REDES NEURONALES ARTIFICIALES

Dr. Jorge Hermosillo
Laboratorio de Semántica Computacional

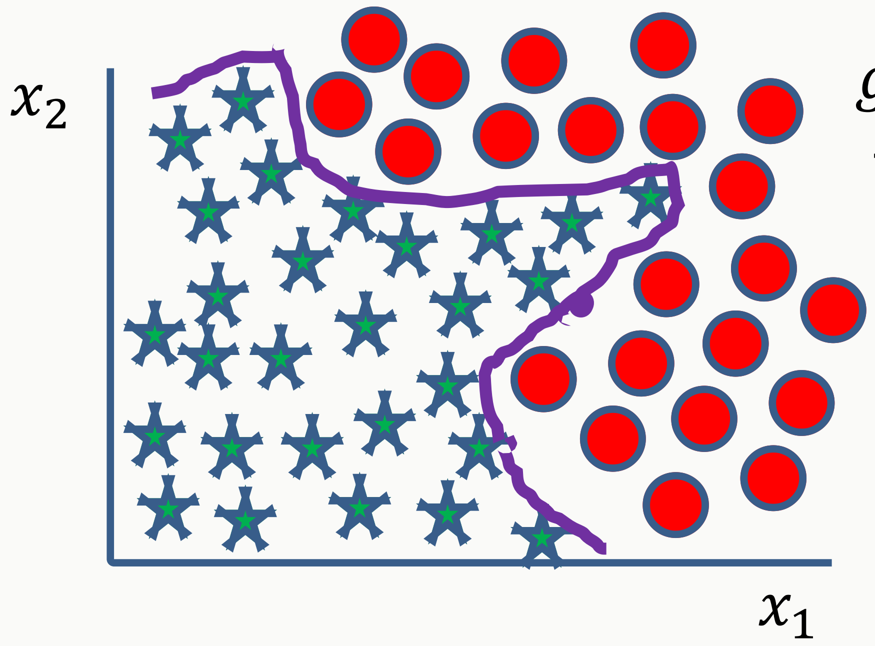
Centro de Investigación en Ciencias

Universidad Autónoma del Estado de Morelos

INTRODUCCIÓN

MOTIVACIÓN PARA UN NUEVO MODELO

- Hemos visto varios algoritmos de clasificación que pueden resolver problemas no lineales: ¿por qué otro más?



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

Problema con n=100 *features*

$$\left.\begin{array}{l} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{100} \end{array}\right\}$$

Términos de segundo orden

$\approx$ 5000 *features* ($O(n^2)$)

Términos de tercer orden

$\approx$ 170000 *features* ($O(n^3)$)

CLASIFICACIÓN DE IMÁGENES

Coche

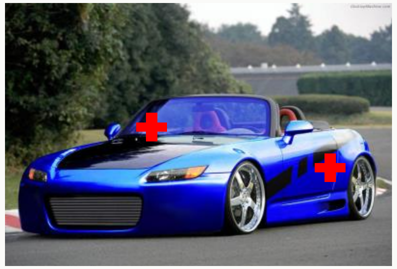
No coche

25	40	40	26	30	80	100
124	235	115	72	97	63	200
97	72	97	63	200	40	40
25	40	40	26	30	80	100
124	235	115	72	97	63	200
25	40	40	26	30	80	100
124	235	115	72	97	63	200
72	97	63	200	40	26	30

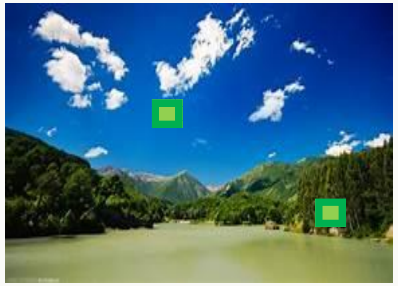
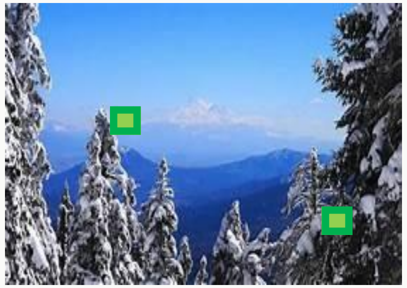
Valor de un pixel

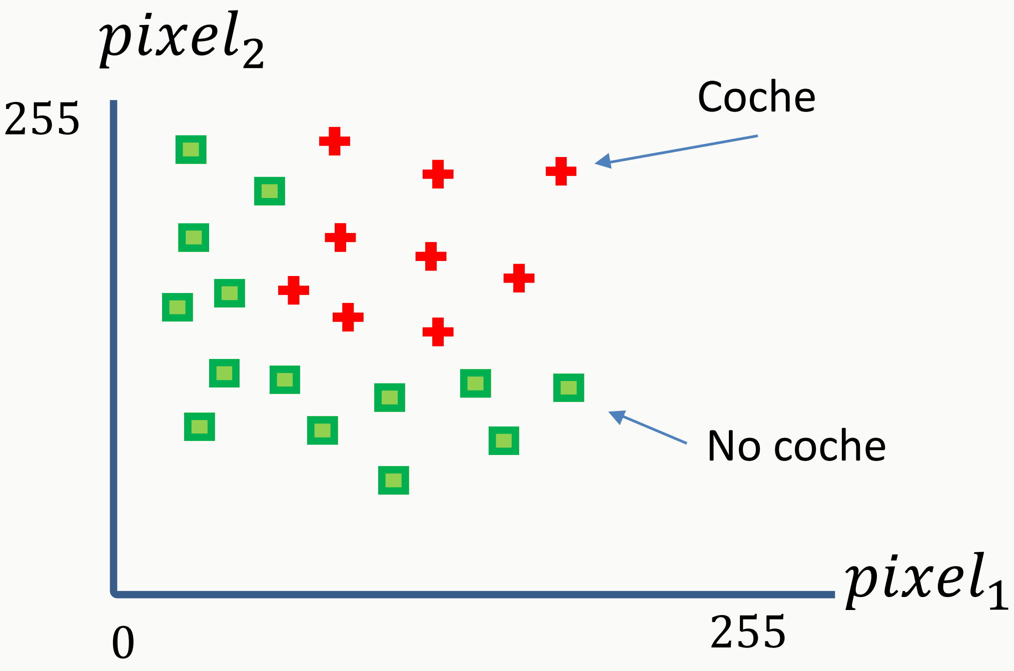
CLASIFICACIÓN DE IMÁGENES

Coche


No coche



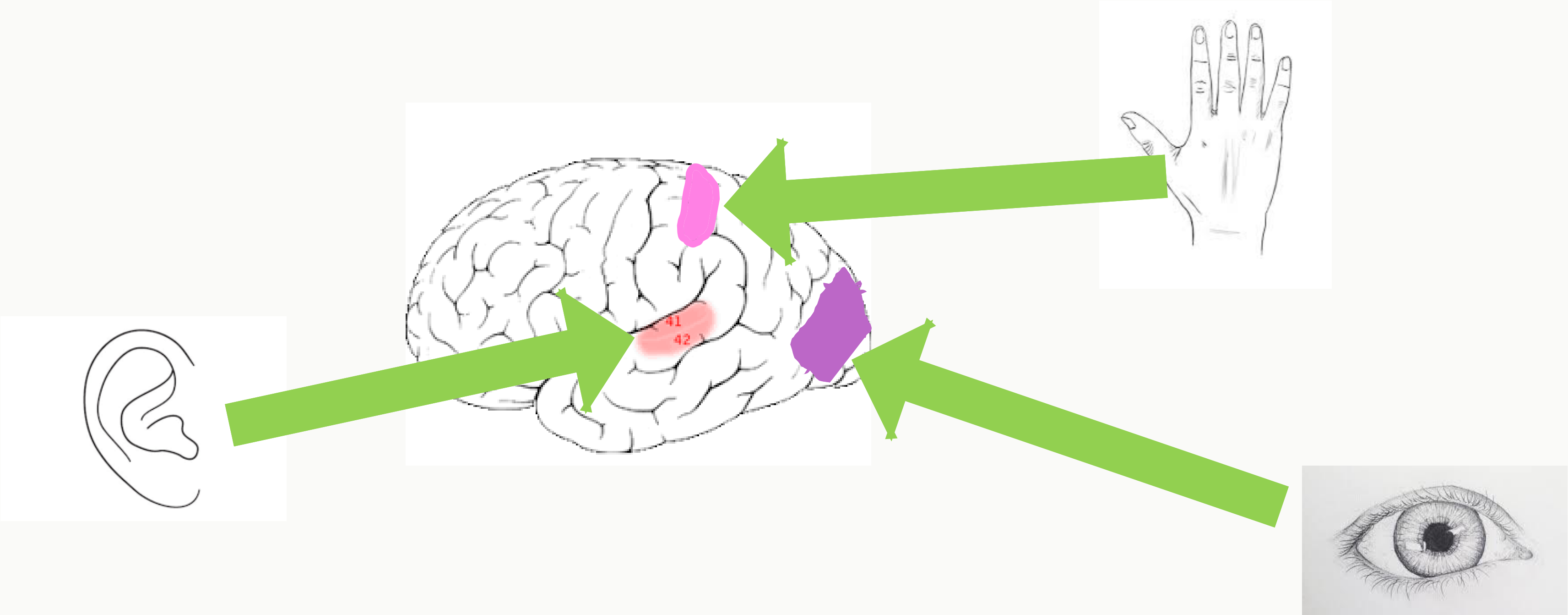
Si fueran imágenes de 50×50 pixeles → 2500 pixeles: (7500 si RGB)

$$\mathbf{x} = \begin{bmatrix} \text{Intensidad pixel}_1 \\ \text{Intensidad pixel}_2 \\ \vdots \\ \text{Intensidad pixel}_{2500} \end{bmatrix}$$

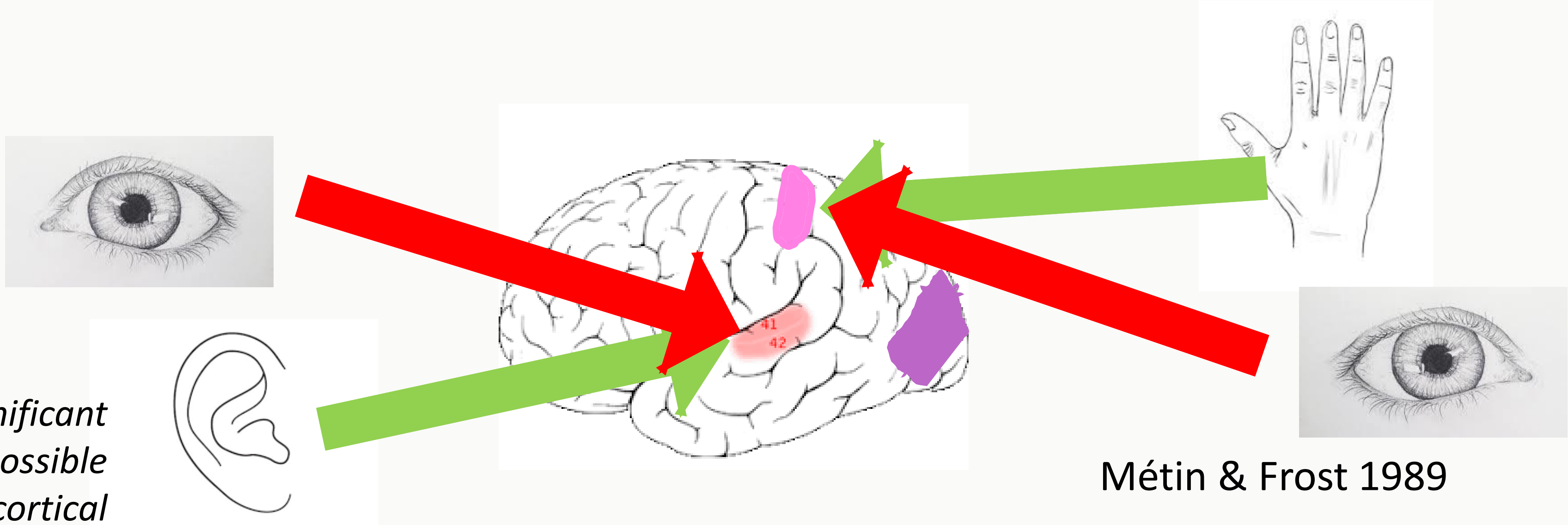
→ Features

Cuadráticas: ≈ 3 millones

EL CEREBRO COMO INSPIRACIÓN BIOLÓGICA: LA HIPÓTESIS DE UN SOLO ALGORITMO DE APRENDIZAJE



EL CEREBRO COMO INSPIRACIÓN BIOLÓGICA: LA HIPÓTESIS DE UN SOLO ALGORITMO DE APRENDIZAJE



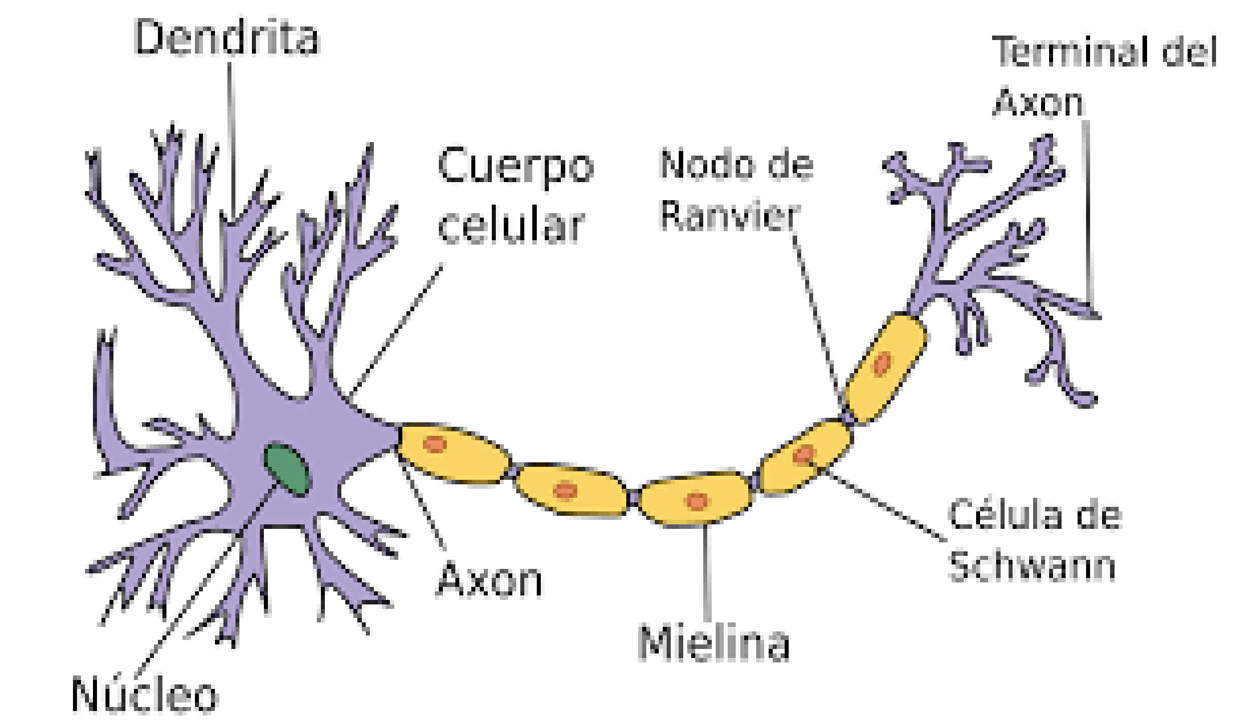
Roe et al. 1992

“These results have significant implications for possible commonalities in intracortical processing circuits between sensory cortices, and for the role of inputs in specifying intracortical circuitry.”

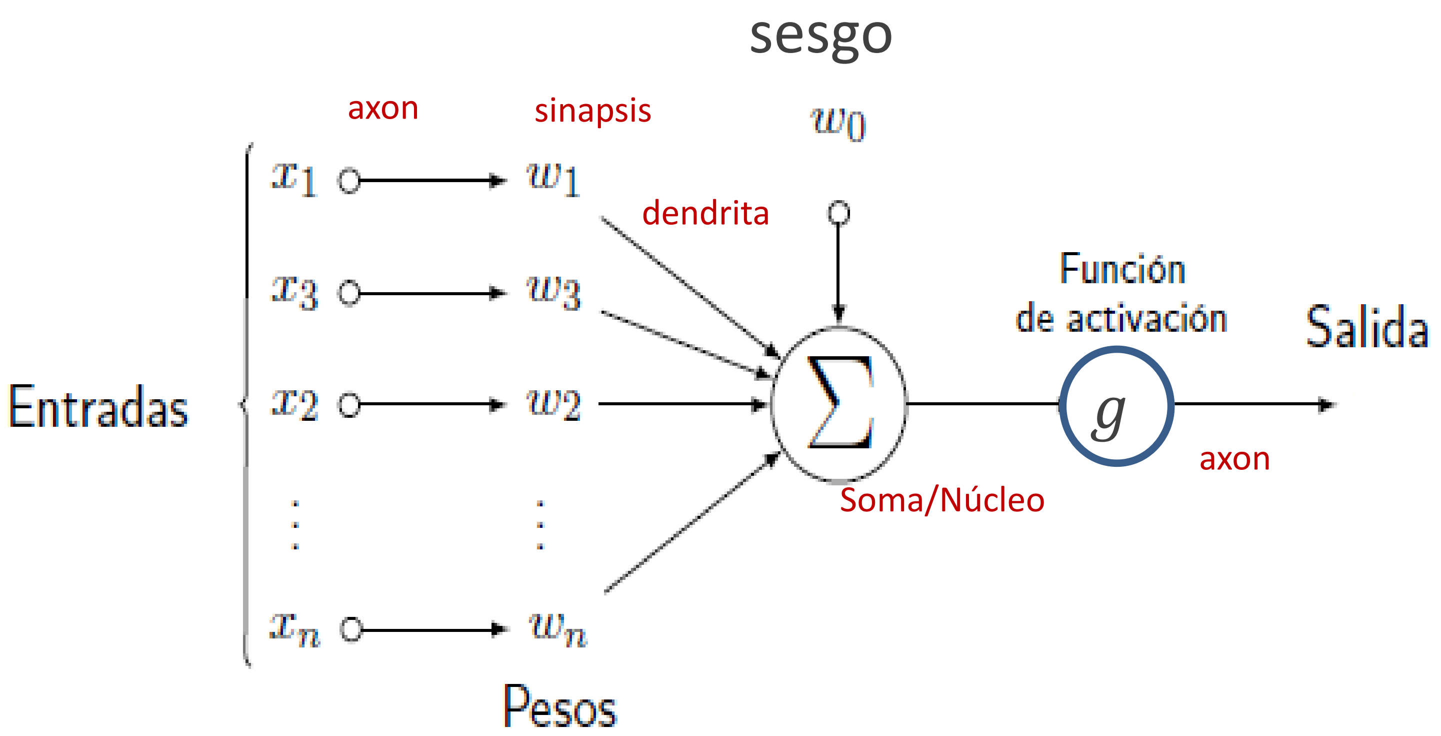
Métin & Frost 1989

“These results suggest that thalamic nuclei or cortical areas at corresponding levels in the visual and somatosensory pathways perform similar transformations on their inputs.”

NEURONA BIOLÓGICA Y RED NEURONAL

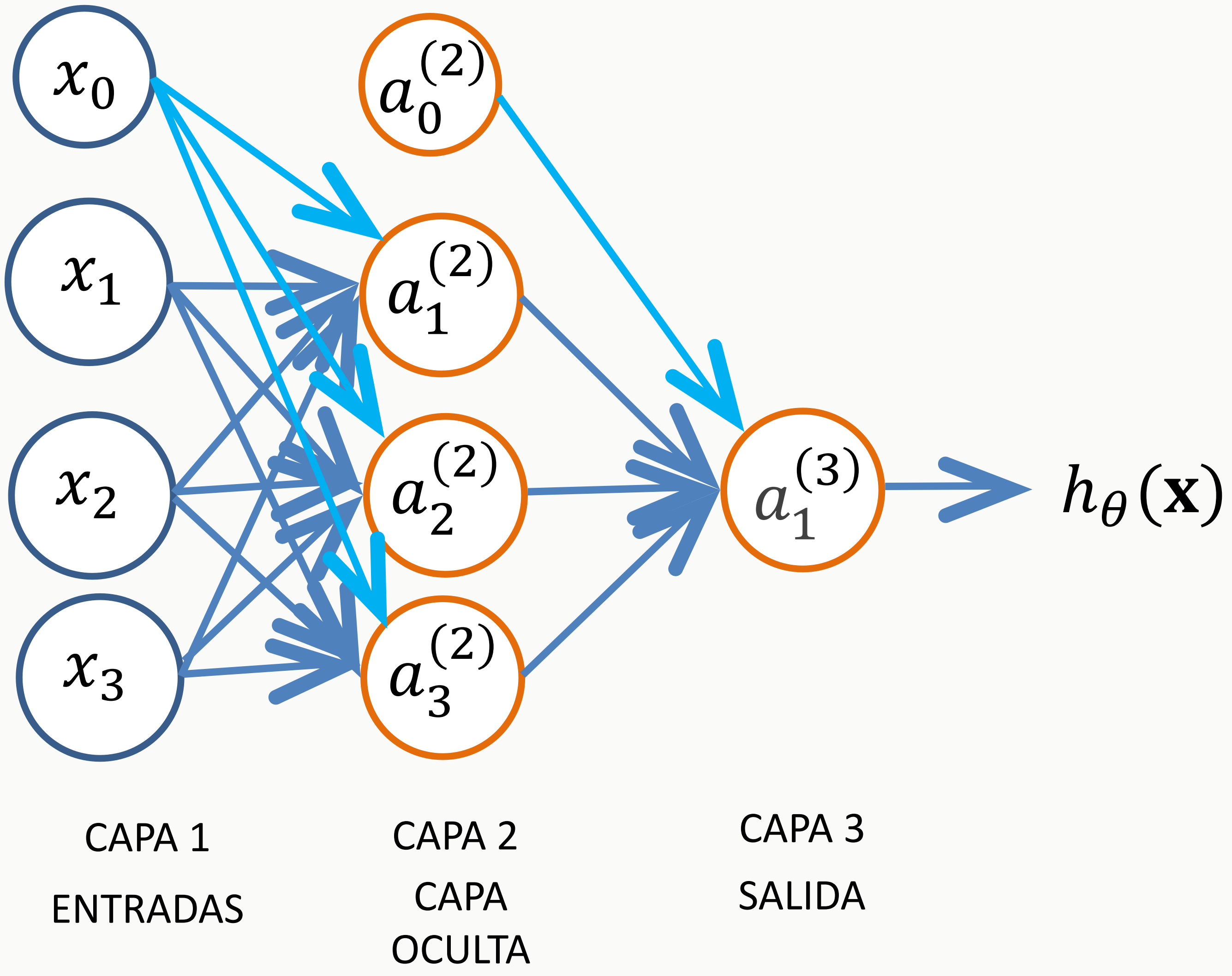


NEURONA ARTIFICIAL

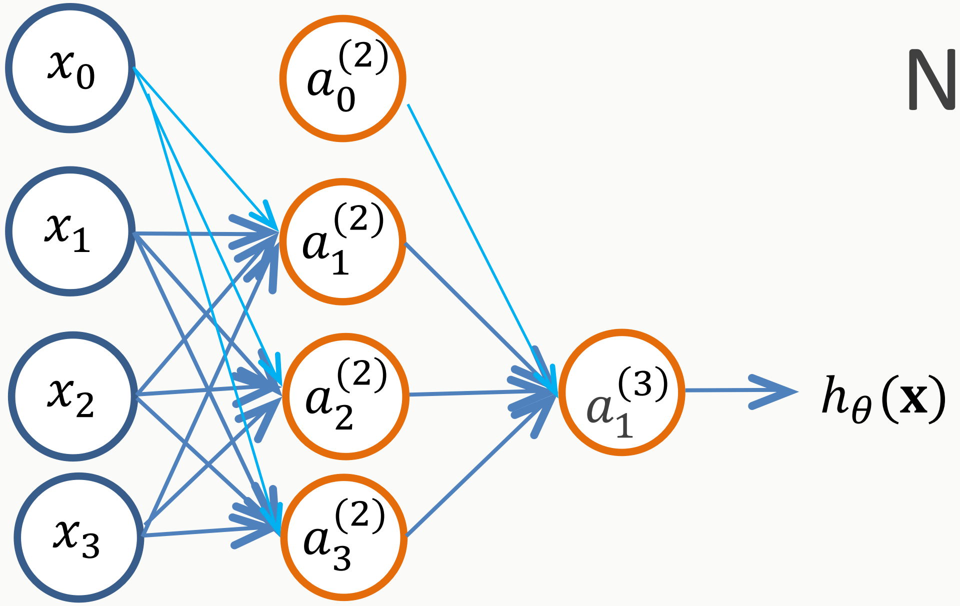


- Regresión logística: $g(z) \equiv \sigma(z) = \frac{1}{1+e^{-z}}$
- Perceptrón: $g(z) \equiv \text{sign}(z)$

RED NEURONAL ARTIFICIAL



NOTACIÓN



$a_i^{(j)}$= "activación" de la unidad i en la capa j

$\Theta^{(j)}$= matriz de pesos que controla la función de mapeo que va de la capa j a la capa $j+1$

Si la red tiene s_j unidades en la capa j, s_{j+1} en la capa $j+1$, entonces $\Theta^{(j)}$ será de dimensión $s_{j+1} \times (s_j + 1)$

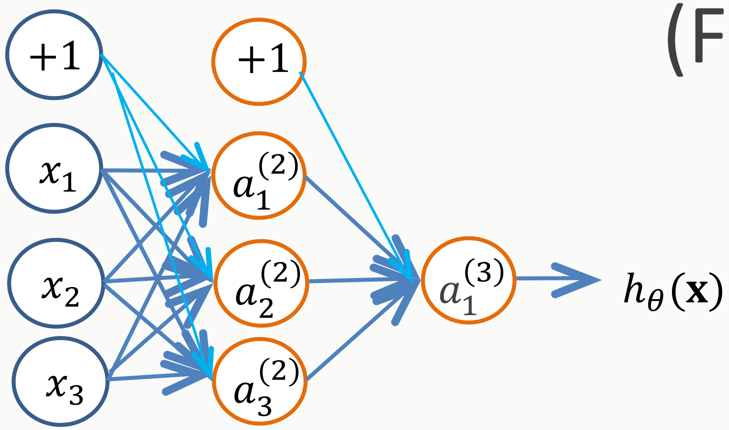
$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

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

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

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

PROPAGACIÓN DIRECTA (FORWARD PROPAGATION)



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

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

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

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

$$a^{(1)} = \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

En notación vectorial:

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

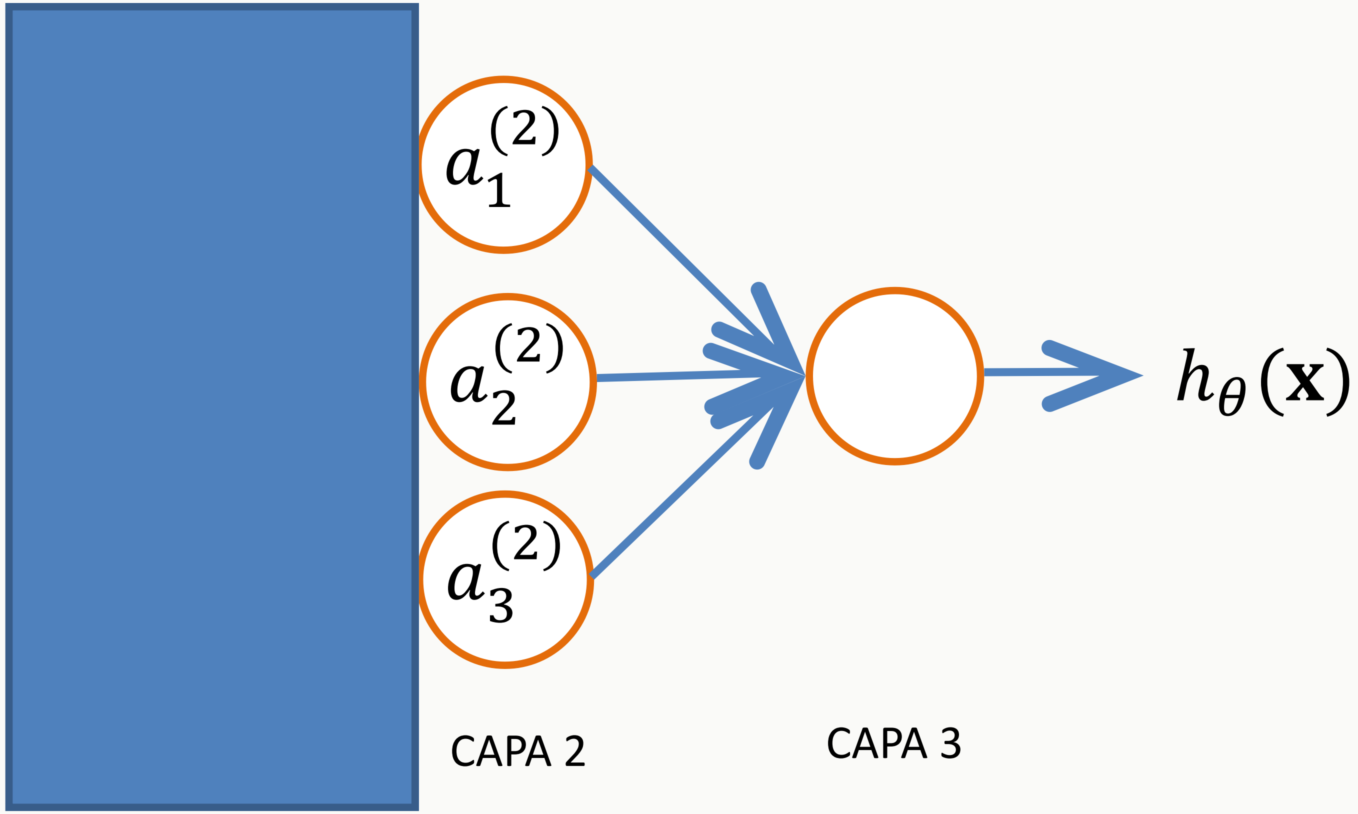
$$a^{(2)} = g\left(z^{(2)}\right)$$

AGREGA $a_0^{(2)} = 1$

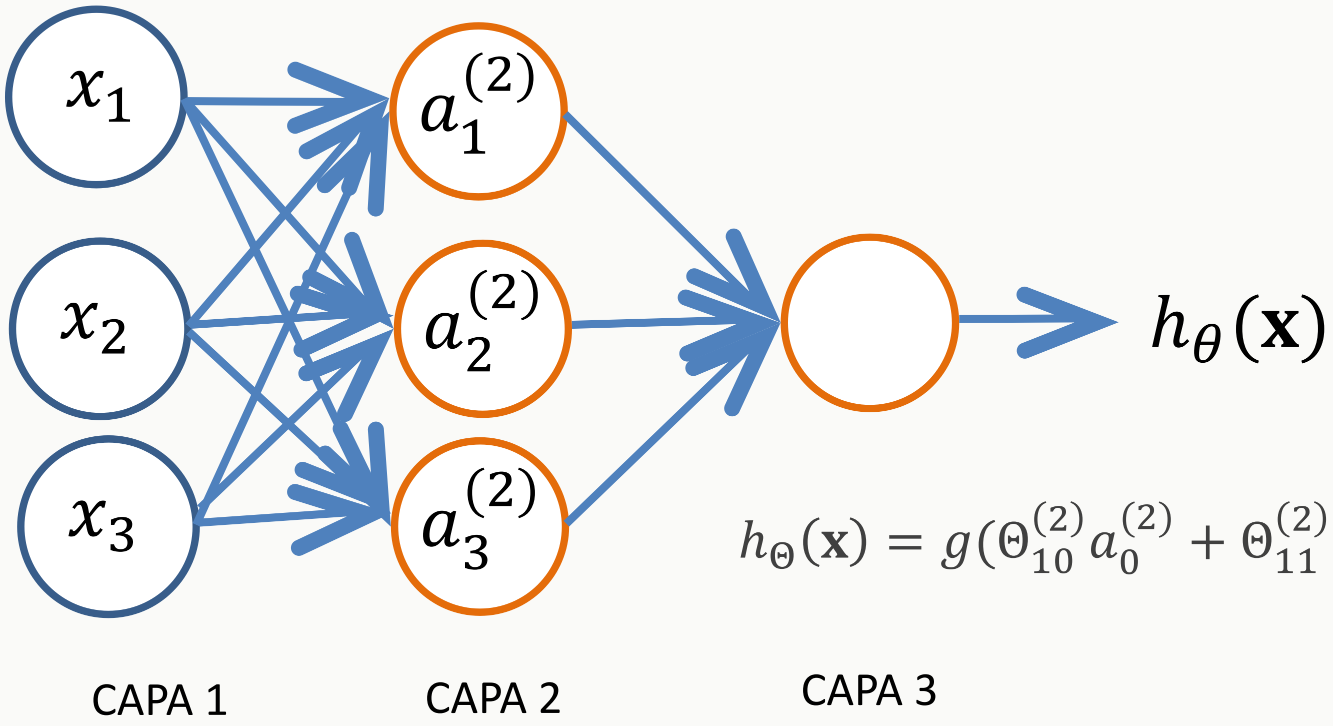
$$z^{(3)} = \Theta^{(2)}a^{(2)}$$

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

RED NEURONAL APRENDE SUS PROPIAS FEATURES

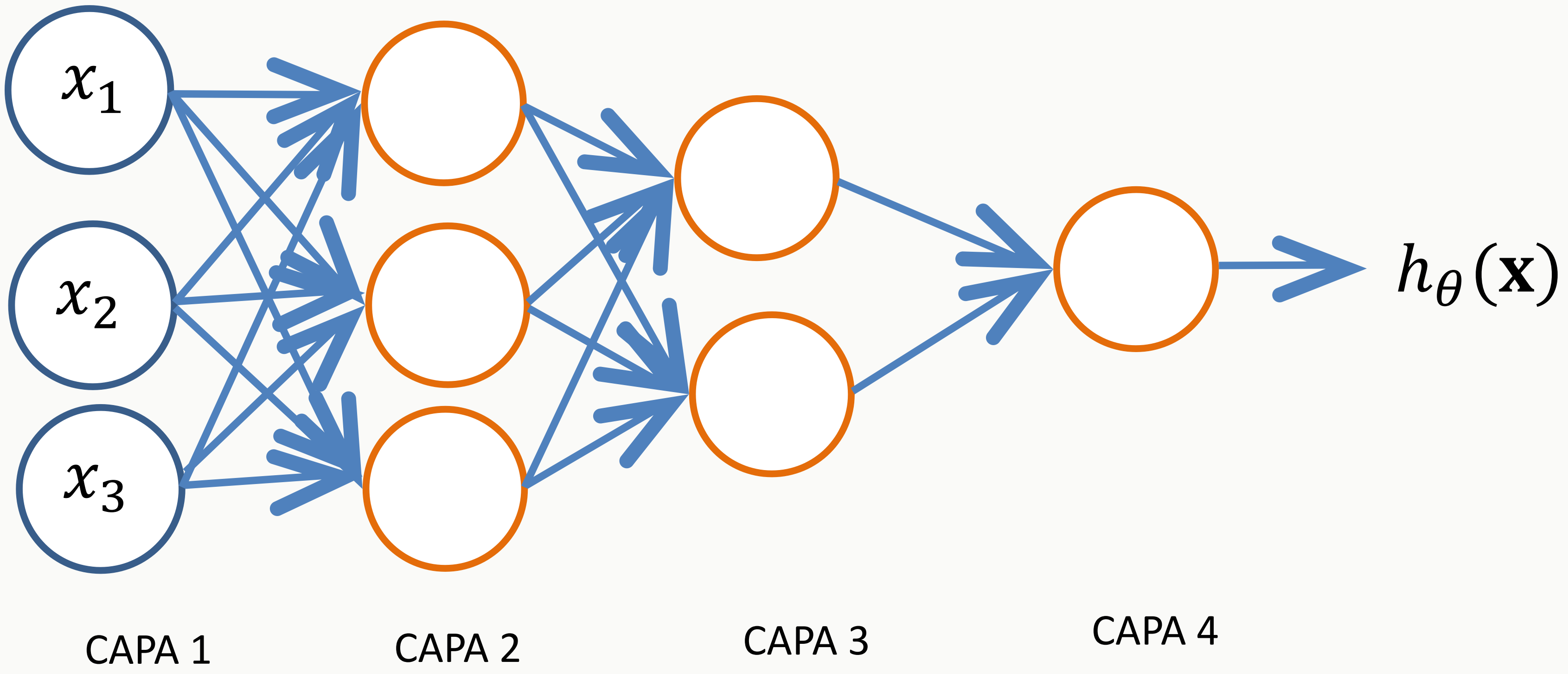


RED NEURONAL APRENDE SUS PROPIAS FEATURES



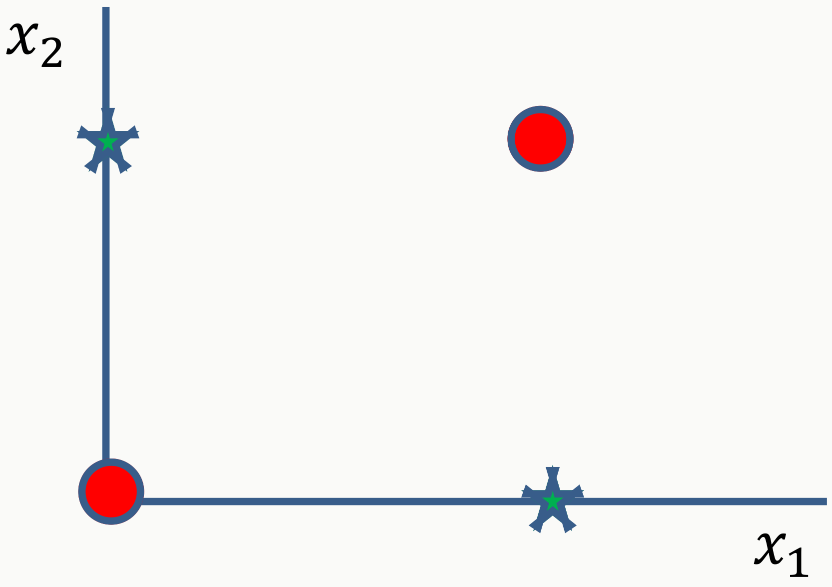
$$h_\Theta(\mathbf{x}) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

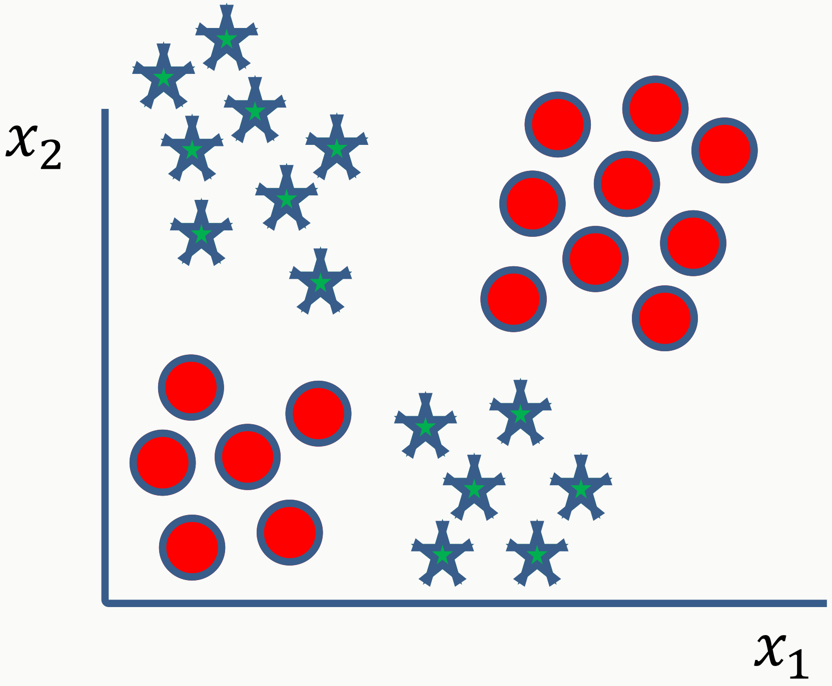
ARQUITECTURAS DE REDES NEURONALES



Dense (*Fully-Connected*) network

EJEMPLO DE CLASIFICACIÓN NO LINEAL



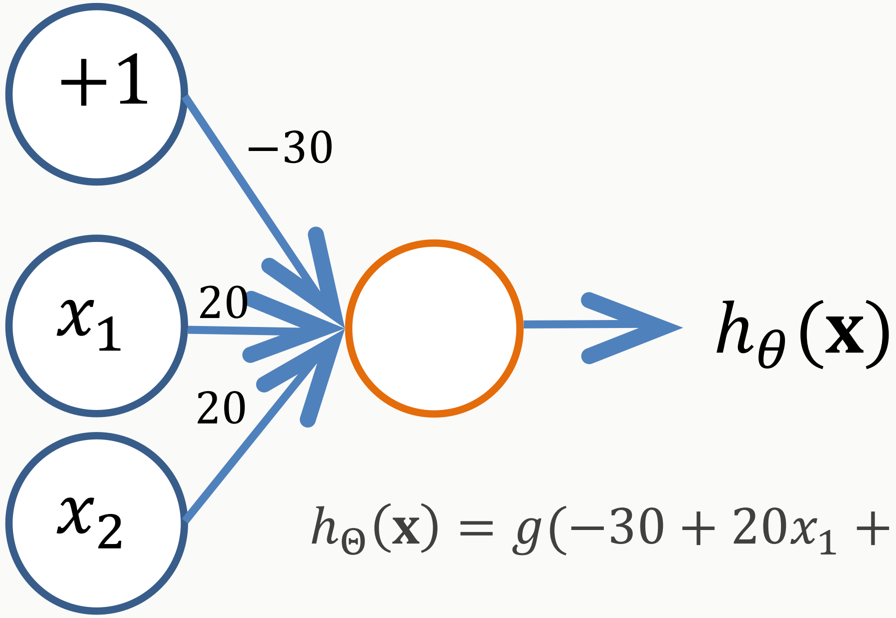


$$y = x_1 \text{XNOR}\, x_2 = \text{NOT}(x_1 \text{XOR}\, x_2)$$

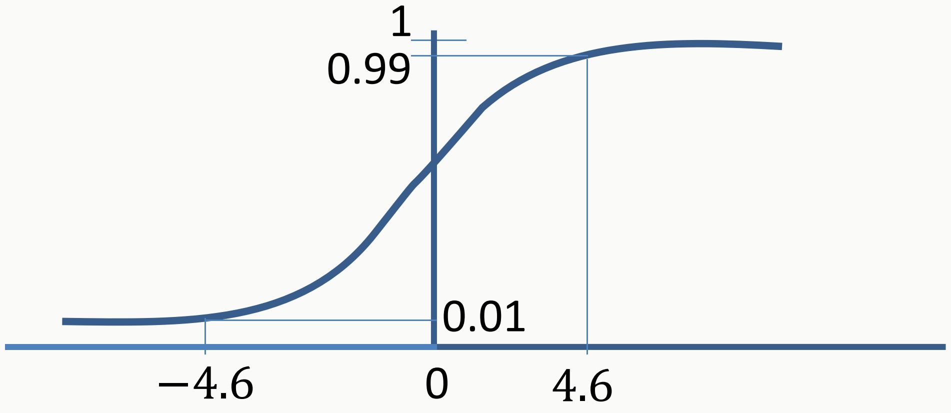
EJEMPLO SENCILLO: AND

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

$y = x_1 \text{AND } x_2$

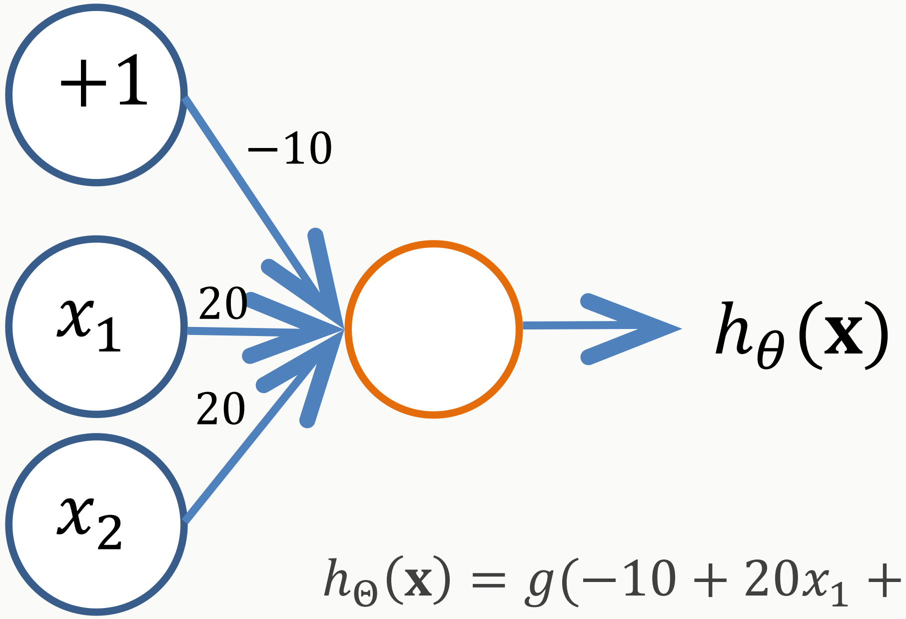


$h_\Theta(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$



x_1	x_2	$h_\Theta(\mathbf{x})$
0	0	
0	1	
1	0	
1	1	

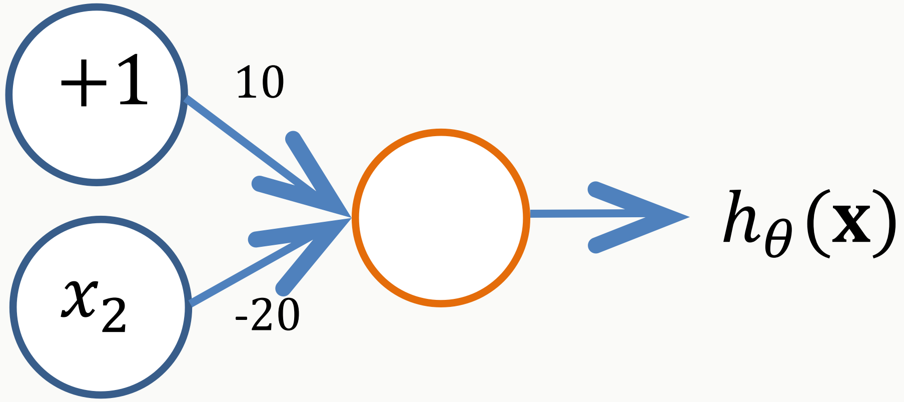
EJEMPLO SENCILLO: OR



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

x_1	x_2	$h_\Theta(\mathbf{x})$
0	0	
0	1	
1	0	
1	1	

EJEMPLO SENCILLO: NOT

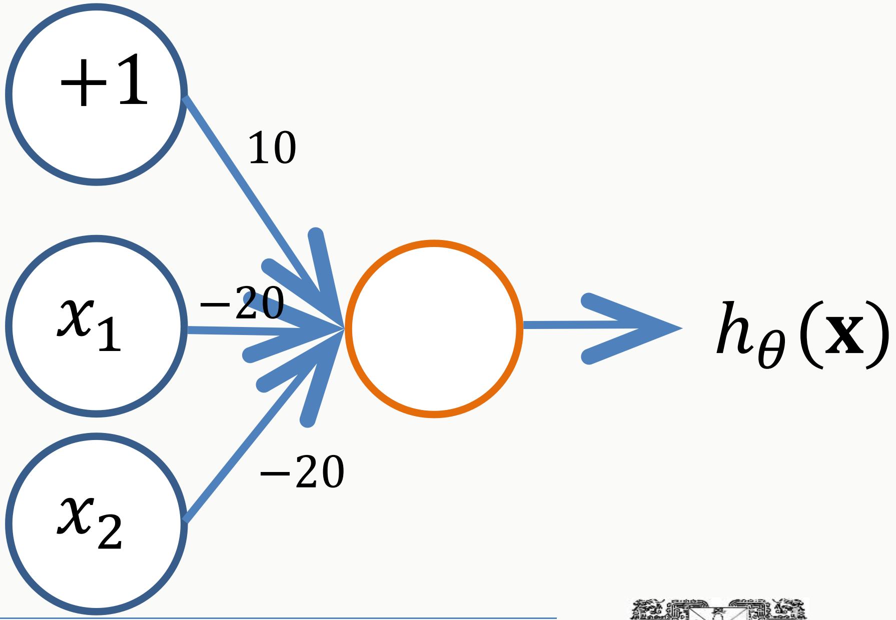


x_1	$h_\Theta(\mathbf{x})$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

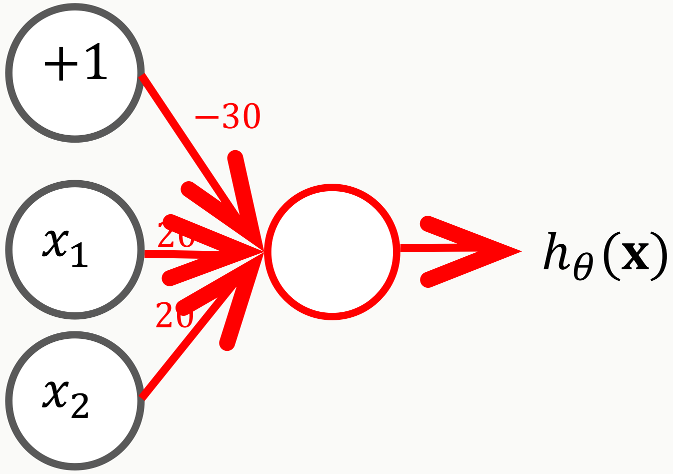
$$h_\Theta(\mathbf{x}) = g(10 - 20x_1)$$

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

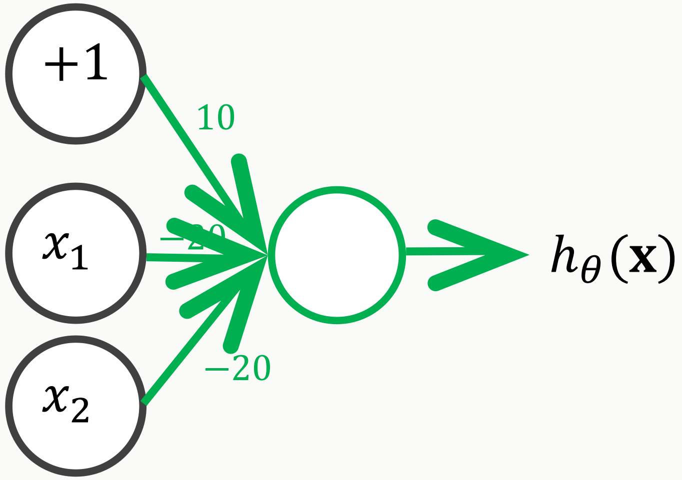
$$y = 1 \; ssi \; x_1 = x_2 = 0$$



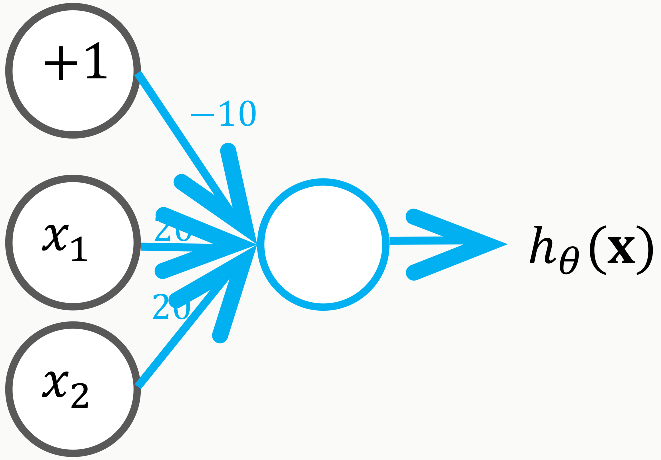
XNOR



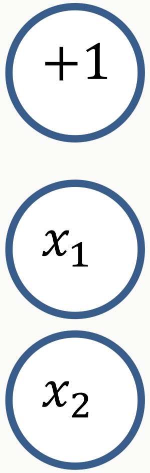
x_1 AND x_2



(NOT x_1) AND (NOT x_2)



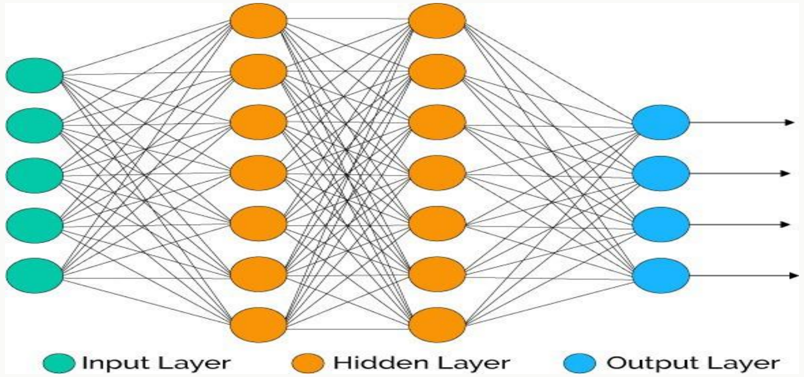
x_1 OR x_2



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

MÚLTIPLES SALIDAS: ONE-VS-ALL

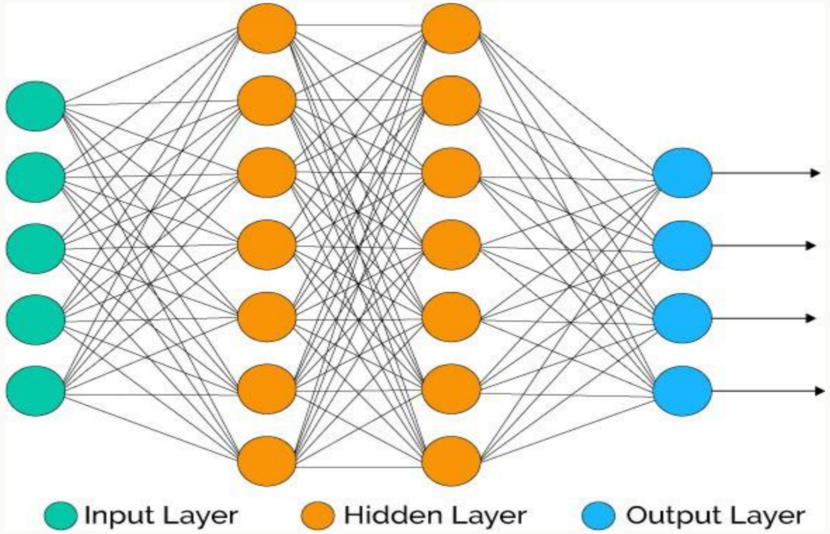
$$\mathbf{h}_\Theta(\mathbf{x}) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{h}_\Theta(\mathbf{x}) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{h}_\Theta(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{h}_\Theta(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



persona
coche
árbol
camión

$$\mathbf{h}_\Theta(\mathbf{x}) \in \mathbb{R}^4$$

MÚLTIPLES SALIDAS: ONE-VS-ALL



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

Queremos: $\mathbf{h}_\Theta(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $\mathbf{h}_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ Etc.

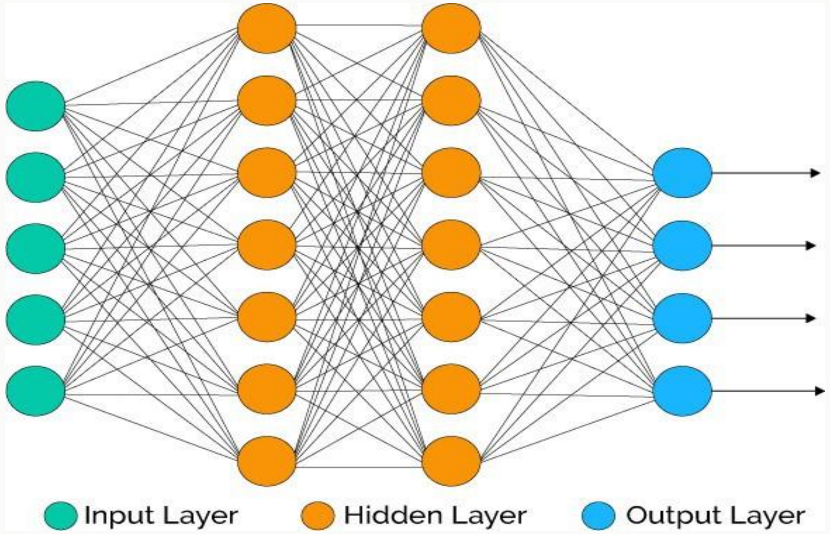
Cuando peatón coche

Conjunto de entrenamiento: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ es uno de: $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

$persona$ $coche$ $árbol$ $camión$

DOS TIPOS DE CLASIFICACIÓN



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

L es el número total de capas

s_l es el número de neuronas (sin contar las neuronas de biasen la capa l

<u>Clasificación binaria</u>

$y = 0 \; o \; 1$

<u>1 salida</u>

<u>Clasificación multiclase (K clases)</u>

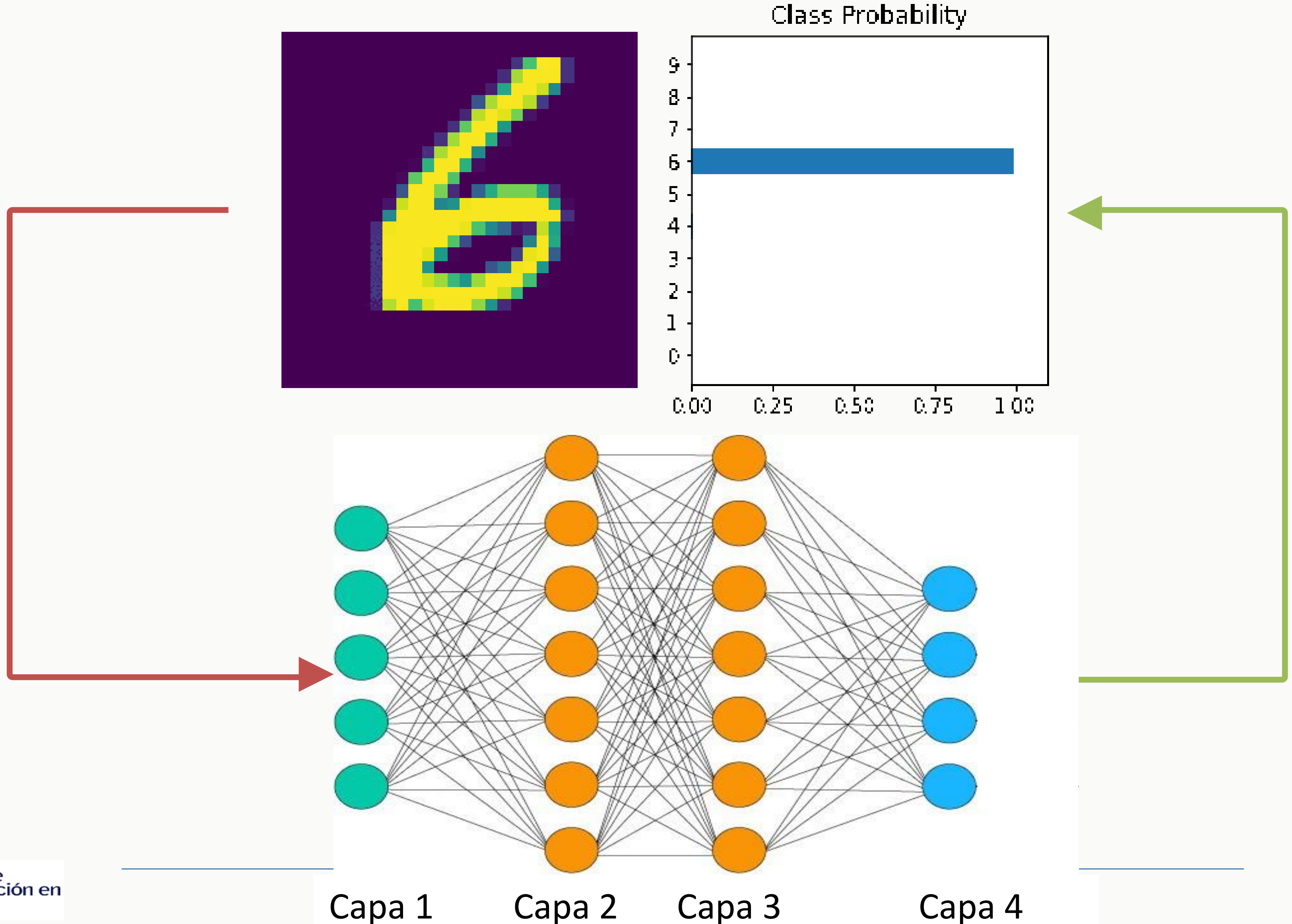
$y \in \mathbb{R}^{\boldsymbol{K}}$

$e.g.$ $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$ $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$

$persona$ $\quad coche$ $\quad árbol$ $\quad camión$

<u>K salidas</u>

CLASIFICACIÓN MULTICLASE



CLASIFICACIÓN CON REDES NEURONALES

FUNCIÓN DE COSTO

Regresión logística

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Red neuronal

$h_\Theta(\mathbf{x}) \in \mathbb{R}^{\boldsymbol{K}}$ $\quad \left(h_\Theta(\mathbf{x})\right)_i = i^{ésima}$ salida; i.e. $a_i^{(L)}$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right) \log\left(1 - \left(h_\Theta\left(x^{(i)}\right)\right)_k\right)\right]$$

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

GRADIENTE

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta\left(x^{(i)}\right)\right)_k + (1 - y_k^{(i)}) \log\left(1 - \left(h_\Theta\left(x^{(i)}\right)\right)_k\right)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

$$\min_{\Theta} J(\Theta)$$

Necesitamos calcular: $J(\Theta)$

$$\frac{\partial}{\partial\Theta} J(\Theta)$$

FORWARD PROPAGATION (FWP)

Supongamos un solo ejemplo de entrenamiento (x, y)

Propagación hacia adelante

$a^{(1)} = \mathbf{x}$

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

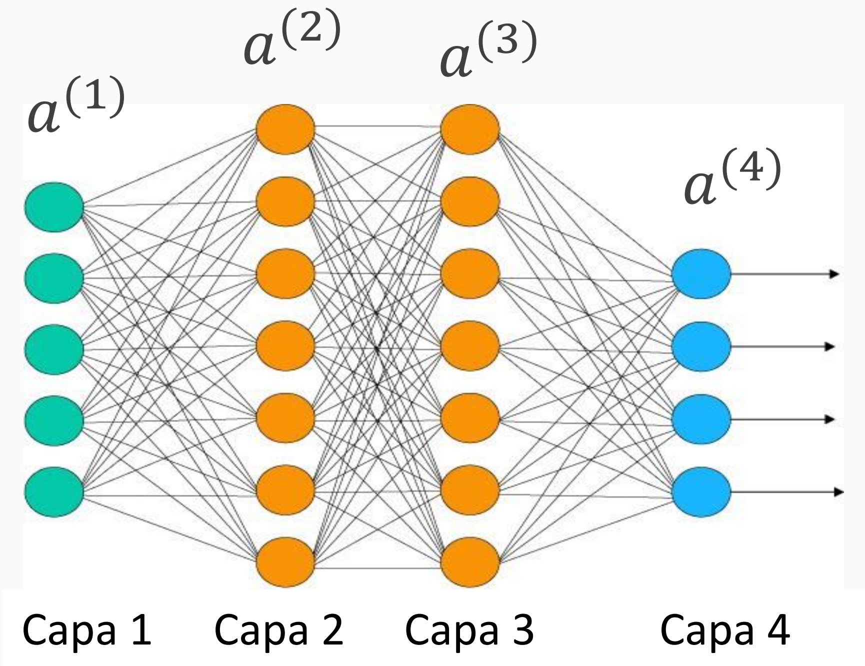
$a^{(2)} = g(z^{(2)})$ (agregamos $a_0^{(2)}$)

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

$a^{(3)} = g(z^{(3)})$ (agregamos $a_0^{(3)}$)

$z^{(4)} = \Theta^{(3)} a^{(3)}$

$a^{(4)} = h_\Theta(\mathbf{x}) = g(z^{(4)})$



BACKPROPAGATION (BKP)

Intuición: $\delta_j^{(l)}$ = *error* del nodo j en la capa l; i.e. *cómo quiero modificar* $a_j^{(l)}$

Para cada nodo de salida (capa L=4): $\delta_j^{(4)} = a_j^{(4)} - y_j$

Para la capa de salida:

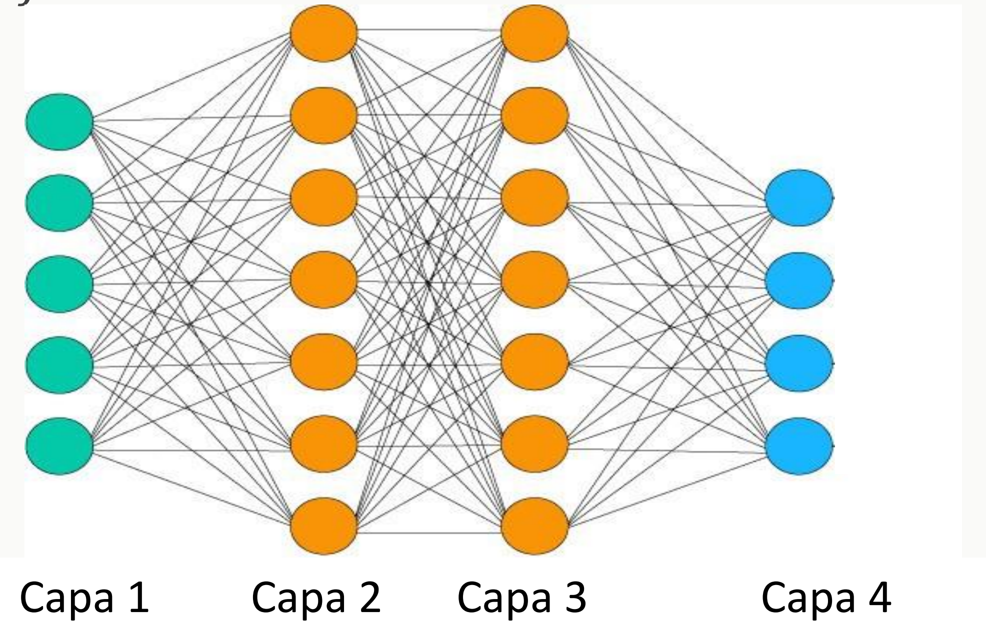
$$\delta^{(4)} = a^{(4)} - y$$

Para las capas ocultas:

$$\delta^{(3)} = \left(\Theta^{(3)}\right)^T \delta^{(4)} .* \, g'\left(z^{(3)}\right)$$

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* \, \underbrace{g'\left(z^{(2)}\right)}$$

$$g'\left(z^{(j)}\right) = a^{(j)} .* \left(1 - a^{(j)}\right)$$



ALGORITMO BACKPROPAGATION

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

Inicializa $\Delta_{ij}^{(l)} = 0$ (utilizada para calcular $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)

Para $i = 1$ a m:

$a^{(1)} = x^{(i)}$

Realiza FWP para calcular $a^{(L)}$ para $l = 2,3, \dots, L$

Utilizando $y^{(i)}$, calcula $\delta^{(L)} = a^{(L)} - y^{(i)}$

Calcula $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

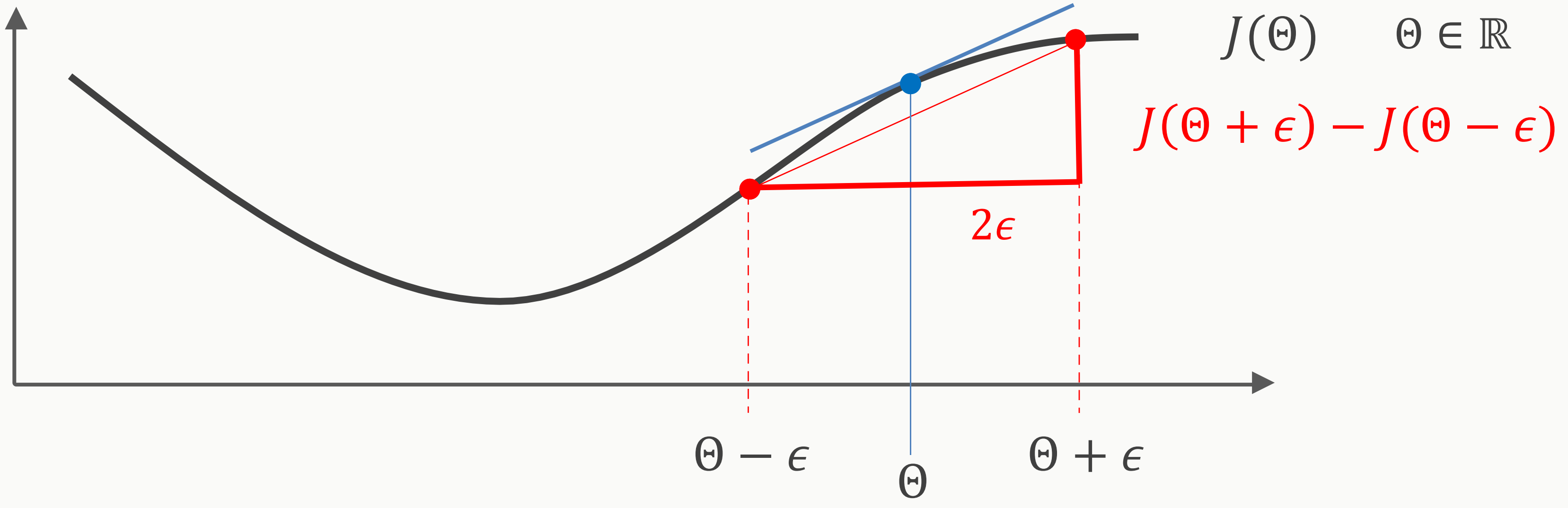
$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \text{ si } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \qquad \text{ si } j = 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

ESTIMACIÓN NUMÉRICA DE GRADIENTES



$$\frac{\mathrm{d}}{\mathrm{d}\Theta} J(\Theta) \approx \frac{J(\Theta+\epsilon) - J(\Theta-\epsilon)}{2\epsilon}$$

$$\epsilon = 10^{-4}$$

ESTIMACIÓN NUMÉRICA DE GRADIENTES

$\Theta \in \mathbb{R}^n$

$\Theta = [\theta_1, \theta_2, \theta_3 \ldots, \theta_n]$

$$\frac{\partial}{\partial \theta_1} J(\theta) \approx \frac{J(\theta_1 + \epsilon, \theta_2, \theta_3, \ldots, \theta_n) - J(\theta_1 - \epsilon, \theta_2, \theta_3, \ldots, \theta_n)}{2\epsilon}$$

$$\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \theta_2 + \epsilon, \theta_3, \ldots, \theta_n) - J(\theta_1, \theta_2 - \epsilon, \theta_3, \ldots, \theta_n)}{2\epsilon}$$

$$\vdots$$

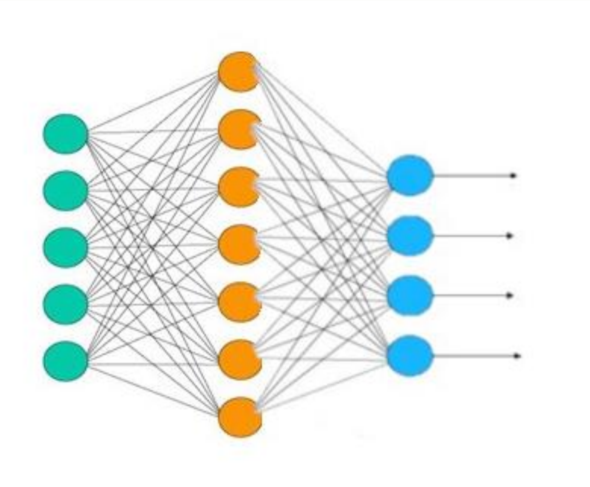
$$\frac{\partial}{\partial \theta_n} J(\theta) \approx \frac{J(\theta_1, \theta_2, \theta_3, \ldots, \theta_n + \epsilon) - J(\theta_1, \theta_2, \theta_3, \ldots, \theta_n - \epsilon)}{2\epsilon}$$

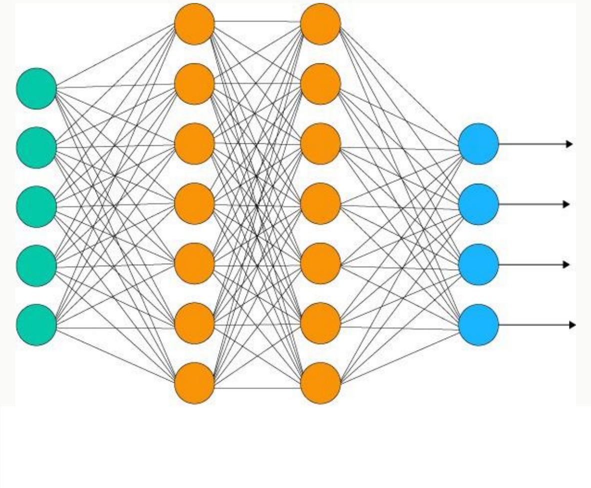
RECOMENDACIONES DE IMPLEMENTACIÓN

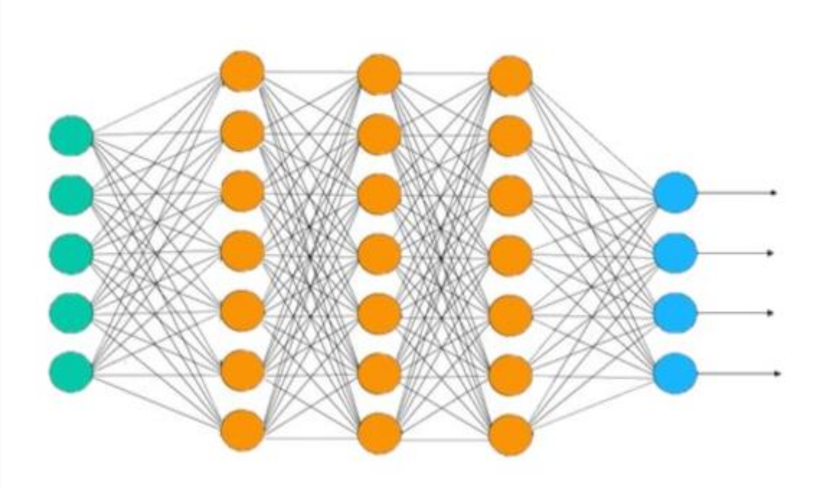
- Implementa BackProp para calcular D_{vec} (versión desenrollada de D)
- Implementa la verificación numérica del gradiente.
- Asegúrate de que den valores similares.
- Apaga la verificación numérica del gradiente para el aprendizaje.

RESUMEN

ARQUITECTURAS







- Elige una arquitectura dependiendo de la complejidad de la tarea
- N° de neuronas de entrada: Dimensión de las *features* $x^{(i)}$
- N° de neuronas de salida: Número de clases
- Por defecto: 1 capa oculta, si más de una, deben tener el mismo número de neuronas ocultas en cada capa (normalmente, entre más mejor)

ENTRENAMIENTO

- Inicializa los pesos de manera aleatoria
- Implementa FWP para obtener $h_{\Theta}\left(x^{(i)}\right)$ para toda $x^{(i)}$
- Implementa el código para calcular la función de costo $J(\Theta)$
- Implementa BKP para calcular $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$
- Para i=1 hasta m:
  - *Ejecuta FWP y BKP utilizando ejemplos* $(x^{(i)}, y^{(i)})$
  - *Calcula las activaciones* $a^{(l)}$ *y los términos* $\delta^{(l)}$ *para* $(l = 2, \ldots, L)$
- Calcula $D_{ij}^{(l)} = \frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$ con y sin regularización

ENTRENAMIENTO

- Verifica el gradiente comparando $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$ calculado mediante BKP contra la aproximación numérica usando $J(\Theta)$.
  - *Luego desactiva el código de verificación del gradiente.*
- Usa Descenso de Gradiente o algún otro método avanzado de optimización para tratar de minimizar $J(\Theta)$ en función de los parámetros Θ.

EJEMPLO DEL DESEMPEÑO DE ALGUNOS MÉTODOS DE OPTIMIZACIÓN