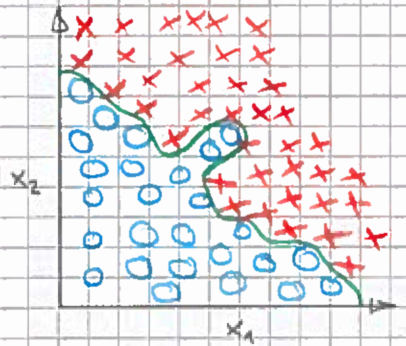
8. NEURAL NETWORKS – NON LINEAR HYPOTHESIS



$$g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1 x_2 + \Theta_4 x_1^2 x_2 + \Theta_5 x_1^3 x_2 + \Theta_6 x_1 x_2^2 + \ldots)$$

x_1 = size
x_2 = # bedrooms
x_3 = # floors
x_4 = age
⋮
x_{100}

n=100

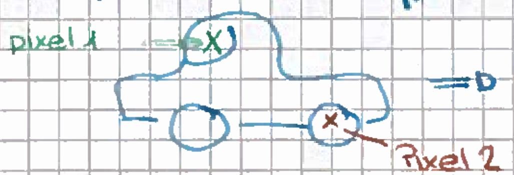
by using only second order therms
$x_1^2, x_1 x_2 \ldots \quad x_{100}$

Including all the quadratic features would mean to have ≈ 5000 features
⇒ grows $O(n^2)$ ⇒ $\frac{n^2}{2}$ ⇒ could lead to overfitting

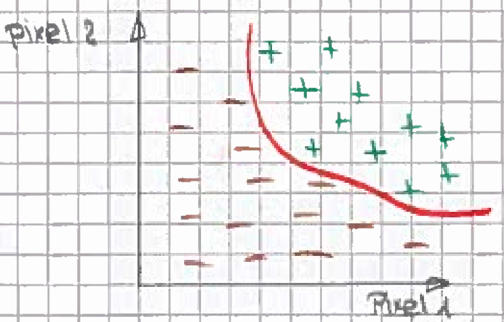
just using the $x_1^2, x_2^2, x^3 \ldots x_{100}^2$ does not allow to create a function as more complex functions. would not fit the complex dataset

while using cubic features it would be about 170'000 features for n = 100 features $O(n^3)$

Why is non linear hypothesis relevant?



⇒ Learning Algorithm



+ Cars
− Non Cars

⇒ 50 × 50 pixels images ⇒ 2500 pixels
n = 2500 (7500 RGB)

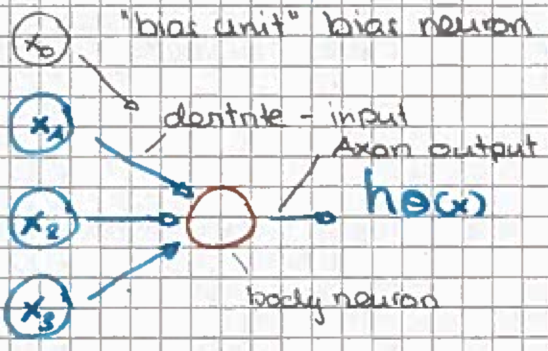
$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{" 2 "} \\ \text{" 3 "} \\ \vdots \\ \text{pixel } n \text{ itensity} \end{bmatrix}$$

Value 0 - 255

2500

by using all quadratic features ($x_i \times x_j$)
≈ 3 million features per training example
is computational expensive
Not good for algorith. with lot of features

Neuron model: Logistic unit



$\Rightarrow x_0 = 1$

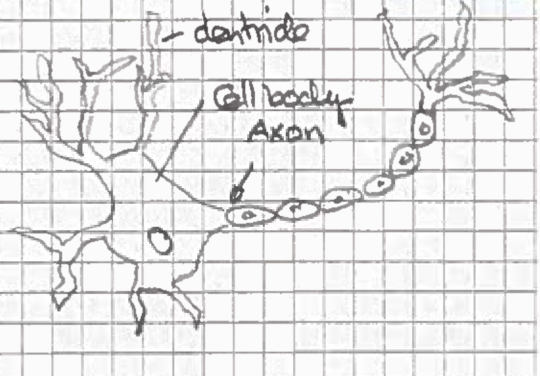
$$\Rightarrow h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

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

Sigmoid (logistic) activation function.

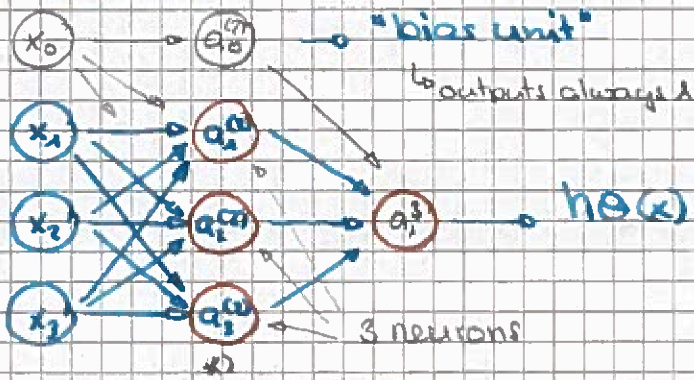
$$g(z) = \frac{1}{1+e^{-z}}$$

sometimes called "weights" - parameter of model



Neuron

Neural Network



Layer 1   Layer 2   Layer 3

≙ x   ≙   ≙ y

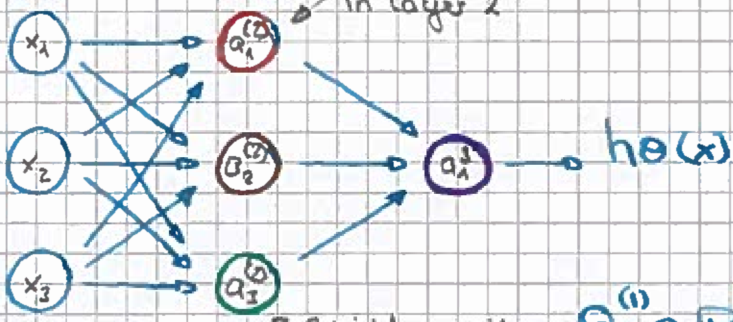
input lay.   hidden lay.   output lay.

*) you don't observe the values processed in the hidden layer

Model Representations I

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

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



$\Theta^{(1)} \in \mathbb{R}^{3\times4}$ dim. Matrix

a "activation" is the value which is computed and output by the node

sigmoid

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

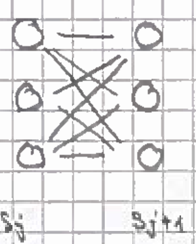
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

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

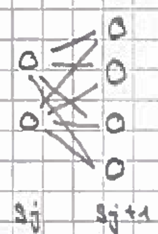
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

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

Example ⇒

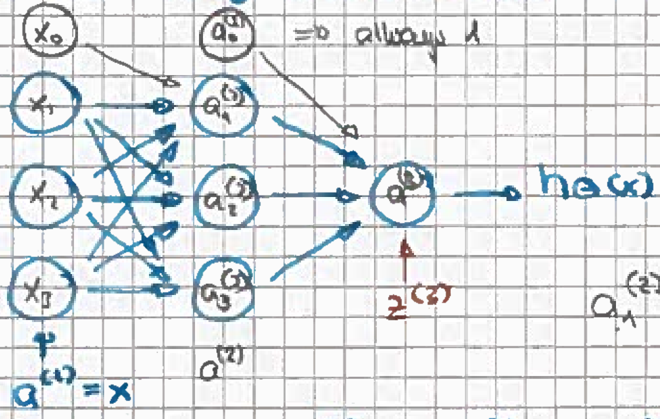


⇒ $\mathbb{R}$ 3 x 4



⇒ 4 x 3

Forward propagation: vectorized implementation



$$a_1^{(2)} = g(z_1^{(2)})$$

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

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

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

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

$$a_3^{(2)} = g(z_3^{(2)})$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ \cancel{z_4^{(2)}} \end{bmatrix}$$

Vectorized implementation

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

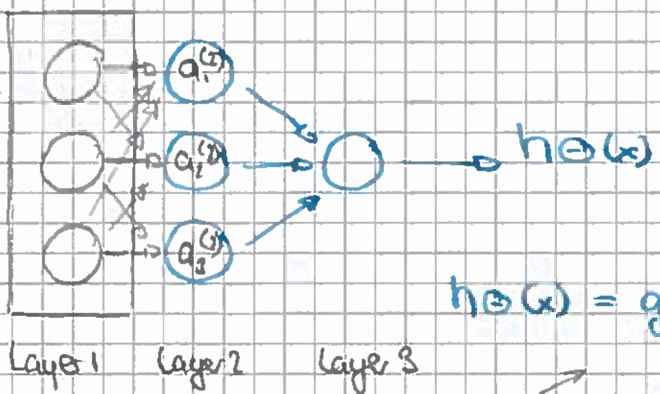
$$\underbrace{a^{(2)}}_{\mathbb{R}^3} = g(\underbrace{z^{(2)}}_{\mathbb{R}^3}) \Rightarrow \text{3 dim. Vector}$$

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

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

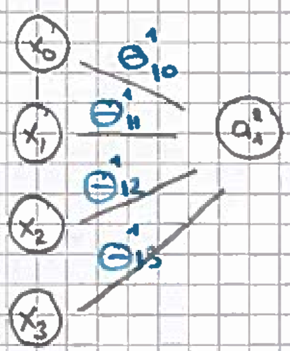
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

Neural Network learning its own features.



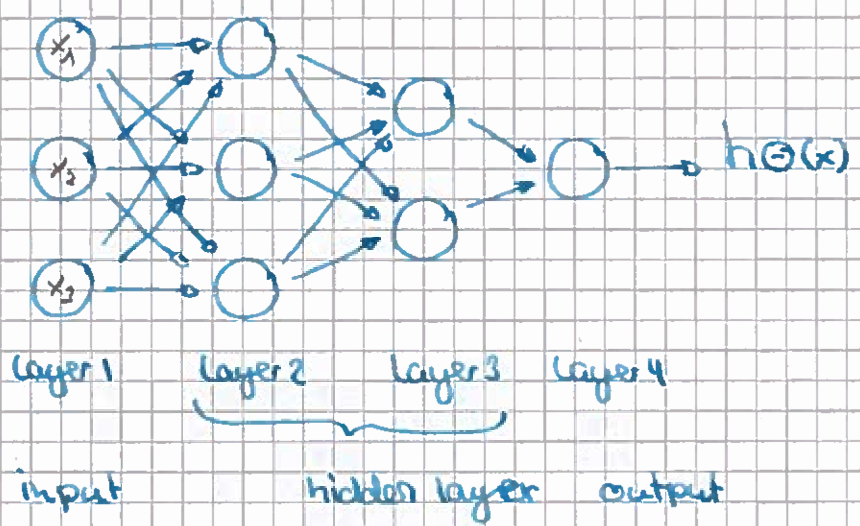
$$h_\Theta(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

it's just logistic regression except using the new features $a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$ instead of x_1, x_2, x_3

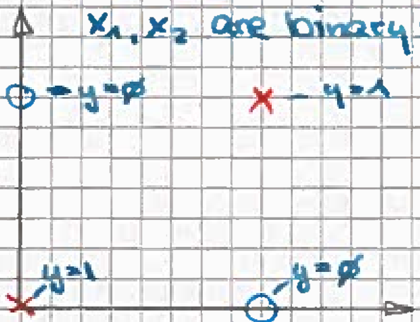


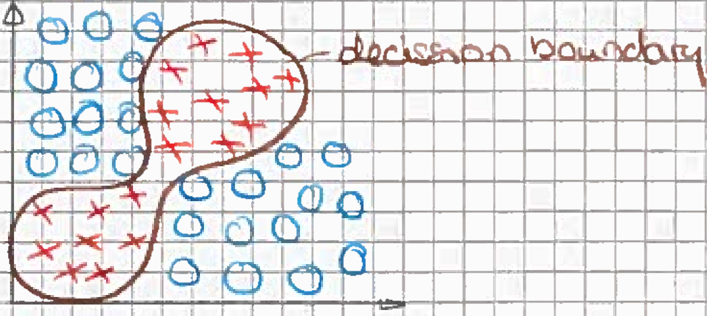
$$\to g(\Theta_{10}^1 x_0 + \Theta_{11}^1 x_1 + \Theta_{12}^1 x_2 + \Theta_{13}^1 x_3)$$

Other network architectures



Non linear classification example: XOR/XNOR





$y = x_1 \text{ XOR } x_2$

$x_1 \text{ XNOR } x_2$

$\text{NOT } (x_1 \text{ XOR } x_2)$

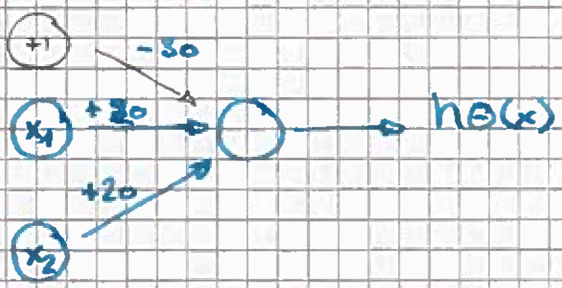
XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Simple example AND:

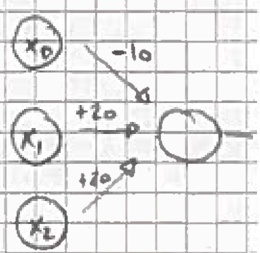
$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$



OR:

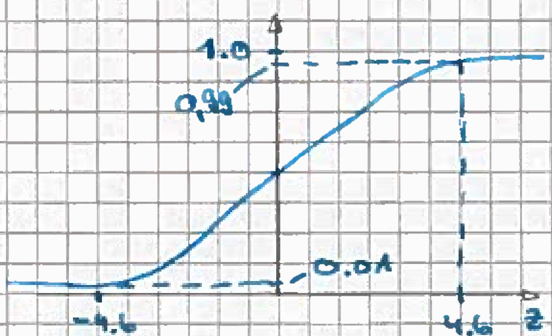
x_1	x_2	$h_\Theta(x)$
0	0	$g(-10) - 0$
0	1	$g(+10) - 1$
1	0	$g(+10) - 1$
1	1	$g(30) - 1$



$g(-10 \cdot 1 + 20 \cdot 0 + 20 \cdot 0) = g(-10$

$$\Rightarrow h_\Theta(x) = g(\underbrace{-30}_{\Theta_{10}^{(1)}} + \underbrace{20}_{\Theta_{11}^{(1)}} x_1 + \underbrace{20}_{\Theta_{12}^{(1)}} x_2)$$

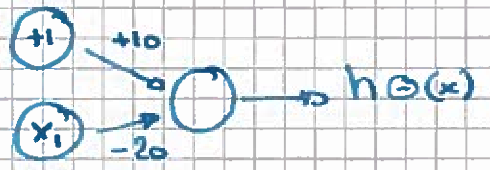
$(-30 + 20 \cdot 1 + 20 \cdot 0)$



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

$x_1 \text{ AND } x_2$

Negation: not x_1

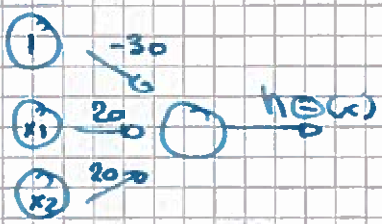


x_1	$h_\Theta(x)$
0	$g(10) \to 1$
1	$g(-10) \to 0$

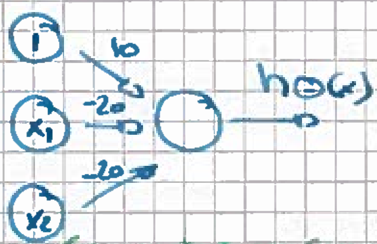
$$h_\Theta(x) = g(\Theta_{10} + \Theta_{11}x_1)$$

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

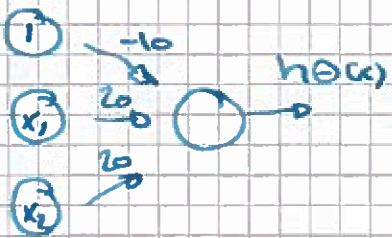
Putting it together: x_1 XNOR x_2



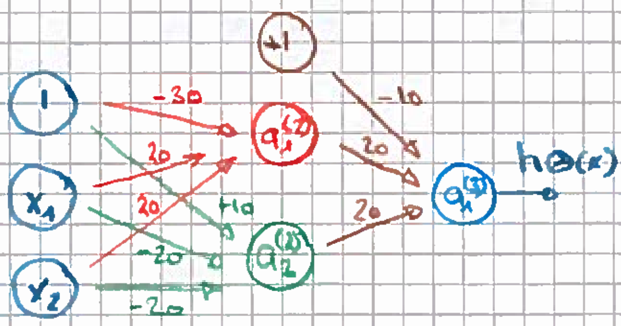
x_1 AND x_2



(NOT x_1) AND (NOT x_2)



x_1 OR x_2



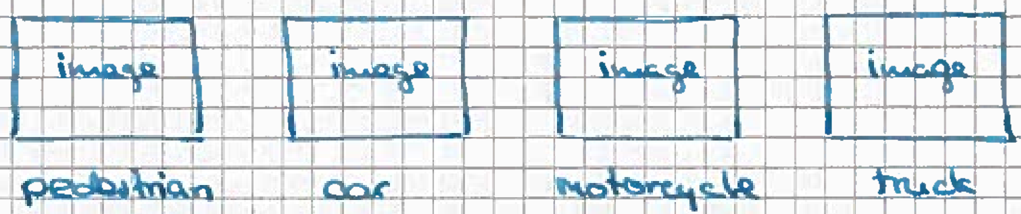
$\bar{x}_1$ AND $\bar{x}_2$

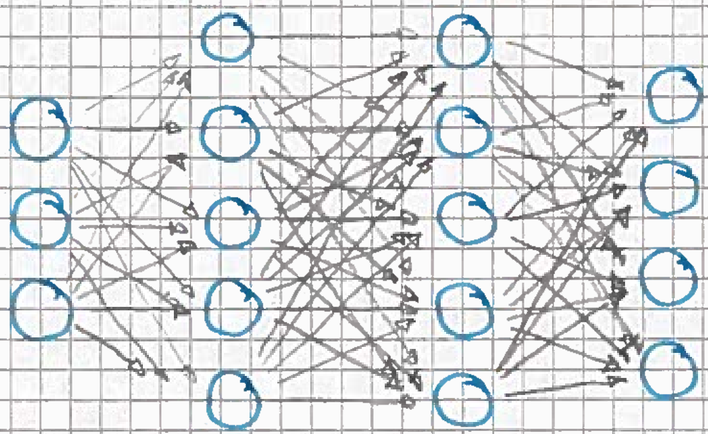
AND OR

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

Multiclass Classification

Multiple output units: One vs. all





output would be a vector

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

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

pedestrian, car, motorcycle, truck

Training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

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

pedestrian, car, motor-cycle, truck

image

$$h_\Theta(x^{(i)}) \approx y^{(i)}$$

$\mathbb{R}^4$

4 dym. vectors