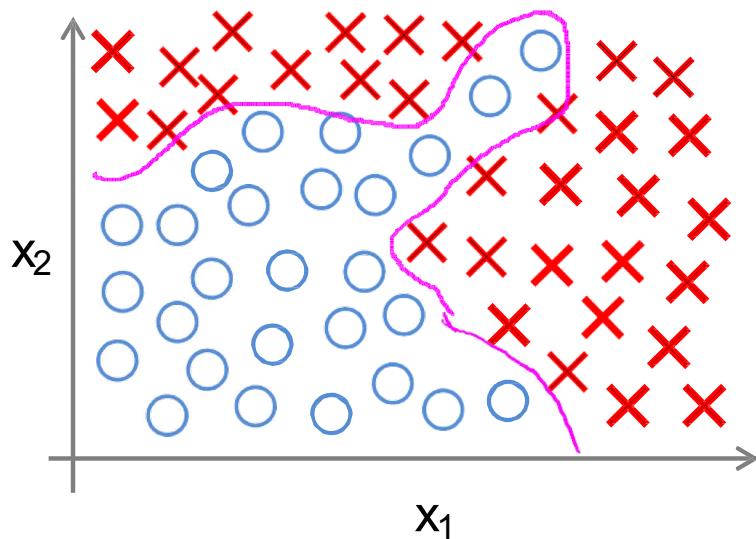


# 21. Neural Networks

Based on slides by Andrew Ng

# Non-linear Classification



$x_1$  = size  
 $x_2$  = # bedrooms  
 $x_3$  = # floors  
 $x_4$  = age  
...  
 $x_{100}$

$\{ h=100 \}$

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

$x_i^2, x_i x_2, x_i x_3, \underline{x_i x_4} \dots x_i x_{100}$   
 $x_i^2, x_i x_3 \dots$

$\approx 5000$  feature       $\mathcal{O}(n^2)$

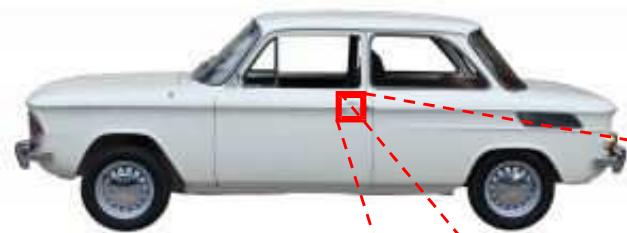
$\rightarrow \underline{x_i^2}, \underline{x_i^2}, \underline{x_3^2}, \dots, \underline{x_{100}^2}$        $\approx \frac{n^2}{2}$

$\rightarrow \underline{x_i x_2 x_3}, \underline{x_i^2 x_2}, \underline{x_{10} x_{11} x_{12}}, \dots$

$\mathcal{O}(n^3)$       170,000

# What is this?

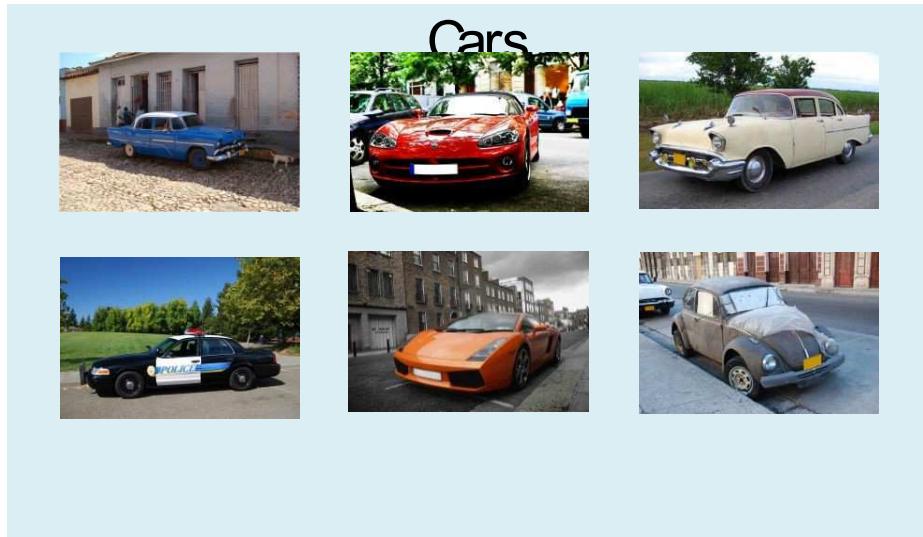
You see this:



But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

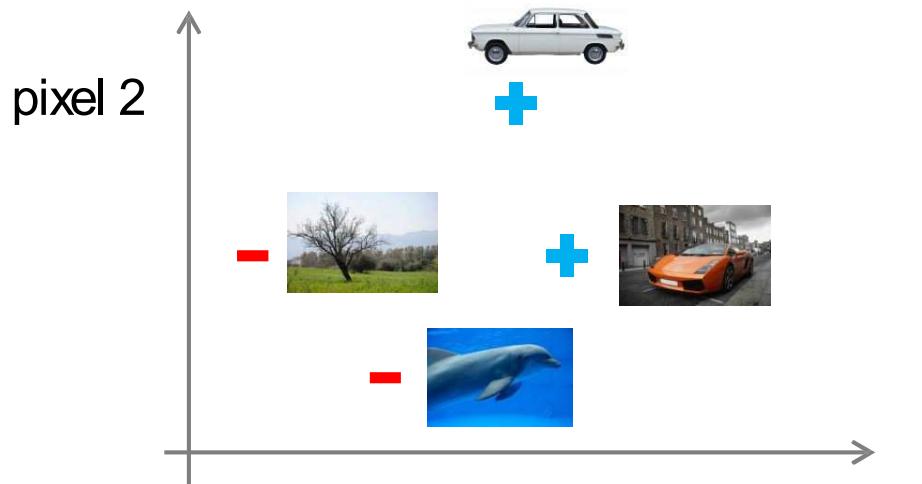
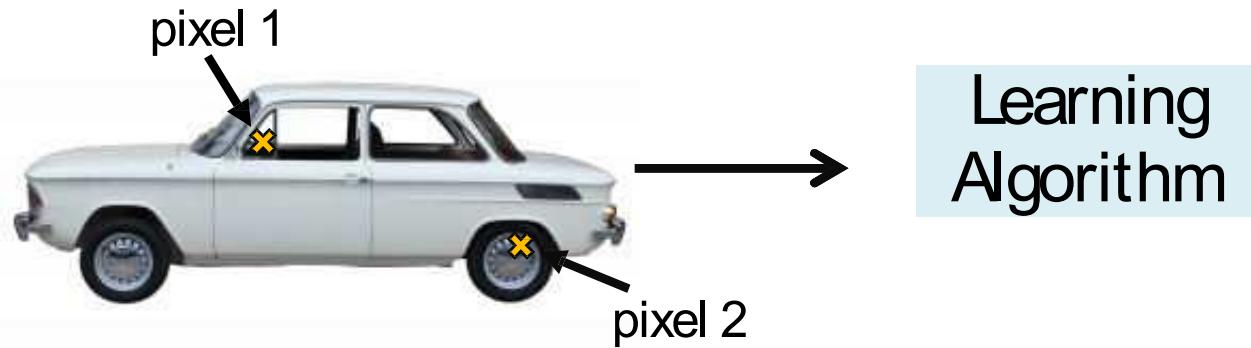
# Computer Vision: Car detection



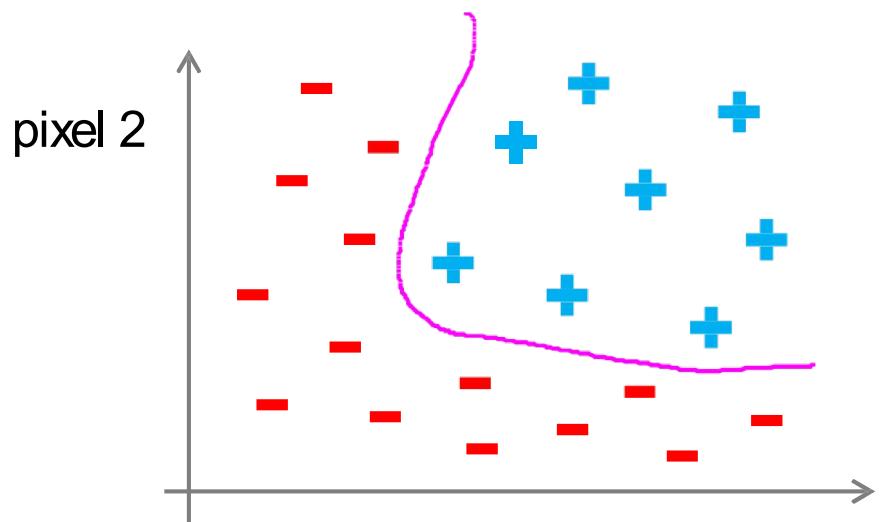
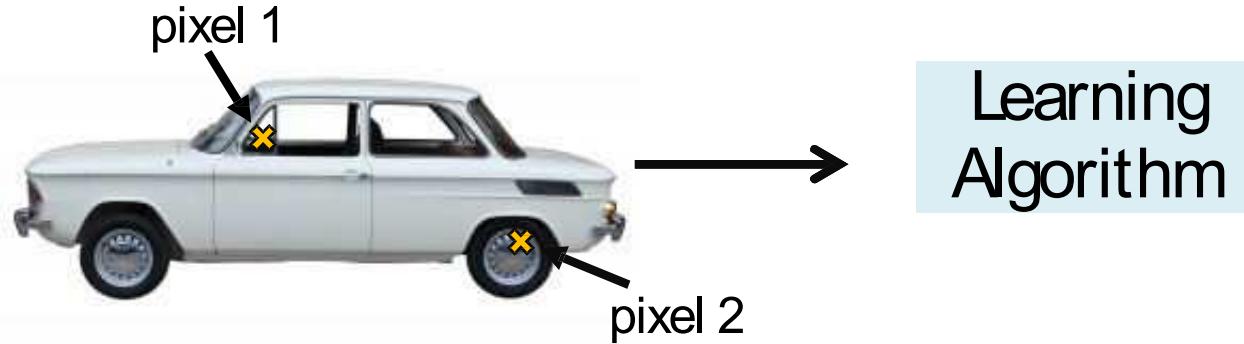
Testing:



What is this?



- + Cars
- "Non"-Cars



+ Cars  
- "Non"-Cars

$50 \times 50$  pixel images  $\rightarrow$  2500 pixels  
 $n = 2500$  (7500 if RGB)

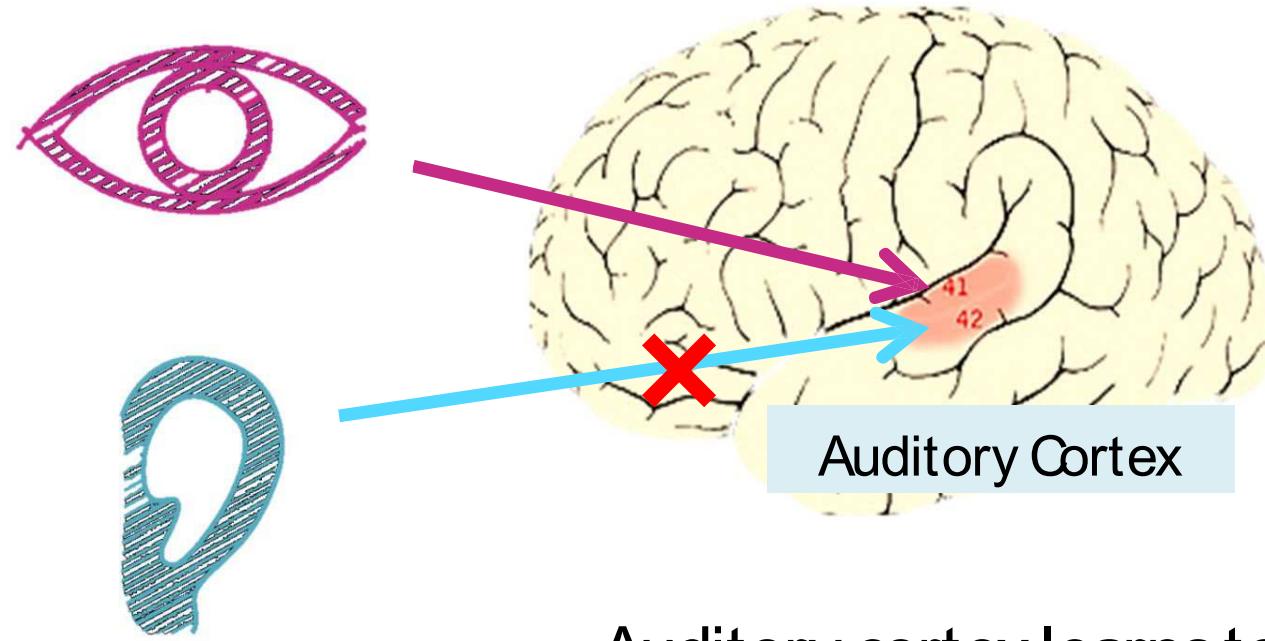
$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

Quadratic features ( $x_i \times x_j$ ):  $\approx 3$  million features

# Neural Networks

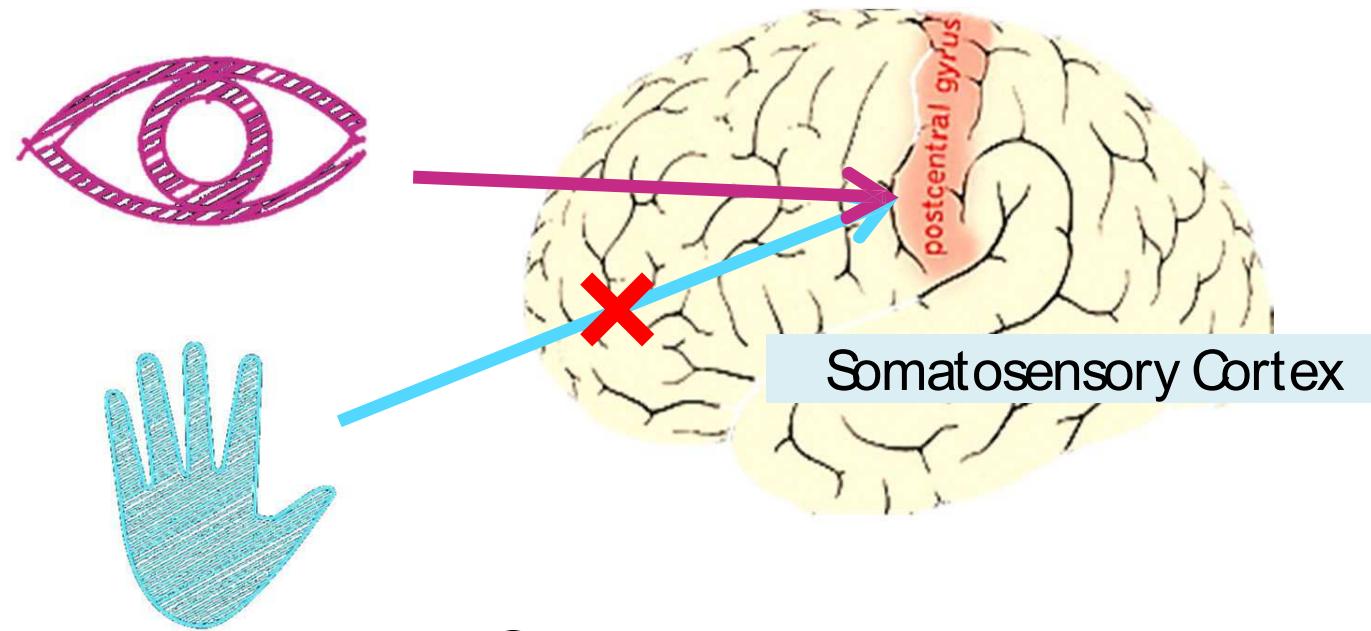
- Origins: Algorithms that try to mimic the brain.
- Was very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications

# The “one learning algorithm” hypothesis



Auditory cortex learns to see

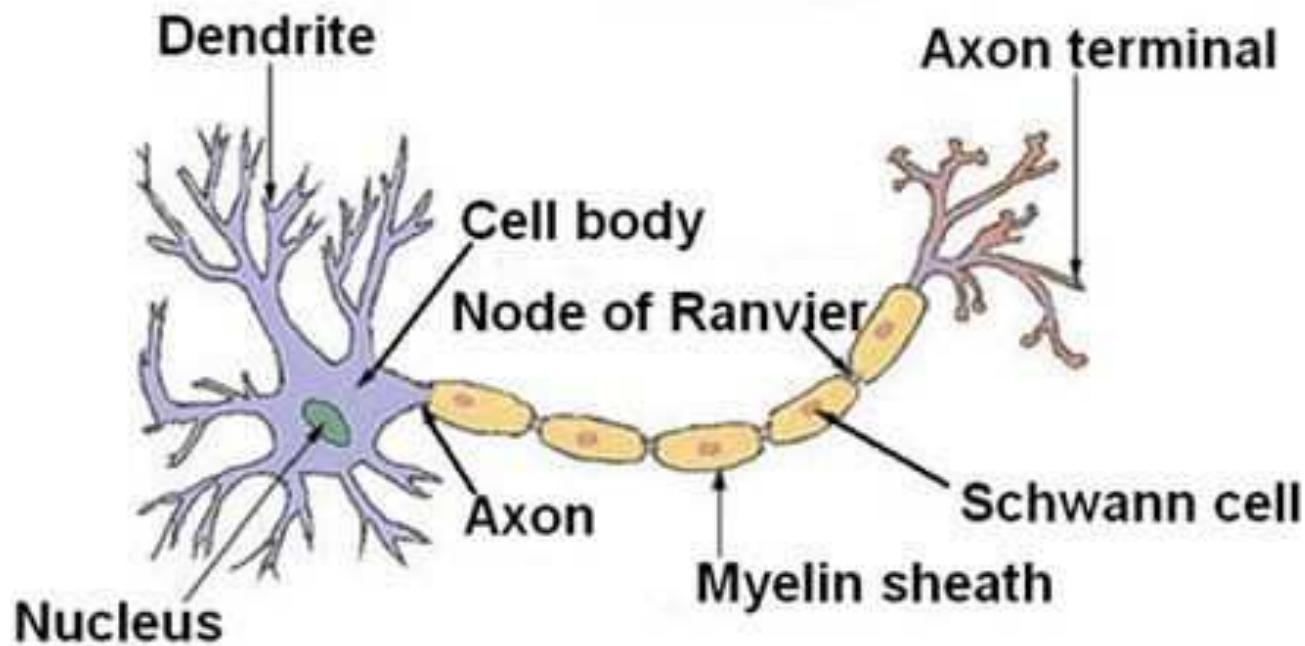
# The “one learning algorithm” hypothesis



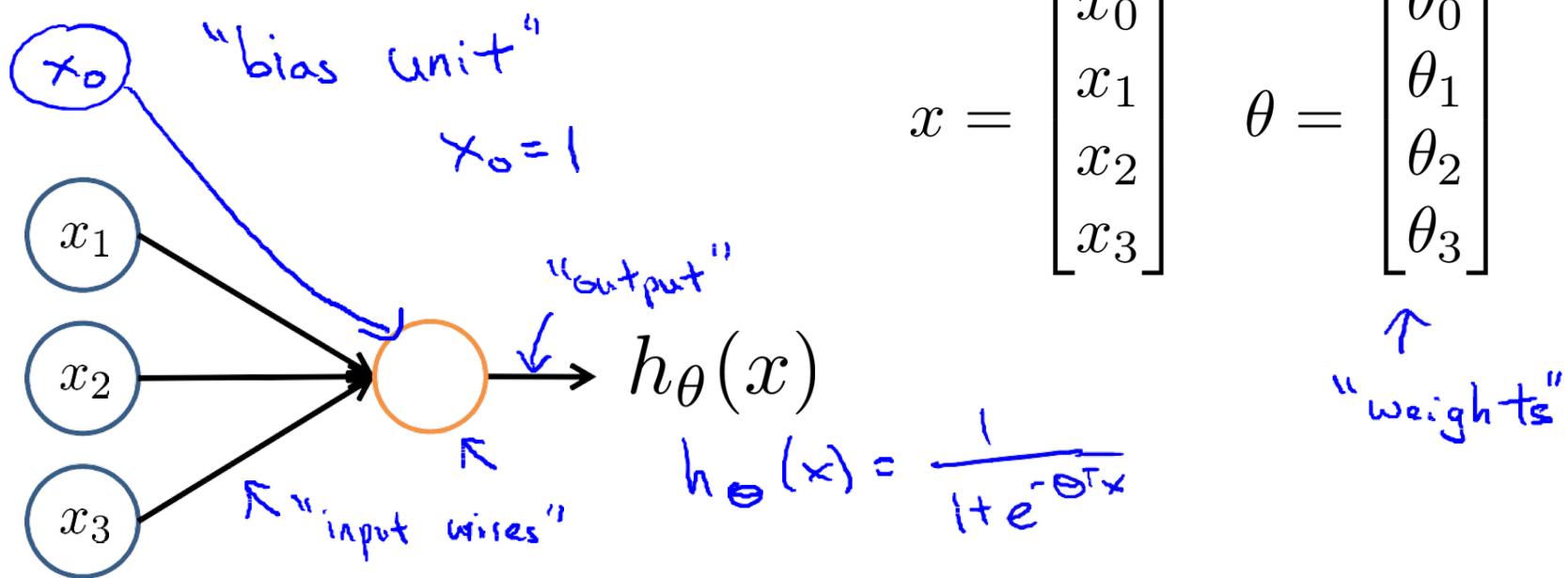
Somatosensory cortex learns to see

[Metin & Frost, 1989]

# Neuron in the brain



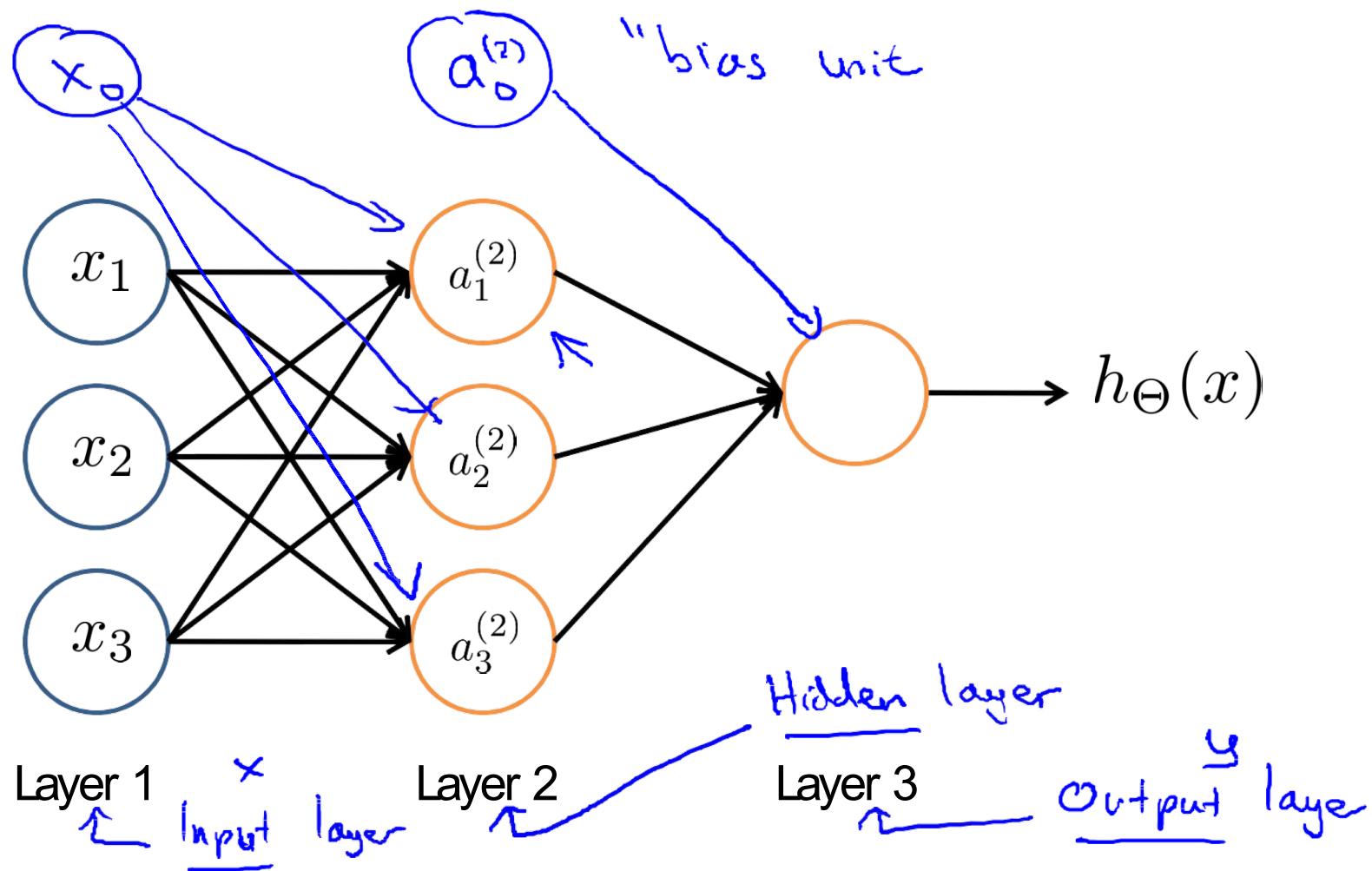
# Neuron model: Logistic unit



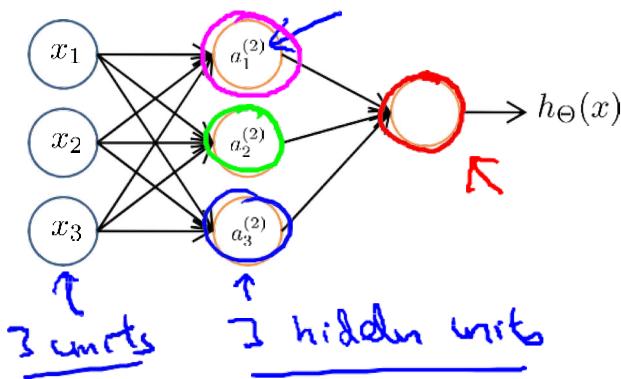
Sigmoid (logistic) activation function.

$$g(z) = \frac{1}{1 + e^{-z}}$$

# Neural Network



# Neural Network



$a_i^{(j)}$  = “activation” of unit  $i$  in layer  $j$

$\Theta^{(j)}$  = matrix of weights controlling  
function mapping from layer  $j$  to  
layer  $j + 1$

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$$

$$a_1^{(2)} = g(\underline{\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3})$$

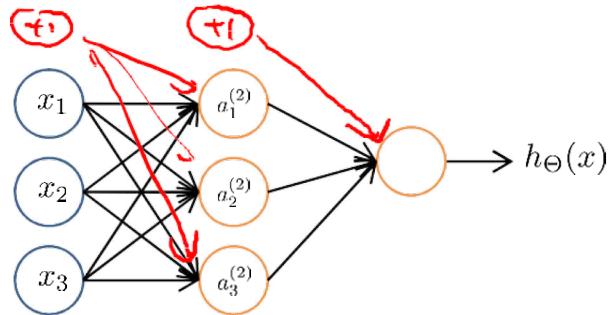
$$a_2^{(2)} = g(\underline{\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3})$$

$$a_3^{(2)} = g(\underline{\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3})$$

$$h_{\Theta}(x) = \underline{a_1^{(3)}} = g(\underline{\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}})$$

If network has  $s_j$  units in layer  $j$ ,  $s_{j+1}$  units in layer  $j + 1$ , then  $\Theta^{(j)}$   
will be of dimension  $s_{j+1} \times (s_j + 1)$ .

# Forward propagation: Vectorized implementation



$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_{\Theta}(x) = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

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

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

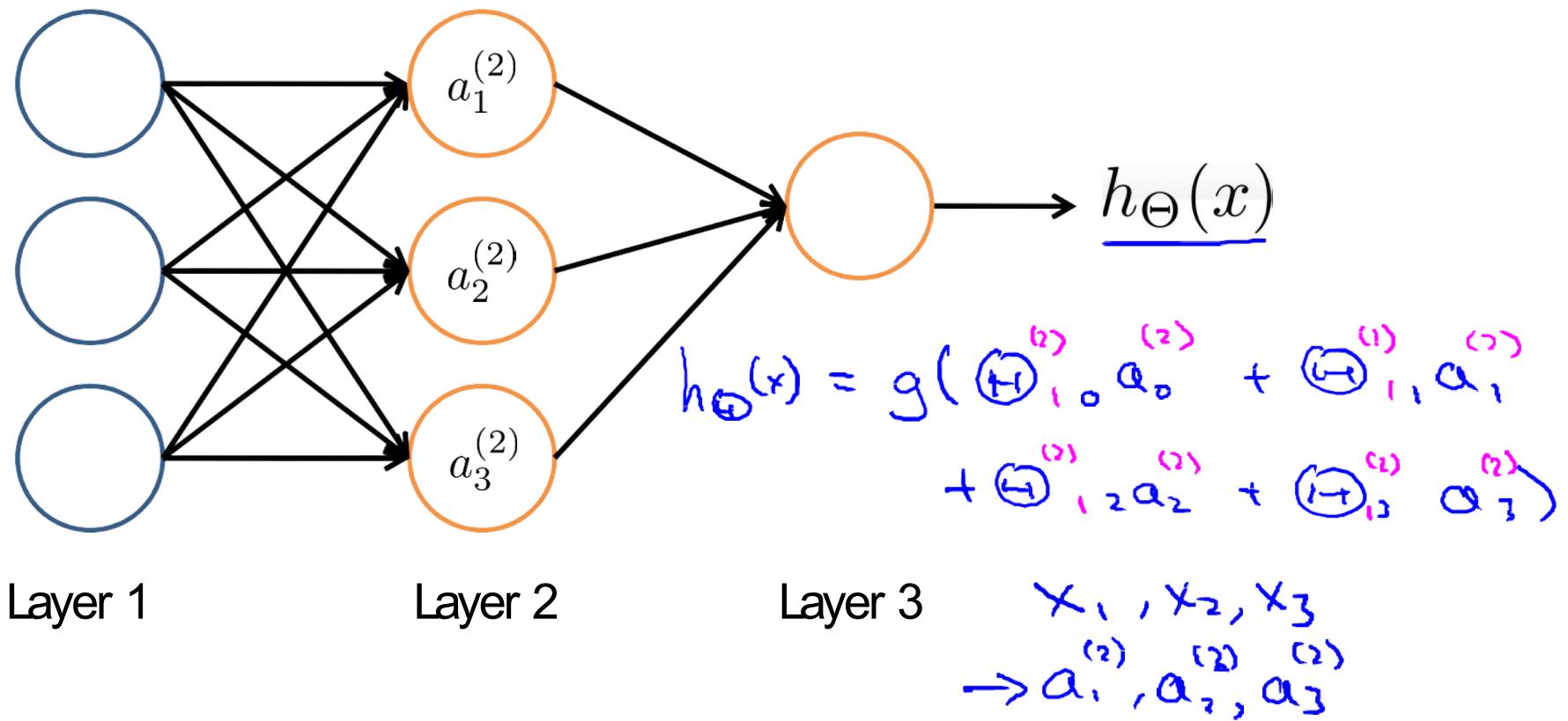
$$\text{Add } \underline{a_0^{(2)}} = 1 \rightarrow \underline{a^{(2)}} \in \mathbb{R}^4$$

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

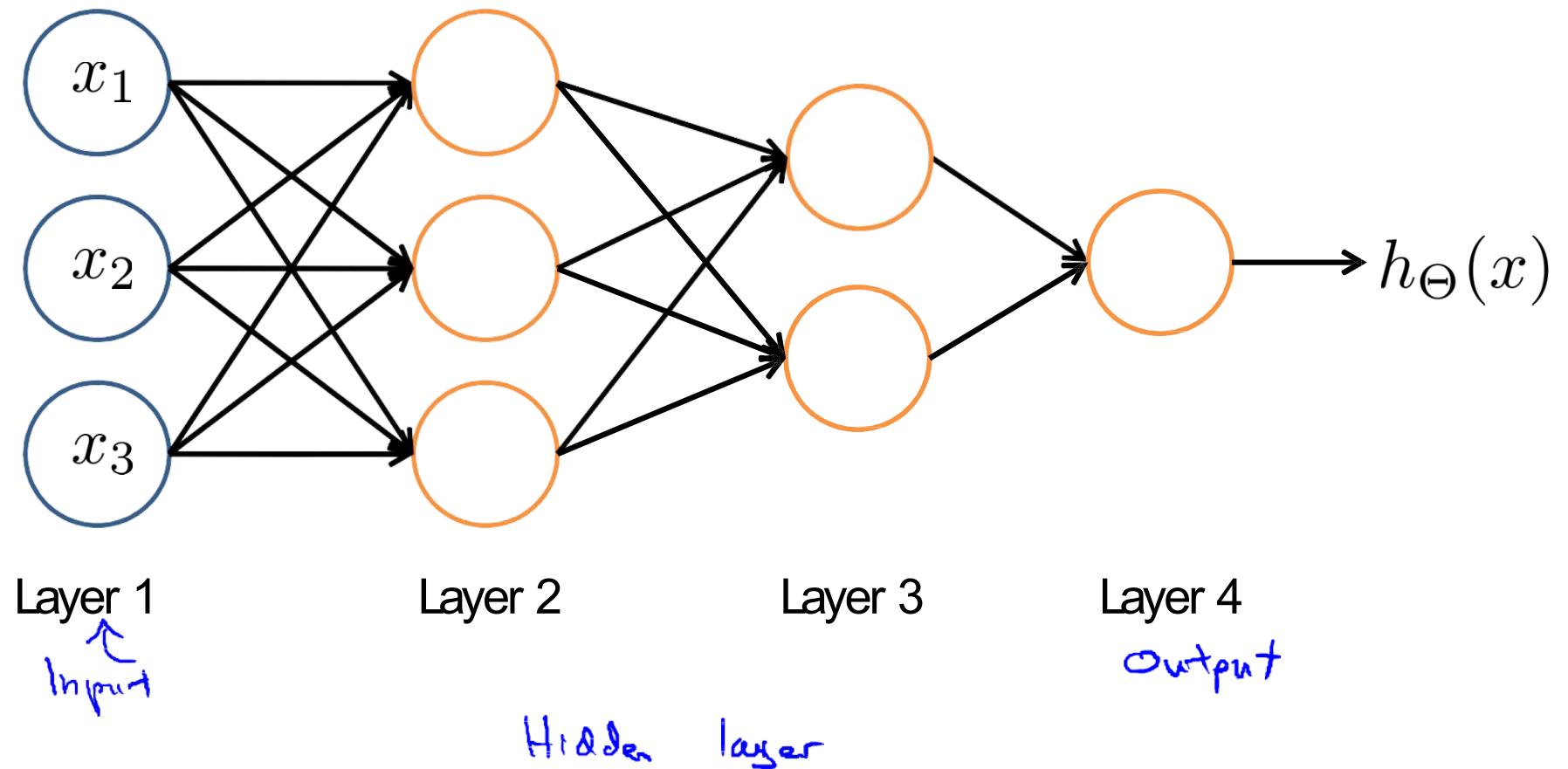
$$h_{\Theta}(x) = a^{(3)} = g(z^{(3)})$$

$a_1^{(2)}$   
 $a_2^{(2)}$   
 $a_3^{(2)}$

# Neural Network learning its own features



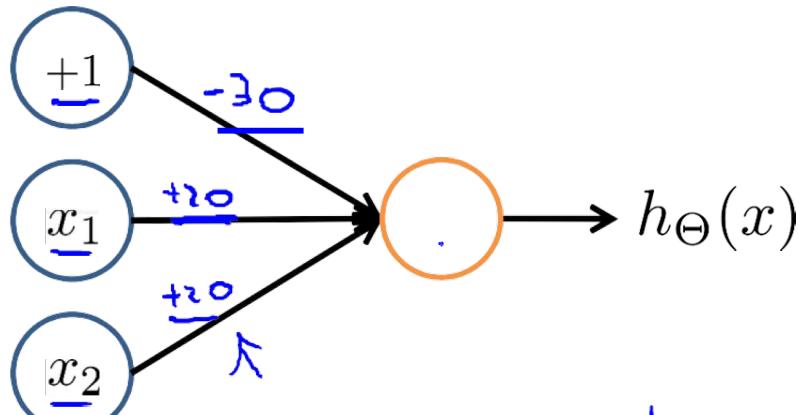
# Other network architectures



# Simple example: AND

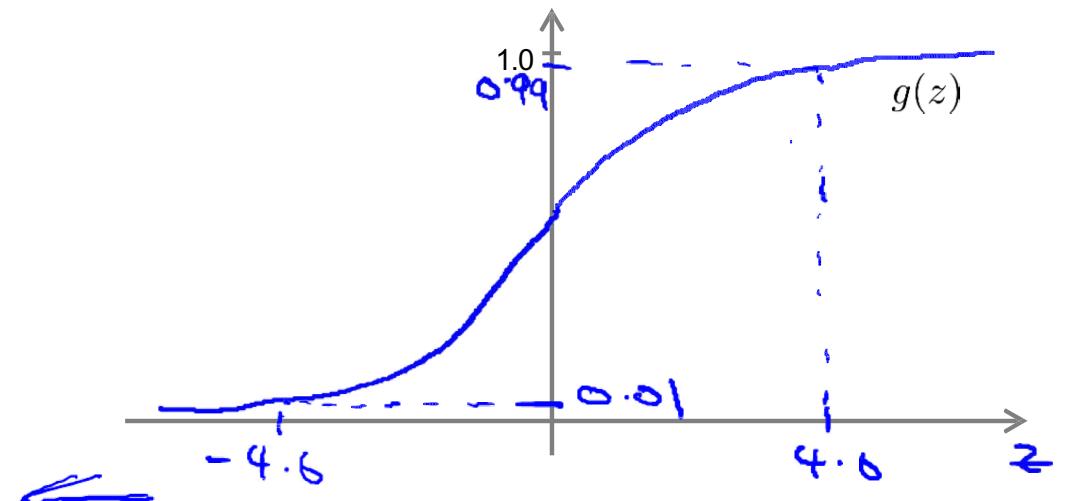
$$x_1, x_2 \in \{0, 1\}$$

$$y = x_1 \text{ AND } x_2$$



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

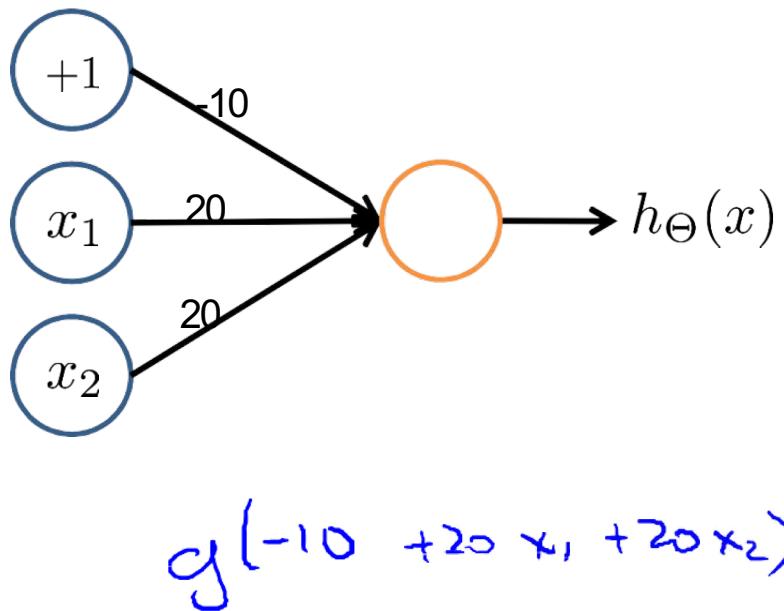
$\Theta_{10}^{(1)}$        $\Theta_{11}^{(1)}$        $\Theta_{12}^{(1)}$



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

$h_{\Theta}(x) \approx x_1 \text{ AND } x_2$

# Example: OR function



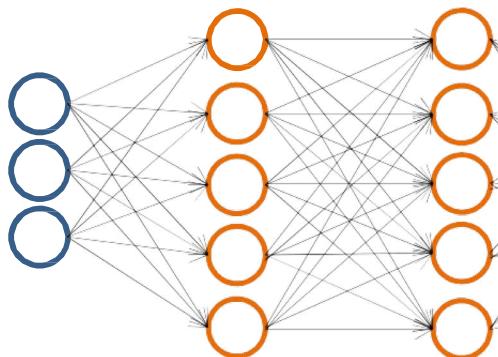
$x_1$	$x_2$	$h_{\Theta}(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	$\approx 1$
1	1	$\approx 1$

The table shows the output of the function  $h_{\Theta}(x)$  for all combinations of  $x_1$  and  $x_2$ . The values are approximated in blue. A pink hand-drawn bracket highlights the row where  $x_1 = 0$  and  $x_2 = 1$ , showing the value  $g(10) \approx 1$ .

# Multiple output units: One-vs-all.



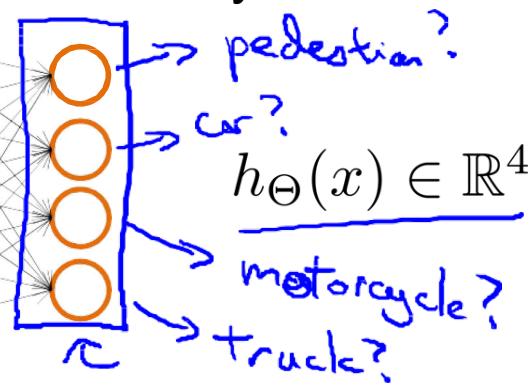
Pedestrian



Car



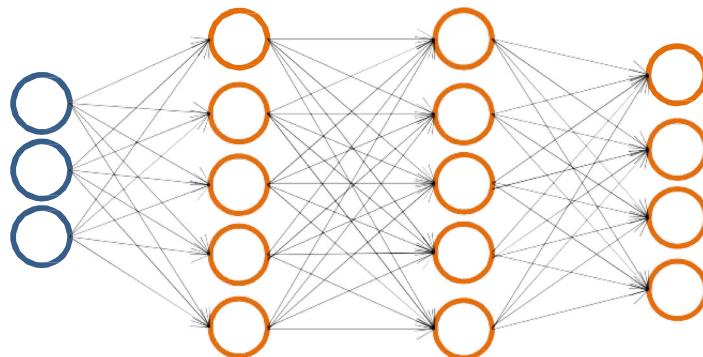
Motorcycle



Truck

Want  $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \underline{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ , etc.  
when pedestrian                    when car                    when motorcycle

# Multiple output units: One-vs-all.



$$h_{\Theta}(x) \in \mathbb{R}^4$$

Want  $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ , etc.

when pedestrian      when car      when motorcycle

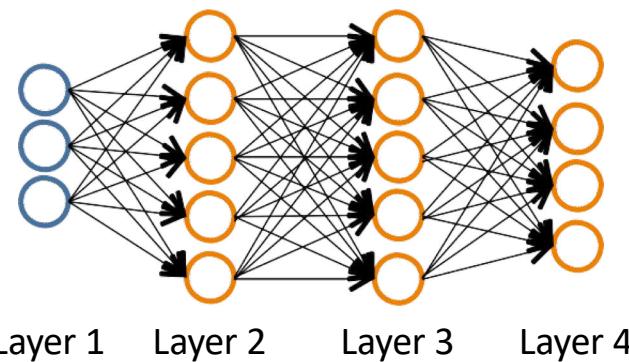
Training set:  $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

$y^{(i)}$  one of  $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

pedestrian    car    motorcycle    truck

$(x^{(i)}, y^{(i)})$

# Neural Network (Classification)



## Binary classification

$y = 0$  or  $1$

1 output unit

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

$L$  = total no. of layers in network

$s_l$  = no. of units (not counting bias unit) in layer  $l$

## Multi-class classification ( $K$ classes)

$$y \in \mathbb{R}^K \text{ E.g. } \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

pedestrian car motorcycle truck

$K$  output units

# Cost function

Logis5c regression:

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

Neural network:

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

$$J(\Theta) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k) \right]$$

$$+ \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

# Gradient computation

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

$$\min_{\Theta} J(\Theta)$$

Need code to compute:

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

$$\Theta_{ij}^{(l)} \in \mathbb{R}$$

# Gradient computation

Given one training example ( $x, y$ ):

Forward propagation:

$$a^{(1)} = x$$

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

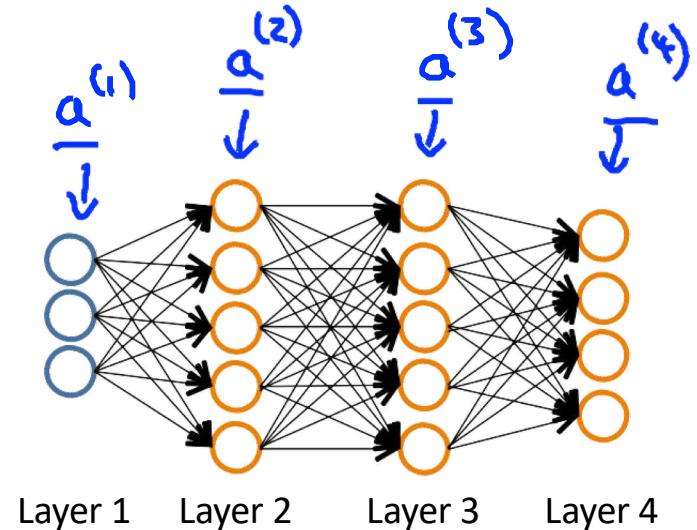
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$

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

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

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

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

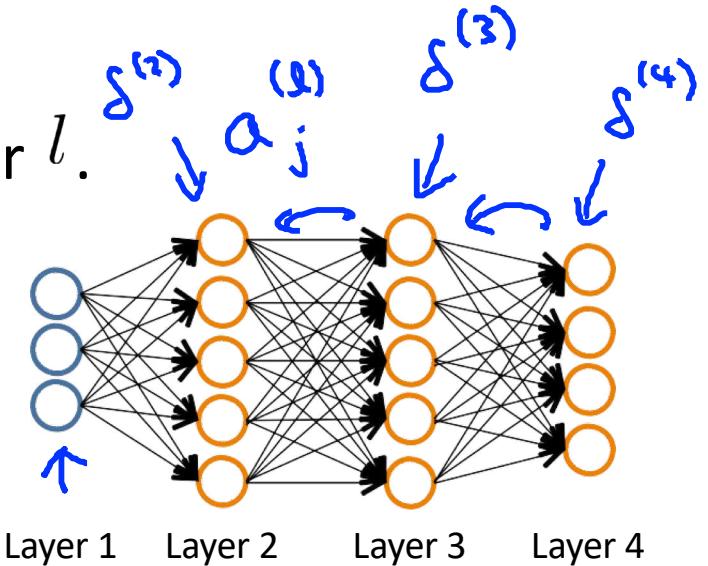


# Backpropagation algorithm

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

For each output unit (layer  $L = 4$ )

$$\delta_j^{(4)} = a_j^{(4)} - y_j \quad (\text{Note } h_{\Theta}(x))_j \quad \underline{\delta^{(4)}} = \underline{a^{(4)}} - \underline{y}$$



$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} * g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} * g'(z^{(2)})$$

(No  $\delta^{(1)}$ )

$$\frac{\partial}{\partial \Theta_{ij}^{(k)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{a^{(3)}}{a^{(2)}} * \frac{(1-a^{(3)})}{(1-a^{(2)})}$$

(ignoring  $\lambda$ ; if  
 $\lambda = 0$ )

# Backpropagation algorithm

Training set  $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Set  $\Delta_{ij}^{(l)} = 0$  (for all  $l, i, j$ ).

(use to compute  $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$ )

For  $i = 1$  to  $m$

$(\underline{x}^{(i)}, \underline{y}^{(i)})$

Set  $a^{(1)} = x^{(i)}$

Perform forward propagation to compute  $a^{(l)}$  for  $l = 2, 3, \dots, L$

Using  $y^{(i)}$ , compute  $\delta^{(L)} = a^{(L)} - y^{(i)}$

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

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

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + \delta^{(l+1)} (a^{(l)})^T.$$

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

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \text{ if } j = 0$$

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