

# An Introduction to AI

WEEK 5, COHORT 4



“aimed at getting you to  
kickass in AI”

# BUZZWORDS



Non-Linear hypothesis

Neuron

OR Gate

Perceptron

Higher Order features

XOR Gate

Cortex

One-hot encoding

AND Gate

LeNet

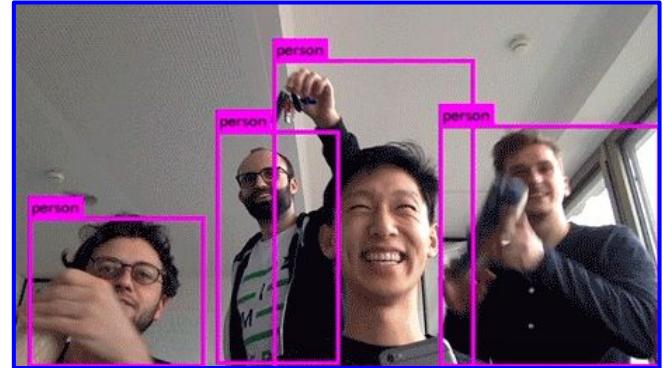
Activation function

Multi-classification

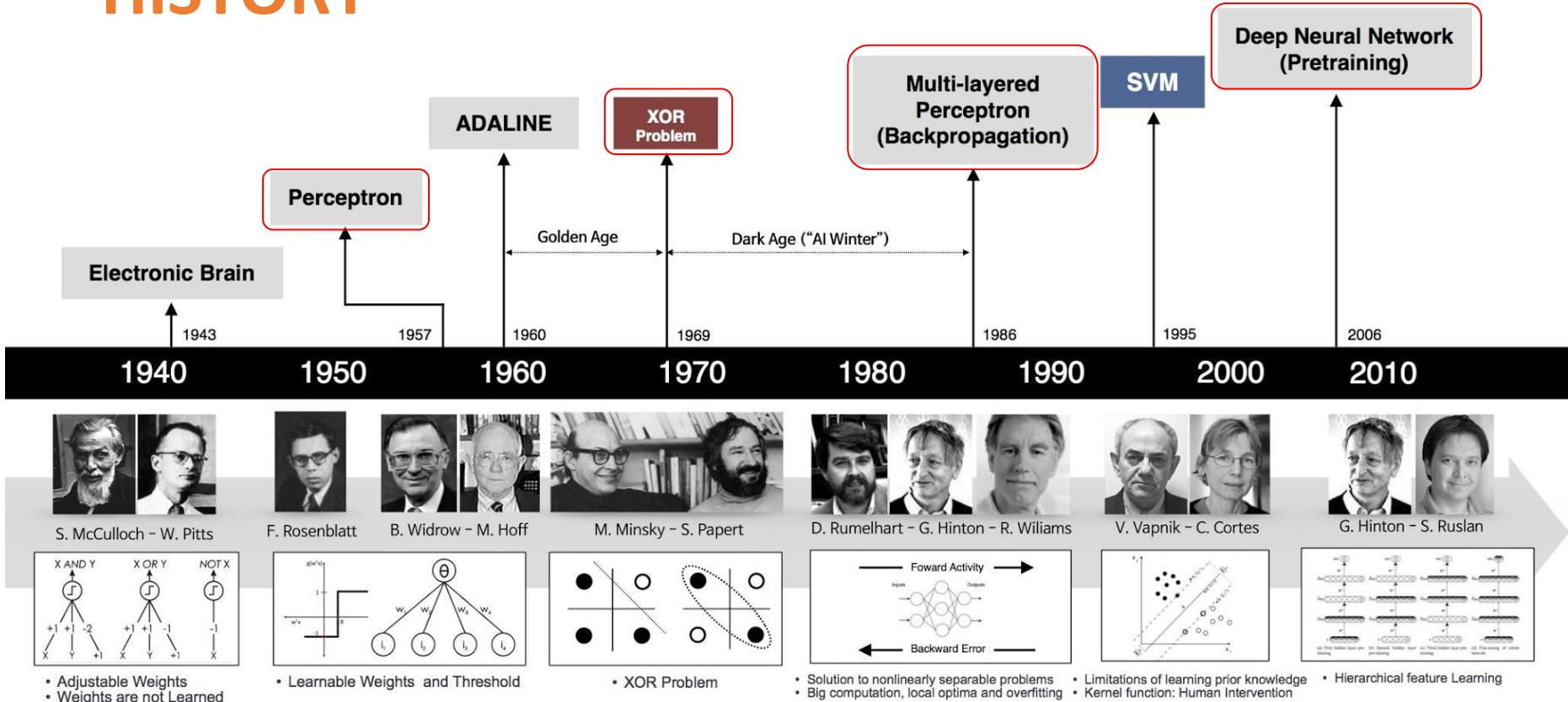
“One learning algorithm”



# What do they have in common?



# HISTORY



Documentary: The Rise of AI



1958 Perceptron

1974 Backpropagation



Convolution Neural Networks for  
Handwritten Recognition

1998



Google Brain Project on  
16k Cores

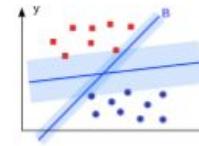
2012



1969  
Perceptron criticized



1995  
SVM reigns



2006  
Restricted  
Boltzmann  
Machine

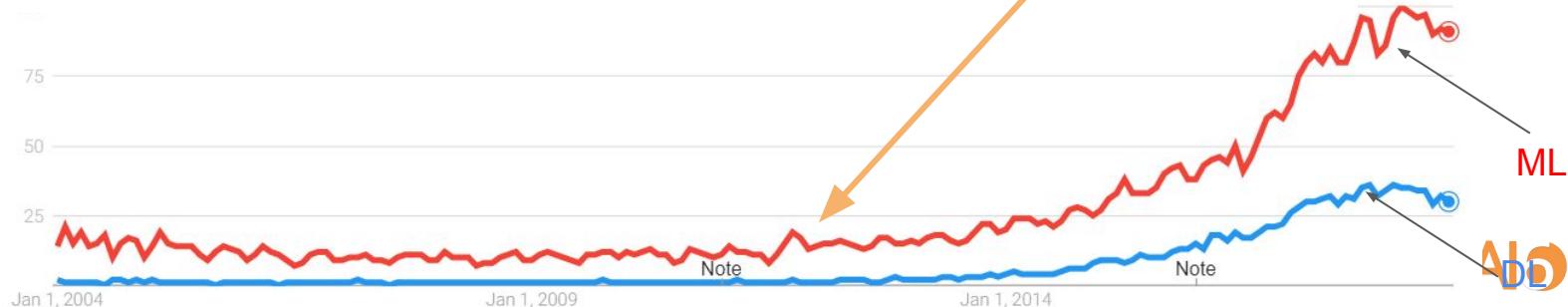


2012  
AlexNet wins  
ImageNet Classification with Deep Convolutional  
Neural Networks

Alex Krizhevsky  
University of Toronto  
kriz@cs.utoronto.ca

Ilya Sutskever  
University of Toronto  
ilya@cs.utoronto.ca

Geoffrey E. Hinton  
University of Toronto  
hinton@cs.utoronto.ca



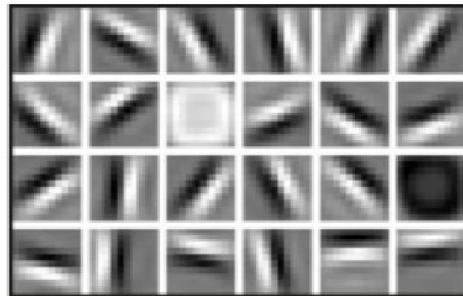
# Why Deep Learning?



Hand engineered features are time consuming, brittle and not scalable in practice

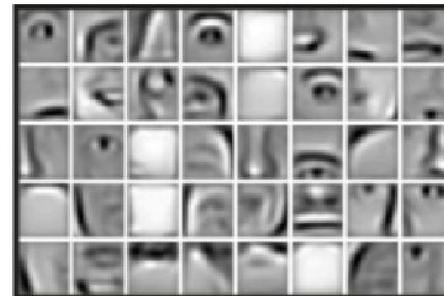
Can we learn the **underlying features** directly from data?

**Low Level Features**



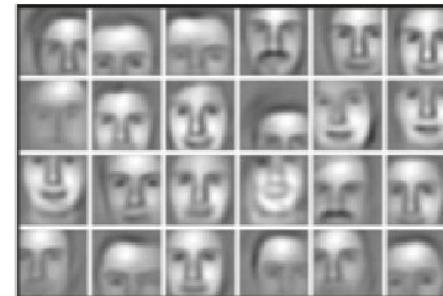
Lines & Edges

**Mid Level Features**



Eyes & Nose & Ears

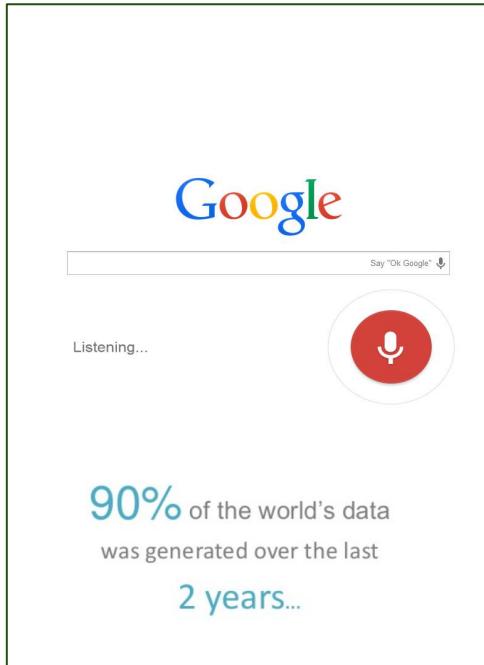
**High Level Features**



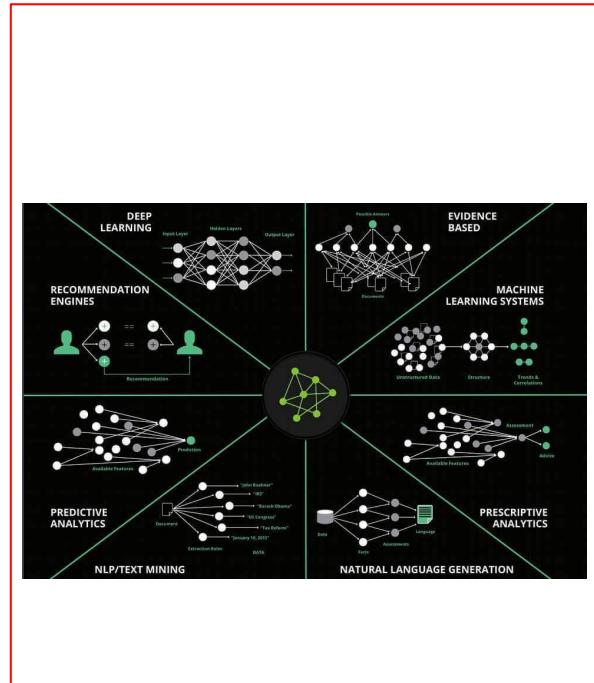
Facial Structure

# Why Deep Learning is having a great impact in the world?

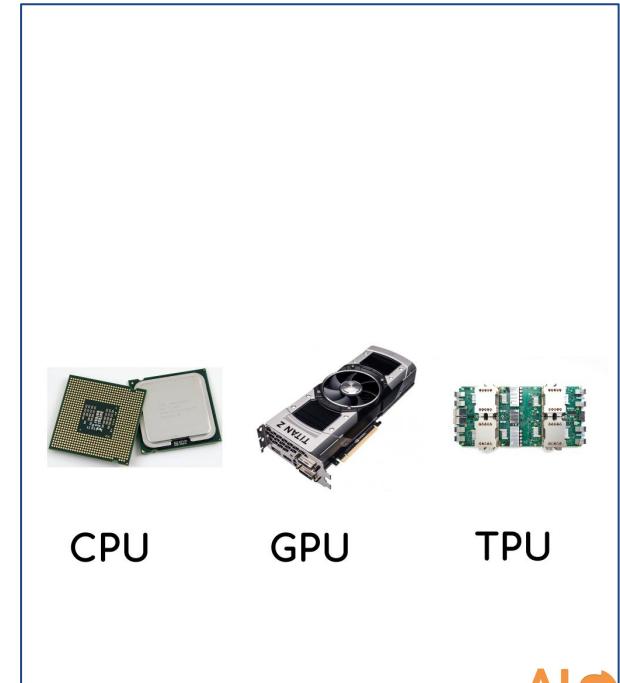
## Massive Data



## Modern Algorithms



## Computational Powerhouse





Soumith Chintala

@soumithchintala

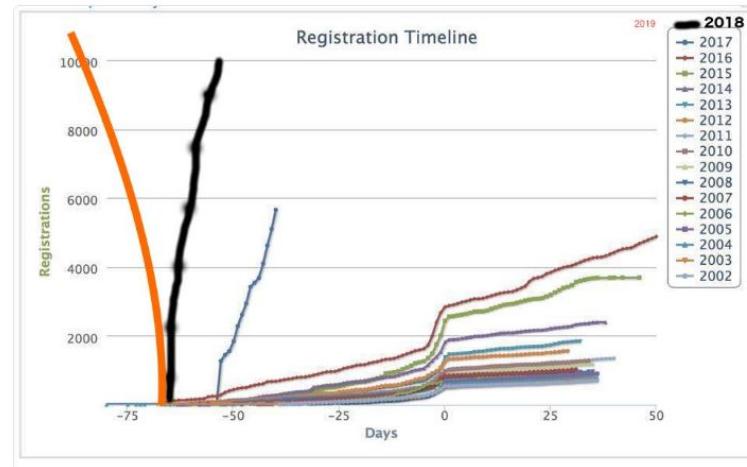
Following



NIPS Conference Registrations 2002 thru 2019.

[2018] War erupts for tickets

[2019] AI researchers discover time travel



6:13 AM · 16 Sep 2017



NIPS @NipsConference · 18 Sep 2017

Replies to @soumithchintala @mikiobraun

The singularity happens in late 2018



1



8



44



Julia Schnabel @ja\_schnabel · Sep 4

Replies to @NipsConference



NIPS

@NipsConference

Follow



#NIPS2018 The main conference sold out in 11 minutes 38 seconds

8:17 AM · 4 Sep 2018

694 Retweets 1,058 Likes



\*Neural Information Processing Systems



**Yann LeCun** @ylecun · 17 Sep 2017

Replying to [@soumithchintala](#)

Registrations for NIPS 2019 will be full before NIPS 2018 takes place.



3



15



161



**Karan Desai** @kdexd · 17 Sep 2017

I'm thinking of getting my son registered for NIPS 2040. Once I get a confirmation I will start searching for a wife.



1



17



234



**Miles Obare** @bdhobare · Sep 4

Replying to [@NipsConference](#)

wow! Coming from Africa where the internet takes those 11mins to get to the payment page, i seriously feel depressed? How are we supposed to even have a chance!



1



4



67



**@micahstubbs** @micahstubbs · Sep 4

special workshop tickets



1



1



**Rediet Abebe** @red\_abeb · Sep 4

The special workshop tickets we got for Black in AI don't even cover half of the people who submitted papers



1



10



# WHY REPRESENTATION MATTERS?



**Rediet Abebe** \*

@red\_abeb

[Follow](#)

If you have a free **#NIPS2018** ticket as AC/top reviewer, please consider transferring your ticket to [@black\\_in\\_ai](#). Many members couldn't register in time due to slow Internet connection etc, and we'd really like to have them at the conference. If you're interested, see form below:

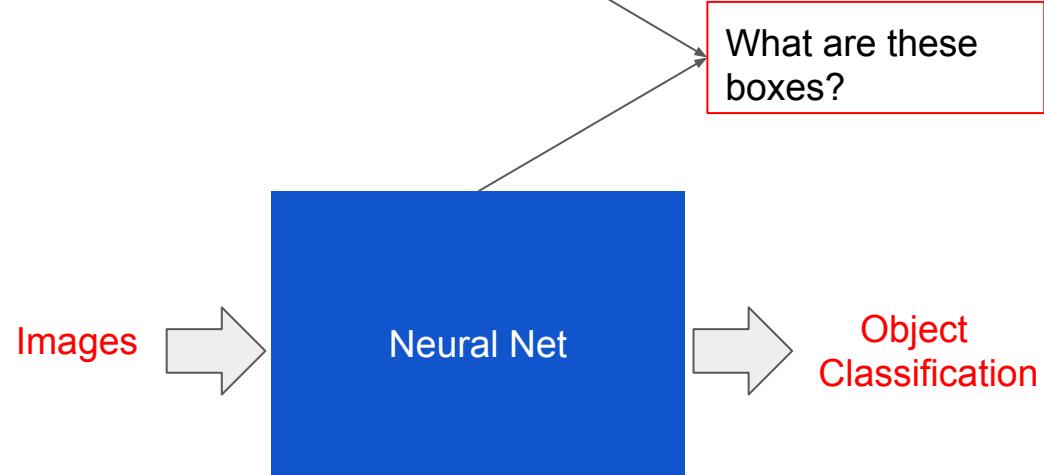
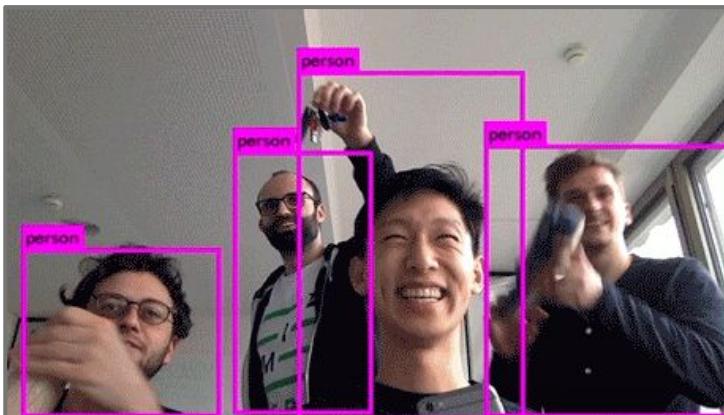
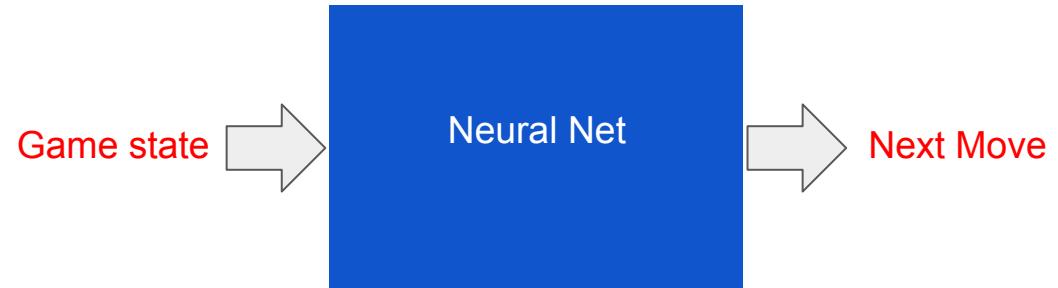
3:29 PM - 4 Sep 2018

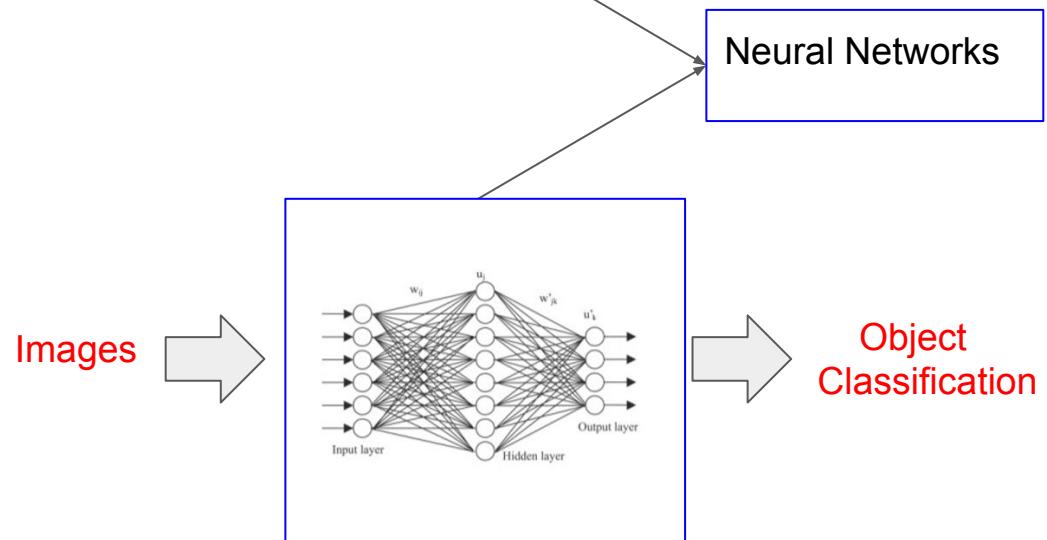
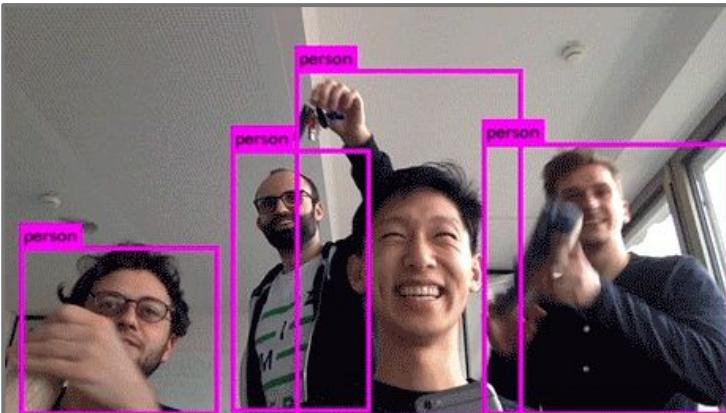
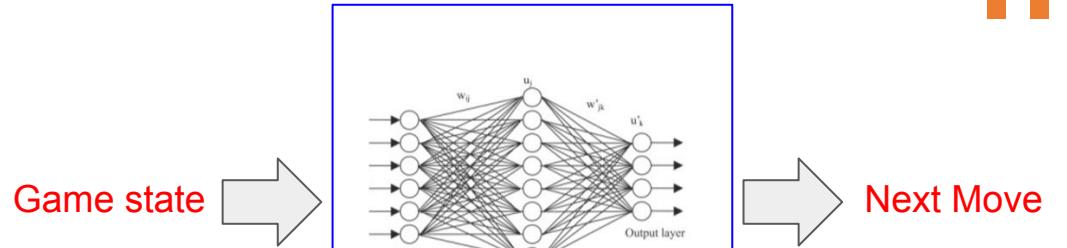
84 Retweets 151 Likes

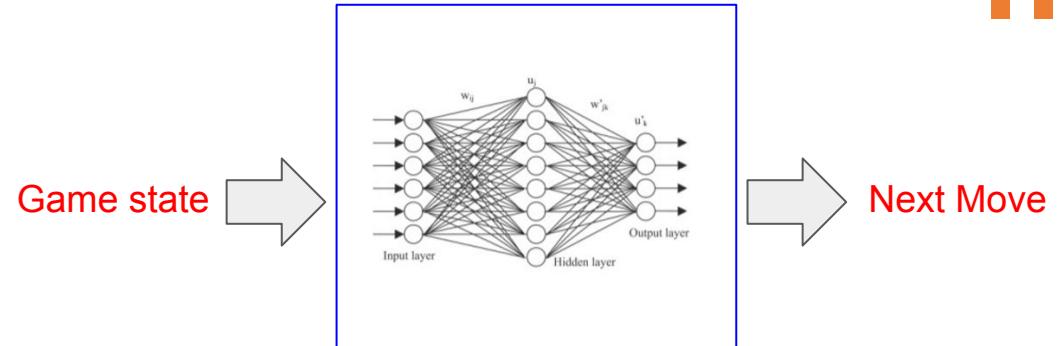


\*My AI crush

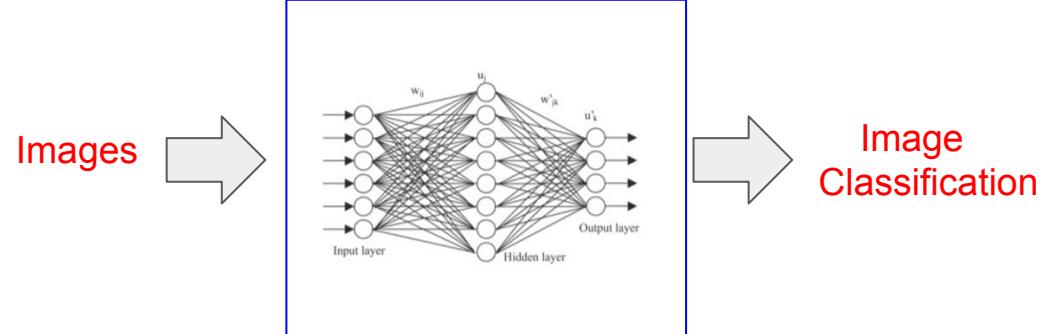
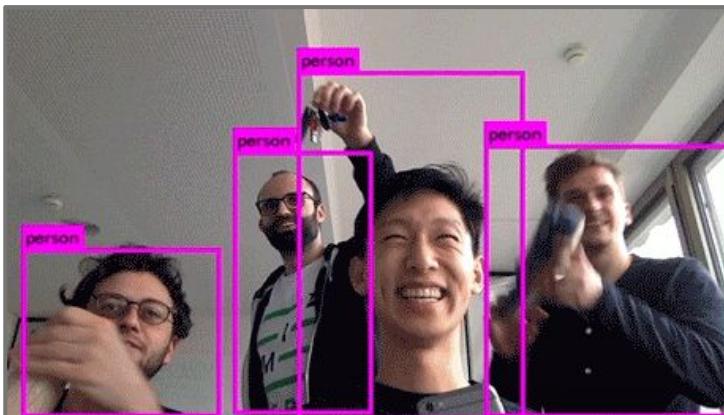
AI



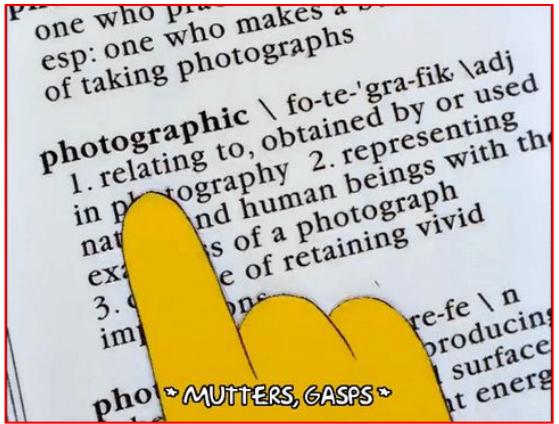




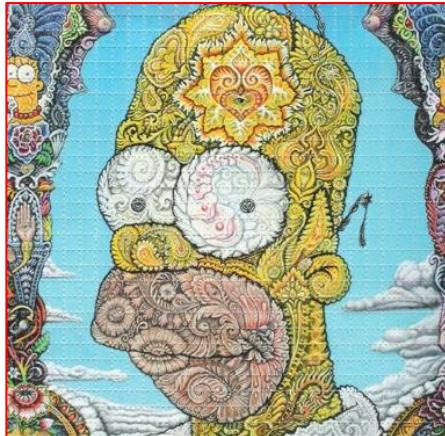
Each of these boxes is actually a function – E.g  
f: Image = Image classification



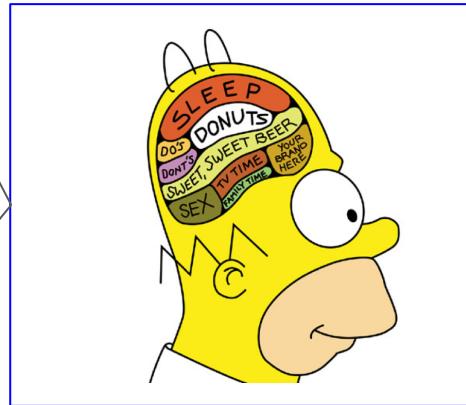
- It can be approximated by a neural network



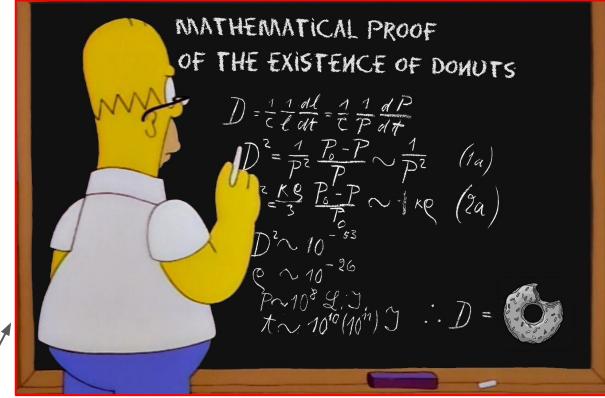
- Learn



- Recognize pattern



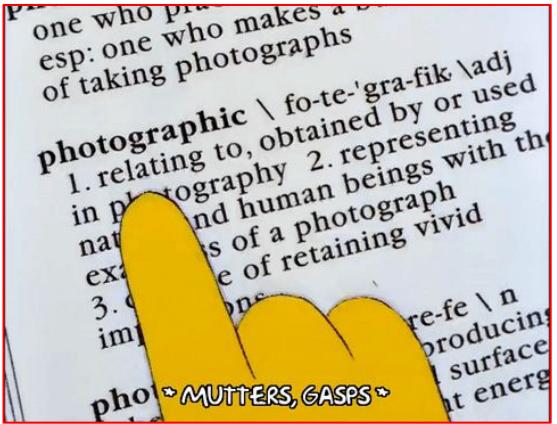
All these are powered by the BRAIN.



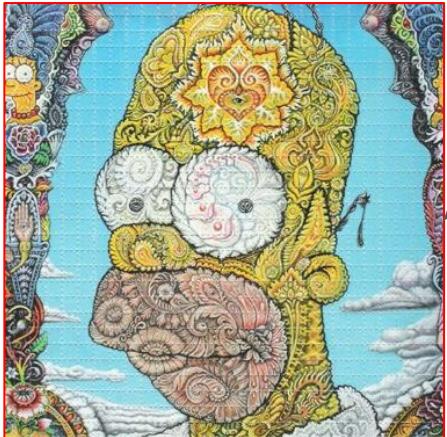
- Solve problem



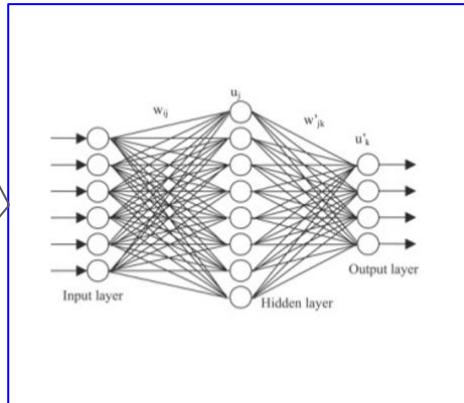
- Create stuffs



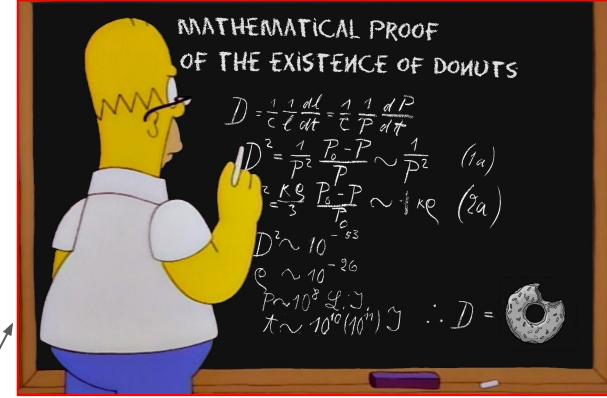
- Learn



- Recognize pattern



What can you do with Neural Network?

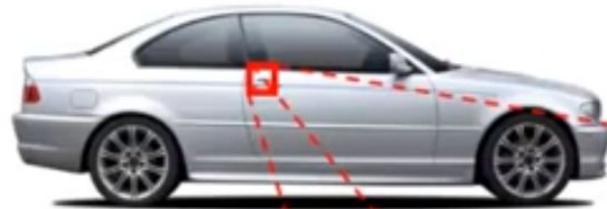


- Solve problem



- Create stuffs

You see this:



Semantic Gap

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

Why computer  
Vision is hard?

# Computer Vision: Car detection



Cars



Not a car

Testing:



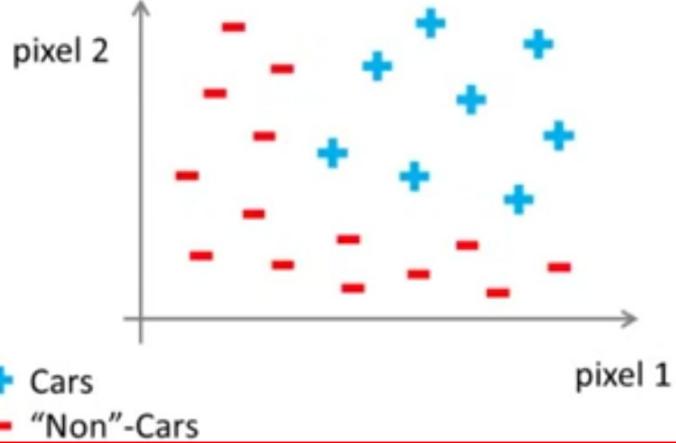
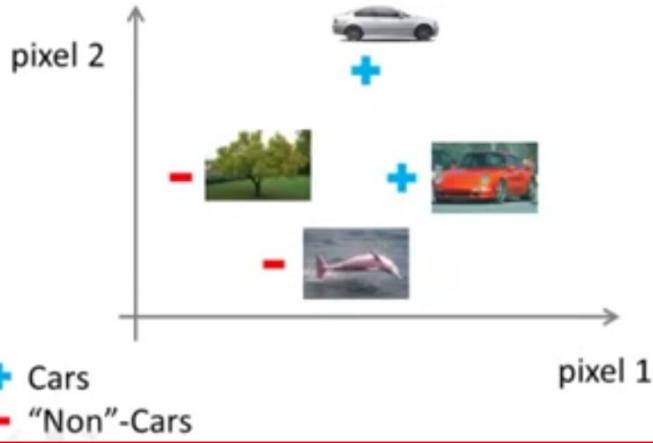
What is this?

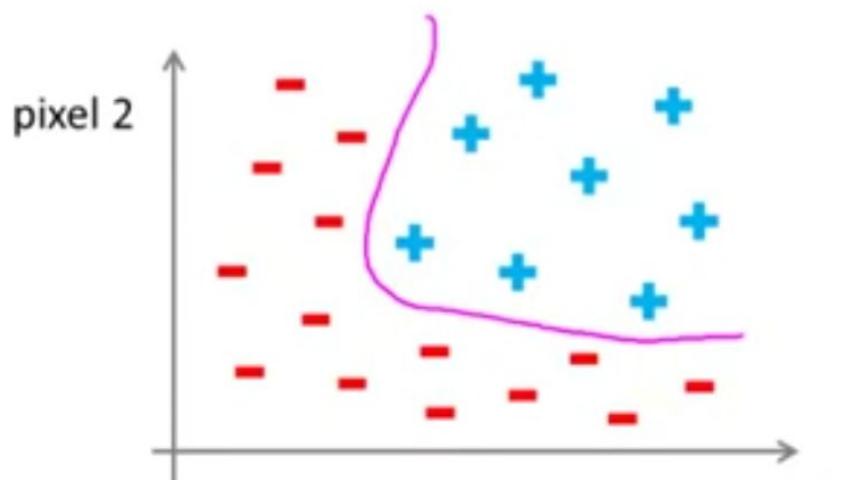
pixel 1



pixel 2

Learning  
Algorithm





+
 Cars  
 -
 "Non"-Cars

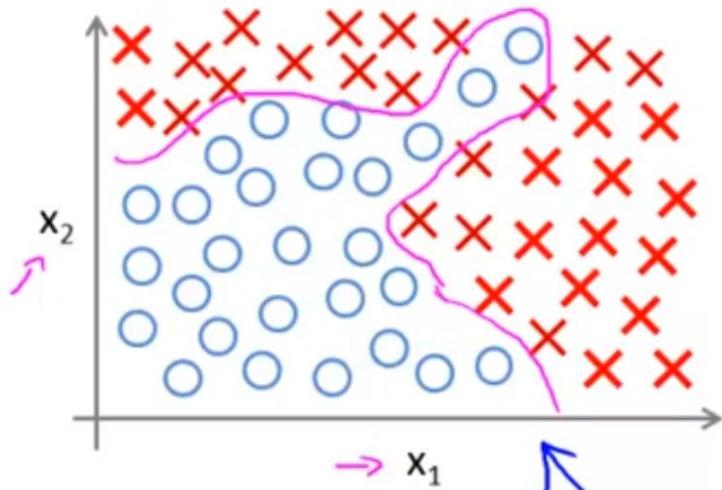
50 x 50 pixel images  $\rightarrow$  2500 pixels  
 $n = 2500$  (7500 if RGB)

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

0 - 255

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

## Non-linear Classification



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

$\left. \begin{array}{c} \\ \\ \\ \\ \end{array} \right\} h=100$

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

$$\rightarrow \underline{x_1^2}, \underline{x_1 x_2}, \underline{x_1 x_3}, \underline{x_1 x_4} \dots \underline{x_1 x_{100}}$$
$$\underline{x_2^2}, \underline{x_2 x_3} \dots$$

$\approx 5000$  feature  $\delta(n^2)$

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

$$\rightarrow \underline{x_1 x_2 x_3}, \underline{x_1^2 x_2}, \underline{x_{10} x_{11} x_{12}}, \dots$$

$O(n^3)$

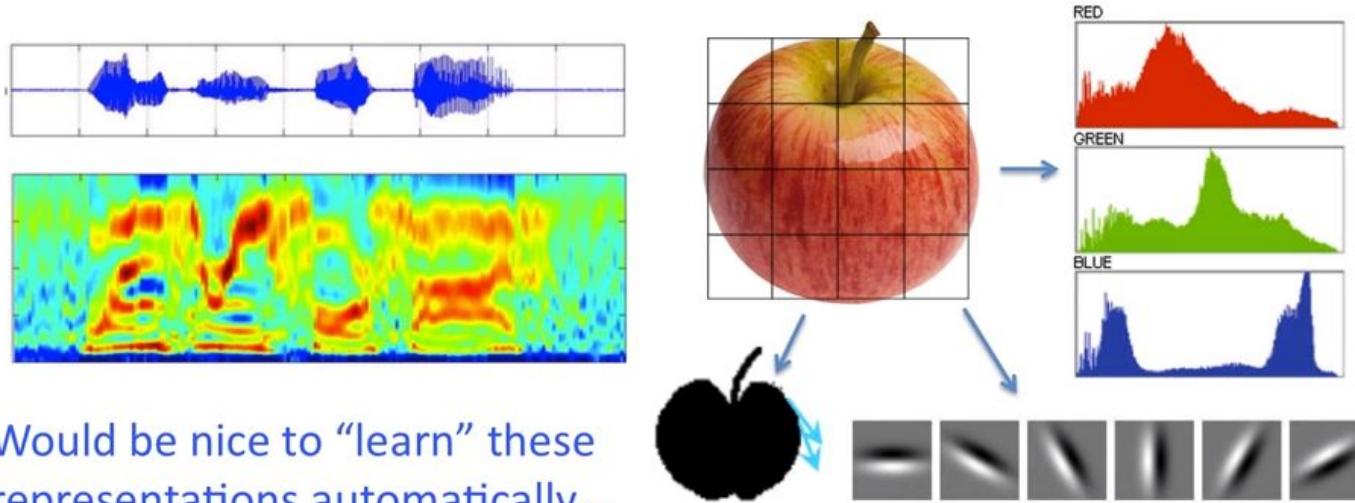
$170,000$

$\cancel{\frac{n^2}{2}}$   $\cancel{10}$

- Higher order poly. features - Overfitting and Computationally expensive

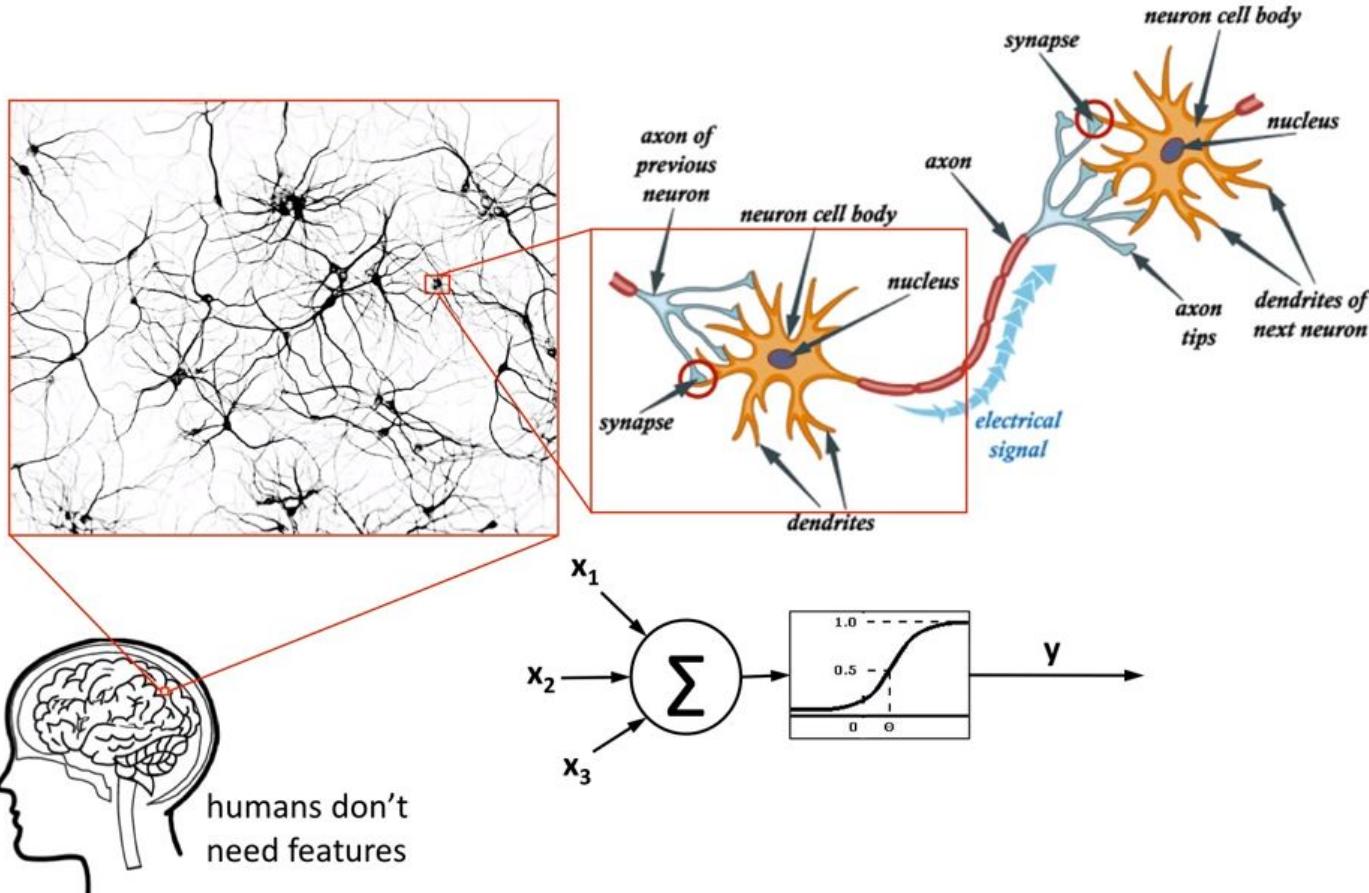
# Features in Machine Learning

- 1-line summary of ML:  $y = \text{sgn}(\mathbf{w}^T \mathbf{x} + b)$ 
  - SVM can learn very effective weights  $\mathbf{w}$
  - ... if you use the right representation  $\mathbf{x}$



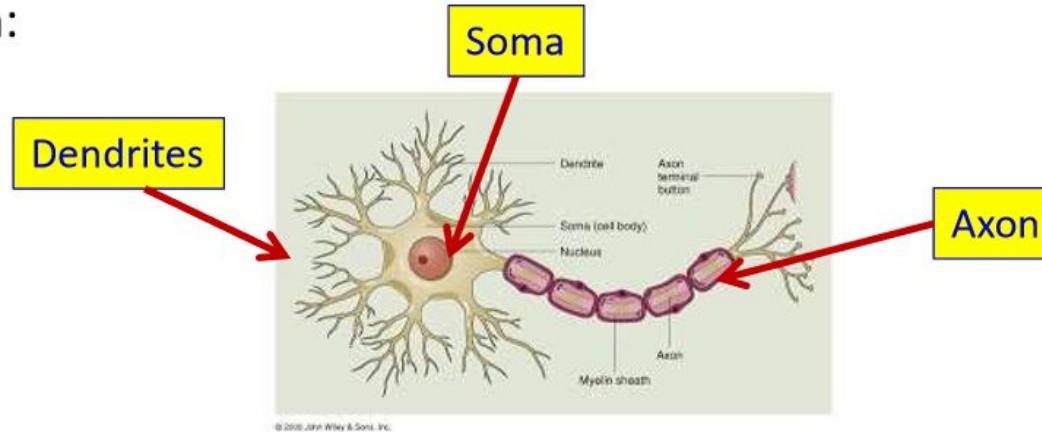


# Neurons and the brain



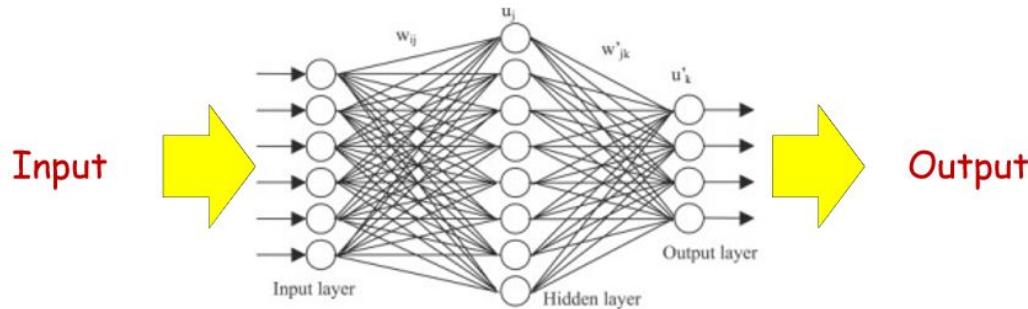
# Biological Inspiration

- What are the units?
- A neuron:



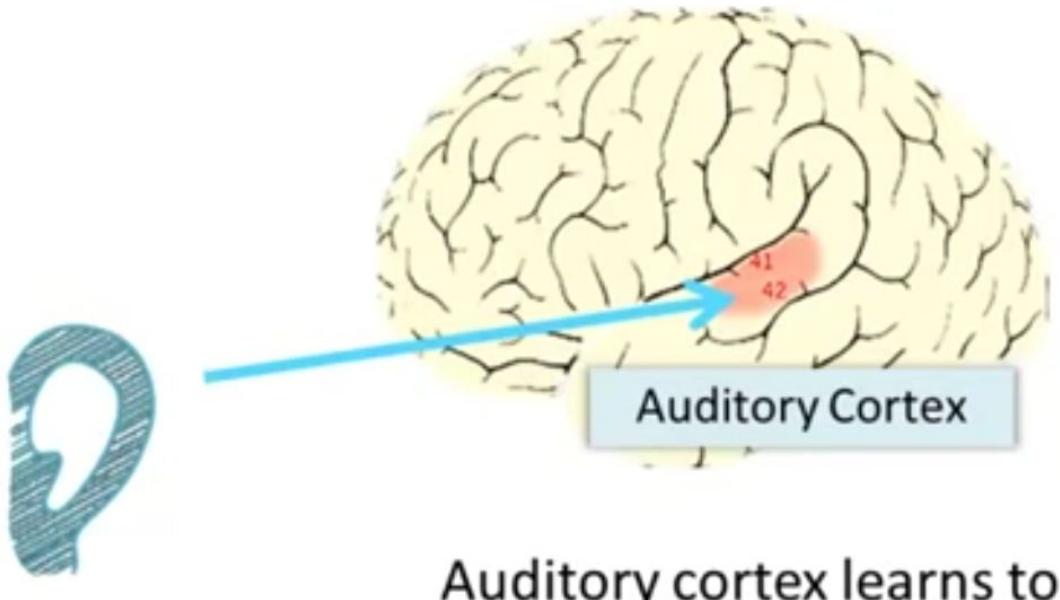
- Signals come in through the dendrites into the Soma
- A signal goes out via the axon to other neurons
- Algorithms that try to mimic the brain. Was very widely used in 80s and early 90s; popularity diminished in late 90s
- Recent resurgence: State-of-the-art for many applications

# The network as a function

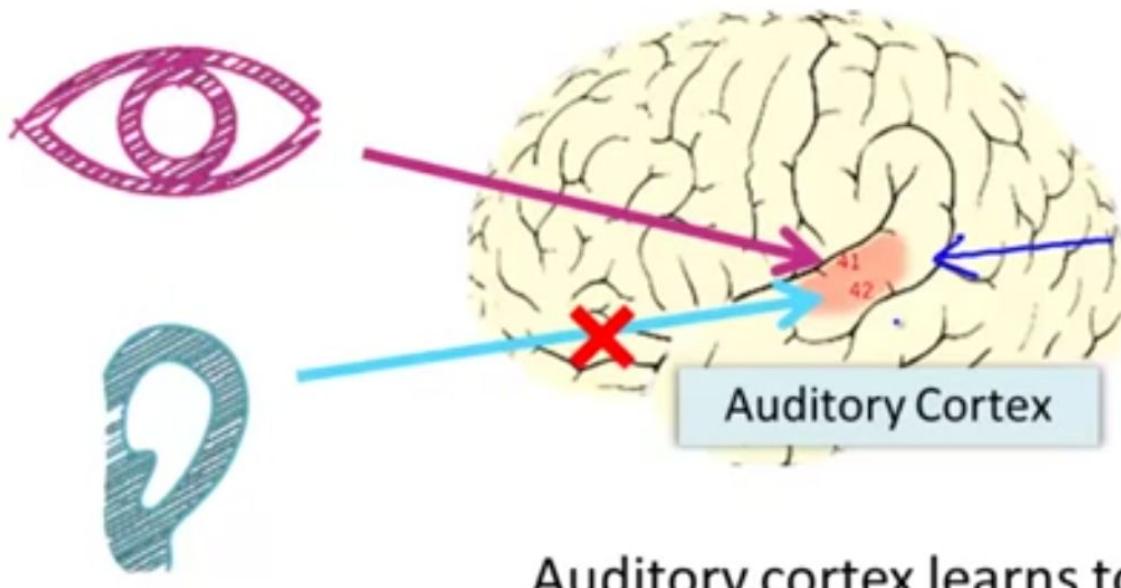


- Inputs are numeric vectors
  - Numeric representation of input, e.g. audio, image, game state, etc.
- Outputs are numeric scalars or vectors
  - Numeric “encoding” of output from which actual output can be derived
  - E.g. a score, which can be compared to a threshold to decide if the input is a face or not
  - Output may be multi-dimensional, if task requires it

## The “one learning algorithm” hypothesis

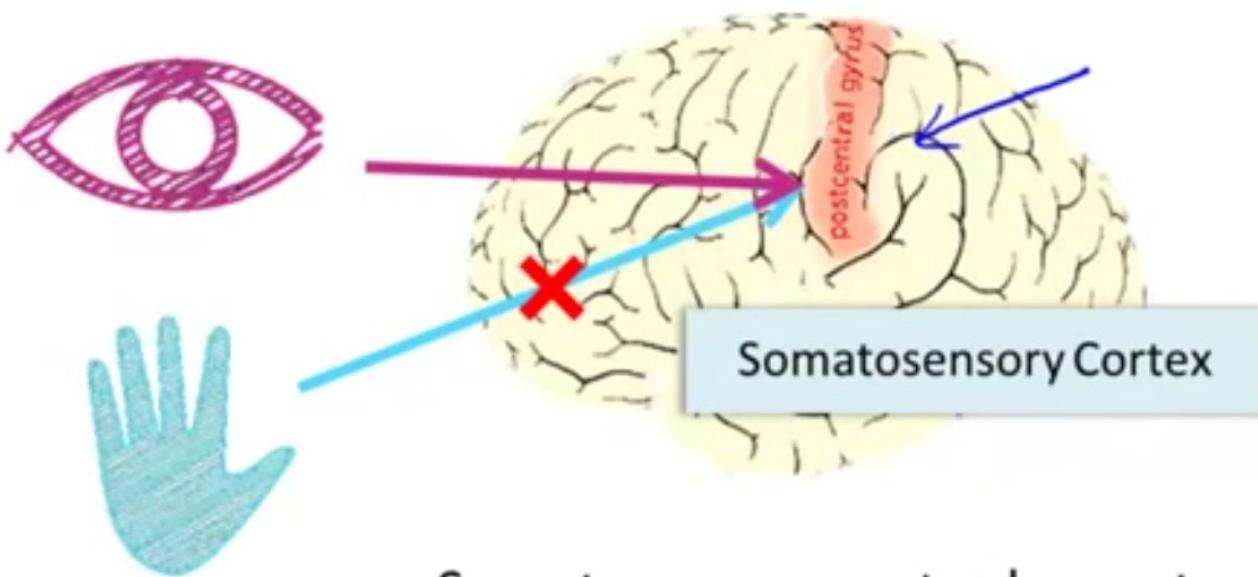


## The “one learning algorithm” hypothesis



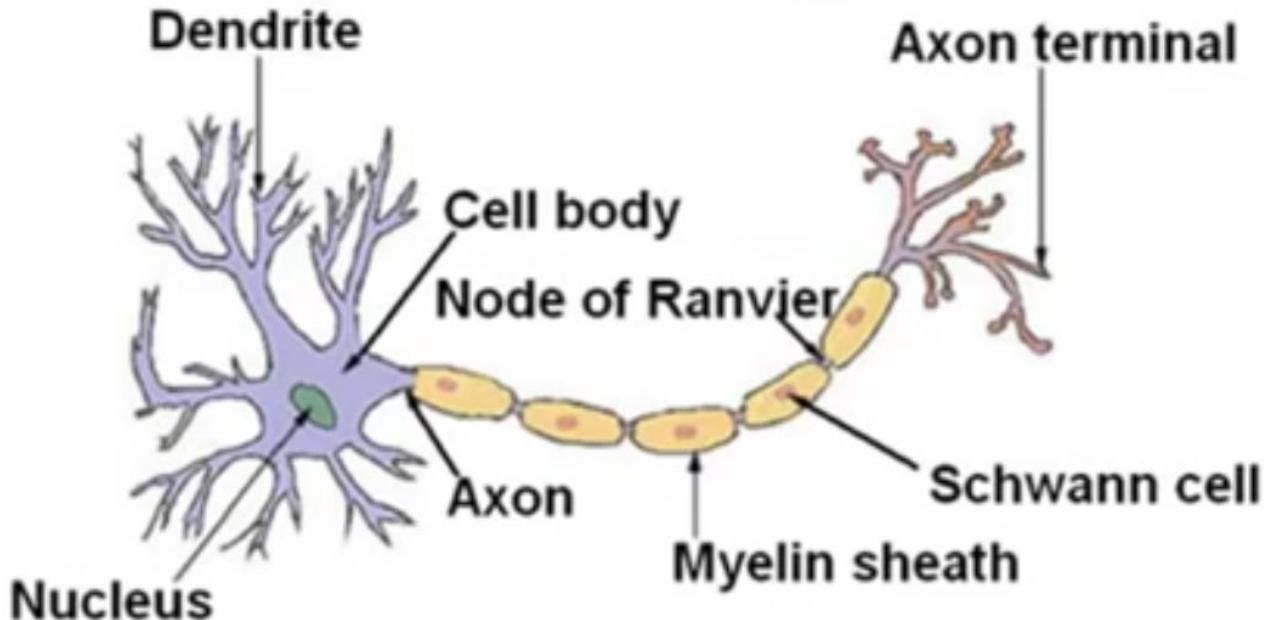
Auditory cortex learns to see

## The “one learning algorithm” hypothesis



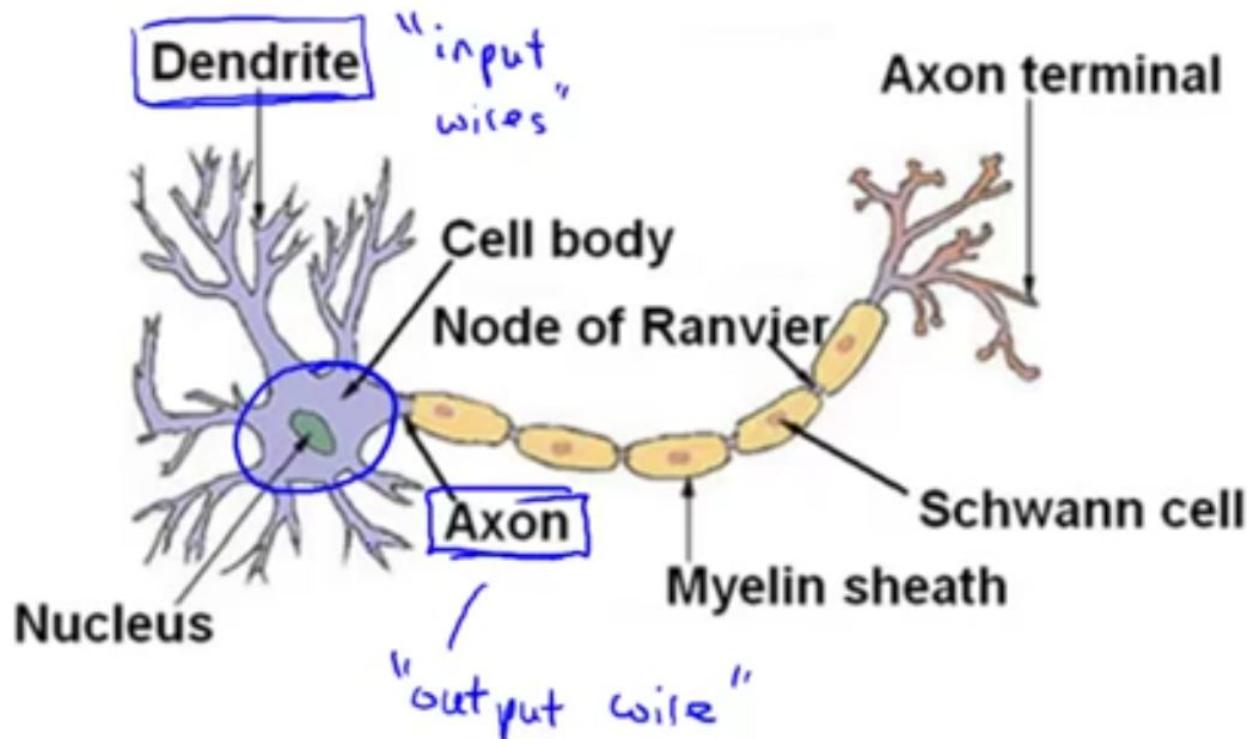
Somatosensory cortex learns to see

# Neuron in the brain

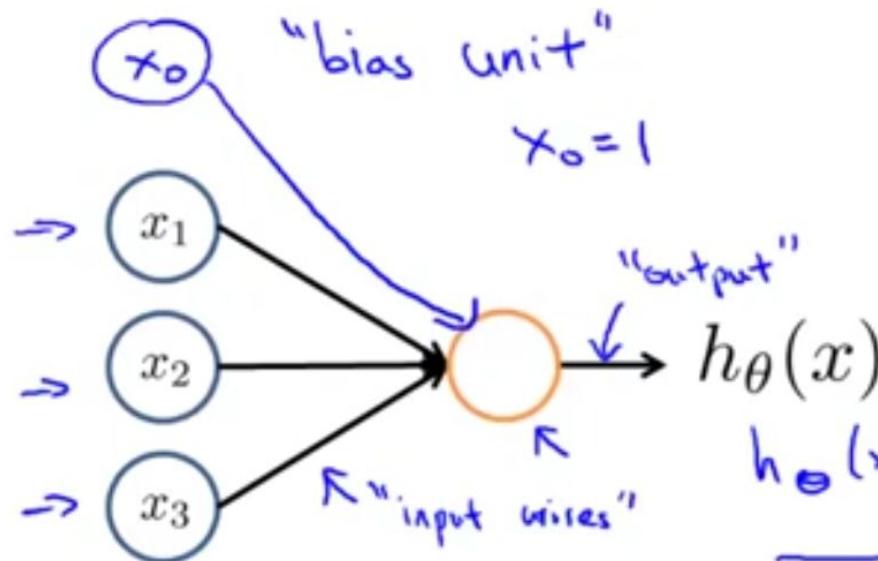


Neural Network Representation- Model Representation

# Neuron in the brain



## Neuron model: Logistic unit



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

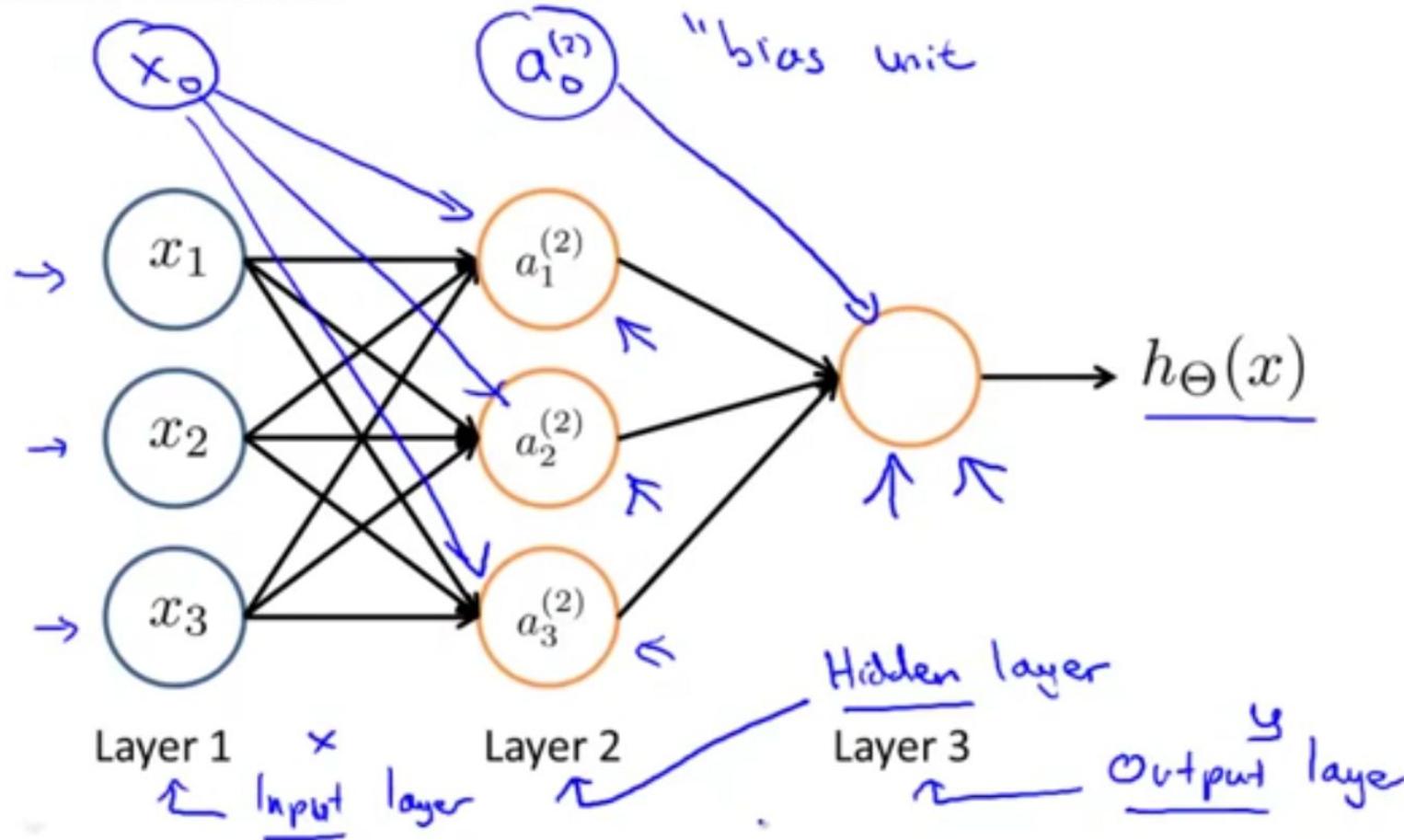
↑  
"weights" ←  
(parameters ↓)

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

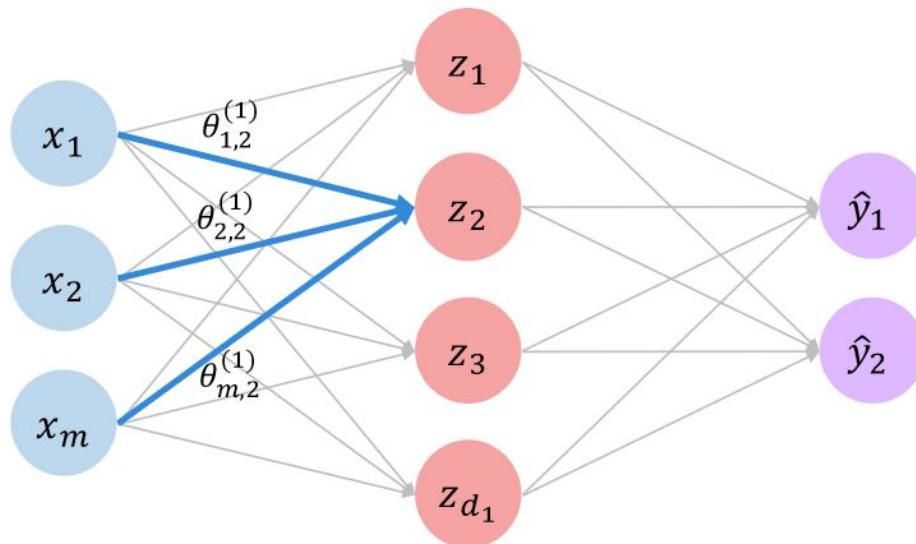
Sigmoid (logistic) activation function.

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

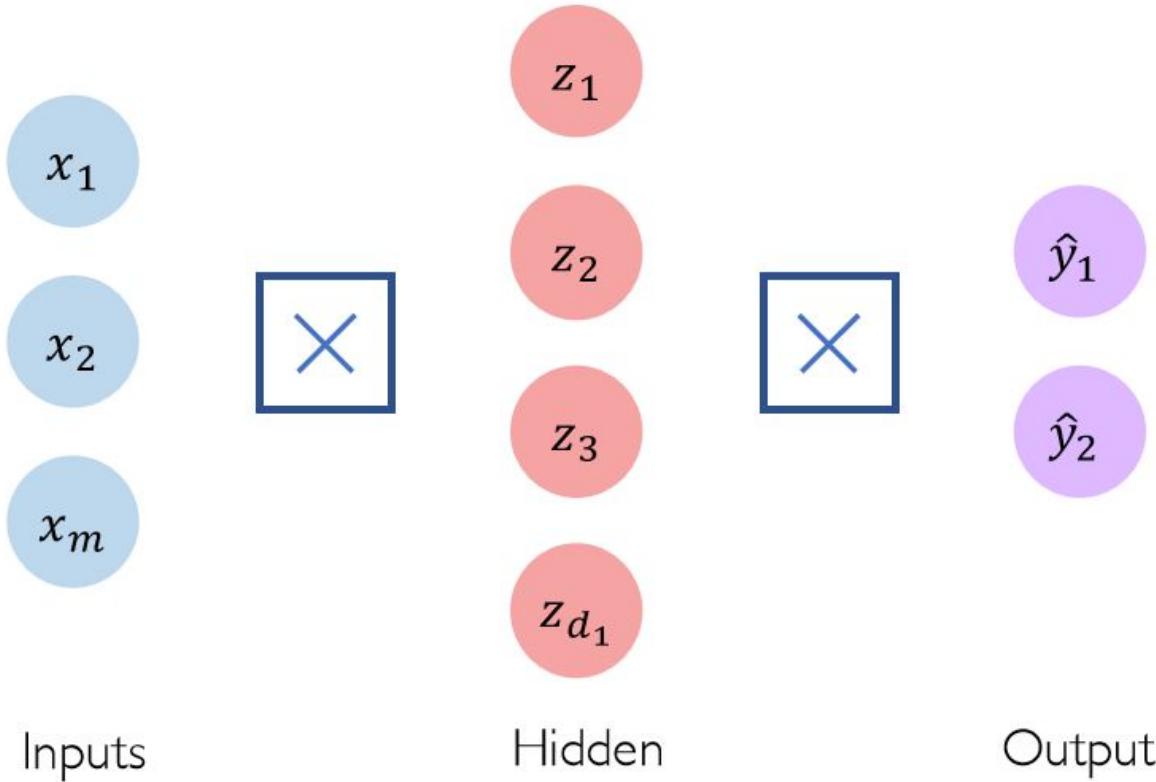
# Neural Network

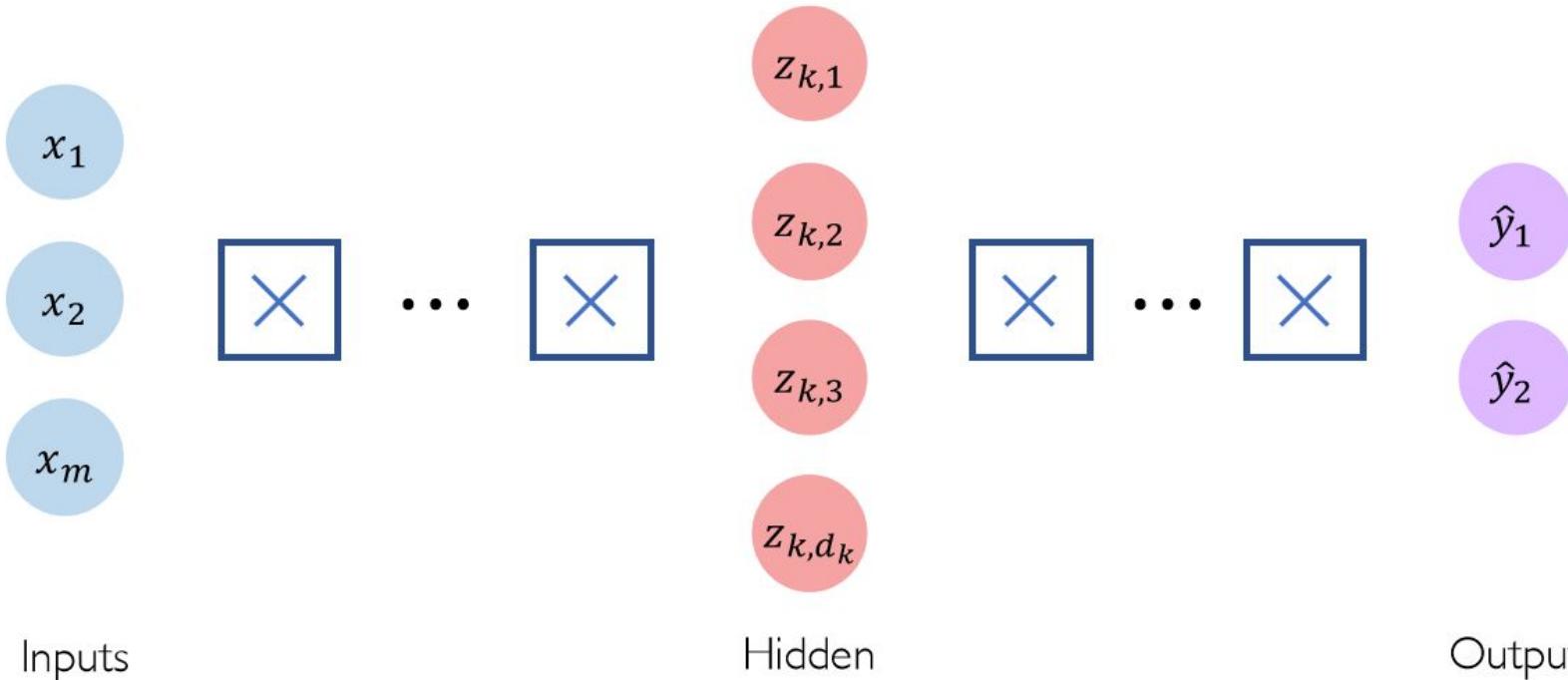


# Single Layer Neural Network



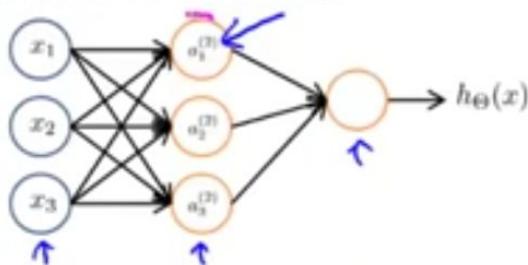
$$\begin{aligned} z_2 &= \theta_{0,2}^{(1)} + \sum_{j=1}^m x_j \theta_{j,2}^{(1)} \\ &= \theta_{0,2}^{(1)} + x_1 \theta_{1,2}^{(1)} + x_2 \theta_{2,2}^{(1)} + x_m \theta_{m,2}^{(1)} \end{aligned}$$





$$z_{k,i} = \theta_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j}) \theta_{j,i}^{(k)}$$

## Neural Network



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

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

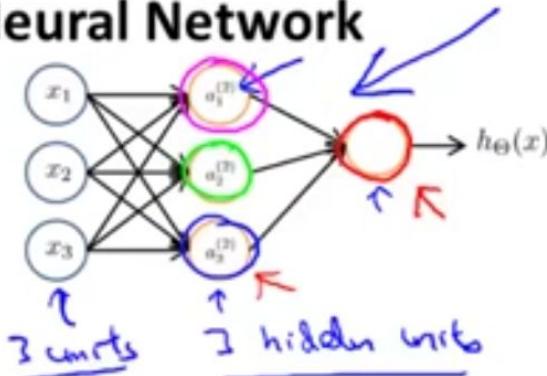
$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

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

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

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

## Neural Network



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

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

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

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

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

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

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

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

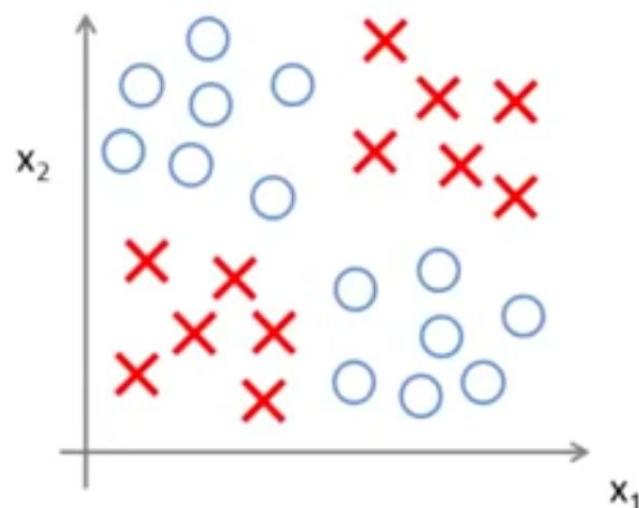
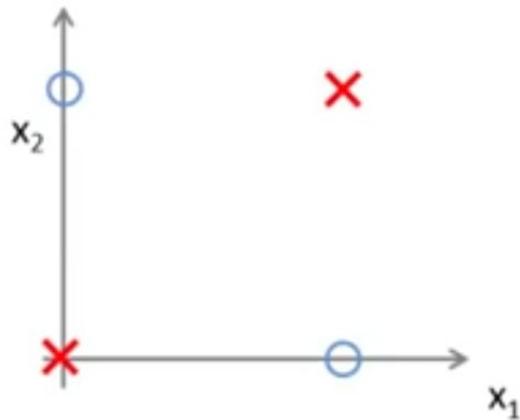
$\hookrightarrow$  If network has  $s_j$  units in layer  $j$ ,  $s_{j+1}$  units in layer  $j + 1$ , then  $\underline{\Theta^{(j)}}$  will be of dimension  $\underline{s_{j+1}} \times (\underline{s_j} + 1)$ .

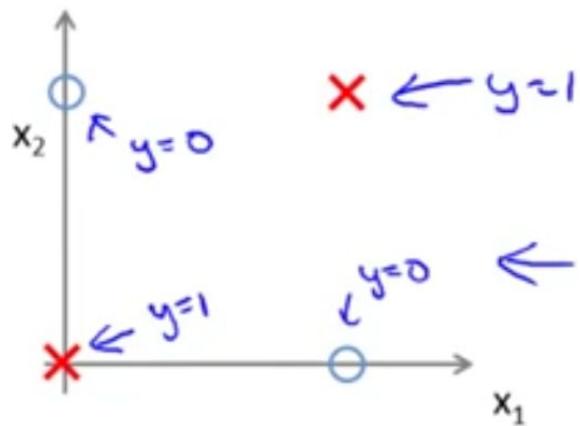
$$s_{j+1} \times (s_j + 1)$$

# Examples and Intuitions 1

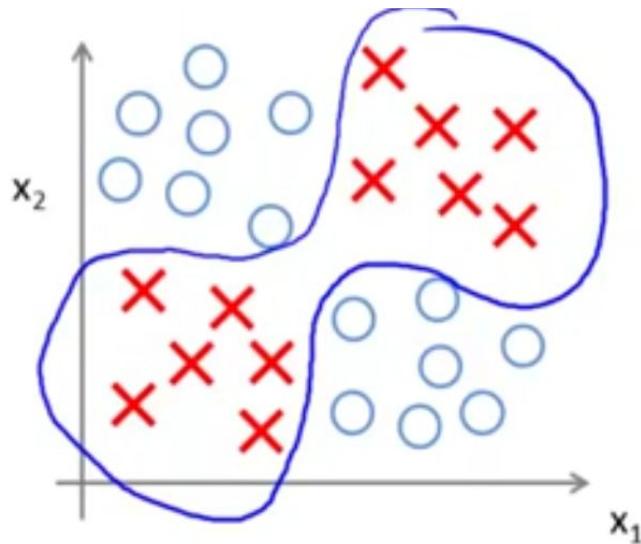
Non-linear classification example: XOR/XNOR

→  $x_1, x_2$  are binary (0 or 1).





$$\begin{aligned}
 y &= \underline{x_1 \text{ XOR } x_2} \\
 &\hookrightarrow \underline{x_1 \text{ XNOR } x_2} \\
 &\hookrightarrow \underline{\text{NOT } (x_1 \text{ XOR } x_2)}
 \end{aligned}$$

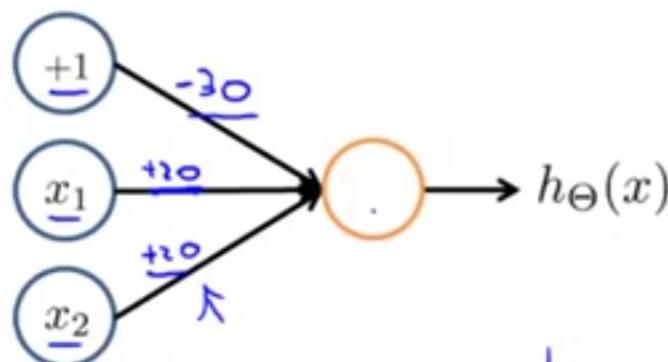




## Simple example: AND

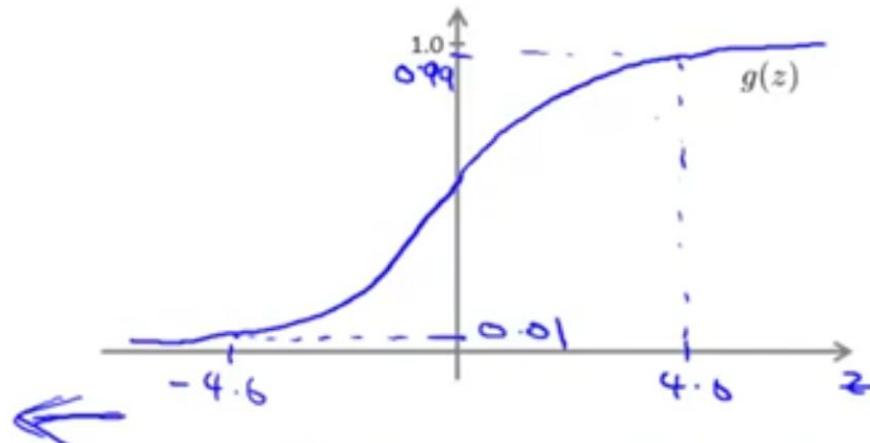
$$\rightarrow x_1, x_2 \in \{0, 1\}$$

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



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

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

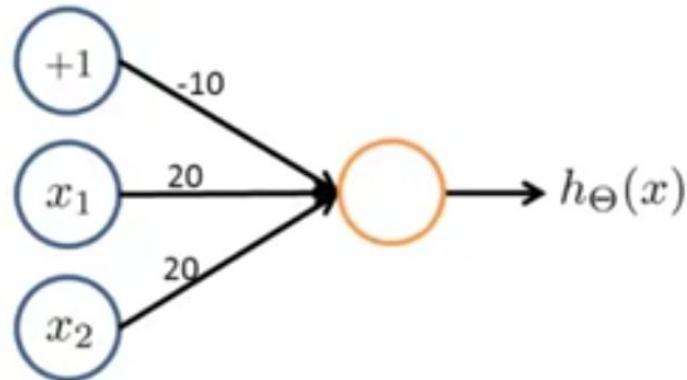


$x_1$	$x_2$	$h_{\Theta}(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

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



## Example: OR function



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

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



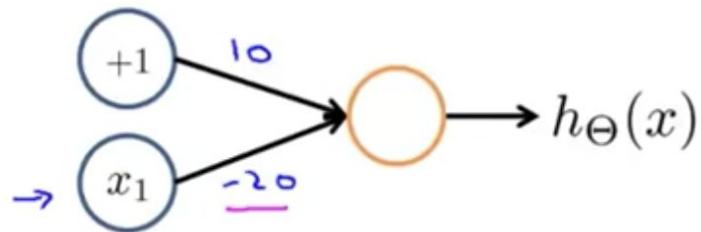
$\rightarrow x_1 \text{ AND } x_2$

$\rightarrow x_1 \text{ OR } x_2$

$\{0, 1\}$ .

### Negation:

NOT  $x_1$

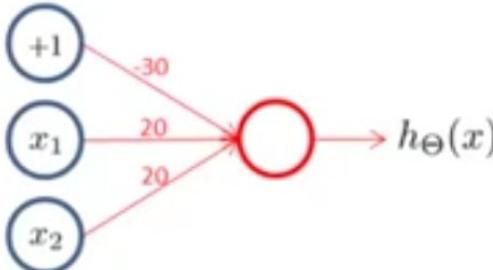


$x_1$	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-20) \approx 0$

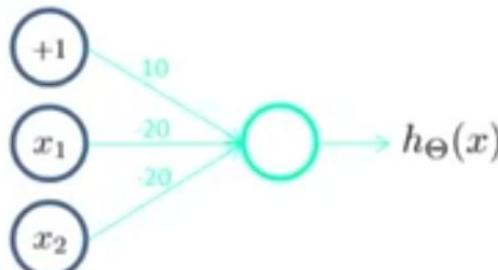
$$h_\Theta(x) = g(10 - 20x_1)$$

$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$   
 ↪ if and only if  
 $\rightarrow x_1 = x_2 = 0$

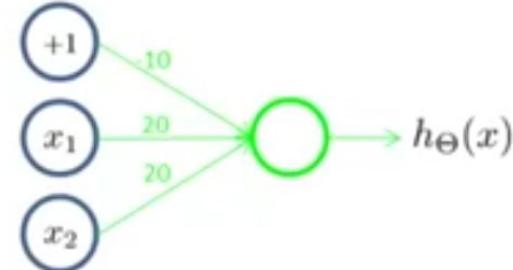
## Putting it together: $x_1$ XNOR $x_2$



\*  $x_1$  AND  $x_2$



(NOT  $x_1$ ) AND (NOT  $x_2$ )

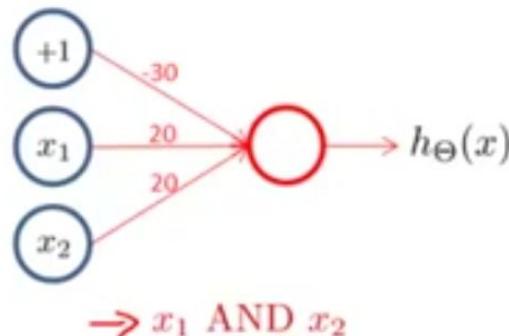


$x_1$  OR  $x_2$

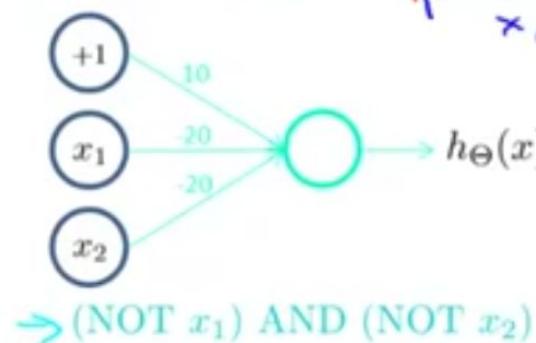


$x_1$	$x_2$	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\Theta}(x)$
0	0			
0	1			
1	0			
1	1			

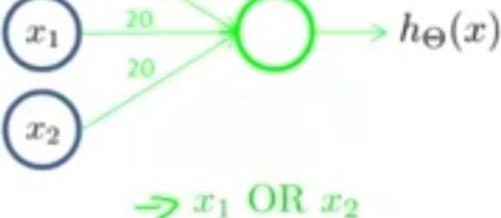
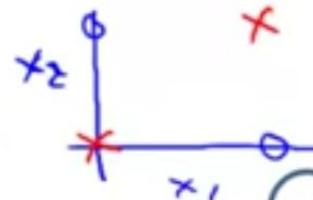
Putting it together:  $x_1 \text{ XNOR } x_2$



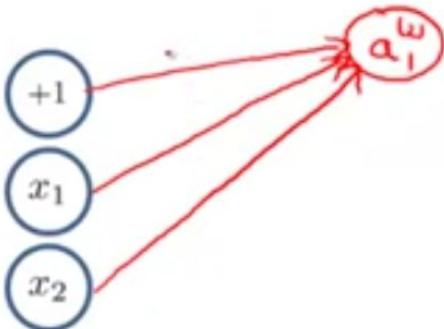
$\rightarrow x_1 \text{ AND } x_2$



$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$

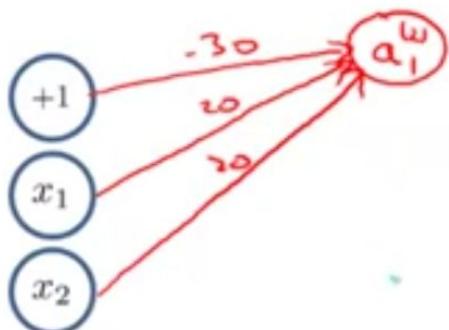
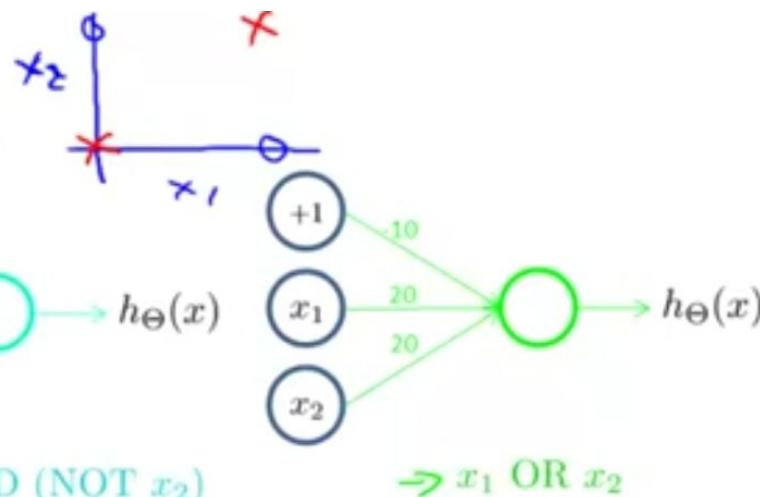
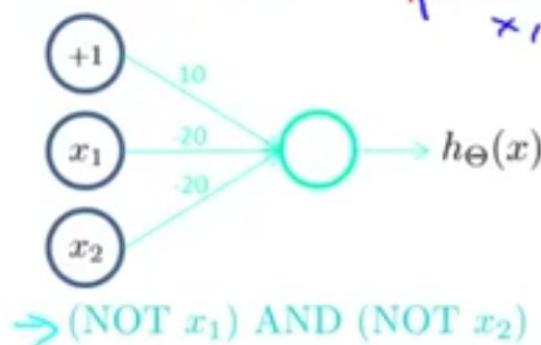
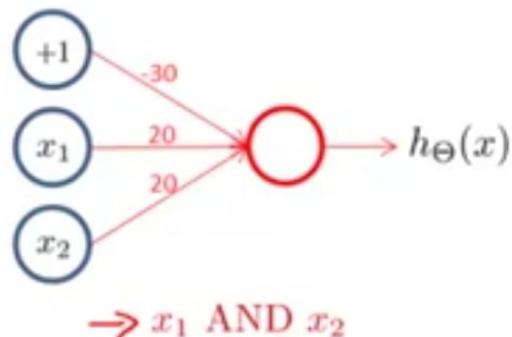


$\rightarrow x_1 \text{ OR } x_2$



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

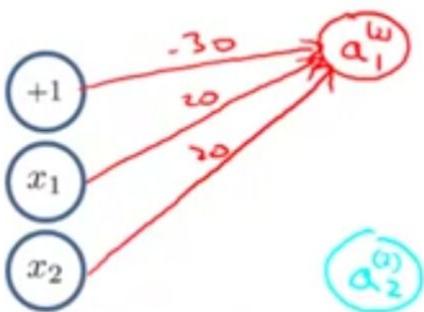
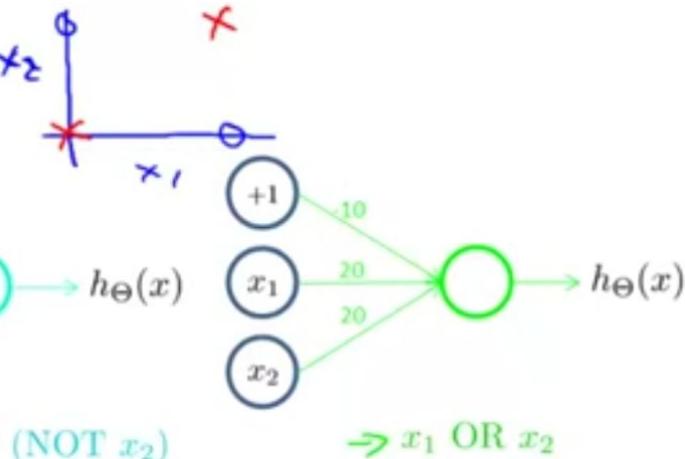
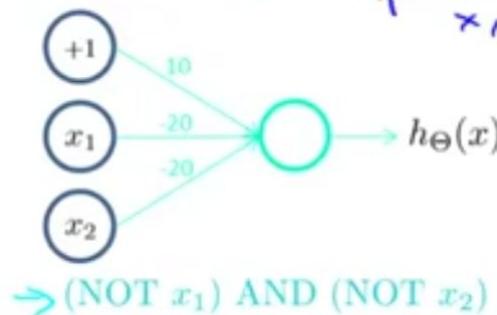
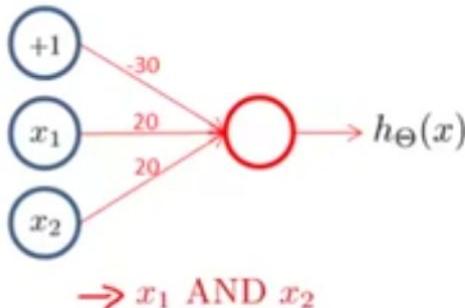
**Putting it together:**  $x_1$  XNOR  $x_2$



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



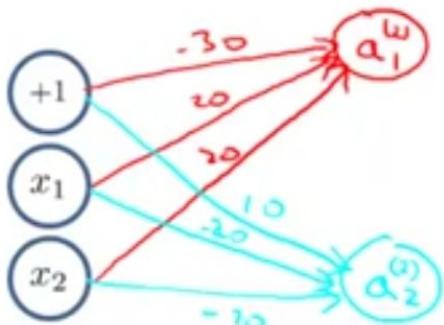
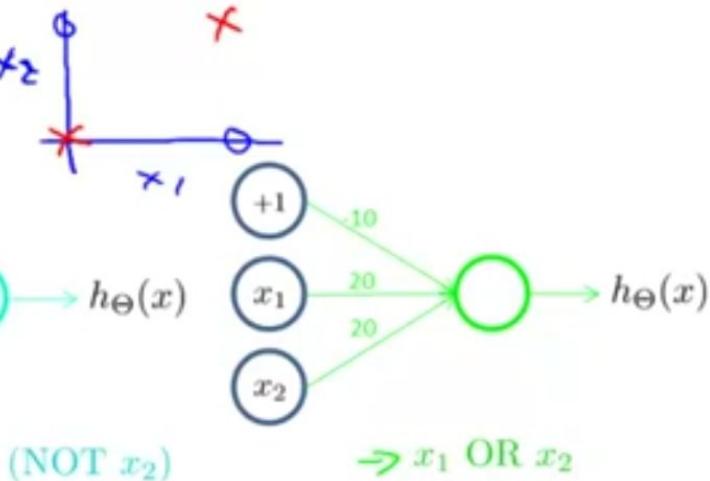
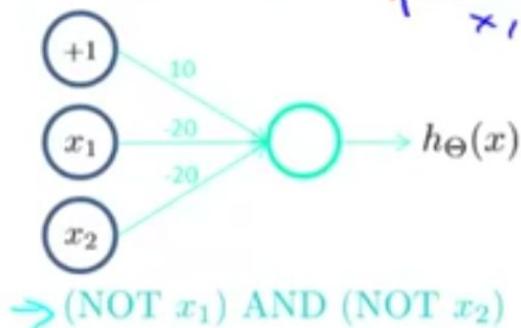
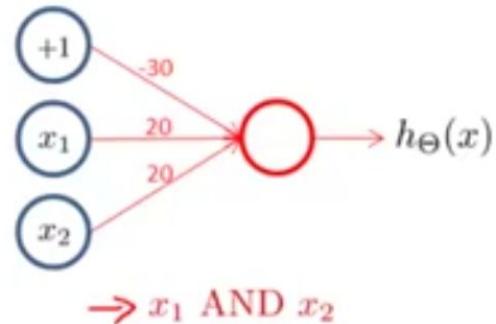
**Putting it together:**  $x_1 \text{ XNOR } x_2$



$x_1$	$x_2$	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\Theta}(x)$
0	0	-30	-30	0
0	1	-30	0	0
1	0	0	-30	0
1	1	0	0	1



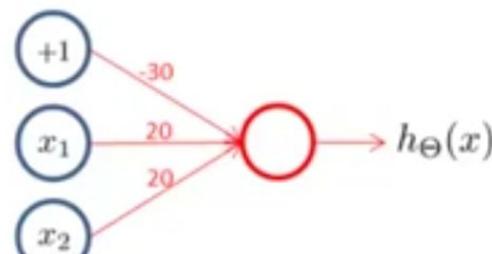
Putting it together:  $x_1$  XNOR  $x_2$



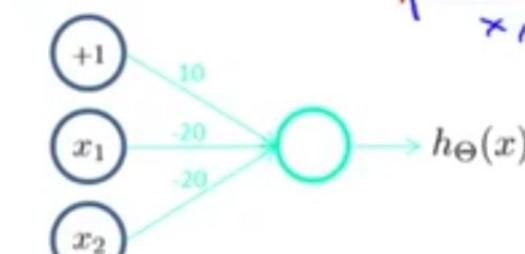
$x_1$	$x_2$	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\Theta}(x)$
0	0	-1	1	-1
0	1	-1	0	-1
1	0	0	-1	-1
1	1	1	1	1



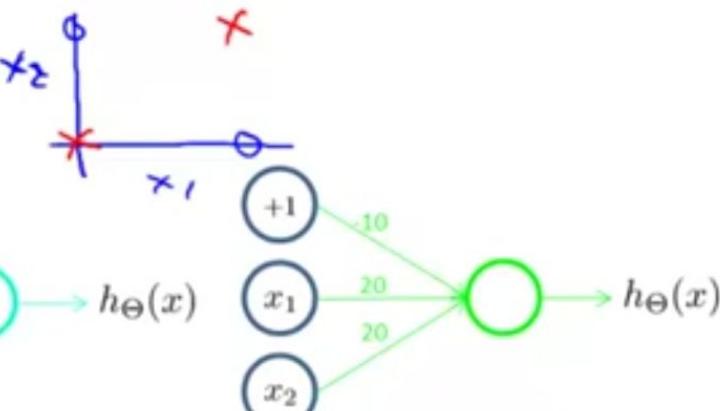
**Putting it together:**  $x_1$  XNOR  $x_2$



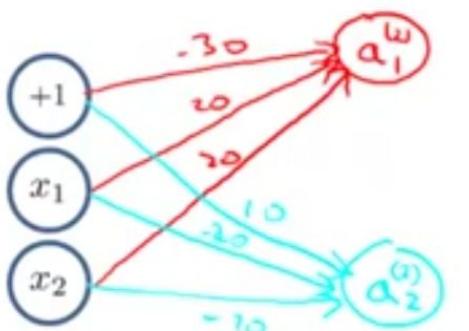
$\rightarrow x_1 \text{ AND } x_2$



$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$

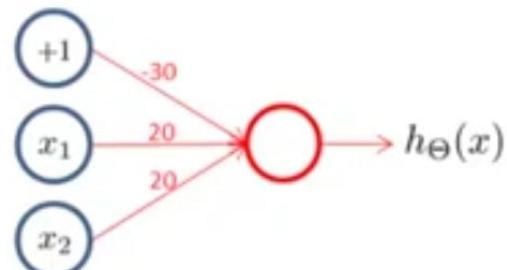


$\rightarrow x_1 \text{ OR } x_2$

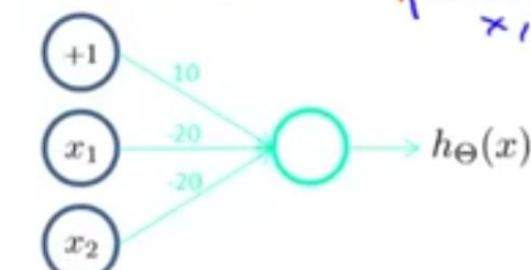


$x_1$	$x_2$	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\Theta}(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

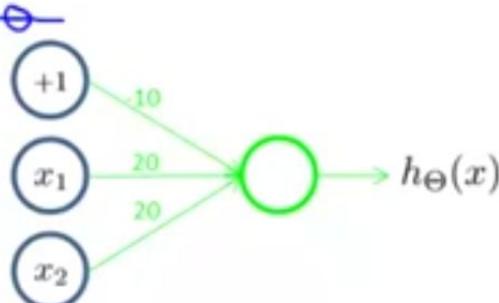
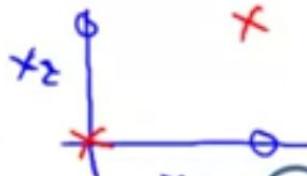
## Putting it together: $x_1 \text{ XNOR } x_2$



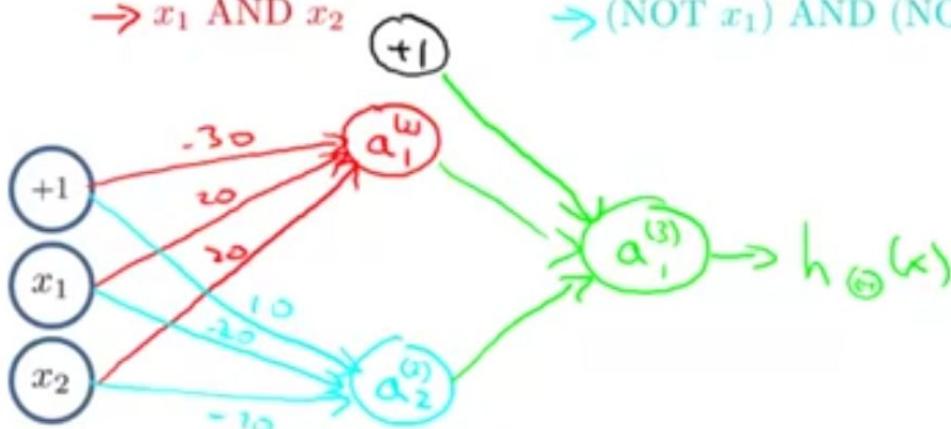
$\rightarrow x_1 \text{ AND } x_2$



$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$

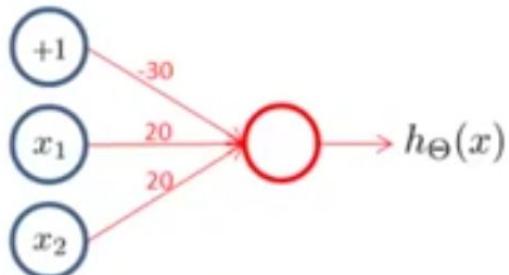


$\rightarrow x_1 \text{ OR } x_2$

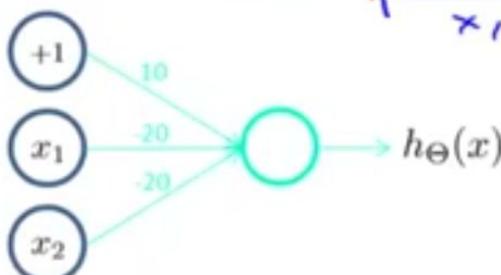


$x_1$	$x_2$	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	0
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

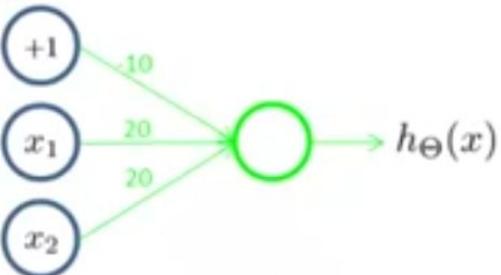
**Putting it together:**  $x_1 \text{ XNOR } x_2$



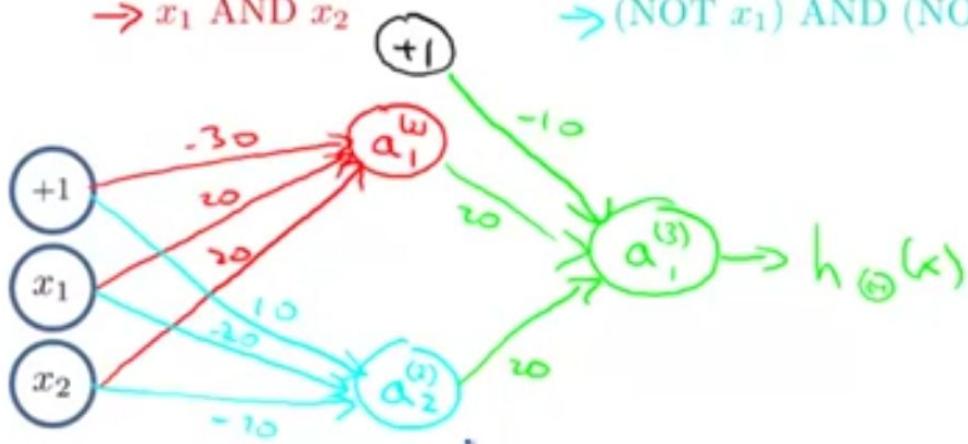
$\rightarrow x_1 \text{ AND } x_2$



$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$



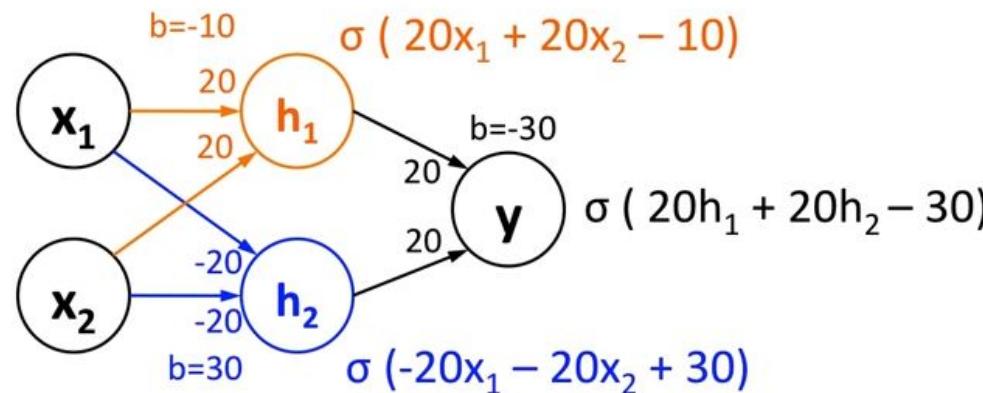
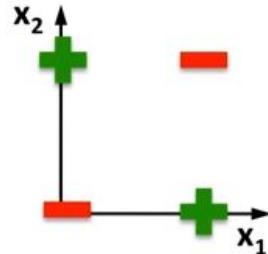
$\rightarrow x_1 \text{ OR } x_2$



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

# Solving XOR with a Neural Net

Linear classifiers  
cannot solve this



$$\sigma(20*0 + 20*0 - 10) \approx 0$$

$$\sigma(20*1 + 20*0 - 10) \approx 1$$

$$\sigma(20*0 + 20*1 - 10) \approx 1$$

$$\sigma(20*1 + 20*0 - 10) \approx 1$$

$$\sigma(-20*0 - 20*0 + 30) \approx 1$$

$$\sigma(-20*1 - 20*0 + 30) \approx 0$$

$$\sigma(-20*0 - 20*1 + 30) \approx 1$$

$$\sigma(-20*1 - 20*0 + 30) \approx 1$$

$$\sigma(20*0 + 20*1 - 30) \approx 0$$

$$\sigma(20*1 + 20*0 - 30) \approx 0$$

$$\sigma(20*1 + 20*1 - 30) \approx 1$$

$$\sigma(20*1 + 20*1 - 30) \approx 1$$

OR

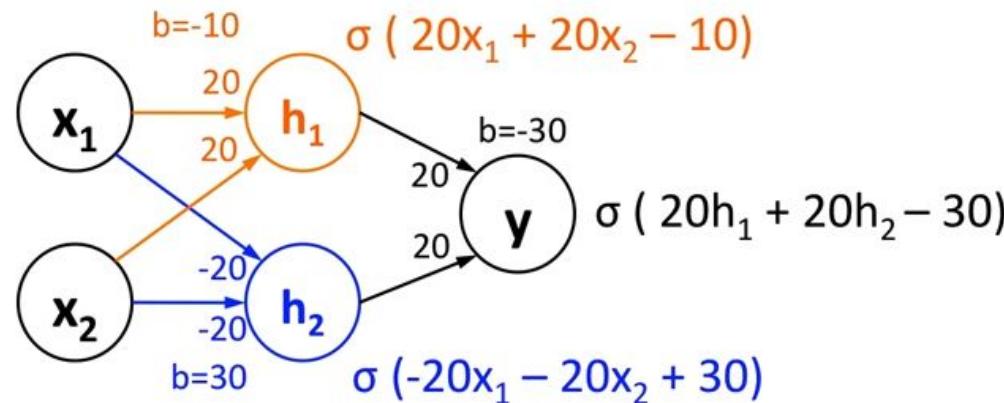
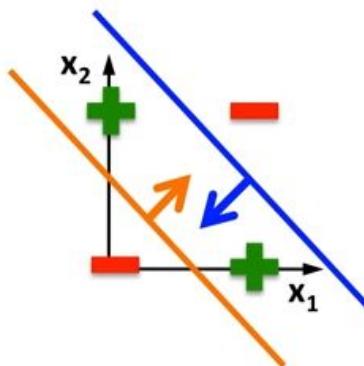
OR

OR



# Solving XOR with a Neural Net

Linear classifiers  
cannot solve this



$$\sigma(20*0 + 20*0 - 10) \approx 0$$

$$\sigma(20*1 + 20*1 - 10) \approx 1$$

$$\sigma(20*0 + 20*1 - 10) \approx 1$$

$$\sigma(20*1 + 20*0 - 10) \approx 1$$

$$\sigma(-20*0 - 20*0 + 30) \approx 1$$

$$\sigma(-20*1 - 20*1 + 30) \approx 0$$

$$\sigma(-20*0 - 20*1 + 30) \approx 1$$

$$\sigma(-20*1 - 20*0 + 30) \approx 1$$

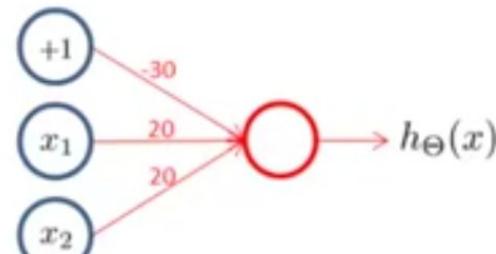
$$\sigma(20*0 + 20*1 - 30) \approx 0$$

$$\sigma(20*1 + 20*0 - 30) \approx 0$$

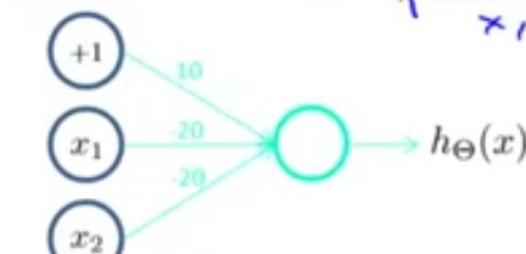
$$\sigma(20*1 + 20*1 - 30) \approx 1$$

$$\sigma(20*1 + 20*1 - 30) \approx 1$$

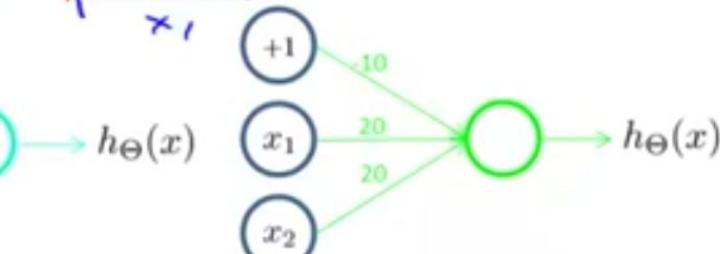
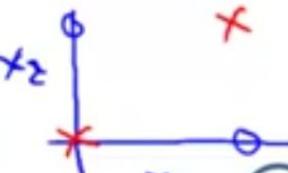
Putting it together:  $x_1 \text{ XNOR } x_2$



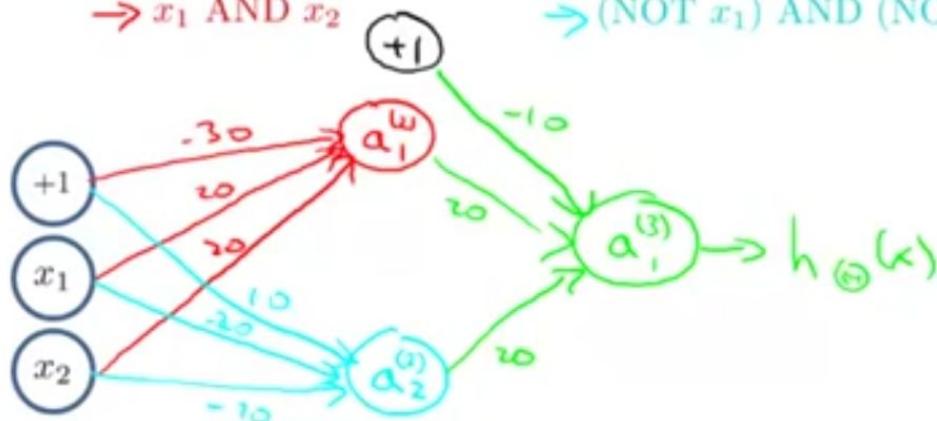
$\rightarrow x_1 \text{ AND } x_2$



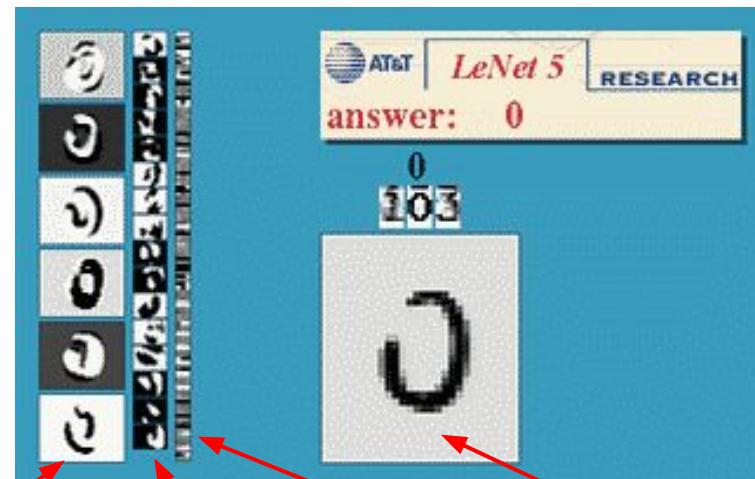
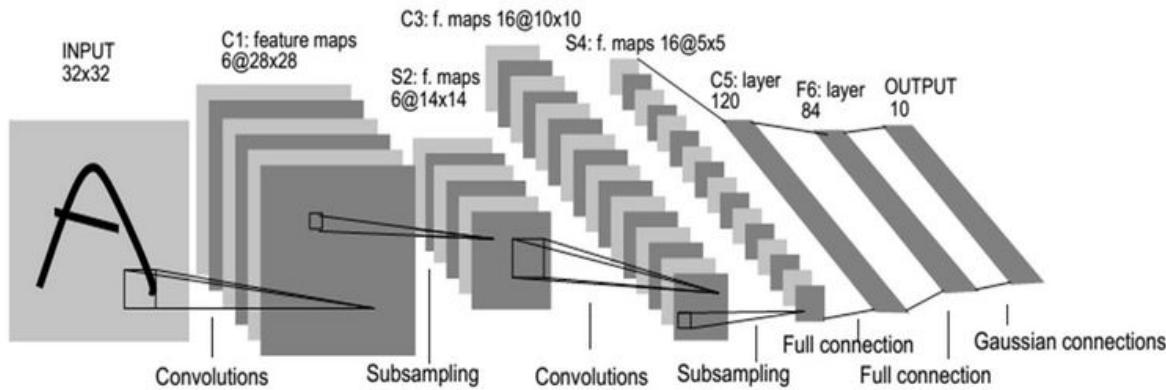
$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$



$\rightarrow x_1 \text{ OR } x_2$



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

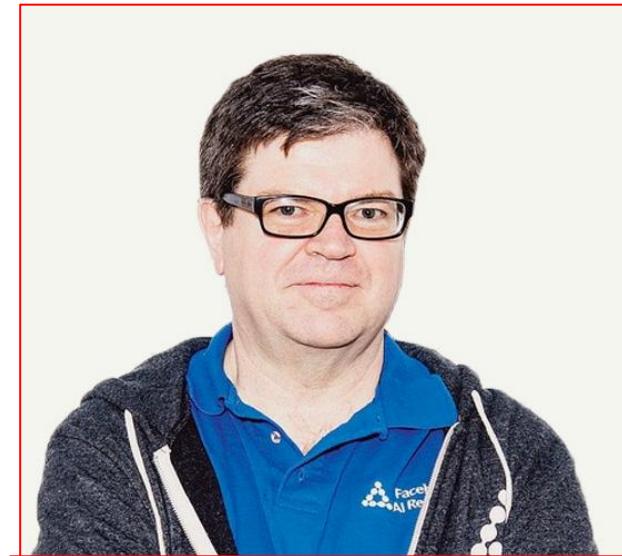


Layer - 1

Layer - 3

Layer - 5

Input



# Yann LeCun

Director of AI Research

# facebook

AI

## Multiple output units: One-vs-all.



Pedestrian



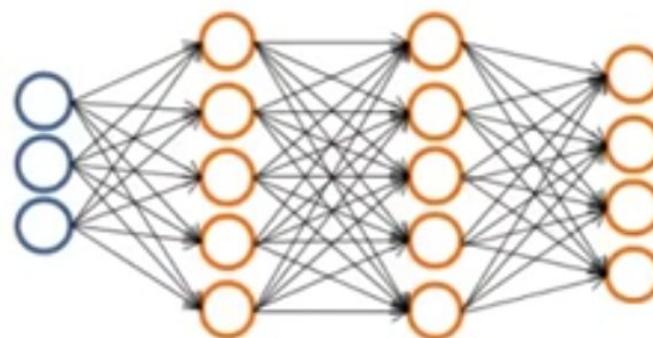
Car



Motorcycle



Truck



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



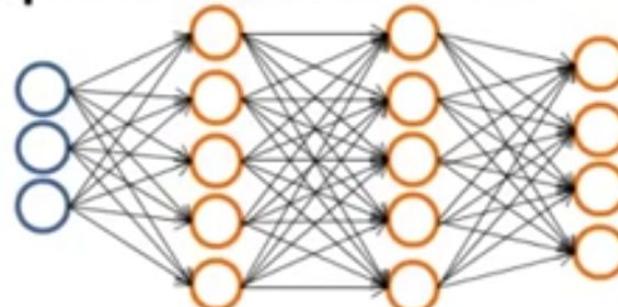
Want  $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ ,  $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ , etc.

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

→  $y^{(i)}$  one of  $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$   
pedestrian    car    motorcycle    truck

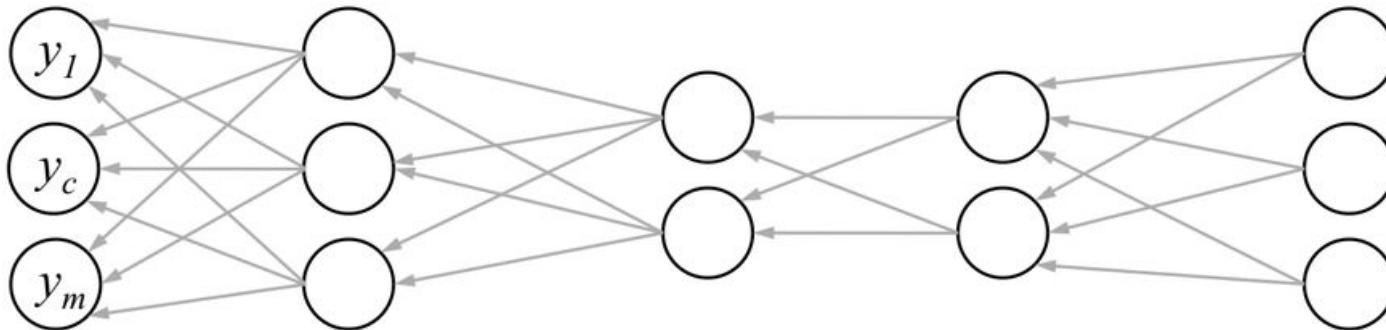
$(x^{(i)}, y^{(i)})$



~~Previously~~  
 $y \in \{1, 2, 3, 4\}$   
 $\underline{h_{\Theta}(x^{(i)}) \approx y^{(i)}} \in \mathbb{R}^4$

Andrew Ng

# Multi-class Neural Networks



- Default NN predicts a single class (or number)
- Add separate output  $y_c$  unit for each class  $c$ 
  - example  $(\mathbf{x}, c)$ : set  $y_c = 1$ , and  $y_i = 0$  for all  $i \neq c$
- Predict class  $c$  with the highest  $y_c$
- Can compute confidence in  $c$ :  
$$P(c | y) = \frac{e^{y_c}}{\sum_c e^{y_c}}$$
  - soft-max rule, same as in logistic

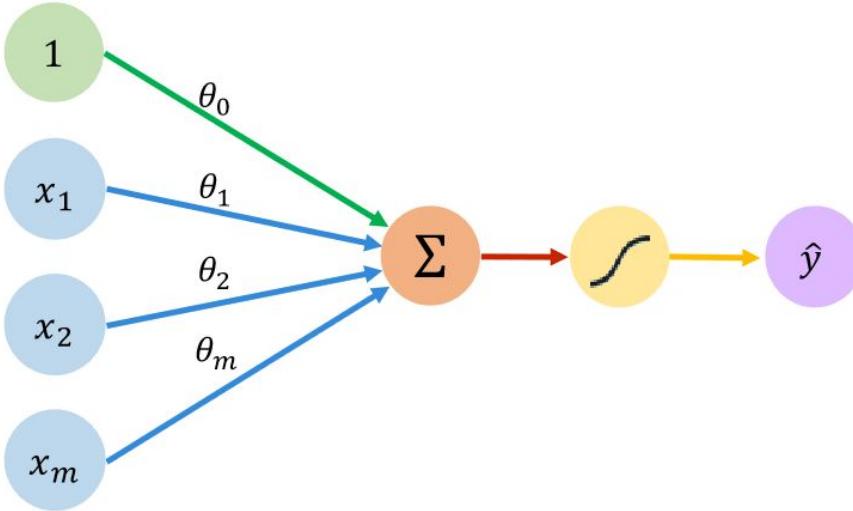


# PERCEPTRON

The fundamental building block of Deep Learning.

PLAYGROUND

Review  

Linear combination of inputs

$$\hat{y} = g \left( \theta_0 + \sum_{i=1}^m x_i \theta_i \right)$$

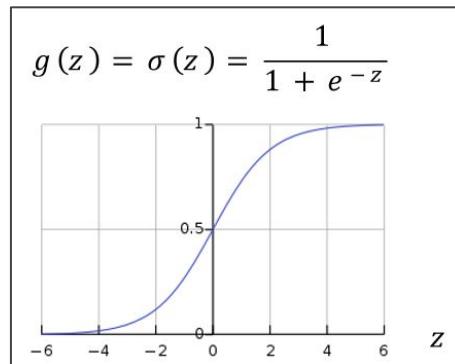
Output

Non-linear activation function

Bias

## Activation Functions

Inputs      Weights      Sum      Non-Linearity      Output



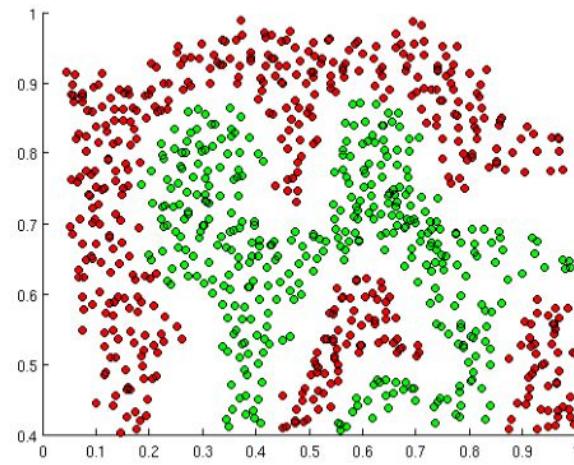
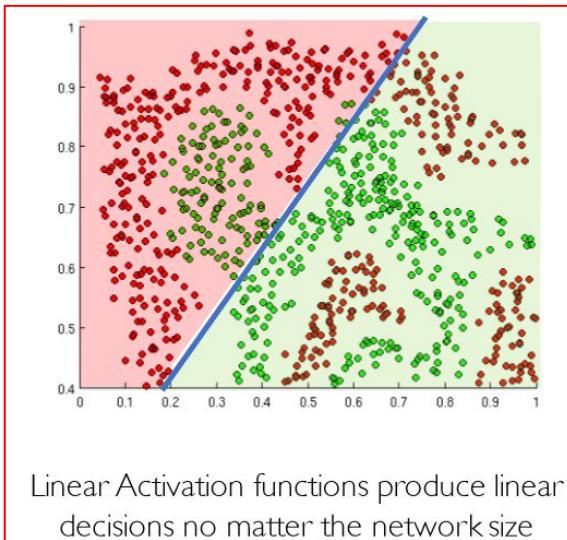
$$\hat{y} = g(\theta_0 + \mathbf{X}^T \boldsymbol{\theta})$$

- Example: sigmoid function

where:  $\mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$  and  $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$

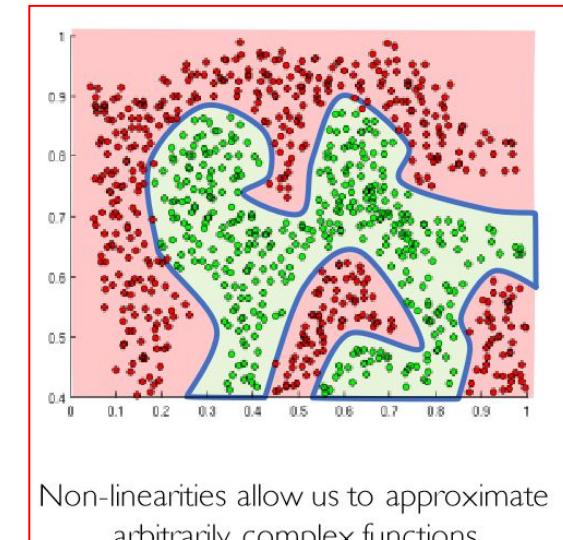
The purpose of activation functions is to introduce non-linearities into the network.

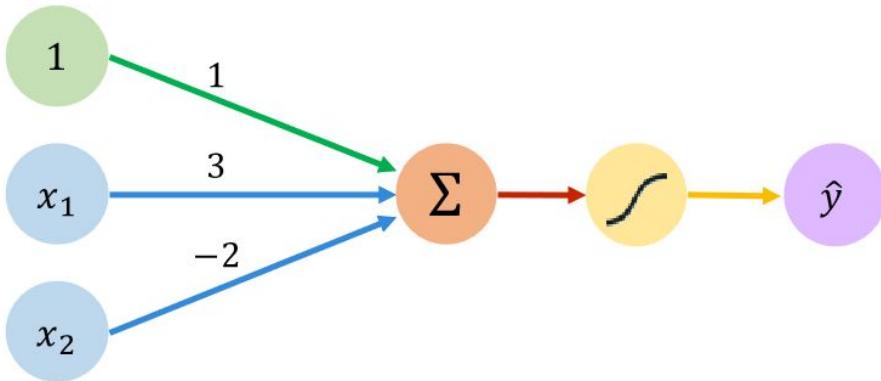
What you can do... 😢



Reality is complex i.e non-linear..

What you want to do... 😎



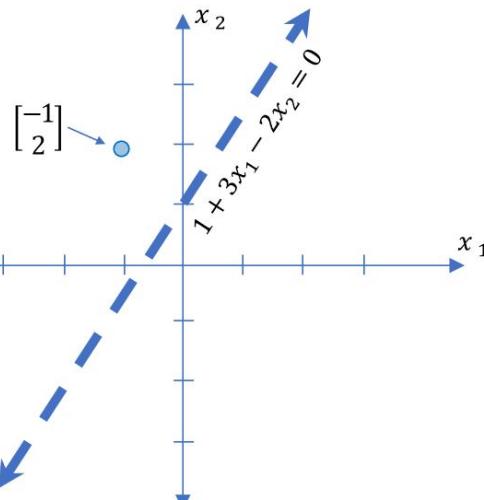
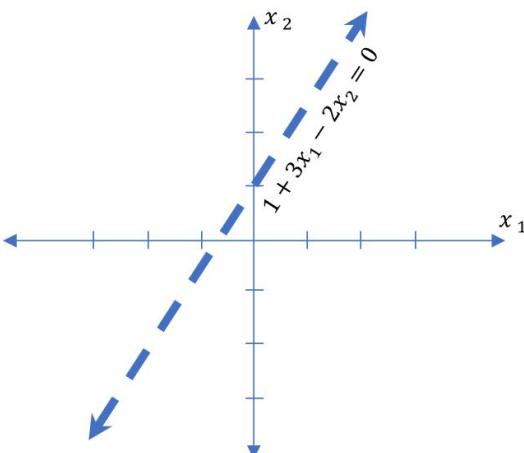


We have:  $\theta_0 = 1$  and  $\boldsymbol{\theta} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

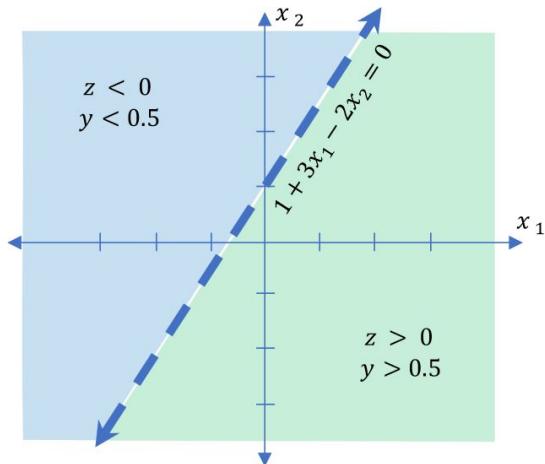
$$\begin{aligned}\hat{y} &= g(\theta_0 + \mathbf{X}^T \boldsymbol{\theta}) \\ &= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right) \\ \hat{y} &= g\left(1 + 3x_1 - 2x_2\right)\end{aligned}$$

This is just a line in 2D!

$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



# Hinton's Closing Prayer

Our father who art in n-dimensions

hallowed by the backprop,

thy loss be minimized,

thy gradients unvarnished,

on earth as it is in Euclidean space.

Give us this day our daily  
hyperparameters,

and forgive us our large learning  
rates,

as we forgive those whose  
parameters diverge,

and lead us not into discrete  
optimization,

but deliver us from local optima.

For thine are dimensions,

and the GPUs, and the glory,

forever and ever. Dropout.



# Q&A SESSION

..

