Cost Function and Backpropagation

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

- **Video:** Autonomous Driving 6 min

Review

Putting it Together

First, pick a network architecture; choose the layout of your neural network, including how many hidden units in each layer and how many layers in total you want to have.

- Number of input units = dimension of features $x^{(i)}$
- Number of output units = number of classes
- Number of hidden units per layer = usually more the better (must balance with cost of computation as it increases with more hidden units)
- Defaults: 1 hidden layer. If you have more than 1 hidden layer, then it is recommended that you have the same number of units in every hidden layer.

Training a Neural Network

1. Randomly initialize the weights
2. Implement forward propagation to get $h_\Theta(x^{(i)})$ 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