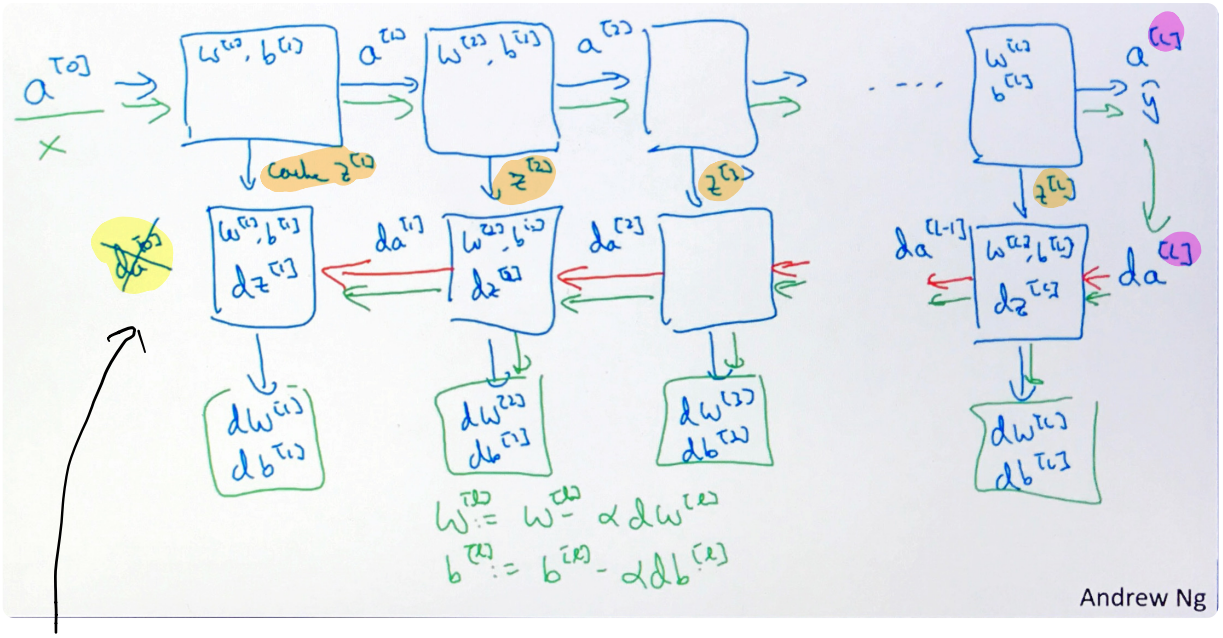
- backward propagation: Input $da^{[l]}$ and store cache $z^{[l]}$. output $da^{[l-1]}$, $dW^{[l]}$ $db^{[l]}$



One iteration of gradient descent : forward, back prop, Cost: total L

Cache data

- we actually also store W and b for the back prop.



not useful for training weights

forward and backward Propagation

- Backward Propagation:

$$\checkmark\ dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]}) \quad (1)$$

(element wise product)

$$\checkmark\ dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$$ this is also needed in the cache

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

$$\checkmark\ da^{[l-1]} = w^{[l]T} \cdot dz^{[l]} \quad (2)$$

Substituting (2) into (1) we get

$$\checkmark\ dz^{[l]} = w^{[l+1]T} dz^{[l+1]} * g^{[l]\prime}(z^{[l]})$$

Vectorized:

$$\checkmark\ dz^{[l]} = dA^{[l]} * g^{[l]\prime}(z^{[l]}) \quad (1)$$

$$\checkmark\ dw^{[l]} = \frac{1}{m} \cdot dz^{[l]} \cdot A^{[l-1]T}$$

- $db^{[l]} = \frac{1}{m} \cdot np.sum(dZ^{[l]}, axis=1, keepdims=True)$
- $dA^{[l-1]} = W^{[l]T} \cdot dZ^{[l]}$ (2)

Parameters vs Hyperparameters

- Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$ (and so on)
- Hyperparameter examples:
  - learning rate α
  - # iterations of gradient descent
  - # hidden layers (L)
  - # of hidden units ($n^{[1]}, n^{[2]}$...)
  - Choice of activation functions.

* all of these hyperparameters we have to specify to our learning algorithm in order to control the ultimate parameters (W and b)
  - it is the hyperparameters that determine the final value of W and b.

- Applied deep learning today is a very empirical process where we have an idea, code it, experiment and then iterate based on the results.
  - it's difficult to know in advance the values of the hyperparameters, therefore, experimentation is required

What does deep learning have to do with the brain?

- not a whole lot...
  - oversimplified reductive explanation.
- deep learning is very good at matching complex functions, but analogies to the brain aren't as useful as they once were.