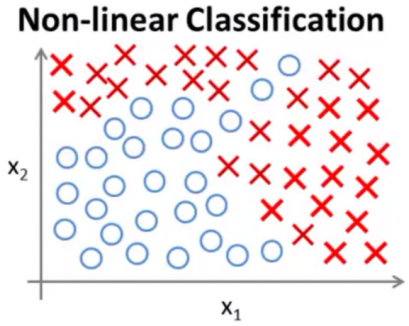
NEURAL NETWORKS
REPRESENTATION

→ MOTIVATIONS

Non-linear Hypothesis:



Applying Logistic Regression:

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

The number of quadratic features grows roughly as order of n^2, where n is the number of original features. ($\approx n^2/2$)

→ Problems:

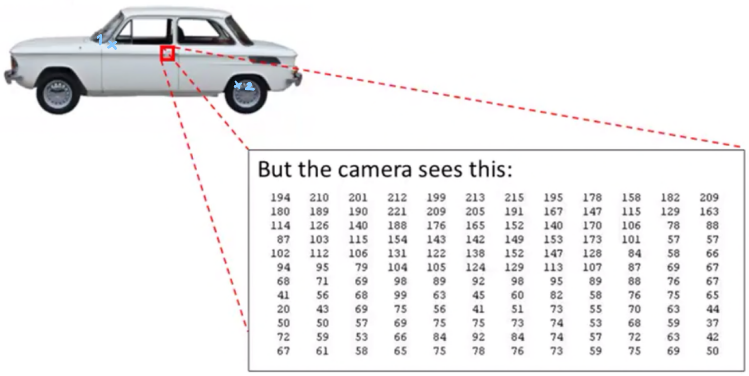
- With a lot of features we can overfit
- It is expensive to compute

For many machine learning problems, n will be very large.

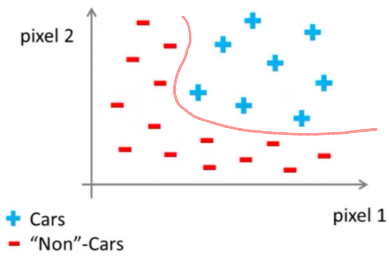
EXAMPLE: COMPUTER VISION

Car Detector:

You see this:



→ The computer sees a matrix of pixel intensity values that tell the brightness of each pixel.



→ non-linear hypothesis to separate the two classes.

Dimension of the feature space: with 50 x 50 pixel images
n = 2500 (grayscale) / n = 7500 (RGB)

→ Simple logistic regression with adding quadratic or cubic features it's not a good way to learn complex nonlinear hypothesis when n is large.

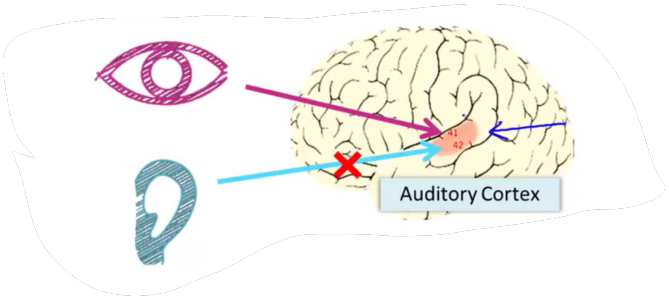
Background

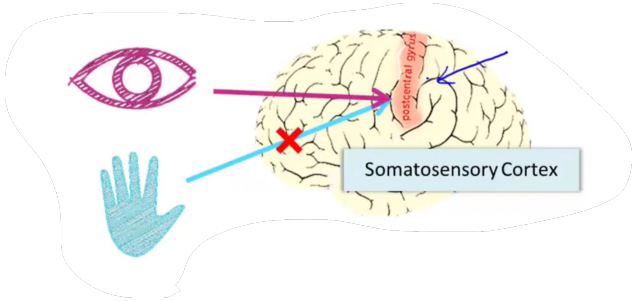
- Algorithms that try to mimic the brain (the most amazing learning machine that we know)
- Widely used in the 80s and early 90s, but popularity diminished in late 90s.
- Recent ressurgence – neural networks are computationally expensive algorithms and only more recently computers became fast enough to run large scale Neural Networks.
- Modern Neural Networks are the state-of-the-art technique for many applications

"One learning algorithm" hypothesis

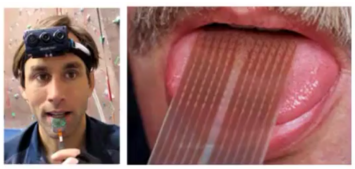
The human brain does so many different and amazing things. It would seem that, in order to mimic the brain, we would need to write lots of different pieces of software → but the way the brain does it seems to be as a single learning algorithm!

→ Neuro-Rewiring Experiments



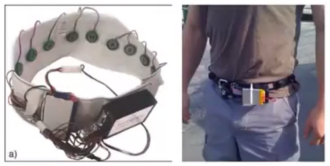


Cutting from ears to auditory cortex and rewiring so that the signal from the eyes gets routed to the auditory cortex, the auditory cortex will learn to see!



Seeing with your tongue

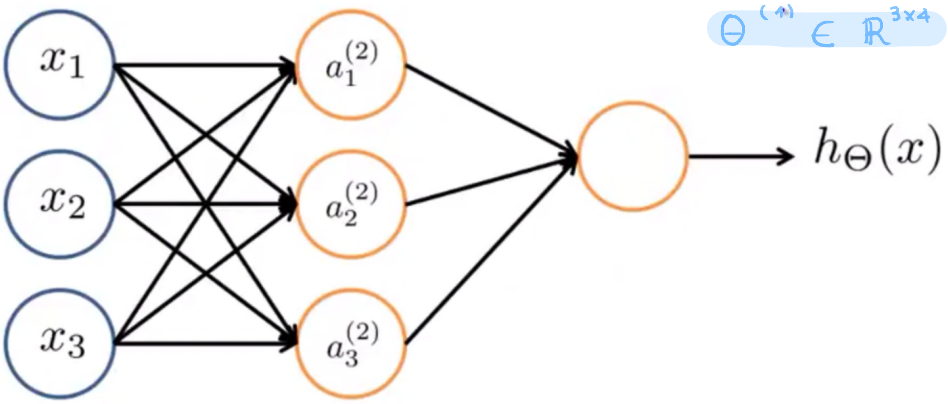
Human echolocation (sonar)



Haptic belt: Direction sense

Implanting a 3rd eye

→ MODEL REPRESENTATION



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

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

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

Input Layer Hidden Layer Output Layer

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

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

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

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

→ sigmoid function / sigmoid activation function

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

→ VECTOR REPRESENTATION

$a_0^{(2)} = 1$ → bias unit

$$a_1^{(2)} = g(\underbrace{\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3}_{z_1^{(2)}}) = g(z_1^{(2)})$$

$$a_2^{(2)} = g(\underbrace{\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3}_{z_2^{(2)}}) = g(z_2^{(2)})$$

$$a_3^{(2)} = g(\underbrace{\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3}_{z_3^{(2)}}) = g(z_3^{(2)})$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \quad z^{(2)} \in \mathbb{R}^3$$

$x_0 = 1$

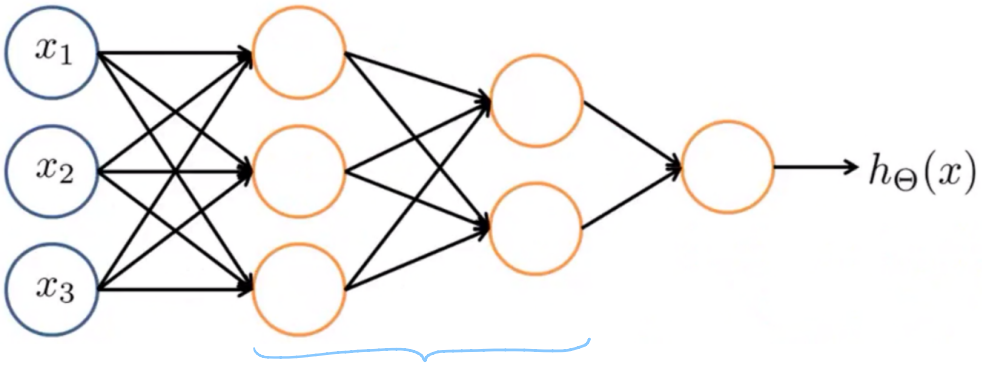
→ we can compute z_2 as $\Theta^{(1)} x$, and $a^{(2)}$ as $g(z^{(2)})$ → element-wise

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

→ This process of computing $h_\Theta(x)$ is also called **Forward Propagation** → computing the activations from input to hidden to output layer.

Other Architectures

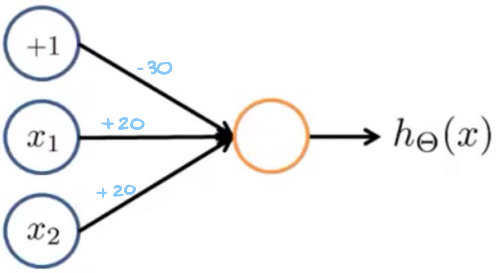
How the different neurons are connected to each other.



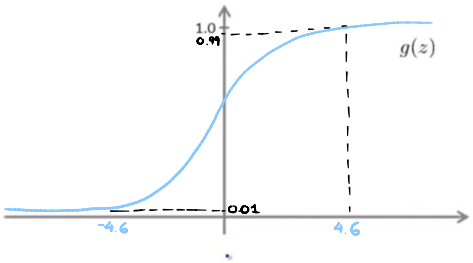
→ With more layers we can compute more complex features.

→ EXAMPLES AND INTUITIONS

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

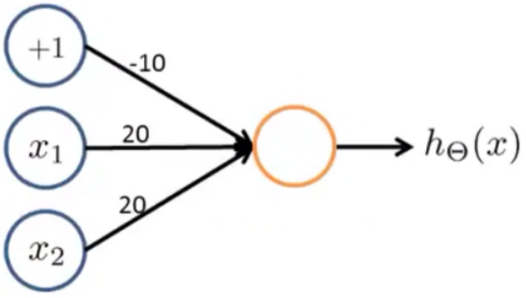


$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$



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

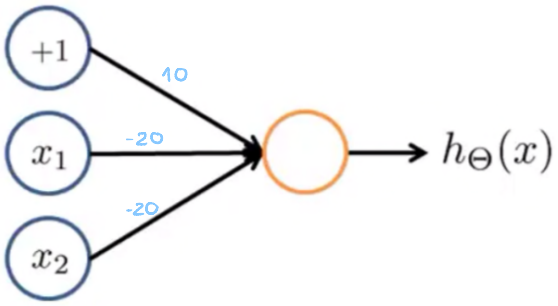
$h_\Theta(x) \approx x_1$ AND x_2



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

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

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

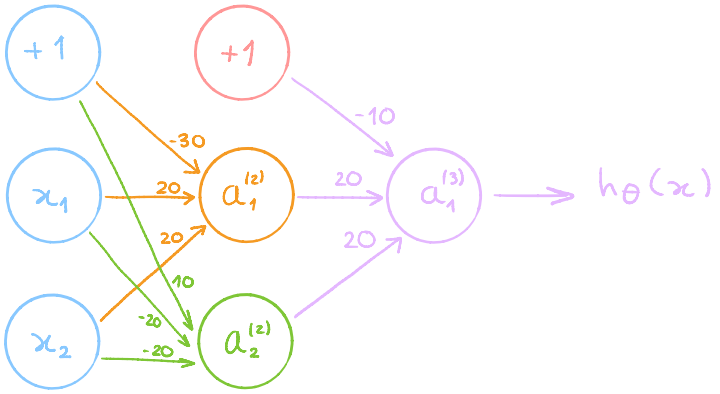


x_1	x_2	$h_\Theta(x)$
0	0	$g(10) \approx 1$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(-30) \approx 0$

$h_\Theta(x) \approx$ (NOT x_1) AND (NOT x_2)

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

Putting it together → x_1 XNOR x_2

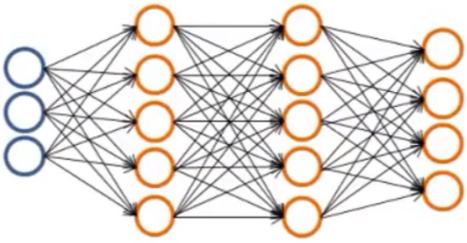


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

→ MULTICLASS CLASSIFICATION

To classify data into multiple classes, we let our hypothesis function return a vector of values.


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