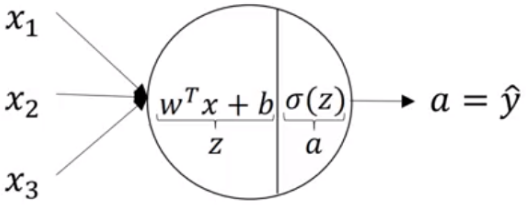
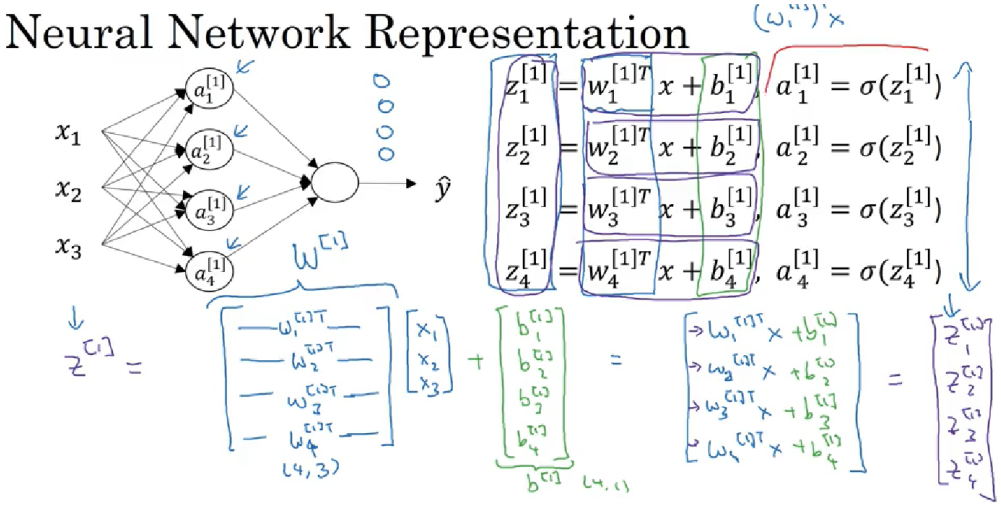
deep Learning_c1_week3

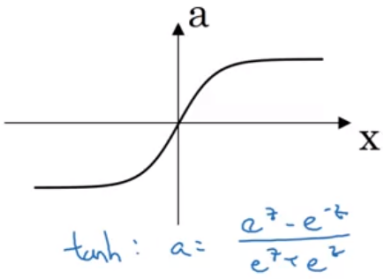
Created:	19/01/04 13:02	Updated:	19/01/04 21:13

shallow neural network

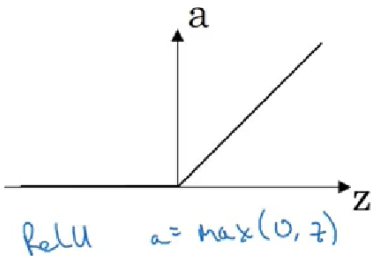
build a neural network with one hidden layer.

1. ## Notation
    1. when are talking about **X layers NN**. The input layer isn't counted.
    2. define `a0 = x` (the input layer)
    3. **[x]** 👈 denote the x th layer, **(i)** 👈denote the i th training examples
    4. a_i , the subscript denote the i th unit in layer.
2. ## Representation
    - a neuron's job:
        - "get input from previous layer, then make **linear transformation** and **execute active function**."
        - 
    - a shallow NN looks like:
        - 
        - in matrix W, different rows represent different units in hidden layer
        - in matrix A,different columns represent different training examples.
    - 👆forwardprop
        - #if we want to compute the n th layer.
        - $$Z^{[n]} = W^{[n]} X A^{[n-1]} + b^{[n]}$$
        - $$A^{[n]} = g(Z^{[n]})$$
        - , while g() is the active function.
3. ## Activation function
    1. ### why
        1. there are many relations in real world can't be fit by linear function.
        2. without activation function, the whole NN would be degenerated into a **linear combination,** which has poor capability to fit complicated model.

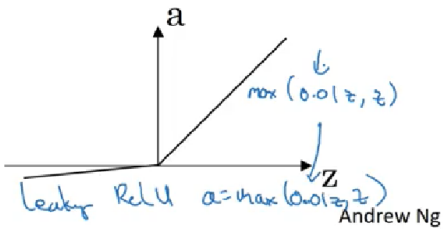
2. **what**
    1. sigmoid
    2. **tanh**



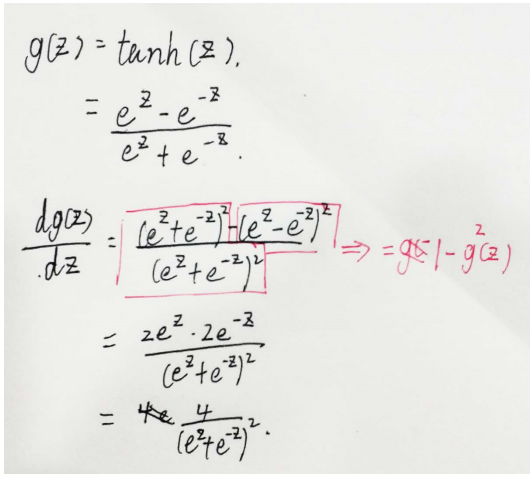
        1.
        2. `A = (np.exp(z) - np.exp(-z)) / (np.exp(z) + np.exp(-z)) # Where z is the input matrix`
    3. **relu** (rectify linear unit)



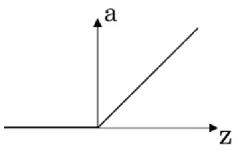
        1.
        2. `RELU = max(0,z)`
    4. leaky relu



        1.
        2. `Leaky_RELU = max(0.01z,z)`
3. **derivative**
    1. **sigmoid:**
        1. `g'(z) = (1 / (1 + np.exp(-z))) * (1 - (1 / (1 + np.exp(-z))))`
        2. `g'(z) = g(z) * (1 - g(z))`
    2. **tanh:**
        - `g'(z) = 1 - np.tanh(z)^2 = 1 - g(z)^2`
        - details
        -



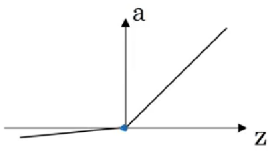
    3. **relu**



ReLU

$g(z) = \max(0, z)$

$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$



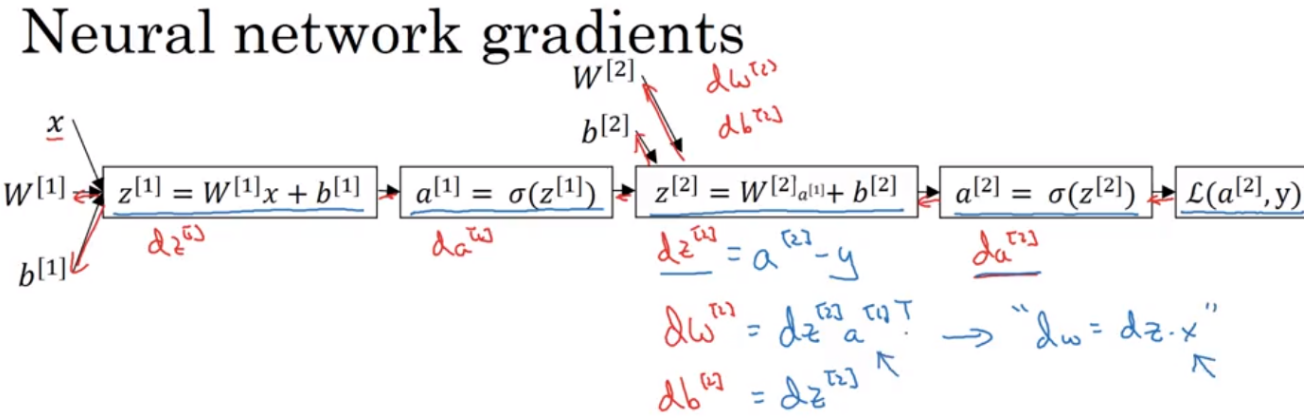
Leaky ReLU

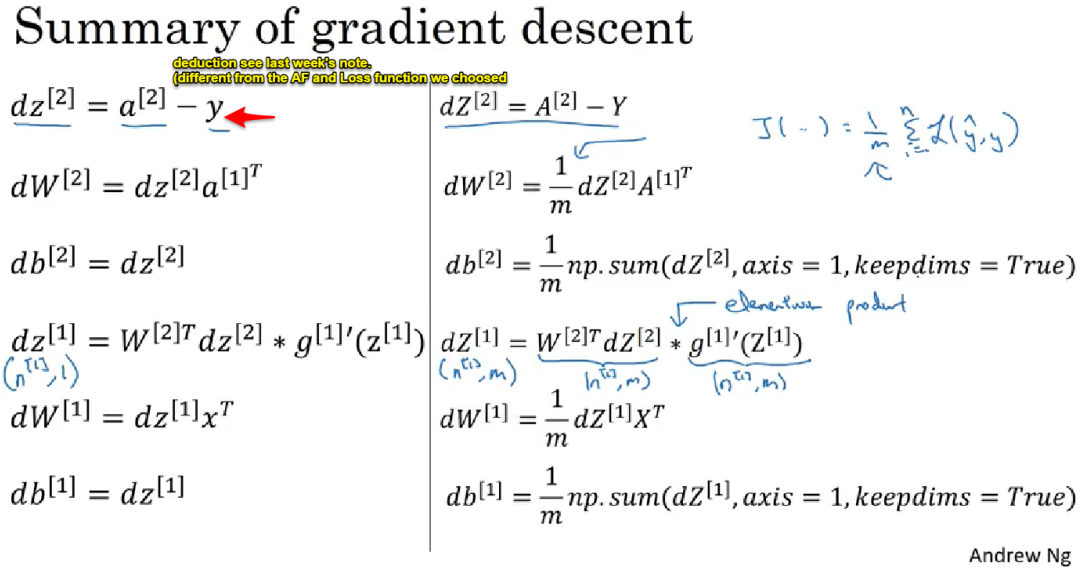
$g(z) = \max(0.01z, z)$

$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$

4. **How to choose**
   - usually, tanh and relu
     - the mean of tanh output is closer to zero, and so it centers the data better for the next layer.
     - they are learning faster than sigmoid (see their derivative!), so we often choose tanh and relu as active function.
   - relu always used in CNN
   - tanh always used in RNN

4. **Gradient Descent (backprop)**
   - once get hold of the **chain rule**, backprop seems no mysterious any more.





   - check matrix dimensions frequently, make sure them **match up!**

5. **Random Initialization**
   - **why?→symmetric breaking!**
     - all hidden units will be completely identical (symmetric) - compute exactly the same function
     - our neural network would degenerate to a logistic regression.
   - **how**
     - `W1 = np.random.randn((nx,m)) * epsilon`

- epsilon should be a small number e.g. **0.01**, which can let **Z** don't get into **the fat part of active function.** (in fat part, the derivative is small, that would lead to a slow learning speed).
- more initialization strategy would be learned in future video.

Assignment:Planar data classification with one hidden layer

- **procedure**

  Reminder: The general methodology to build a Neural Network is to:

  1. Define the neural network structure (# of input units, # of hidden units, etc).
  2. Initialize the model's parameters
  3. Loop:
     - Implement forward propagation
     - Compute loss
     - Implement backward propagation to get the gradients
     - Update parameters (gradient descent)