

## Implementation Note: Unrolling Parameters

### Advanced optimization

Let's say we've implemented a cost function that takes the parameters  $\theta$  and returns the cost function and returns derivatives, then we can pass this to an advanced optimization algorithm by `fminunc` (the function `fminunc` takes the `costFunction` and initial  $\theta$  values). Both routines assume that  $\theta$  and the initial value of  $\theta$  are parameters vector ( $\mathbb{R}^{n+1}$ ), and it also assumes that, our cost function will return as a second value the gradient which is also a vector ( $\mathbb{R}^{n+1}$ ).

```
function [jVal, gradient] = costFunction(theta)
...
optTheta = fminunc(@costFunction, initialTheta, options)
```

For NNs, our parameters are matrices e.g.

Neural Network (L=4):

$\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$  - matrices (**Theta1, Theta2, Theta3**)

$D^{(1)}, D^{(2)}, D^{(3)}$  - matrices (**D1, D2, D3**)

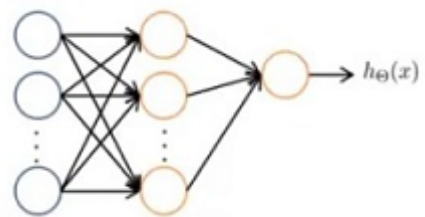
“Unroll” into vectors

### Example

$$s_1 = 10, s_2 = 10, s_3 = 1$$

$$\Theta^{(1)} \in \mathbb{R}^{10 \times 11}, \Theta^{(2)} \in \mathbb{R}^{10 \times 11}, \Theta^{(3)} \in \mathbb{R}^{1 \times 11}$$

$$D^{(1)} \in \mathbb{R}^{10 \times 11}, D^{(2)} \in \mathbb{R}^{10 \times 11}, D^{(3)} \in \mathbb{R}^{1 \times 11}$$



We can use the `thetaVec = [ Theta1(:); Theta2(:); Theta3(:)]`; notation to unroll the  $\theta$  matrices into a long vector.

And we can also use the `DVec = [ D1(:); D2(:); D3(:)]`; notation to unroll the  $D$  matrices into a big long vector.

To go back the  $\theta$  matrices we can use:

```
Theta1 = reshape(thetaVec(1:110), 10, 11);
```

```
Theta2 = reshape(thetaVec(111:220), 10, 11);
```

```
Theta3 = reshape(thetaVec(221:231), 1, 11);
```

**Video Question:** Suppose  $D1$  is a  $10 \times 6$  matrix and  $D2$  is a  $1 \times 11$  matrix. You set:

```
DVec = [D1(:); D2(:)];
```

Which of the following would get  $D2$  back from `DVec`?

- `reshape(DVec(60:71), 1, 11)`
- `reshape(DVec(61:72), 1, 11)`

```
reshape(DVec(61:71), 1, 11)
```

- `reshape(DVec(60:70), 11, 1)`

## Summary

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 10 x 11, Theta2 is 10 x 11 and Theta3 is 1 x 11, 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`.