

Cost Function and Backpropagation

- ✓ **Video:** Cost Function
6 min
- ✓ **Reading:** Cost Function
4 min
- ▶ **Video:** Backpropagation Algorithm
11 min
- ✓ **Reading:** Backpropagation Algorithm
10 min
- ▶ **Video:** Backpropagation Intuition
12 min
- ✓ **Reading:** Backpropagation Intuition
4 min

Backpropagation in Practice

- ▶ **Video:** Implementation
Note: Unrolling Parameters
7 min
- ✓ **Reading:** Implementation
Note: Unrolling Parameters
3 min
- ▶ **Video:** Gradient Checking
11 min
- ✓ **Reading:** Gradient Checking
3 min
- ▶ **Video:** Random Initialization
6 min
- 📖 **Reading:** Random Initialization
3 min
- ▶ **Video:** Putting It Together
13 min
- 📖 **Reading:** Putting It Together
4 min

Application of Neural Networks

Review

Random Initialization

Initializing all theta weights to zero does not work with neural networks. When we backpropagate, all nodes will update to the same value repeatedly. Instead we can randomly initialize our weights for our Θ matrices using the following method:

Random initialization: Symmetry breaking

→ Initialize each $\Theta_{ij}^{(l)}$ to a random value in $[-\epsilon, \epsilon]$
(i.e. $-\epsilon \leq \Theta_{ij}^{(l)} \leq \epsilon$)

E.g.

→ **Theta1** = $\text{rand}(10, 11) * (2 * \text{INIT_EPSILON}) - \text{INIT_EPSILON};$ [$-\epsilon, \epsilon$]

→ **Theta2** = $\text{rand}(1, 11) * (2 * \text{INIT_EPSILON}) - \text{INIT_EPSILON};$

Hence, we initialize each $\Theta_{ij}^{(l)}$ to a random value between $[-\epsilon, \epsilon]$. Using the above formula guarantees that we get the desired bound. The same procedure applies to all the Θ 's. Below is some working code you could use to experiment.

```
1 If the dimensions of Theta1 is 10x11, Theta2 is 10x11 and Theta3 is 1x11.
2
3 Theta1 = rand(10,11) * (2 * INIT_EPSILON) - INIT_EPSILON;
4 Theta2 = rand(10,11) * (2 * INIT_EPSILON) - INIT_EPSILON;
5 Theta3 = rand(1,11) * (2 * INIT_EPSILON) - INIT_EPSILON;
6
```

rand(x,y) is just a function in octave that will initialize a matrix of random real numbers between 0 and 1.

(Note: the epsilon used above is unrelated to the epsilon from Gradient Checking)