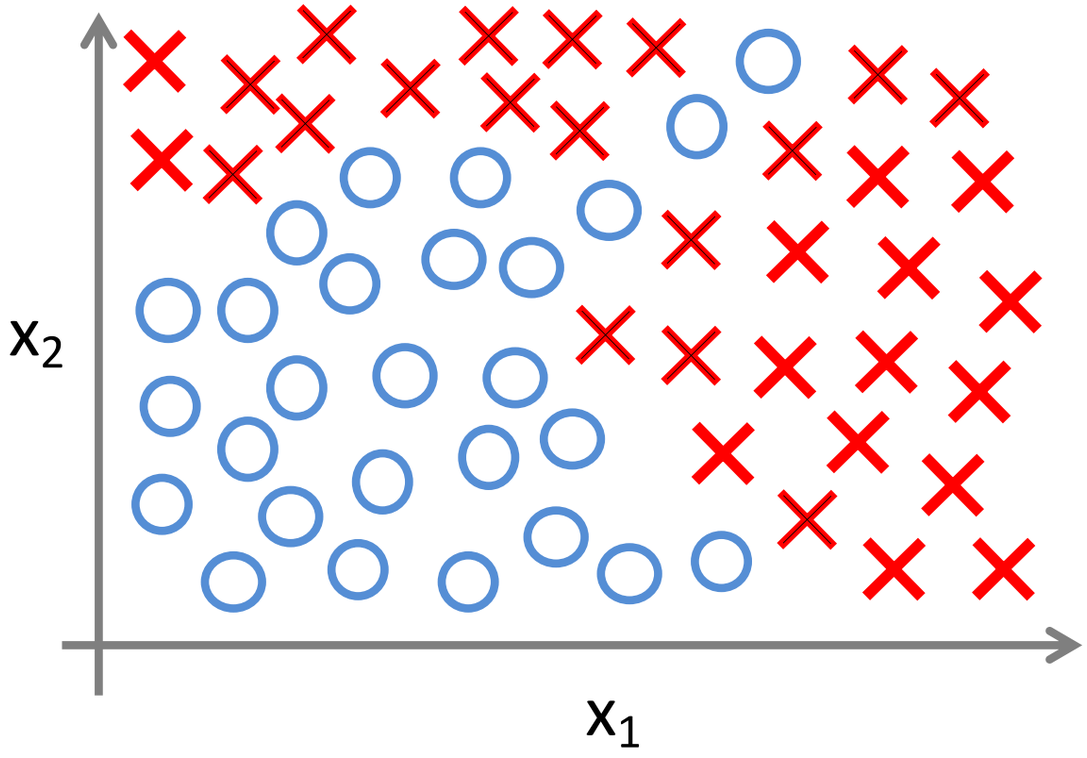
Neural networks representation

Andrew Ng

Neural networks:
representation
Non-linear hypothesis

Non-linear classification



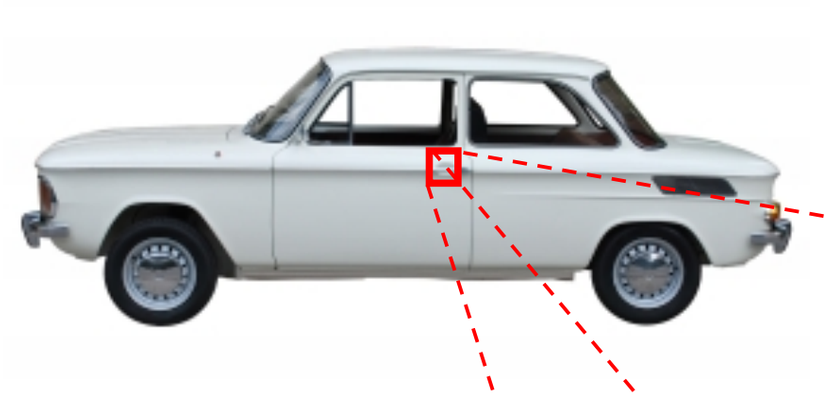
$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \ldots)$$

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

$O(n^2)$ quadratic features: x_1^2, x_1x_2, x_1x_3, …
$O(n^3)$ cubic features: 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






Cars

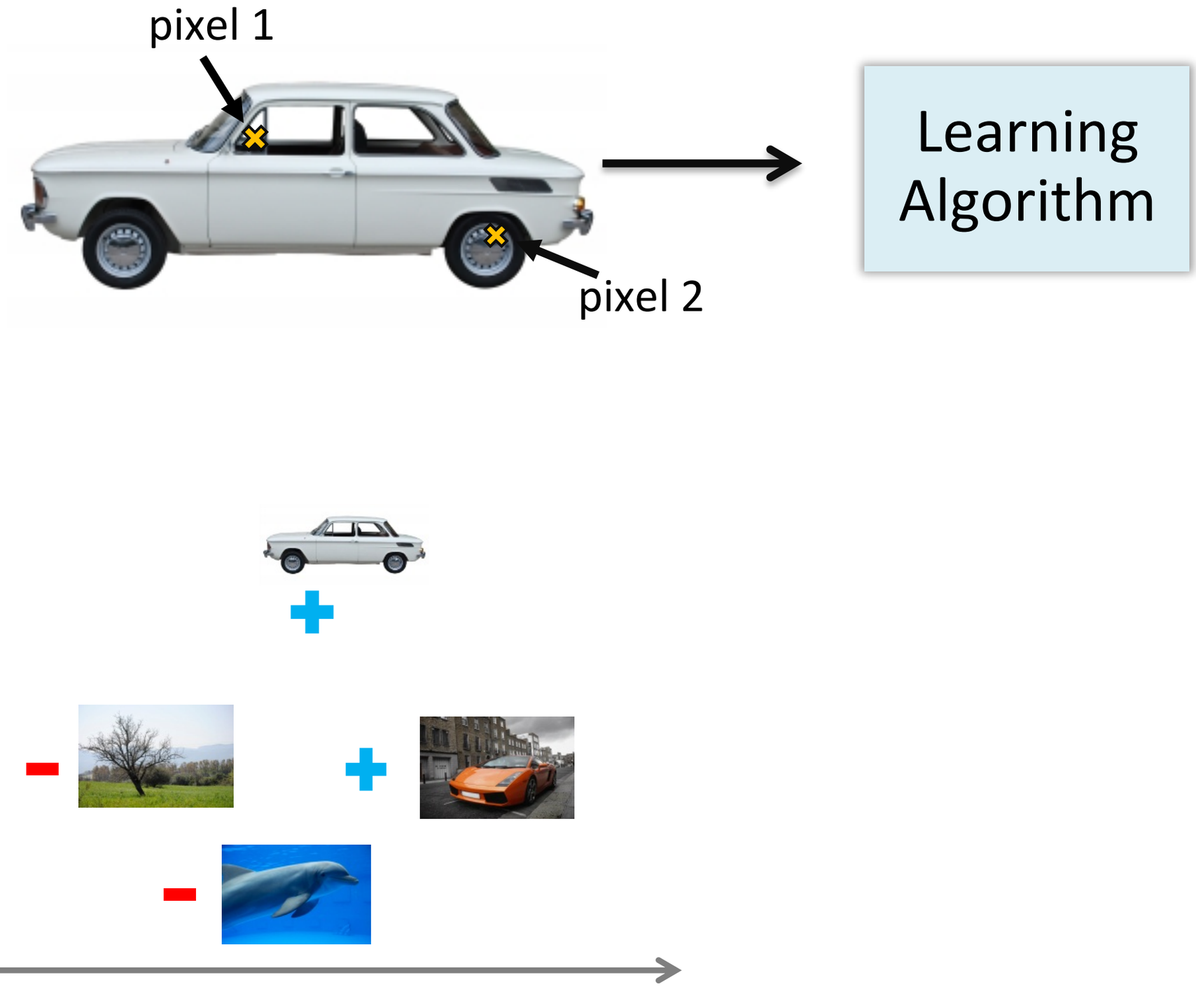


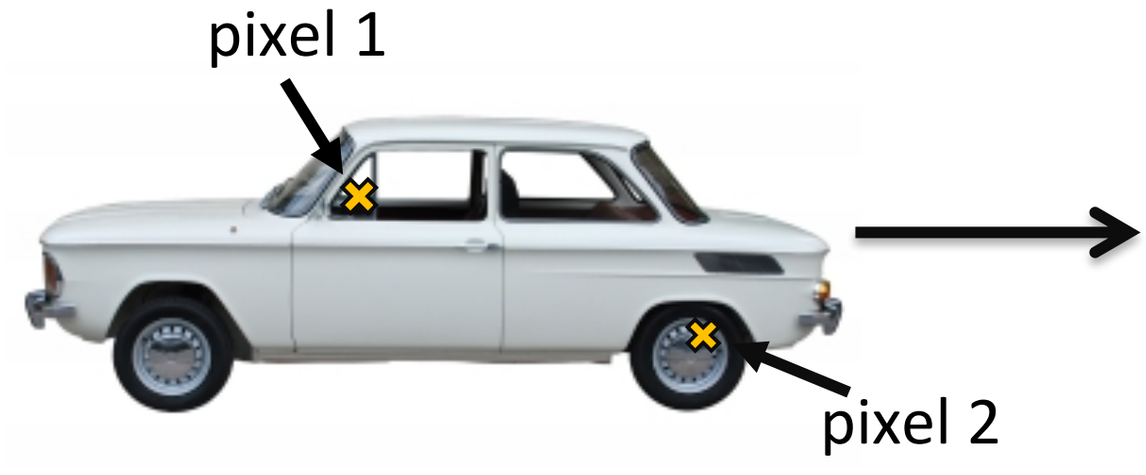



Not a car

Testing:

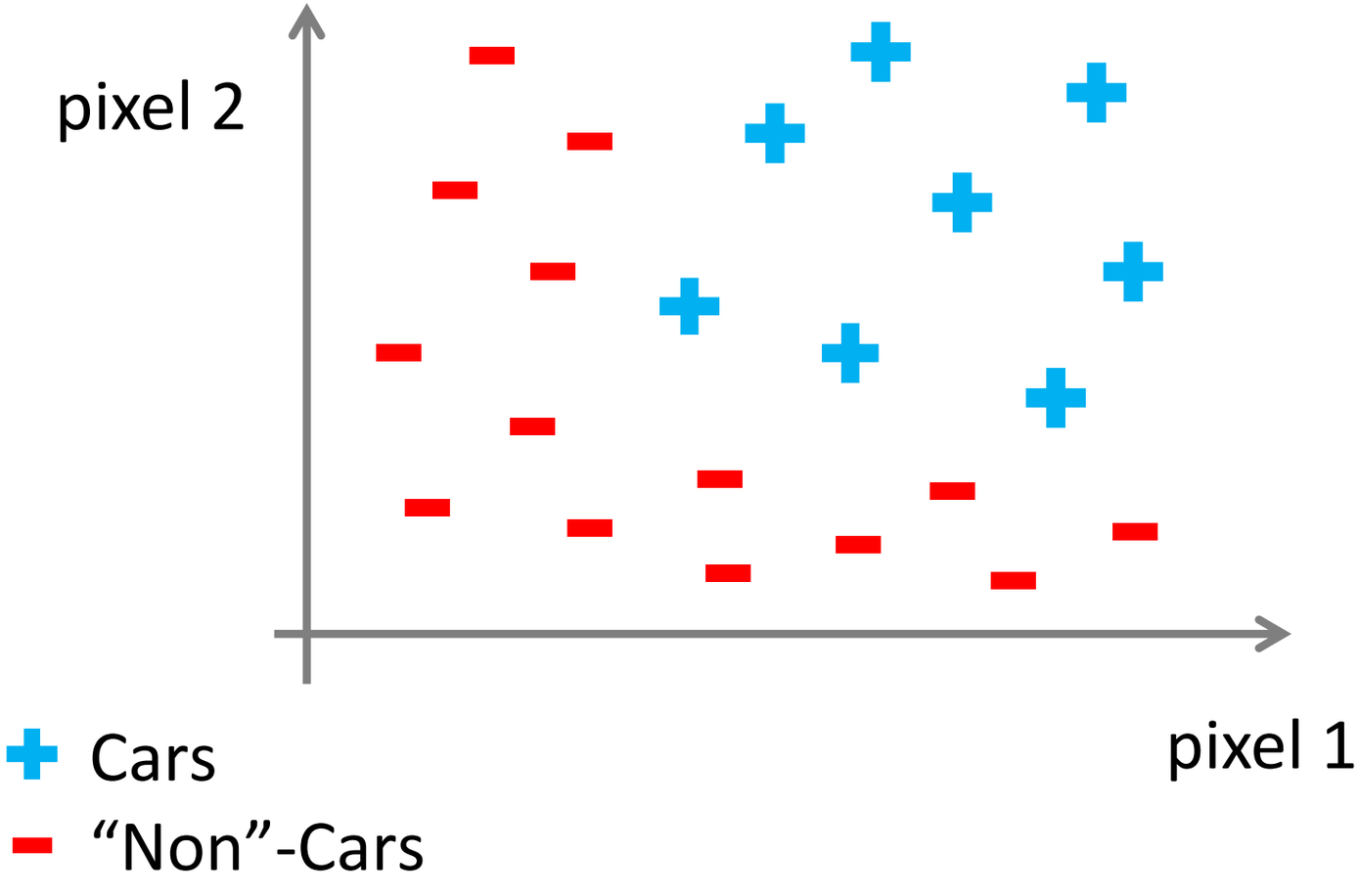
What is this?


pixel 1
pixel 2
Learning Algorithm
pixel 2
pixel 1
Cars
"Non"-Cars



Learning Algorithm

50 x 50 pixel images→ 2500 pixels
(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

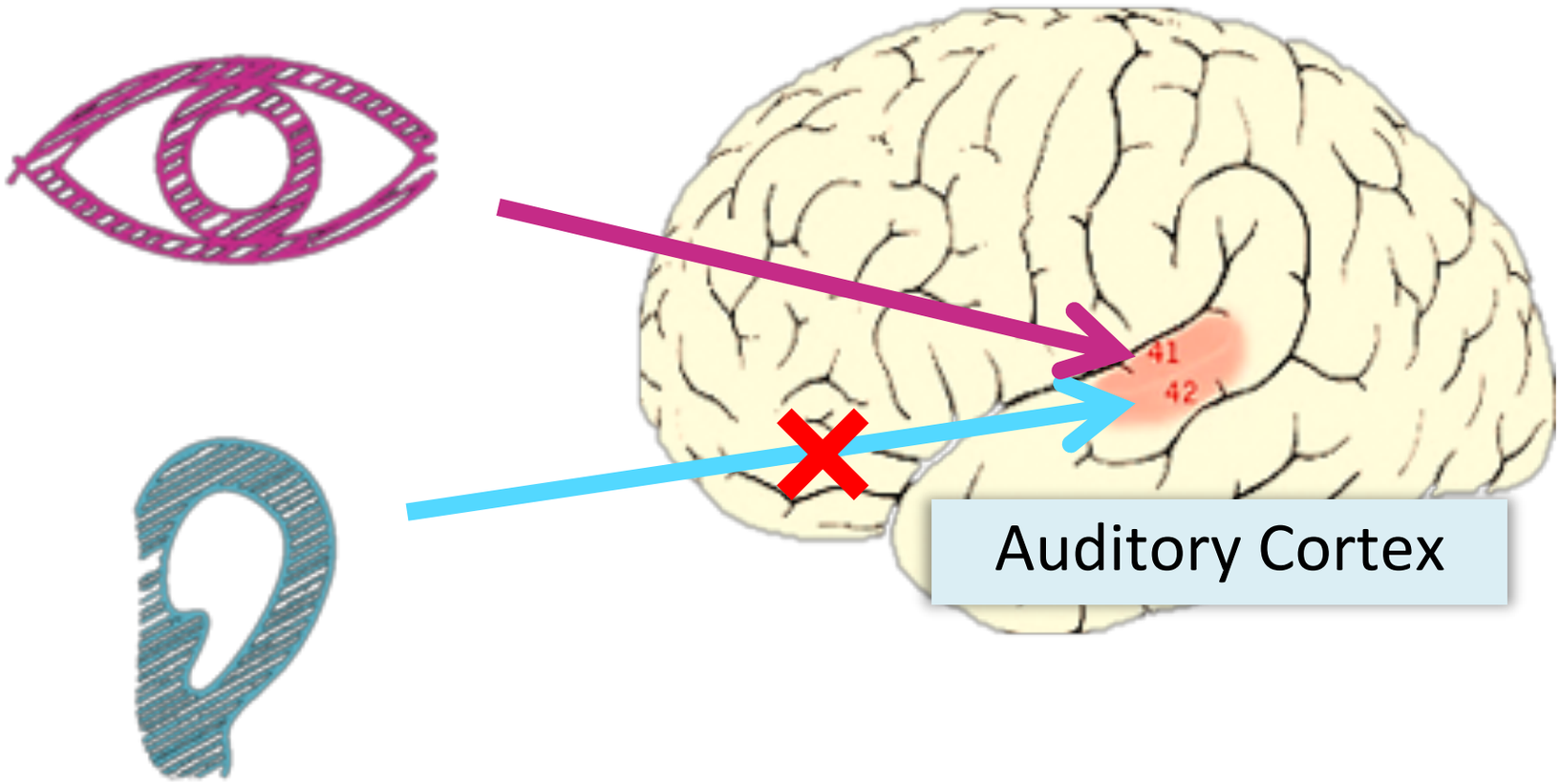
Neural networks:
representation
Neurons and the brain

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


Hebb's rule:
neurons that fire together wire together

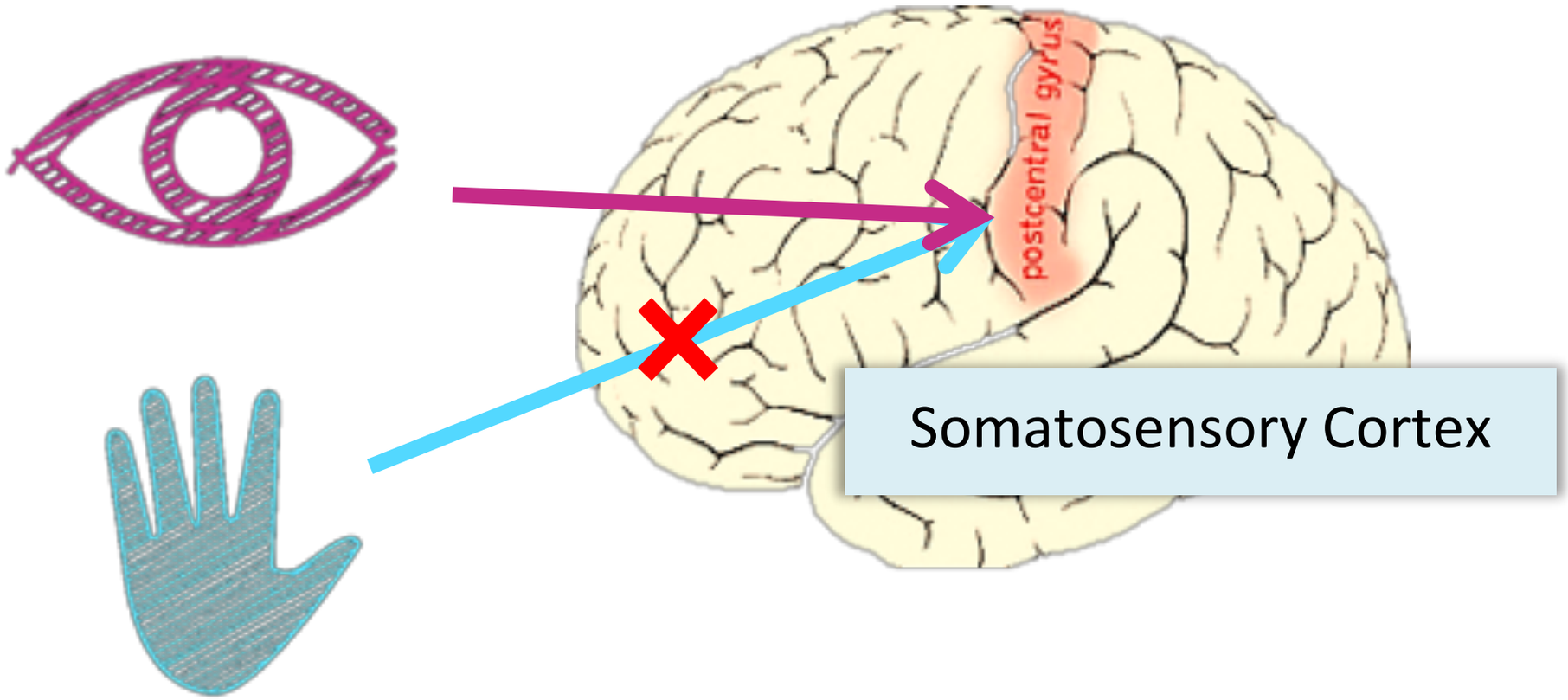
The “one learning algorithm” hypothesis



Auditory cortex learns to see

[Roe et al., 1992]

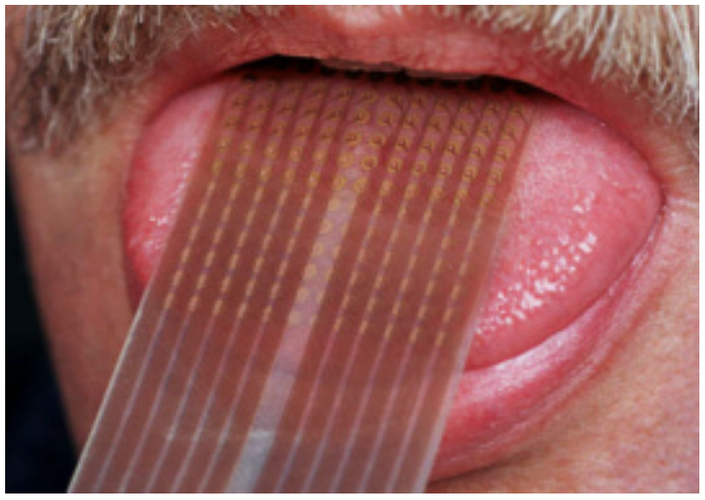
The “one learning algorithm” hypothesis



Somatosensory cortex learns to see

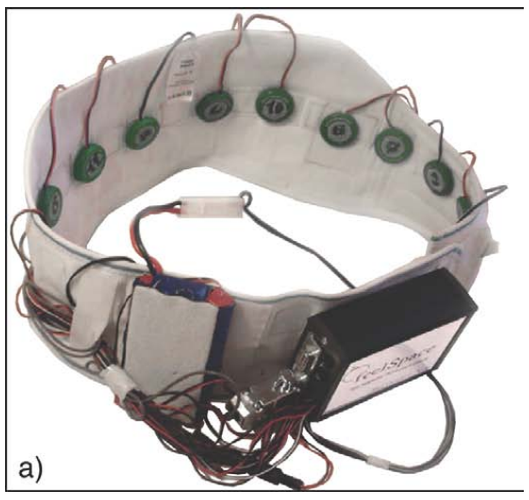
[Metin & Frost, 1989]

Sensor representations in the brain

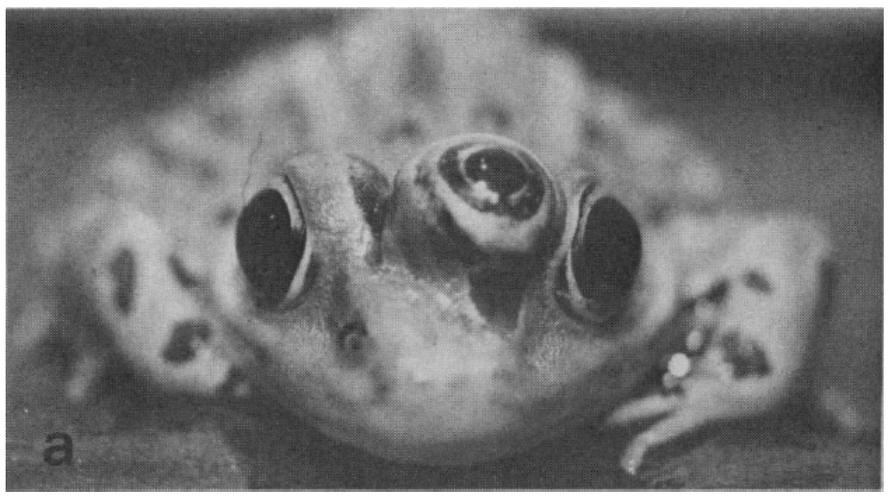


Seeing with your tongue

Human echolocation (sonar)



Haptic belt: Direction sense



Implanting a 3rd eye

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

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

THE BLUE BRAIN PROJECT - A SWISS BRAIN INITIATIVE

In brief

The goal of the Blue Brain Project is to build biologically detailed digital reconstructions and simulations of the rodent, and ultimately the human brain.

The supercomputer-based reconstructions and simulations built by the project offer a radically new approach for understanding the multilevel structure and function of the brain.

The project's novel research strategy exploits interdependencies in the experimental data to obtain dense maps of the brain, without measuring every detail of its multiple levels of organization (molecules, cells, micro-circuits, brain regions, the whole brain).

This strategy allows the project to build digital reconstructions (computer models) of the brain at an unprecedented level of biological detail.

Supercomputer-based simulation of their behavior turns understanding the brain into a tractable problem, providing a new tool to study the complex interactions within different levels of brain organization and to investigate the cross-level links leading from genes to cognition.

THE WHITE HOUSE IS ANNOUNCING

OVER $300 MILLION IN PUBLIC AND PRIVATE INVESTMENTS IN SUPPORT OF THE BRAIN INITIATIVE

---

the WHITE HOUSE

BRAIN INITIATIVE

BRAIN RESEARCH THROUGH ADVANCING INNOVATIVE NEUROTECHNOLOGIES

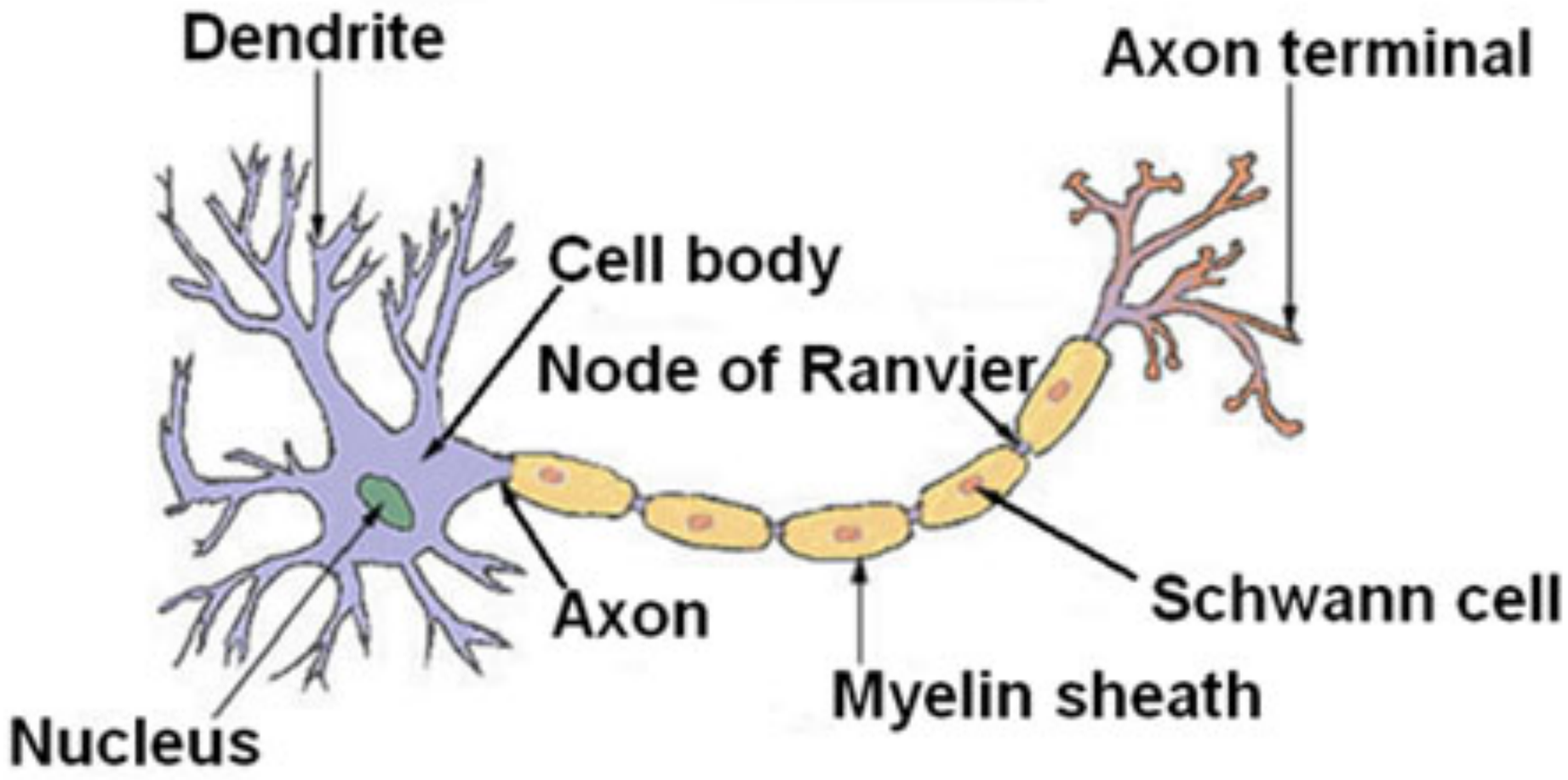
Since President Obama announced the **BRAIN Initiative** in April 2013, dozens of leading technology firms, academic institutions, scientists and other key contributors to the field of neuroscience have answered his call and made significant commitments to advancing the Initiative.

Neural networks:

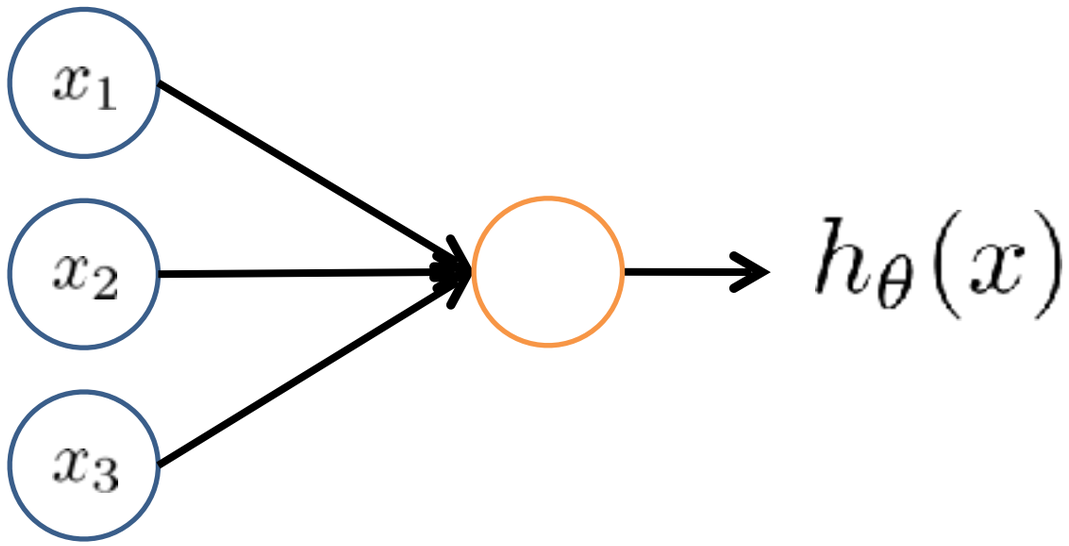
representation

Model representation

Neuron in the brain



Neuron model: Logistic unit



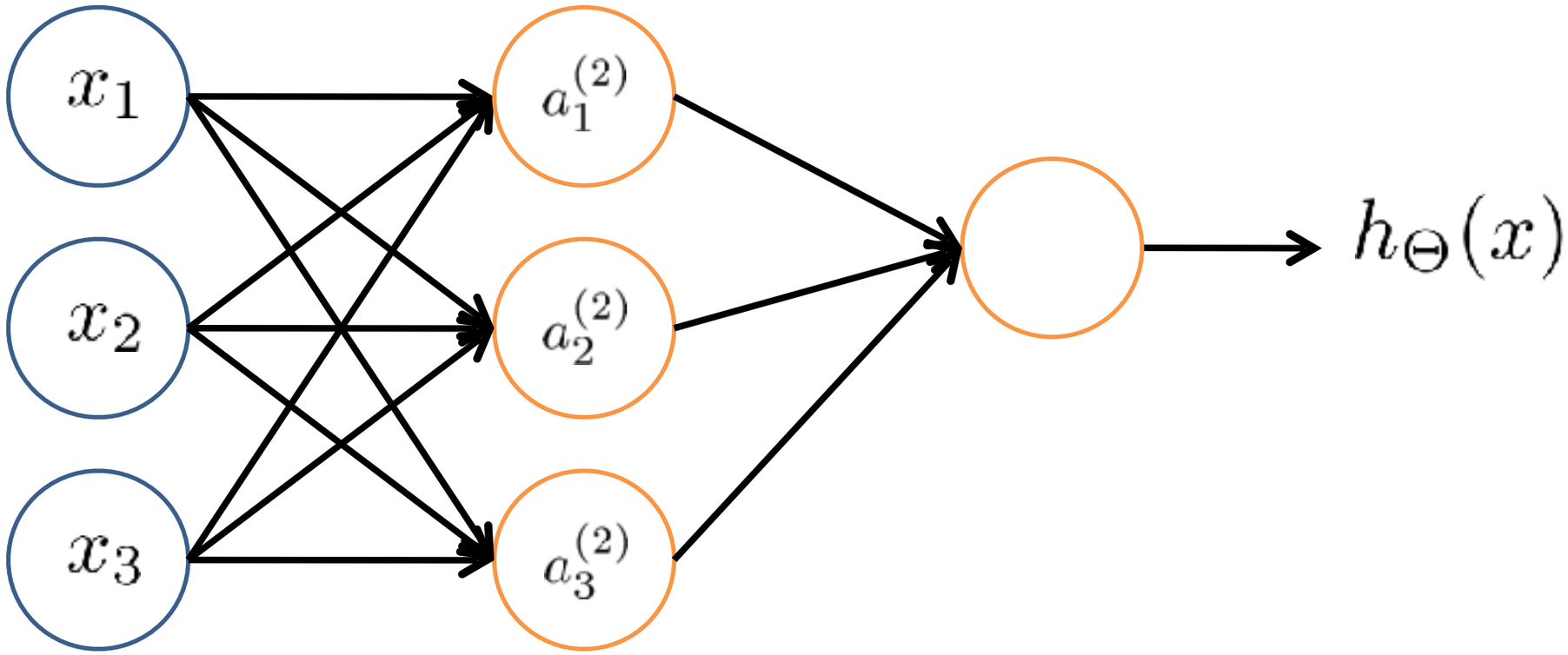
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

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

Sigmoid (logistic) activation function

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

Neural network

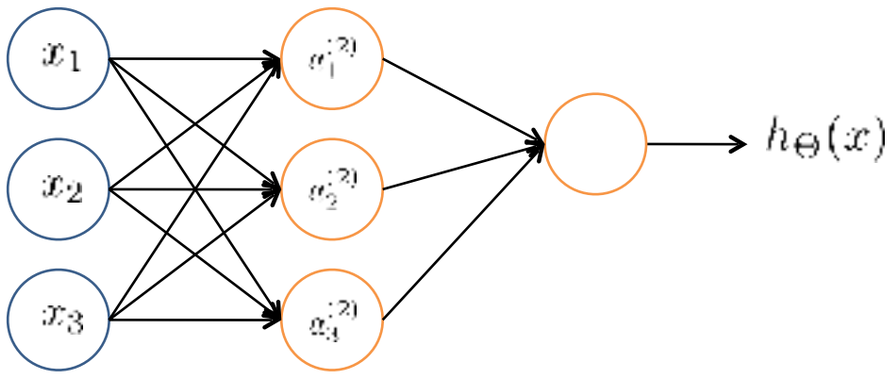


Layer 1 input layer

Layer 2 hidden layer

Layer 3 output layer

Neural network



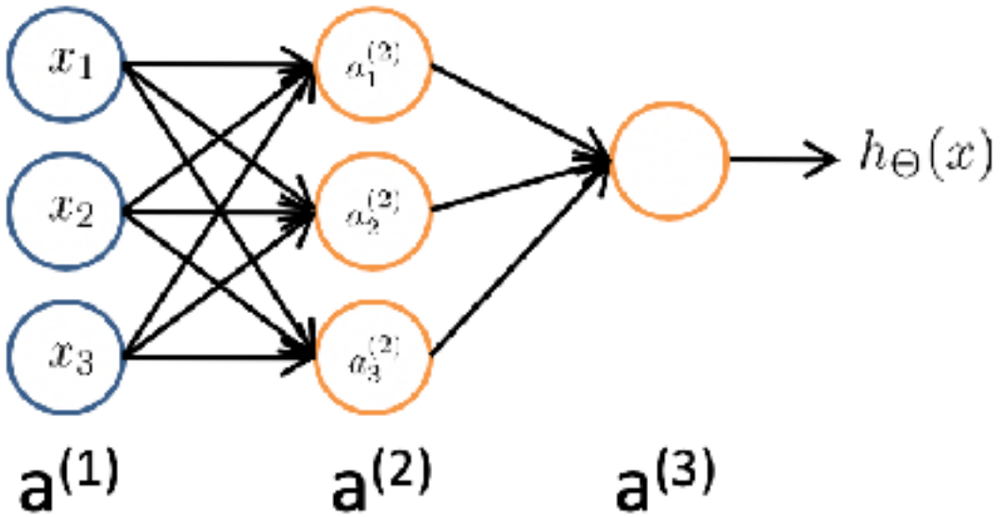
$a_i^{(j)}$ = "activation" of unit i in layer j

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

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

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

Forward propagation: Vectorized implementation



$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$a_1^{(2)} = g(z_1^{(2)}); a_2^{(2)} = g(z_2^{(2)}); a_3^{(2)} = g(z_3^{(2)})$$
$$h_\Theta(x) = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

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

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

Add $a_0^{(2)} = 1$

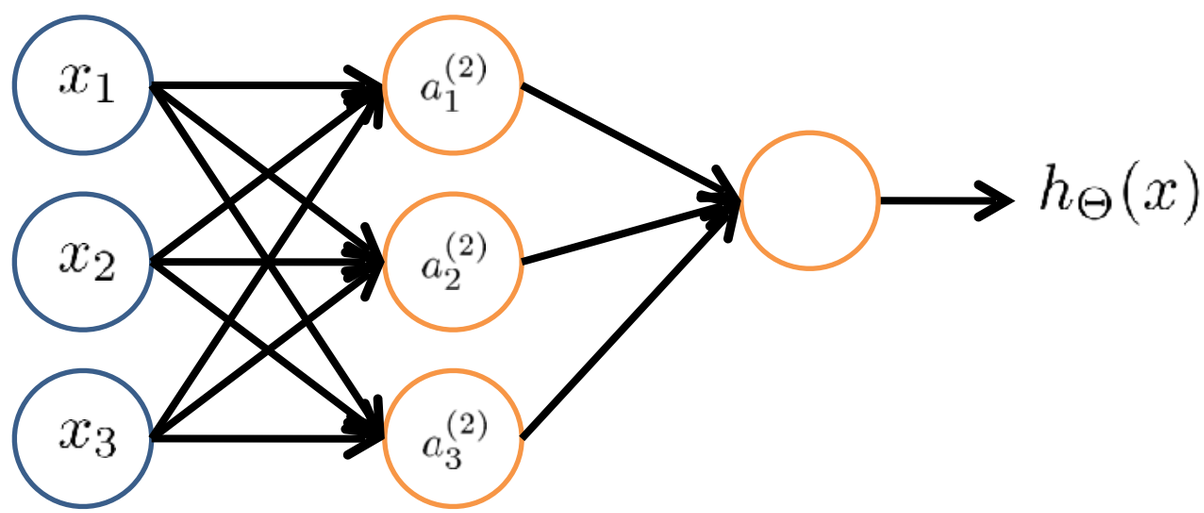
$$z^{(3)} = \Theta^{(2)}a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

Neural Network learning its own features

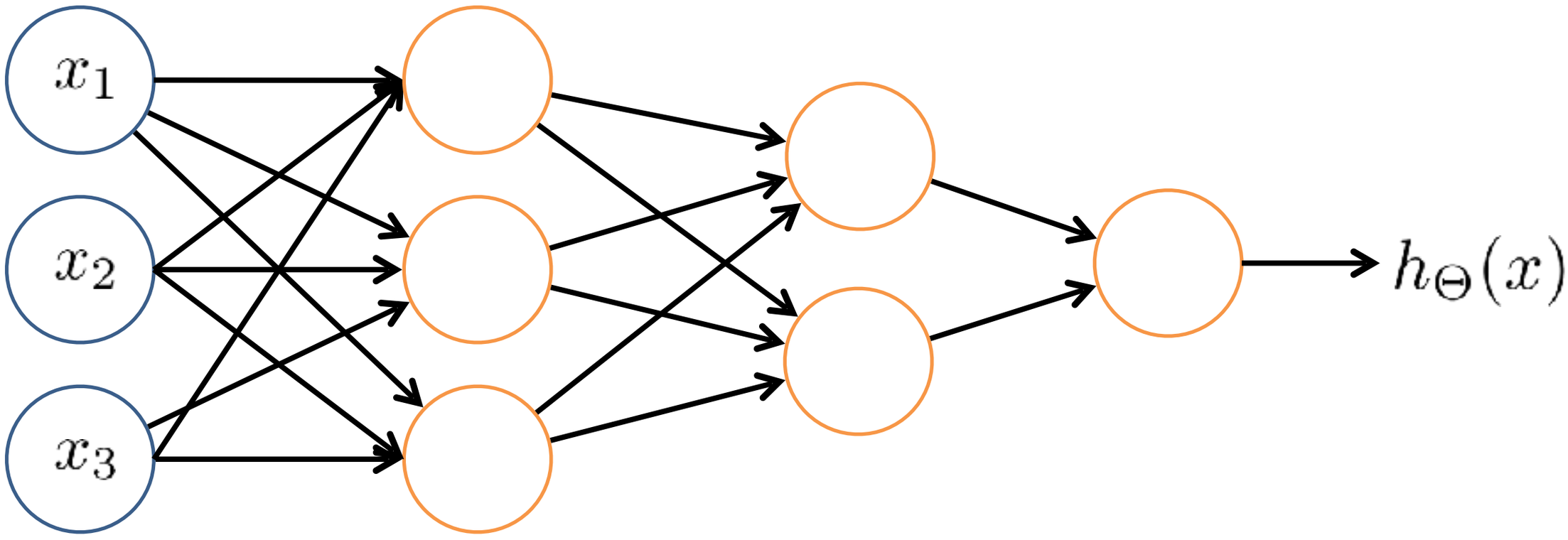
50 x 50 pixel images→ 2500 pixels
(7500 if RGB)

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

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



Other network architectures

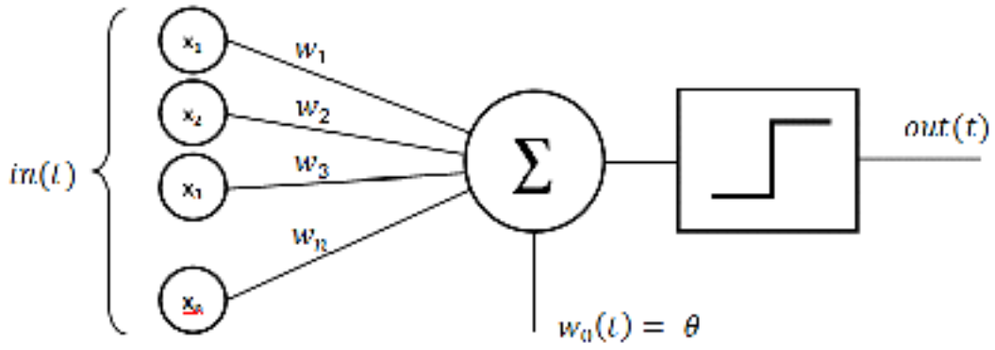


Neural networks:
representation
Examples and intuitions

A bit of history

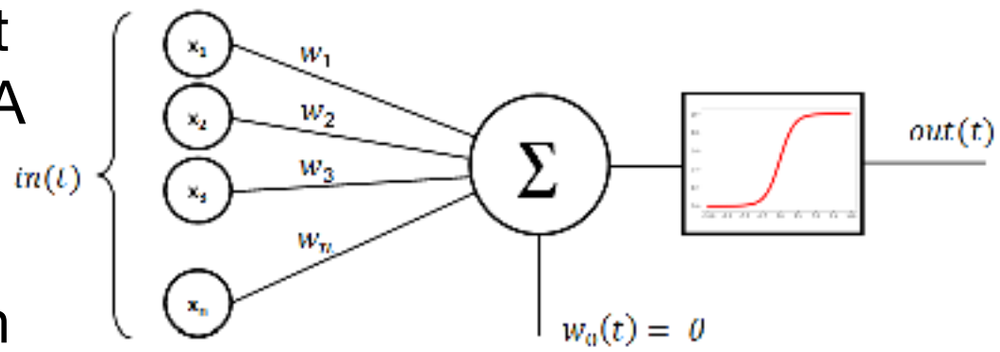
- The first formal model of a neuron was proposed by Warren McCulloch and Walter Pitts in 1943: AND and OR gates (the weights cannot be learned from data)

- Perceptrons were invented in the late 1950s by Frank Rosenblatt: if the positive and negative examples could be separated by a hyperplane, the perceptron would find it (weights can be learned from data for linearly separable classes)



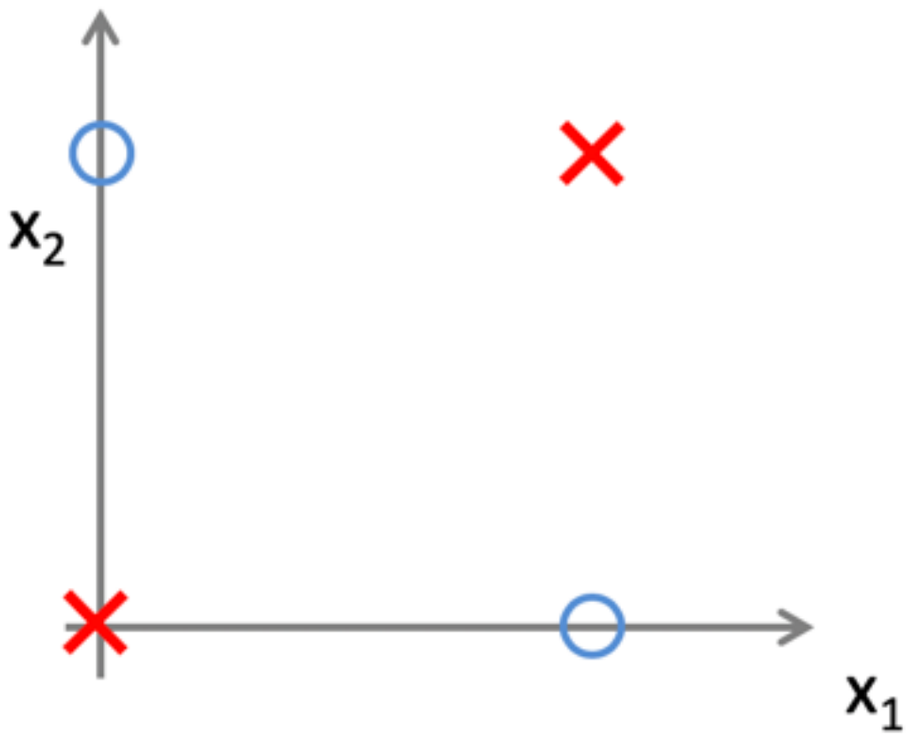
- In 1969, Minsky and his colleague Seymour Papert published *Perceptrons*, a book detailing the shortcomings of the perceptron algorithm, with example after example of simple things it couldn't learn. The simplest one was the exclusive-OR function (non-linearly separable classes require multi-layer networks)

- Backprop was invented in 1986 by David Rumelhart with the help of Geoff Hinton and Ronald Williams. A key aspect of the algorithm is the use of sigmoid instead of step function as neuron activation (weights in multi-layer networks can be learned with gradient descent)



Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).



depending on what we consider 0 or 1 (XOR vs. XNOR)

$$y = x_1 \text{ XOR } x_2$$
$$x_1 \text{ XNOR } x_2$$
$$\text{NOT } (x_1 \text{ XOR } x_2)$$

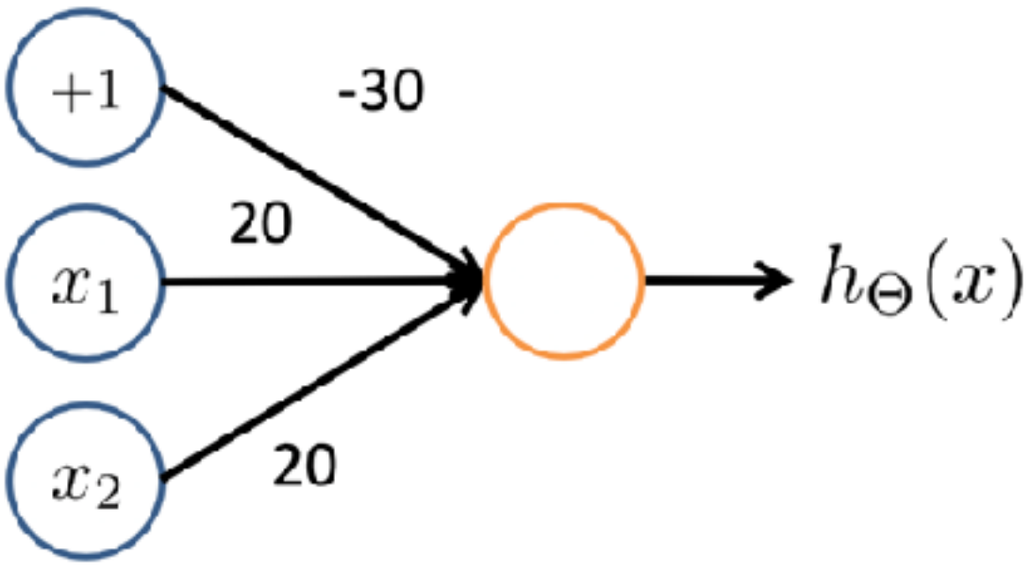
Any Boolean function can be put into the canonical disjunctive normal form:

x_1 XNOR x_2 =
(x_1 AND x_2) OR
((NOT x_1) AND (NOT x_2))

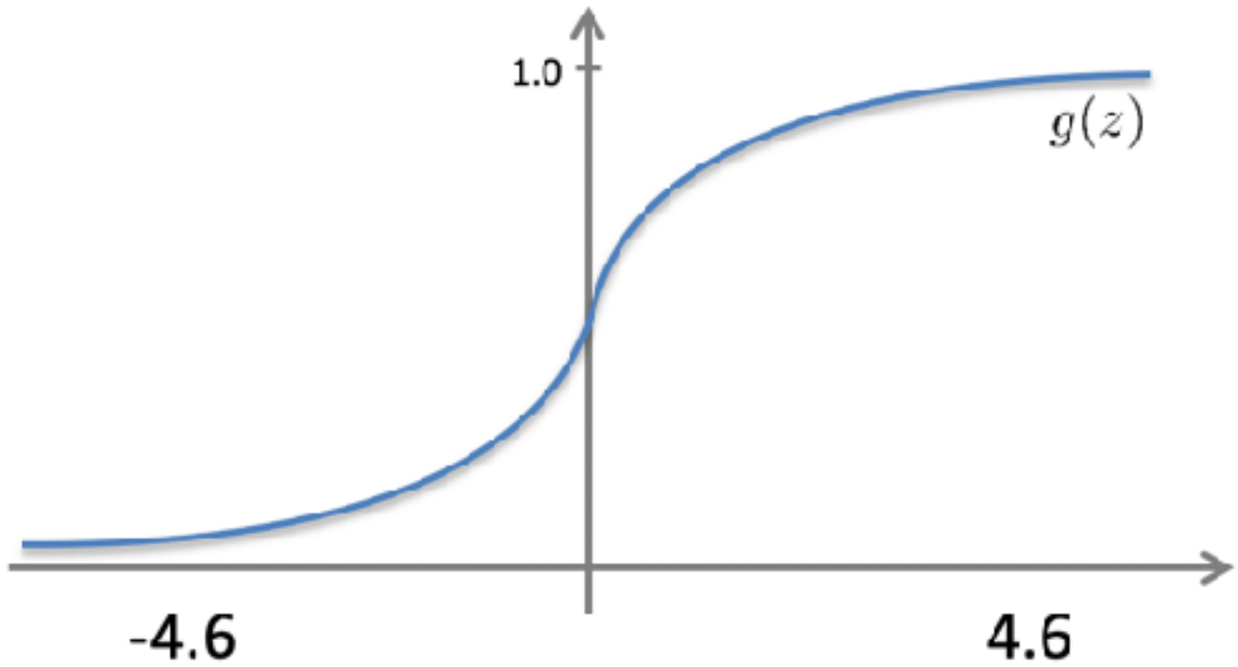
Simple example: AND

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

$y = x_1 \text{ AND } x_2$

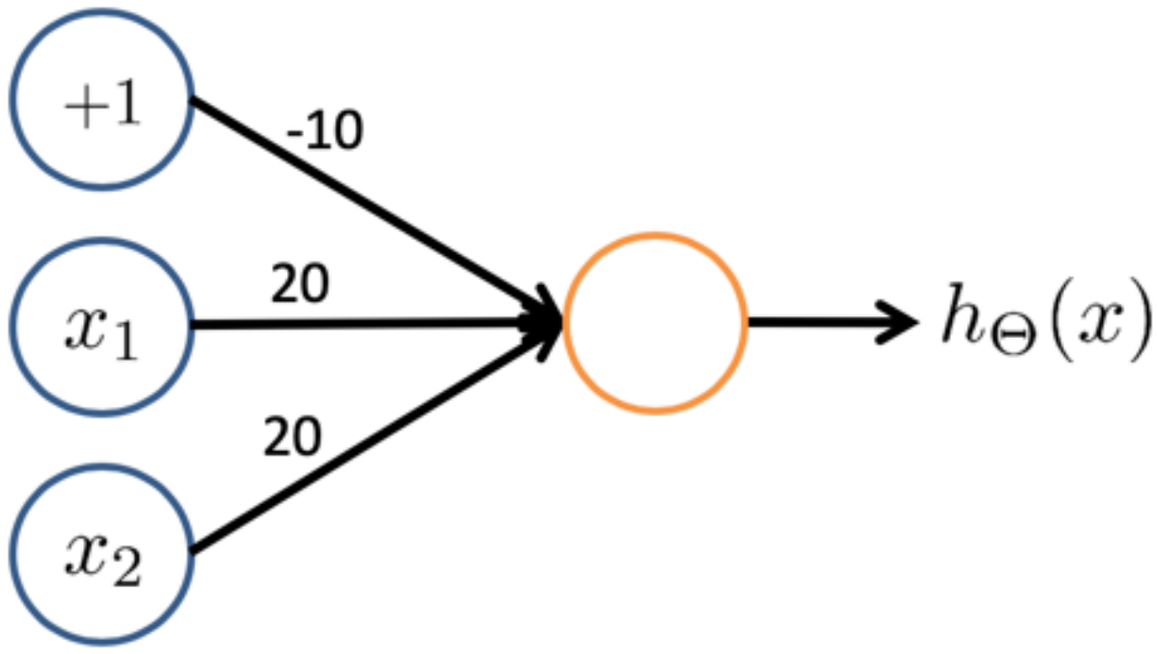


$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$



x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

Example: OR

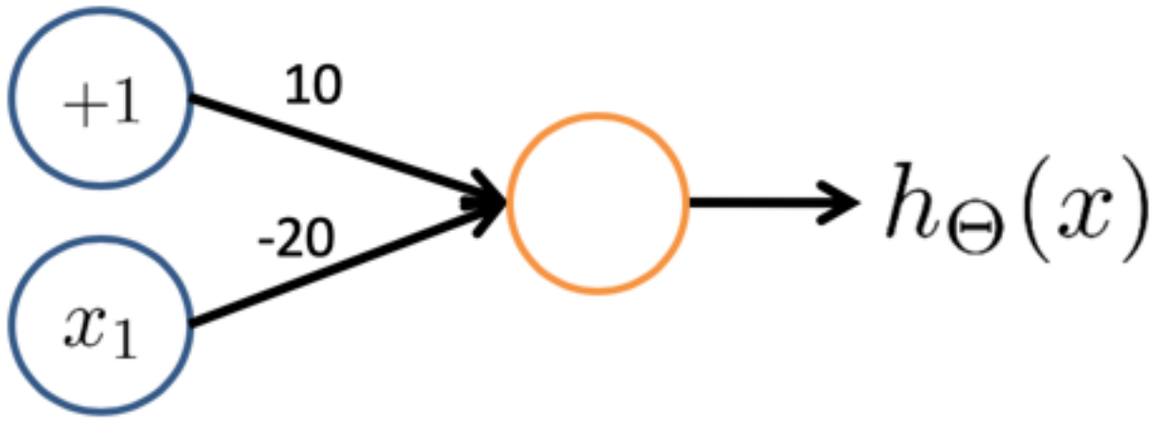


$$h_\Theta(x) = g(-10 + 20x_1 + 20x_2)$$

x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

x_1 AND x_2 | x_1 OR x_2

Negation:

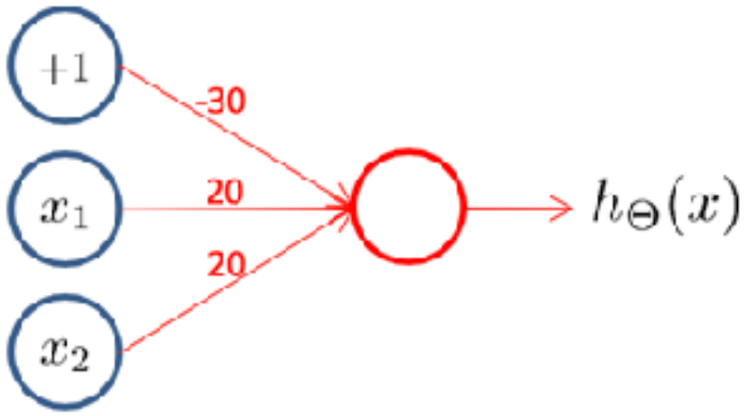


x_1	$h_\Theta(x)$
0	
1	

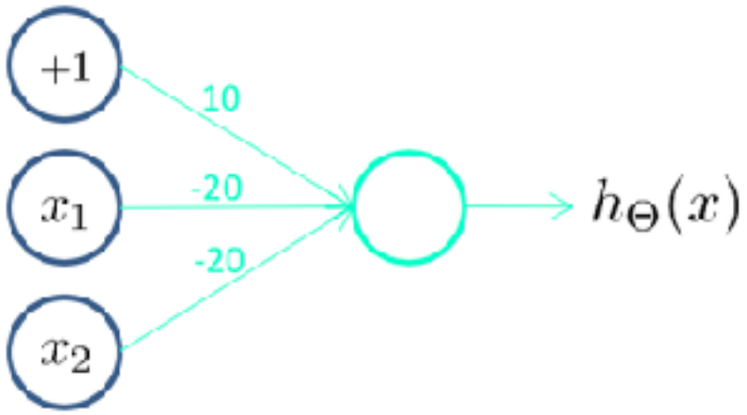
$h_\Theta(x) = g(10 - 20x_1)$ | (NOT x_1) AND (NOT x_2)?

Putting it together

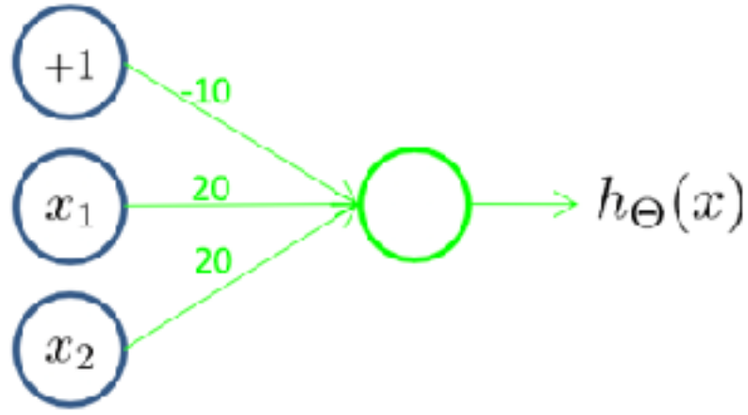
x_1 XNOR x_2



x_1 AND x_2



(NOT x_1) AND (NOT x_2)



x_1 OR x_2

x_1 XNOR x_2 =

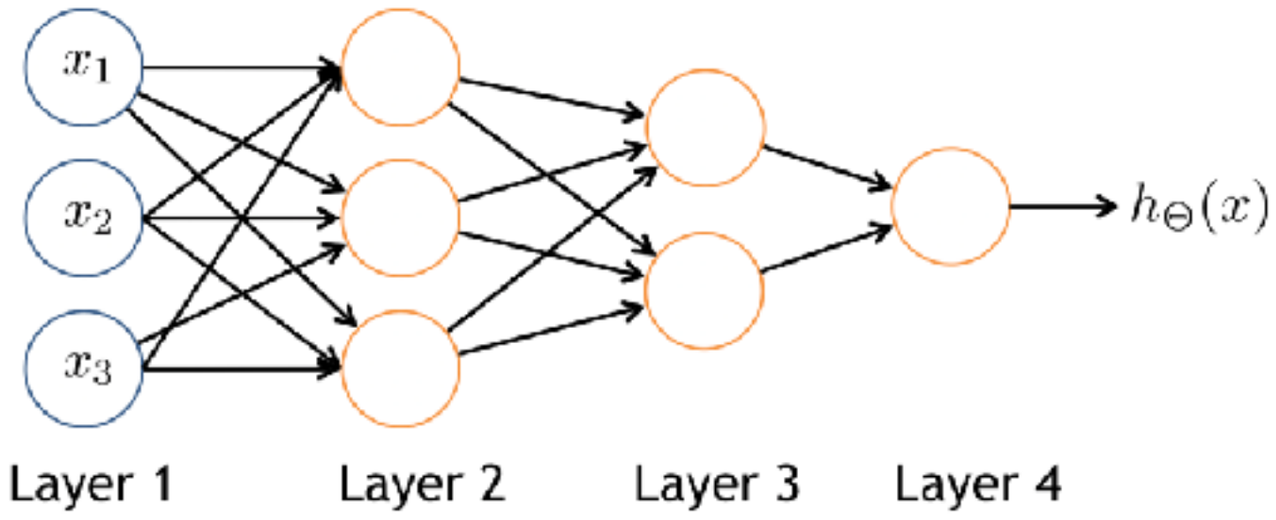
(x_1 AND x_2) OR

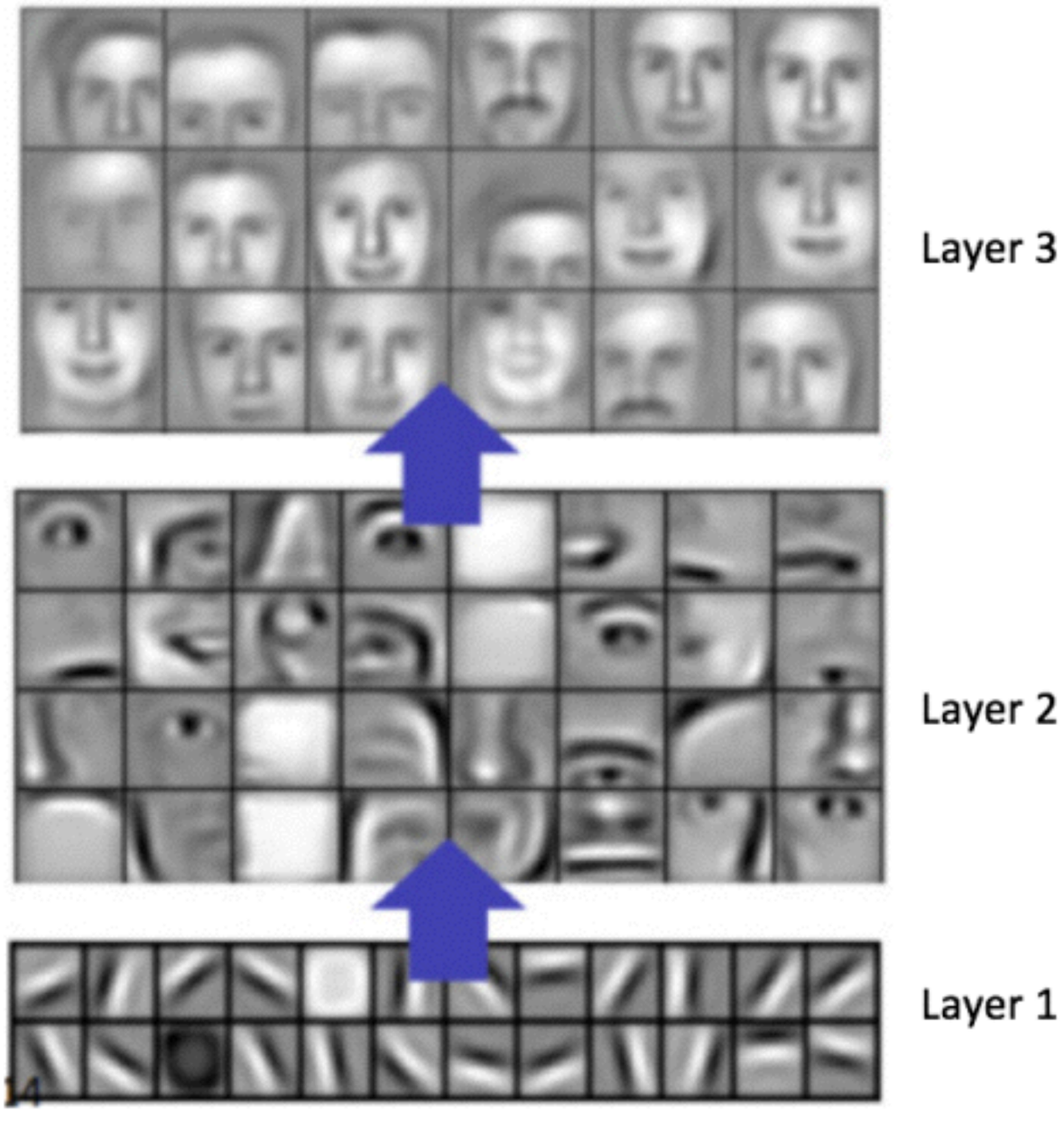
((NOT x_1) AND (NOT x_2))

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

Neural networks:
representation
ANN learns its own features

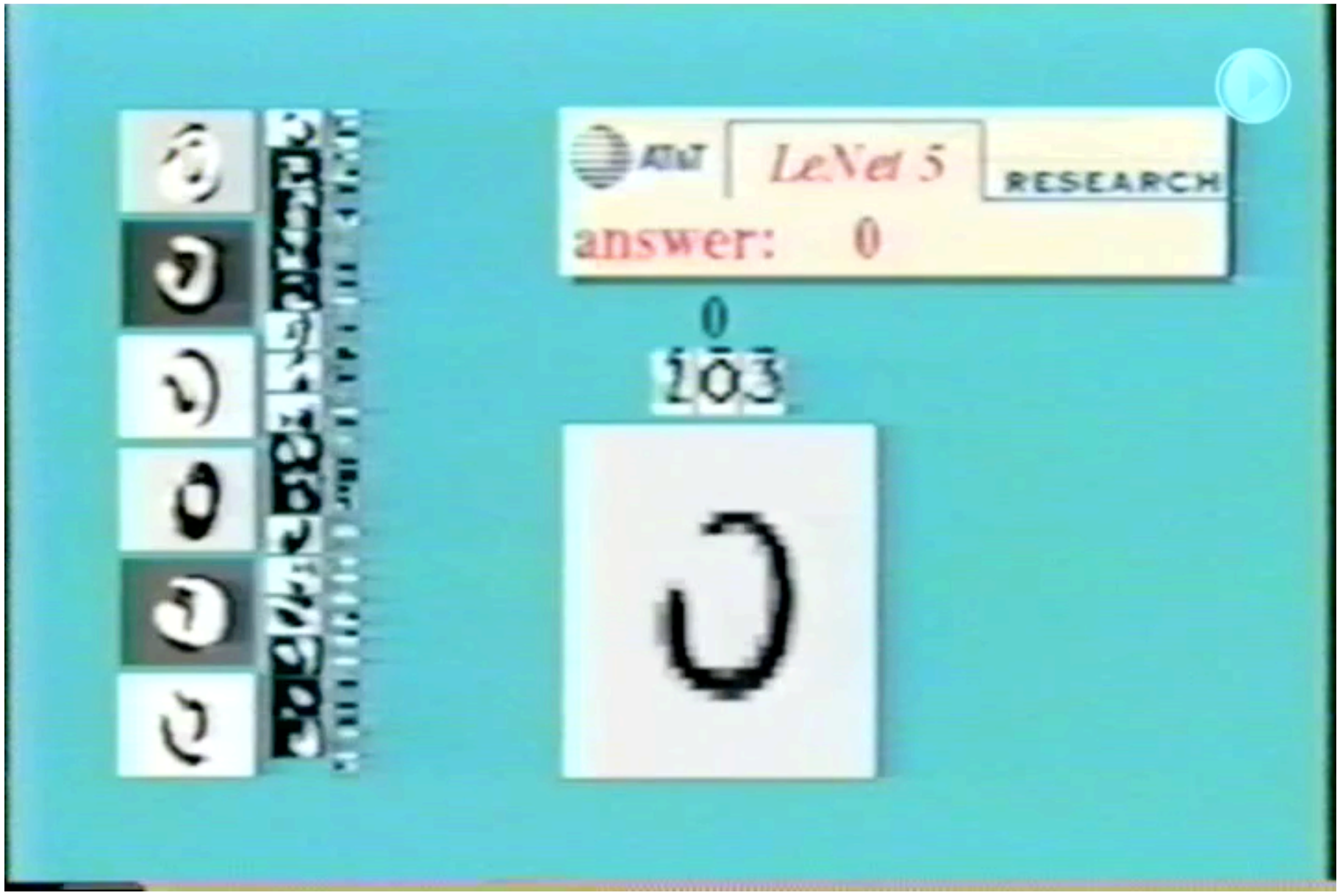
Neural network intuition





http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

Handwritten digit classification



https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Neural networks:
representation
Multi-class classification

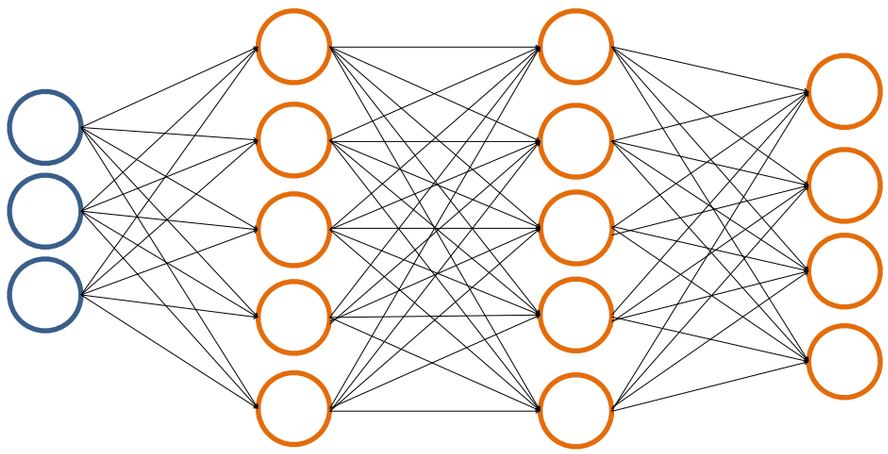
Multiple output units: One-vs-all

Pedestrian

Car

Motorcycle

Truck

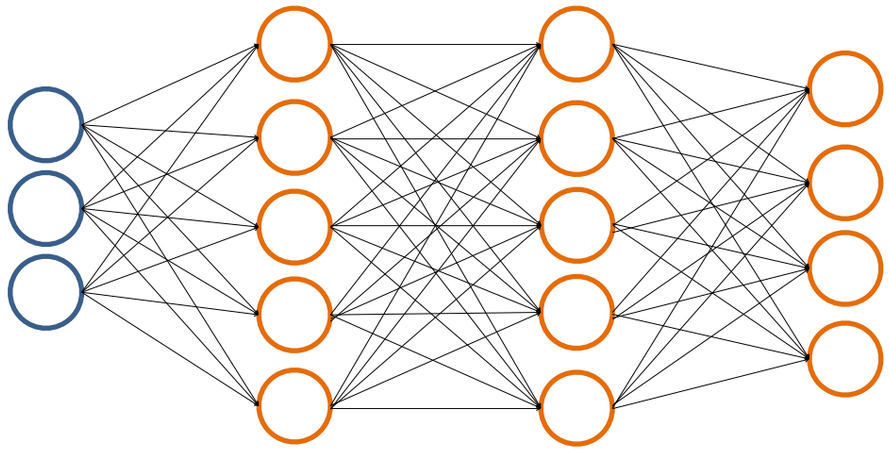


$h_\Theta(x) \in \mathbb{R}^4$

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

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