Week 5.2 Backpropagation in Practice

Unrolling Parameters

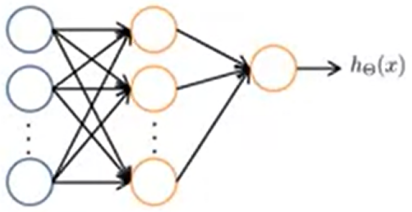
- unroll a matrix into a big long vector

Example

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

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

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




```
thetaVec = [ Theta1(:); Theta2(:); Theta3(:)];
DVec = [D1(:); D2(:); D3(:)];

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

Gradient Checking

- two-side difference:

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

- one-side difference:

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

- check $gradApprox \approx Dvec$

Random Initialization (Symmetry breaking)

- Initialize $\Theta_{ij}^{(l)}$ to a random value in $[-\epsilon, \epsilon]$

Putting it Together

Training a Neural Network

1. Randomly initialize the weights
2. Implement forward propagation to get hΘ(x(i)) for any x(i)

3. Implement the cost function
4. Implement backpropagation to compute partial derivatives
5. Use gradient checking to confirm that your backpropagation works. Then disable gradient checking.
6. Use gradient descent or a built-in optimization function to minimize the cost function with the weights in theta