Implementation Note: Unrolling Parameters

With neural networks, we are working with sets of matrices:

$$\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}, \dots$$
$$D^{(1)}, D^{(2)}, D^{(3)}, \dots$$

In order to use optimizing functions such as "fminunc()", we will want to "unroll" all the elements and put them into one long vector:

```
thetaVector = [ Theta1(:); Theta2(:); Theta3(:); ]
deltaVector = [ D1(:); D2(:); D3(:) ]
```

If the dimensions of Theta1 is 10x11, Theta2 is 10x11 and Theta3 is 1x11, then we can get back our original matrices from the "unrolled" versions as follows:

```
Theta1 = reshape(thetaVector(1:110),10,11)
Theta2 = reshape(thetaVector(111:220),10,11)
Theta3 = reshape(thetaVector(221:231),1,11)

```

To summarize:

Learning Algorithm

→ Have initial parameters $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$.
→ Unroll to get `initialTheta` to pass to
→ `fminunc(@costFunction, initialTheta, options)`

`function [jval, gradientVec] = costFunction(thetaVec)`
From `thetaVec`, get $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$.
Use forward prop/back prop to compute $D^{(1)}, D^{(2)}, D^{(3)}$ and $J(\Theta)$.
Unroll $D^{(1)}, D^{(2)}, D^{(3)}$ to get `gradientVec`.

10% off Select Top Specializations **10** Coupons