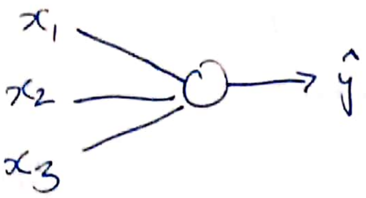
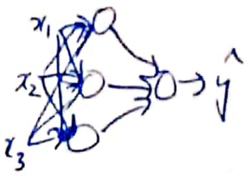
Week 4: Deep Neural Networks

Deep L-layer NN

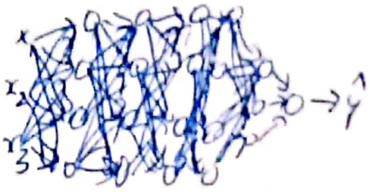


logistic regression

shallow



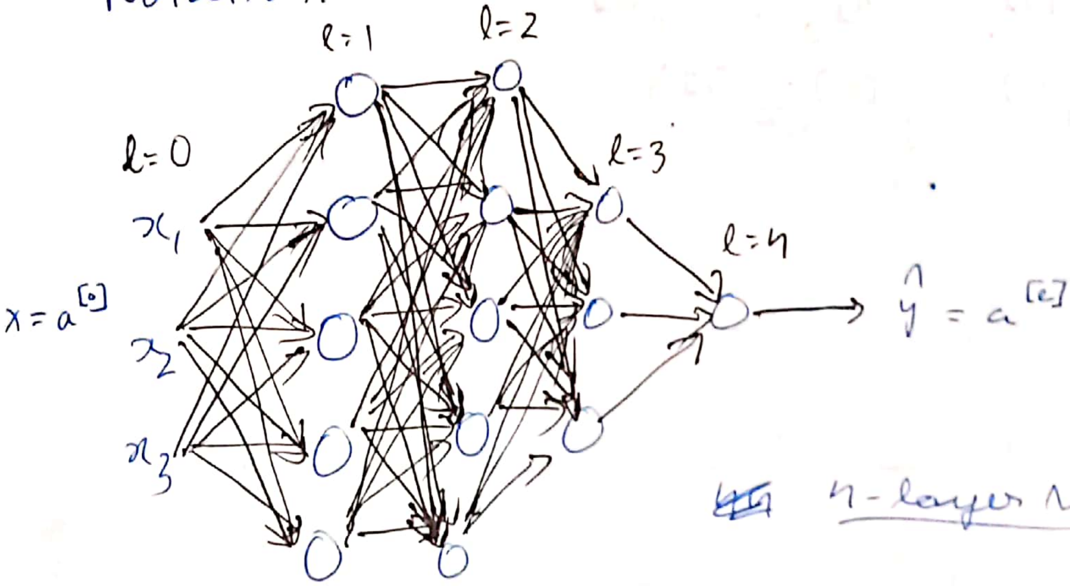
1 hidden layer



5 hidden layers

deep

Notation:



~~4L~~ 4-layer NN

$L = 4$ (# layers)

$n^{[l]}$ = # units in layer l - $n^{[1]} = 5, n^{[2]} = 5, n^{[3]} = 3, n^{[4]} = 1$

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

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

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

$W^{[l]}$ = weights for $z^{[l]}$

$b^{[l]}$ = bias for $z^{[l]}$

Forward Propogation in a Deep NN

It's similar to the Forward Prop. that we learnt before, just with repeated steps (since we have more layers).

For the 4 layer NN diagram given before,

$$z^{[1]} = W^{[1]} x + b^{[1]} \quad (x = a^{[0]})$$

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

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

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

$$\cdots$$

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

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

$\therefore$ The general equation is,

$$\boxed{\begin{aligned} Z^{[l]} &= W^{[l]} A^{[l-1]} + b^{[l]} \\ A^{[l]} &= g^{[l]}(Z^{[l]}) \end{aligned}}$$

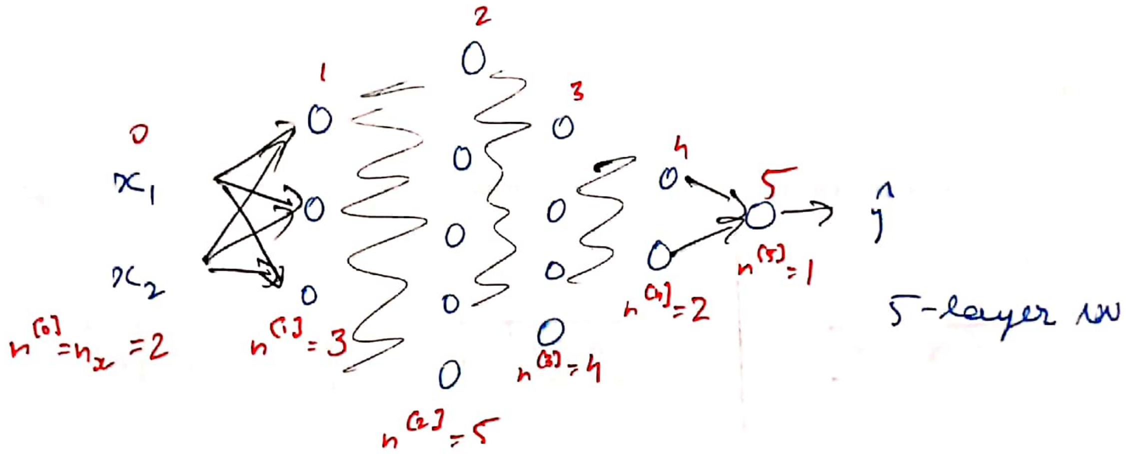
We can use a for loop to repeat it (we can't use vectorization).

for $l = 1 \ldots 4$:

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

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

Matrix Dimensions



For single training example,

$$z^{[1]} = w^{[1]} \cdot X + b^{[1]}$$

$(3, 1)$	$(3, 2)$	$(2, 1)$	$(3, 1)$
$(n^{[1]}, 1)$	$(n^{[1]}, n^{[0]})$	$(n^{[0]}, 1)$	$(n^{[1]}, 1)$

∴ General equations,

$W^{[l]}$ and $dW^{[l]}$:	$(n^{[l]}, n^{[l-1]})$
$b^{[l]}$ and $db^{[l]}$:	$(n^{[l]}, 1)$
$Z^{[l]}$ and $a^{[l]}$ and $dZ^{[l]}$ & $dA^{[l]}$:	$(n^{[l]}, 1)$
X or $a^{[l-1]}$:	$(n^{[l-1]}, 1)$

1 becomes m

For vectorized implementation,

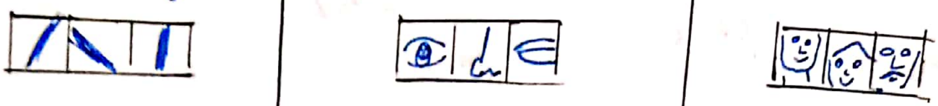
$W^{[l]}$ and $dW^{[l]}$:	$(n^{[l]}, n^{[l-1]})$
$b^{[l]}$ and $db^{[l]}$:	$(n^{[l]}, m)$
$Z^{[l]}$ and $A^{[l]}$ and $dZ^{[l]}$ and $dA^{[l]}$:	$(n^{[l]}, m)$
X or $A^{[l-1]}$:	$(n^{[l-1]}, m)$

Why deep representation

Image recognition:

1st hidden layer → 2nd Hidden Layer → 3rd Hidden Layer

Identifies edges | Identifies parts | Identifies faces

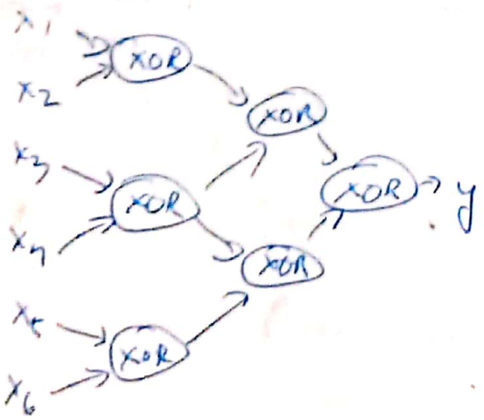


Each hidden layer uses the previous layer to build on the complexity. The ~~so~~ lower hidden layers look at smaller parts of the image.

Circuit Theory:

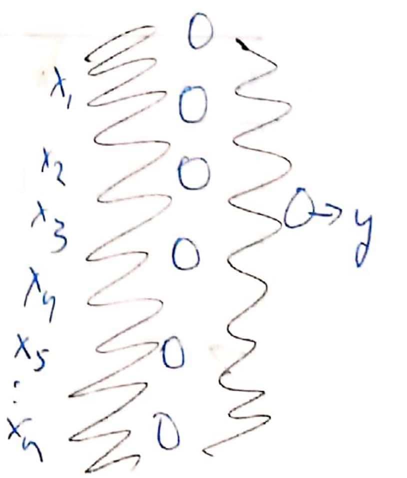
There are functions you can compute with a "small" L-layer deep ~~net~~ neural network that shallower networks require exponentially more hidden units to compute

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



Small L-layers deep NN ✓

VS

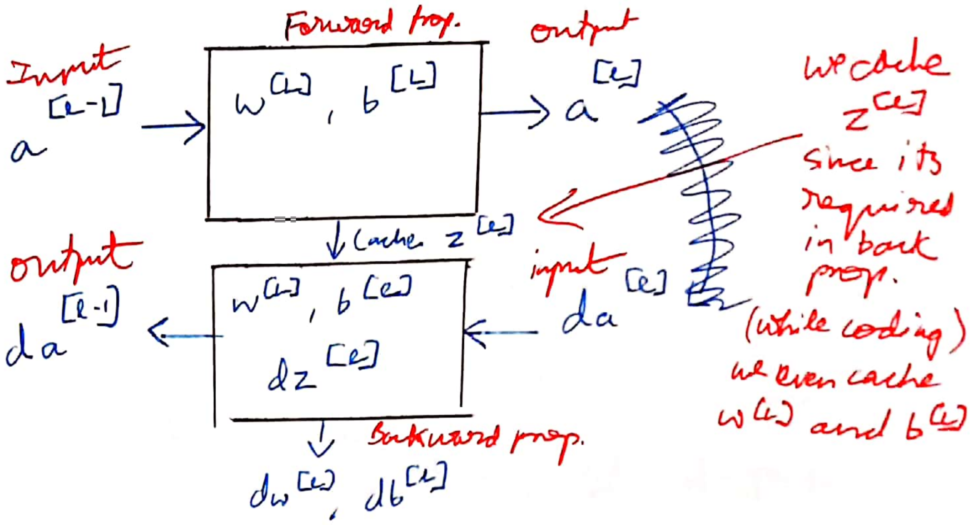


Shallow network with exponentially more hidden units ($\approx 2^{n-1}$ units)

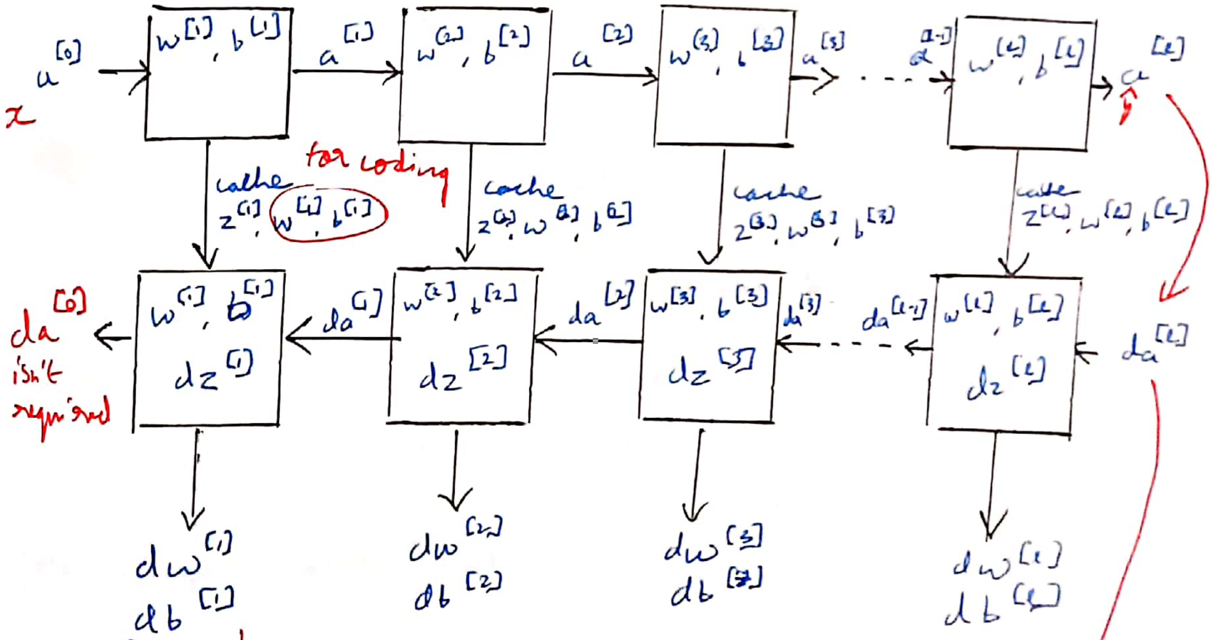
Building Blocks of Deep Neural Network

Here we build a general building block that represents ~~the~~ the calculations of a single hidden layer

layer l



Entire neural network:



Used to update values

$$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}$$

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

$$da^{[l]} = -\frac{y}{a} + \frac{(1-y)}{(1-a)}$$

$$dA^{[l]} = \left(\frac{-y^{(1)}}{a^{(1)}} + \frac{(1-y^{(1)})}{(1-a^{(1)})} \cdots \frac{-y^{(m)}}{a^{(m)}} + \frac{(1-y^{(m)})}{(1-a^{(m)})}\right)$$

Forward ~~fo~~ And Backward Propogation

Forward Propogation:

→ Input $a^{[l-1]}$

→ Output $a^{[l]}$, cache $(z^{[l]})$

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

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

Backward propogation:

→ Input $da^{[l]}$

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

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

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

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

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

→ In vector form:

$$dz^{[l]} = dA^{[l]} * g^{[l]'}(z^{[l]})$$

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

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

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

Parameters vs Hyperparameters

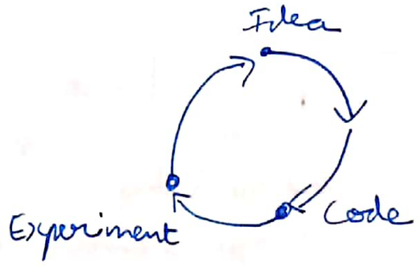
Parameters: $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, w^{[3]}, b^{[3]} \ldots$

Hyperparameters - they determine and control the parameters.

- Learning rate α
- # iterations
- # hidden layers l
- # hidden units $n^{[1]}, n^{[2]} \ldots n^{[l]}$
- Choice of activation function, etc

How to choose the best hyperparameters?

By trial and error method.



Try for different values and see what works best.

In the second course, some methods of choosing hyperparameters will be discussed.

Is DL like the brain?

No. Although a neuron and a DL unit ~~seems~~ seem similar, we don't know how the neuron works and it is more complicated. So this analogy isn't true.

Scanned with CamScanner