Neural Networks

1. **Representation**

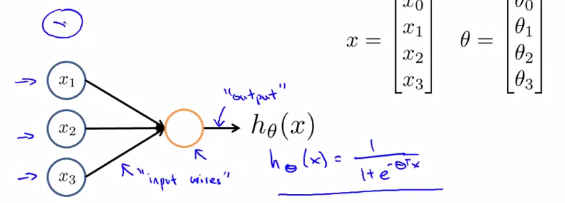
Origins: Algorithms that try to mimic the brain.

Was very widely used in 80s and early 90s; popularity diminised in late 90s

Recent resurgence: State-of-the-art technique for many applications

* 1. **Model representation:**

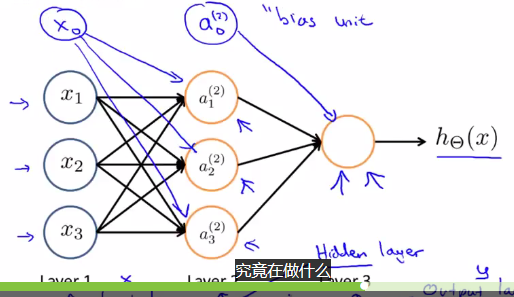
Neuron model : logistic unit, Sigmoid (logistic ) activation function.

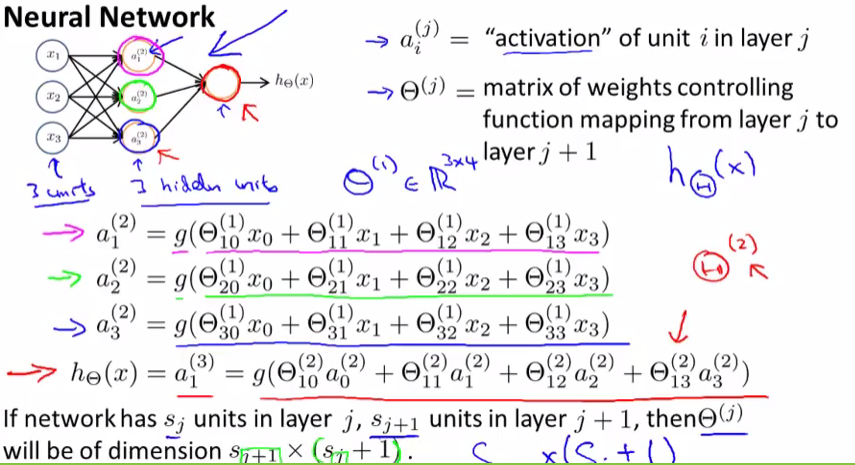


Another common choice for activation function is the hyperbolic tangent, or tanh,function:

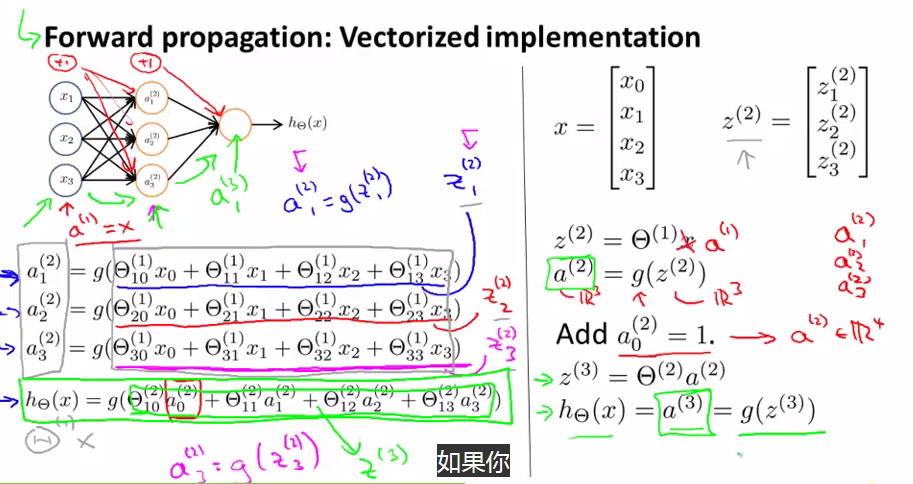
C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(1).png

* 1. **Neural Network**



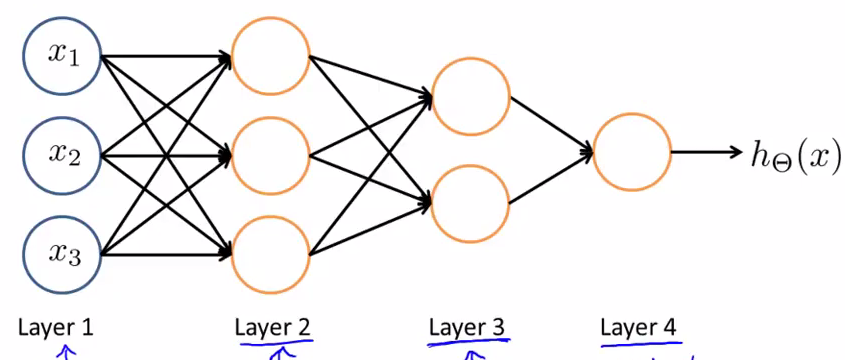


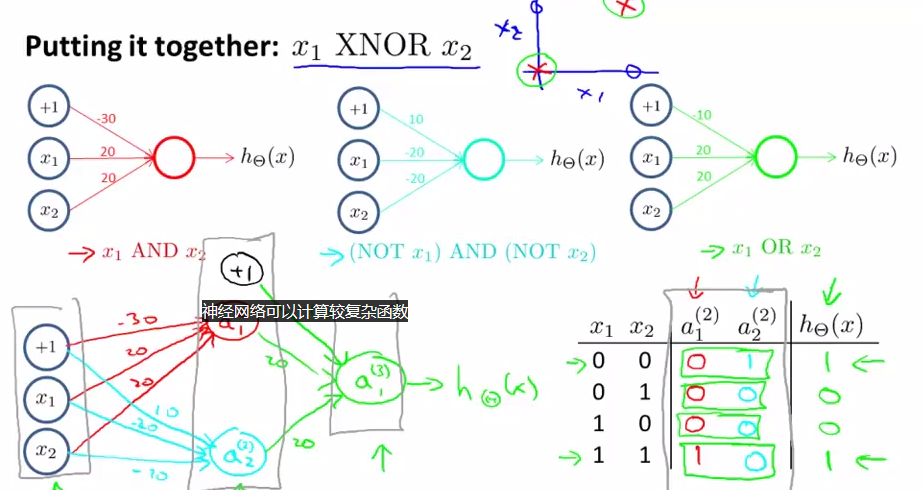
Forward propagation : Vectorized implementaion



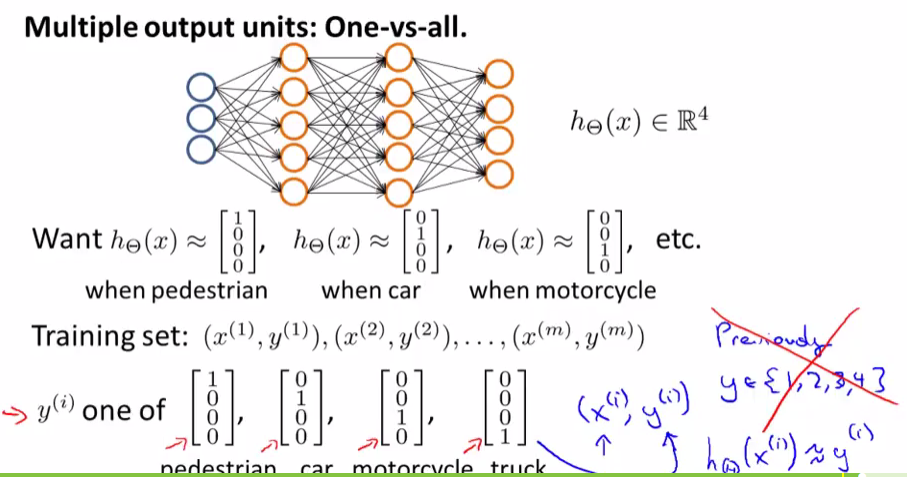
Neural Network learning its own features

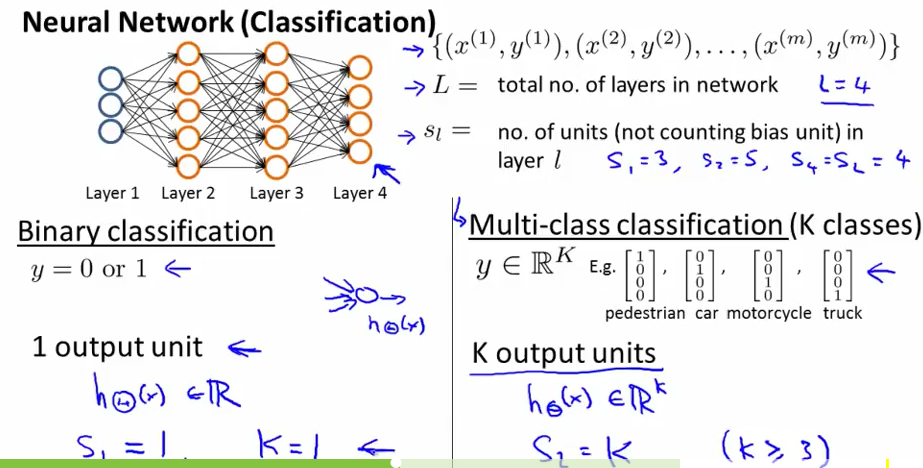
Other network architectures

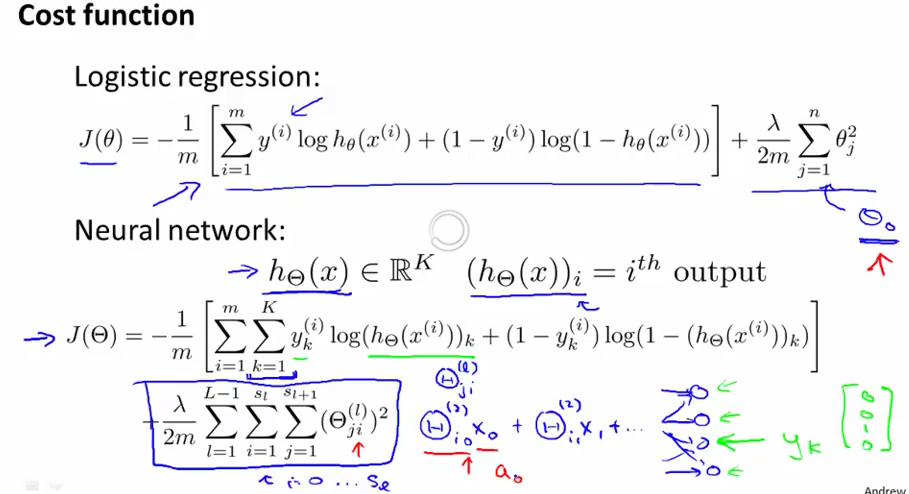




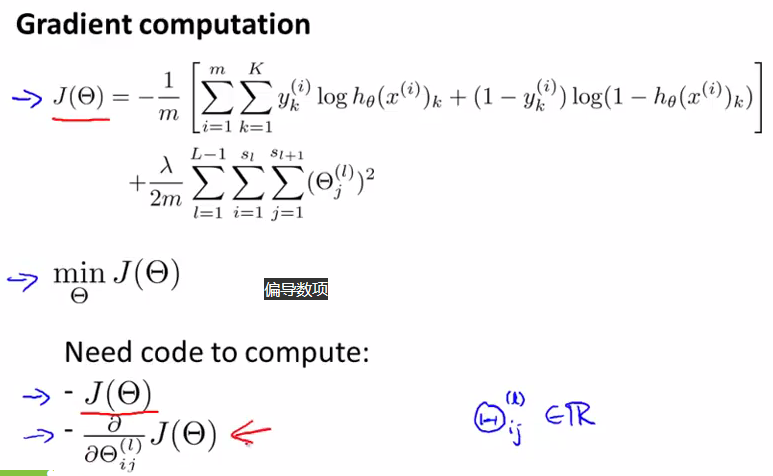
Multiclass Classification

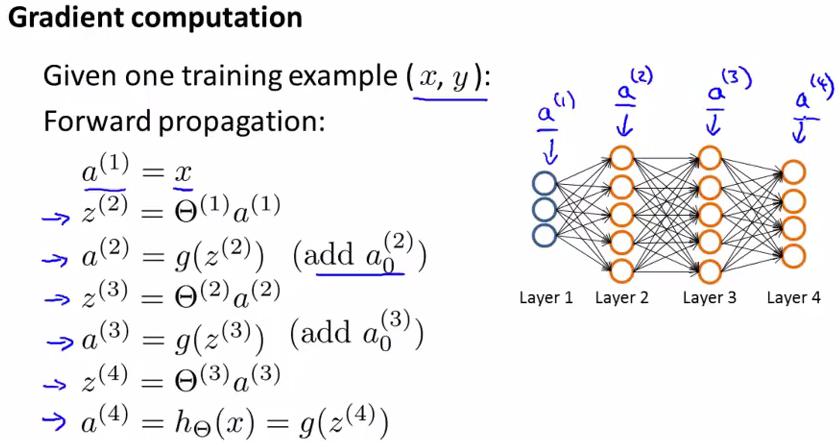


1. **Learning**

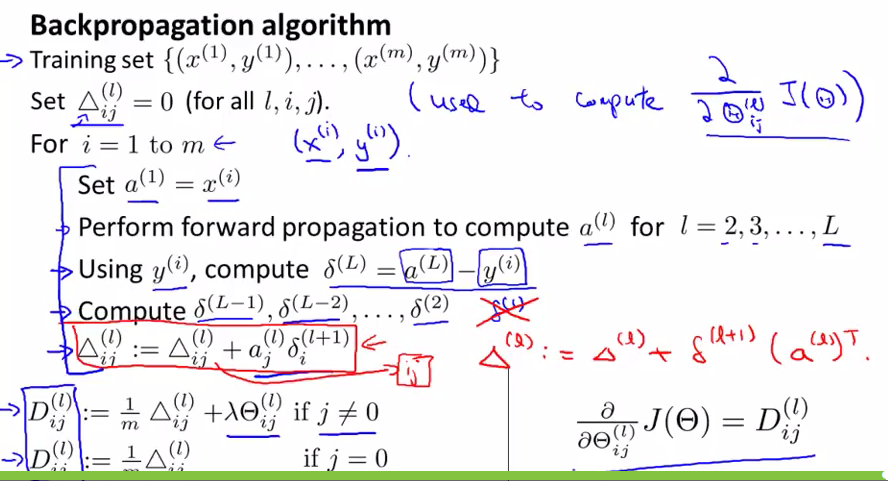


* 1. **Backpropagation Algorithm**

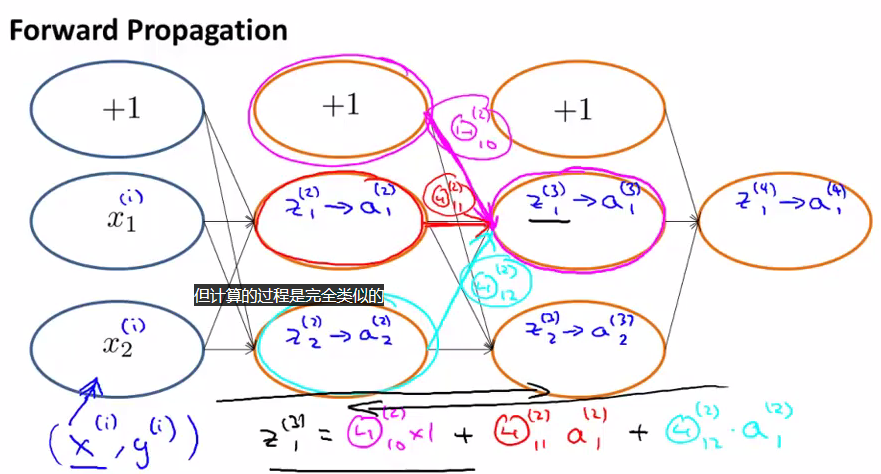


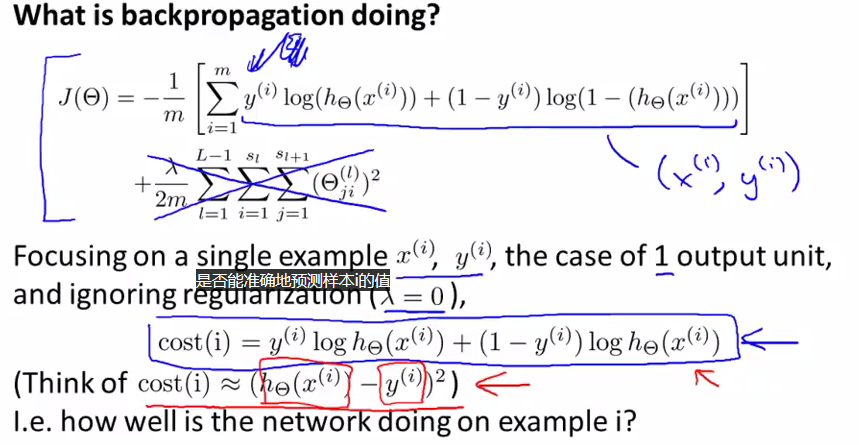


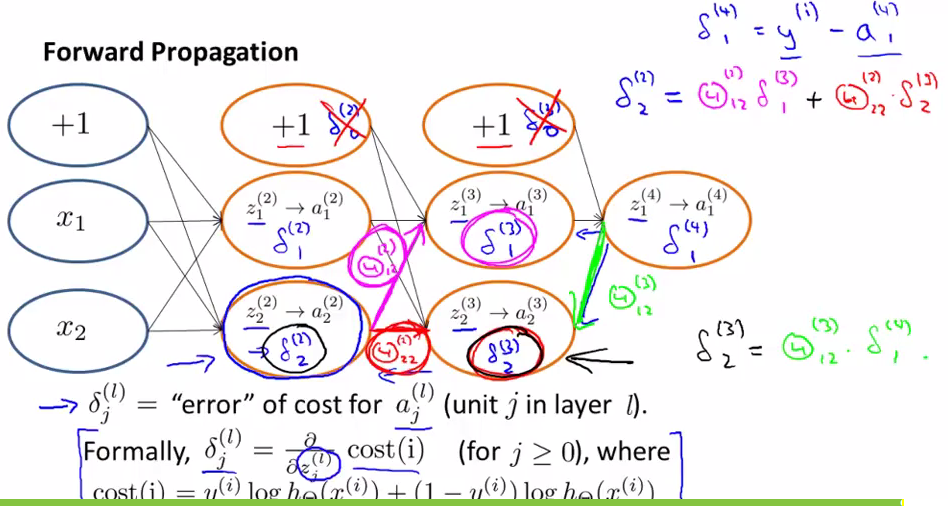




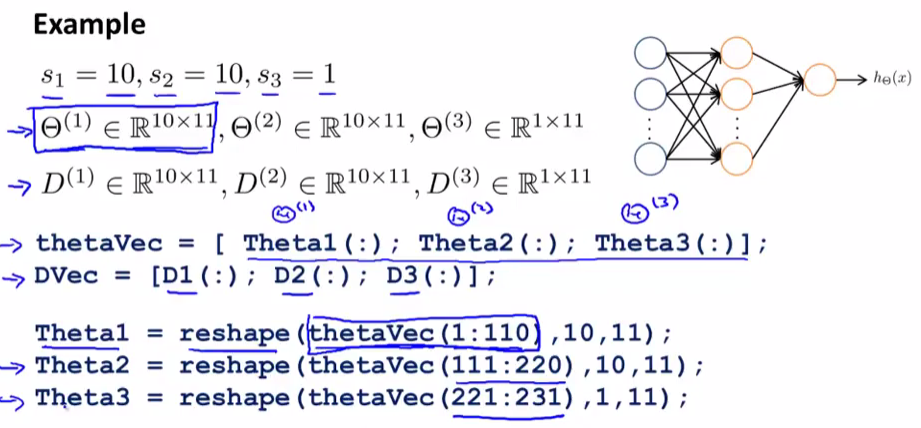
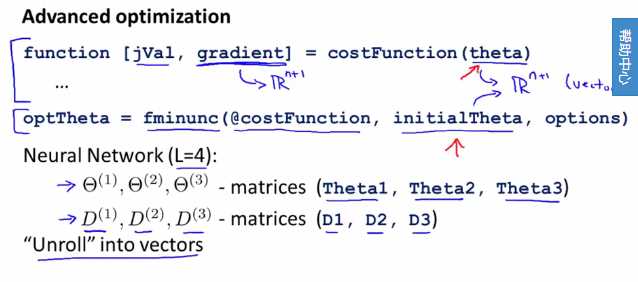
Backpropagation Intuition

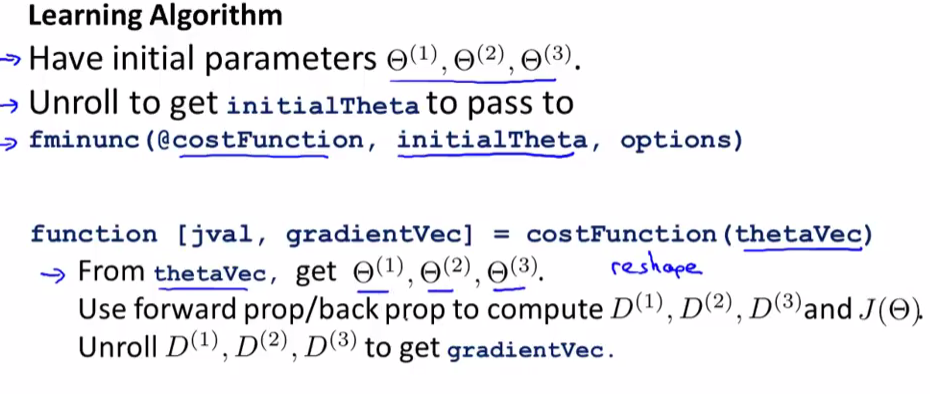




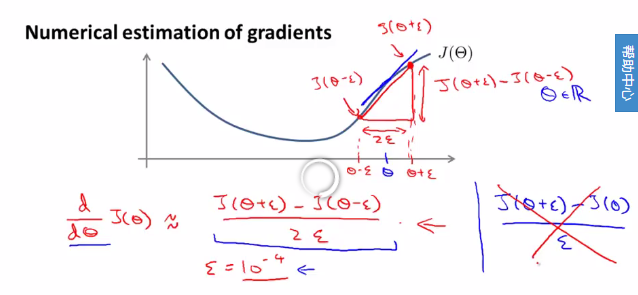


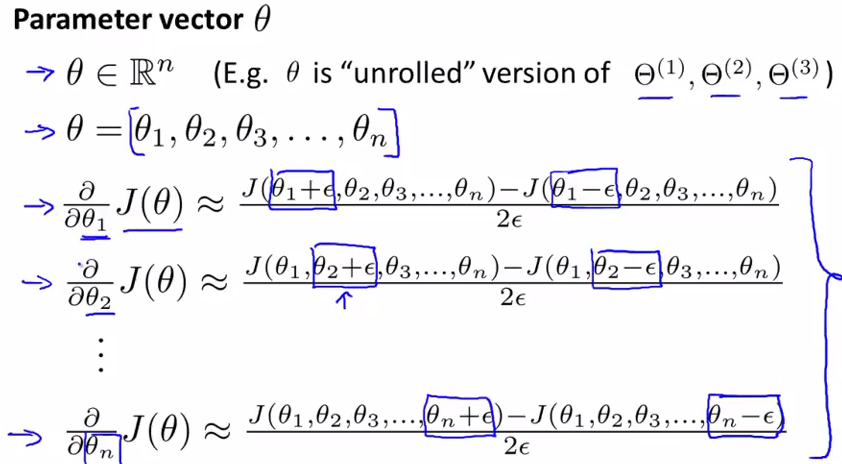
**2.2 Implementation Note : Unrolling Parameters**

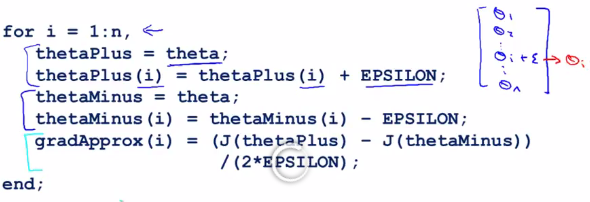




**2.3 Gradient Checking**







Check that grad Approx ~~Dvec

**2.4 Implementation Note:**

- Implementation backprop to compute DVec (Unrolled D(1),D(2),D(3))

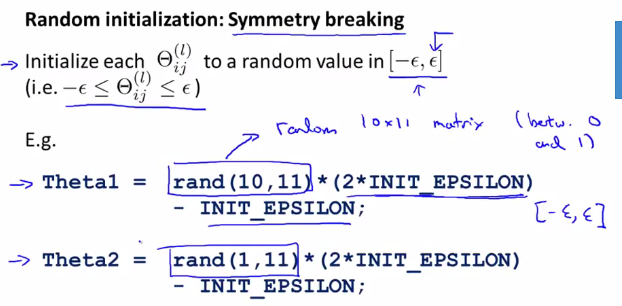
- Implement numerical gradient chekc to compute gradApprox

- Make sure they give similar values

- Turn off gradient checking. Using backprop code for learning.

- Be sure to disable your gradient checking code before training your classifier. If you run numerical gradient computation on every iteration of gradient descent (or in the inner loop of costFunction(...)) your code will be very slow.

**2.5 Random Initialization**



Putting it Together

