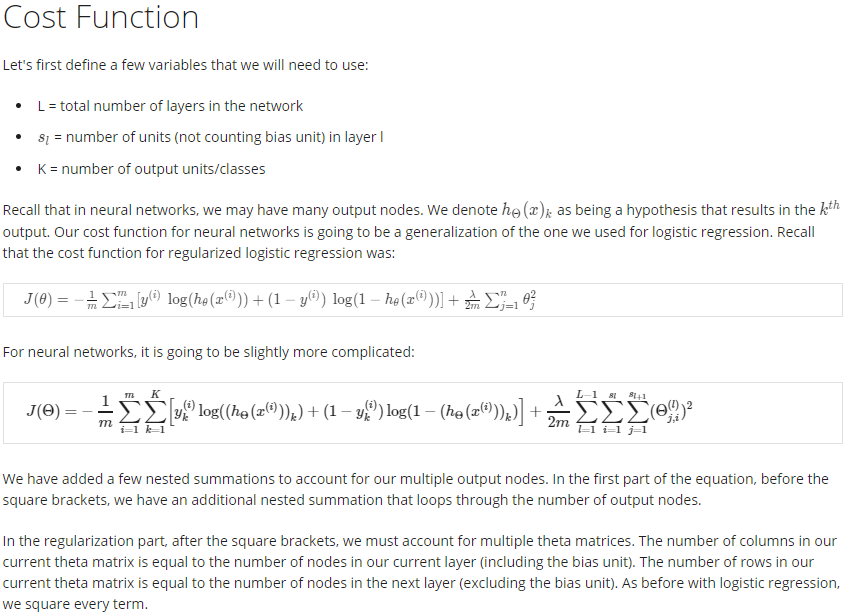
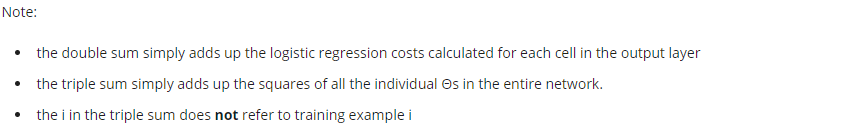
# NEURAL NETWORKS: LEARNING

## Cost Function and Backpropagation

*Cost Function*

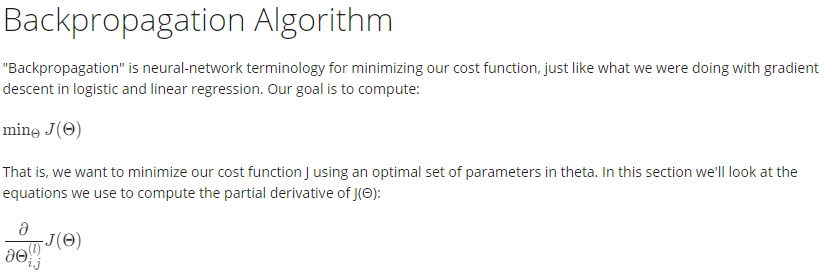
* Neural networks (classification): L, sl
* Binary
* SL = K = 1
* Multi-class
* SL = K >= 3
* Cost Function
* Logistic regression vs. neural networks

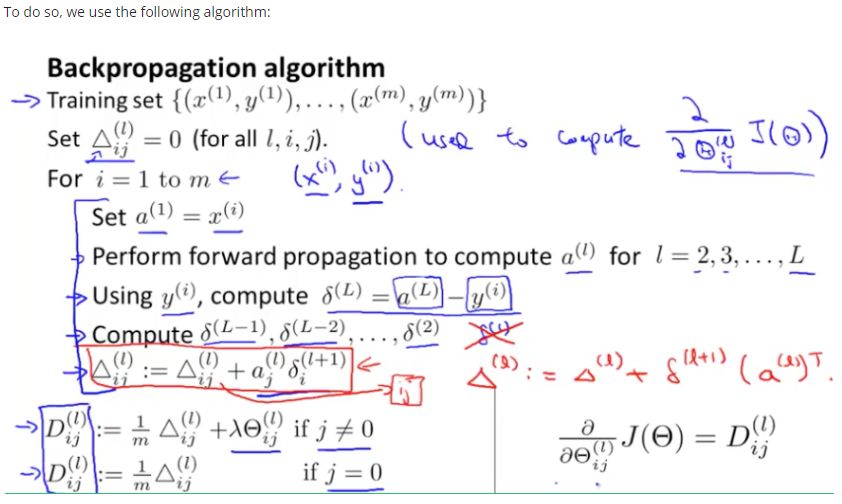


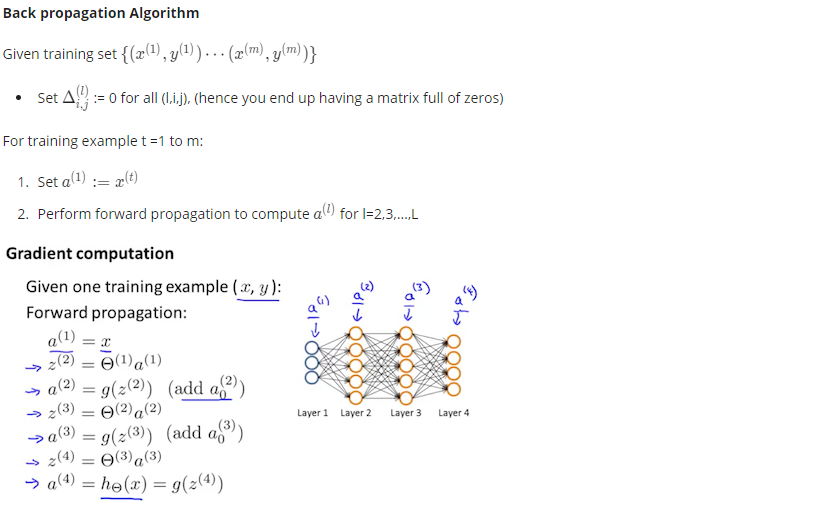


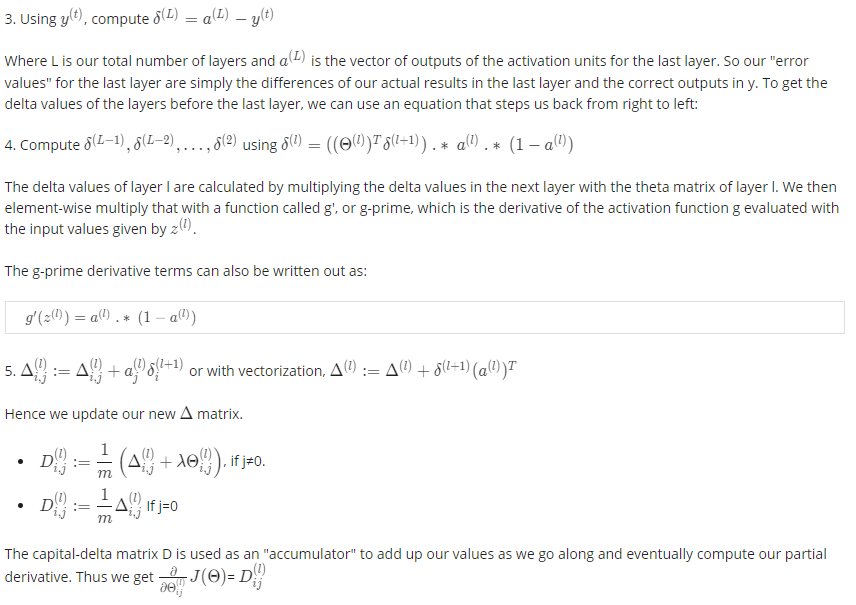
*Backpropagation Algorithm*

* Gradient computation
* *One training example at a time?: FP then BP*





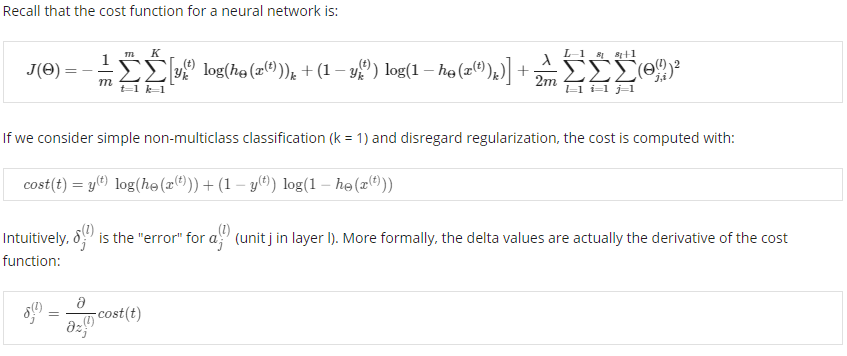


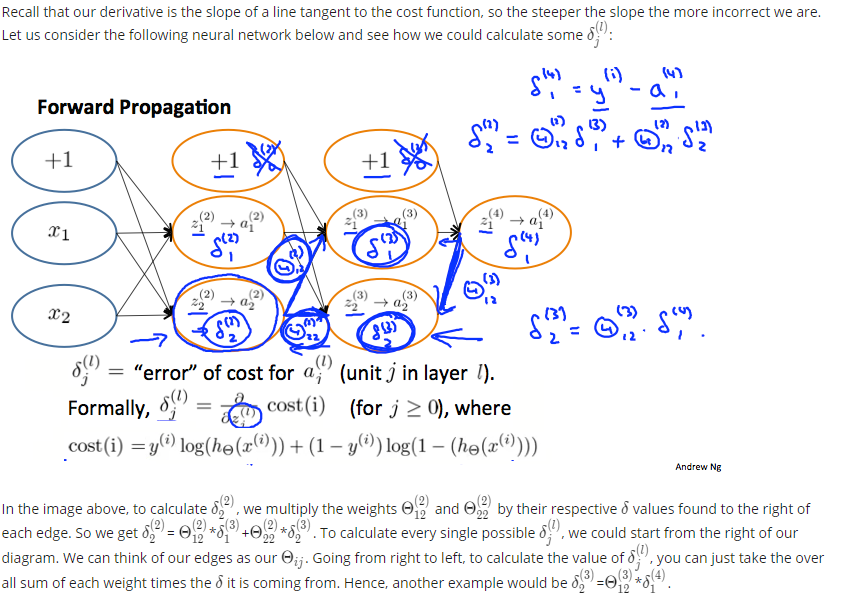


*Backpropagation Intuition*

* Error of cost for unit j in layer l

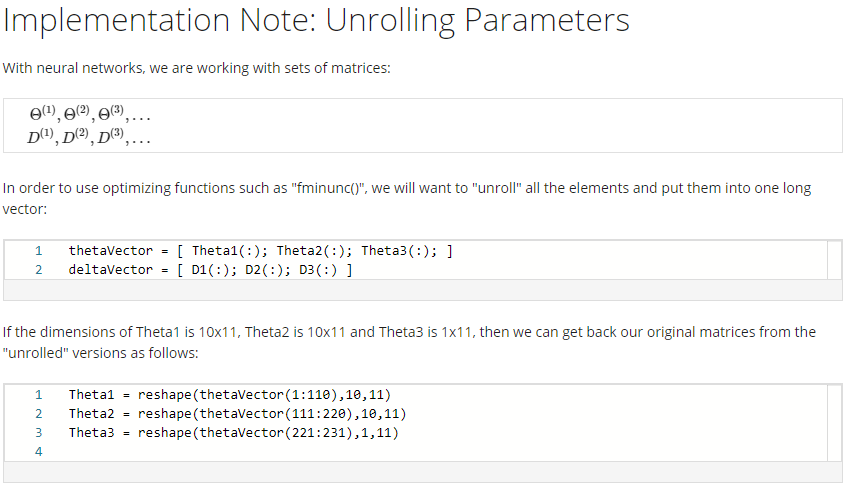


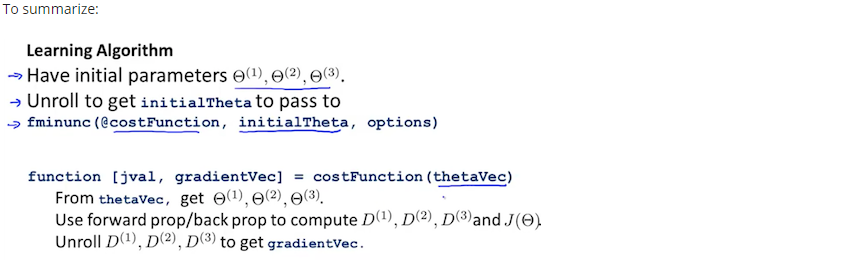




## Backpropagation in Practice

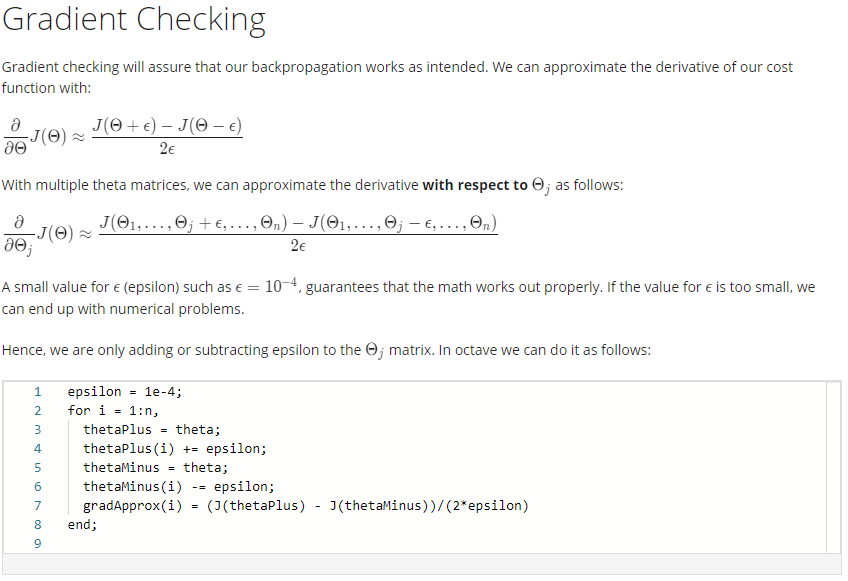
*Implementation Note: Unrolling Parameters*

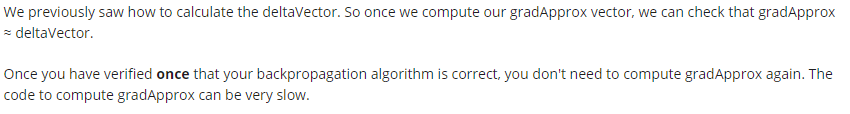




*Gradient Checking*

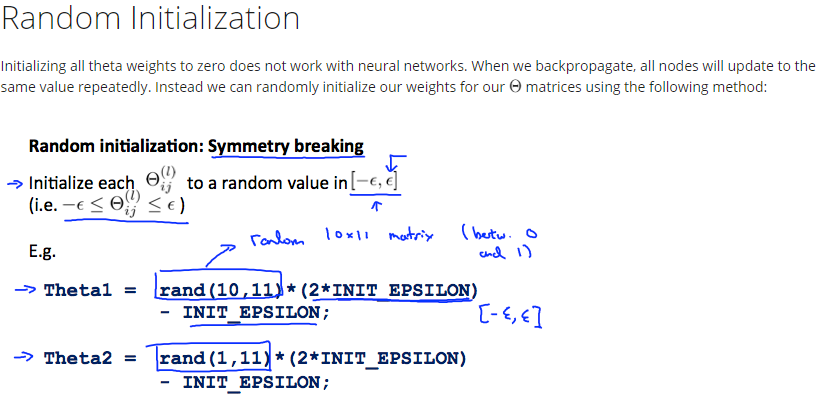
* Numerical estimation of gradients
* gradApprox ~ DVec
* *Why not use gradApprox?*
* Computationally expensive; backprop more efficient

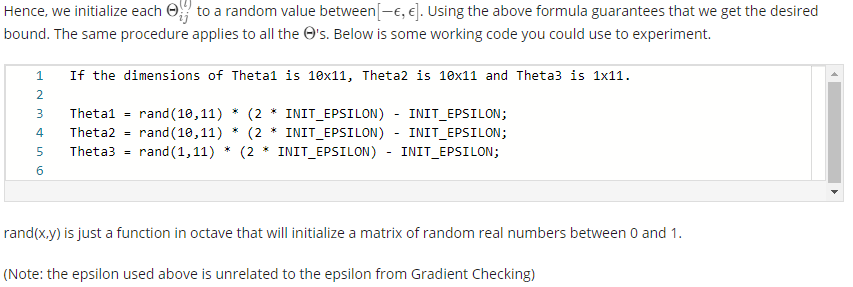




*Random Initialization*

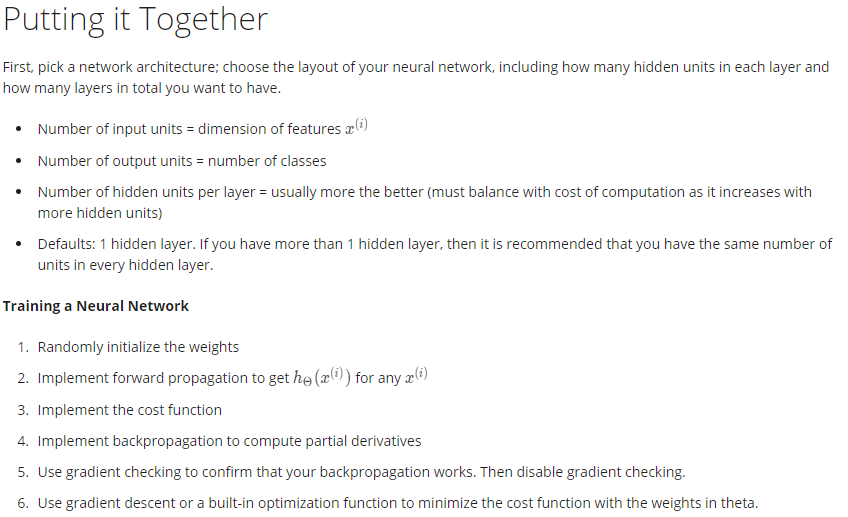
* Zero initialization works fine with Logistic regression, but not with neural networks
* Problem of symmetric weights: equal activation functions; highly redundant; output unit would see one feature
* Random initialization: symmetry breaking

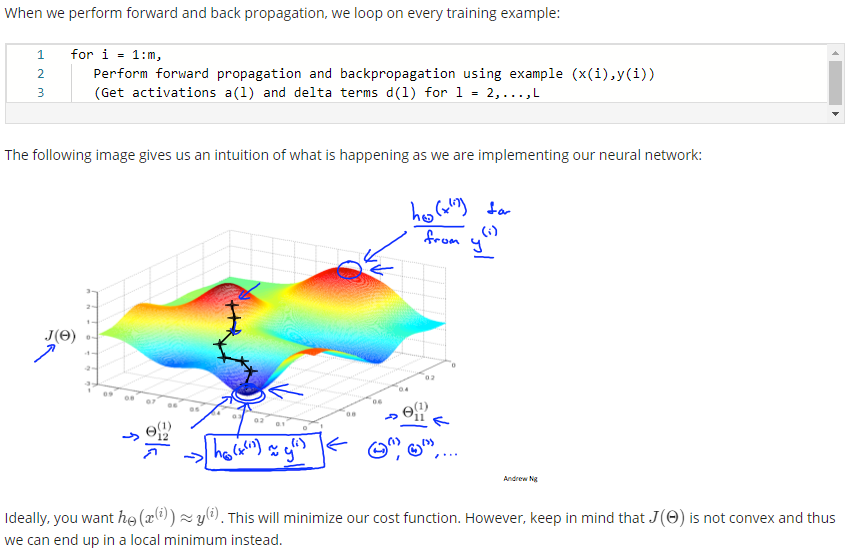




*Putting it Together*

* Training a neural network
* Pick network architecture
* 1 hidden layer default and most common
* If >1, same no. of hidden units
* The more hidden units the better unless too computationally costly
* No. of hidden units comparable to no. of input features
* Basic choice guidelines; more to come
* Random weight init
* FP
* J computing code
* BP
* Grad checking; disable
* min J (non-convex problem)
* local minima: hypothesis fitting





## Applications of Neural Networks

*Autonomous Driving*