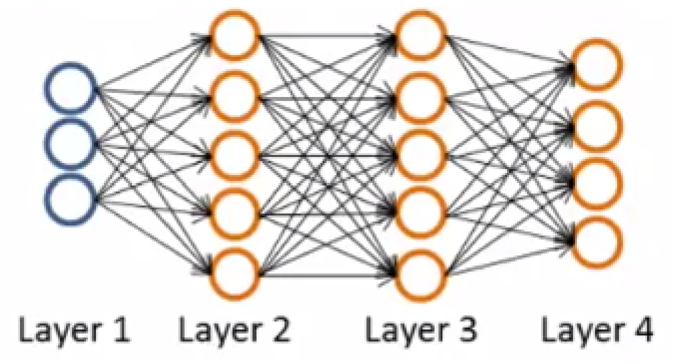
Cost Function

Suppose we have a training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$ and the below network:



Consider,

- $L =$ Total number of layers
- $s_l =$ Number of Units/Neurons (not counting bias unit) in layer l

We'll consider the following:

- **Binary Classification**

  $h_\theta(x) \in R$

  $s_l = 1$ [$l =$ Index of the output layer]

  $K = 1$ [$K =$ Number of units in the output layer]

- **Multi-class Classification**

  $h_\theta(x) \in R^K$

  $s_l = K$ [$l =$ Index of the output layer]

  $K \geq 3$ [$K =$ Number of units in the output layer]

Cost Function

For Logistic Regression, we had

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

In Neural Networks, our cost function will be a generalization of the above,

$$h_\theta(x) \in R^K \qquad (h_\theta(x))_i = i^{th} \text{ output}$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(\theta_{ji}^{(l)})^2$$

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.

Suppose we want to try to minimize $J(\theta)$ as a function of θ, using one of the advanced optimization methods (fminunc, conjugate gradient, BFGS, L-BFGS, etc.). What do we need to supply code to compute (as a function of θ)?

- [ ] θ
- [ ] $J(\theta)$
- [ ] The partial derivative terms $\frac{\partial}{\partial\theta_{ij}^{(l)}}$ for every i, j, l
- [✓] $J(\theta)$ and the partial derivative terms $\frac{\partial}{\partial\theta_{ij}^{(l)}}$ for every i, j, l

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 θ's in the entire network
- The i in the triple sum does **not** refer to training example i

Backpropagation

For Gradient computation, our goal is to compute the following:

- $J(\theta)$
- $\frac{\partial}{\partial\theta_{ij}^{(l)}} J(\theta)$

Given one training example (x, y). We'll first apply forward propagation as follows:

$a^{(1)} = x$

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

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

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

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

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

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

Now, for Gradient computation or Backpropagation, we'll do the following:

Intuition: $\delta_j^{(l)} =$ "error" of node j in layer l

For each output unit (Layer $L = 4$), $\delta_j^{(4)} = (h_\theta(x))_j - y_j = a_j^{(4)} - y_j$

Now, we compute for earlier layers,

$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} \cdot * g'(z^{(3)}) \qquad \left[g'(z^{(3)}) = a^{(3)} \cdot * (1 - a^{(3)})\right]$

$\delta^{(2)} = (\theta^{(2)})^T \delta^{(3)} \cdot * g'(z^{(2)}) \qquad \left[g'(z^{(2)}) = a^{(2)} \cdot * (1 - a^{(2)})\right]$

There is no $\delta^{(1)}$ as layer 1 is the input layer

Now suppose we have a large training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

Set $\triangle_{ij}^{(l)} = 0 \qquad \text{for all } l, i, j$

for $i = 1$ to m:

- Set $a^{(i)} = x^{(i)}$
- Perform Forward Propagation to compute $a^{(l)}$ for $l = 2, 3, \cdots, L$
- Using $y * (i)$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \cdots, \delta^{(2)}$
- $\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)} + \lambda \theta_{ij}^{(l)} \qquad \text{if } j \neq 0$

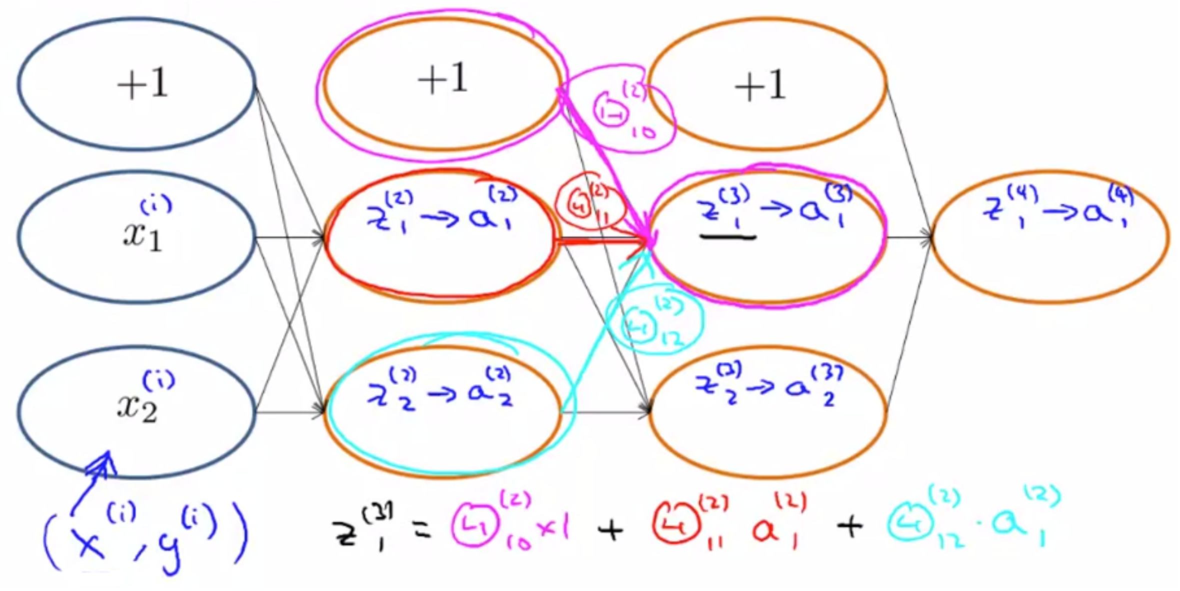
$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)} \qquad \text{if } j = 0 \text{ (Bias Term)}$

The above is equal to:

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)}$$

Intuition

Let's consider the following Forward Propagation,



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

What is Backpropagation doing?

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

Focussing on a single example $x^{(i)}, y^{(i)}$, the case of 1 output unit, and ignoring regularization ($\lambda = 0$),

$$\text{cost}(i) = y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log h_\theta(x^{(i)})$$

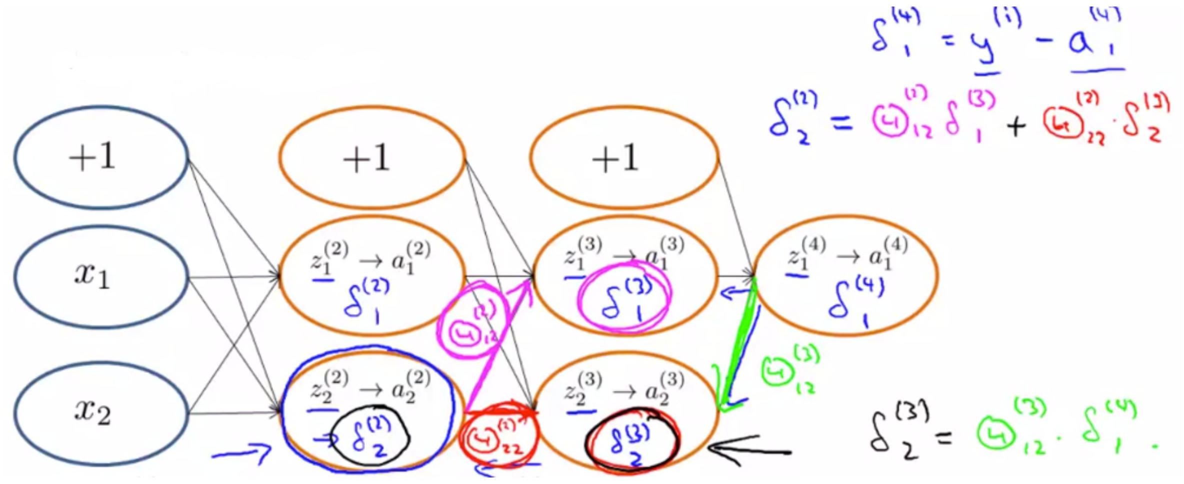
Assume, $\text{cost}(i) \approx (h_\theta(x^{(i)}) - y^{(i)})^2$

It basically finds how well is the network doing on example i

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

$$\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i) \qquad \text{(for } j \geq 0\text{)},$$

$$\text{Where,} \quad \text{cost}(i) = y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log h_\theta(x^{(i)})$$



$\delta_1^{(4)} = \underline{y}^{(i)} - \underline{a}_1^{(4)}$
$\delta_2^{(2)} = \Theta_{12}^{(2)} \delta_1^{(3)} + \Theta_{22}^{(2)} \cdot \delta_2^{(3)}$
$+1$
x_1
x_2
$z_1^{(2)} \to a_1^{(2)}$
$\delta_1^{(2)}$
$z_2^{(2)} \to a_2^{(2)}$
$\delta_2^{(2)}$
$z_1^{(3)} \to a_1^{(3)}$
$\delta_1^{(3)}$
$z_2^{(3)} \to a_2^{(3)}$
$\delta_2^{(3)}$
$z_1^{(4)} \to a_1^{(4)}$
$\delta_1^{(4)}$
$\Theta_{12}^{(2)}$
$\Theta_{22}^{(2)}$
$\Theta_{12}^{(3)}$
$\delta_2^{(3)} = \Theta_{12}^{(3)} \cdot \delta_1^{(4)}$.