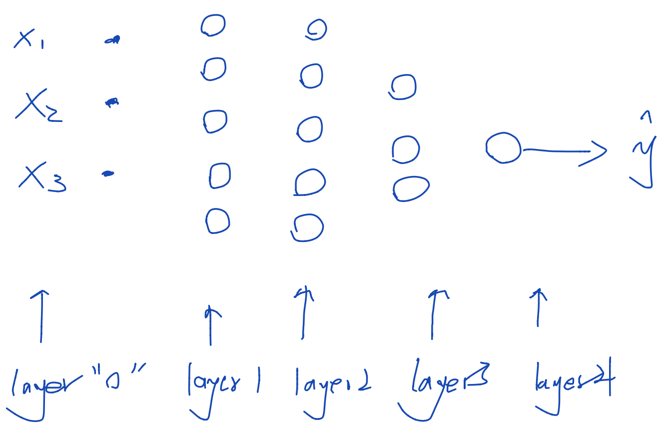
- Deep L-layer neural work
  - what is deep neural network?
    - 2 or more hidden layer

- Deep neural network notation



$L = 4$ (# layers) $n^{[1]} = 5$ $n^{[2]} = 5$ $n^{[3]} = 3$ $n^{[4]} = 1$

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

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

- Forward Propagation in Deep Neural Network

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

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

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

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

$\cdots\cdots$

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

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

- Getting your Matrix dimension right

  eg: x_1 x_2 . . . . $2 = 5$

  $W^{[l]} = n^{[l]} \times n^{[l-1]}$

- Vectorized Implementation

  m: traning examples

  $Z^{[l]}, A^{[l]}: n^{[l]} \times m$

  $\underset{(n^{[l]}, m)}{Z^{[l]}} = \underset{(n^{[l]}, n^{[l-1]})}{W^{[l]}} \underset{(n^{[l-1]}, m)}{X} + \underset{(n^{[l]}, 1)}{b^{[l]}}$ ← b broad casting

- Why Deep Representations?

  Low layer learn simple features, higher layes put these features together

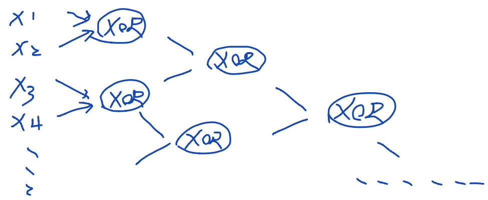
- Circuit theory and deep learning

  Informally: There are functions you can compute with a "small" 2-layer deep neural Network that shallower networks require expontationally more hidden units to compute.

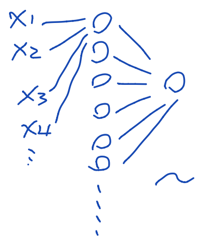
  e.g. XOR tree:

  X_1 XOR X_2 XOR X_3 · · · · XOR X_n $\quad O(\log(n))$

[Deep neural network]



[shallow network]



$\sim 2^{n-1}$ neurons in the hidden layer

exponentially larger

- Building Blocks of Deep Neural Networks

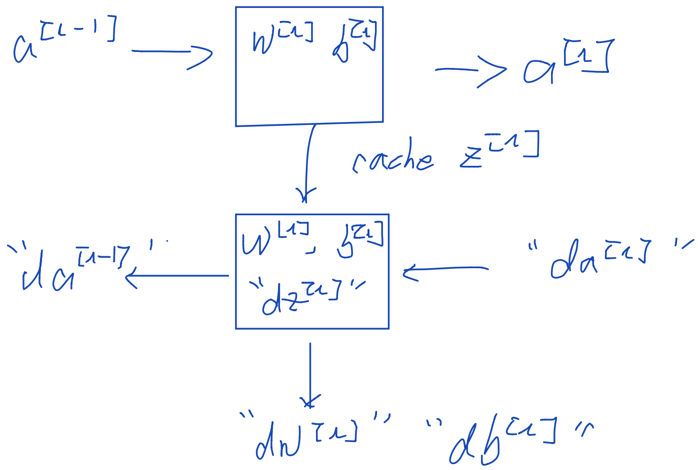
large l: $w^{[l]}$, $b^{[l]}$

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

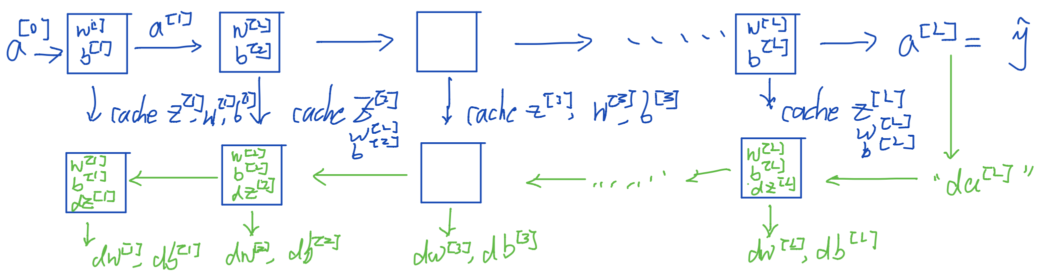
$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]} \quad \text{cache } z^{[l]}$$

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

Backward: Input "$da^{[l]}$", output "$da^{[l-1]}$"



∘ Forward and Backward Functions



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

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

∘ Forward and Backward Propagation

Forward Propagation for layer l:

Input $a^{[l-1]}$

Output: $a^{[l]}$, cache $z^{[l]}, w^{[l]}, b^{[l]}$

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

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

Backward Propagation for layer l

Input $da^{[l]}$

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

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

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

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

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

Vectorized:

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

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

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

$dA^{[l-2]} = W^{[l]T} \cdot dZ^{[l]}$

- Parameters VS hyperparameters

- What are hyperparameters?

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

  Hyperparameters: learning rate α

  \# iterations

  \# hidden layer L

  \# hidden unit $n^{[1]}, n^{[2]}, \ldots$

  choice of activation function

  "The control parameters"

- Applied deep learning is a very empirical process

$$\text{idea} \to \text{code} \to \text{experiment} \to \text{idea}$$