These slides were assembled by Byron Boots, with only minor modifications from Eric Eaton’s slides and grateful acknowledgement to the many others who made their course materials freely available online. Feel free to reuse or adapt these slides for your own academic purposes, provided that you include proper attribution.
Forward Propagation

- Given one labeled training instance \((x, y)\):

Forward Propagation

- \(a^{(1)} = x\)
- \(z^{(2)} = \Theta^{(1)}a^{(1)}\)
- \(a^{(2)} = g(z^{(2)}) \quad \text{[add } a_0^{(2)}\text{]}\)
- \(z^{(3)} = \Theta^{(2)}a^{(2)}\)
- \(a^{(3)} = g(z^{(3)}) \quad \text{[add } a_0^{(3)}\text{]}\)
- \(z^{(4)} = \Theta^{(3)}a^{(3)}\)
- \(a^{(4)} = h_\Theta(x) = g(z^{(4)})\)
Backpropagation: Gradient Computation

Let $\delta_j^{(l)} =$ “error” of node $j$ in layer $l$

Backpropagation
- $\delta^{(4)} = \mathbf{a}^{(4)} - \mathbf{y}$
- $\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} \cdot g'(\mathbf{z}^{(3)})$
- $\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} \cdot g'(\mathbf{z}^{(2)})$
- (No $\delta^{(1)}$)

\[
\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)} \quad \text{(ignoring } \lambda; \text{ if } \lambda = 0)\]

Based on slide by Andrew Ng
Backpropagation

Set $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$

For each training instance $(x_i, y_i)$:

Set $a^{(1)} = x_i$

Compute $\{a^{(2)}, \ldots, a^{(L)}\}$ via forward propagation

Compute $\delta^{(L)} = a^{(L)} - y_i$

Compute errors $\{\delta^{(L-1)}, \ldots, \delta^{(2)}\}$

Compute gradients $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

Compute avg regularized gradient $D_{ij}^{(l)} = \begin{cases} \frac{1}{n} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{n} \Delta_{ij}^{(l)} & \text{otherwise} \end{cases}$

$D^{(l)}$ is the matrix of partial derivatives of $J(\Theta)$

Note: Can vectorize $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ as $\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)} a^{(l)^T}$
Training a Neural Network via Gradient Descent with Backprop

Given: training set \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

Initialize all \( \Theta^{(l)} \) randomly (NOT to 0!)

Loop // each iteration is called an epoch

Set \( \Delta_{ij}^{(l)} = 0 \quad \forall l, i, j \) (Used to accumulate gradient)

For each training instance \((x_i, y_i)\):

Set \( a^{(1)} = x_i \)

Compute \( \{a^{(2)}, \ldots, a^{(L)}\} \) via forward propagation

Compute \( \delta^{(L)} = a^{(L)} - y_i \)

Compute errors \( \{\delta^{(L-1)}, \ldots, \delta^{(2)}\} \)

Compute gradients \( \Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)} \)

Compute avg regularized gradient \( D_{ij}^{(l)} = \begin{cases} \frac{1}{n} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{n} \Delta_{ij}^{(l)} & \text{otherwise} \end{cases} \)

Update weights via gradient step \( \Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha D_{ij}^{(l)} \)

Until weights converge or max #epochs is reached

Based on slide by Andrew Ng
Backprop Issues

“Backprop is the cockroach of machine learning. It’s ugly, and annoying, but you just can’t get rid of it.”
—Geoff Hinton

Problems:
• black box
• local minima
Putting It All Together
Training a Neural Network

Pick a network architecture (connectivity pattern between nodes)

- # input units = # of features in dataset
- # output units = # classes

Reasonable default: 1 hidden layer
- or if >1 hidden layer, have same # hidden units in every layer (usually the more the better)

Based on slide by Andrew Ng
Training a Neural Network

1. Randomly initialize weights
2. Implement forward propagation to get \( h_\Theta(x_i) \) for any instance \( x_i \)
3. Implement code to compute cost function \( J(\Theta) \)
4. Implement backprop to compute partial derivatives
   \[
   \frac{\partial}{\partial \Theta^{(l)}_{jk}} J(\Theta)
   \]
5. Optional: Use gradient checking to compare \( \frac{\partial}{\partial \Theta^{(l)}_{jk}} J(\Theta) \)
6. Use gradient descent with backprop to fit the network

Based on slide by Andrew Ng