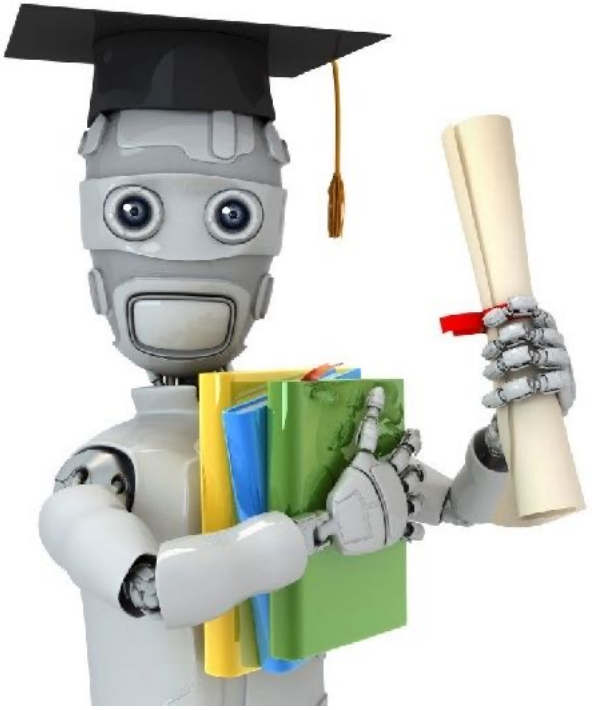


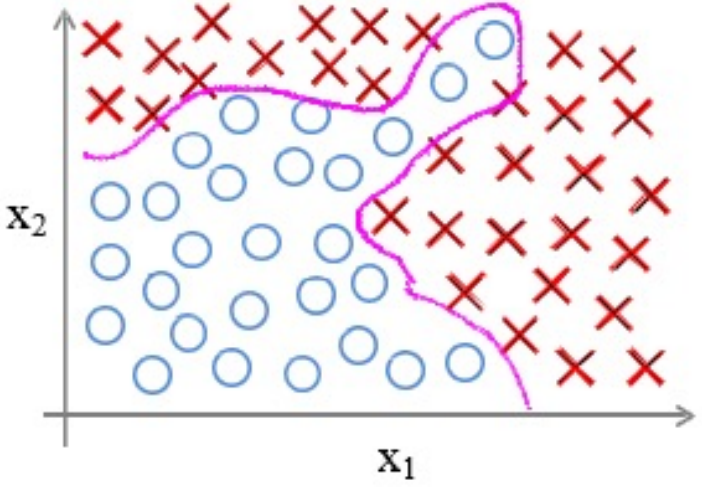
Machine Learning

Neural Networks: Representation

Non-linear hypotheses

https://medium.com/@Coursesteach/machine-learning-part-34-non-linear-hypotheses-273044b7bfdb

Non-linear Classification



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

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

$n = 100$

$\Rightarrow x_1^2, x_1x_2, x_1x_3, x_1x_4 \dots x_1x_{100}$
$x_2^2, x_2x_3 \dots$
≈ 5000 feature $O(n^2)$

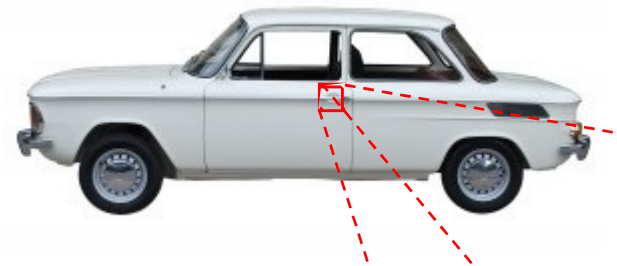
$\Rightarrow x_1^2, x_2^2, x_3^2, \dots, x_{100}^2$ $\approx \frac{n^2}{2}$

$\Rightarrow x_1x_2x_3, x_1^2x_2, x_{10}x_{11}x_{17}, \dots$
$O(n^3)$ $170{,}000$

Andrew Ng

What is this?

You see this:

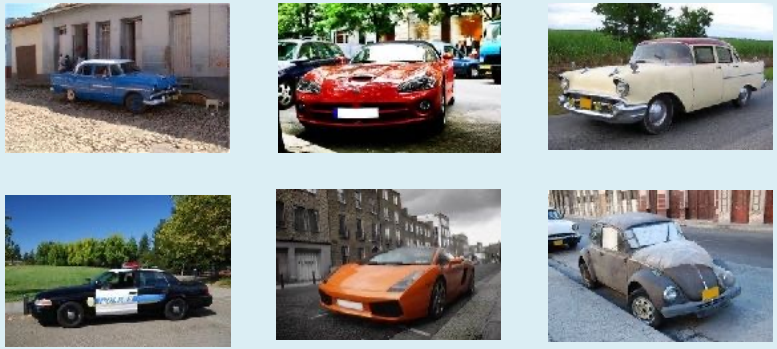


But the camera sees this:

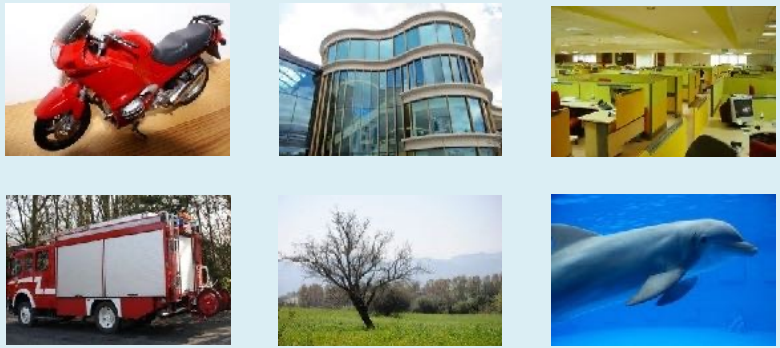
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	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

So, like, when we were looking at it, we realized we could come up with a ton of features, like maybe a hundred different features for different houses. If we wanted to include all the quadratic terms, like all of them, even all the second or polynomial terms, there would be a whole bunch.. There would be terms like x1 squared, x1x2, x1x3, you know, x1x4 up to x1x100 and then you have x2 squared, x2x3 and so on. And if you include just the second order terms, that is, the terms that are a product of, you know, two of these terms, x1 times x1 and so on, then, for the case of n equals 100, you end up with about five thousand features.

Computer Vision: Car detection



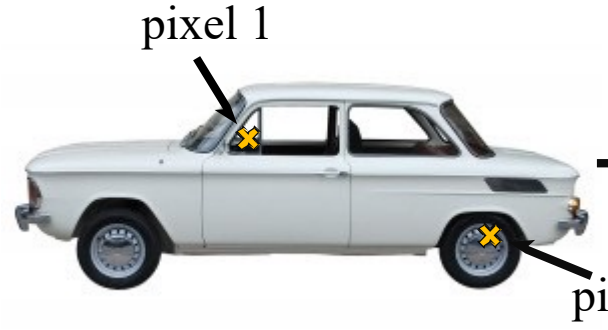
Cars



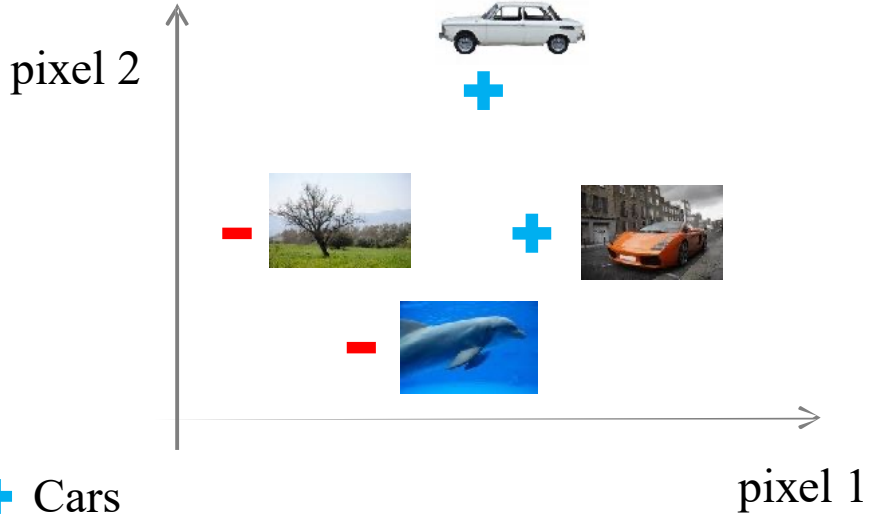
Not a car

Testing:

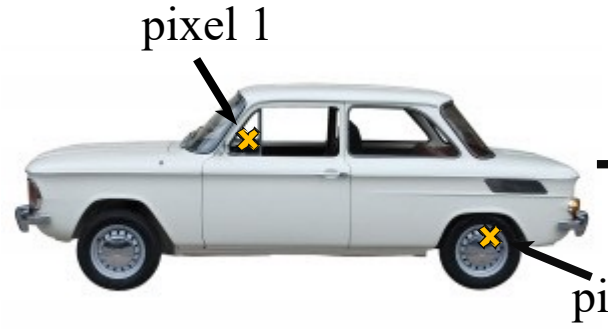
What is this?



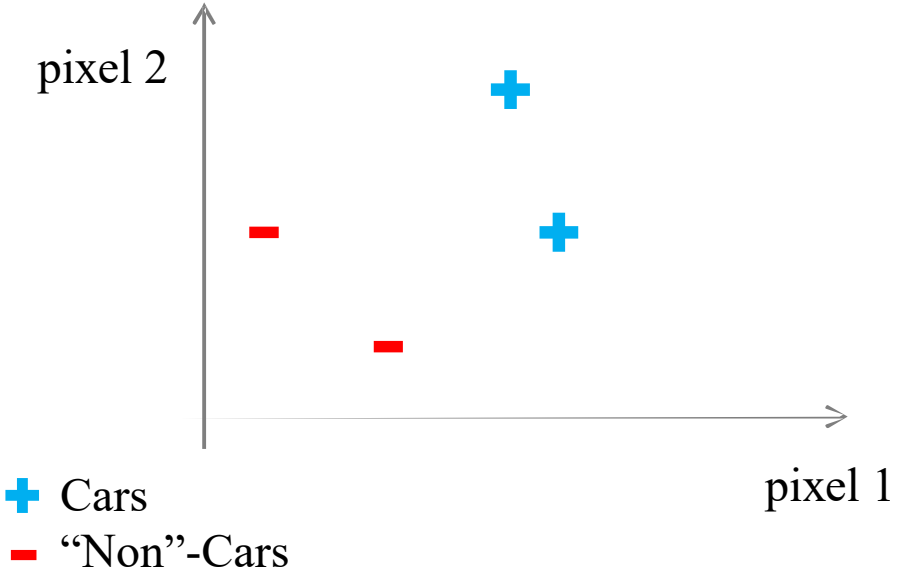
Learning Algorithm

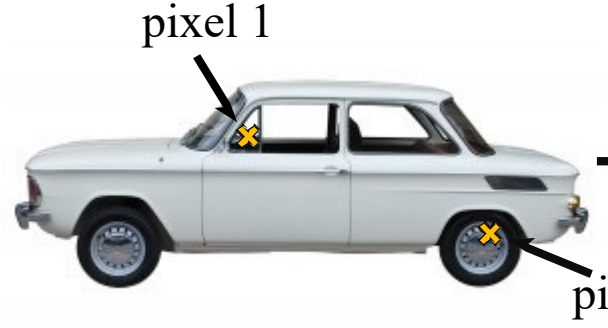


+ Cars
- "Non"-Cars

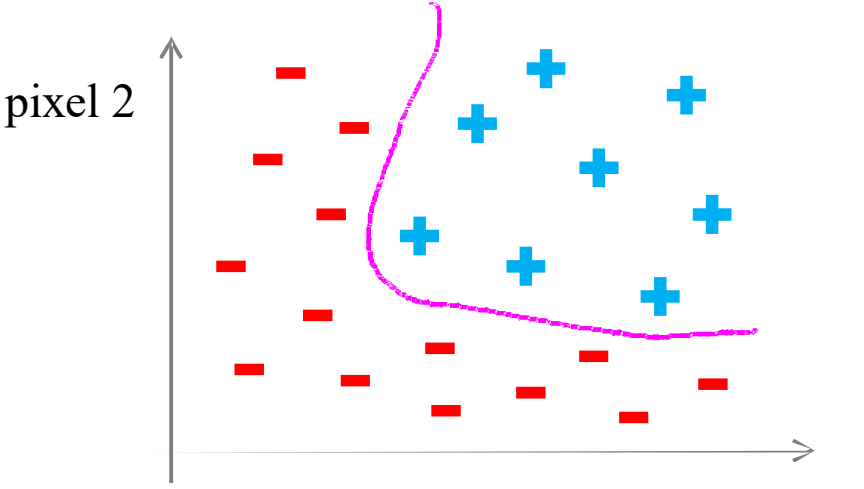


Learning Algorithm





Learning Algorithm



Cars

"Non"-Cars

50 x 50 pixel images→ 2500 pixels

$n = 2500$ (7500 if RGB)

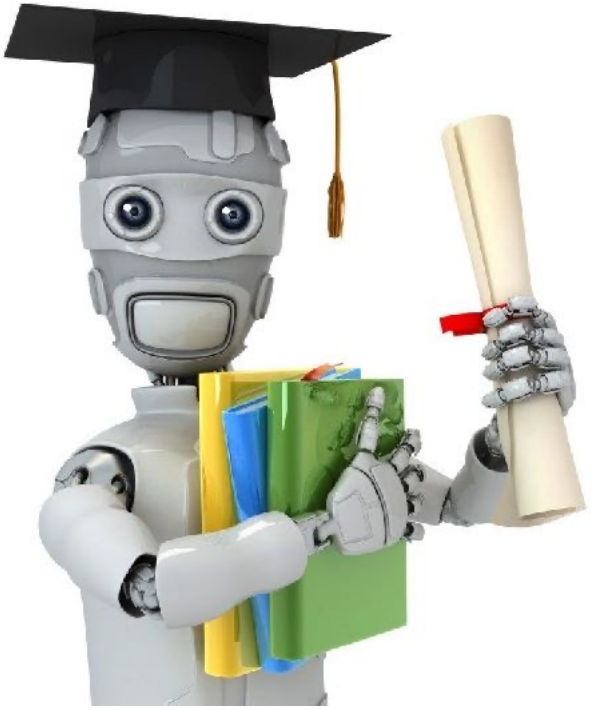
$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

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

Suppose you are learning to recognize cars from 100 x 100 pixel images (grayscale, not RGB). Let the features be pixel intensity values. If the train logistic regression include all the quadratic terms (x_i, x_j) as features, about how many features will you have?

1. 5,000
2. 100,000
3. 50 million (5×10^7)
4. 5 billion (5×10^9)

$N*(N+1)/2$



Machine Learning

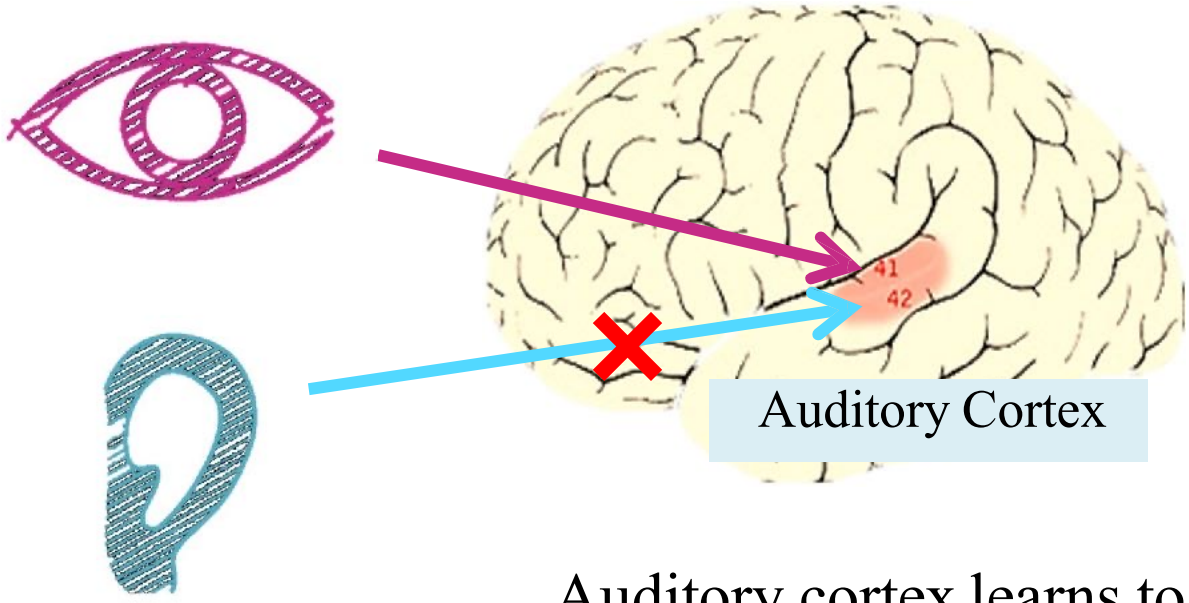
Neural Networks: Representation

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

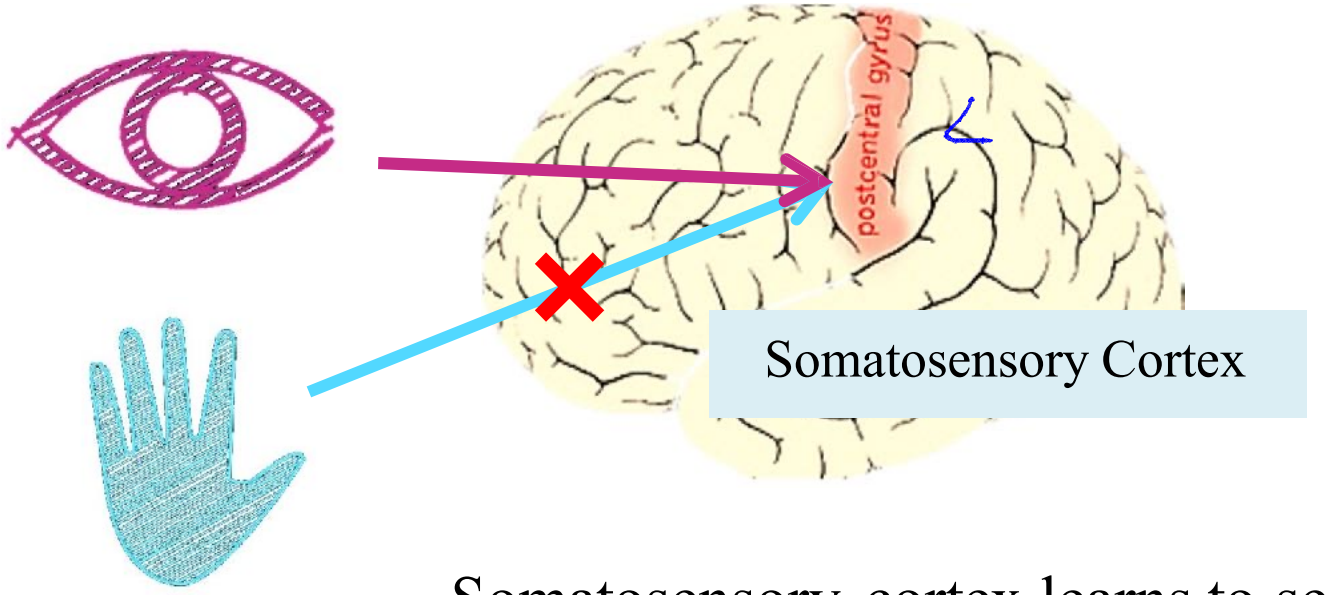
The "one learning algorithm" hypothesis



Auditory cortex learns to see

[Roe et al., 1992]

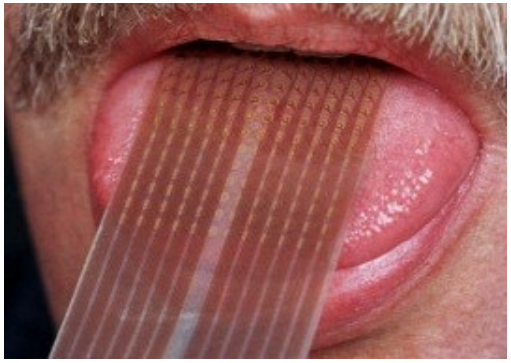
The "one learning algorithm" hypothesis



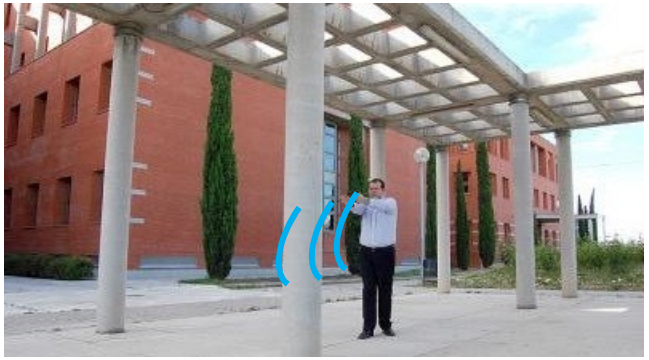
Somatosensory cortex learns to see

[Metin & Frost, 1989]

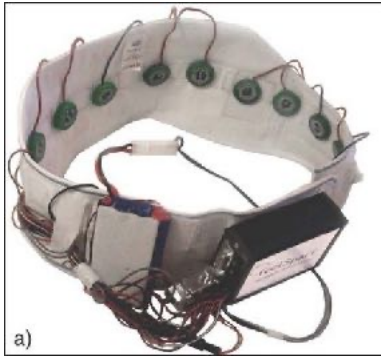
Sensor representations in the brain



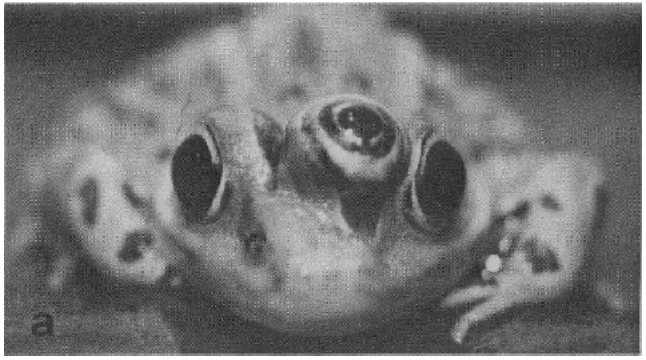
Seeing with your tongue



Human echolocation (sonar)

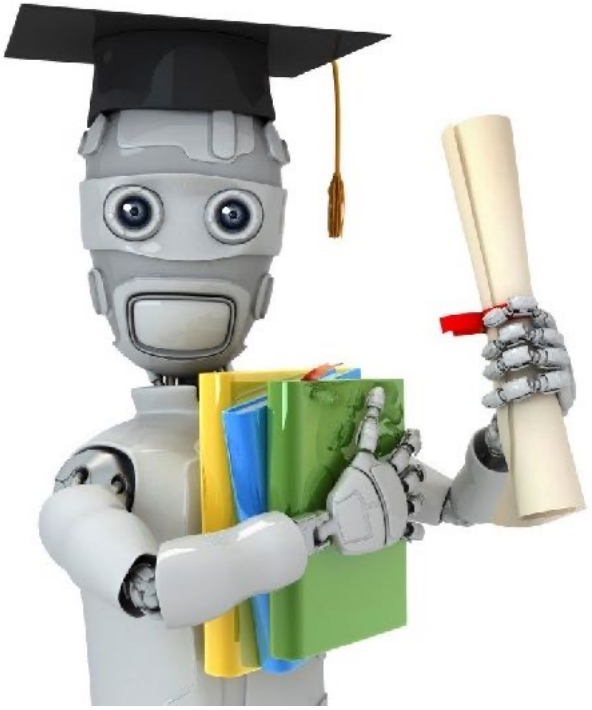


Haptic belt: Direction sense



Implanting a 3rd eye

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

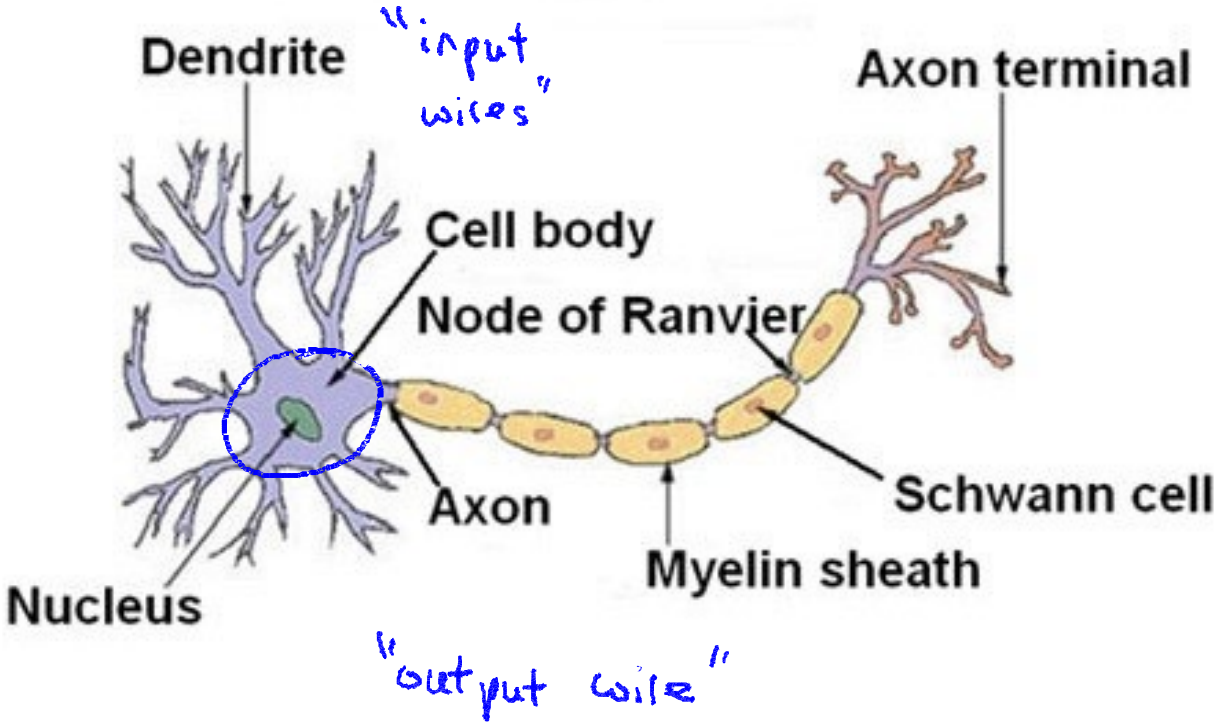


Machine Learning

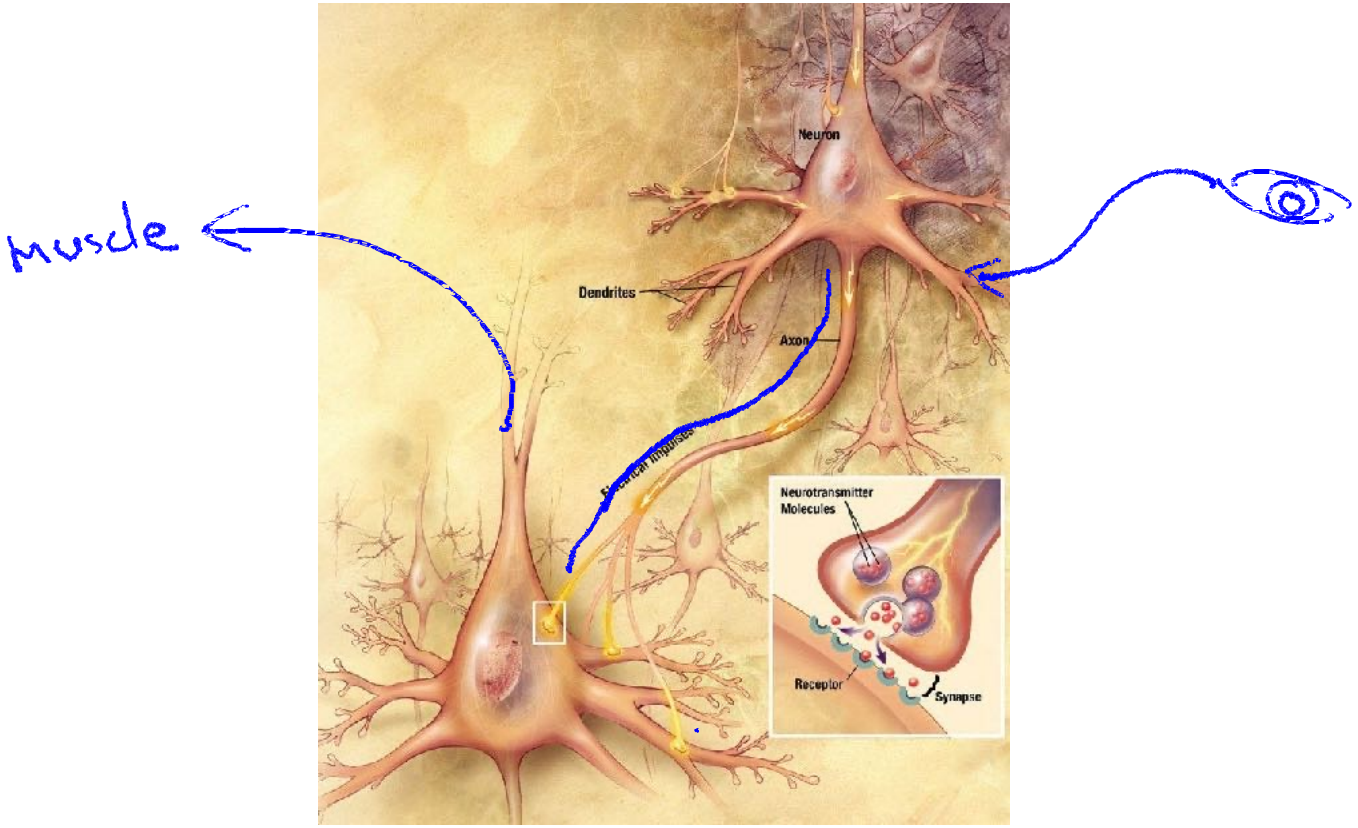
Neural Networks: Representation

Model representation I

Neuron in the brain

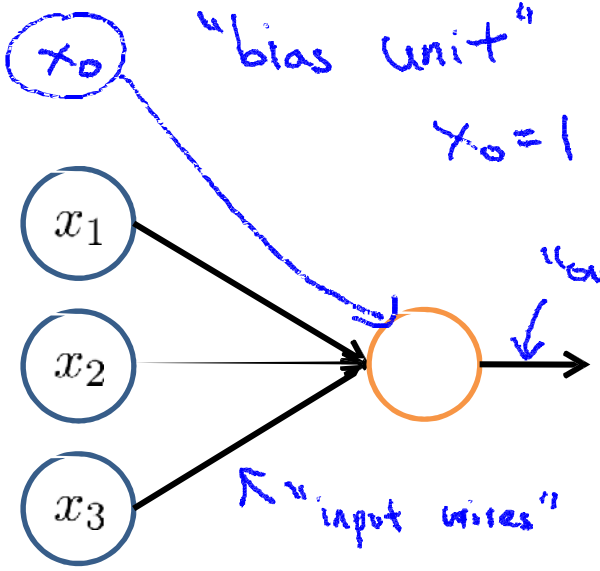


Neurons in the brain



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

Neuron model: Logisticunit



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

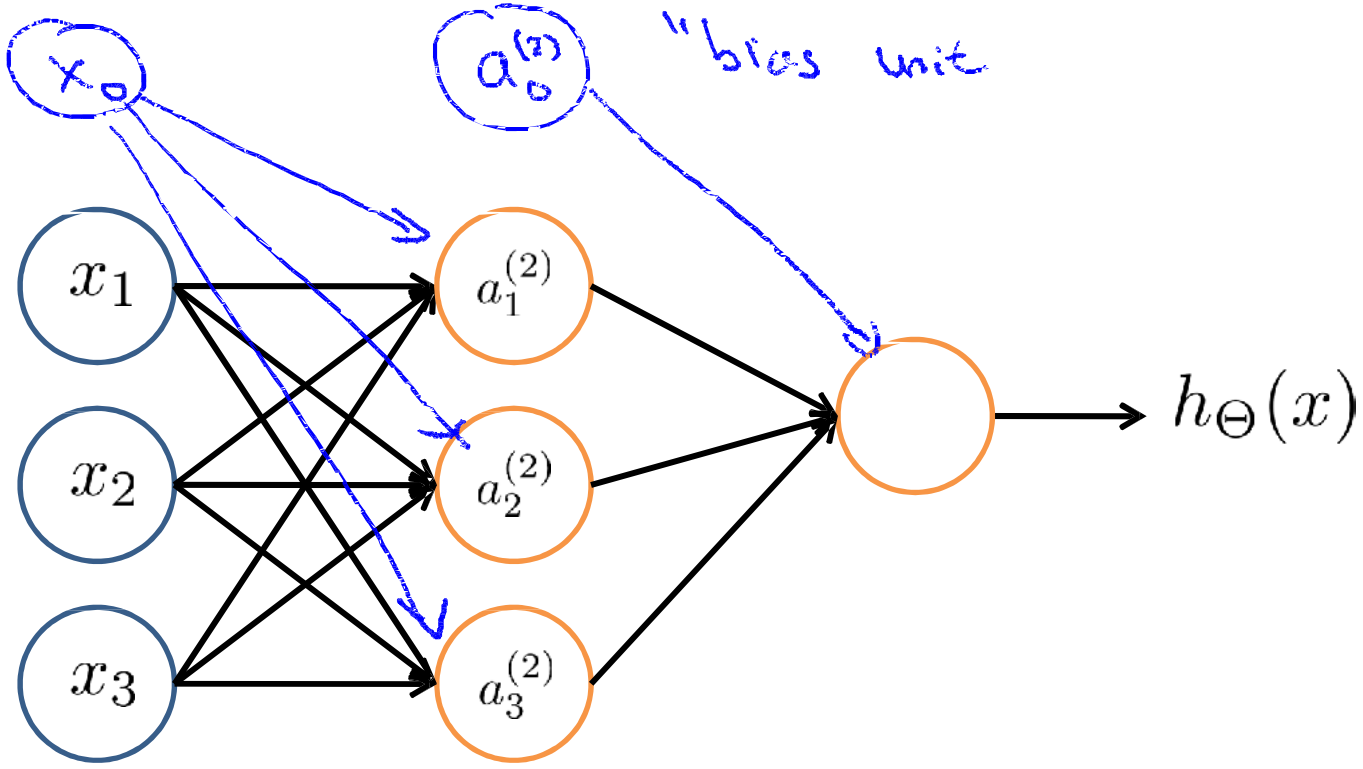
"weights" (parameters)

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

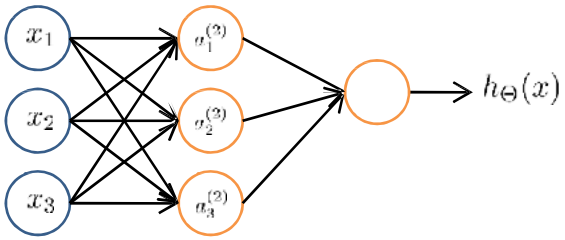
Sigmoid (logistic) activation function.

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

Neural Network



Neural Network



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

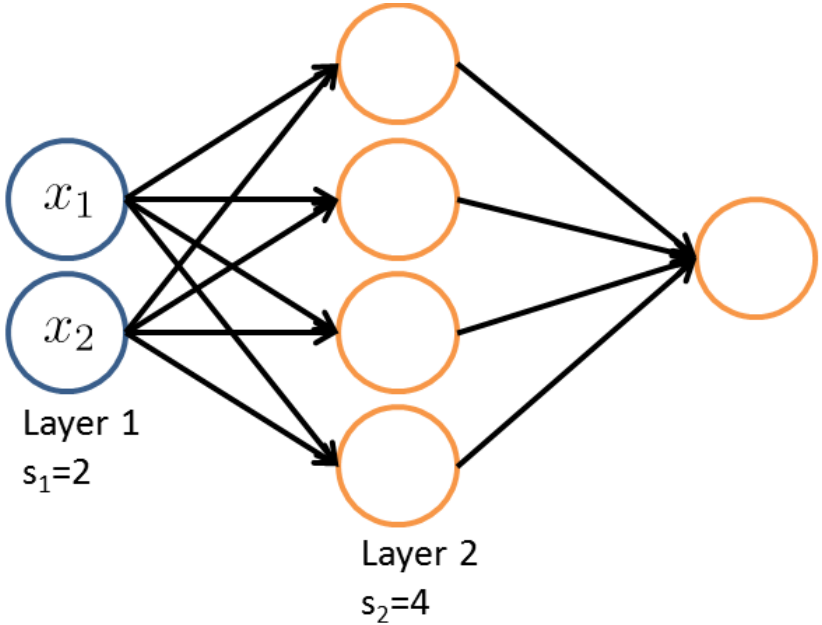
$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

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

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

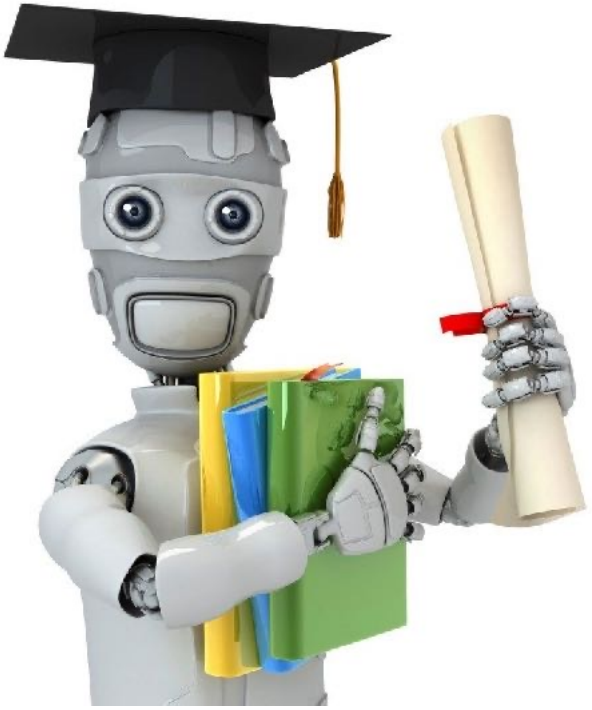
If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

Consider the following neural network,



What is the dimension of $\Theta^{(1)}$?

1. 2 x 4
2. 4 x 2
3. 3 x 4
4. 4 x 3

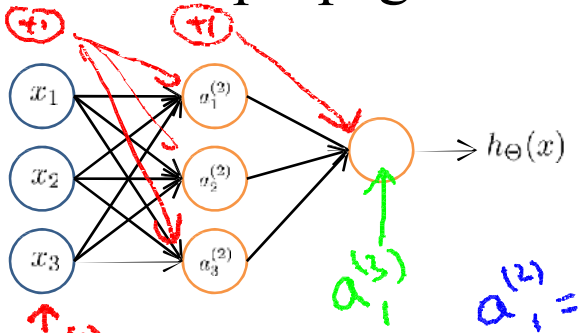


Machine Learning

Neural Networks: Representation

Model representation II

Forward propagation: Vectorized implementation



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

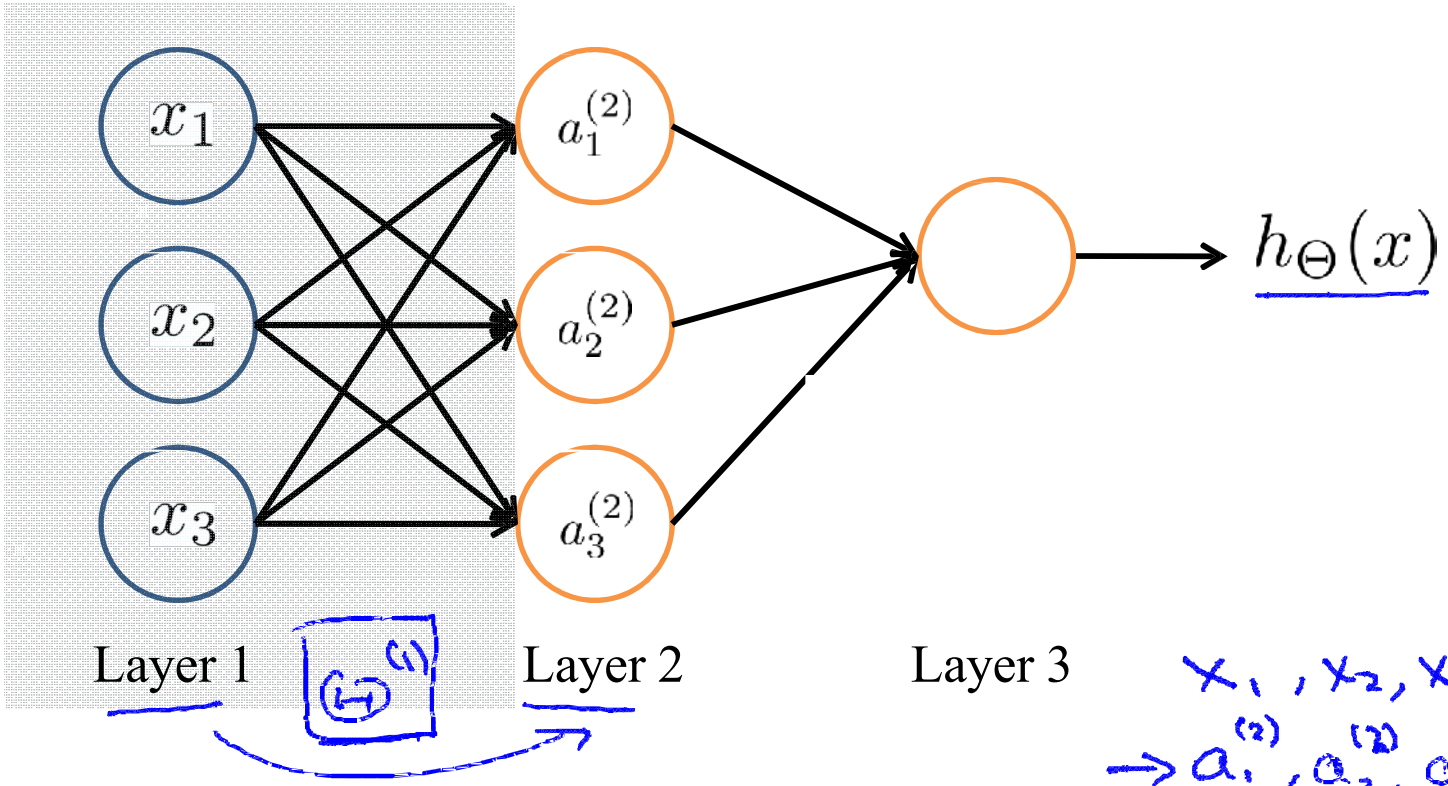
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

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

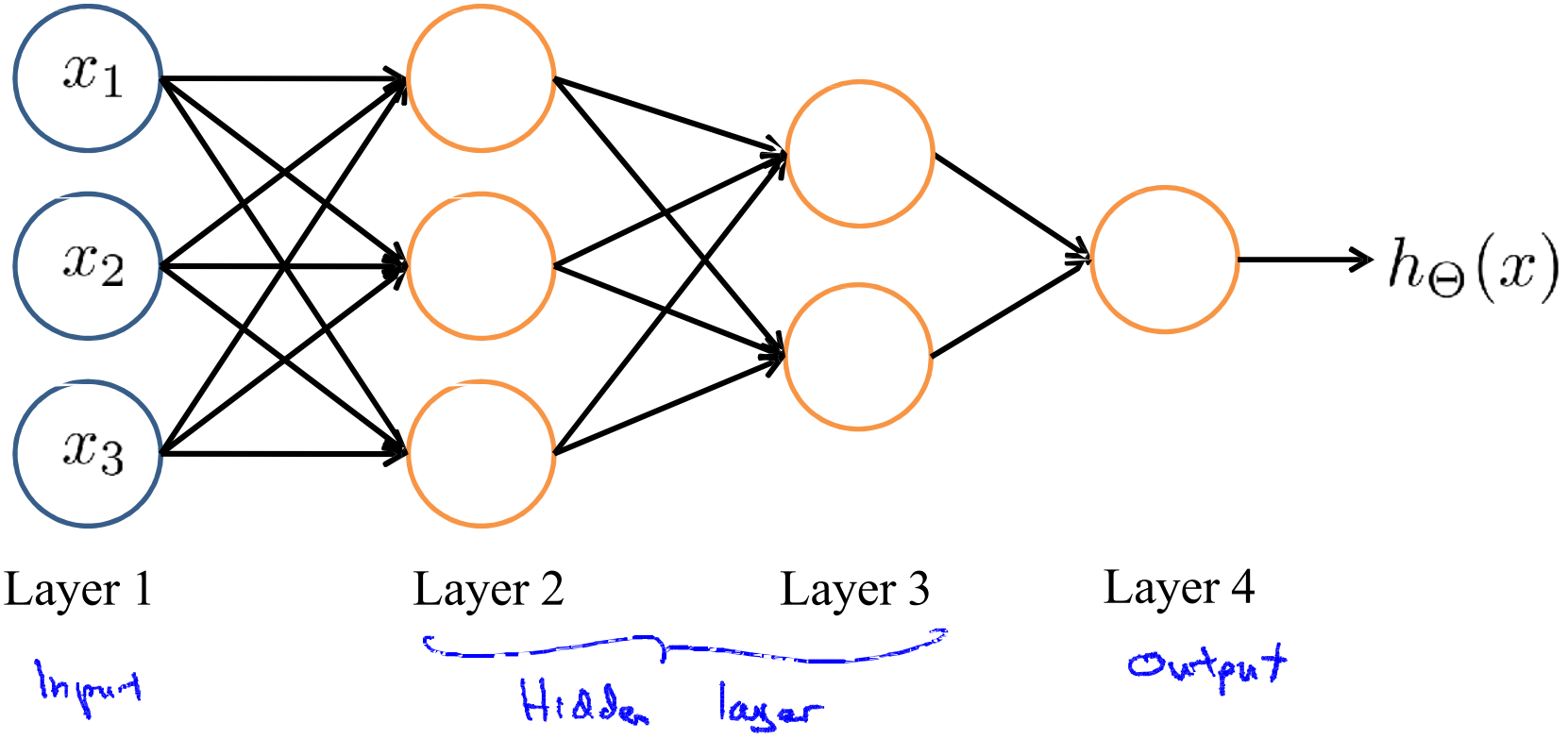
Add $a_0^{(2)} = 1.$ $\in \mathbb{R}^4$

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

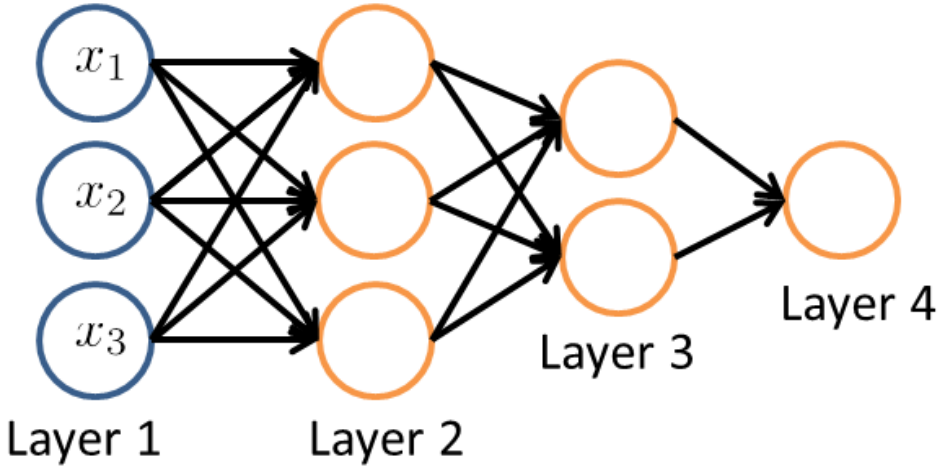
Neural Network learning its own features



Other network architectures



Consider the network



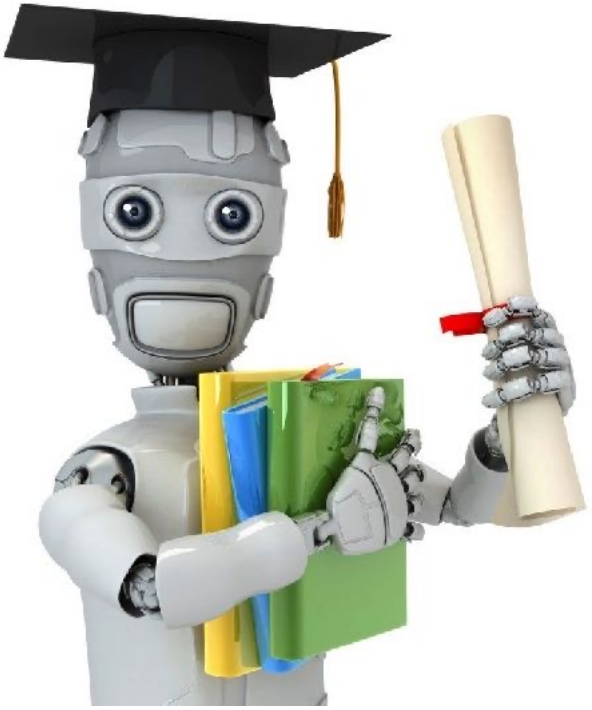
Let $a^{(1)} = x \in R^{n+1}$ denote the input with $(a_0^{(1)} = 1)$,. How you compute $a^{(2)}$?

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

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

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

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



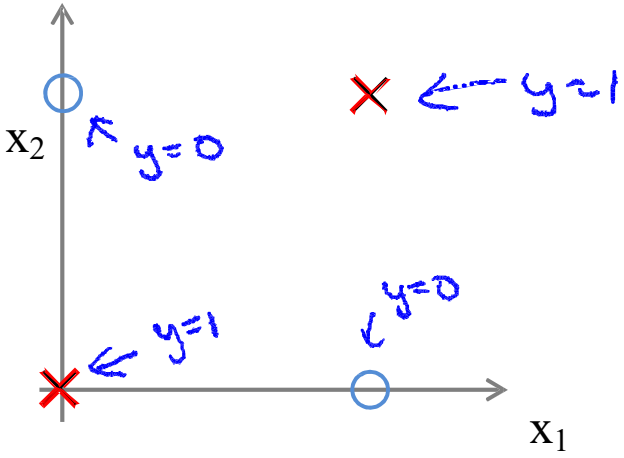
Machine Learning

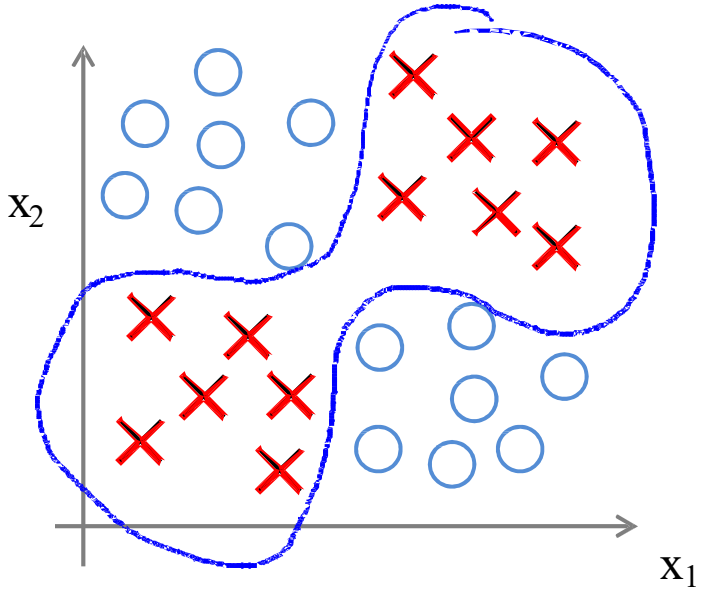
Neural Networks: Representation

Examples and intuitions I

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).





$y = x_1$ XOR x_2

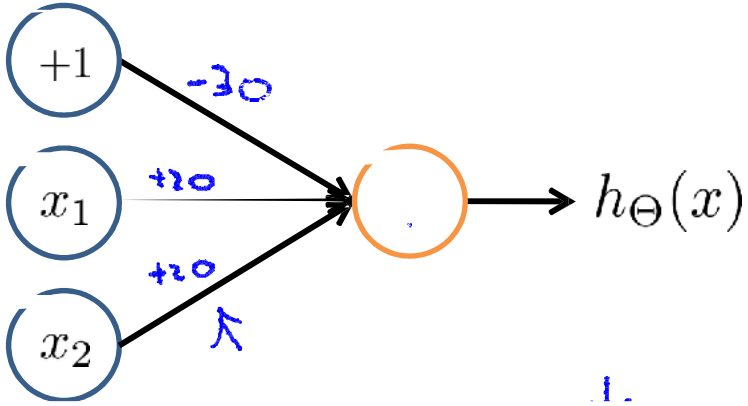
x_1 XNOR x_2

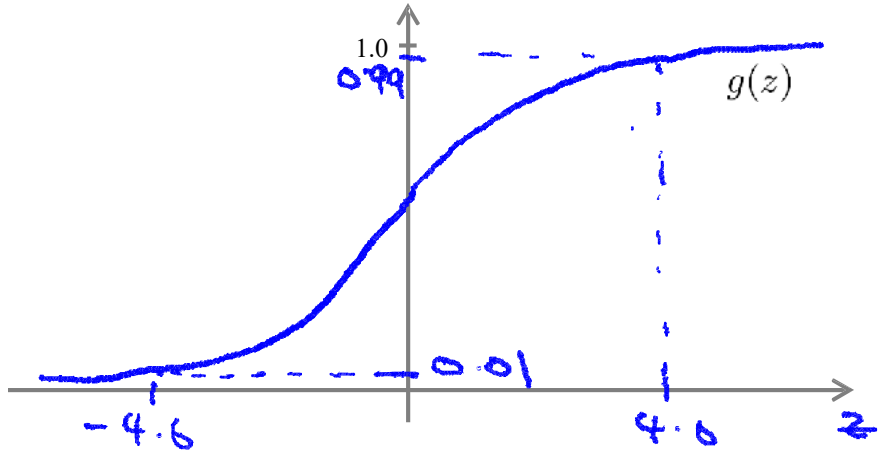
NOT (x_1 XOR x_2)

Simple example: AND

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

$y = x_1 \text{ AND } x_2$

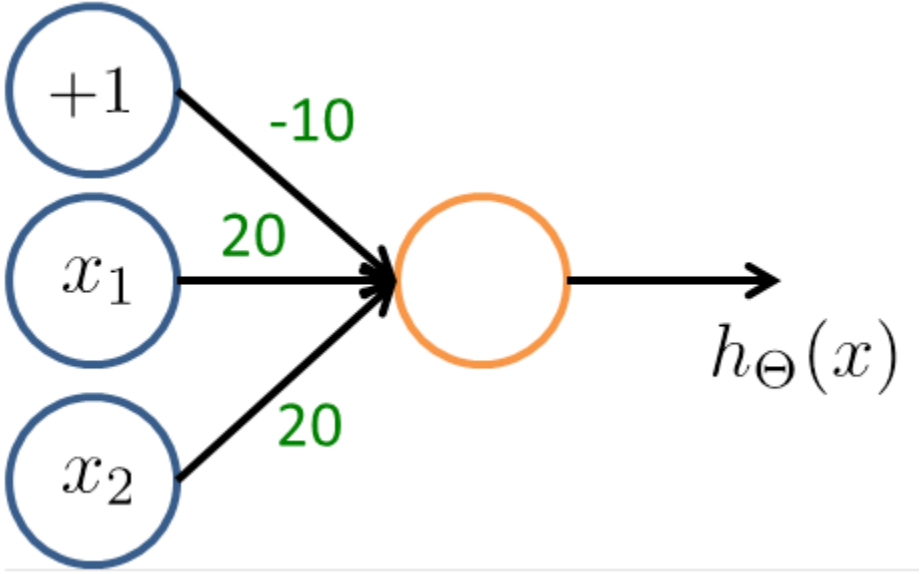




x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

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

Suppose x_1 , and x_2 are binary valued (0 or 1). What boolean function does the network show below(approximately) compute? (Hint: One possible way to answer this is to draw out a truth table)



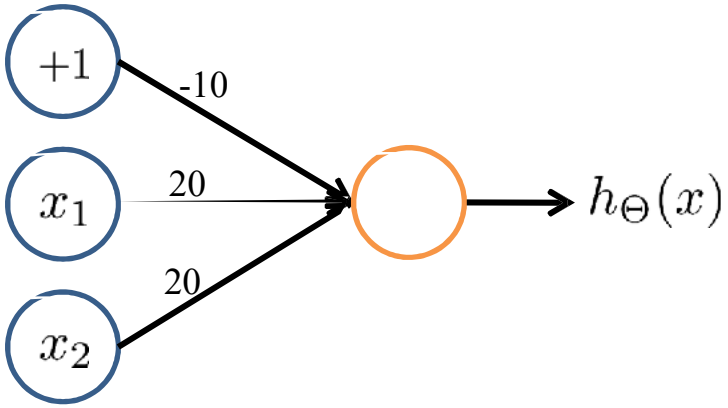
1 x_1 AND x_2

2 (NOT x_1) OR (NOT x_2)

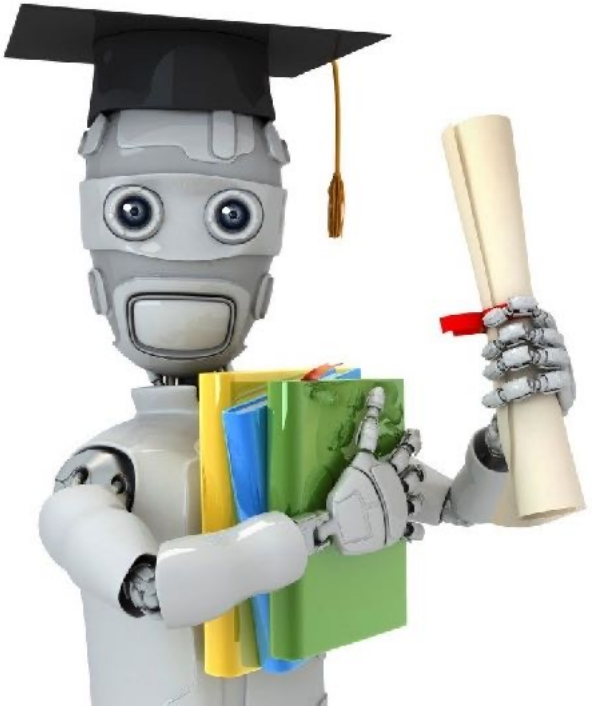
3 x_1 OR x_2

4 (NOT x_1) AND (NOT 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	≈ 1
1	1	≈ 1



Machine Learning

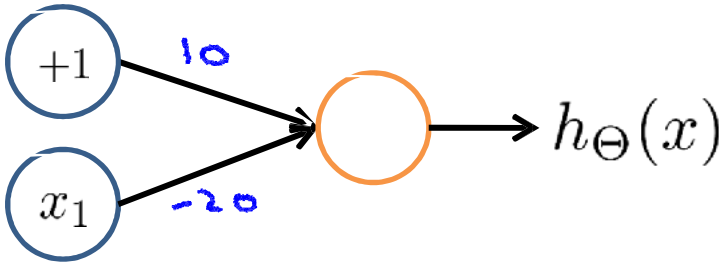
Neural Networks: Representation

Examples and intuitions II

x_1 AND x_2 x_1 OR x_2

Negation:

NOT x_1



x_1	$h_\Theta(x)$
0	
1	

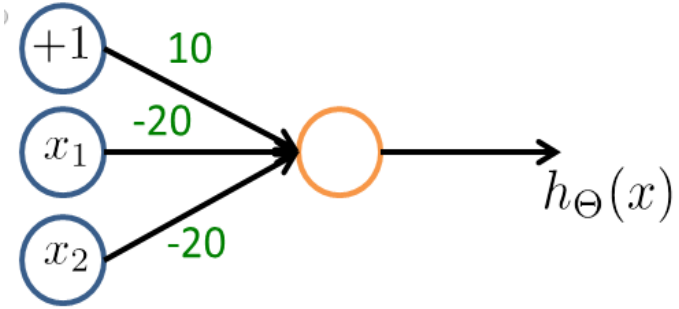
→ (NOT x_1) AND (NOT x_2)

(=1 if and only if

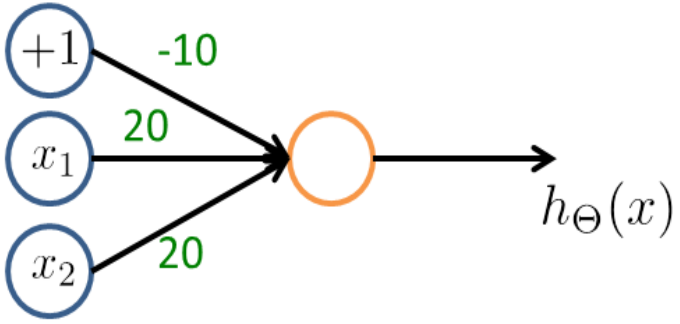
→ $x_1 = x_2 = 0$

Suppose that x_1 and x_2 are binary valued (0 and 1). Which of the following network (approximately) computes the boolean function (NOT x_1) AND (NOT x_2)?

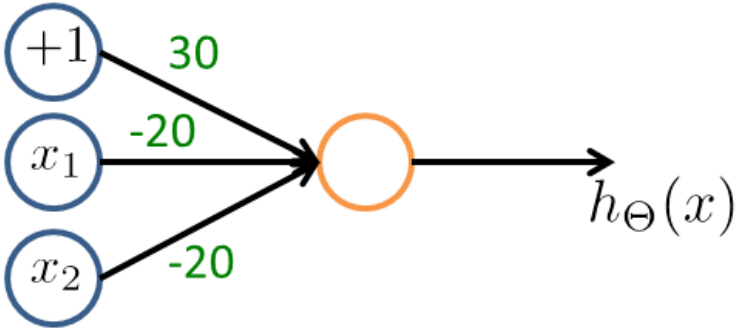
1



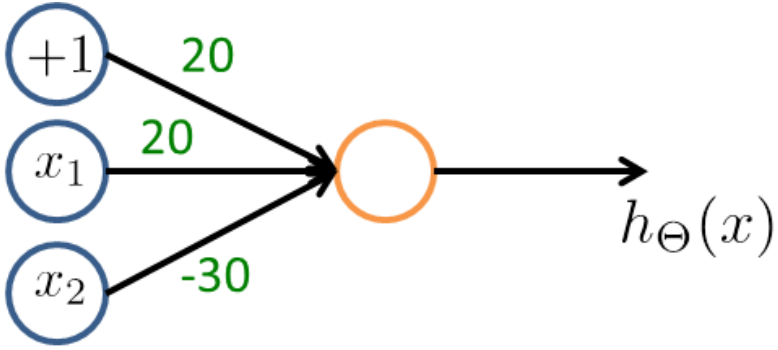
2



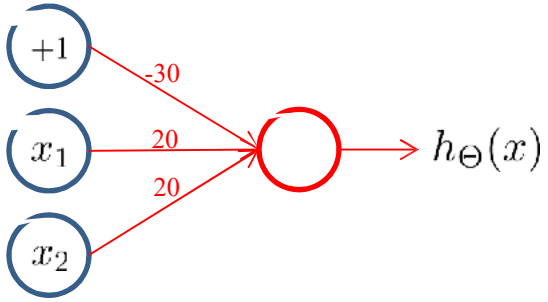
3



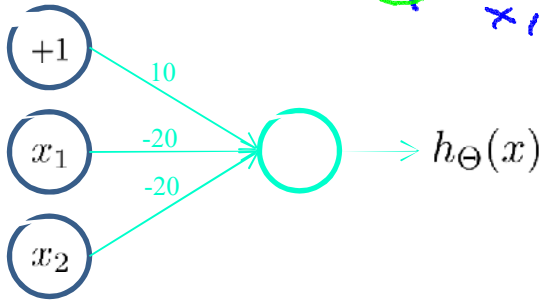
4



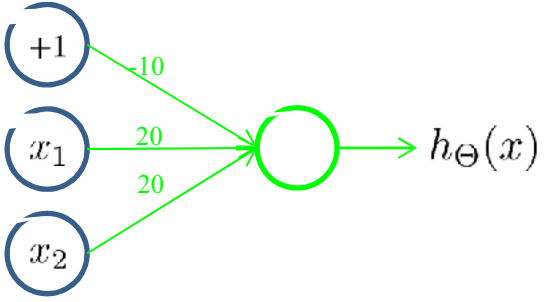
Putting it together: x_1 XNOR x_2



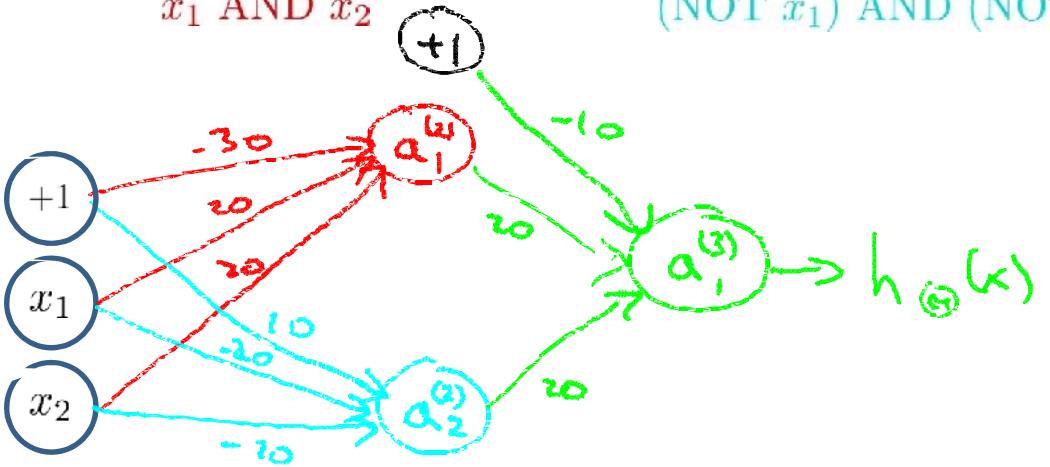
x_1 AND x_2



(NOT x_1) AND (NOT x_2)

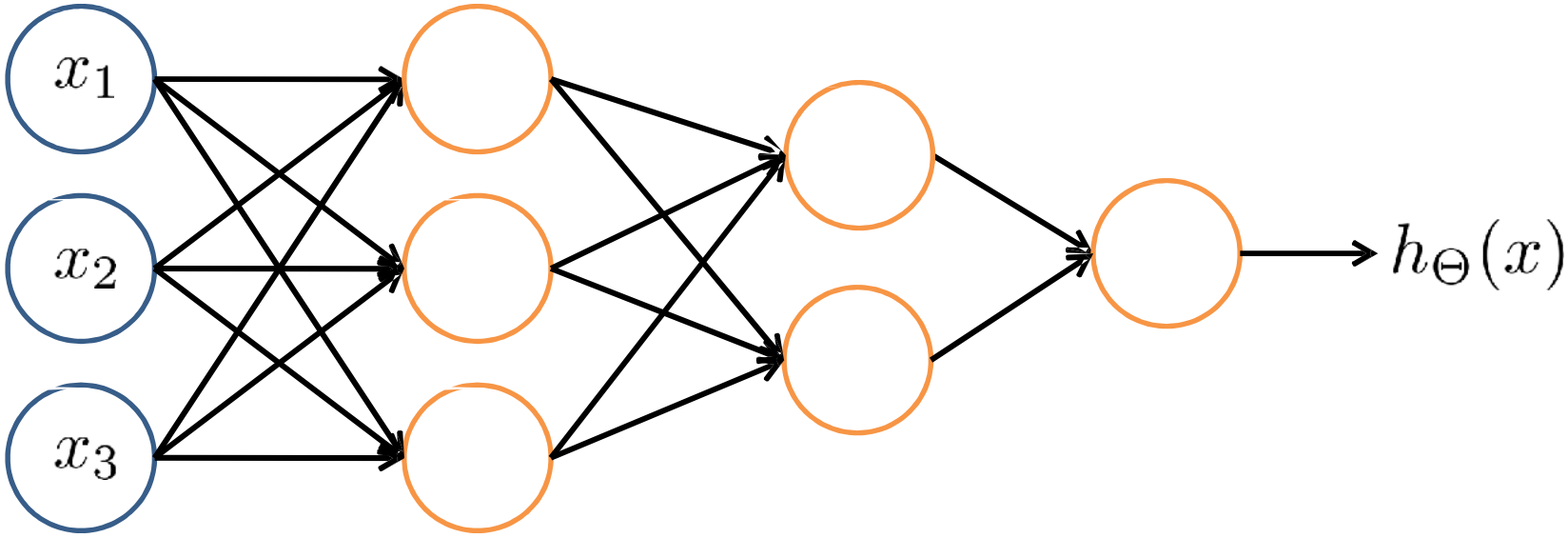


x_1 OR x_2

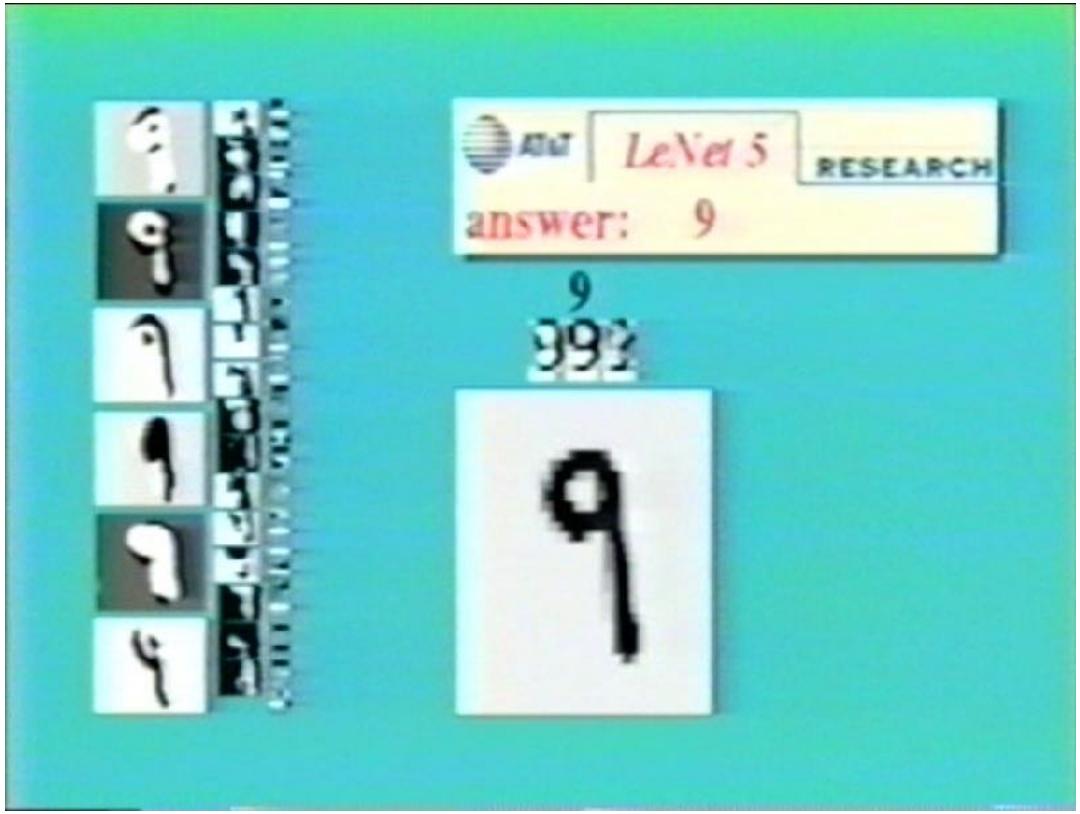


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Neural Network intuition

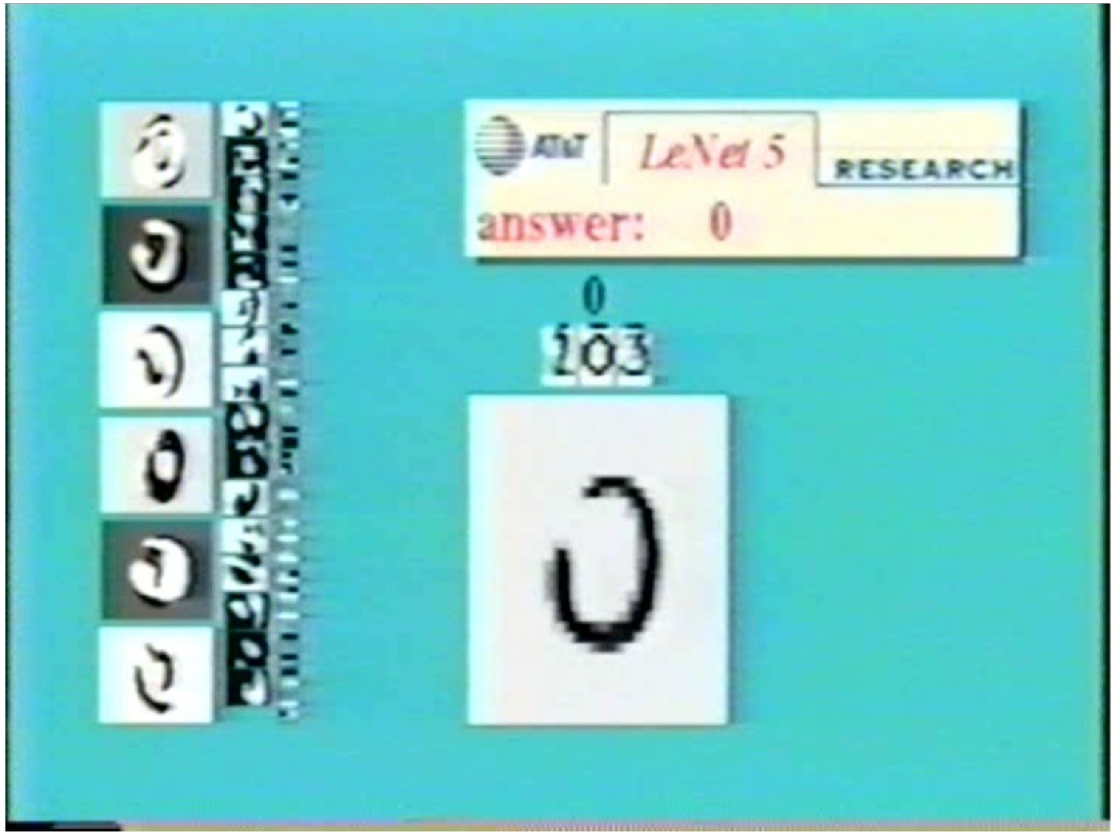


Handwritten digit classification

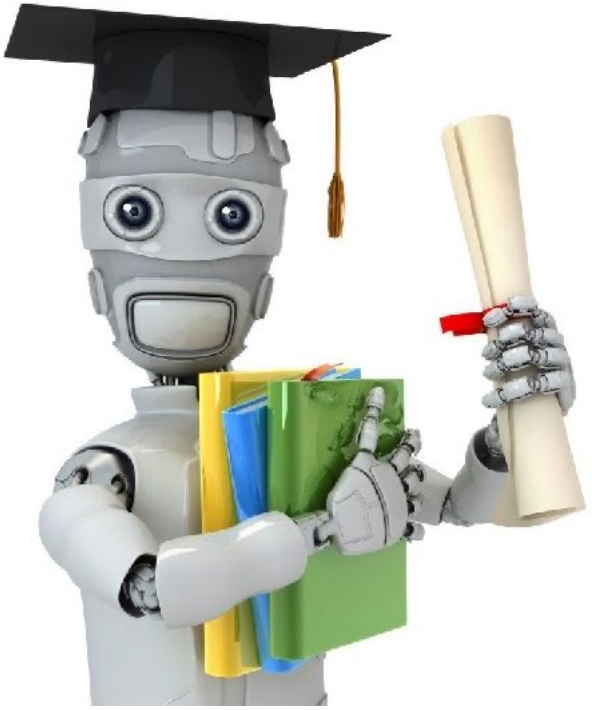


[Courtesy of Yann LeCun]

Handwritten digit classification



[Courtesy of Yann LeCun]



Machine Learning

Neural Networks: Representation

Multi-class classification

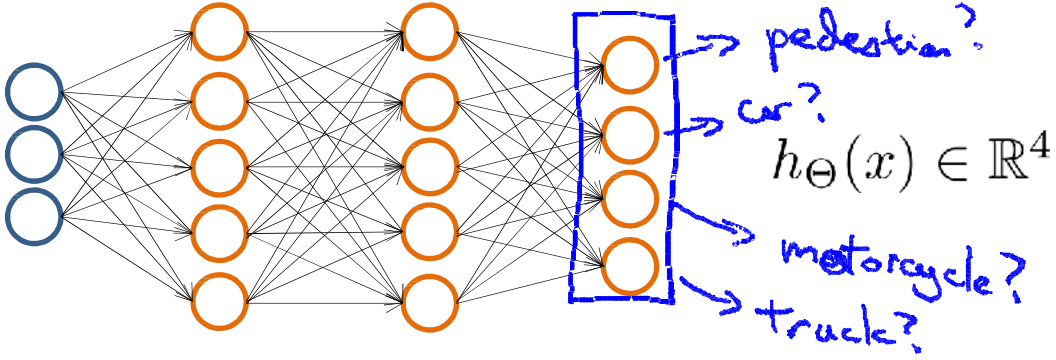
Multiple output units: One-vs-all.

Pedestrian

Car

Motorcycle

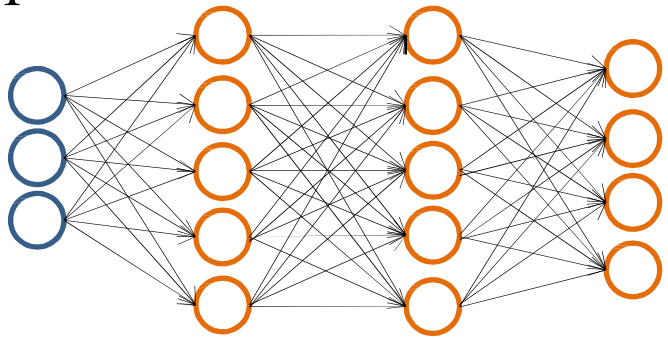
Truck



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.



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

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

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

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

$\in \mathbb{R}^4$

Suppose you have multi-class classification problem with 10 classes. Your neural network has 3 layers, and the hidden layer (layer 2) has 5 units. Using one-vs-all method described here how many element does $\Theta^{(2)}$ have?

1. 50
2. 55
3. 60
4. 66