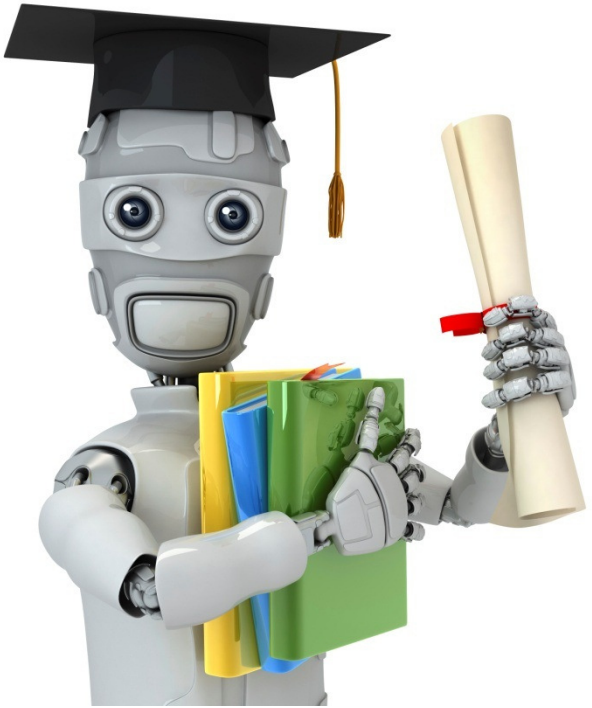


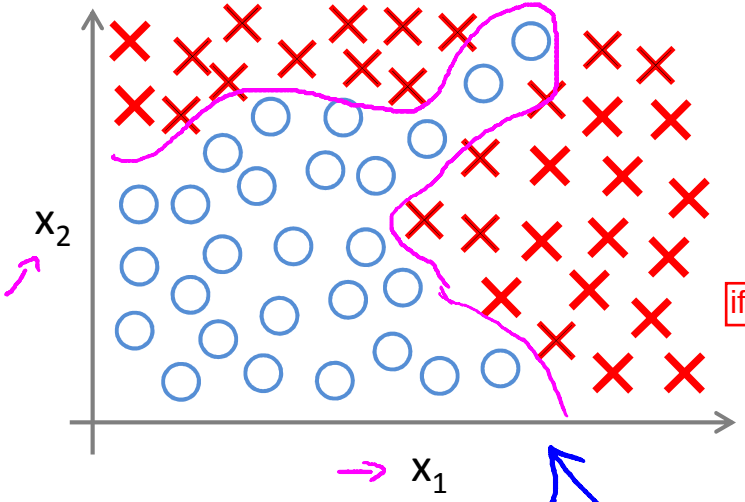
Machine Learning

Neural Networks: Representation

Non-linear hypotheses

if we have a lot more features :

Non-linear Classification



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

if we use quadratic :

$\rightarrow x_1^2, x_1x_2, x_1x_3, x_1x_4 \dots x_1x_{100}$

$x_2^2, x_2x_3 \dots$

≈ 5000 feature

$O(n^2)$

$\approx \frac{n^2}{2}$

if only include a subset of it :

$\rightarrow x_1^2, x_2^2, x_3^2, \dots, x_{100}^2$

$n = 100$

if we use cubic polynimial :

$\rightarrow x_1x_2x_3, x_1^2x_2, x_{10}x_{11}x_{17}, \dots$

$O(n^3)$

$170{,}000$

$\rightarrow x_1$ = size

x_2 = # bedrooms

x_3 = # floors

x_4 = age

$\dots$

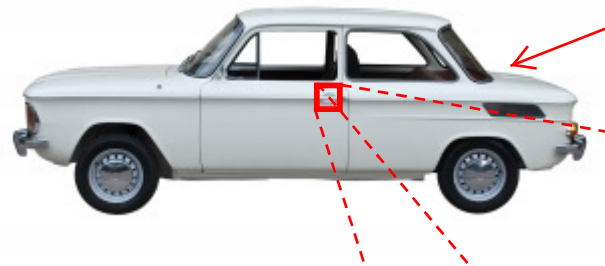
x_{100}

computer vision :

What is this?

why computer vision can be difficult ?
we see this
computer see this

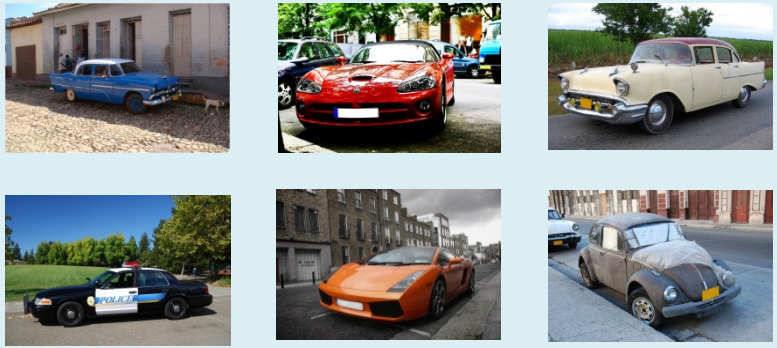
You see this:



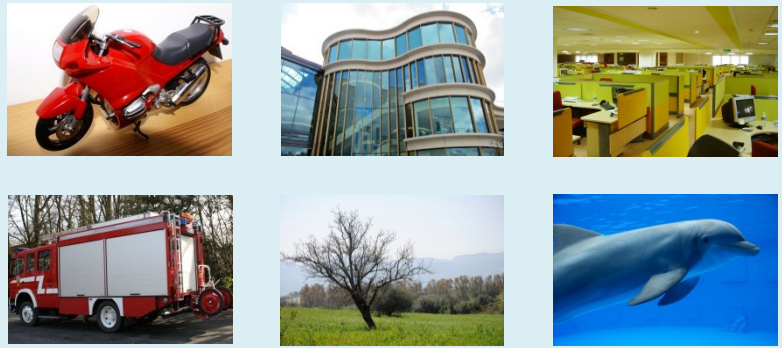
But the camera sees this:

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



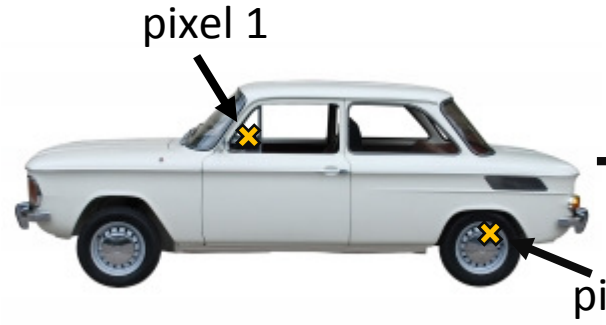
Cars



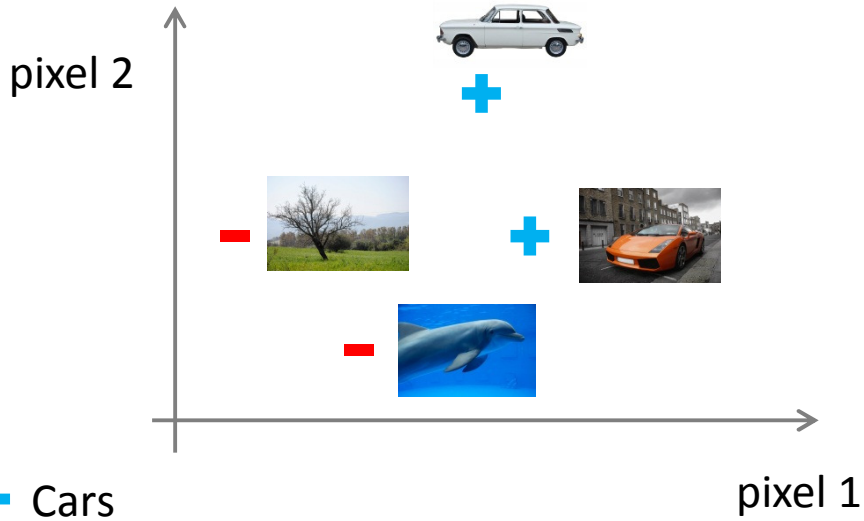
Not a car

Testing:

What is this?

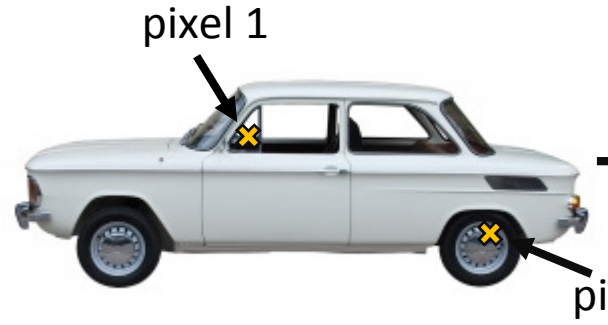


Learning Algorithm

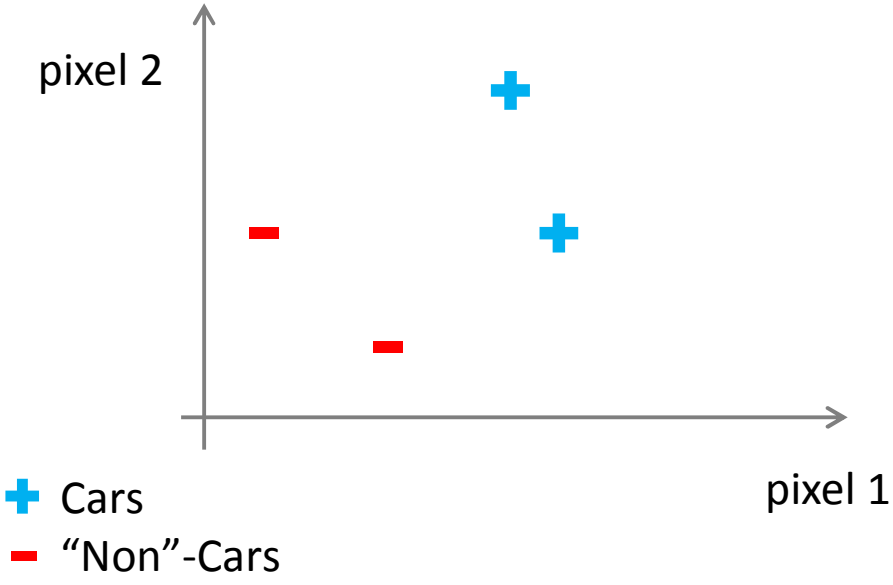


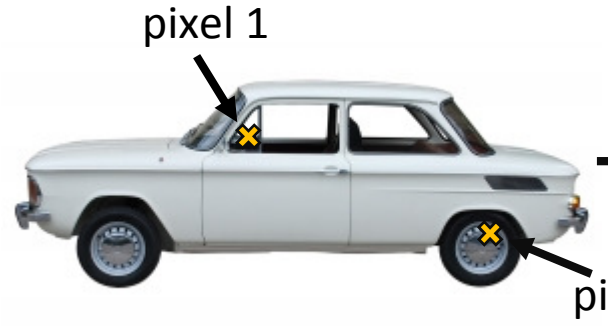
+ Cars

- “Non”-Cars



Learning Algorithm





Learning Algorithm

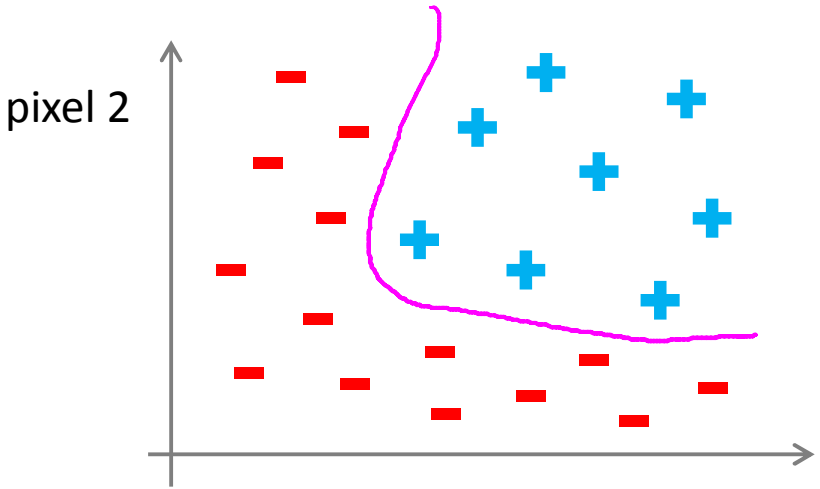
if we use 50x50 pixel images

50 x 50 pixel images→ 2500 pixels

$n = 2500$ (7500 if RGB)

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

0-255



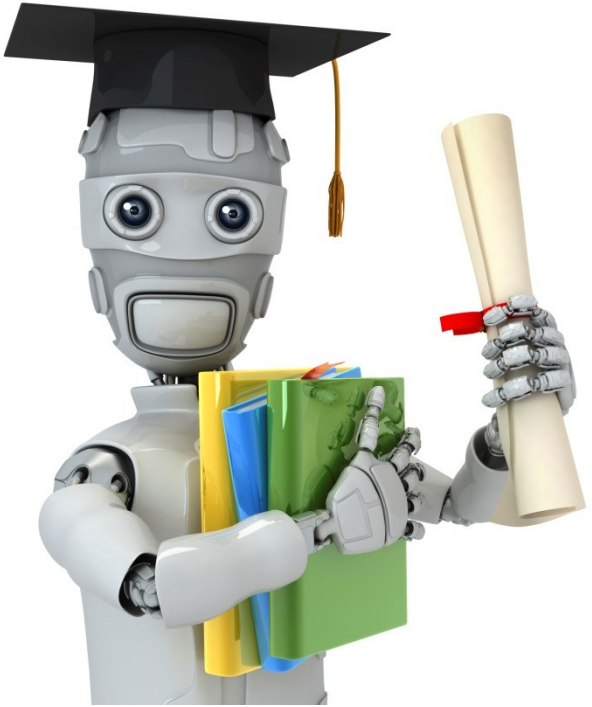
Cars

"Non"-Cars

if

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

too many features and cannot be handles !!



Machine Learning

Neural Networks: Representation

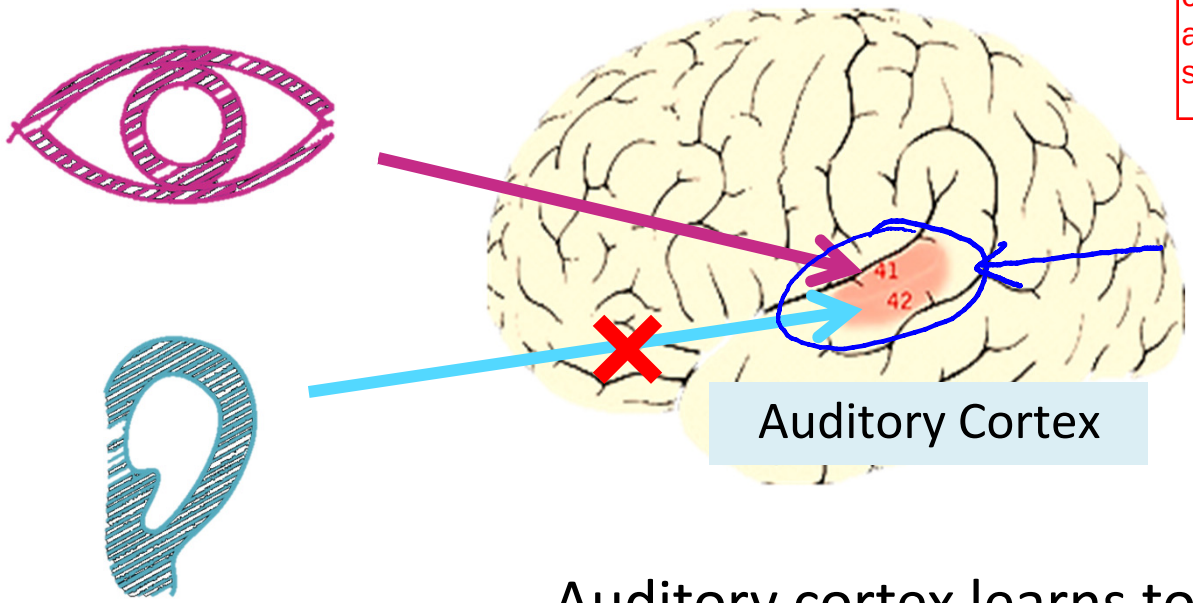
Neurons and the brain

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

cut the wire from ear to the auditory cortex and rewire eye to the brain, so the signal from the eye gets routed to the auditory cortex, and it turns our the auditory cortex will learn to see

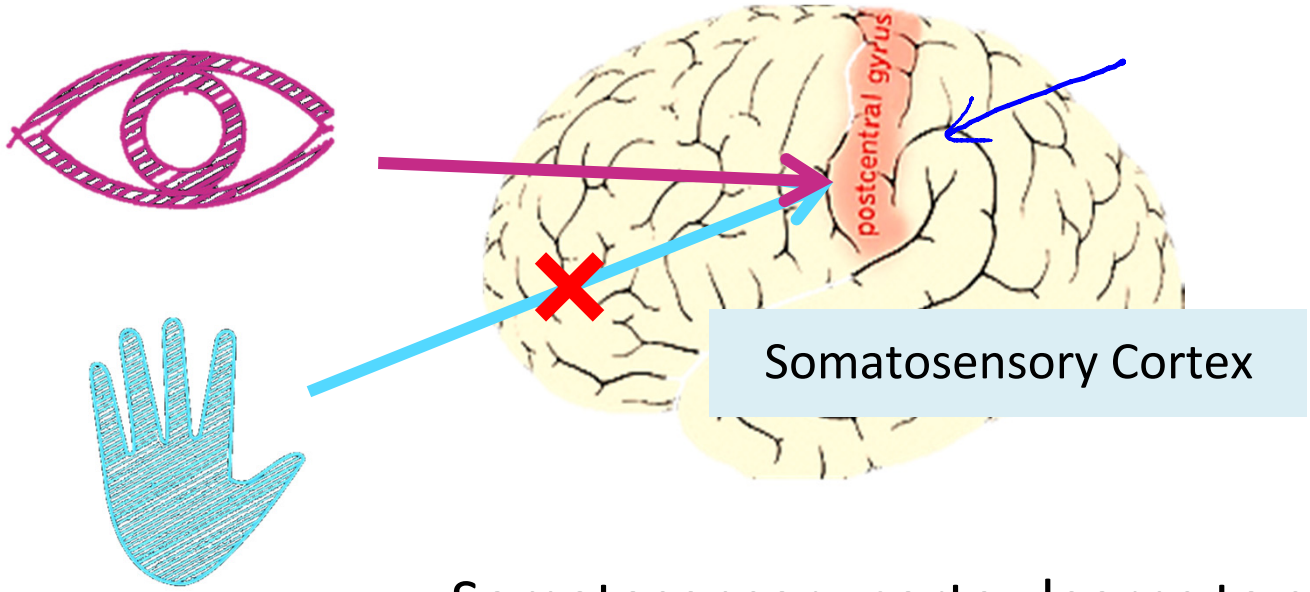


Auditory cortex learns to see

[Roe et al., 1992]

neural-rewiring experiments

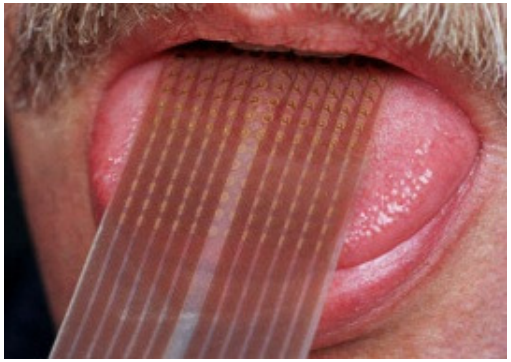
The "one learning algorithm" hypothesis



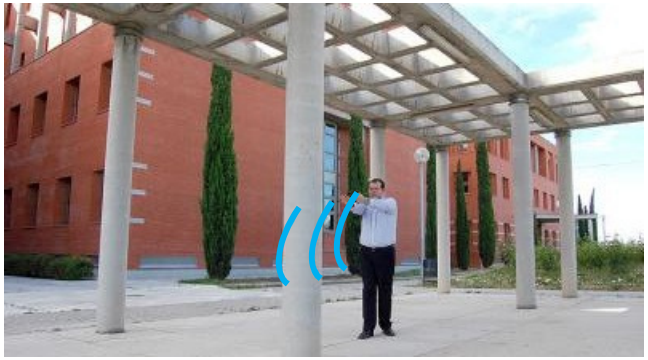
Somatosensory cortex learns to see

[Metin & Frost, 1989]

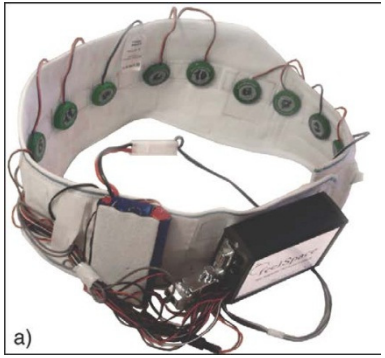
Sensor representations in the brain



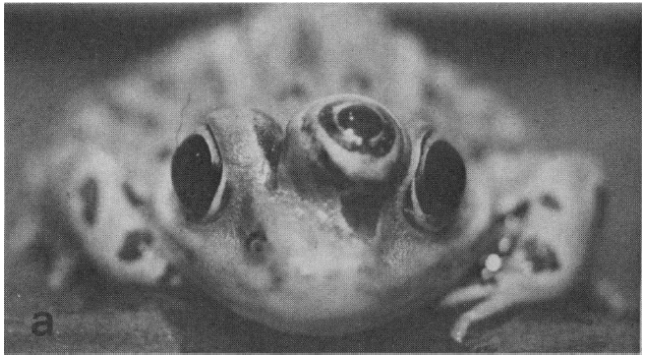
Seeing with your tongue



Human echolocation (sonar)

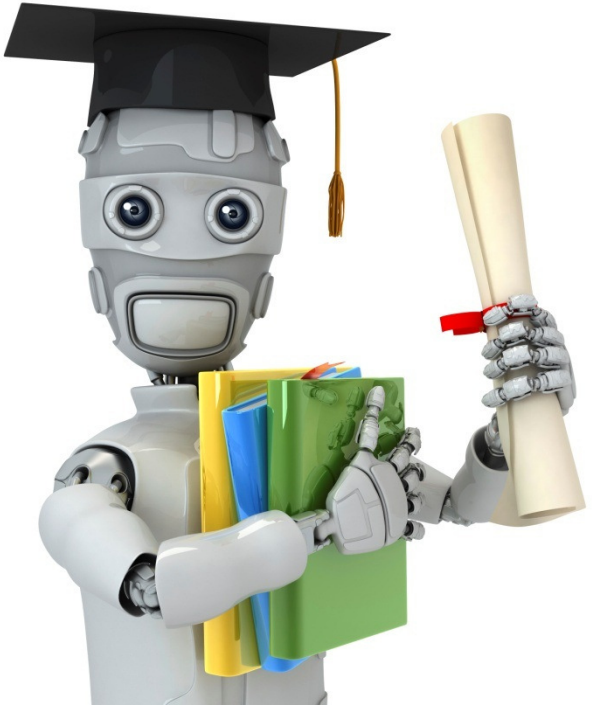


Haptic belt: Direction sense



Implanting a 3rd eye

[BrainPort; Welsh & Blasch, 1997; Nagel et al., 2005; Constantine-Paton & Law, 2009]



Machine Learning

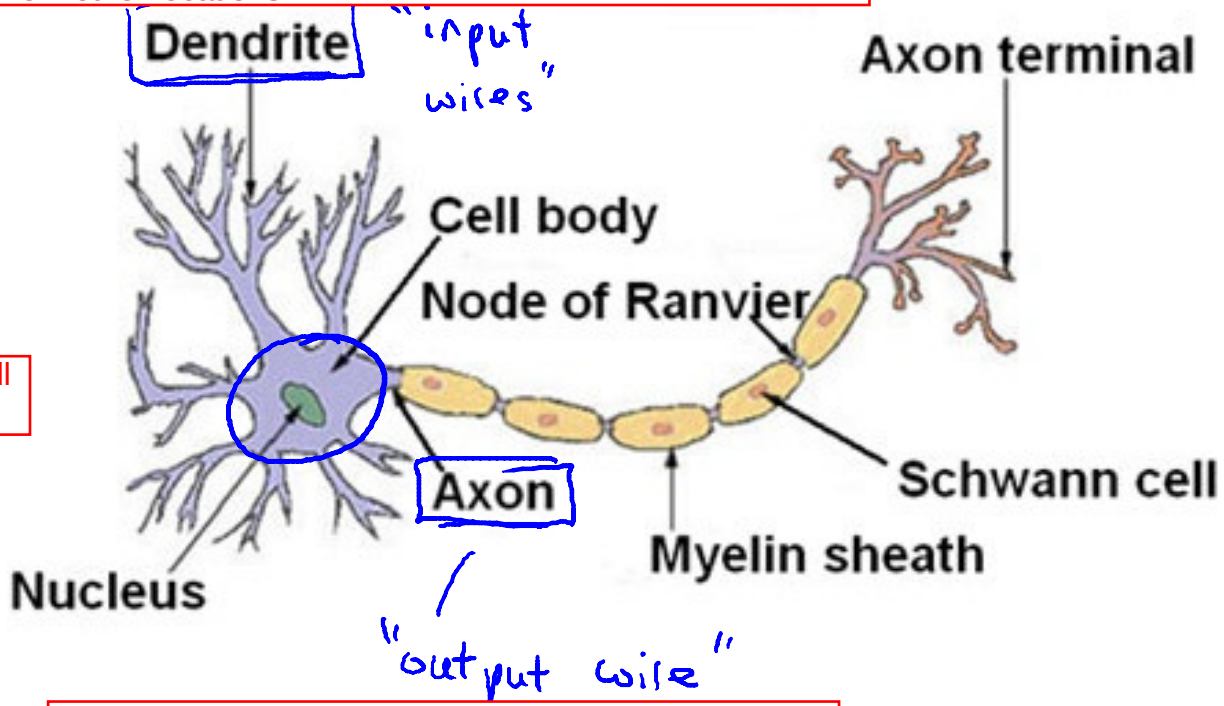
Neural Networks: Representation

Model representation I

at a simplistic level what a neuron is, is a computational unit that gets a number of inputs through it input wires and does some computation and then it says outputs via its axon to other nodes or to other neurons in the brain.

Neuron in the brain

the neuron has a number of input wires, and these are called the dendrites. You think of them as input wires, and these receive inputs from other locations.

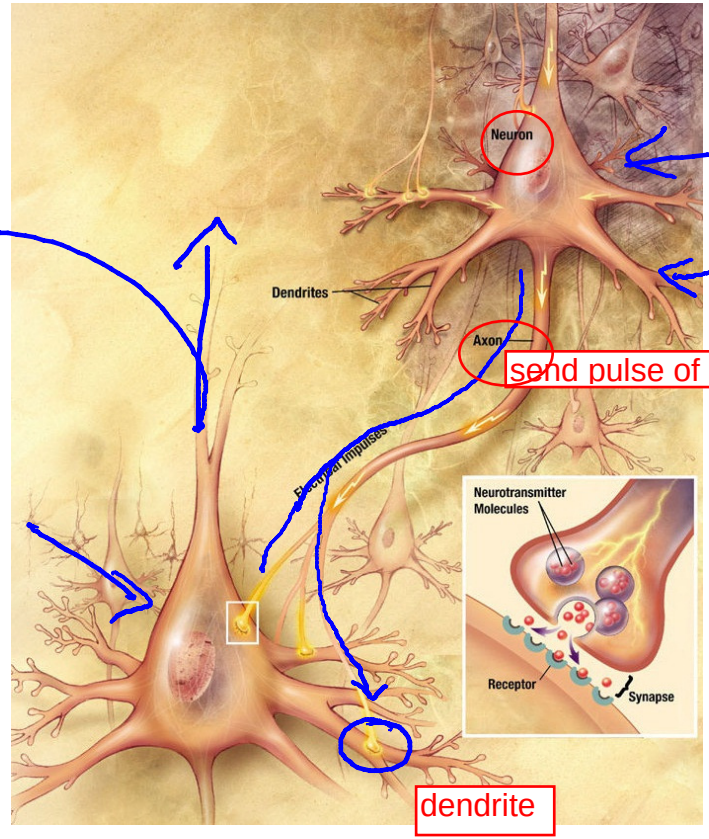


The neuron has a cell body

And a neuron also has an output wire called an Axon, and this output wire is what it uses to send signals to other neurons, so to send messages to other neurons.

Neurons in the brain

this is also how sensors and muscle works as well



muscle

eye (sensor) send pulse to the brain

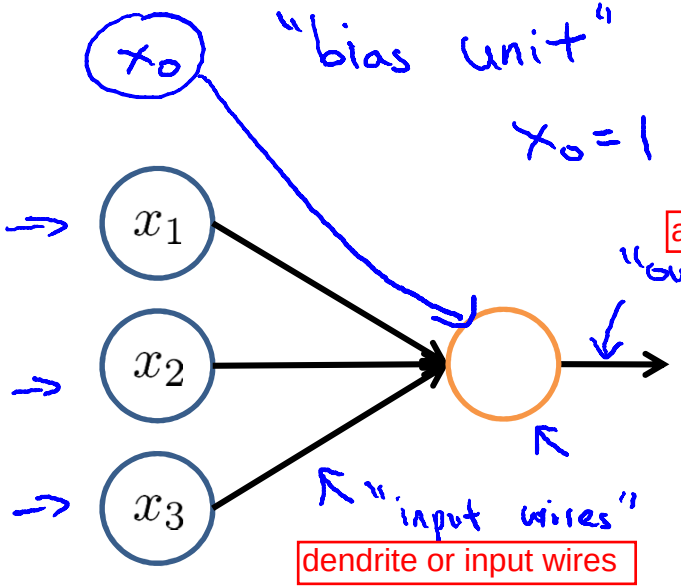
send pulse of electricity

dendrite

[Credit: US National Institutes of Health, National Institute on Aging]

single artificial neuron

Neuron model: Logistic unit



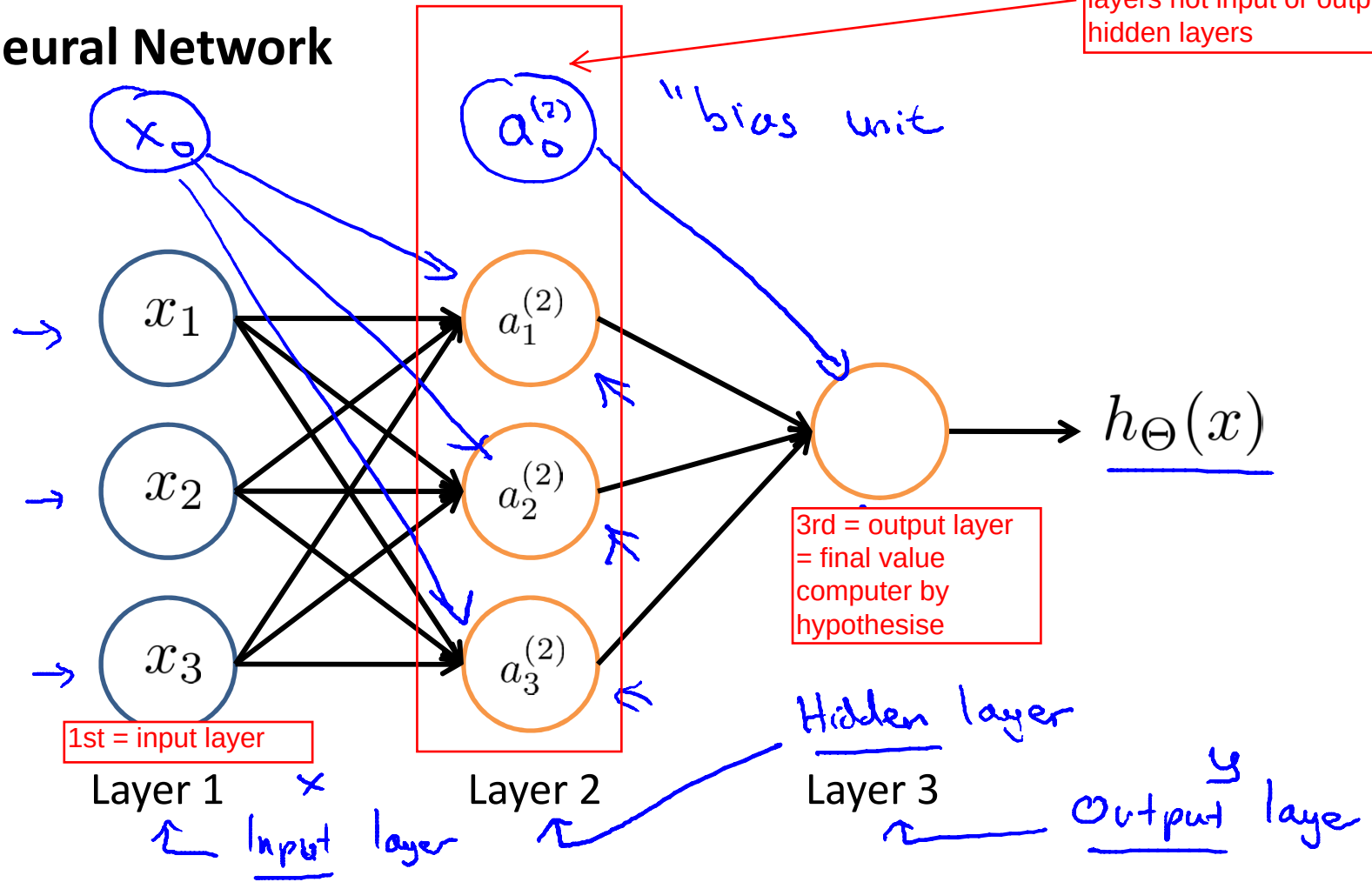
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

"weights" (parameters)

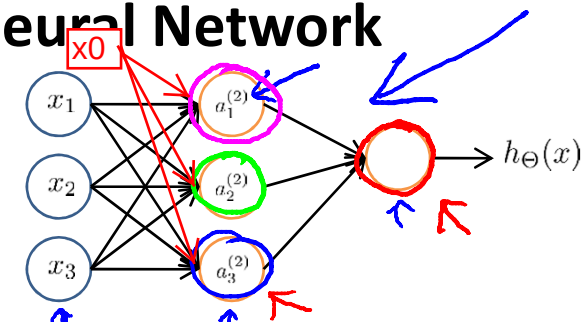
Sigmoid (logistic) activation function.

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

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 $j+1$

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

$h_\Theta(x)$

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

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

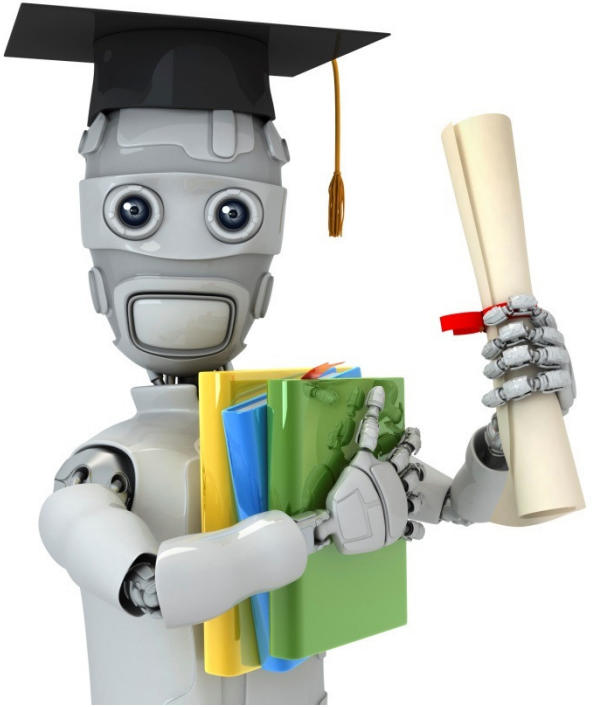
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$h_\Theta(x)^{(2)}$

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

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

$s_{j+1} \times (s_j + 1)$

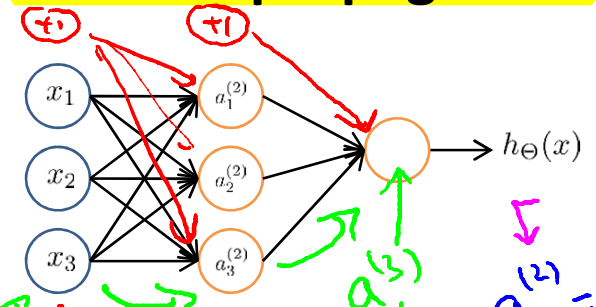


Machine Learning

Neural Networks: Representation

Model representation II

Forward propagation: Vectorized implementation



$a^{(1)} = x$

$a_1^{(2)} = g(z_1^{(2)})$

$z_1^{(2)}$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$z_2^{(2)}$, $z_3^{(2)}$

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

$a_3^{(2)} = g(z_3^{(2)})$ → $z^{(3)}$

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

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

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

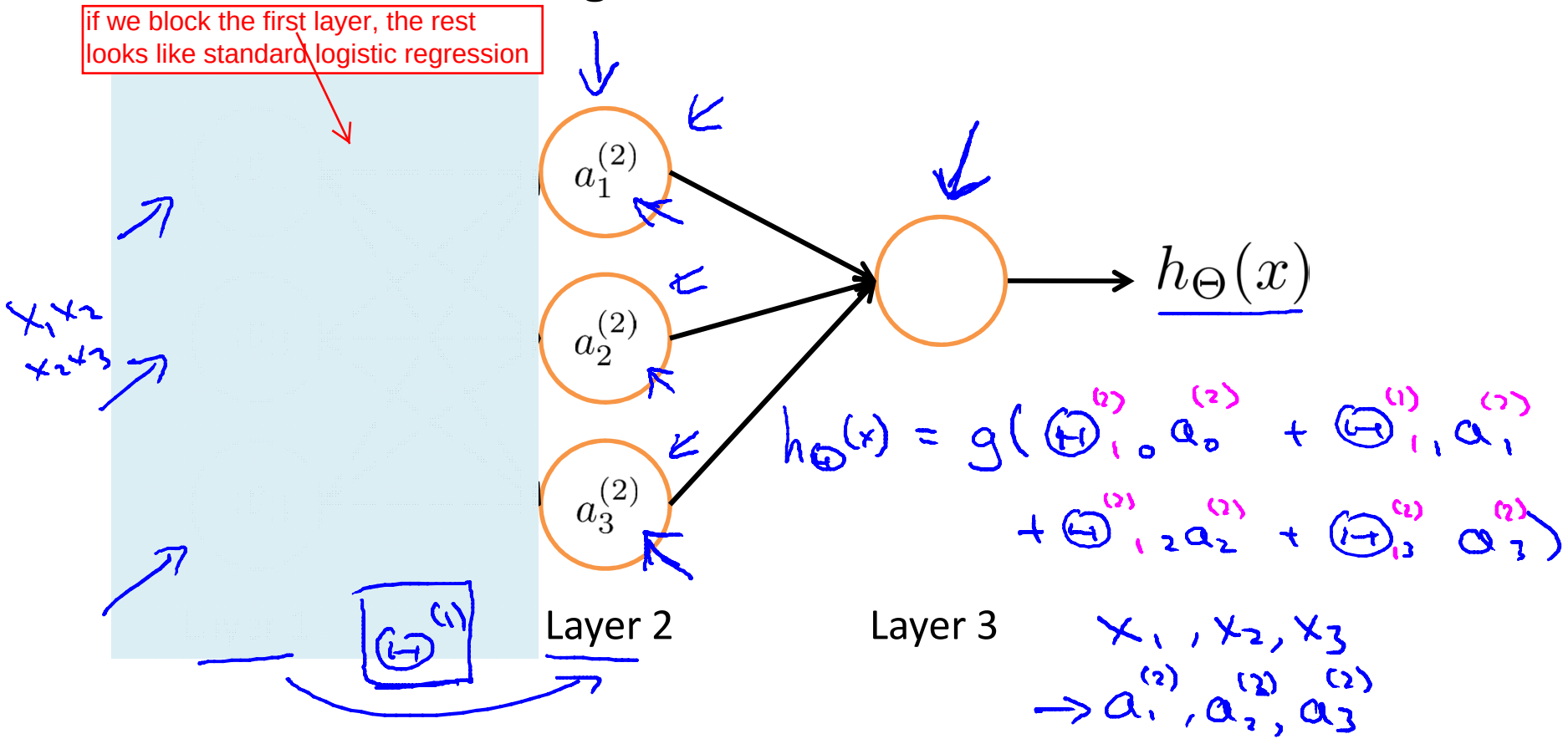
$a^{(2)} \in \mathbb{R}^3$, $z^{(2)} \in \mathbb{R}^3$

Add $a_0^{(2)} = 1.$ → $a^{(2)} \in \mathbb{R}^4$

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

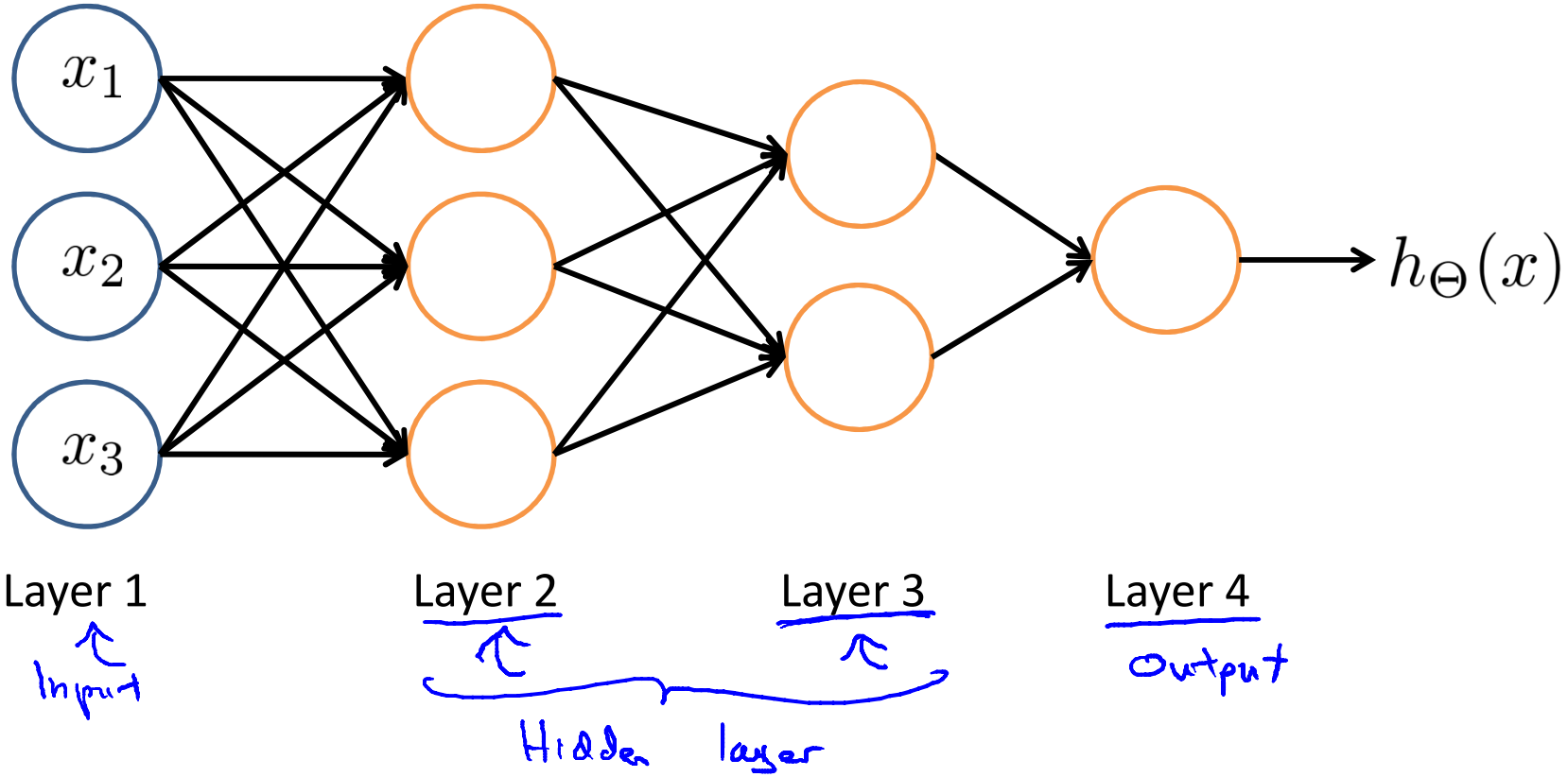
bias unit in the hidden layer

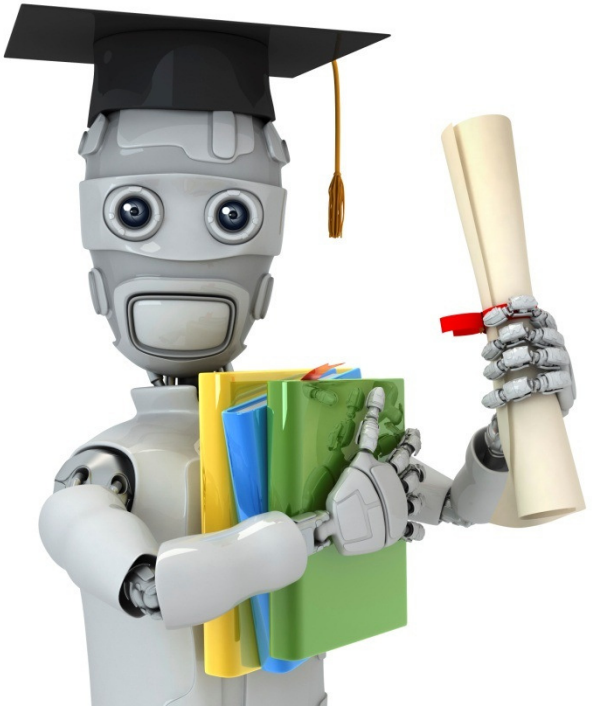
Neural Network learning its own features



Other network architectures

how neuron can connect to each other





Machine Learning

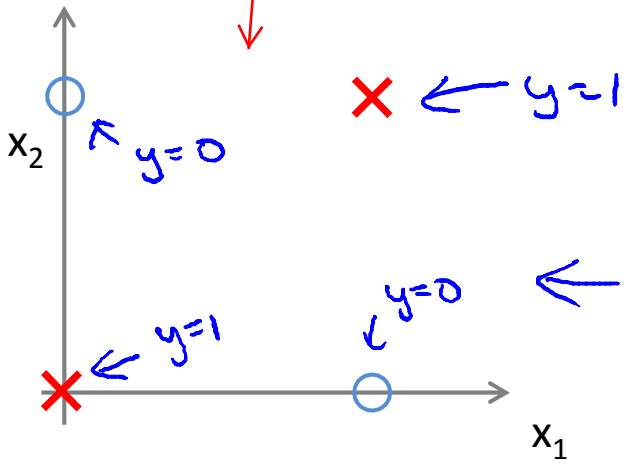
Neural Networks: Representation

Examples and intuitions I

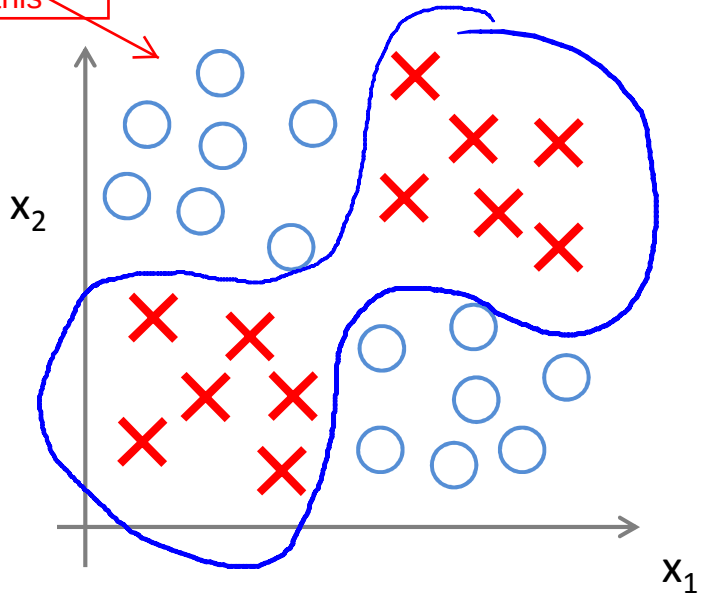
xor=exclusive or=A不等於B: 兩數相同為否兩數不同為真
XNOR=相同為1,相異為0

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).



an simplified example of this



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

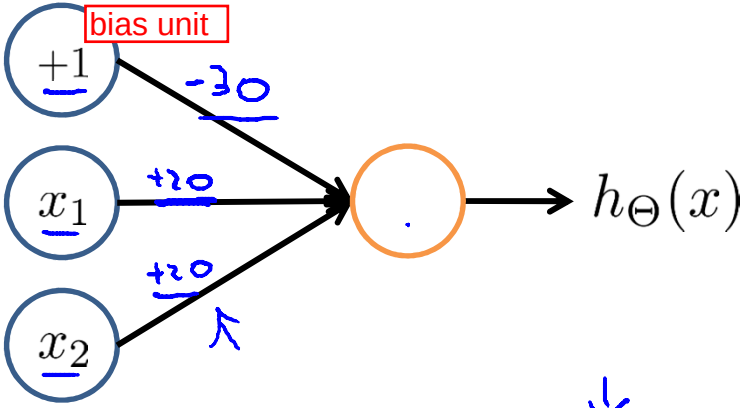
$$x_1 \text{ XNOR } x_2$$

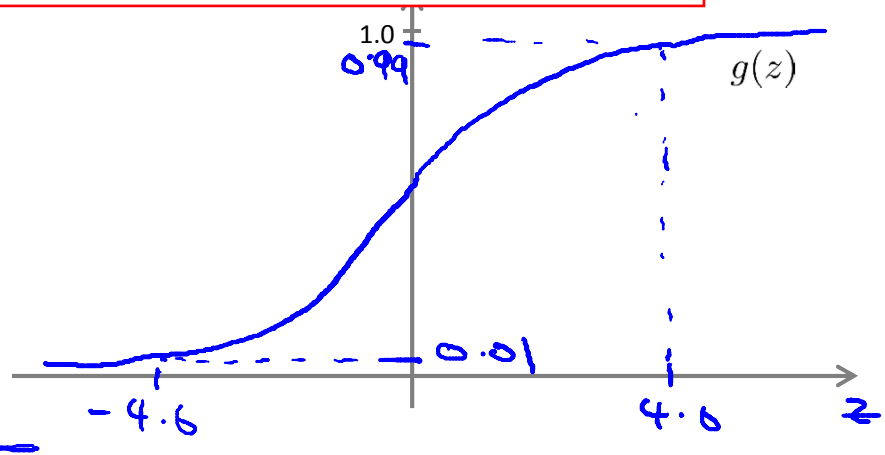
$$\text{NOT } (x_1 \text{ XOR } x_2)$$

AND:仅当输入均为高電壓（1）时，输出才为高電壓（1）时；若输入中至多有一个高電壓时，则输出为低電壓。

Simple example: AND

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





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

$\Theta^{(1)}_{10}$ $\Theta^{(1)}_{11}$ $\Theta^{(1)}_{12}$

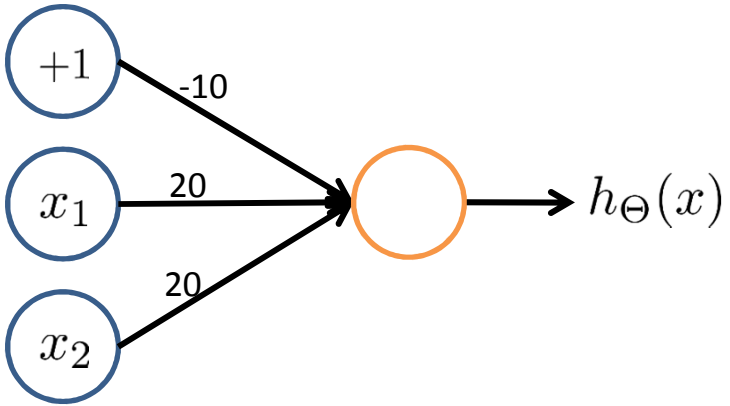
its logical AND

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	$g(10) \approx 1$

$h_\Theta(x) \approx x_1$ AND x_2

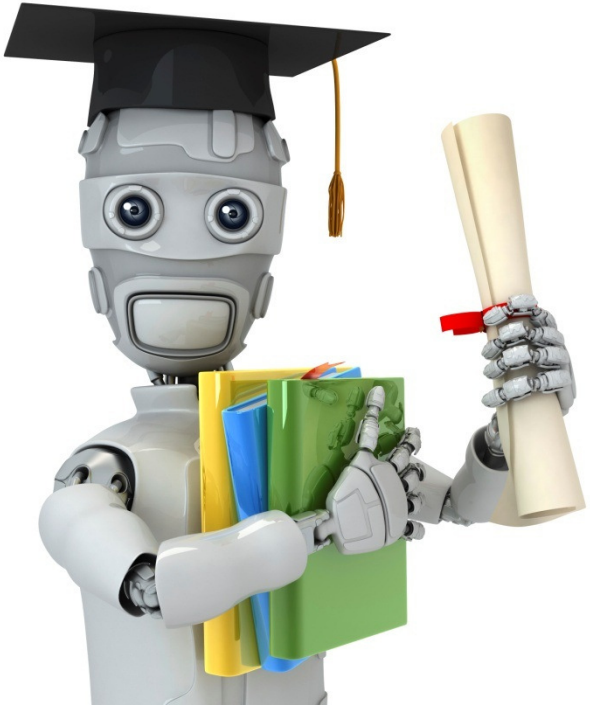
OR:**只要两个**输入中至少有一个为高电平（1），则输出为高电平（1）；**若两个**输入均为低电平（0），输出才为低电平（0）

Example: OR function



$g(-10 + 20x_1 + 20x_2)$

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



Machine Learning

Neural Networks: Representation

Examples and intuitions II

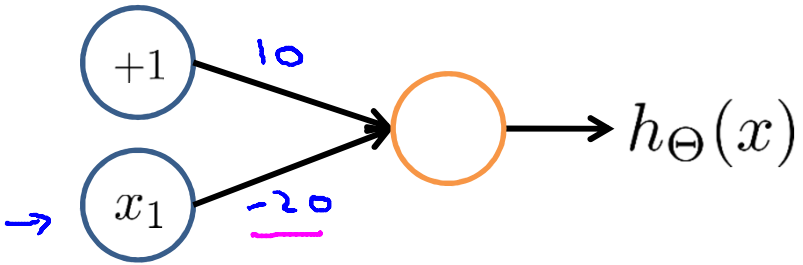
→ x_1 AND x_2

→ x_1 OR x_2

{0,1}.

Negation: =

NOT x_1



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

$h_\Theta(x) = g(10 - 20x_1)$

x1=0 x2=0

→ (NOT x_1) AND (NOT x_2)

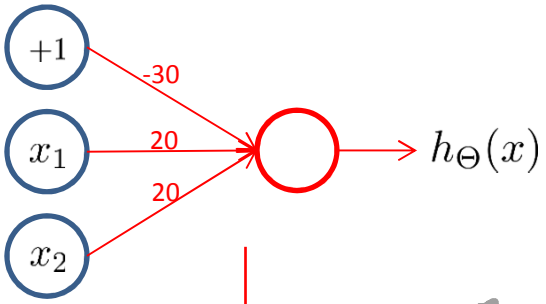
(=1 if and only if

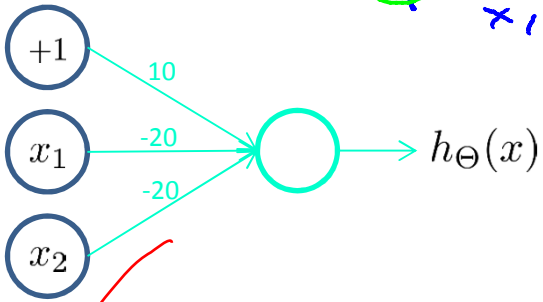
→ $x_1 = x_2 = 0$

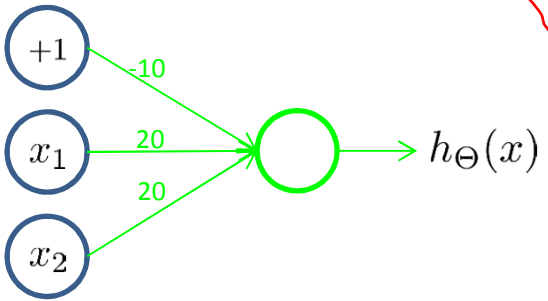
XNOR=相同為1,相異為0

Putting it together: x_1 XNOR x_2

x1 xnor x2



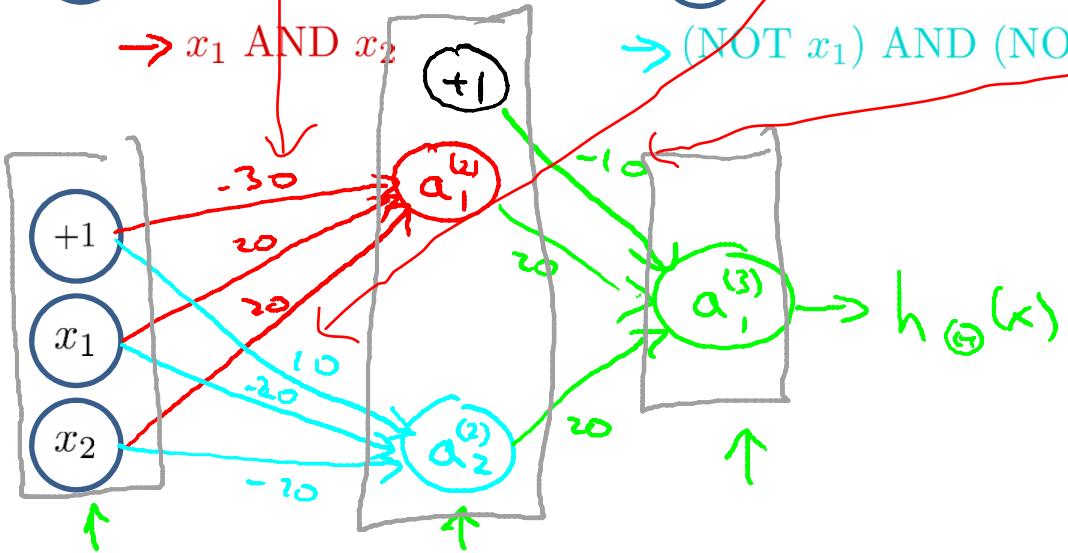




→ x_1 AND x_2

→ (NOT x_1) AND (NOT x_2)

→ x_1 OR x_2



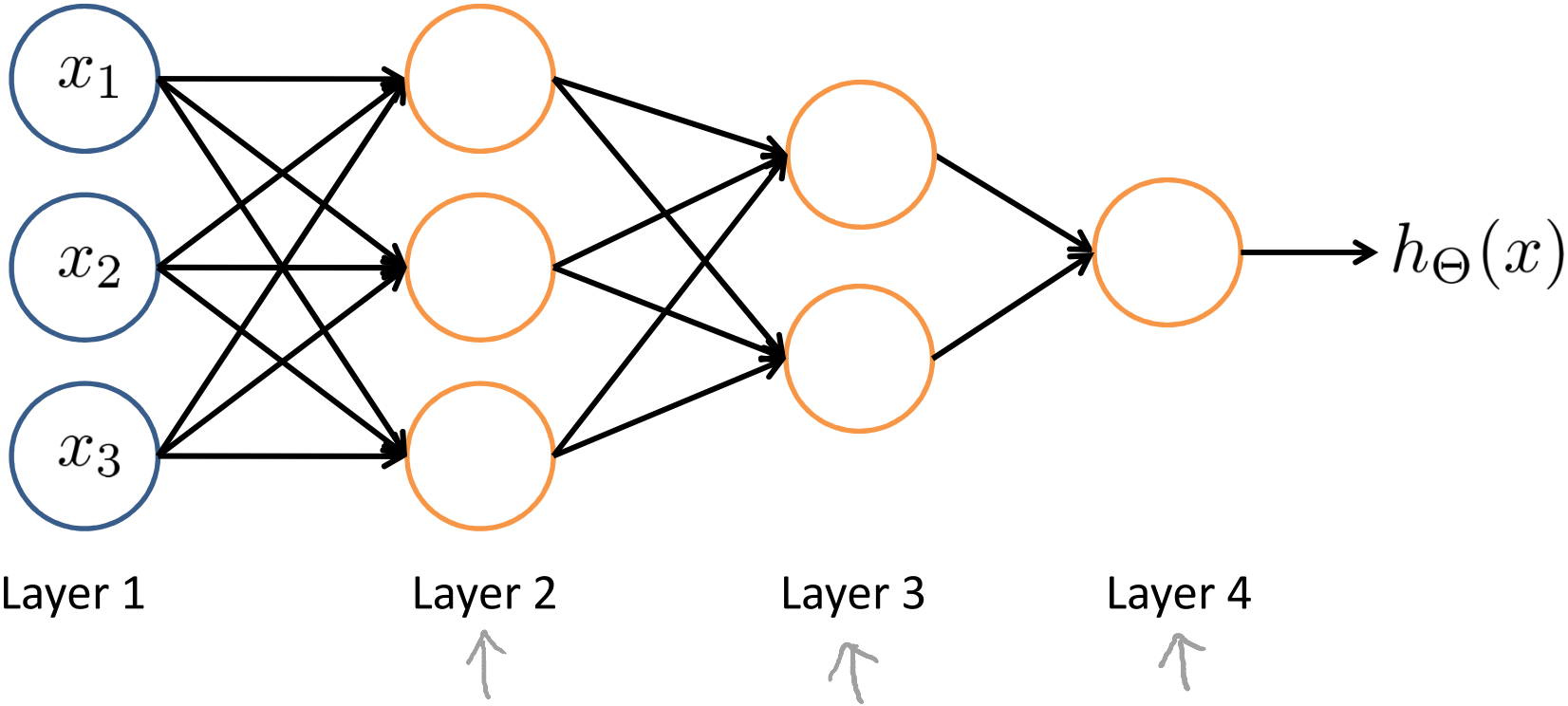
x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
→ 0	0	0	1	1 ←
0	1	0	0	0
1	0	0	0	0
→ 1	1	1	0	1 ←

nonlinear decision boundary that compute xnor function

ew Ng

Neural Network intuition

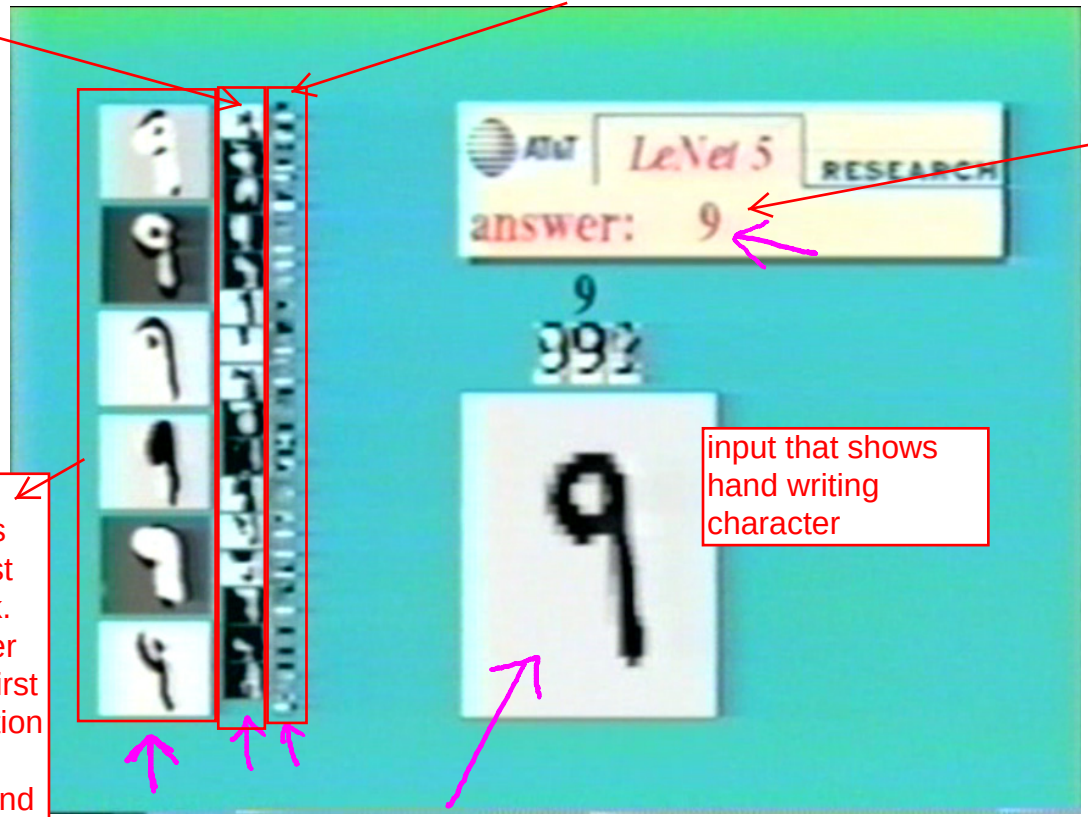
multiple layers can compute complicated functions



Handwritten digit classification

and that's a visualization of why the next hidden layer is confusing. You probably have a hard time seeing what's going on much beyond the first hidden layer.

This is a visualization of the next hidden layer. It's kinda harder to see, harder to understand the deeper, hidden layers,



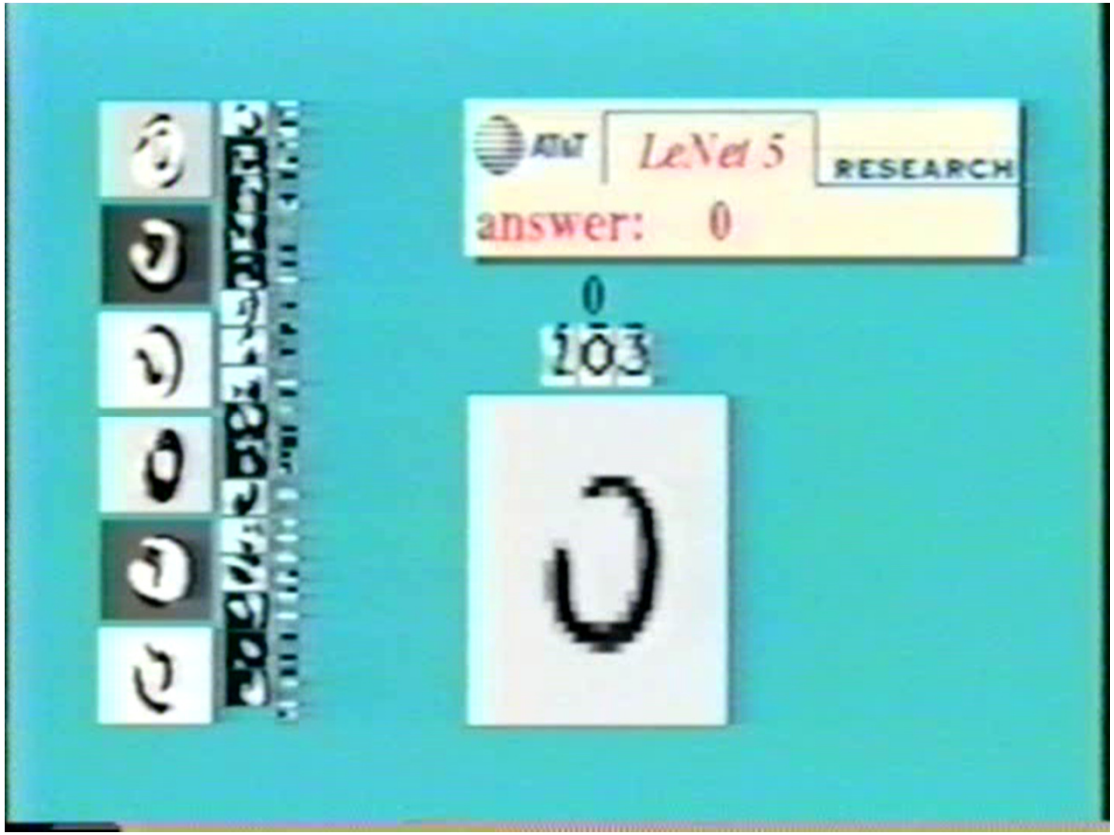
but then finally, all of these learned features get fed to the upper layer. And shown over here is the final answer, it's the final predictive value for what handwritten digit the neural network thinks it is being shown.

input that shows hand writing character

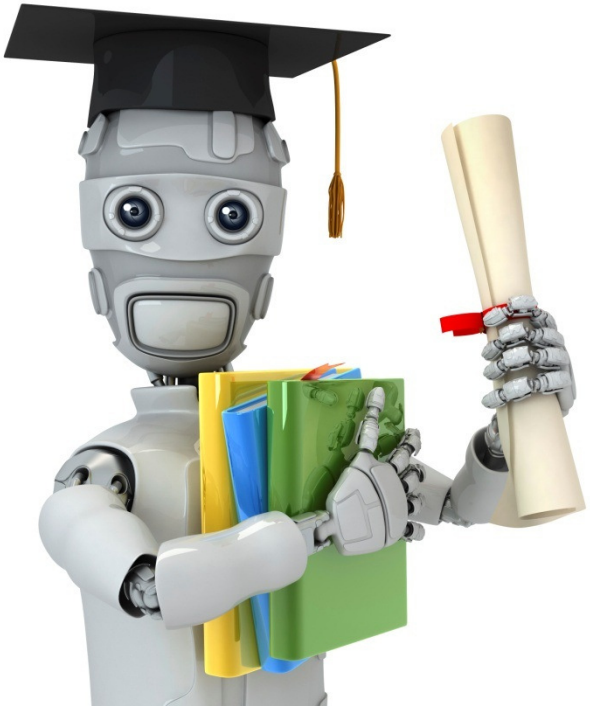
This column here shows a visualization of the features computed by sort of the first hidden layer of the network. So that the first hidden layer of the network and so the first hidden layer, this visualization shows different features. Different edges and lines and so on detected.

[Courtesy of Yann LeCun]

Handwritten digit classification



[Courtesy of Yann LeCun]



Machine Learning

Neural Networks: Representation

Multi-class classification

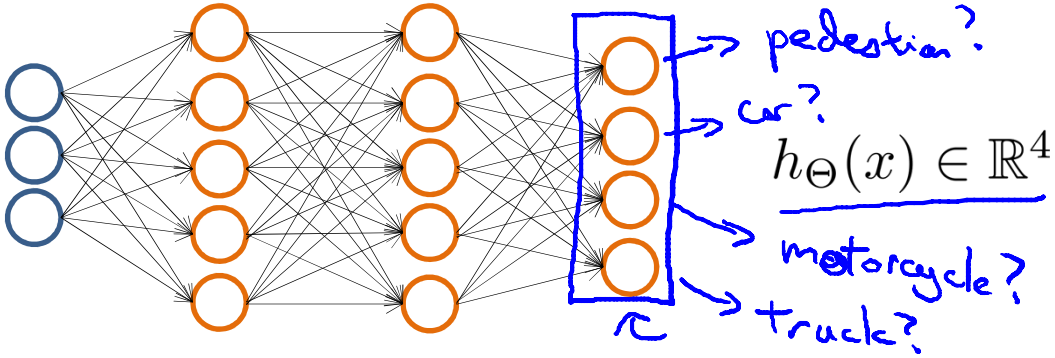
Multiple output units: One-vs-all.

Pedestrian

Car

Motorcycle

Truck

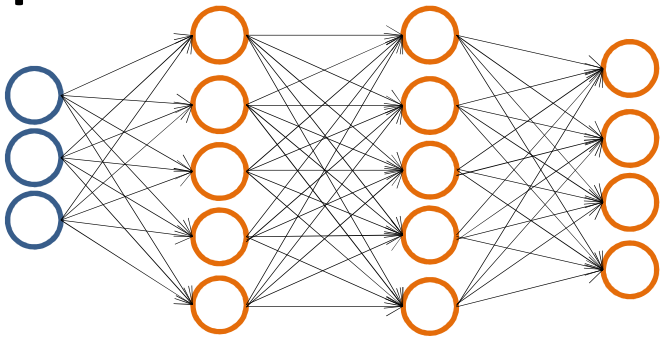


Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Multiple output units: One-vs-all.

where Xi is an image with one of the four objects and Yi will be one of these vectors.



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

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

$y^{(i)}$ 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

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

Previously $y \in \{1, 2, 3, 4\}$

$h_\Theta(x^{(i)}) \approx y^{(i)}$

$\in \mathbb{R}^4$