



## 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

## Application of Neural Networks

## Review



# Cost Function

Let's first define a few variables that we will need to use

- $L$  = total number of layers in the network
- $s_l$  = number of units (not counting bias unit) in layer  $l$
- $K$  = number of output units/classes

Recall that in neural networks, we may have many output hypotheses that result in the  $k^{th}$  output. Our cost function is a generalization of the one we used for logistic regression. The cost function for logistic regression was:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

For neural networks, it is going to be slightly more complicated.

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log((h_{\Theta}(x^{(i)}))_k) + (1 - y_k^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))_k)$$

We have added a few nested summations to account for the equation. Before the square brackets, we have a summation through the number of output nodes.

In the regularization part, after the square brackets, we have a summation over the number of columns in our current theta matrix is equal to the number of nodes in the next layer (including the bias unit). The number of rows in our current theta matrix is equal to the number of nodes in the previous layer (excluding the bias unit). We square every term.

Note:

- the double sum simply adds up the logistic regression cost for each output layer
- the triple sum simply adds up the squares of all the parameters in the network