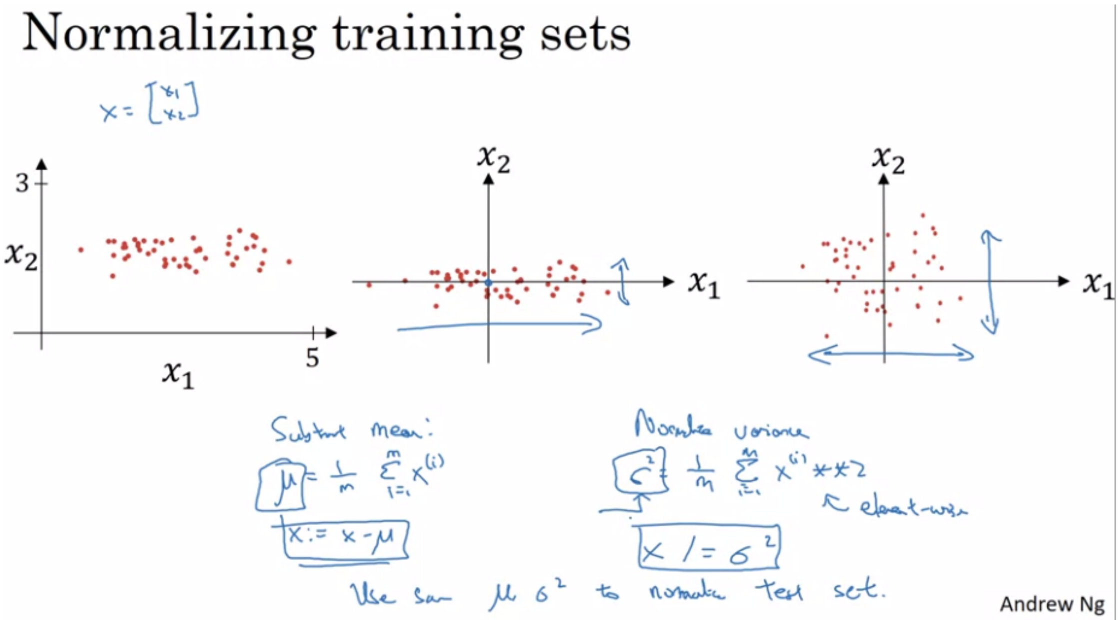
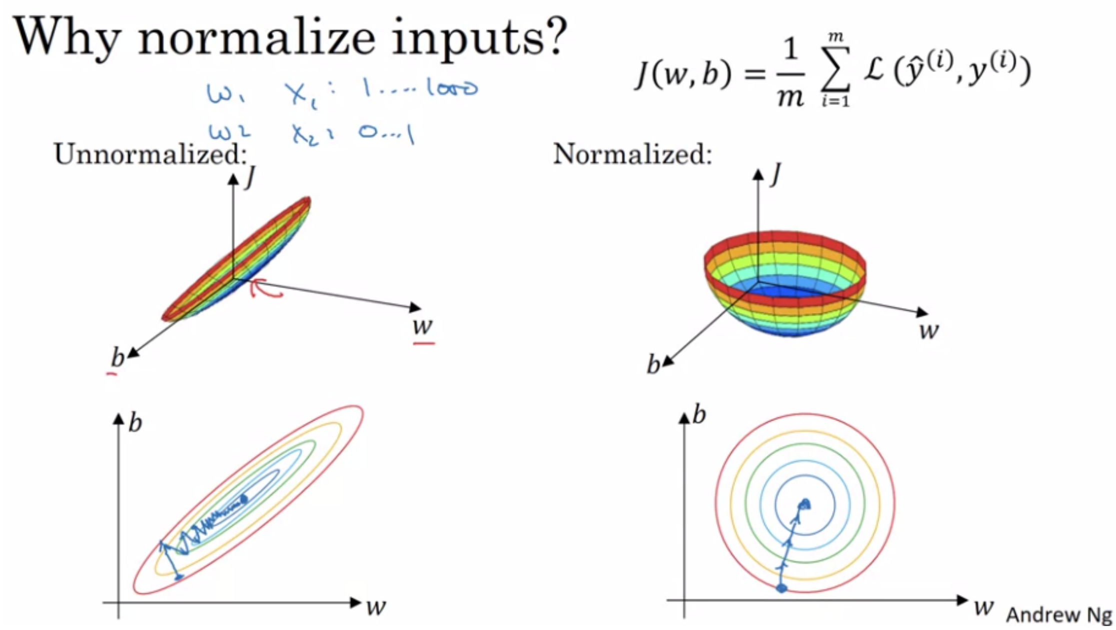
Week 1-3 Problem Set-up (Normalization, Initialization, G checking)

笔记本:	DL 2 - Deep NN Hyperparameter Tunning, Regularization & Optimization		
创建时间:	2021/1/9 09:43	**更新时间:**	2021/1/9 10:17



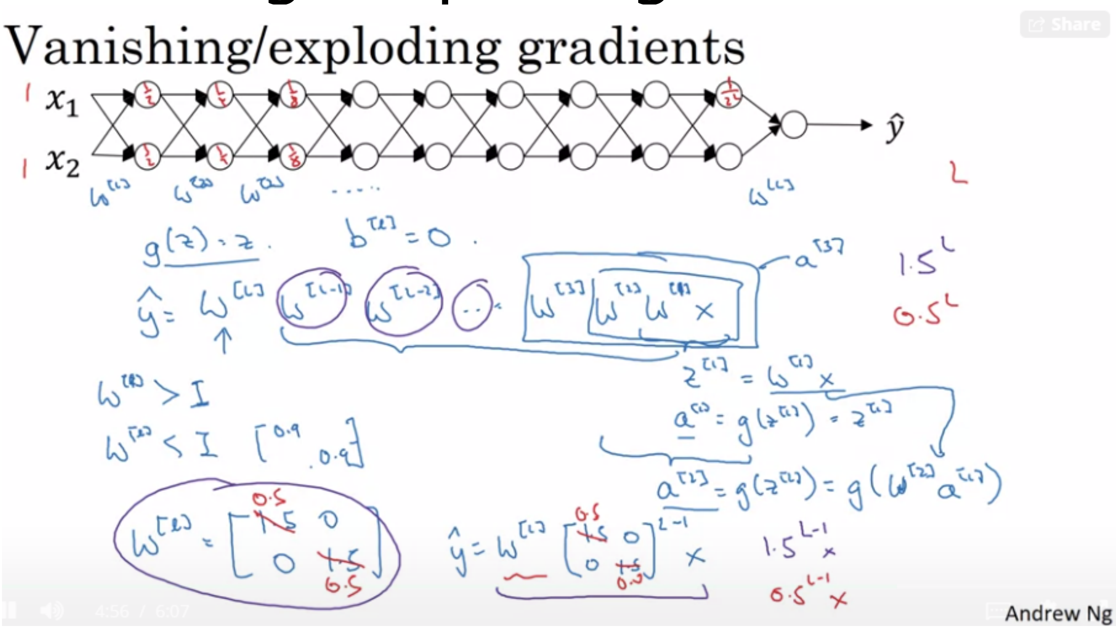
also do with test set, (divide by sigma rather than sigma^2)

Why normalize?



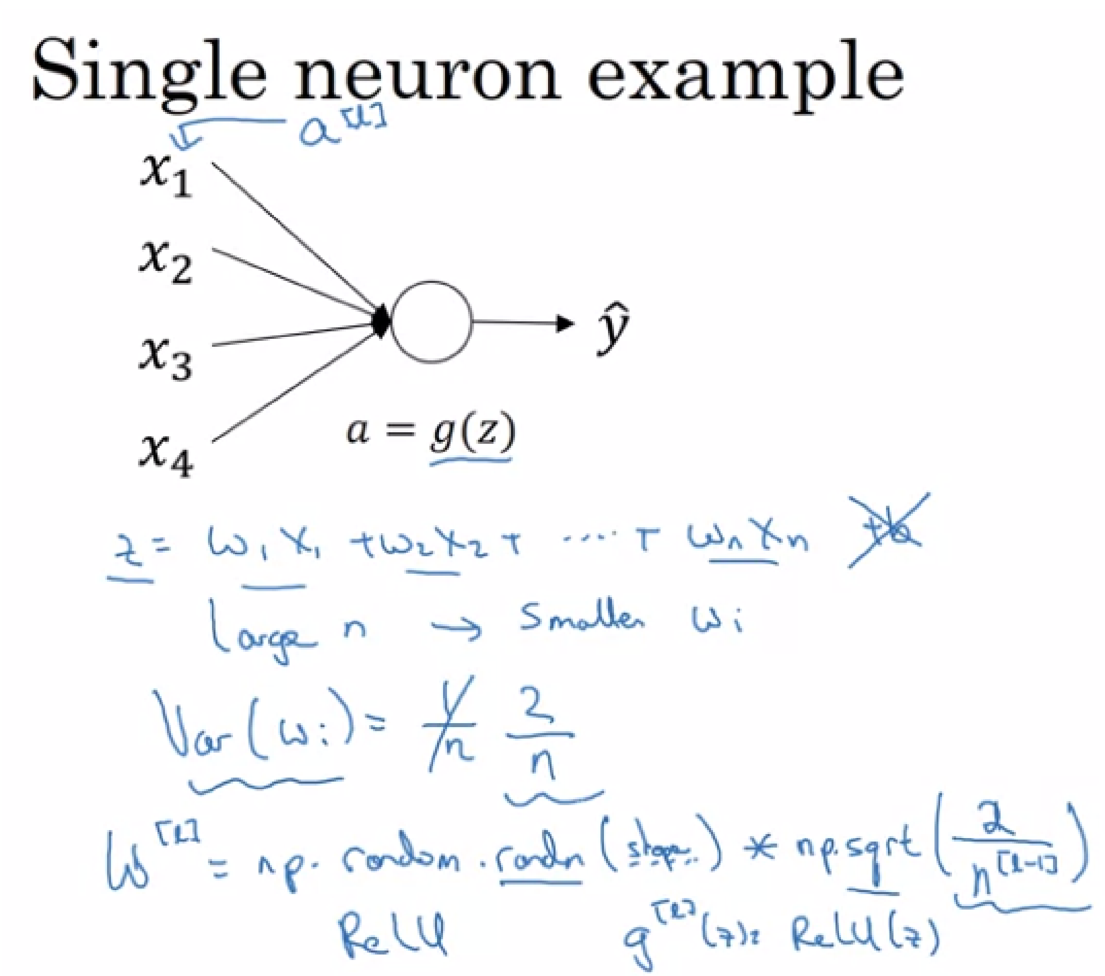
if you're running gradient descent on the cost function like the one on the left, then you might have to use a very **small learning rate** because if you're here that gradient descent might need a lot of steps to oscillate back and forth before it finally finds its way to the minimum. Whereas if you have a more spherical contours, then wherever you start gradient descent can pretty much go straight to the minimum. You can take much **larger steps** with gradient descent rather than needing to oscillate around like like the picture on the left.

Vanishing / Exploding Gradients



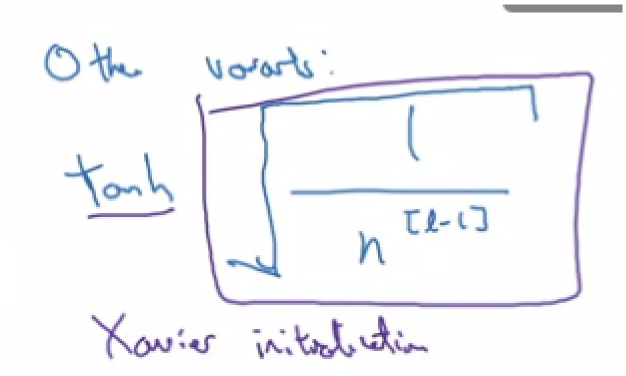
(dW will also have such extreme problems)

Weight Initialization for NN



ReLU: He Initialization, sqrt(2/n^([l-1))

Xavier Initialization for tanh

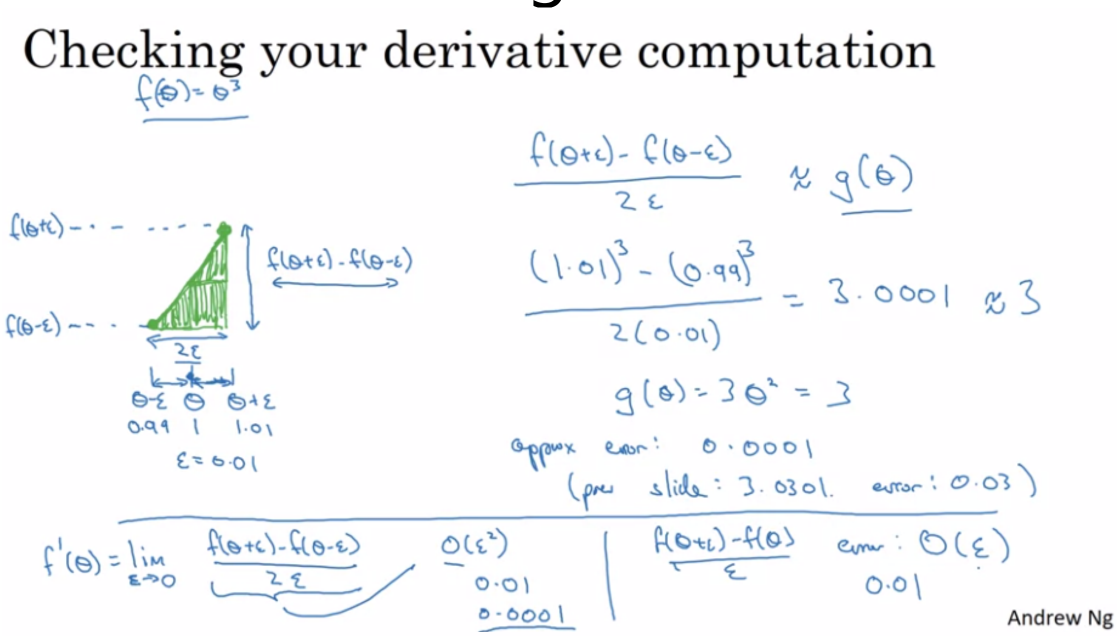


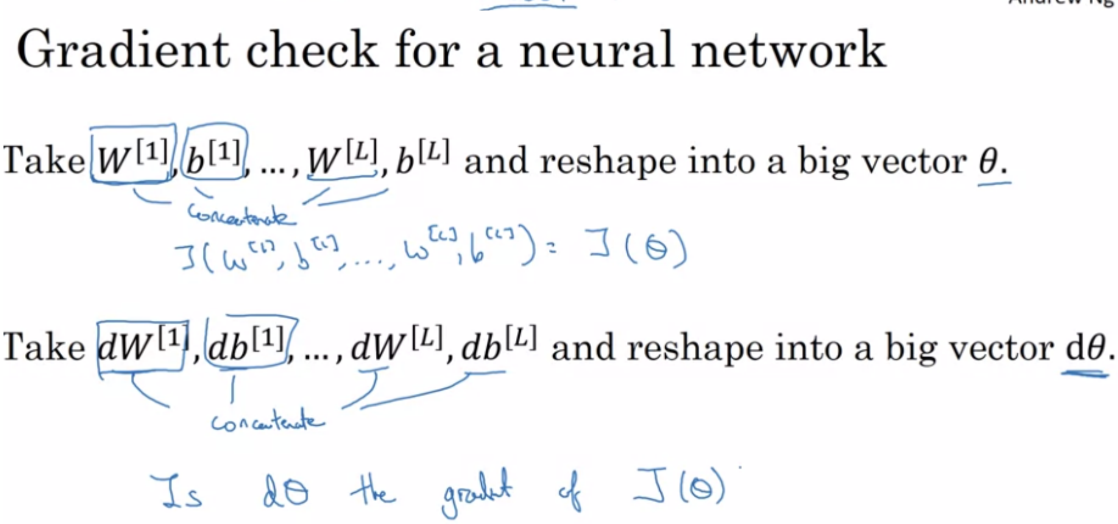
Let's derive Xavier Initialization now, step by step.

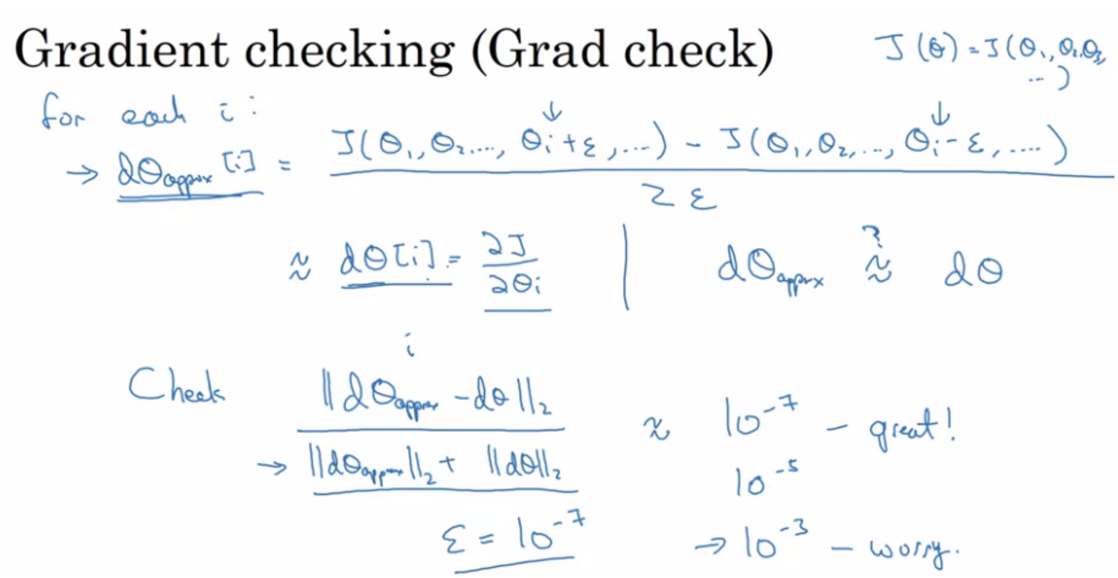
Our full derivation gives us the following initialization rule, which we apply to all weights:

$$W_{i,j}^{[l]} = \mathcal{N}\left(0, \frac{1}{n^{[l-1]}}\right)$$

Gradient Checking

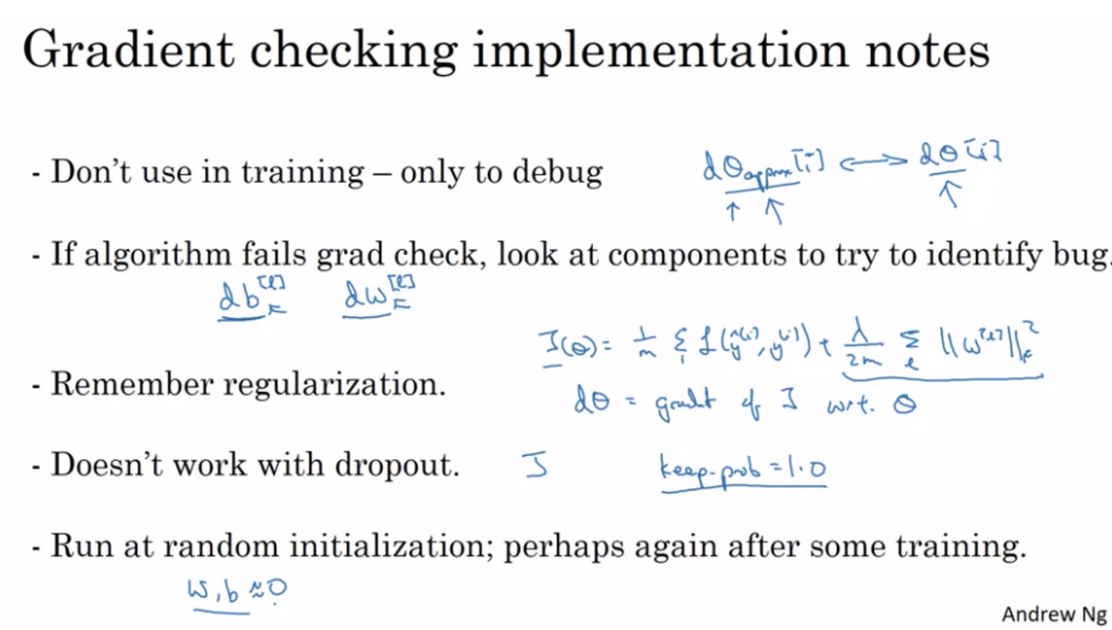






10^{-7} -> great / 10^{-5} / 10^{-3} -> wrong

Some other notes:
turn-off GC in training (only do this in debugging)



it's not impossible that your implementation of gradient descent is **correct when w and b are close to 0**, so at random initialization. But that as you run gradient descent and w and b become bigger, maybe your implementation of backprop is correct only when w and b is close to 0, but it gets more inaccurate when w and b become large. So one thing you could do, I don't do this very often, but one thing you could do is run grad check **at random initialization and then train the network for a while so that w and b have some time to wander away from 0,** from your small random initial values. And then run grad check again after you've trained for some number of iterations.