Advanced Optimization

```
function [jVal, grad] = costFunction(theta)
```

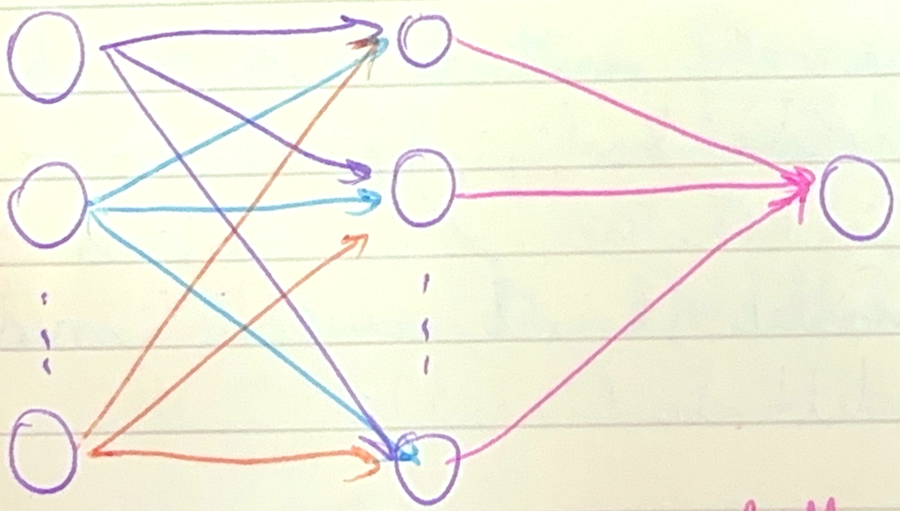
→ Vectors $\mathbb{R}^n$ or $\mathbb{R}^{n+1}$

```
optTheta = fminunc(@costFunction, ...
                   initialTheta, options)
```

temos que "unroll" into vectors as matrizes: $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}, D1, D2, D3$

Exemplo Encurtado:

$s(1) = 10$ $s(2) = 10$ $s(3) = 1$



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

$D1 \in \mathbb{R}^{10 \times 11}$ $D2 \in \mathbb{R}^{1 \times 11}$

Sugestão do prof. Andrew:

```
thetaVec = [Theta1(:); Theta2(:)]
DVec = [D1(:); D2(:)]
```

- Depois:

```
Theta1 = reshape(thetaVec(1:110),10,11);
Theta2 = reshape(thetaVec(111:121),1,11);
```

Achei meio braçal... but... will do.

```
function [jVal, gradientVec] = ...
    costFunction(thetaVec)
```

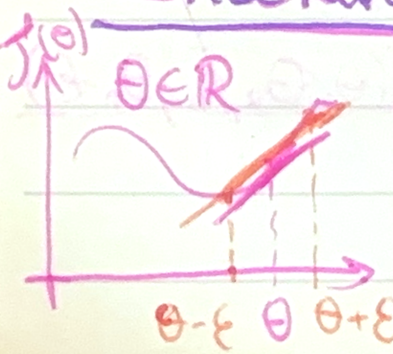
→ from thetaVec, reshape $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$

→ then compute $D^{(1)}, D^{(2)}, D^{(3)}$, and $J(\Theta)$

→ unroll $D^{(1)}, D^{(2)}, D^{(3)}$ to get gradientVec

Checking Gradients

Ng → $\varepsilon = 10^{-4}$



$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$$