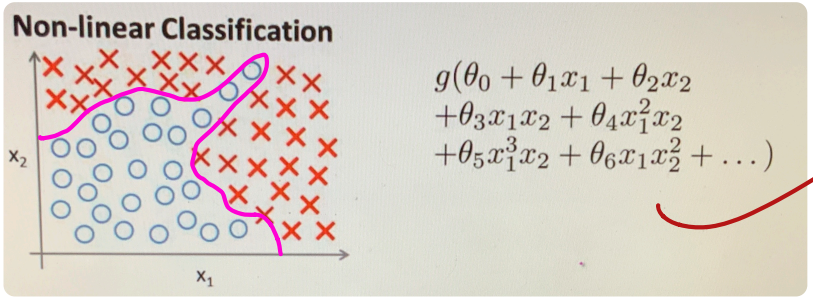
Stanford ML

Representing Neural Networks

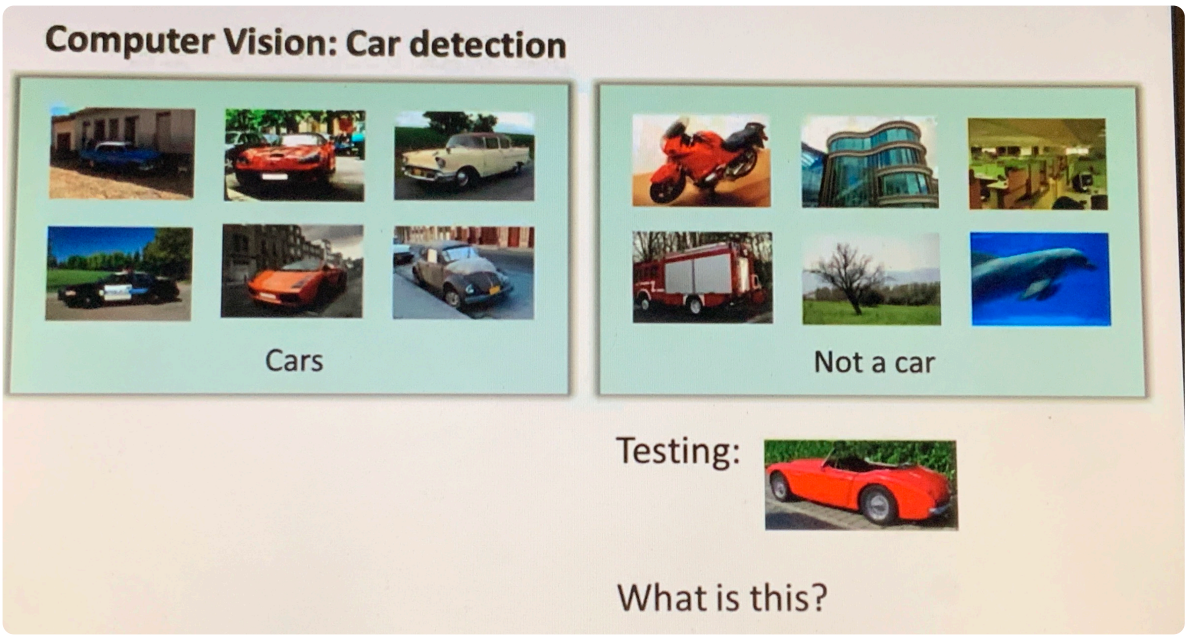
- Why linear and logistic regression aren't enough.

Ex: Non-Linear Classification



- Easy to create a non-linear hypothesis with only a small number of features
- If $n \sim 100$, you end up with ≈ 5000 features to $O(n^2)$
- Complexity: $O(n^2/2)$
- Can lead to OVERFITTING via too many features

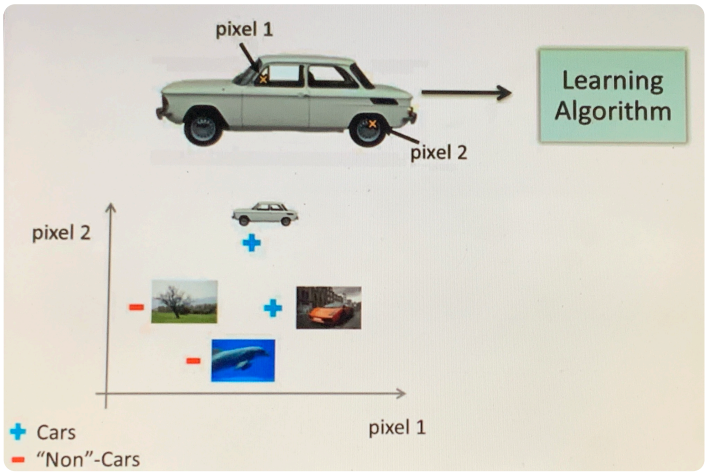
Computer Vision

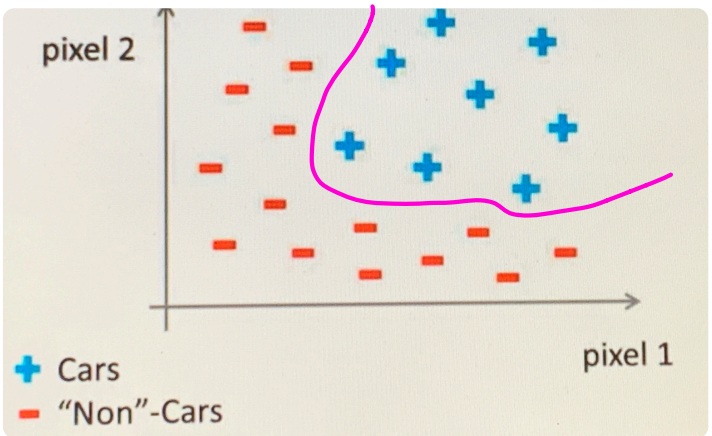


- Identifying what is a car involves giving the

algorithm a labelled data set for training the classifier

- By comparing pixel intensities between images like on the right, we end up with something like





Therefore we need a non-linear hypothesis in order to classify images into cars and not cars

USING 50 x 50 images

$\Rightarrow$ 2500 pixels $\Rightarrow$ $n = 2500$

OR $n = 7500$ (RGB)

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

$\therefore$ in order to create a QUADRATIC $h_\theta(x)$...

Quadratic Features $(x_i \times x_j)$ $\approx$ 3×10^6

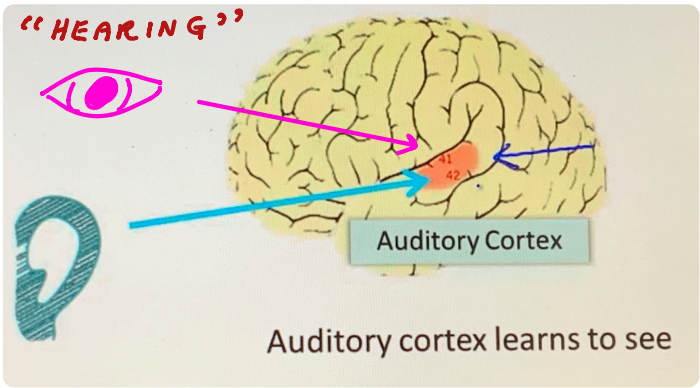
WAY TOO BIG

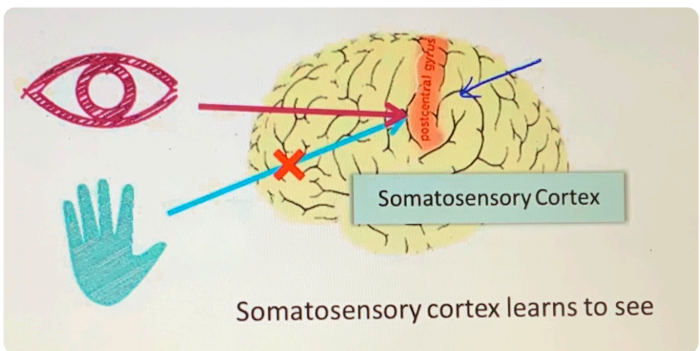
Neurons and the Brain

- ORIGIN: Algorithms that try mimic the brain
- The "one learning algorithm" hypothesis
- If you re-wire the signal from the ears to the auditory cortex to now come from the EYE

=> The auditory cortex LEARNS TO SEE

- The same happens with your somatosensory cortex, responsible for the sensation of touch





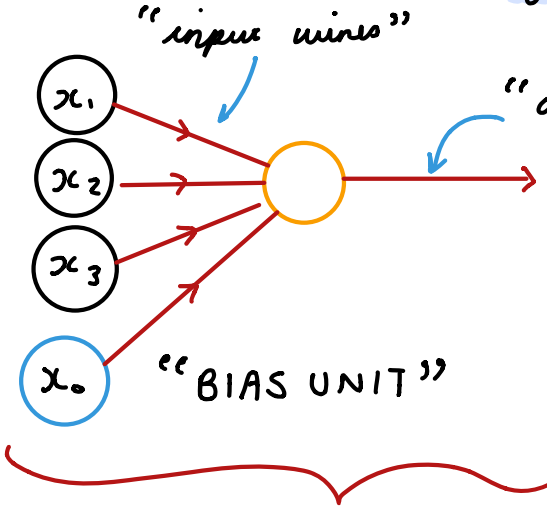
NEURO-REWIRING EXPERIMENTS

We wish to emulate how the brain does this, what algorithm does it use?

EXAMPLES

Model Representation

NEURON MODEL: Logistic unit



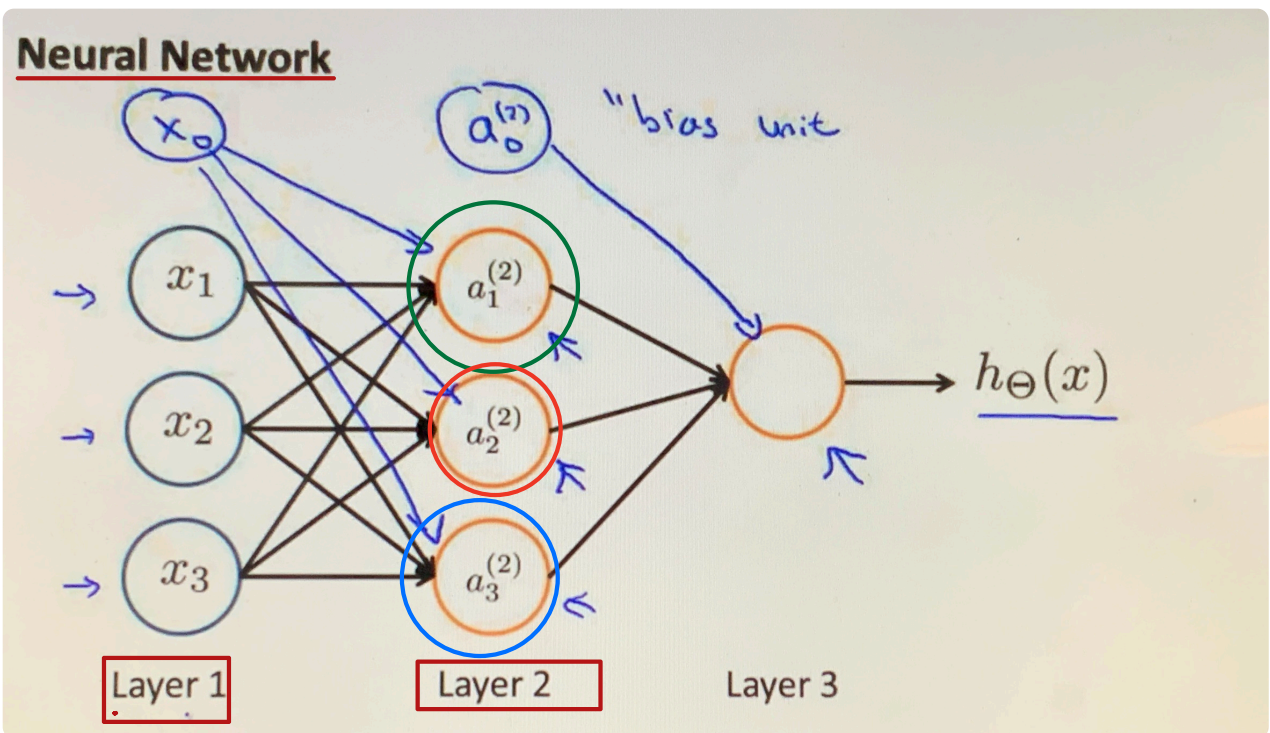
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

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

SINGLE NEURON (SIGMOID ACTIVATION FUNCTION)

WEIGHTS / PARAMETERS

- A neural network can simply be characterized as a collection of these neurons:



INPUT — HIDDEN — OUTPUT

x — y

"3 units" "3 units"

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

NOMENCLATURE:

* $a_i^{(j)}$ = ACTIVATION of unit i in layer j
* $\Theta^{(j)}$ = matrix of weights controlling function mapping from layer $j \longrightarrow j+1$

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

• In general, if a network has s_j units in layer j, s_{j+1} units in layer $j+1$, then

$$\Theta^{(j)} \in \mathbb{R}^{s_{j+1} \times (s_j + 1)}$$

Layer 1: $s_1 = 3$
Layer 2: $s_2 = 3$
} $\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$

Layer 3: $s_3 = 1$
} $\Theta^{(2)} \in \mathbb{R}^{1 \times 4}$

Representing the Hypothesis

VECTORIZATION

$a_1^{(3)} = g\{\Theta$

• let $a_1^{(2)} = g(z_1^{(2)})$ where

$$z_1^{(2)} = \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 = \left[\Theta_{1j}^{(1)}\right]^T x$$

• The same can be done for all activation nodes

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

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

$\Rightarrow$ Add $a_0^{(2)} = 1 \;\therefore\; \underline{a}^{(2)} \in \mathbb{R}^4$

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

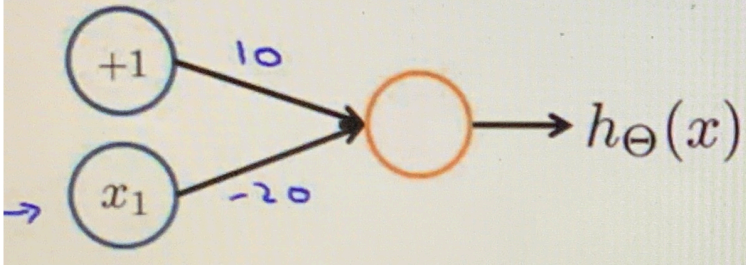
- In general: ↳ g applied element-wise

$$\underline{a}^{(i)} = g(\underline{z}^{(i)}) = g(\underline{\Theta}^{(i-1)} \underline{a}^{(i-1)})$$

FORWARD PROPAGATION $a_1^{(3)} = g(\theta^{(2)} a^{(2)})$

- Essentially, multiple logistic regressions where the algorithm tends towards the most OPTIMAL FEATURES before the final output layer performs logistic regression with those $\underline{a}^{(N-1)}$ for N layers.

Negation: NOT x_1



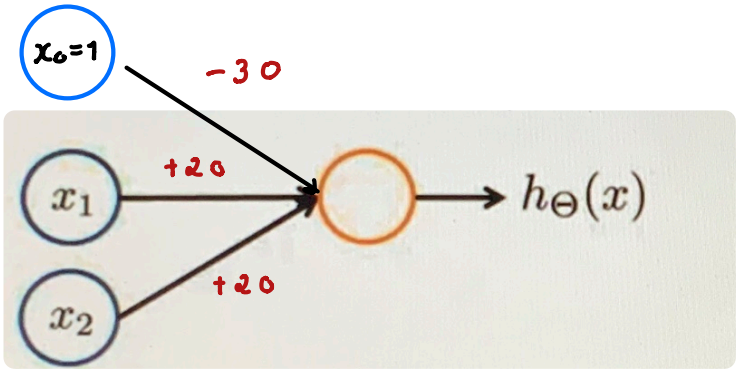
x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

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

- Simply place 'large' negative weight infront of the feature / variable you wish to negate

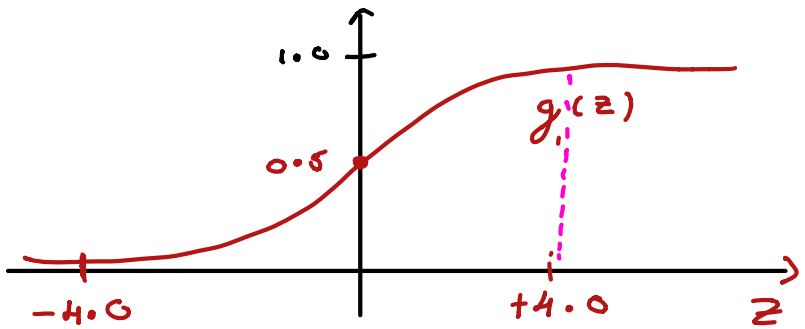
AND GATE EXAMPLE

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



- Assigning weights of $\underline{\Theta} = (-30, +20, +20)$

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

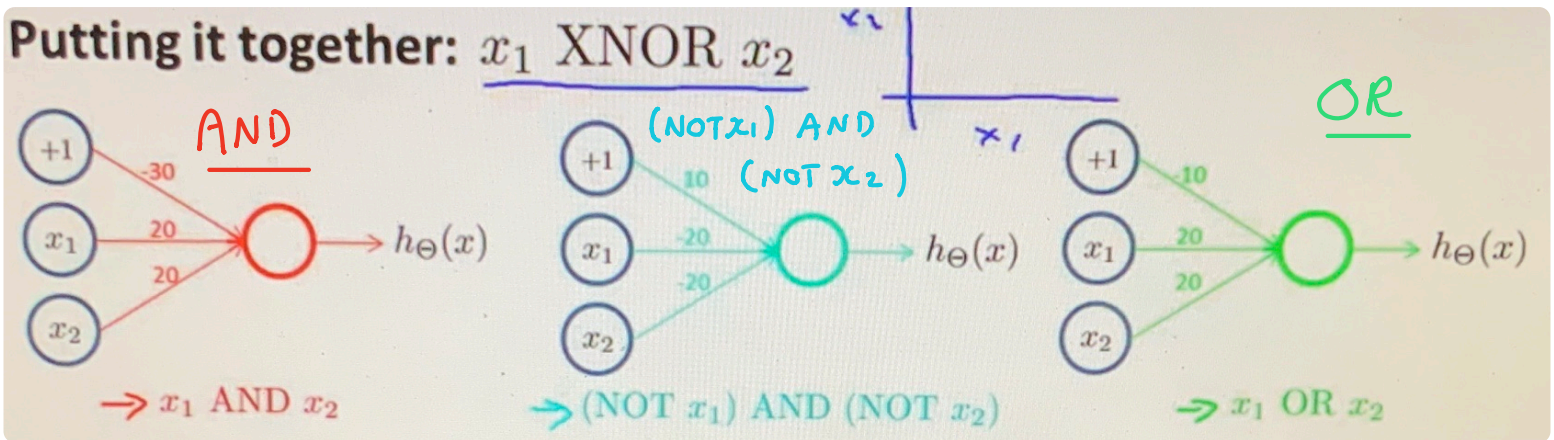


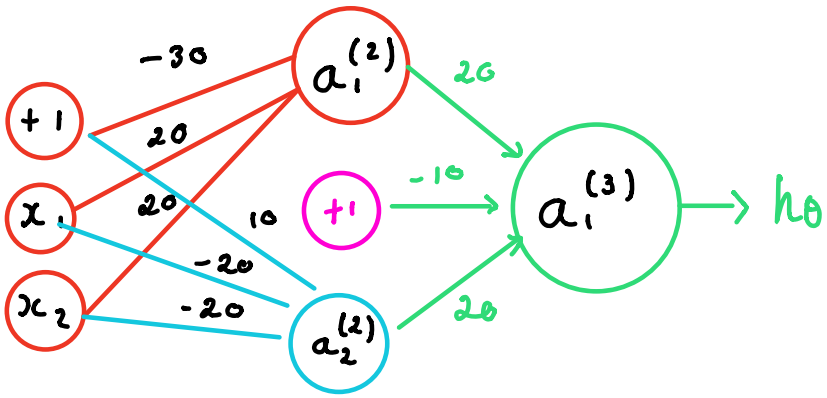
x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

$\therefore h_\Theta \approx x_1$ AND x_2

- If you set

Θ_0 : -30	⟶ -10
h_Θ : AND	⟶ OR



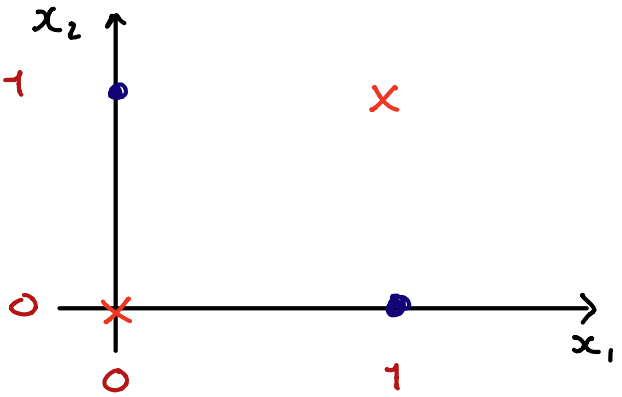


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

$$\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{bmatrix};$$

$$\Theta^{(2)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix};$$

- This neural network composed of simple functions has created a non linear hypothesis for us to classify this data



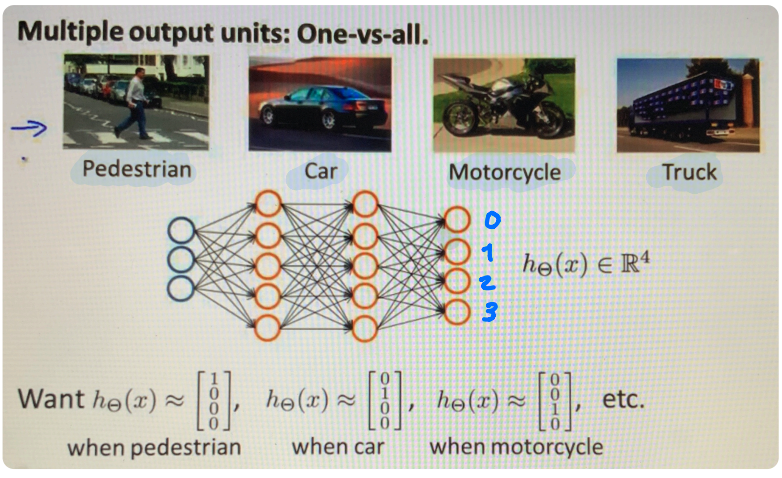
x_1 XNOR x_2

X = 1 ; • = 0

ANN: Multiclass Classification

- Extension of one vs. all logistic regression algorithm
- The output can now be one of 4 outcomes

∴ $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ when pedest.



$\therefore h_\theta(x) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when car etc.

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

  where $y^{(i)}$ is one of:

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

person car bike cruck

↳ Rather than $y = \{1, 2, 3, 4\}$ as in LOG REG.

- We want $h_\theta \approx y$