Week 4

Deep neural networks

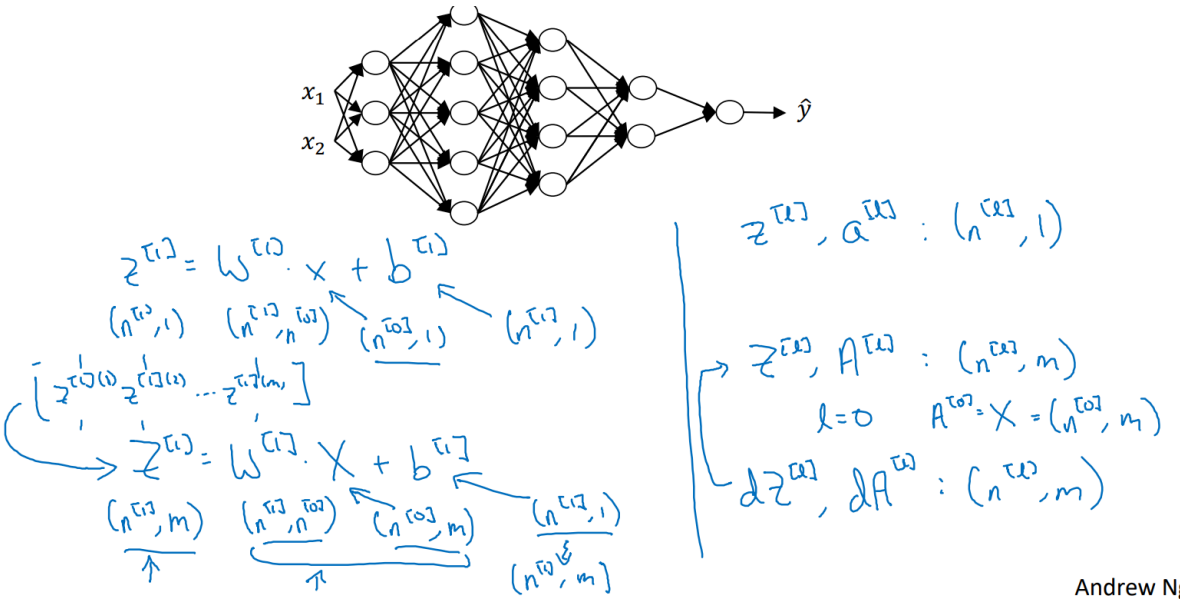
Neural networks are of two types - Shallow and deep
Shallow are the ones with one layer and deep have 5 or more layer
the shallow one can be said to be a Logistic neural network

If y hat for some nn is a[L] , i.e., L no of layers present in the nn, then w, b, g,z will have same super script ‘L’ , instead of layer no.

n[0] (superscript = 0) can be given as n_x (subscript = x) and equals no of inputs to the NN

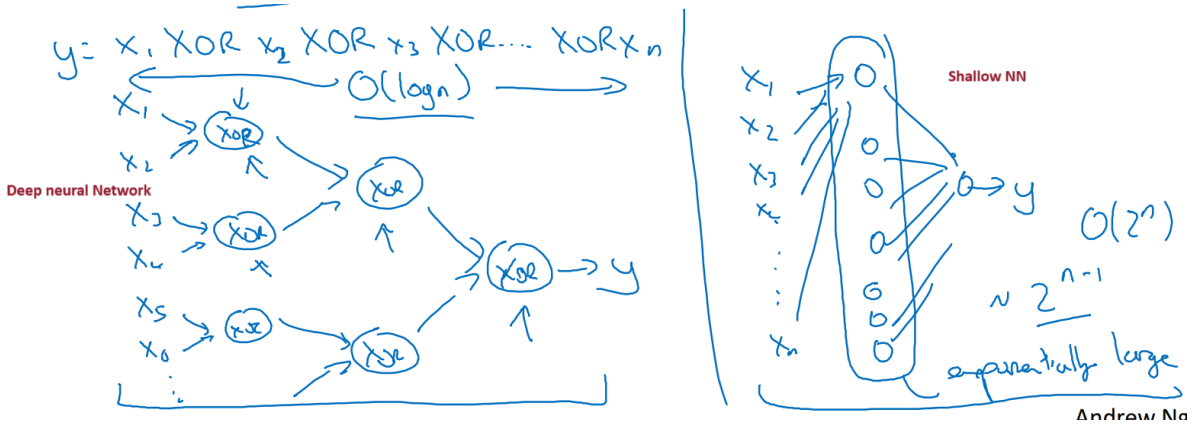
Dimensions

- Z^[l]: n^[l] by 1
- W^[l]: n^[l] by n^[l-1]
- X: n^[l-1] by 1
- B^[l]: n^[l] by 1
- DW^[l]: n^[l] by n^[l-1]
- DB^[l]: n^[l] by 1
- A^[l]: n^[l] by 1
- DZ^[l]: n^[l] by 1
- DA^[l]: n^[l] by 1



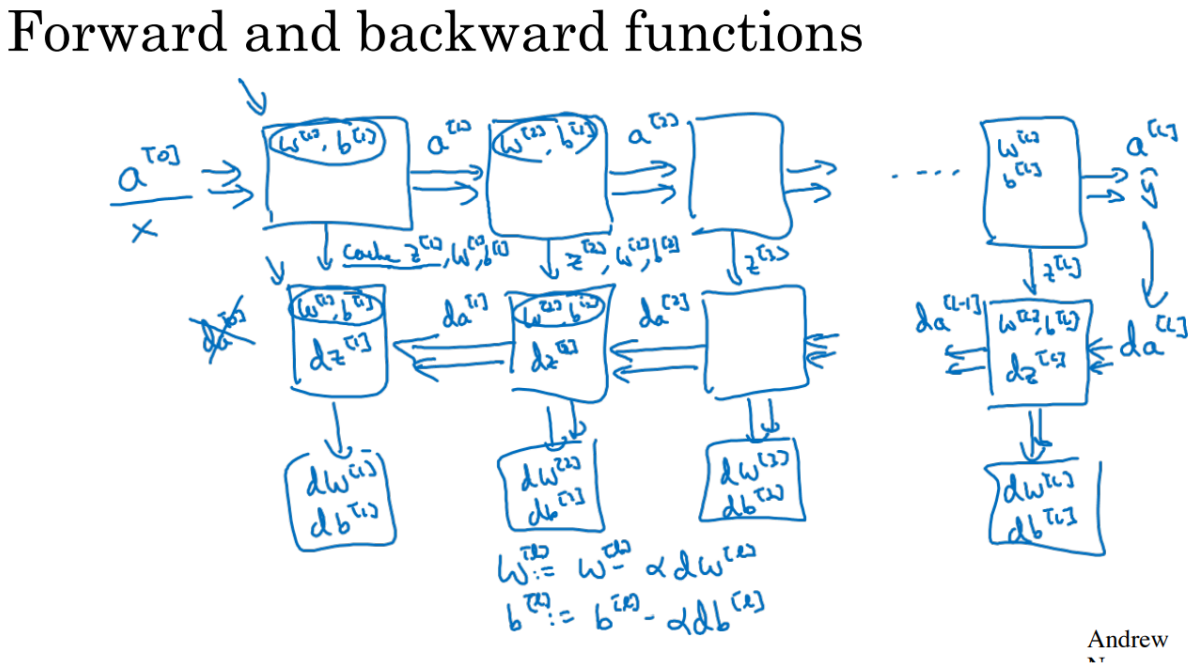
Andrew Ng

The image below is on the circuit theory and DL



This fig shows that the deep neural network is preferable than shallow as the O(n) is less for deep

The below imag represents how forward and backward propagation occurs.



Z, w, b is the cache i.e., stored temporarily to be used in back function/ propagation for dZ, dw , db.
As seen the parameters are calculated from previous functions[for eg. w2 from w1 in forward function] and the cache is collected in forward function
Once, y hat is calculated, da[L] is calculated by differentiating it.
Go thru the above chart once

da[0] is computed at the end of backward function but isn't needed

Forward Propagation:

Input -a^(l-1), and output - a^l, cache(z^l)

- Forward function: a^l = g(z^l), where z^l = w^l * a^(l-1) + b^l
- Vectorized implementation: a^l = g(z^l) = g(w^l * a^(l-1) + b^l)

Backward Propagation:

Input - da^l and output - dw^l, db^l, da^(l-1)

- Derivative of z^l: dz^l = da^l * g'(z^l)
- Derivative of w^l: dw^l = dz^l * a^(l-1)
- Derivative of b^l: db^l = dz^l
- Derivative of a^(l-1): da^(l-1) = wlT * dz^l

Vectorized Backward Propagation:

- Derivative of z^l: dz^l = da^l * g'(z^l)
- Derivative of w^l: dw^l = (1/m) * dz^l * a(l-1)T
- Derivative of b^l: db^l = (1/m) * np.sum(dz^l, axis=1, keepdims=True)
- Derivative of a^(l-1): da^(l-1) = wlT * dz^l

Hyperparameters: control the parameters

- Learning rate (alpha)
- Number of iterations
- Number of hidden layers (capital L)
- Number of hidden units
- Choice of activation function (RELU, tangent, sigmoid)

Parameters:

- Model parameters (W and B)

Applied deep learning is a very empirical process [trial n erro]

As the cycle goes of idea->code->experiment->idea…