
Deep Learning

PhD. (c) Junior Fabian A.

**La presentación fue amablemente compartida por Prof. Anderson Rocha - (RECOD - IC - UNICAMP)*



Many inventions were inspired by Nature ...

Birds inspired us to fly



Many inventions were inspired by Nature ...

Burdock plants inspired velcro



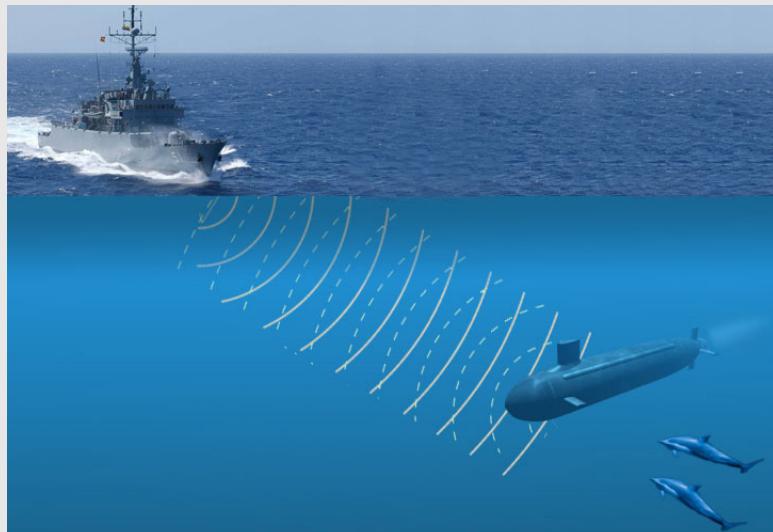
<http://www.youngworldclub.com/wp-content/uploads/2017/04/velcro.jpeg>



https://c1.staticflickr.com/3/2882961455_f2d6bec40c_b.jpg

Many inventions were inspired by Nature ...

Dolphins inspired sonar development



https://sites.google.com/site/echolocationkawproject/_/rsrc/1459209762464/sonars/image.jpeg



Many inventions were inspired by Nature ...

Dogs inspired ...



https://farm2.staticflickr.com/1586/24734777053_d1b6b8bb4_o.jpg

It seems logical to look at the
brain's architecture for inspiration on
how to build an intelligent machine.

Today's Agenda

- Artificial Neural Networks
 - From Biological to Artificial Neurons
 - Biological Neurons
 - Logical Computations with Neurons
 - The Perceptron
 - Multi-Layer Perceptron and Backpropagation

From Biological to Artificial Neurons

From Biological to Artificial Neurons

- 1943: Artificial Neural Networks (ANNs) were first introduced by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts.

“A Logical Calculus of Ideas Immanent in Nervous Activity”,
Warren McCulloch and Walter Pitts. The bulletin of mathematical
biophysics (1943).

From Biological to Artificial Neurons

- **Until the 1960s:** The early successes of ANNs led to the widespread belief that we would soon be conversing with truly intelligent machines.
- When it became clear that this promise would go unfulfilled funding flew elsewhere and ANNs entered a long dark era.

From Biological to Artificial Neurons

- **1980s:** There was a revival of interest in ANNs as new network architectures were invented and **better training techniques** were developed.

“Learning representations by backpropagating errors”. David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams. Nature (1986).

From Biological to Artificial Neurons

- **1990s: Powerful alternative Machine Learning techniques** such as Support Vector Machines were favored by most researchers, as they seemed to offer better results and stronger theoretical foundations.

From Biological to Artificial Neurons

- **2010s:** We are now witnessing yet another wave of interest in ANNs.

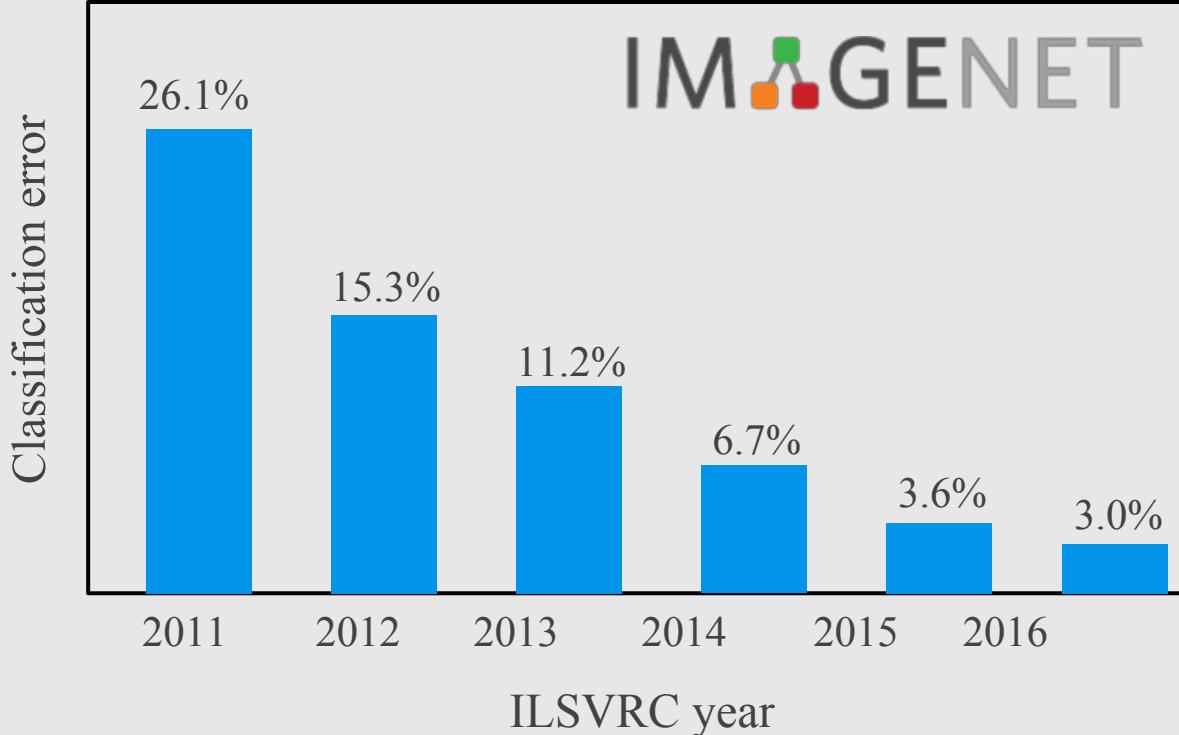


ILSVRC 2012 — Image Classification task

Rank	Name	Error Rate (%)	Description
1	University of Toronto	15.3	Deep Learning
2	University of Tokyo	26.2	Hand-crafted features and learning models
3	University of Oxford	26.9	
4	Xerox/INRIA	27.0	

Object recognition over 1,000,000 images and 1,000 categories (2 GPU)

“ImageNet classification with deep convolutional neural networks”. Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. In: NIPS, 2012.



“ImageNet classification with deep convolutional neural networks”. Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. In: NIPS, 2012.

From Biological to Artificial Neurons

- **2010s:** We are now witnessing yet another wave of interest in ANNs.

Will this wave die out like the previous ones did?

From Biological to Artificial Neurons

1. There is now a **huge quantity of data** available to train neural networks.



www.image-net.org

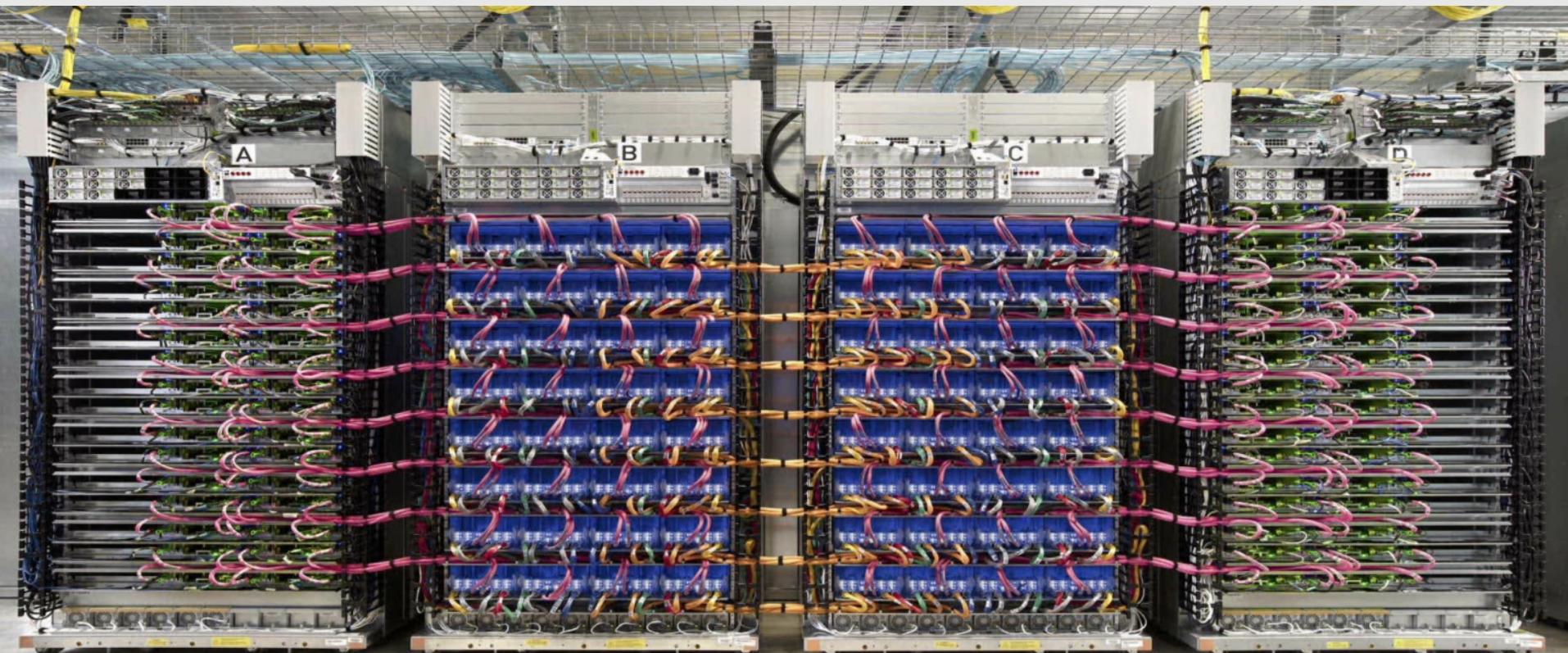
22K categories and **14M** images

- Animals
 - Bird
 - Fish
 - Mammal
 - Invertebrate
- Plants
 - Tree
 - Flower
- Food
- Materials
- Structures
- Artifact
 - Tools
 - Appliances
 - Structures
- Person
- Scenes
 - Indoor
 - Geological Formations
- Sport Activities



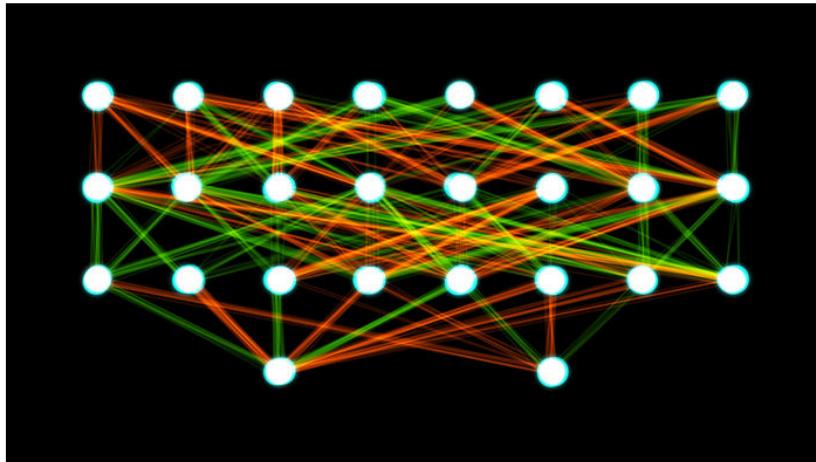
From Biological to Artificial Neurons

1. There is now a **huge quantity of data** available to train neural networks.
2. **Computing power** now makes it possible to train large neural networks in a reasonable amount of time.



From Biological to Artificial Neurons

1. There is now a **huge quantity of data** available to train neural networks.
2. **Computing power** now makes it possible to train large neural networks in a reasonable amount of time.
3. The **training algorithms** have been **improved**.

SHARE

A representation of a neural network.

Akritasa/Wikimedia Commons

Brainlike computers are a black box. Scientists are finally peering inside

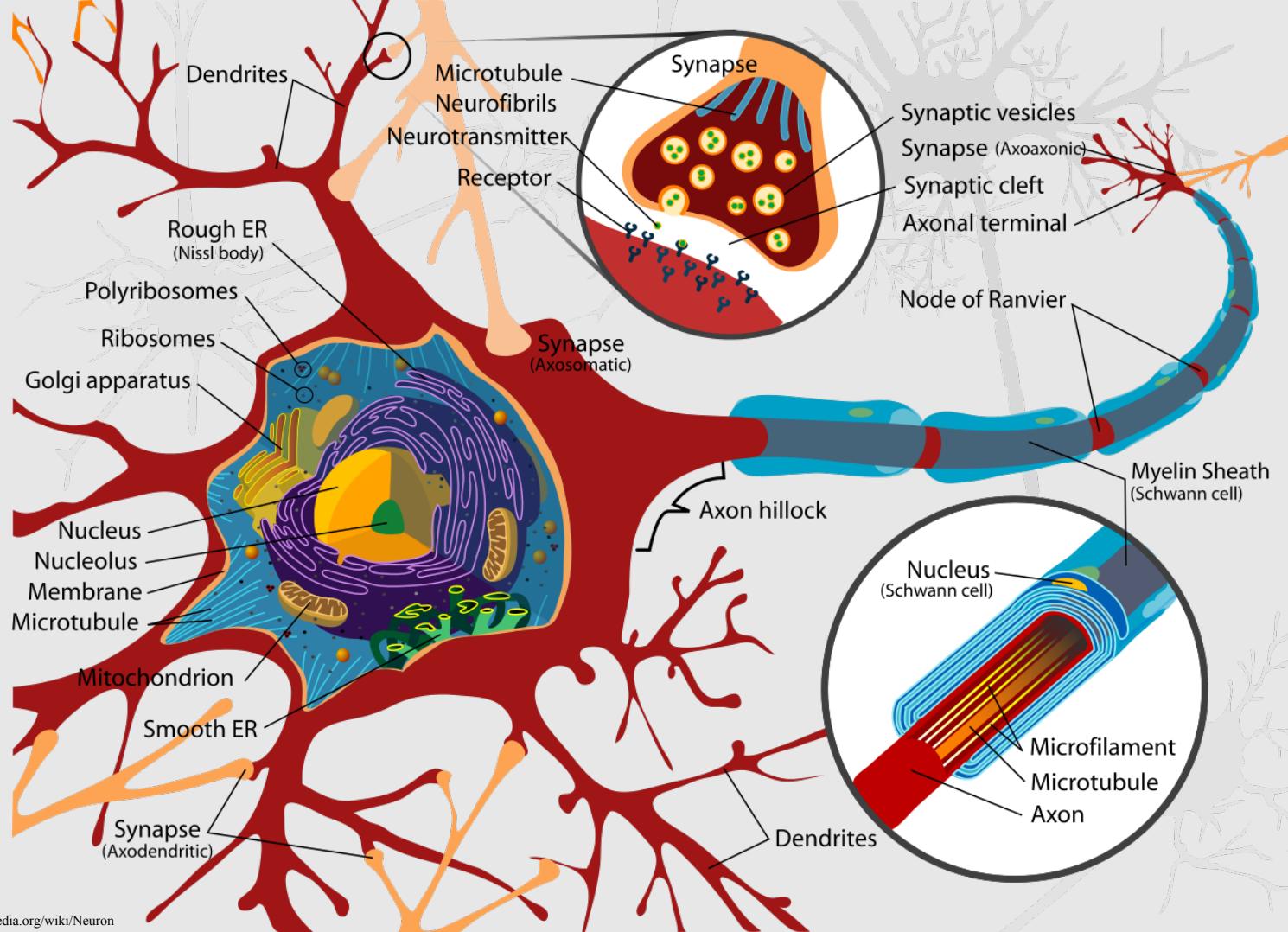
By **Jackie Snow** | Mar. 7, 2017 , 3:15 PM

Last month, Facebook announced software that could simply look at a photo and tell, for example, whether it was a picture of a cat or a dog. A related program identifies cancerous

From Biological to Artificial Neurons

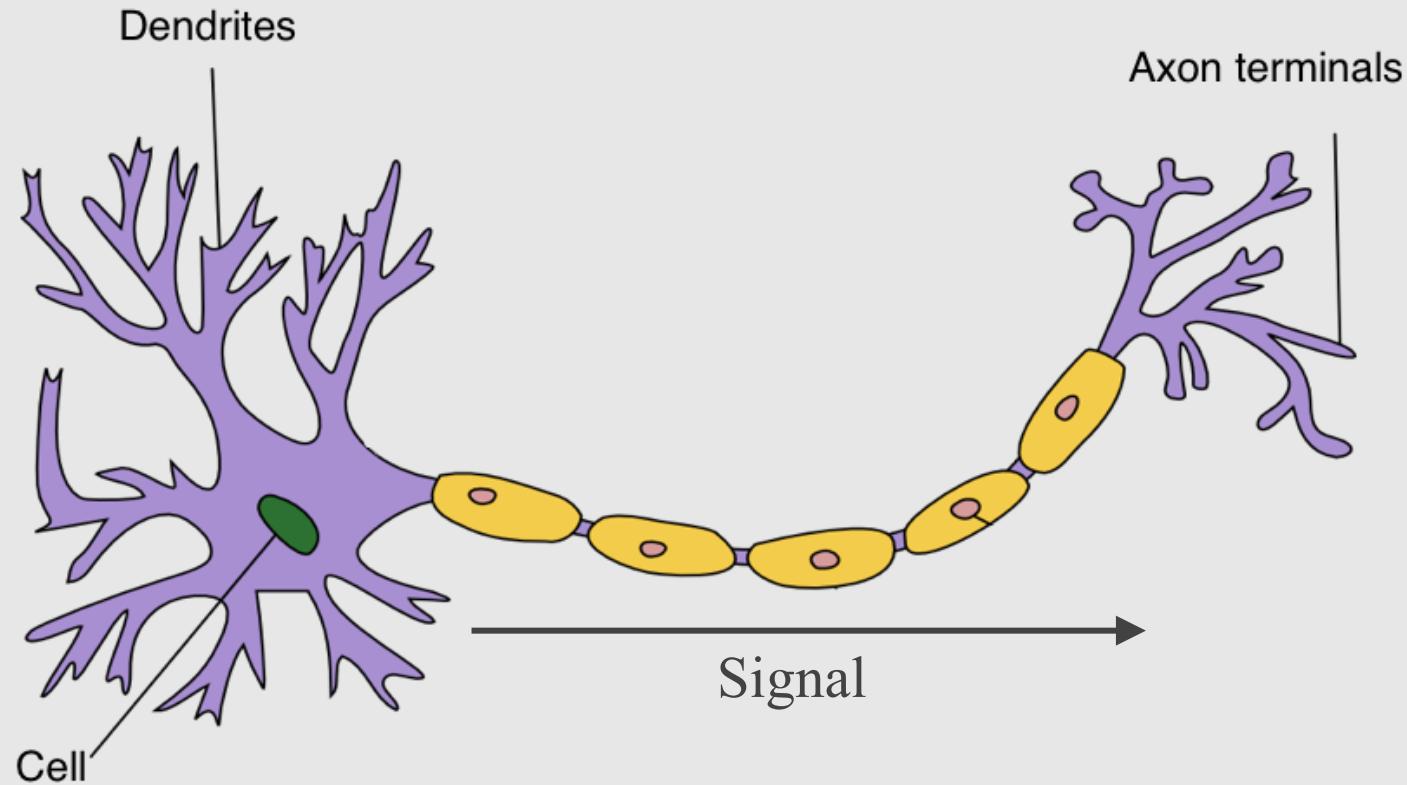
1. There is now a **huge quantity of data** available to train neural networks.
2. **Computing power** now makes it possible to train large neural networks in a reasonable amount of time.
3. The **training algorithms** have been **improved**.
4. ANNs seem to have entered a virtuous circle of funding and progress.

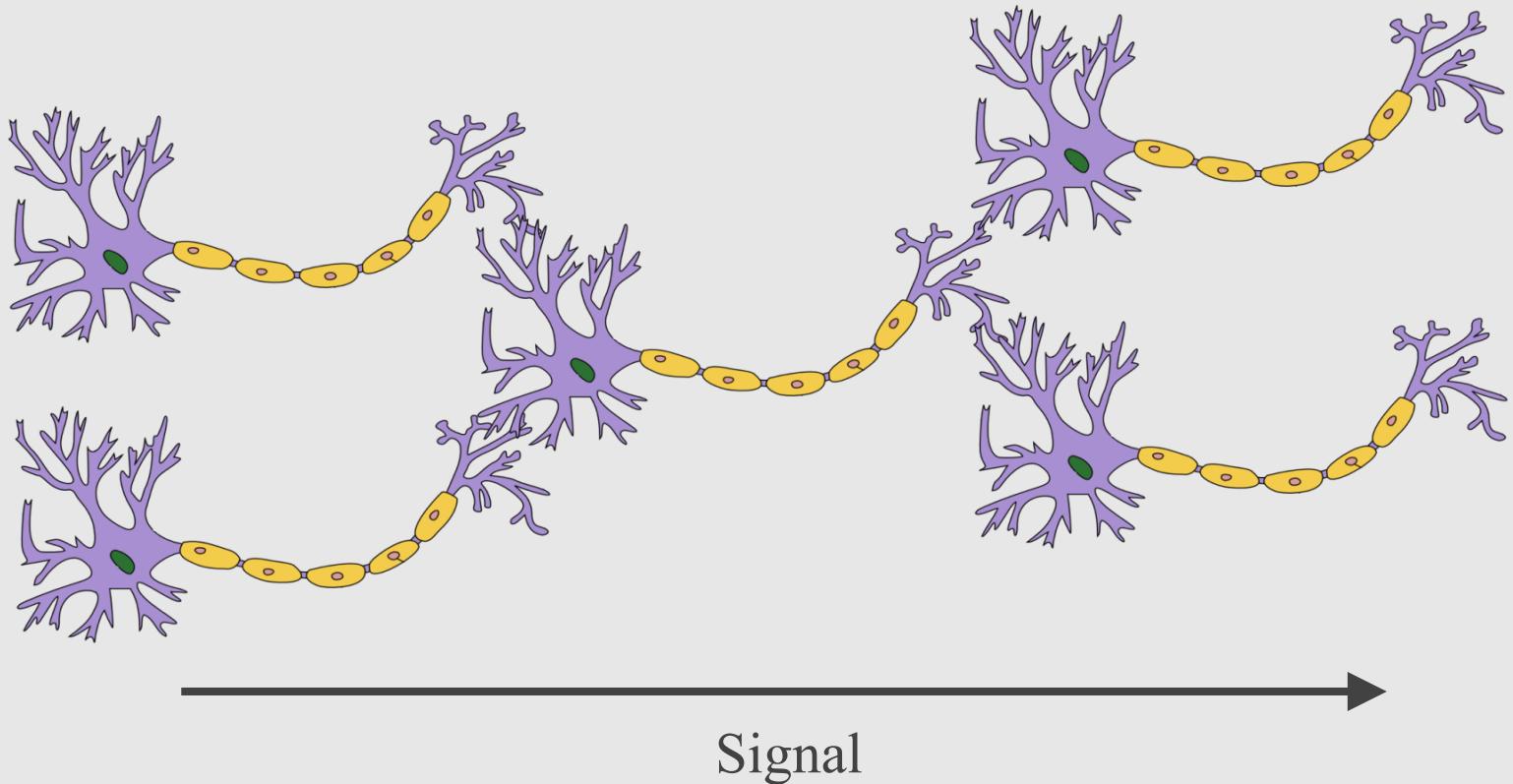
Biological Neurons





<https://www.youtube.com/watch?v=A9Xru1ReRwc>





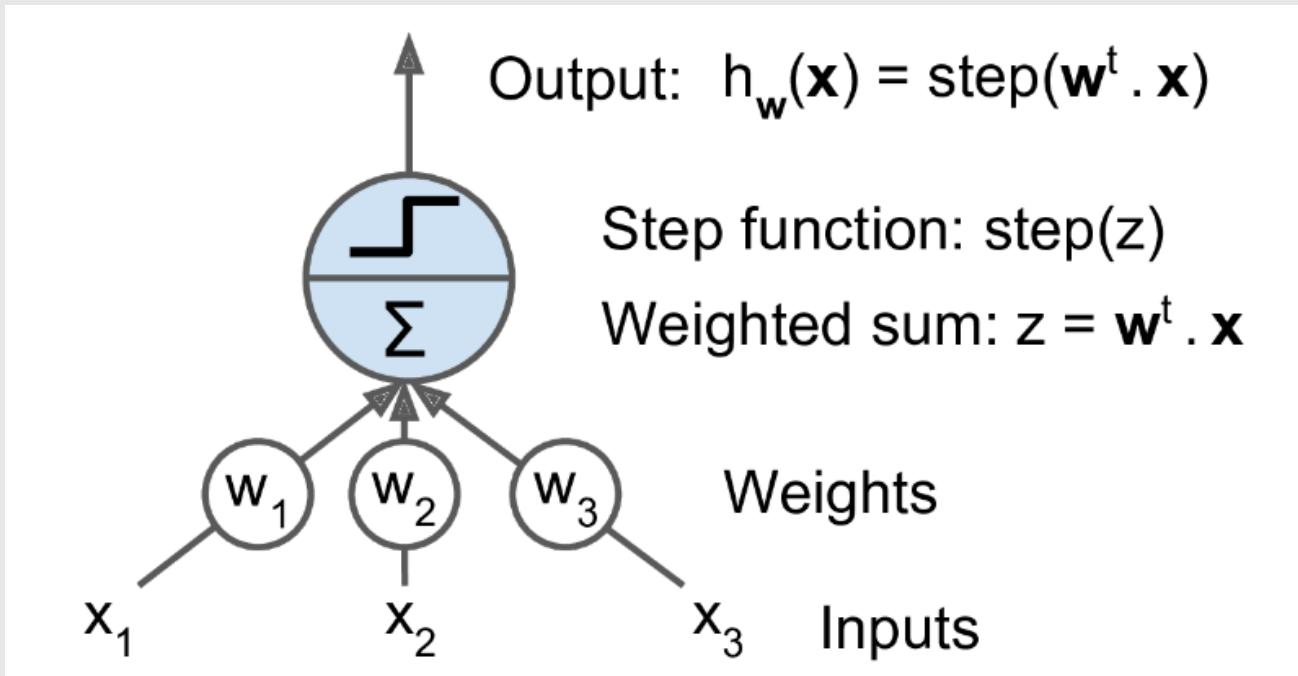
The Perceptron

The Perceptron

Invented in 1957 by Frank Rosenblatt.

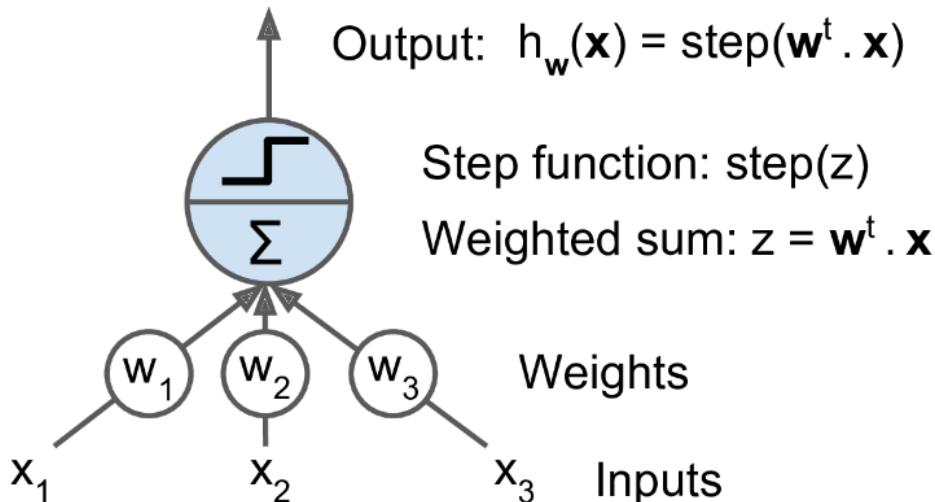
- It is based on a Linear Threshold Unit (LTU):
 - The inputs and output are now **numbers** (instead of binary on/off values) and each input connection is associated with a **weight**.
- The LTU computes a weighted sum of its inputs then it applies a step function to that sum and outputs the result.

The Perceptron



Linear Threshold Unit

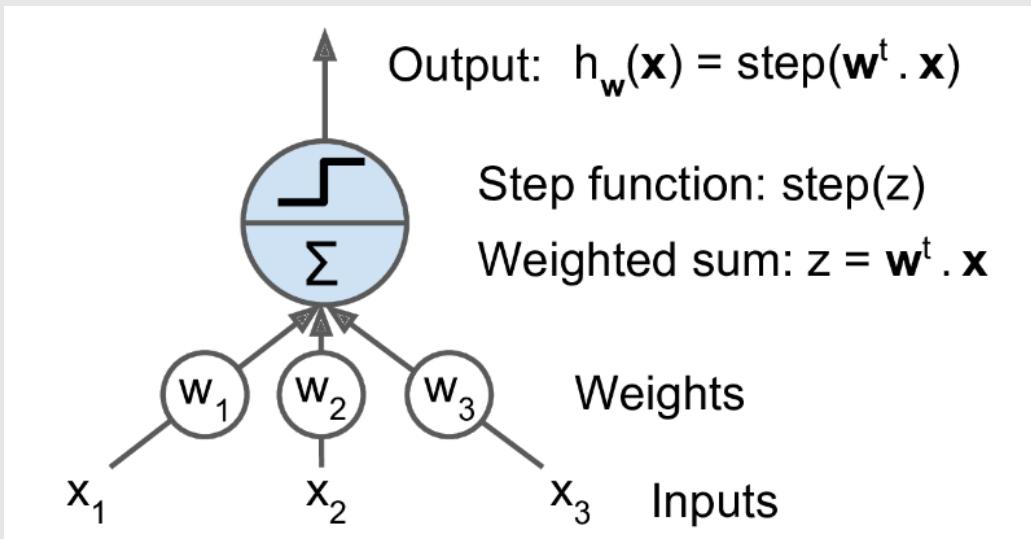
The Perceptron



Linear Threshold Unit

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

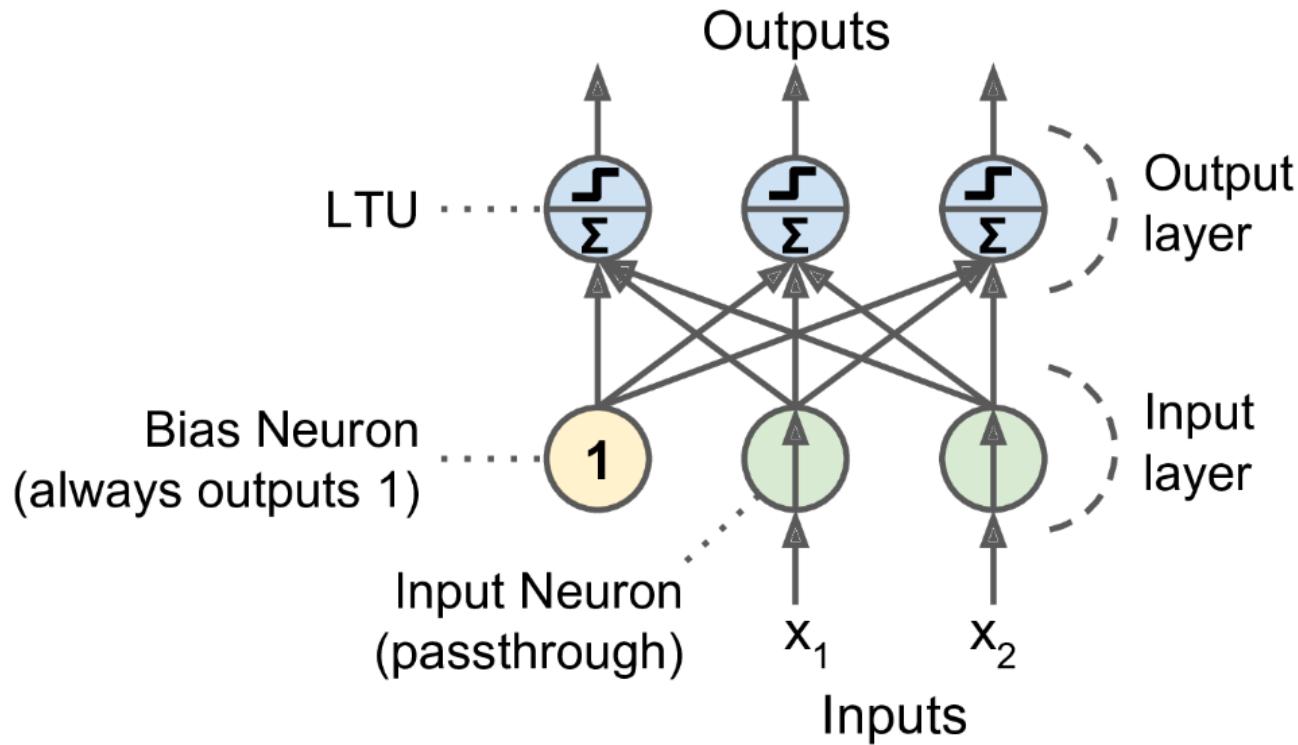
The Perceptron



Linear Threshold Unit

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

$$\text{sign}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$



The Perceptron

So how is a Perceptron trained?

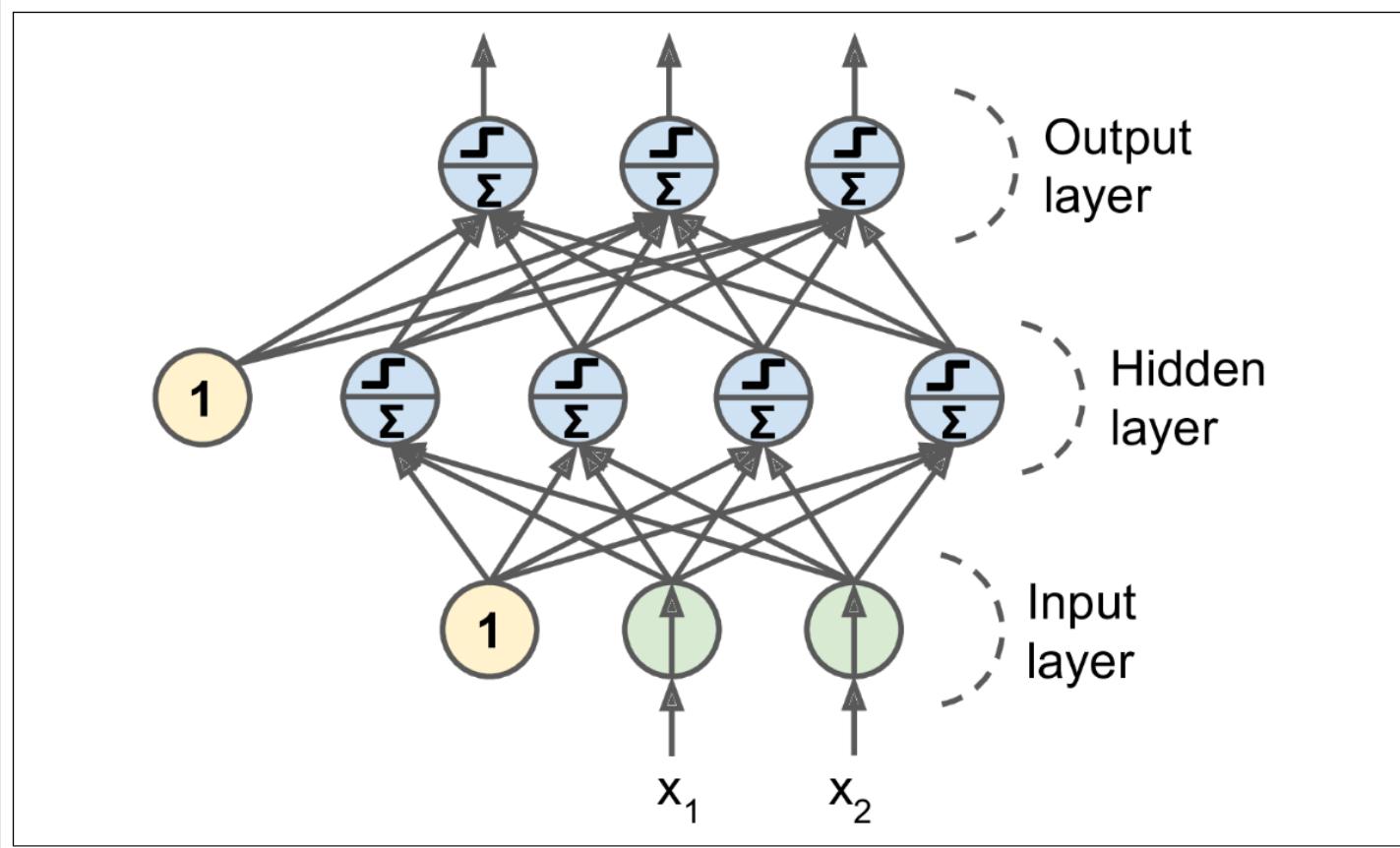
Hebb's rule (1949): the connection weight between two neurons is increased whenever they have the same output.

- The Perceptron is fed one training instance at a time, and for each instance it makes its predictions.
- For every output neuron that produced a wrong prediction, it reinforces the connection weights from the inputs that would have contributed to the correct prediction.

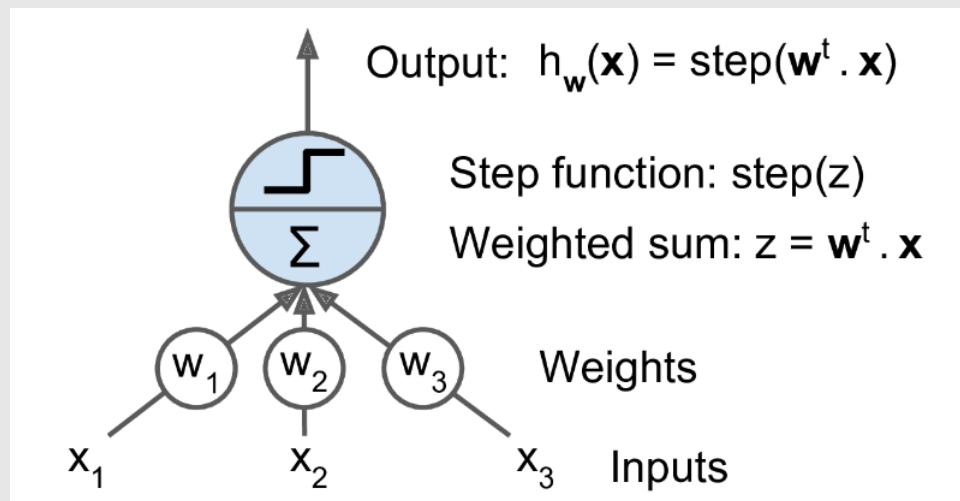
So how is a Perceptron trained?

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \alpha(\hat{y}_j - y_j)x_i$$

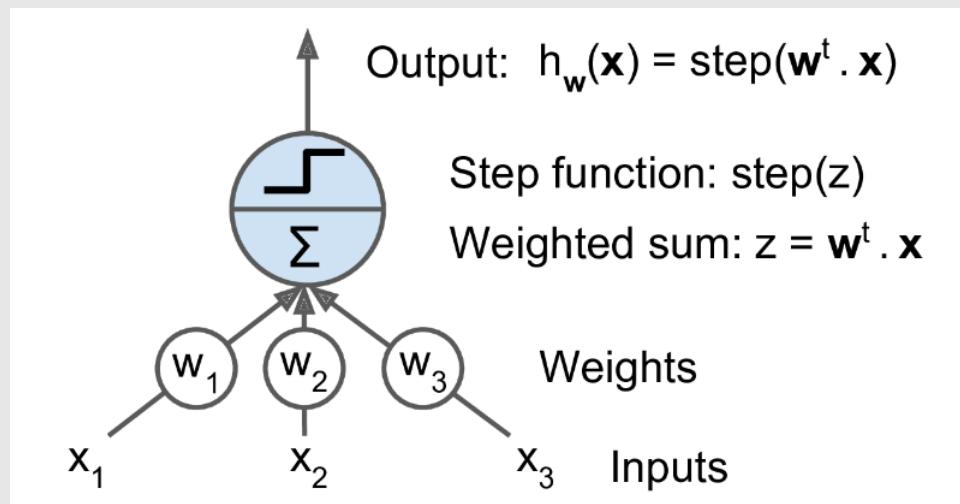
Multi-Layer Perceptron and Backpropagation



Neuron Model: Logistic Unit



Neuron Model: Logistic Unit



Inputs



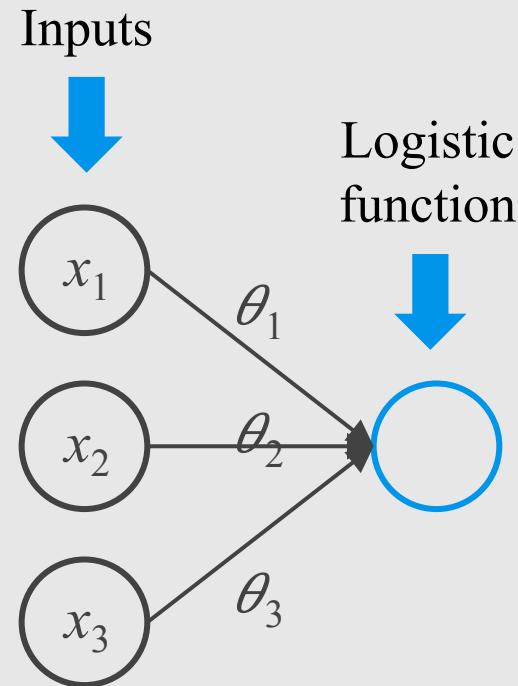
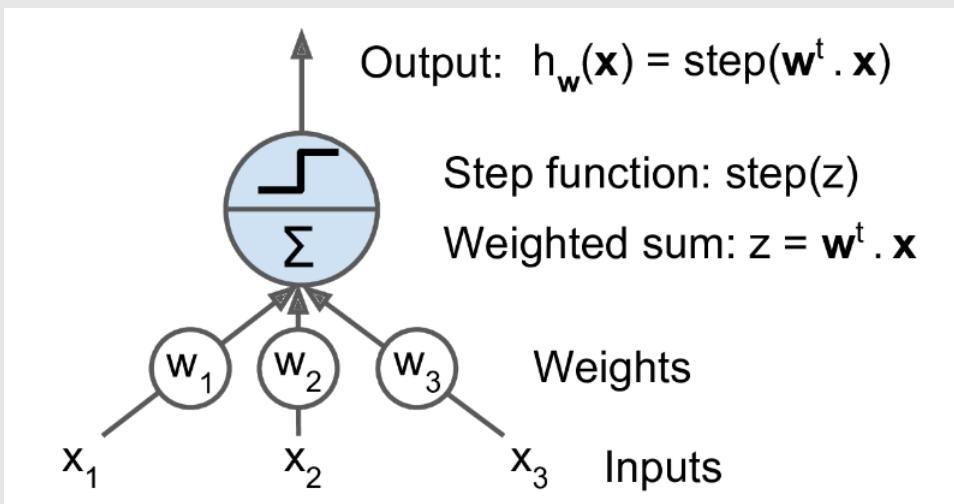
x_1

x_2

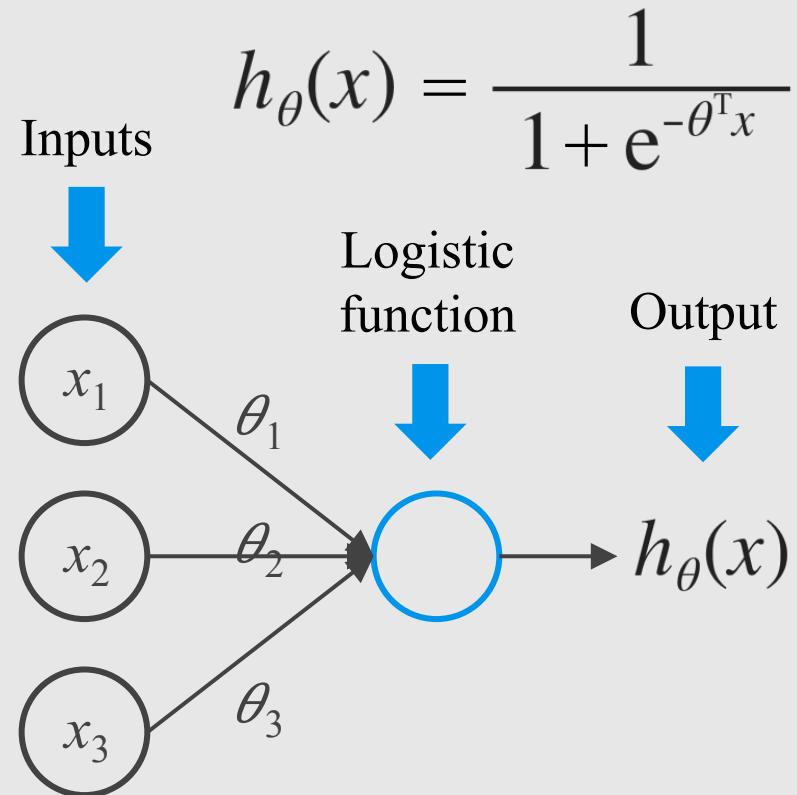
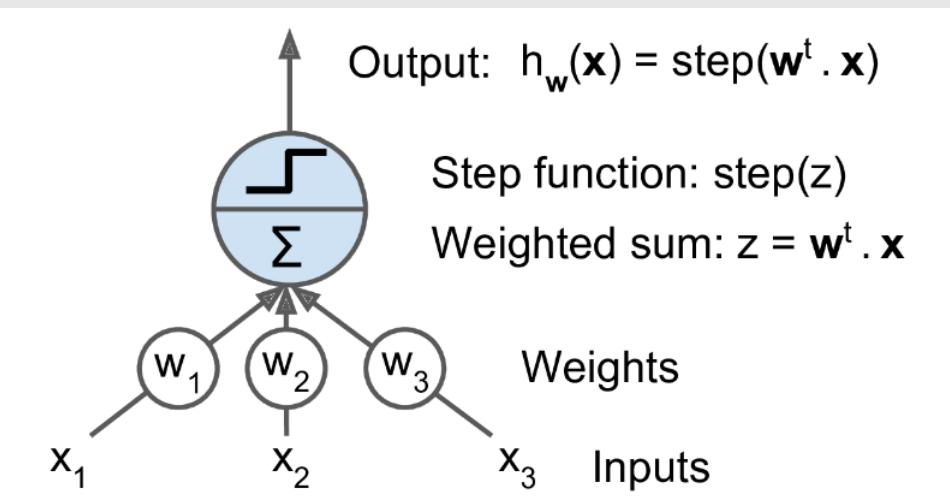
x_3

Neuron Model: Logistic Unit

$$\frac{1}{1 + e^{-\theta^T x}}$$



Neuron Model: Logistic Unit

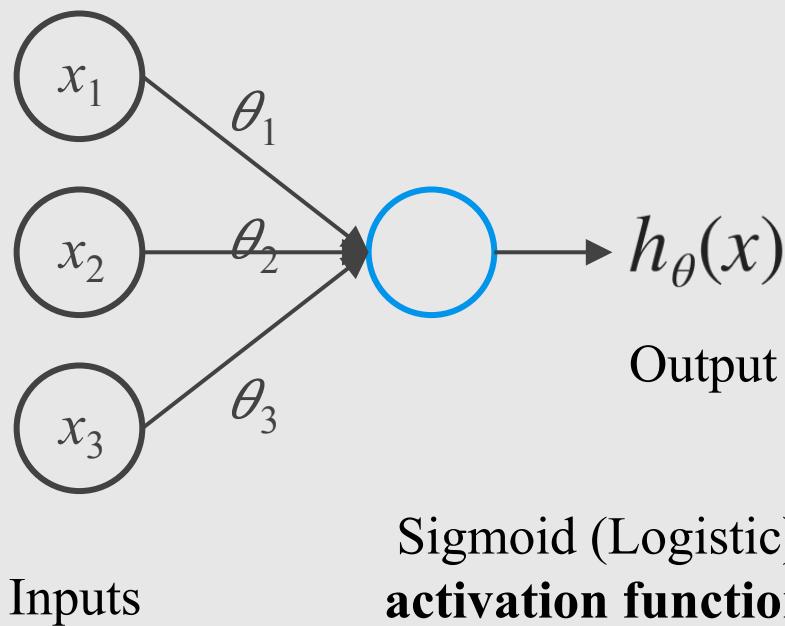


$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

weights



Neuron Model: Logistic Unit



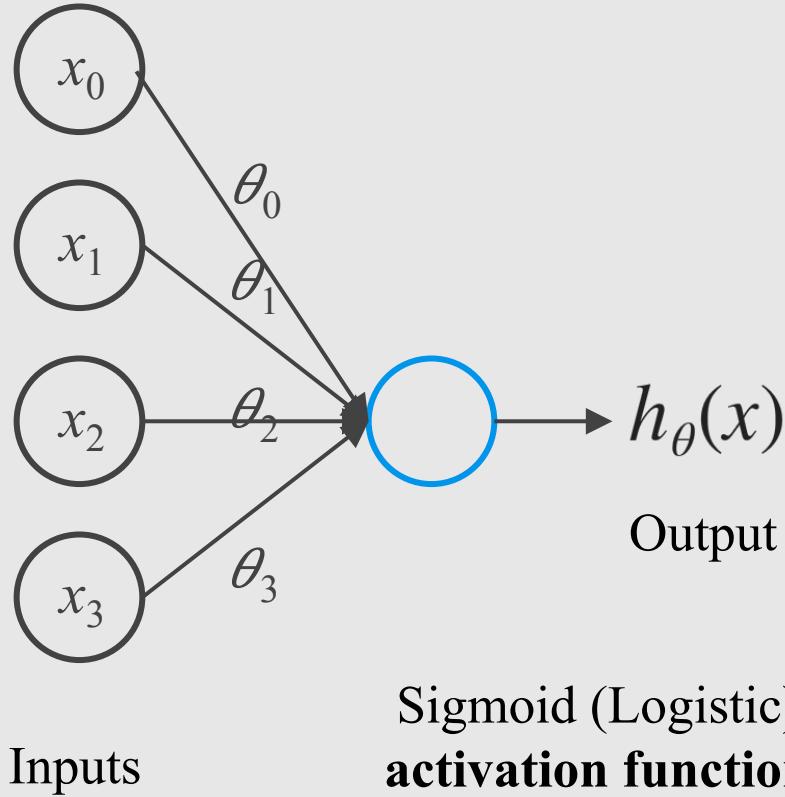
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

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

weights



Neuron Model: Logistic Unit



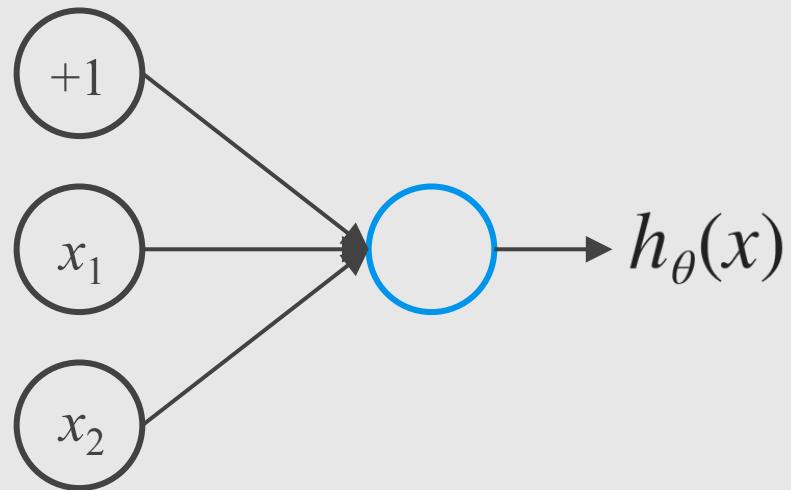
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

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

Examples

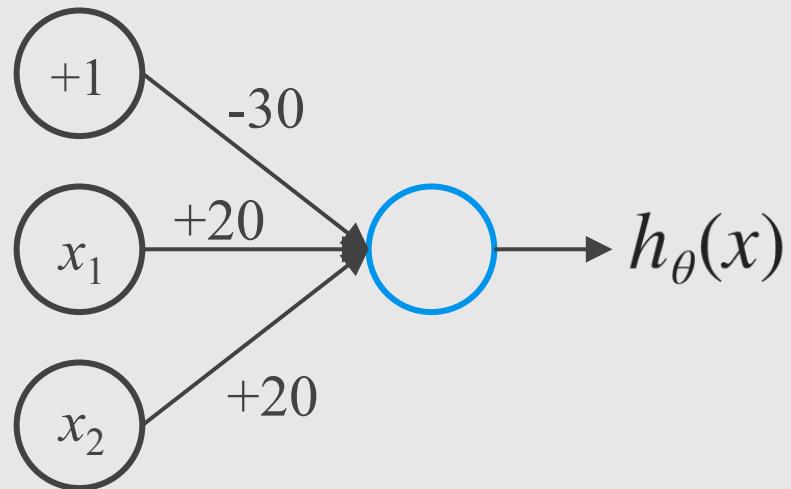
Simple Example: AND

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



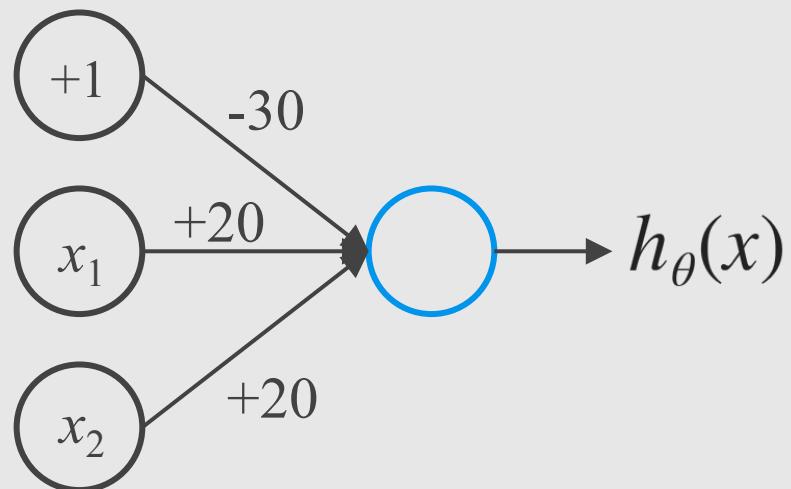
Simple Example: AND

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



Simple Example: AND

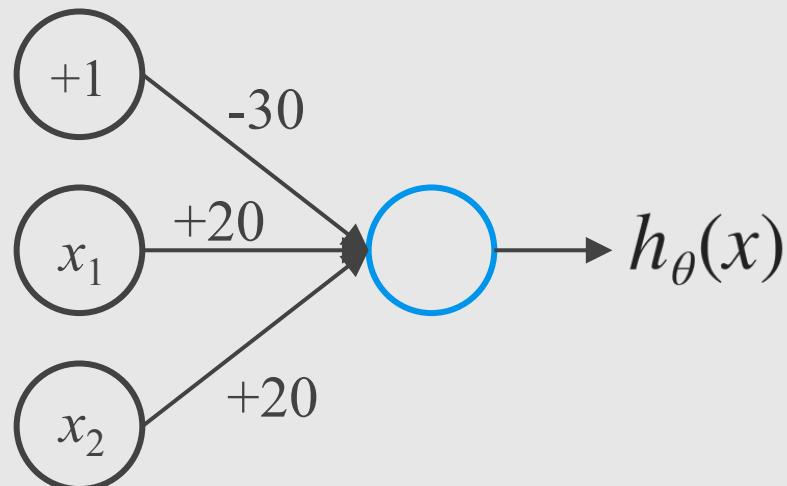
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ AND } x_2$$



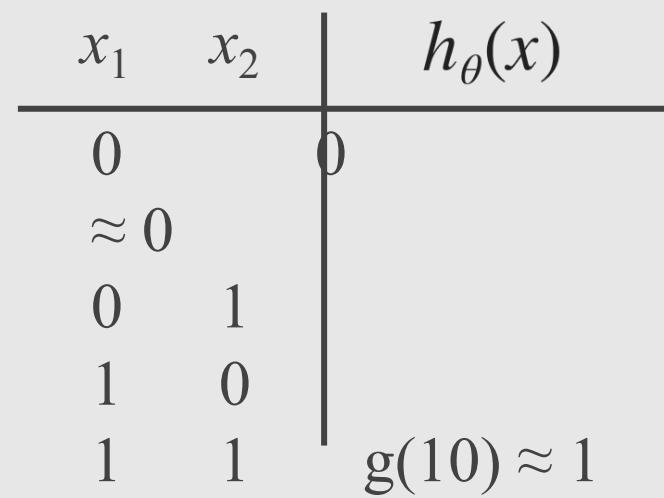
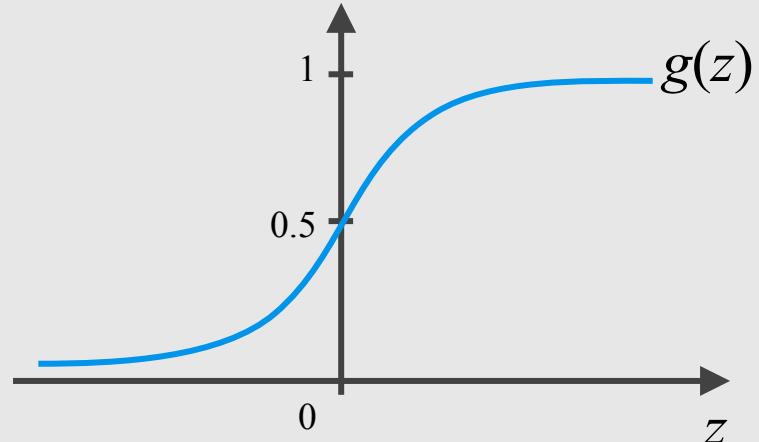
$$h_{\theta}(x) = g(-30 + 20x_1 + 20x_2)$$

Simple Example: AND

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

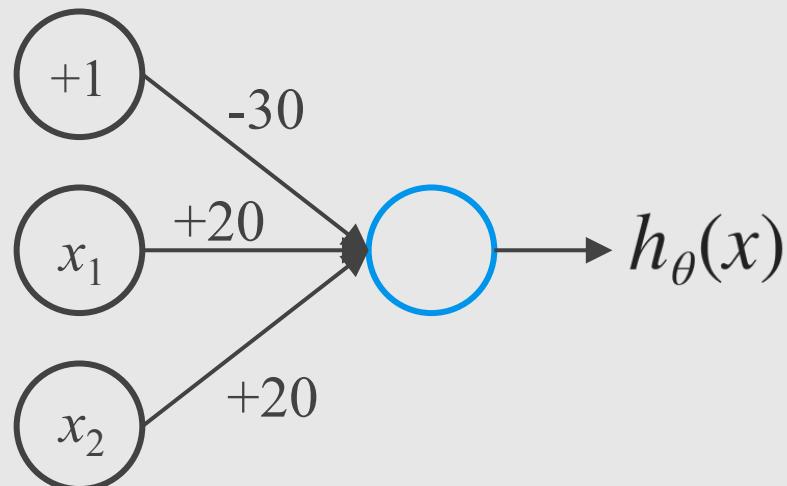


$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$

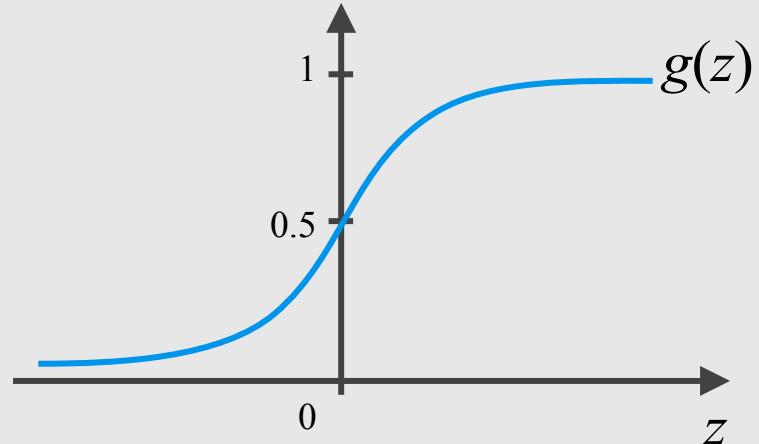


Simple Example: AND

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



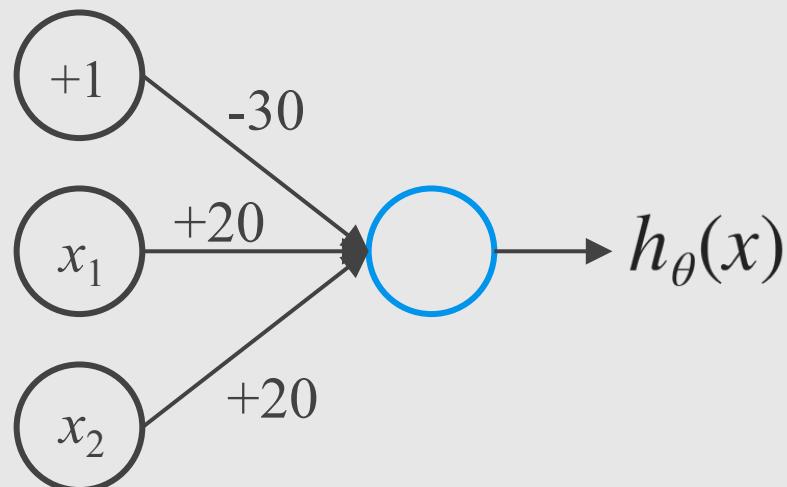
$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$



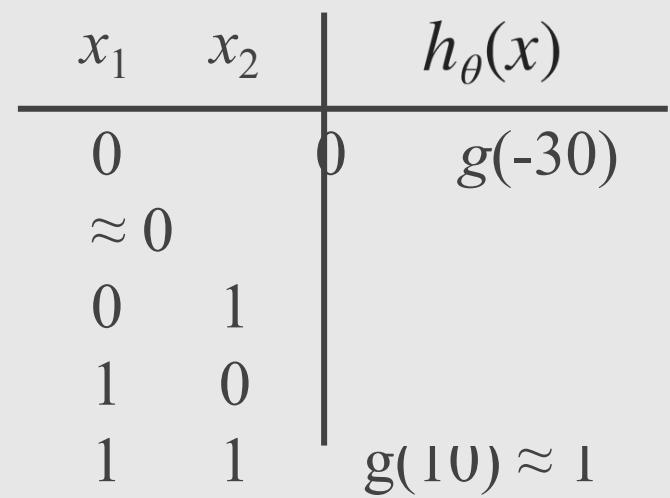
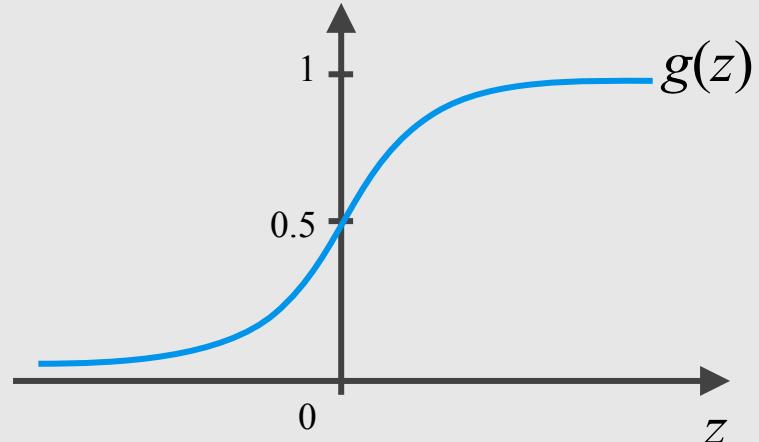
x_1	x_2	$h_\theta(x)$
0	0	$g(-30) \approx 0$
0	1	0
1	0	0
1	1	$g(10) \approx 1$

Simple Example: AND

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

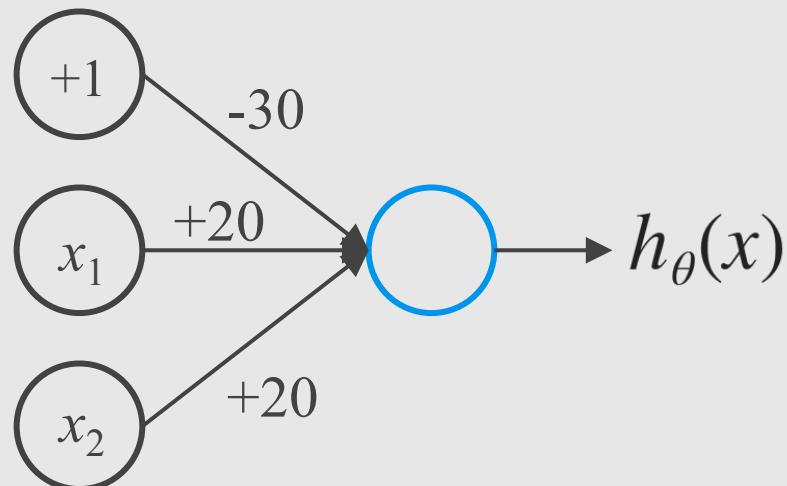


$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$

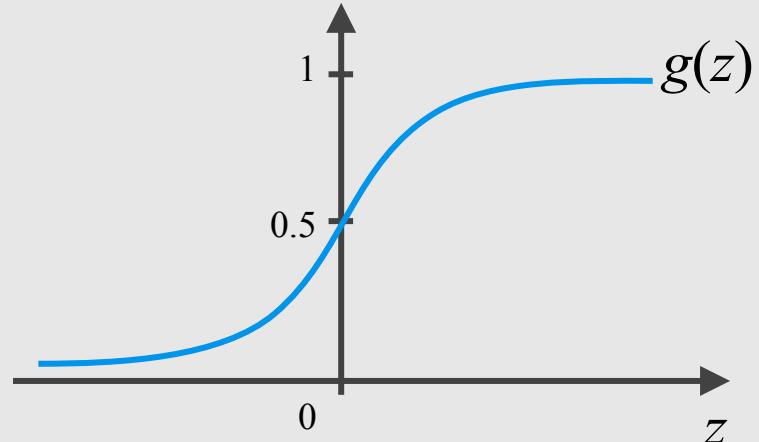


Simple Example: AND

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



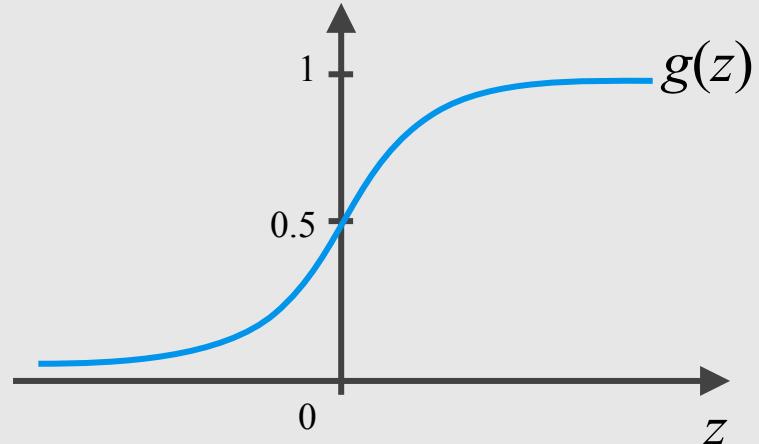
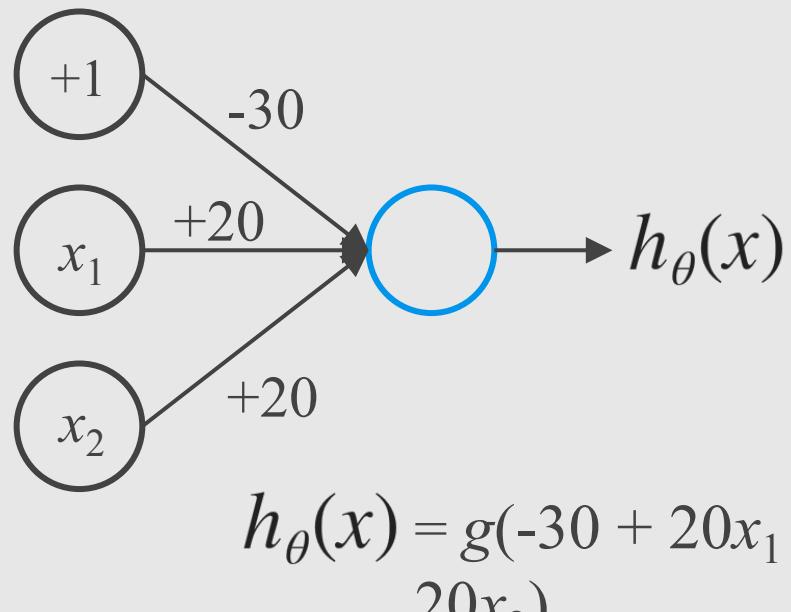
$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$



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

Simple Example: AND

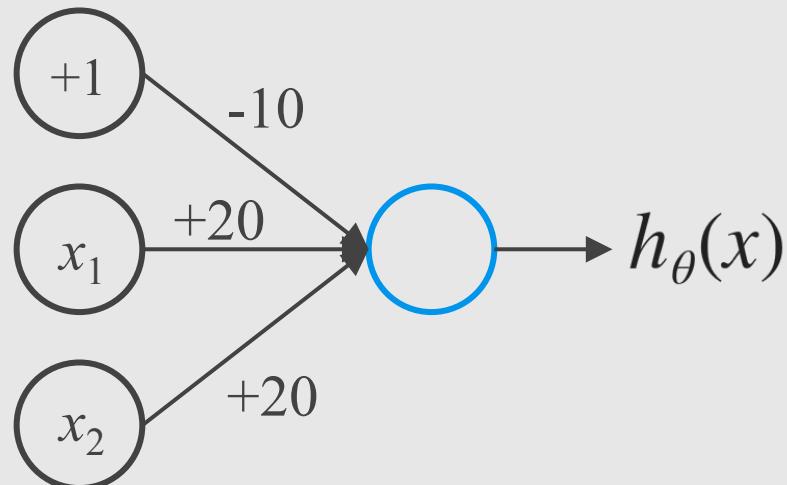
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ AND } x_2$$



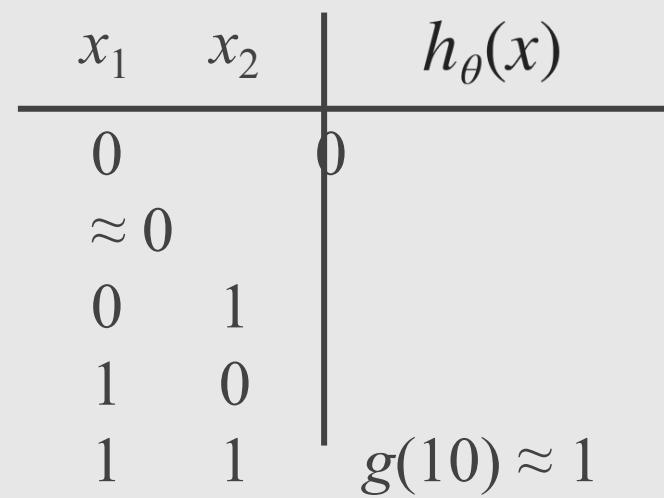
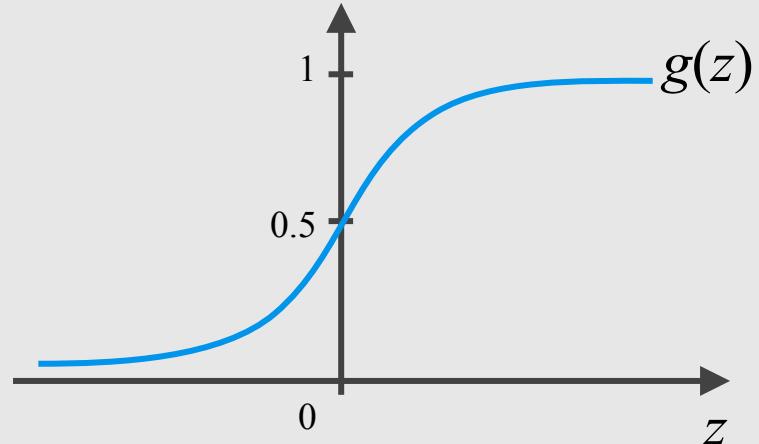
x_1	x_2	$h_\theta(x)$
0	0	$g(-30)$
≈ 0		
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

Simple Example: OR

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

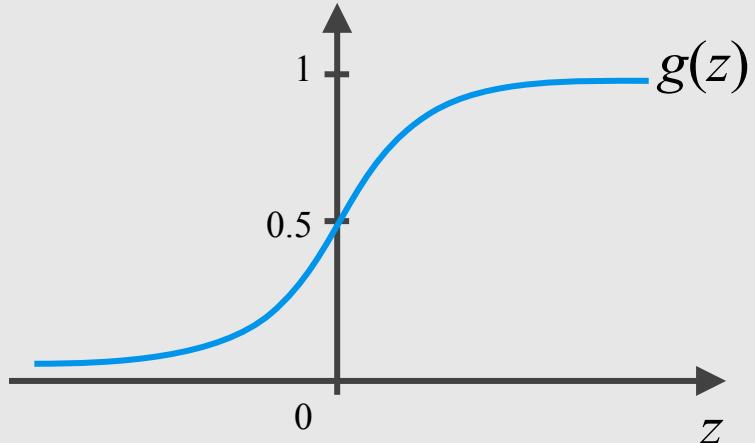
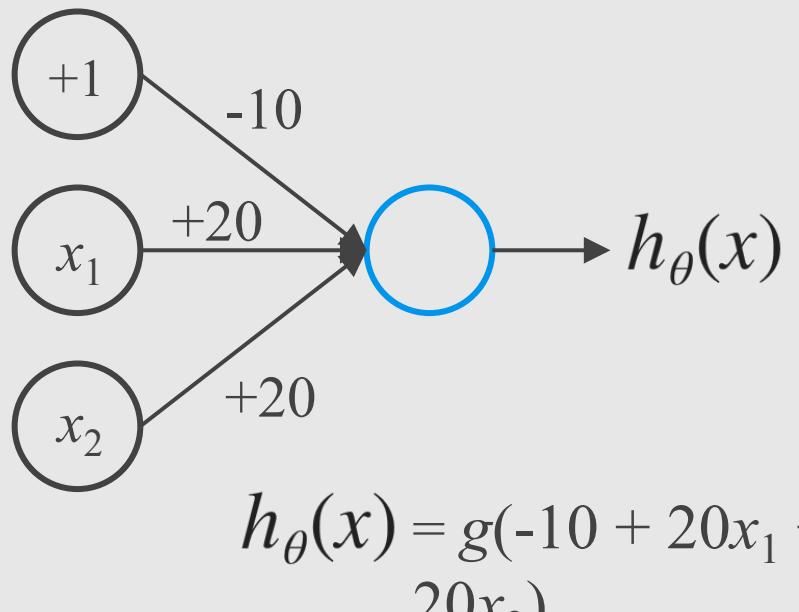


$$h_\theta(x) = g(-10 + 20x_1 + 20x_2)$$



Simple Example: OR

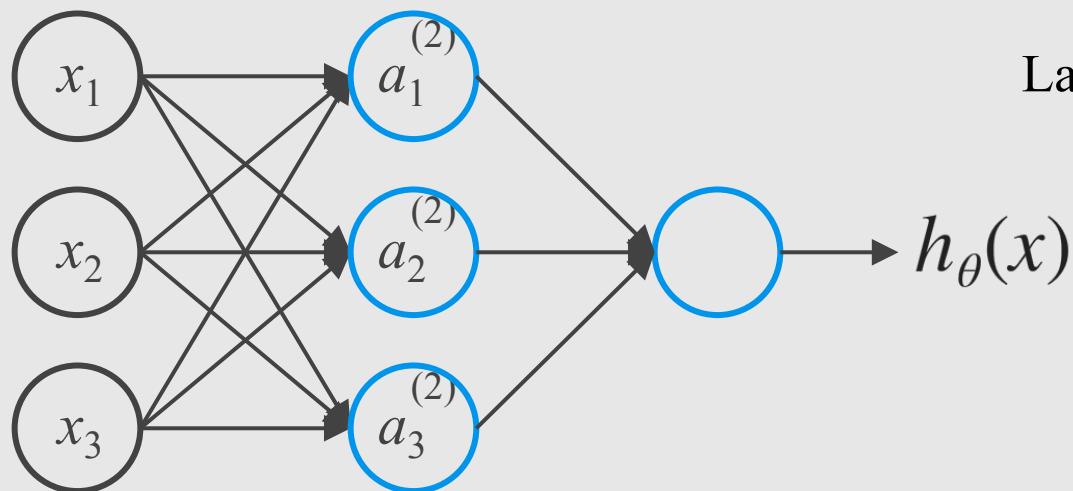
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ OR } x_2$$



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

Neural Network

Neural Network



Layer 1

Layer 2

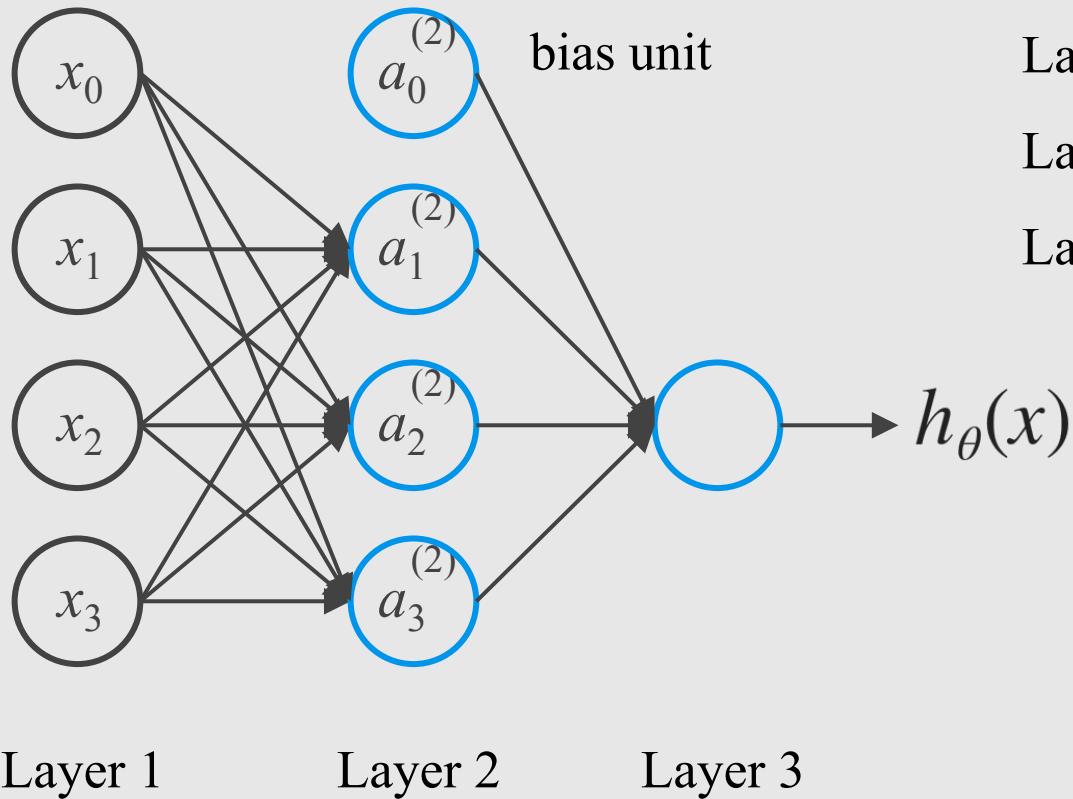
Layer 3

Layer 1 = Input layer

Layer 2 = Hidden layer

Layer 3 = Output layer

Neural Network

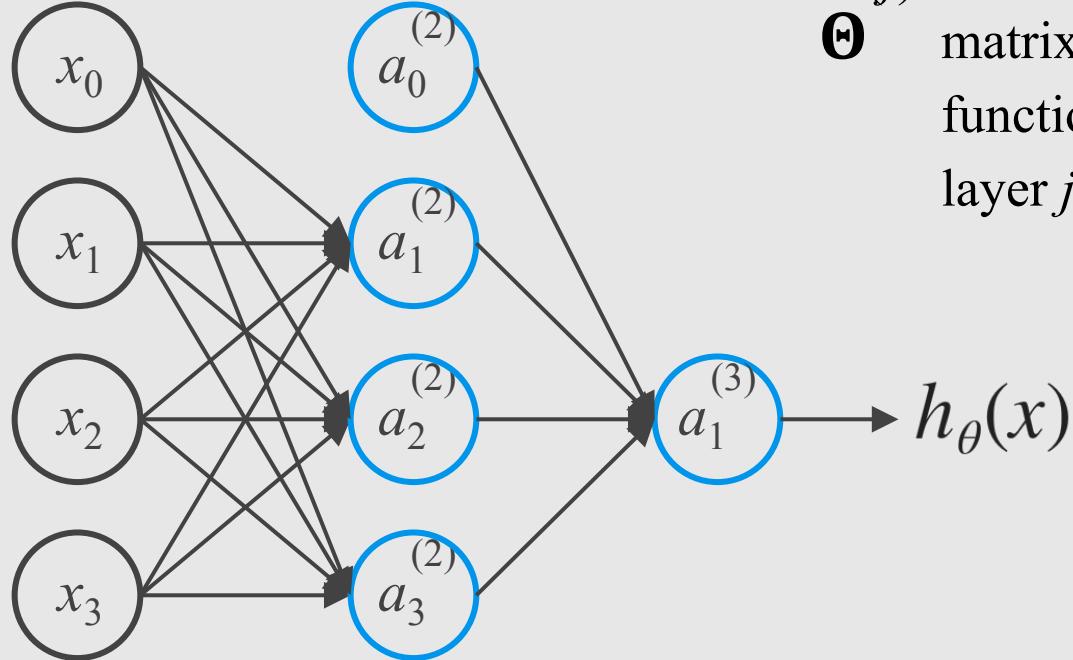


Layer 1 = Input layer

Layer 2 = Hidden layer

Layer 3 = Output layer

Neural Network



Layer 1

Layer 2

Layer 3

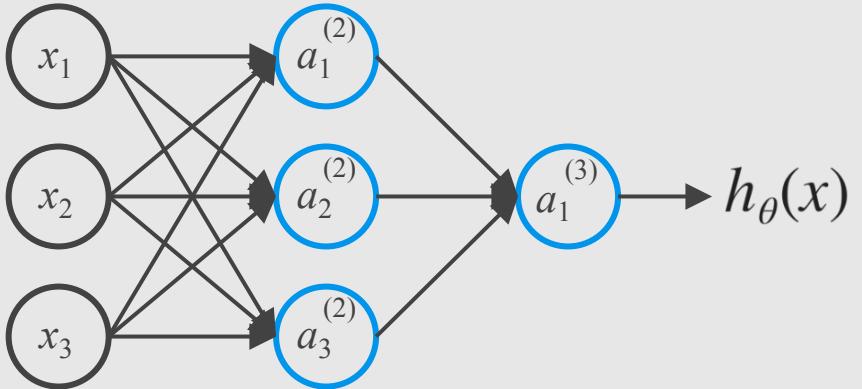
$a_i^{(j)}$

“activation” of unit i in layer j

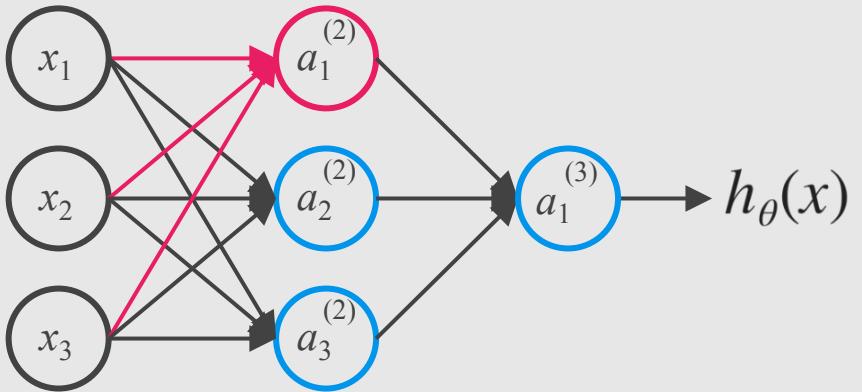
Θ

matrix of weights controlling
function mapping from layer j to
layer $j + 1$

$h_{\theta}(x)$

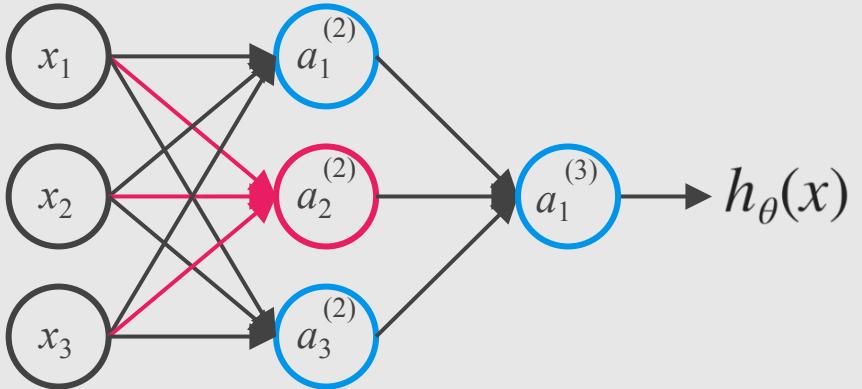


$a_i^{(j)}$ “activation” of unit i in layer j
 Θ matrix of weights controlling
function mapping from layer j to
layer $j + 1$



$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

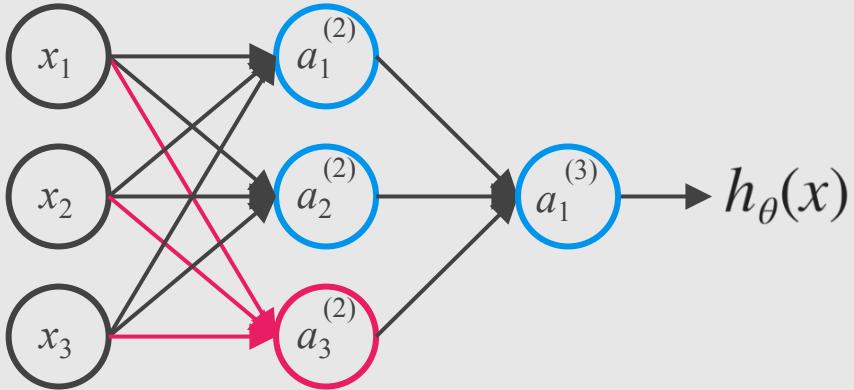
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$



$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

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

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

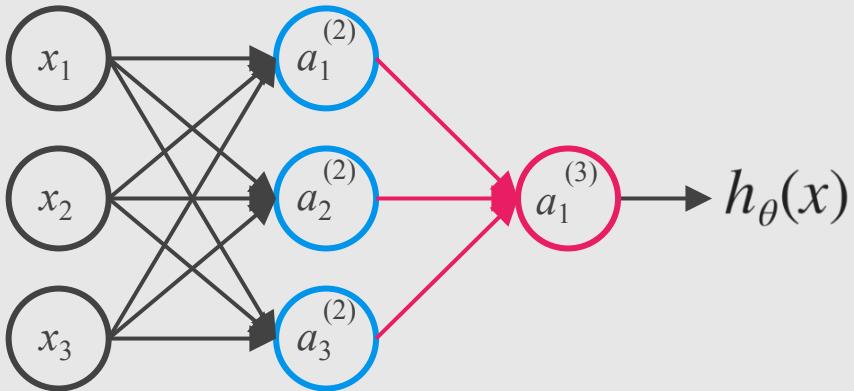


$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

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

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

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



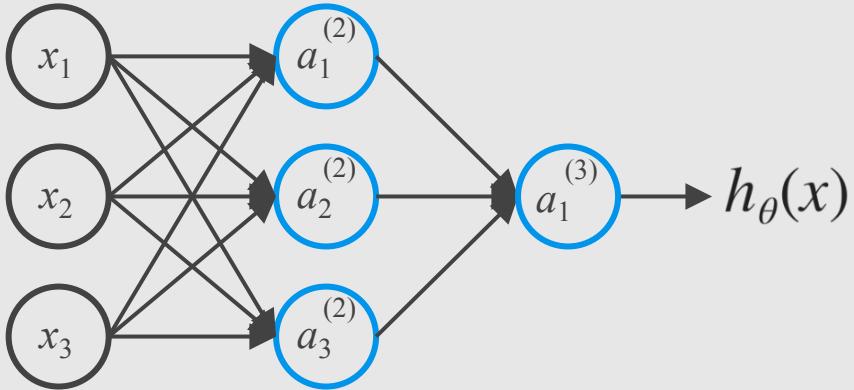
$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

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

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

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

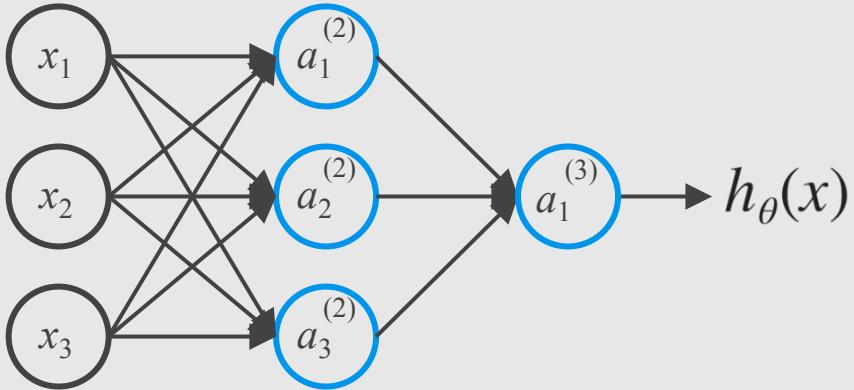
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$



$a_i^{(j)}$ “activation” of unit i in layer j
 Θ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

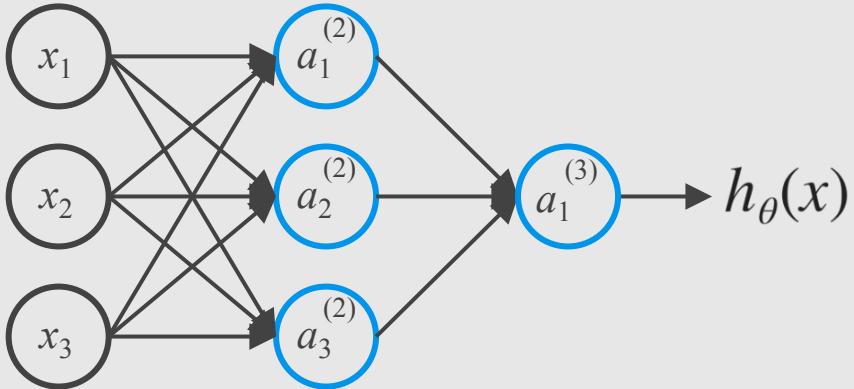
Feedforward Neural Network (forward propagating)

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



$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

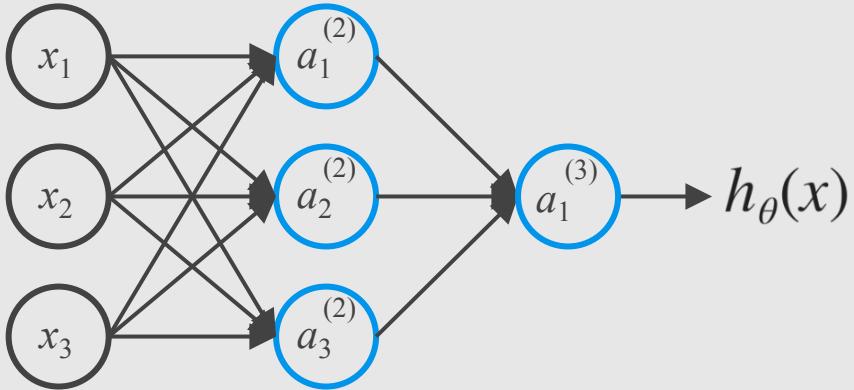
If network has s_j units in layer j , s_{j+1} units in layer $j+1$, then
 will be of dimension $\Theta^{(j)} s_{j+1} \times (s_j + 1)$



$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

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

E.g.: If layer 1 has 2 input nodes and layer 2 has 4 activation nodes,

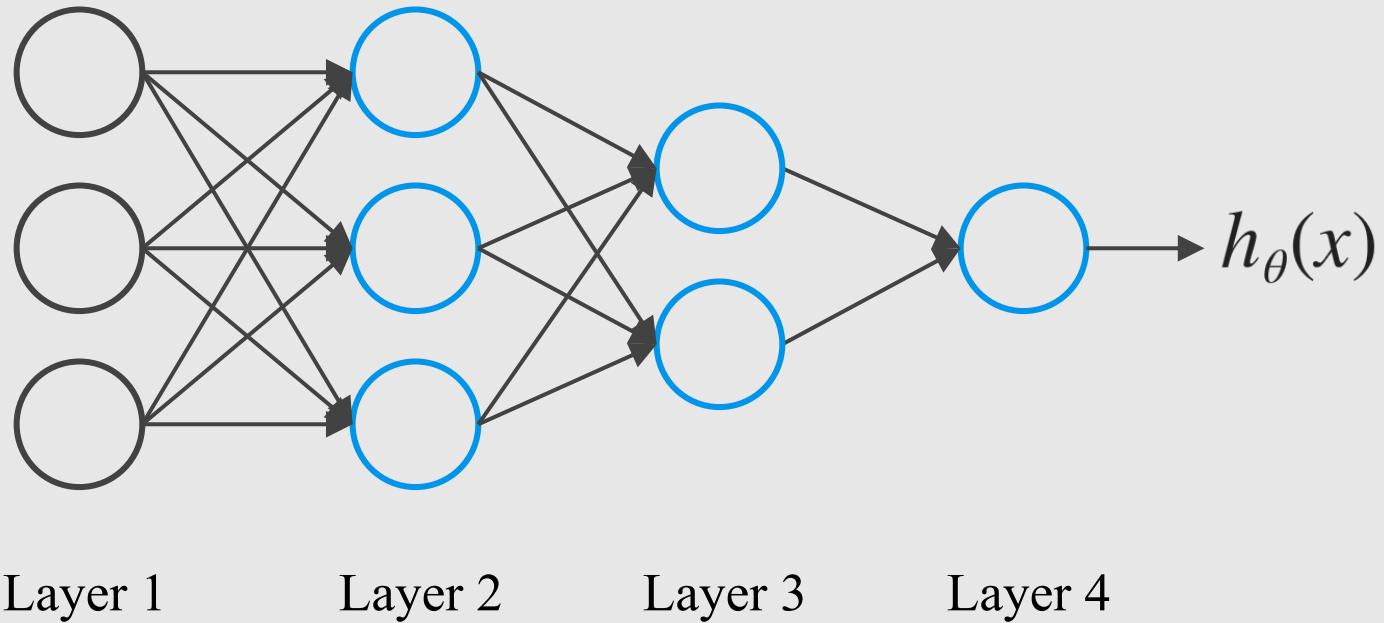


$a_i^{(j)}$ “activation” of unit i in layer j
 $\Theta^{(j)}$ matrix of weights controlling
 function mapping from layer j to
 layer $j + 1$

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

E.g.: If layer 1 has 2 input nodes and layer 2 has 4 activation nodes,
 dimension of $\Theta^{(1)}$ is going to be 4×3 where $S_j = 2$ and $S_{j+1} = 4$, so
 $S_{j+1} \times (S_j + 1) = 4 \times 3$.

Other Network Architectures

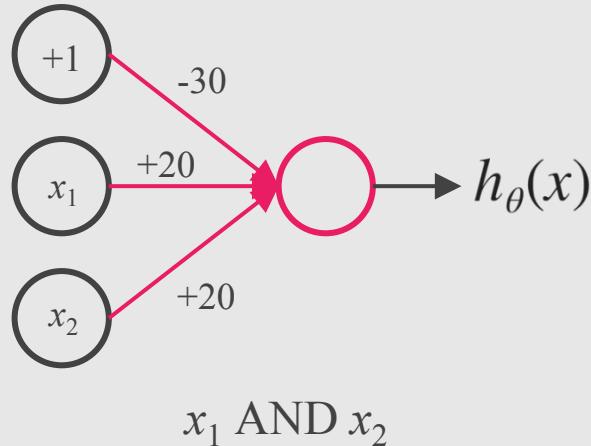


XNOR

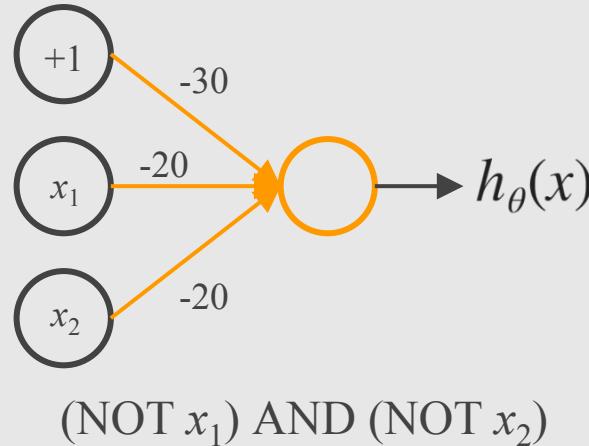
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ XNOR } x_2$$

XNOR

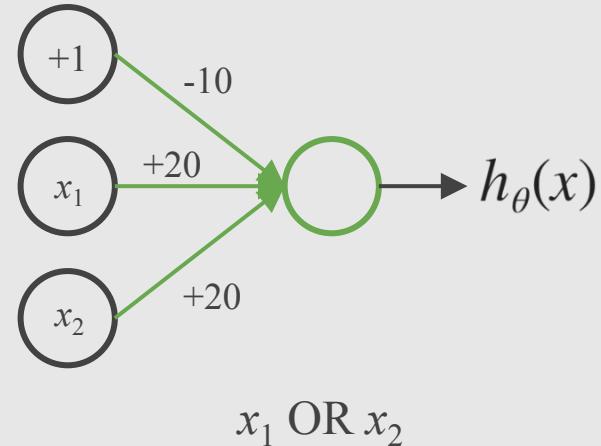
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ XNOR } x_2$$



x_1 AND x_2



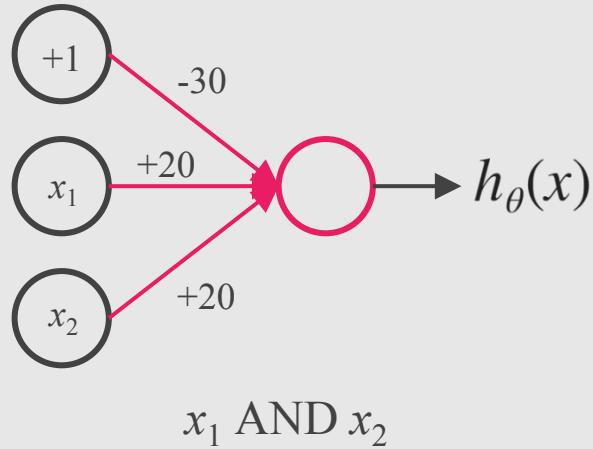
(NOT x_1) AND (NOT x_2)



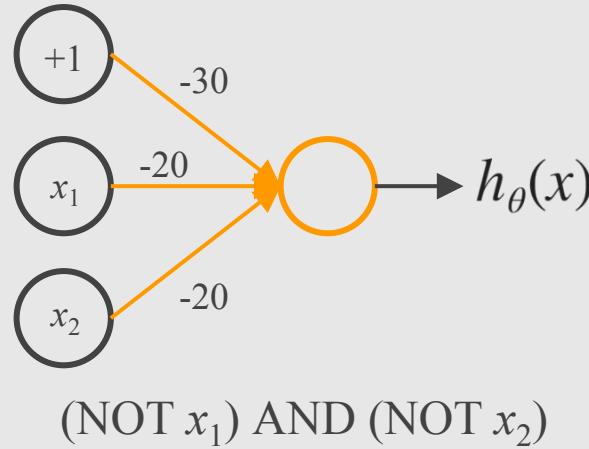
x_1 OR x_2

XNOR

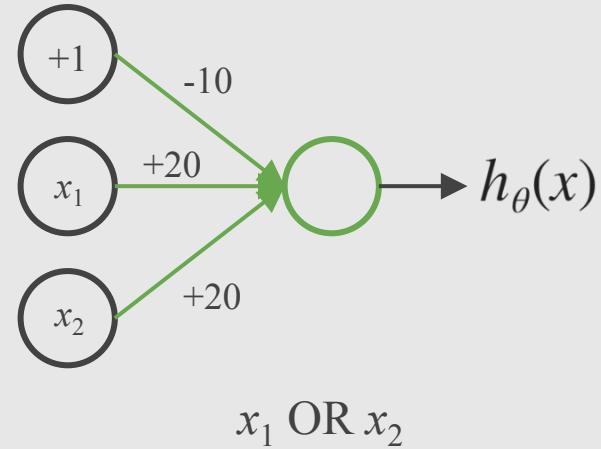
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ XNOR } x_2$$



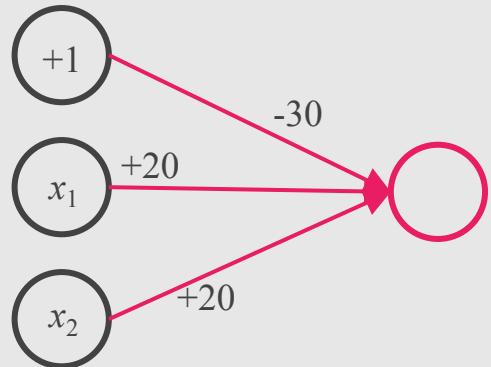
x_1 AND x_2



(NOT x_1) AND (NOT x_2)

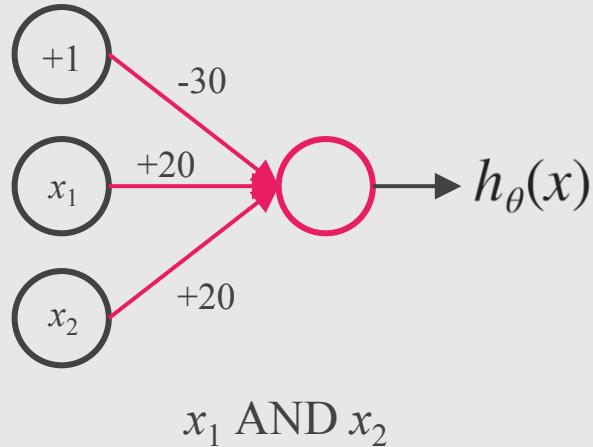


x_1 OR x_2

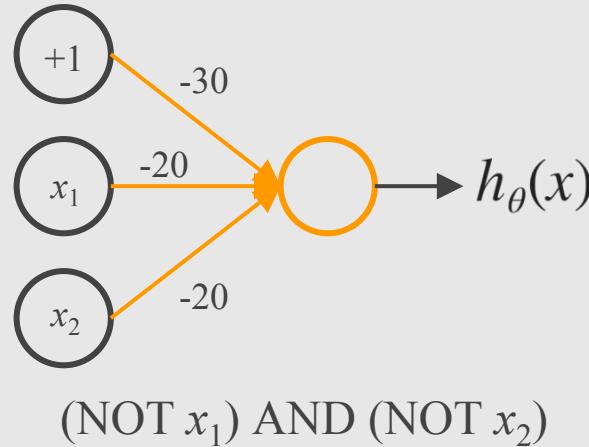


XNOR

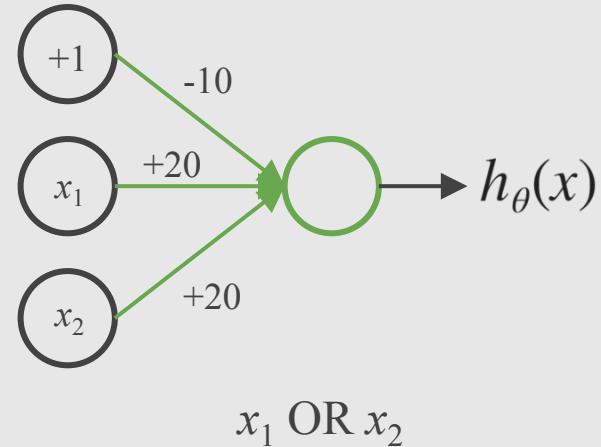
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ XNOR } x_2$$



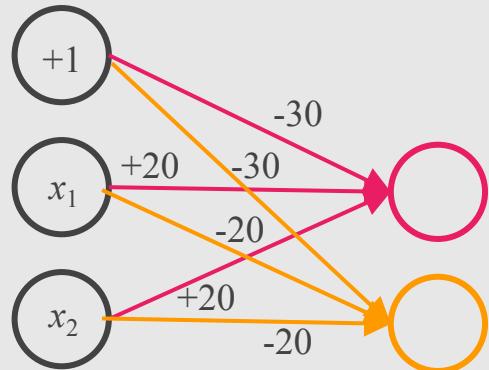
$x_1 \text{ AND } x_2$



(NOT x_1) AND (NOT x_2)

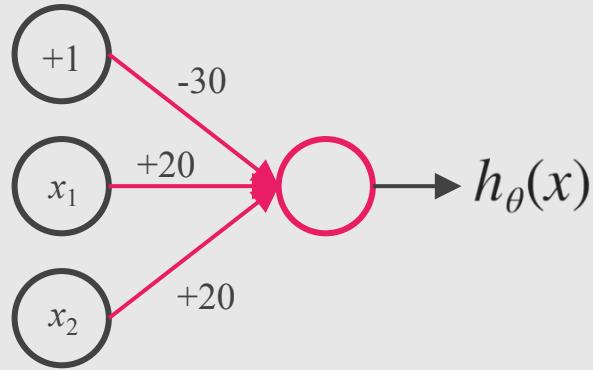


$x_1 \text{ OR } x_2$

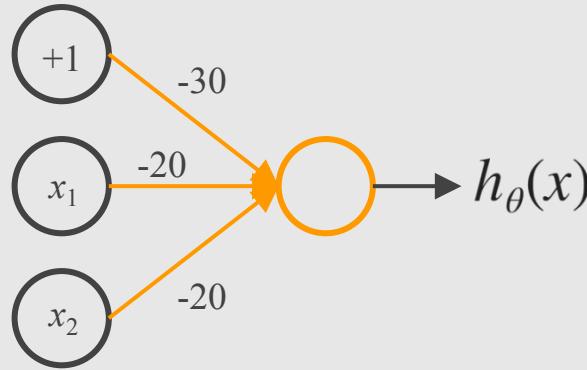


XNOR

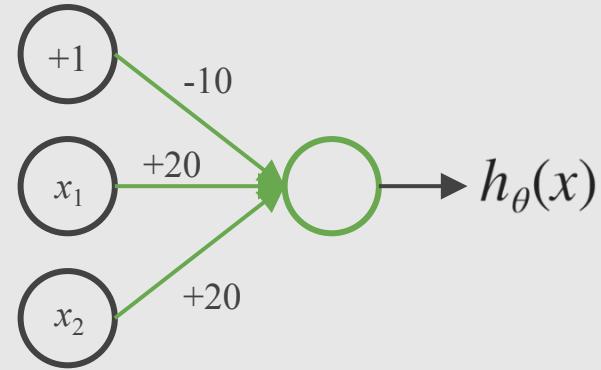
$$x_1, x_2 \in \{0,1\} \quad y = x_1 \text{ XNOR } x_2$$



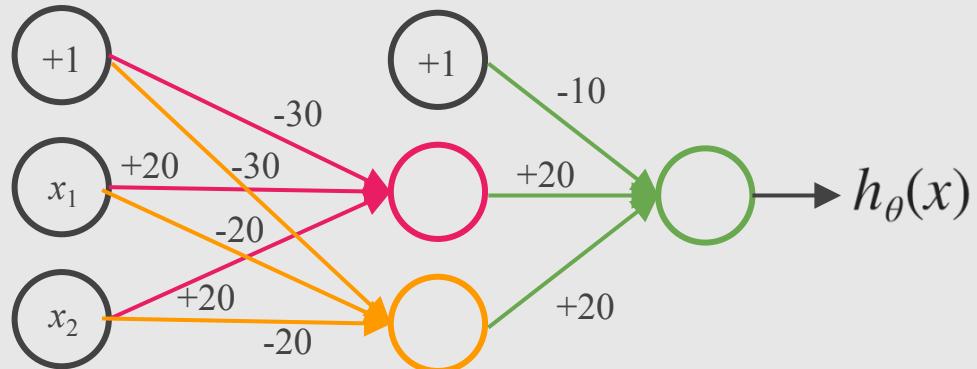
$x_1 \text{ AND } x_2$



(NOT x_1) AND (NOT x_2)

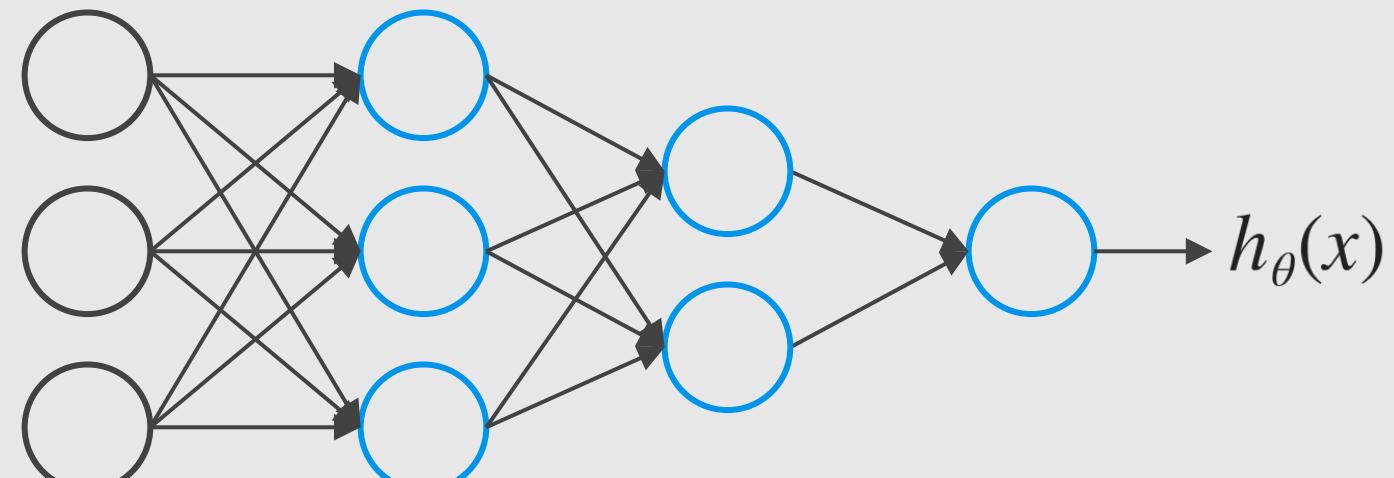


$x_1 \text{ OR } x_2$



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

Neural Network Intuition

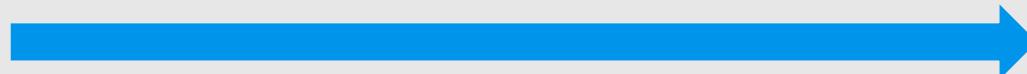


Layer 1

Layer 2

Layer 3

Layer 4

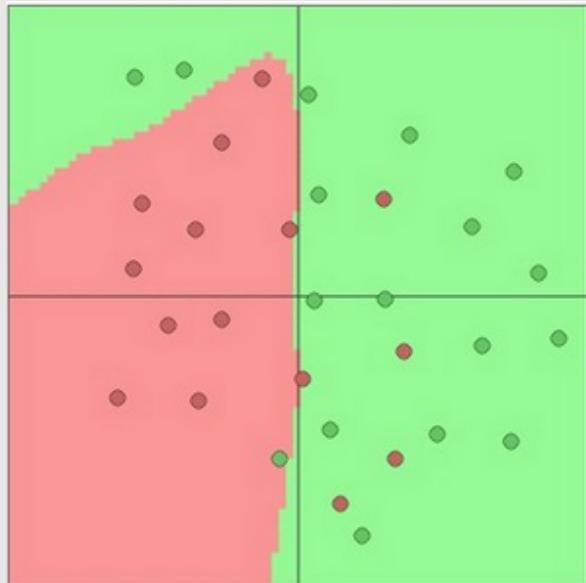


Complexity

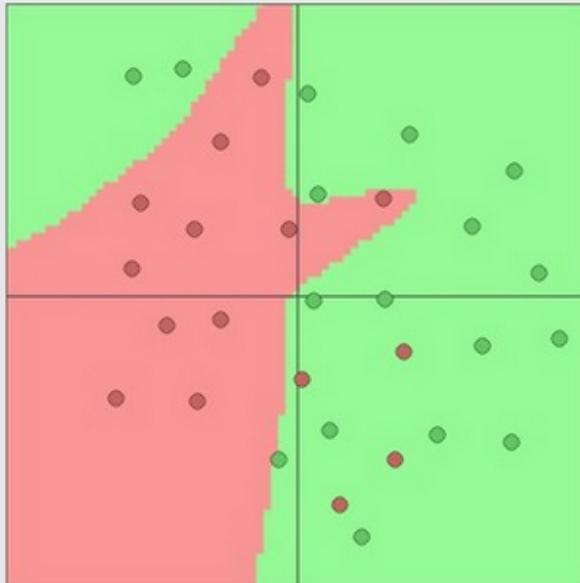
Neural Network Intuition

Toy 2d classification with 2-layer neural network

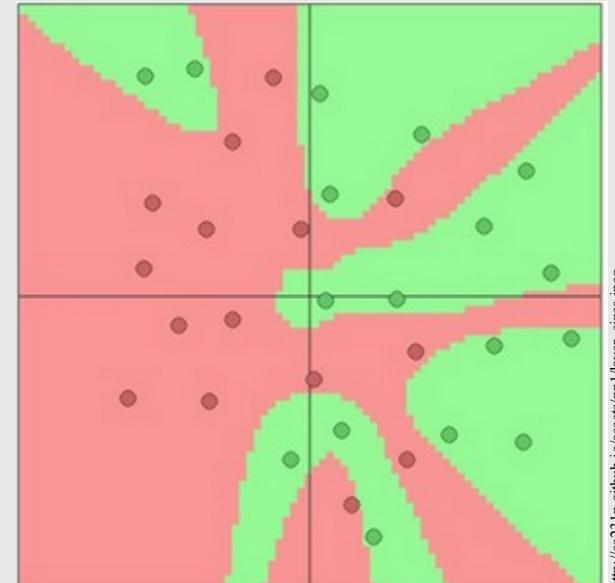
<http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>



3 hidden neurons

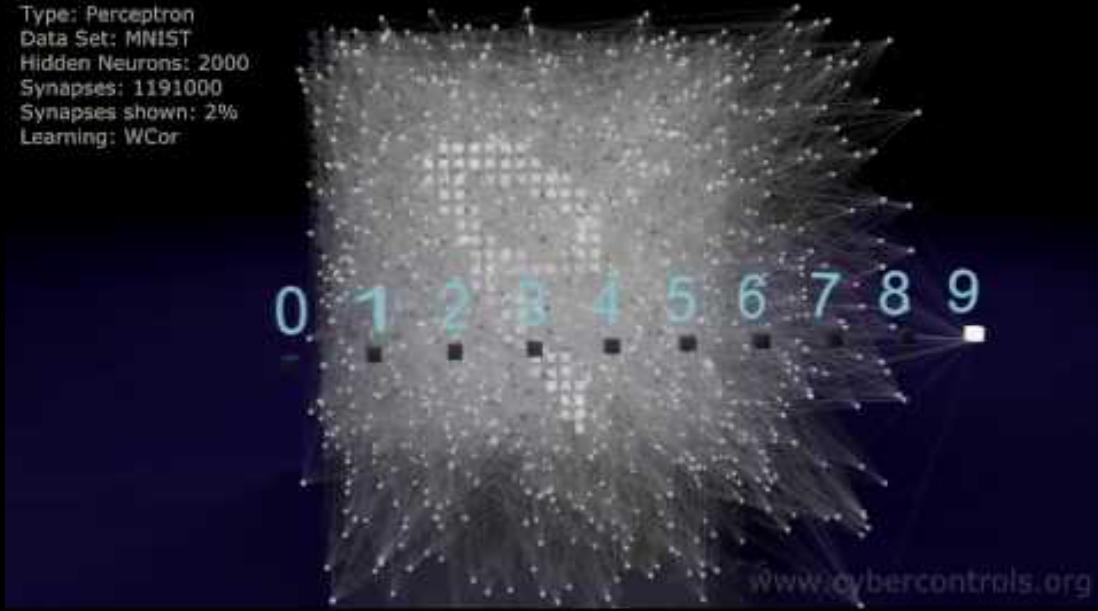


6 hidden neurons



20 hidden neurons

Type: Perceptron
Data Set: MNIST
Hidden Neurons: 2000
Synapses: 1191000
Synapses shown: 2%
Learning: WCor



<https://youtu.be/3JQ3hYko51Y>

Multi-class Classification



Cat



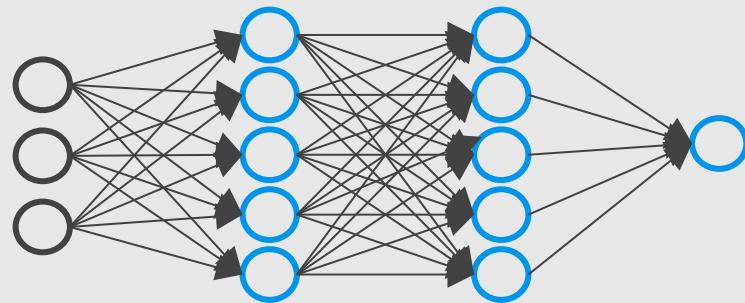
Dog



Frog



Car





Cat



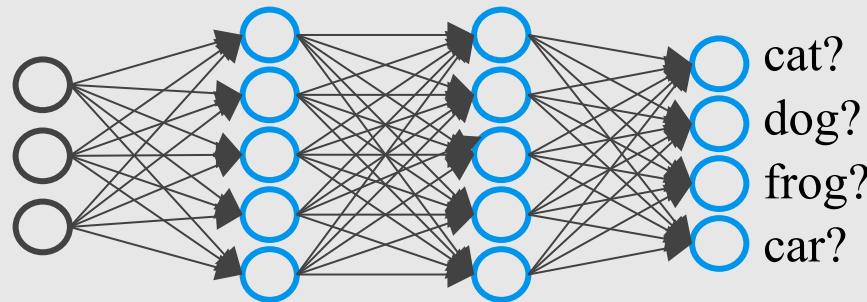
Dog



Frog



Car





Cat



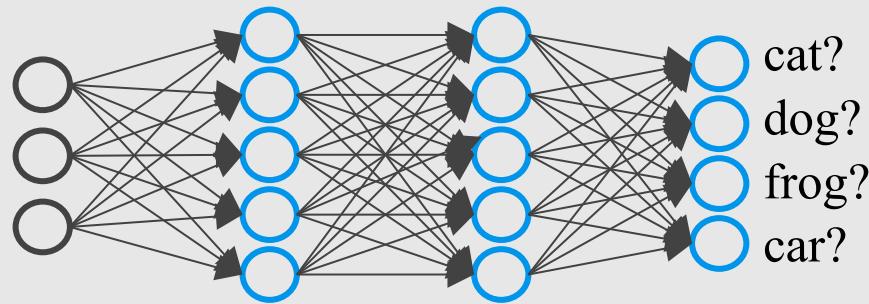
Dog



Frog



Car



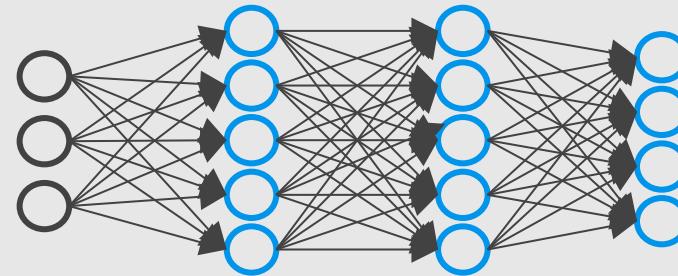
$$\text{Want } h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{when cat}$$

$$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{when dog}$$

$$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{when frog}$$

$$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{when car}$$

Multiple Output Units



Want $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ when cat $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when dog $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ when frog $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ when car

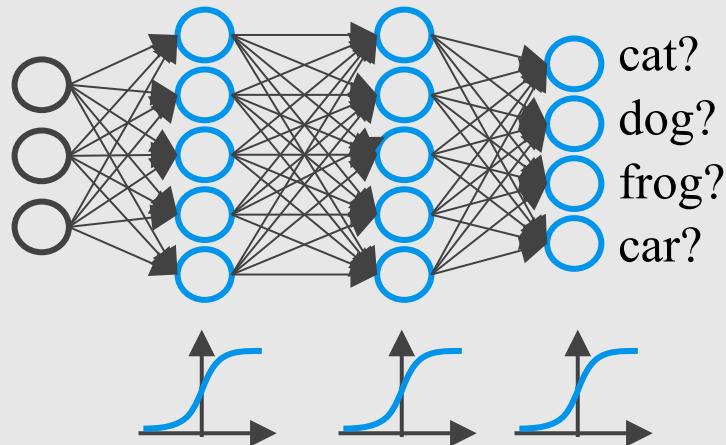
Training Set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

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

Softmax Classification

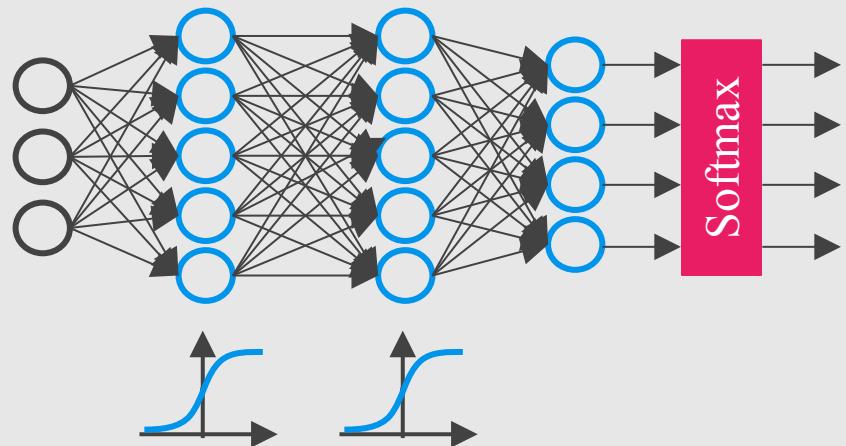
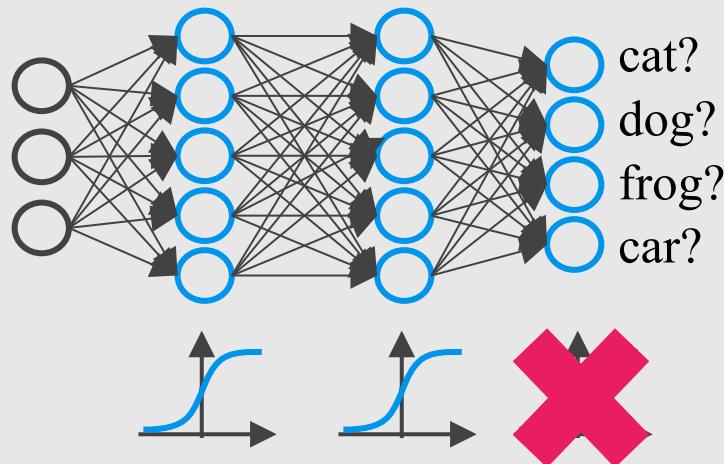
Softmax Classification

The **output layer** is typically modified by replacing the individual activation functions **by a shared softmax** function.



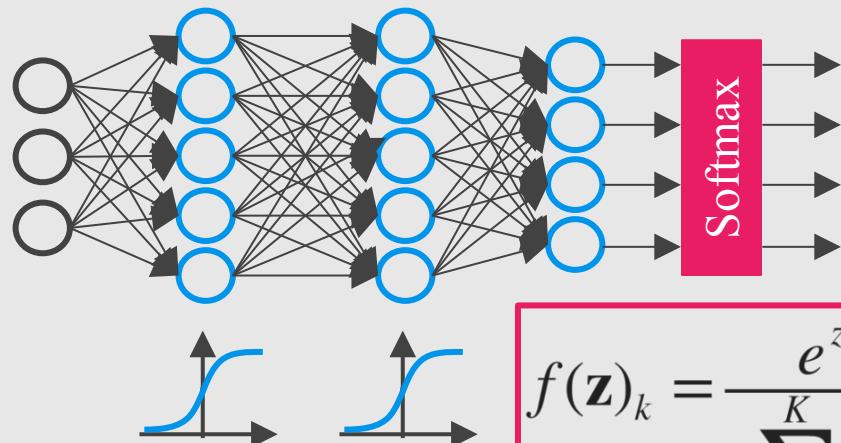
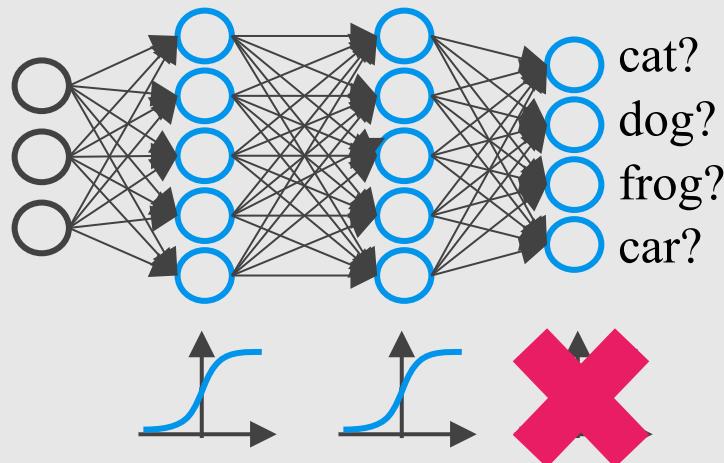
Softmax Classification

The **output layer** is typically modified by replacing the individual activation functions **by a shared softmax** function.



Softmax Classification

The **output layer** is typically modified by replacing the individual activation functions **by a shared softmax** function.



$$f(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$