Week 05

Neural Networks: Learning

- Cost Function and Backpropagation
- Backpropagation in Practice
- Application of Neural Networks

I. Cost Function and Backpropagation

1. Cost function

- 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
- $h\Theta(x)_k$ as being a hypothesis that results in the k^{th} output
- Cost function for neural networks is going to be a generalization of the one we used for logistic regression.
- Cost function for regularized logistic regression was:

  $J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\ \log(h_\theta(x^{(i)})) + (1-y^{(i)})\ \log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$
- For neural networks, it is going to be slightly more complicated:

  $J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$
  - At first, nested summation that loops through the number of output nodes.
  - In the regularization part, after the square brackets, we must account for multiple theta matrices. The number of columns in our current theta matrix is equal to the number of nodes in our current layer (including the bias unit). The number of rows in our current theta matrix is equal to the number of nodes in the next layer (excluding the bias unit). As before with logistic regression, we square every term.
  - Note:
    - the double sum simply adds up the logistic regression costs calculated for each cell in the output layer
    - the triple sum simply adds up the squares of all the individual Θs in the entire network.
    - the i in the triple sum does **not** refer to training example i

2. Backpropagation algorithm

- "Backpropagation" is neural-network terminology for minimizing the cost function (like gradient descent in logistic and linear regression).
- Our goal is to compute: $\min_\Theta J(\Theta)$m
- Minimize J using an optimal set of parameters in Θ. Let's look at the equations to compute the partial derivative of J(Θ): $\dfrac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta)$
- To do so, we use the following algorithm:
  - Training set: $(x^{(1)}, y^{(1)}), .., (x^{(m)}, y^{(m)})$
  - Set $\Delta_{i,j}^{(l)} = 0$ for all i, j, l
  - For $i = 1$ to m
    - set $a^{(1)} = a^{(i)}$
    - perform FP for $a^{(1)}$ for $l = 2, .., L$
    - using $y^{(i)}$ compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
    - compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)} \text{ and no } \delta^{(1)}$
    - $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta^{(l+1)}$
  - $D_{i,j}^{(l)} = 1/m * \Delta_{i,j}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j != 0$
  - $D_{i,j}^{(l)} = 1/m * \Delta_{i,j}^{(l)}$ if $j = 0$
  - Note: $\dfrac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

  Example:

  Given training example (x, y)

  $a^{(1)} = x$

  $z^{(2)} = \Theta^{(1)} a^{(1)}$

  $a^{(2)} = g(z^{(2)})$ add $a_0^{(2)}$

  $z^{(3)} = \Theta^{(2)} a^{(2)}$

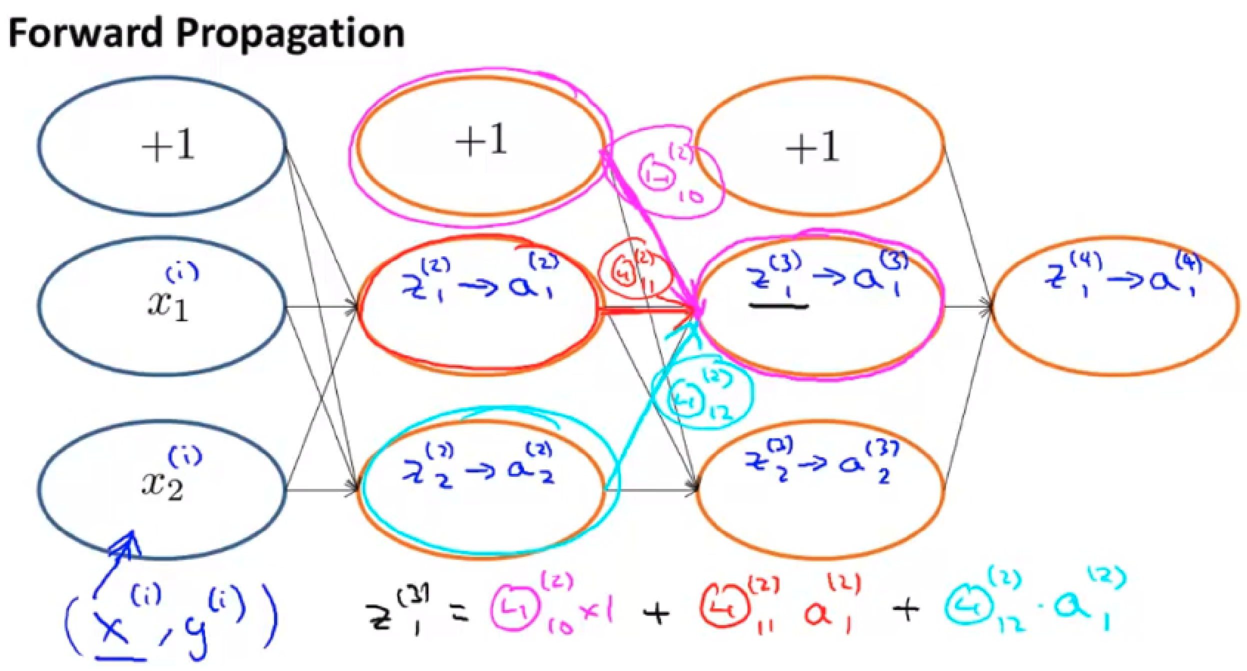
  $a^{(3)} = g(z^{(3)})$ add $a_0^{(3)}$

  $z^{(4)} = \Theta^{(3)} a^{(3)}$

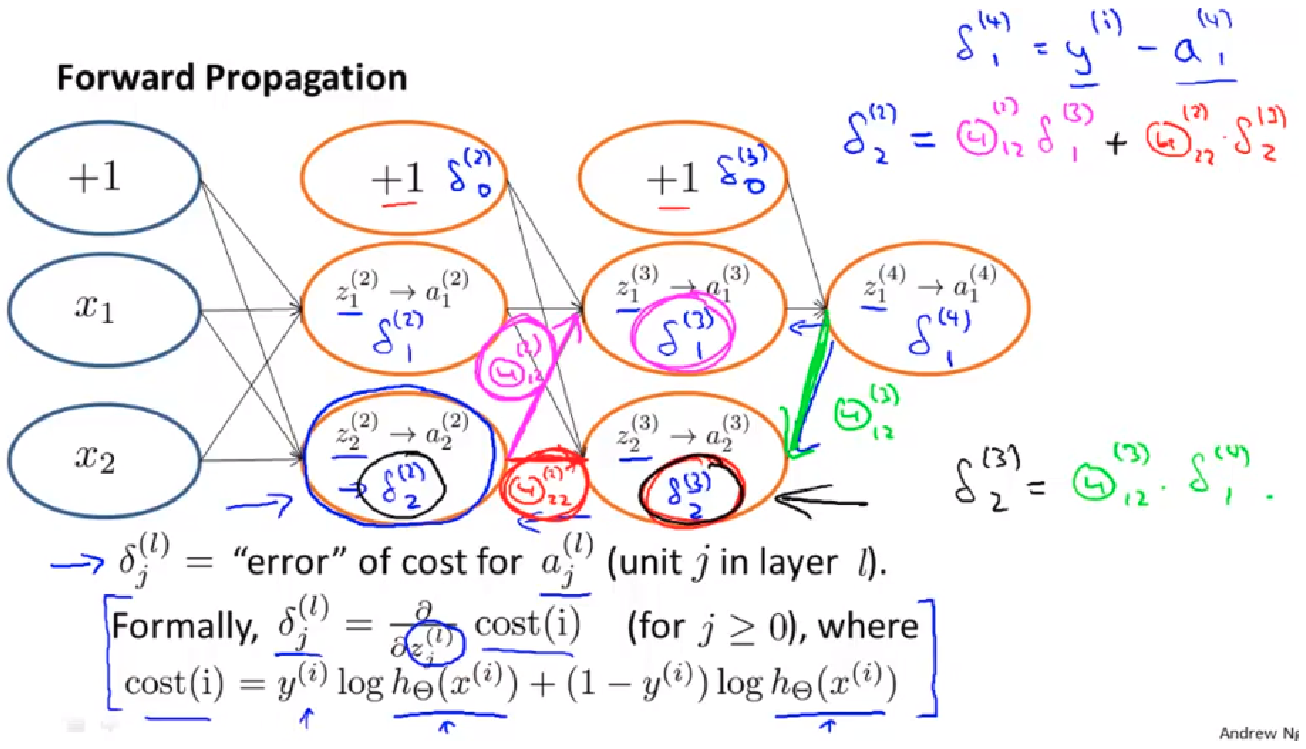
  $a^{(4)} = h_\Theta(x) = g(z^{(3)})$

  ....

3. Backpropagation algorithm


Forward Propagation
+1
$x_1^{(i)}$
$x_2^{(i)}$
$(x^{(i)}, y^{(i)})$
$z_1^{(2)} \to a_1^{(2)}$
$z_2^{(2)} \to a_2^{(2)}$
$z_1^{(3)} \to a_1^{(3)}$
$z_2^{(3)} \to a_2^{(3)}$
$z_1^{(4)} \to a_1^{(4)}$
$\Theta_{10}^{(2)}$
$\Theta_{11}^{(2)}$
$\Theta_{12}^{(2)}$

$$z_1^{(3)} = \Theta_{10}^{(2)} \times 1 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} \cdot a_1^{(2)}$$


Forward Propagation
+1
x_1
x_2
$+1 \ \delta_0^{(2)}$
$+1 \ \delta_0^{(3)}$
$z_1^{(2)} \to a_1^{(2)}$ $\delta_1^{(2)}$
$z_2^{(2)} \to a_2^{(2)}$ $\delta_2^{(2)}$
$z_1^{(3)} \to a_1^{(3)}$ $\delta_1^{(3)}$
$z_2^{(3)} \to a_2^{(3)}$ $\delta_2^{(3)}$
$z_1^{(4)} \to a_1^{(4)}$ $\delta_1^{(4)}$
$\Theta_{12}^{(2)}$
$\Theta_{22}^{(2)}$
$\Theta_{12}^{(3)}$
Andrew Ng

$$\delta_1^{(4)} = y^{(i)} - a_1^{(4)}$$

$$\delta_2^{(2)} = \Theta_{12}^{(2)} \delta_1^{(3)} + \Theta_{22}^{(2)} \cdot \delta_2^{(3)}$$

$$\delta_2^{(3)} = \Theta_{12}^{(3)} \cdot \delta_1^{(4)}.$$

$\delta_j^{(l)}$ = "error" of cost for $a_j^{(l)}$ (unit j in layer l).

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i)$ (for $j \geq 0$), where
$\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})$

II. Backpropagation in Practice

1. Implementation

2. Gradient checking

3. Random initialization

4. Summary

III. Application of Neural Networks