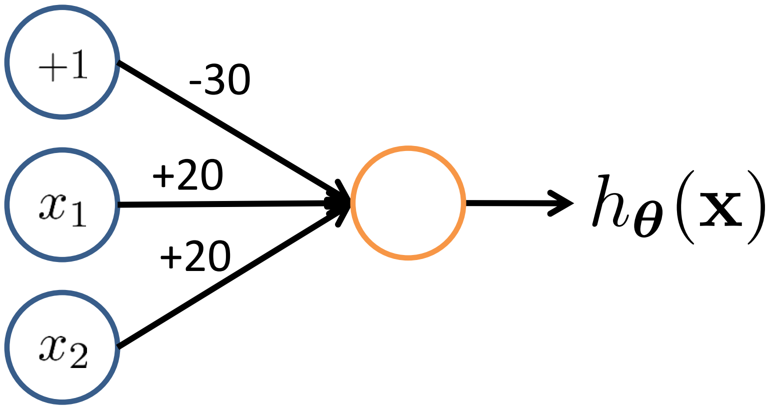
Neural Networks (Continued)

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.

Representing Boolean Functions

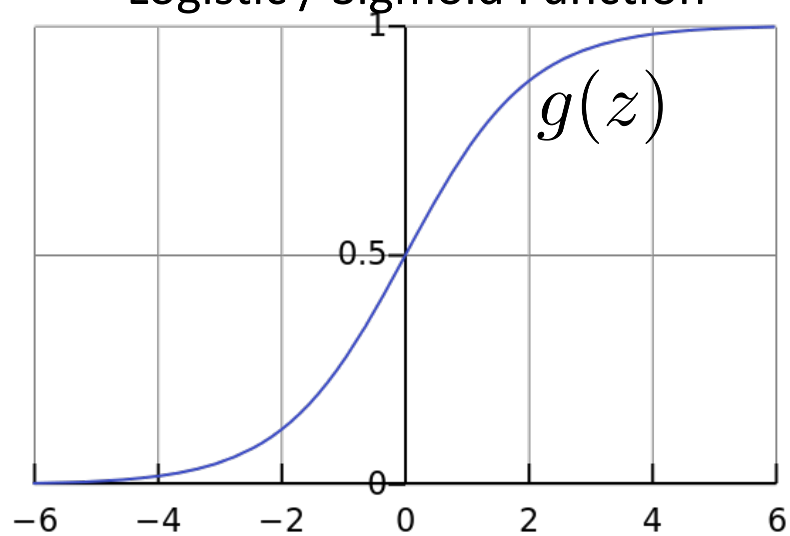
Simple example: AND

$$x_1, x_2 \in \{0, 1\}$$
$$y = x_1 \text{ AND } x_2$$



$$h_\Theta(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

Logistic / Sigmoid Function

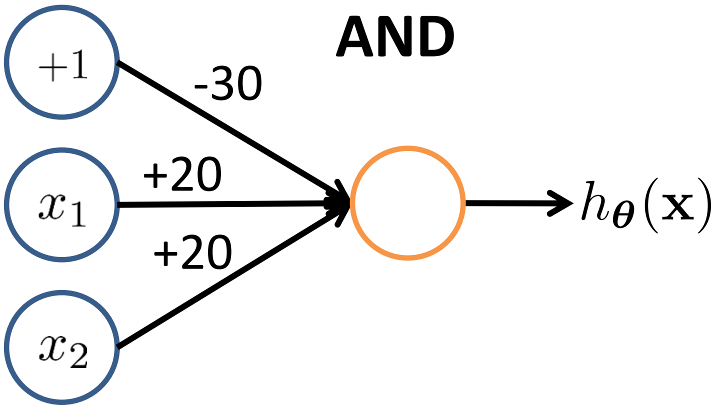


x_1	x_2	$h_\Theta(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

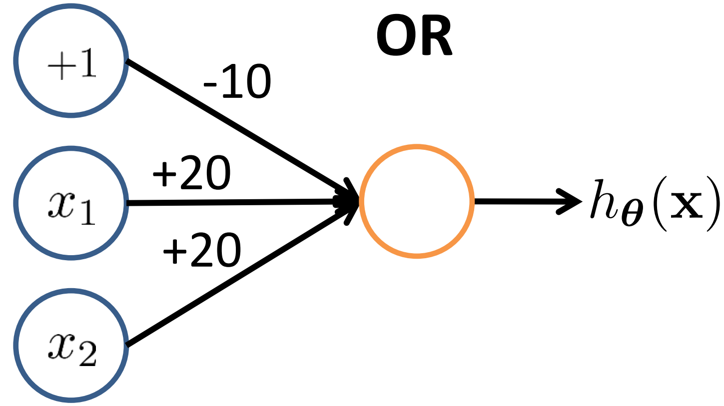
Based on slide and example by Andrew Ng

Representing Boolean Functions

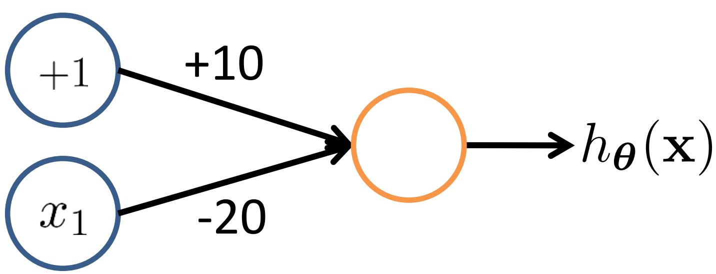
AND



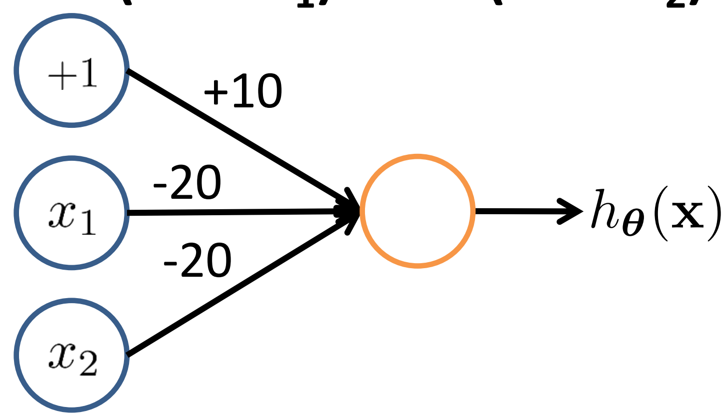
OR



NOT

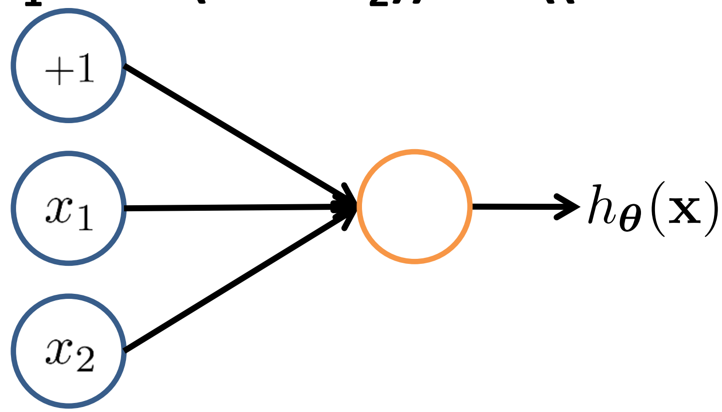


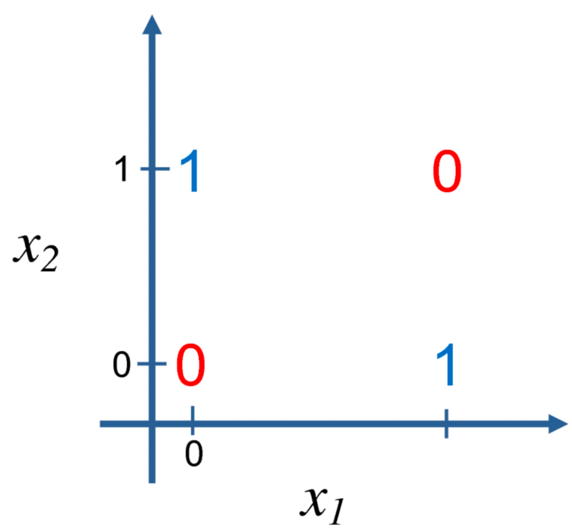
NOR: (NOT x_1) AND (NOT x_2)



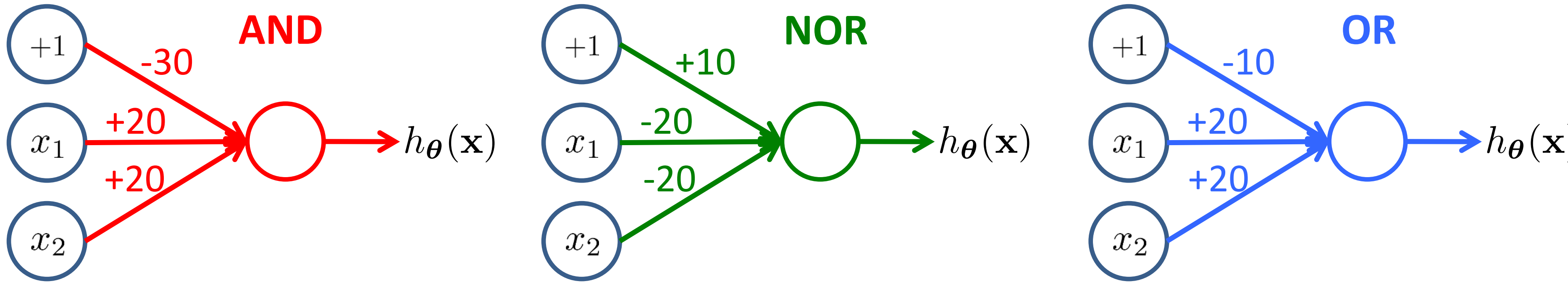
Representing Boolean Functions

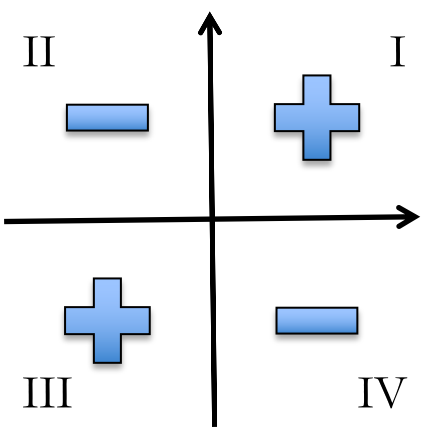
XOR: (x_1 AND (NOT x_2)) OR ((NOT x_1) AND x_2)

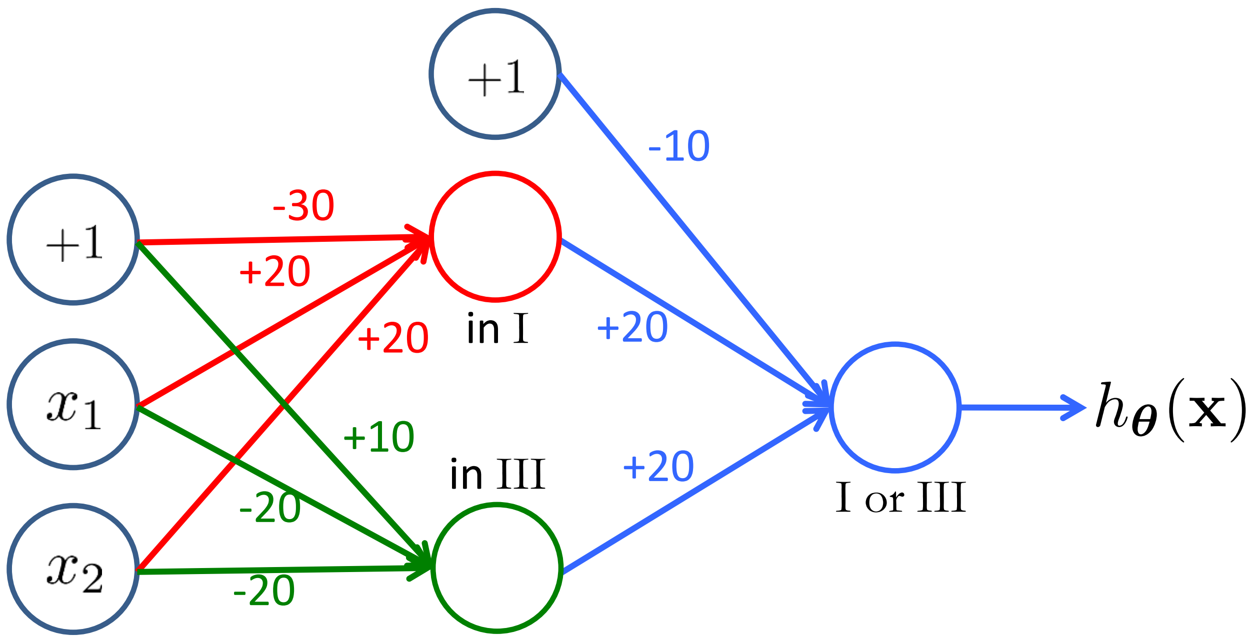




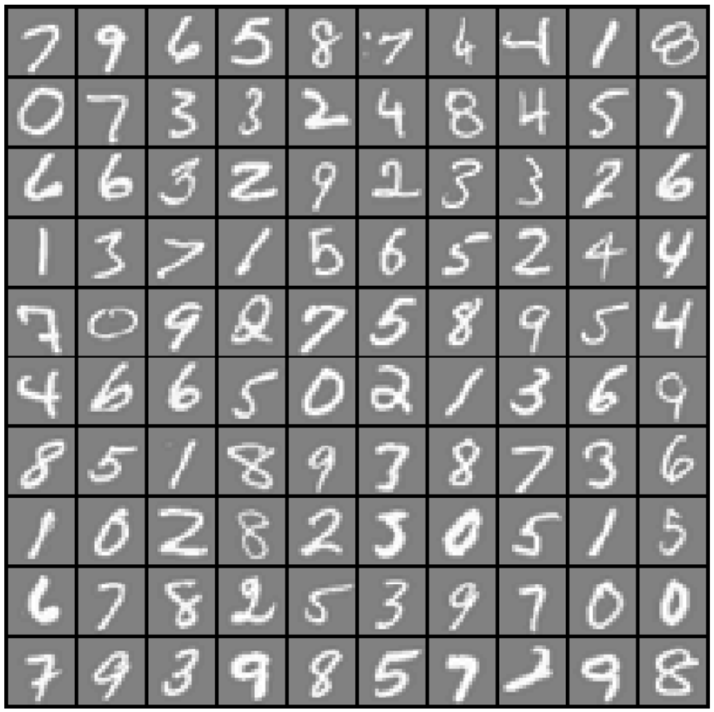
Combining Representations to Create Non-Linear Functions





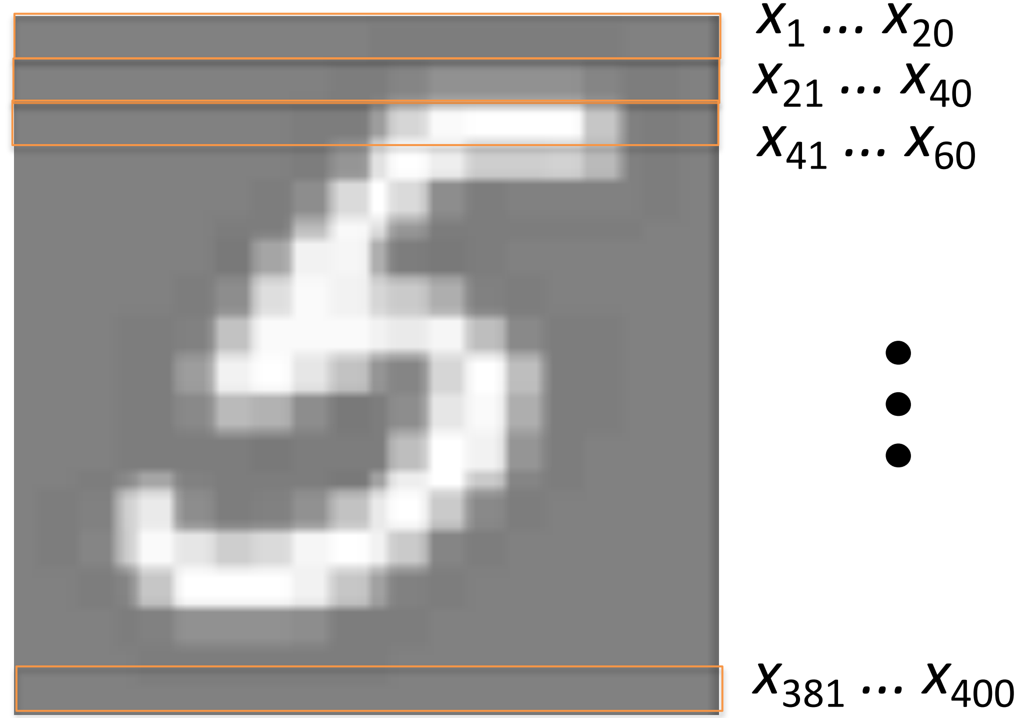


Layering Representations



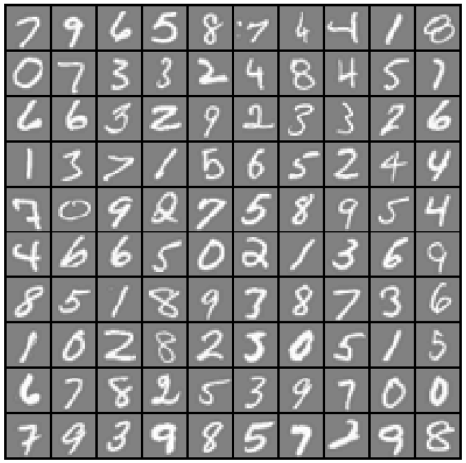
20 × 20 pixel images

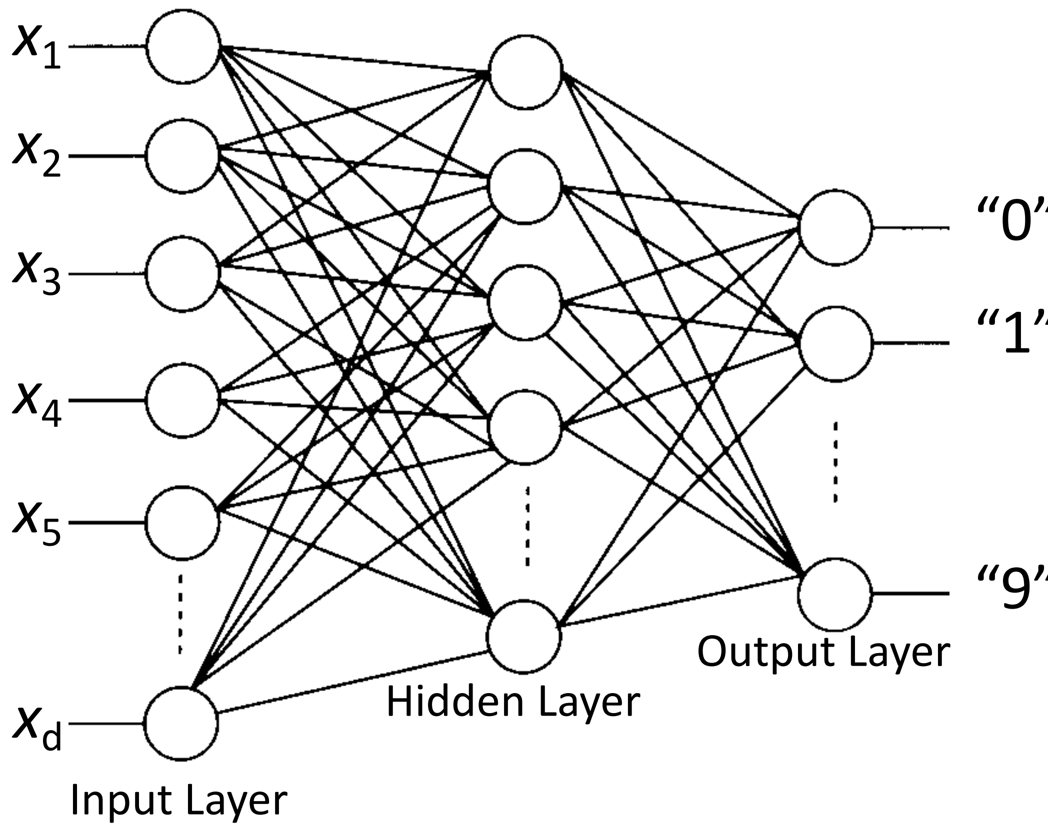
d = 400   10 classes

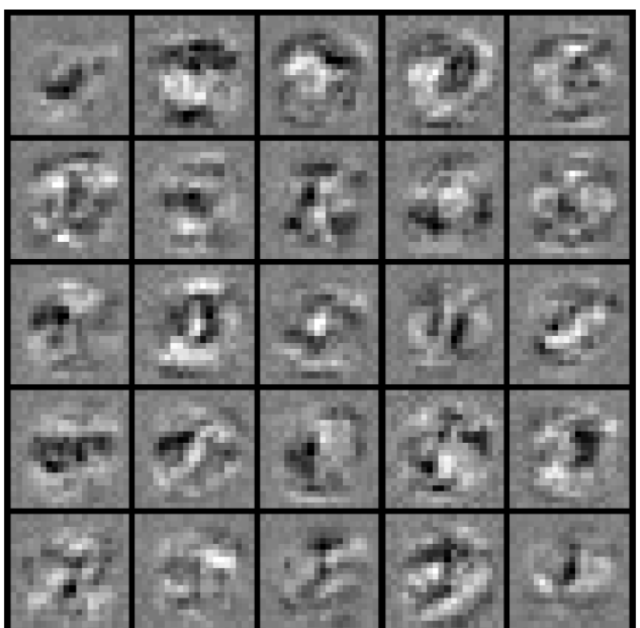


Each image is “unrolled” into a vector **x** of pixel intensities

Layering Representations







Visualization of
Hidden Layer

Neural Network Learning

Cost Function

Logistic Regression:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log h_{\boldsymbol{\theta}}(\mathbf{x}_i) + (1-y_i)\log(1-h_{\boldsymbol{\theta}}(\mathbf{x}_i))] + \frac{\lambda}{2n}\sum_{j=1}^{d}\theta_j^2$$

Neural Network:

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

$$J(\Theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\log(h_\Theta(\mathbf{x}_i))_k + (1-y_{ik})\log\left(1-(h_\Theta(\mathbf{x}_i))_k\right)\right]$$
$$+\frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(\Theta_{ji}^{(l)}\right)^2$$

k^{th} class: true, predicted
not k^{th} class: true, predicted

Based on slide by Andrew Ng

Optimizing the Neural Network

$$J(\Theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\log(h_\Theta(\mathbf{x}_i))_k + (1-y_{ik})\log\Big(1-(h_\Theta(\mathbf{x}_i))_k\Big)\right]$$
$$+\frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(\Theta_{ji}^{(l)}\right)^2$$

Solve via: $\min_{\Theta} J(\Theta)$

$J(\Theta)$ is not convex, so GD on a neural net yields a local optimum

- But, tends to work well in practice

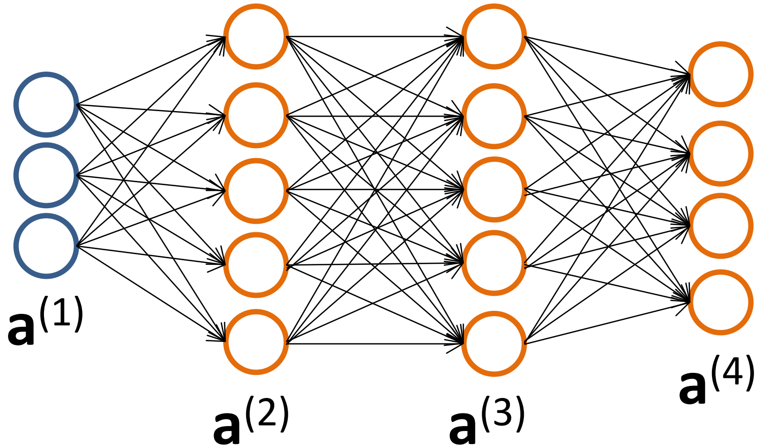
Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Based on slide by Andrew Ng

Forward Propagation

- Given one labeled training instance ($\mathbf{x}$, y):



<u>Forward Propagation</u>

- $\mathbf{a}^{(1)} = \mathbf{x}$
- $\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{a}^{(1)}$
- $\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$ [add $a_0^{(2)}$]
- $\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$
- $\mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$ [add $a_0^{(3)}$]
- $\mathbf{z}^{(4)} = \Theta^{(3)}\mathbf{a}^{(3)}$
- $\mathbf{a}^{(4)} = h_\Theta(\mathbf{x}) = g(\mathbf{z}^{(4)})$

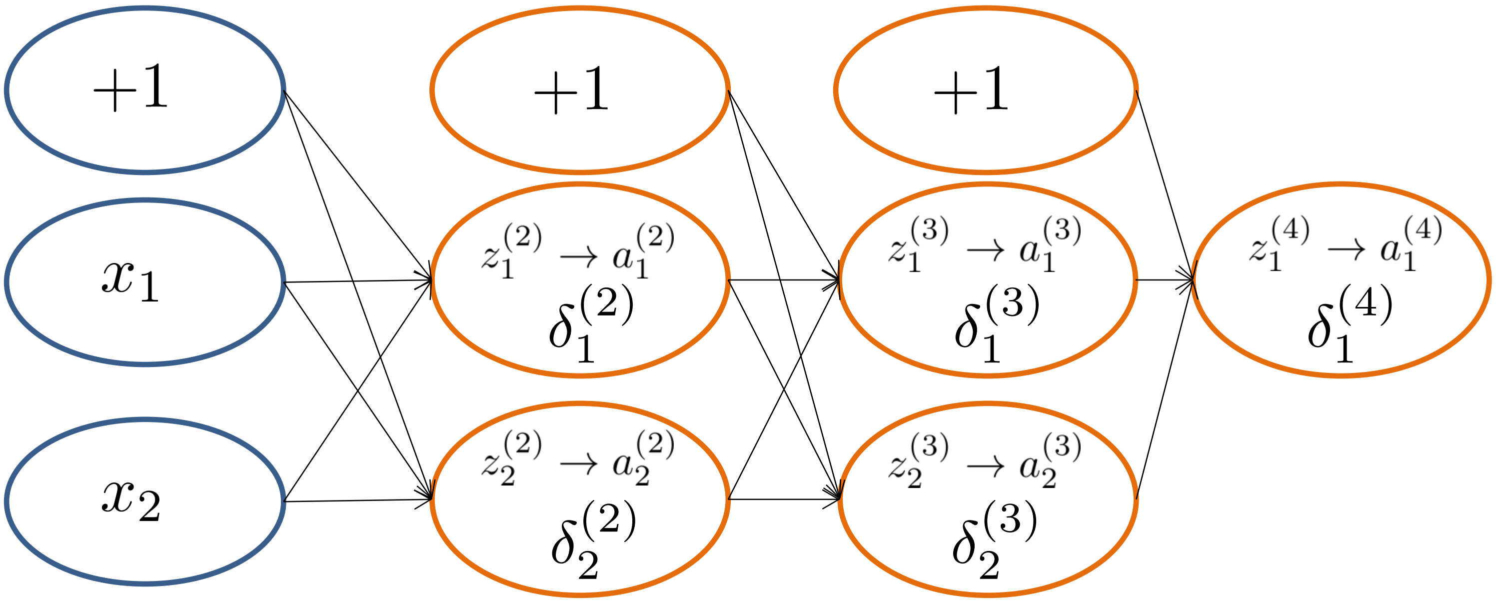
Backpropagation

- Each hidden node *j* is "responsible" for some fraction of the error $\delta_j^{(l)}$ in each of the output nodes to which it connects

- $\delta_j^{(l)}$ is divided according to the strength of the connection between hidden node and the output node

- Then, the "blame" is propagated back to provide the error values for the hidden layer

Backpropagation

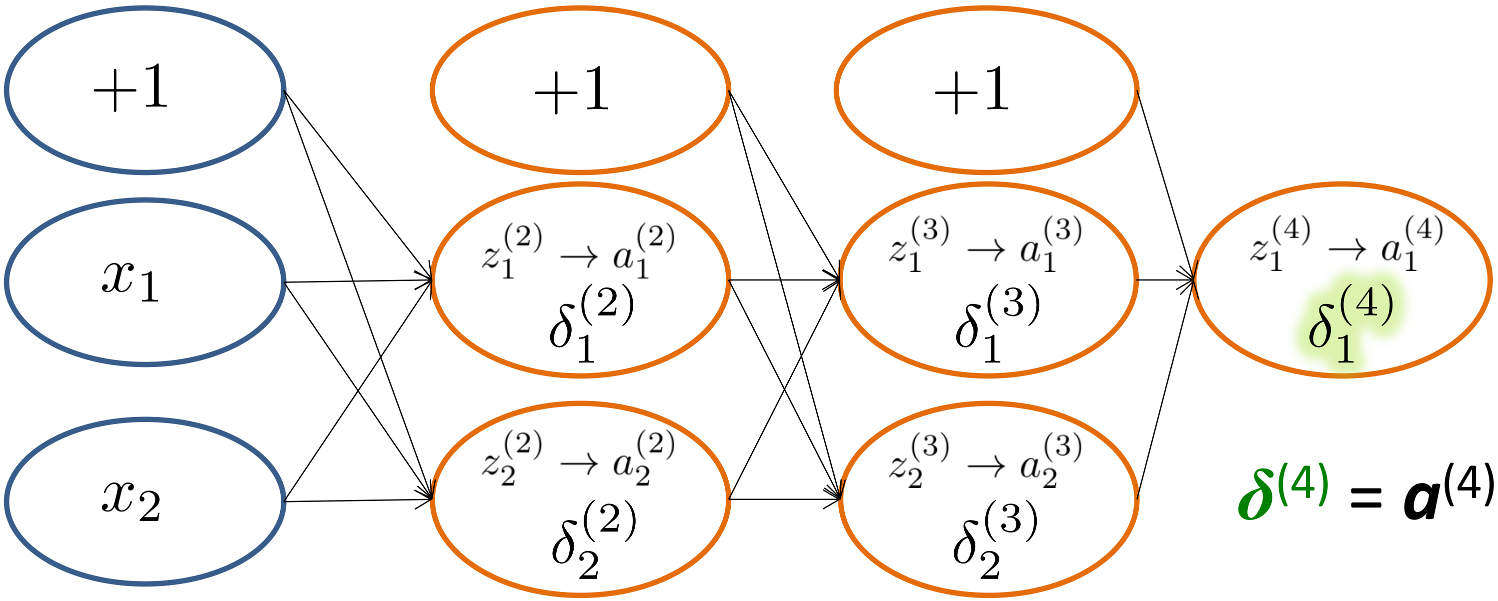


$\delta_j^{(l)}$ = "error" of node *j* in layer *l*

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

$$\text{where } \text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$$

Backpropagation



$\boldsymbol{\delta}^{(4)} = \boldsymbol{a}^{(4)} - \mathbf{y}$

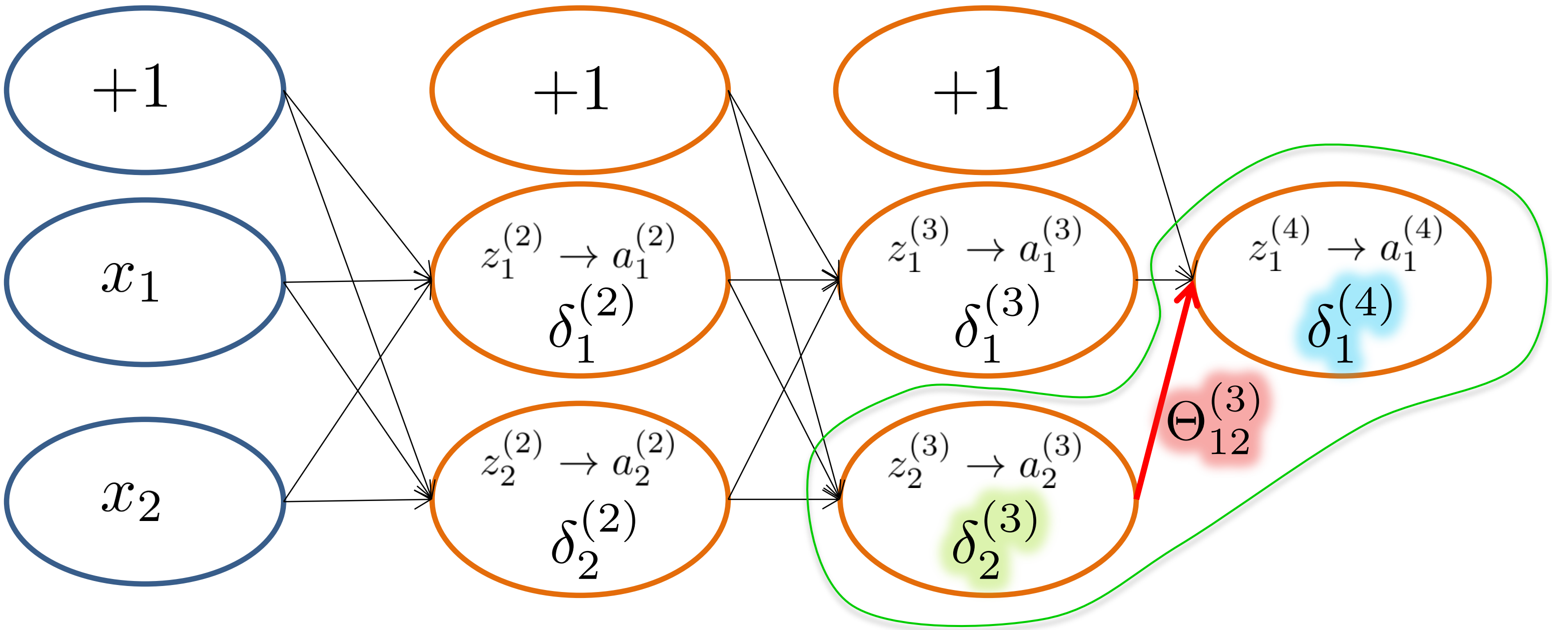
$\delta_j^{(l)}$ = "error" of node *j* in layer *l*

Formally, $$\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$$

$$\text{where } \text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$$

Based on slide by Andrew Ng

Backpropagation



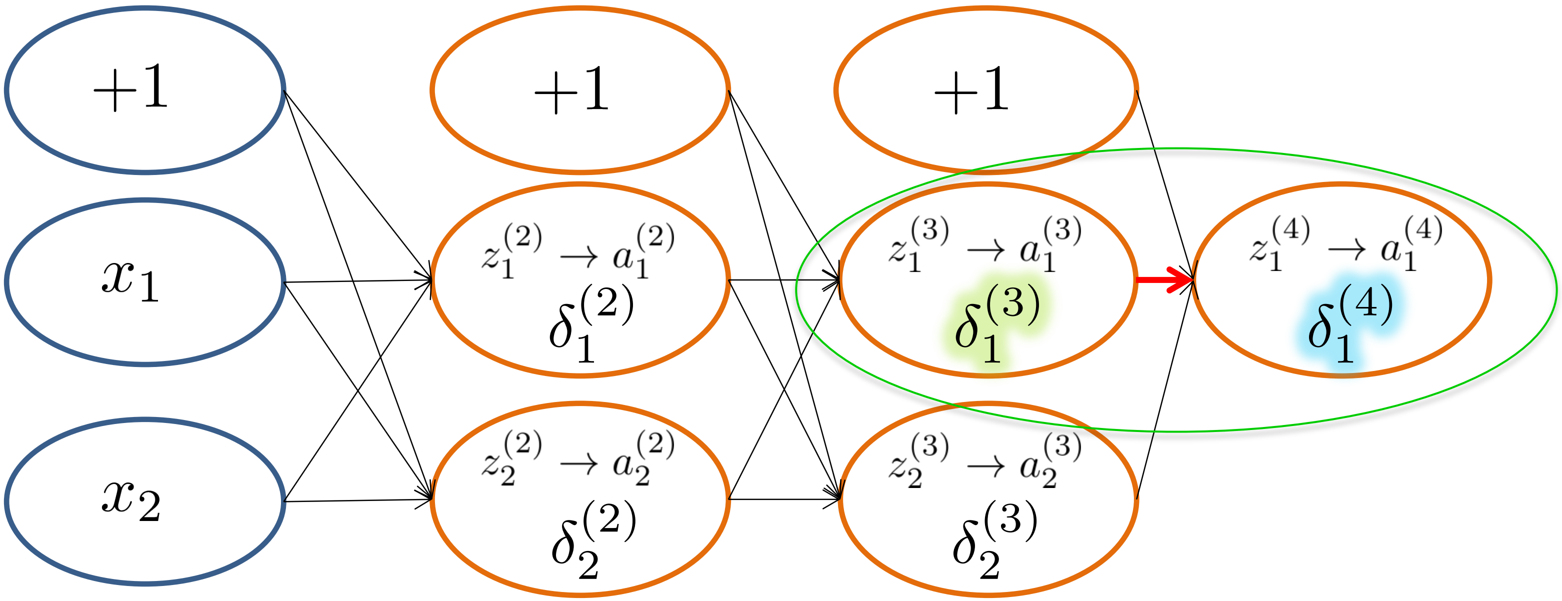
$$\delta_2^{(3)} = \Theta_{12}^{(3)} \times \delta_1^{(4)} \times a'^{(3)}_2$$

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

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

$$\text{where } \text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$$

Backpropagation



$\delta_2^{(3)} = \Theta_{12}^{(3)} \times \delta_1^{(4)} \times a'{}_2^{(3)}$

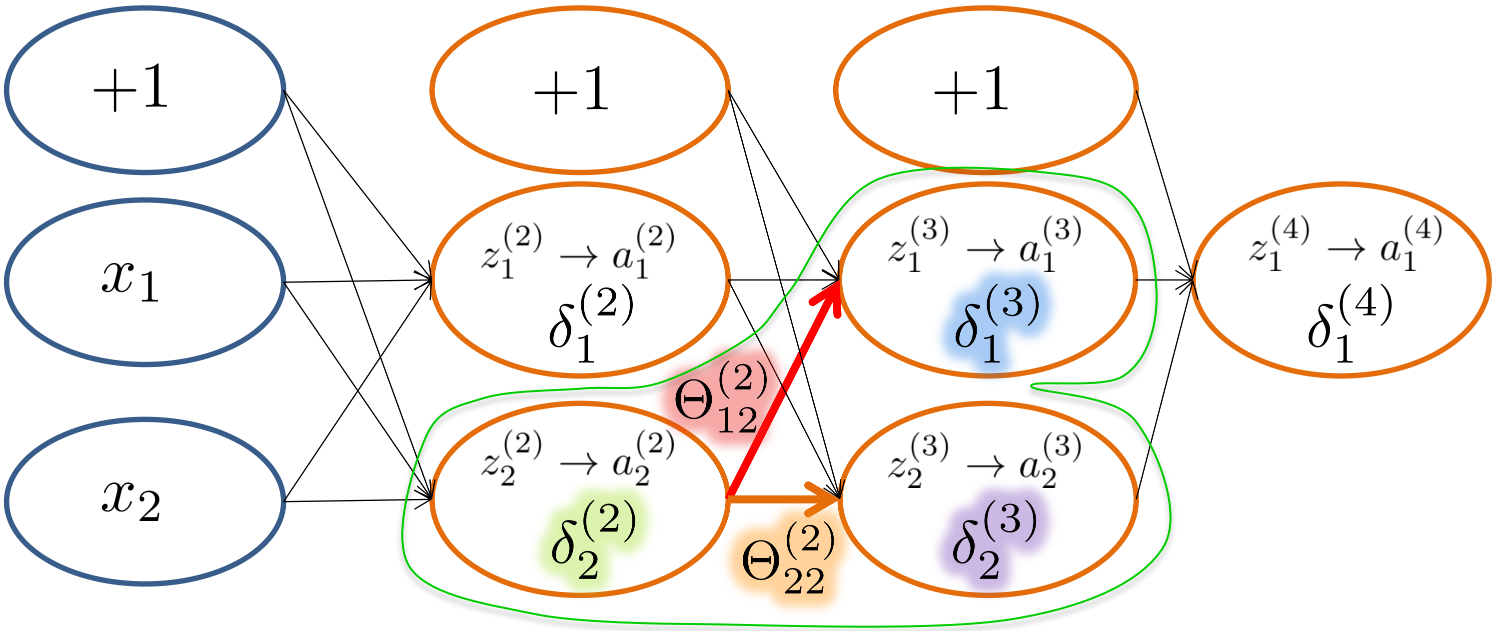
$\delta_1^{(3)} = \Theta_{11}^{(3)} \times \delta_1^{(4)} \times a'{}_1^{(3)}$

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

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

$$\text{where } \text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$$

Backpropagation



$$\delta_2^{(2)} = \Theta_{12}^{(2)} \times \delta_1^{(3)} \times a'^{(2)}_2 + \Theta_{22}^{(2)} \times \delta_2^{(3)} \times a'^{(2)}_2$$

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

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

$$\text{where } \text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$$