INTRODUCCIÓN A LAS REDES NEURALES

Aplicaciones en Python

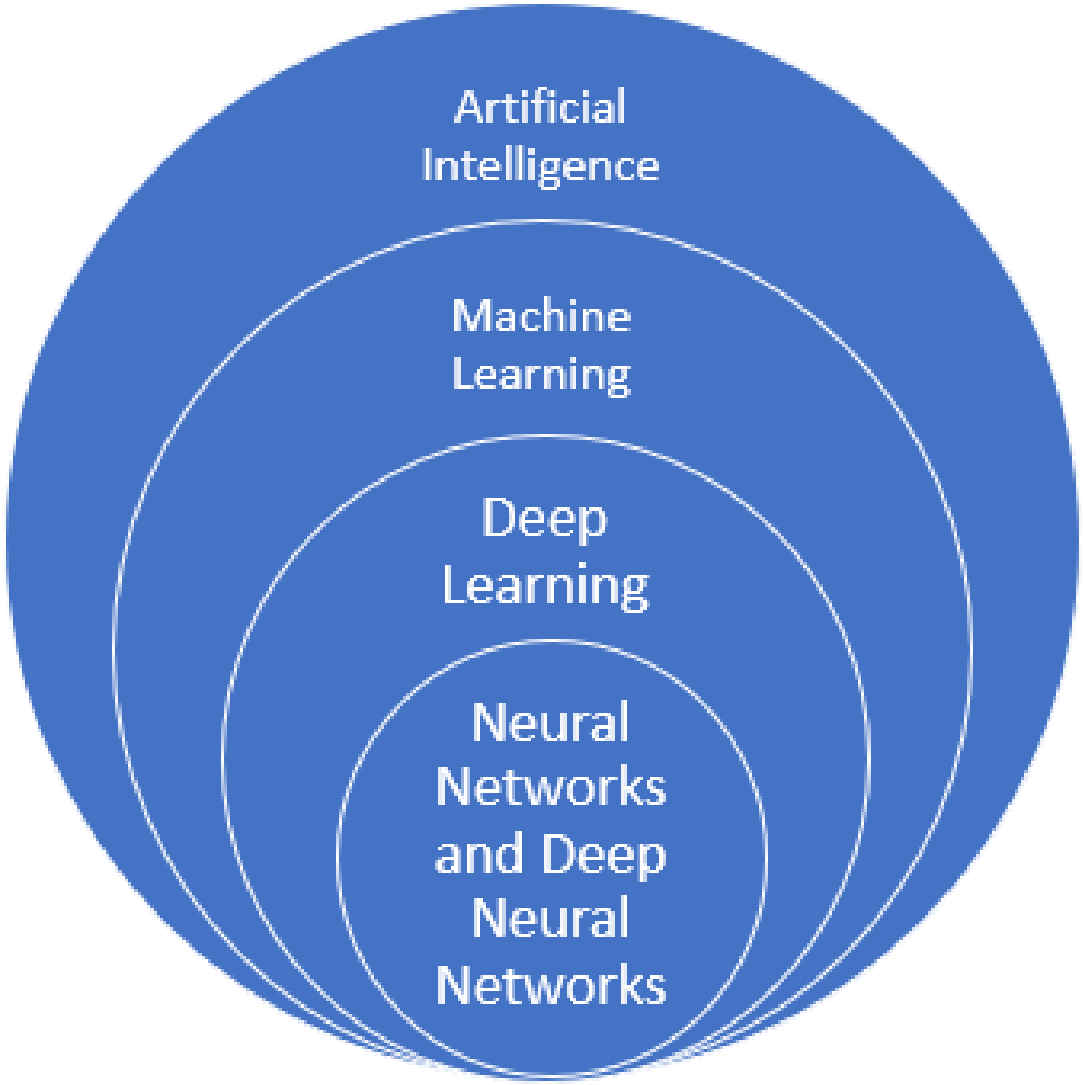
2 de septiembre de 2021

Siria Sadeddin

Universidad Simón Bolívar
Consultora de Machine Learning en Wolfram Research

Introducción

Inteligencia artificial y Redes Neurales



El cerebro, inspiración para la inteligencia artificial

Red Neural Artificial vs Red Neural Natural

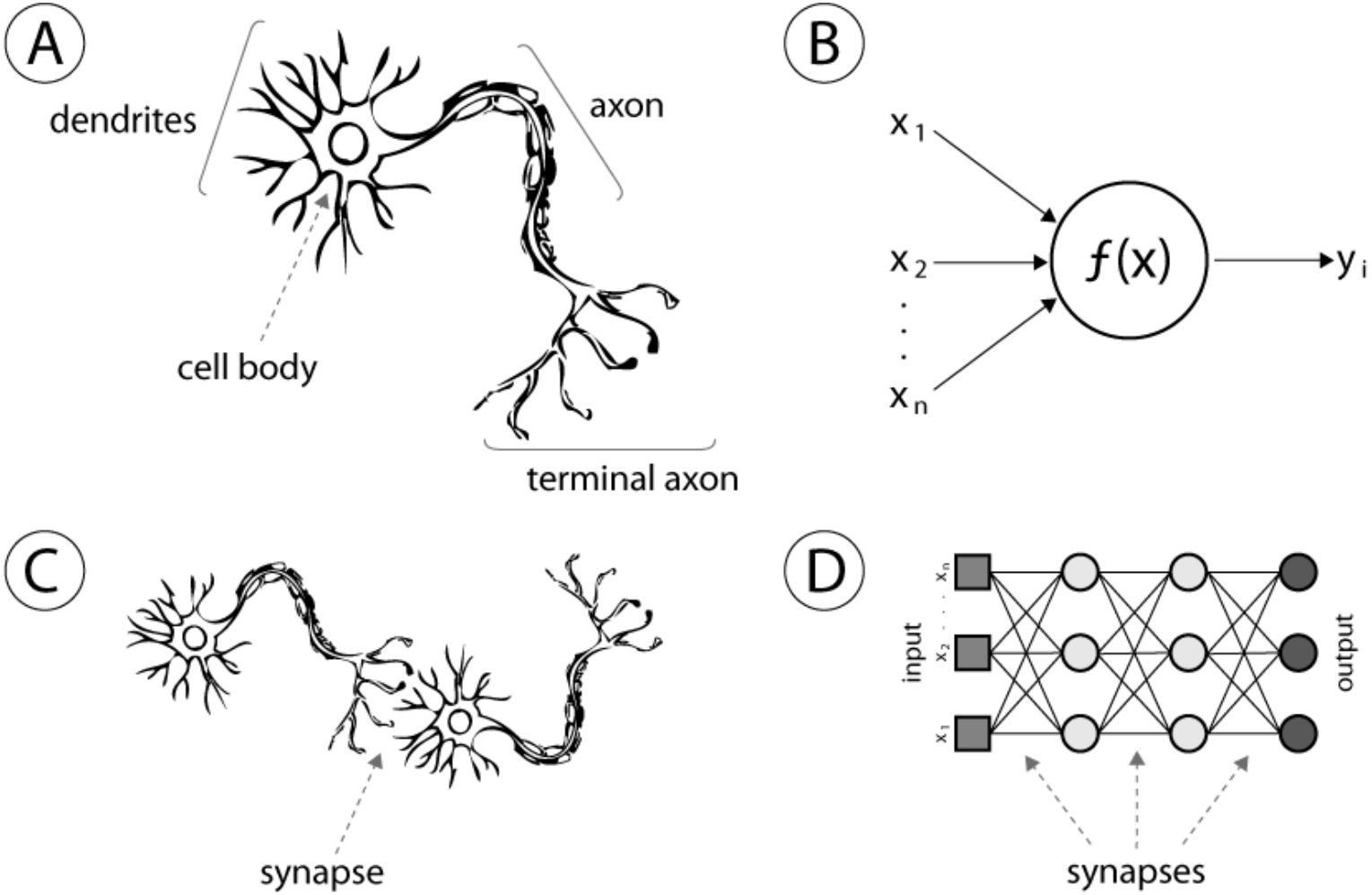


Figura 1: Redes Neurales Artificiales inspiradas en el cerebro

Red Neural y Perceptrón

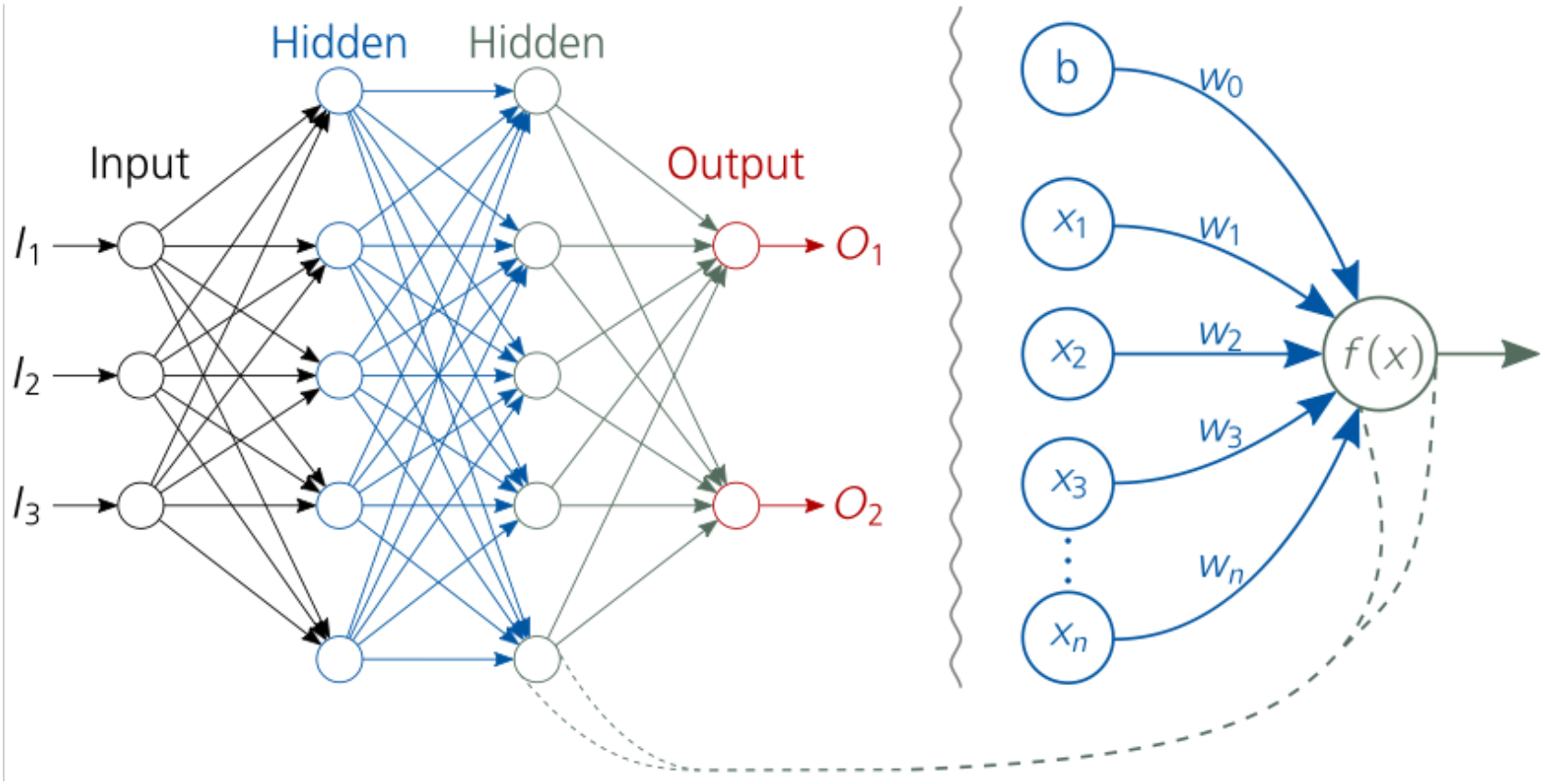


Figura 2: El perceptrón como elemento fundamental de la Red Neural

El Perceptrón

Matemáticas del Perceptrón

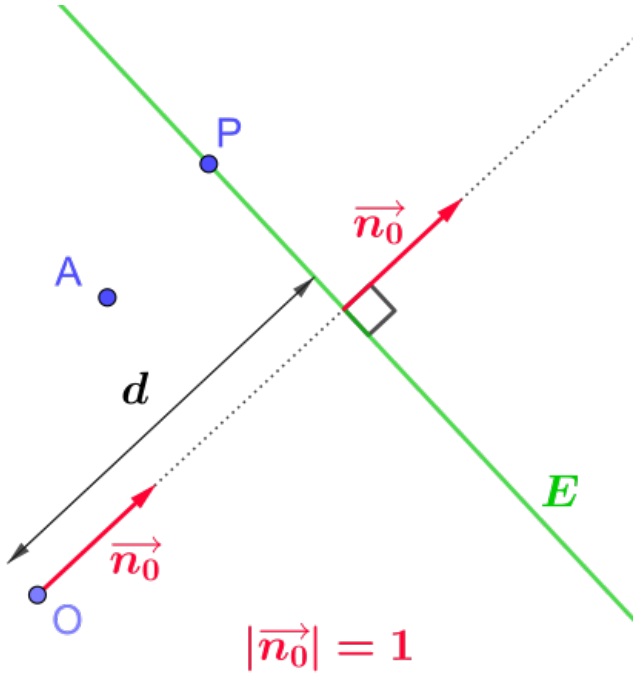


Figura 3: Forma Normal de Hesse

$$\overrightarrow{r} . \overrightarrow{n}_0 - d = 0$$

Matemáticas del Perceptrón

- g(x) es una función lineal

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

- Hiperplano en el espacio de características

- vector normal al hiperplano

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

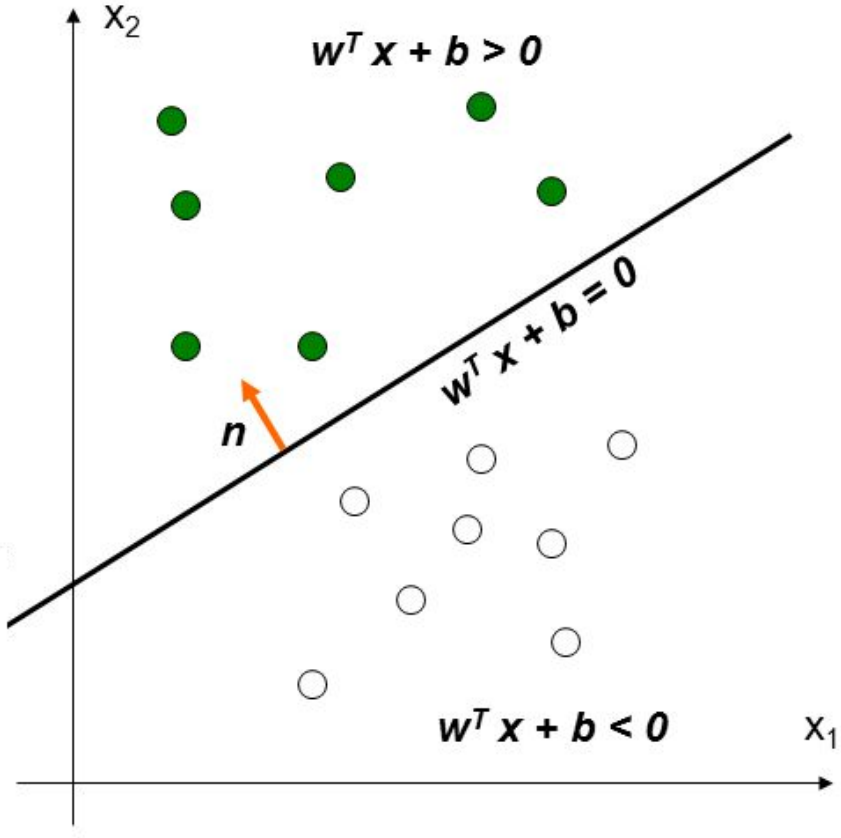
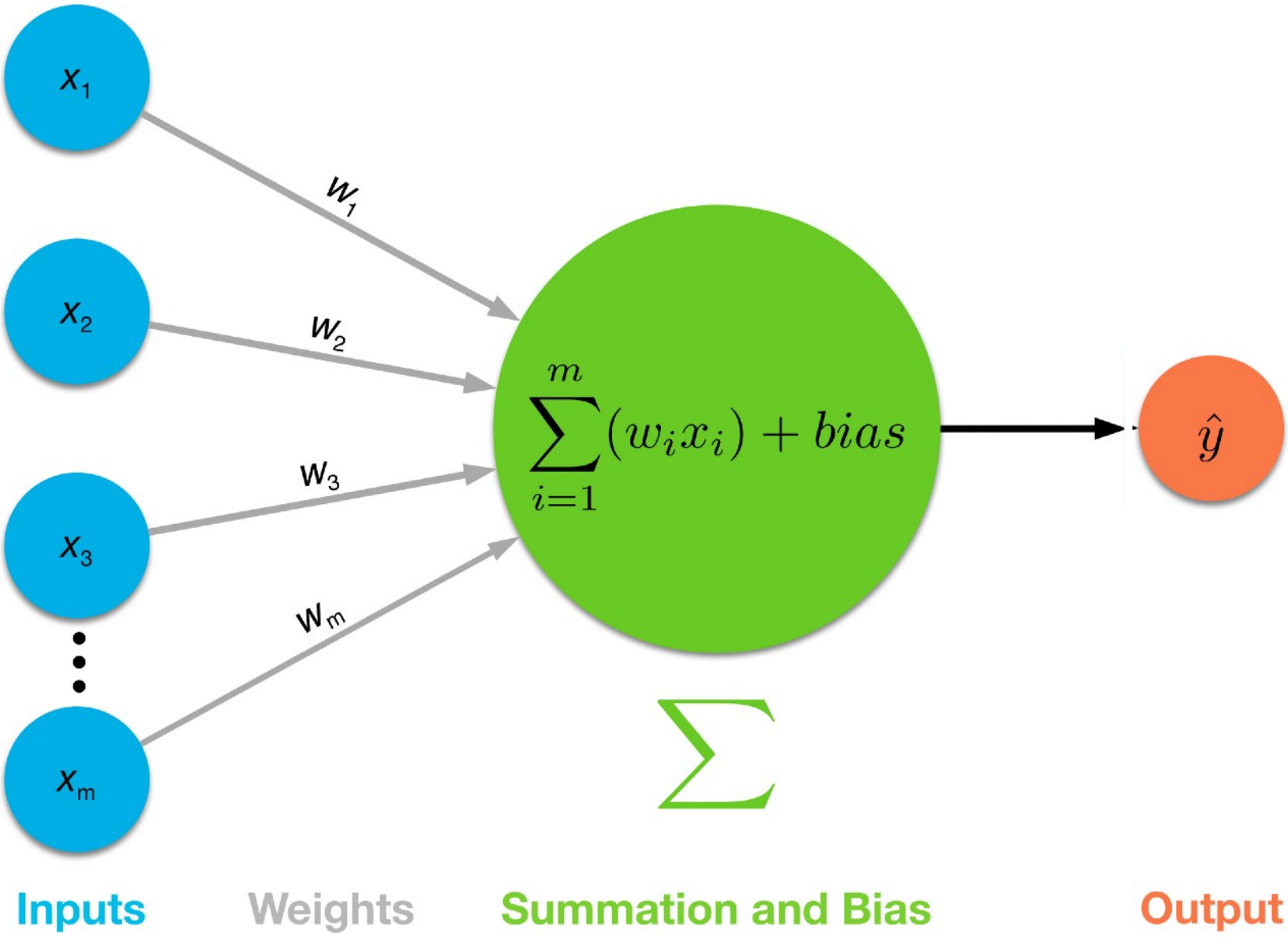


Figura 4: Clasificador Lineal

Esquema del perceptrón



Pseudocódigo del perceptrón

Para un dataset con n ejemplos $\mathcal{D}_n \in \mathbb{R}^d$ y vector de clases $y \in \{-1, 1\}$, el algoritmo de Perceptrón entrena un clasificador binario $h(x; \theta, \theta_0)$ usando el siguiente algoritmo en τ pasos:

$\textsc{Perceptron}(\tau, \mathcal{D}_n)$

1 $\quad \theta = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}^T$
2 $\quad \theta_0 = 0$
3 $\quad$ **for** $t = 1$ **to** τ
4 $\quad\quad$ **for** $i = 1$ **to** n
5 $\quad\quad\quad$ **if** $y^{(i)} \left(\theta^T x^{(i)} + \theta_0 \right) \leqslant 0$
6 $\quad\quad\quad\quad \theta = \theta + y^{(i)} x^{(i)}$
7 $\quad\quad\quad\quad \theta_0 = \theta_0 + y^{(i)}$
8 $\quad$ **return** θ, θ_0

Ajuste del separador

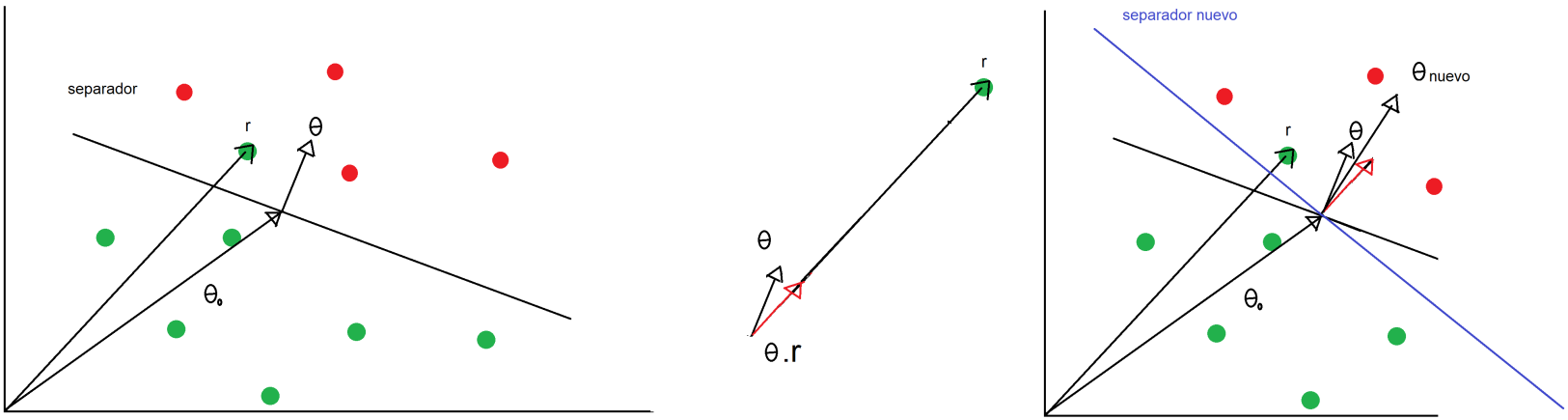


Figura 5: Ajuste del separador

Propiedades del Algoritmo de Perceptrón

Separabilidad Lineal

Un conjunto de entrenamiento $\mathcal{D}_n$ es linalmente separable si existen θ y θ_0, tal que, para todo $i = 1, 2, \ldots, n$

$$y_i(\theta^T x_i + \theta_0) > 0$$

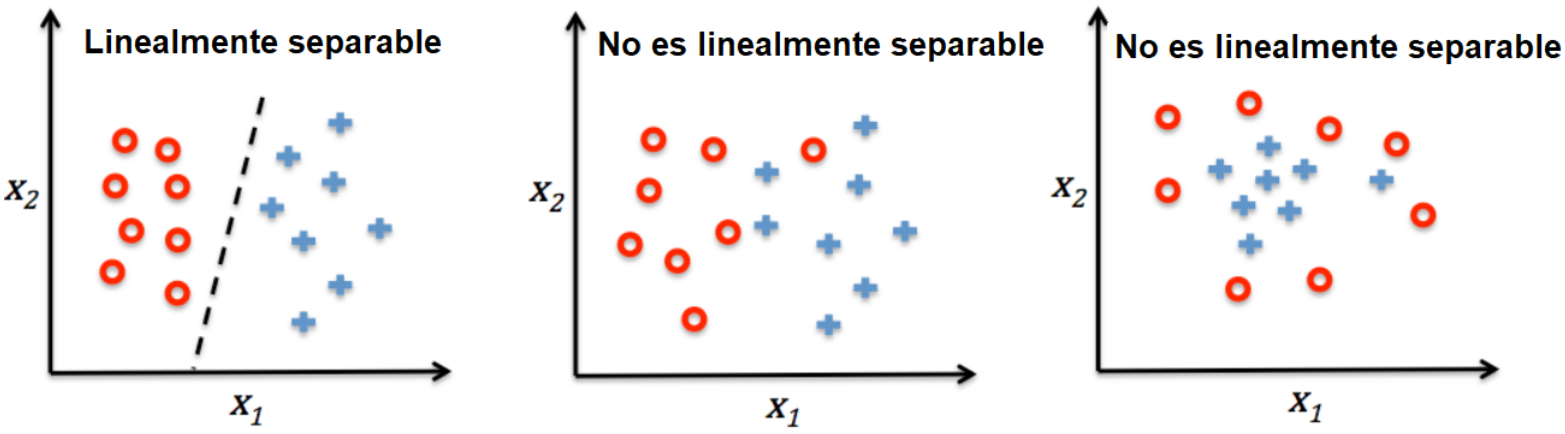


Figura 6: Separabilidad lineal

Teorema de la Convergencia

Teorema de la convergencia del Algoritmo del Perceptrón

Sea $\mathcal{D}_n$ un set de datos que es linealmente separable. Entonces está garantizado que el Algoritmo de Perceptrón encontrará un separador lineal.

1

[1] MIT Introduction to Machine Learning, Chapter 3: The Perceptron

Debilidades del algoritmo

- Aunque pueden existir dos separadores con igual número de errores (L2 y L3) uno de ellos esta más cerca del óptimo (L1). Por lo tanto no es posible conocer que separador elegir con la información del error.
- Todas las predicciones son categóricas, por lo que no es posible saber el grado de confianza de la predicción dada por el separador

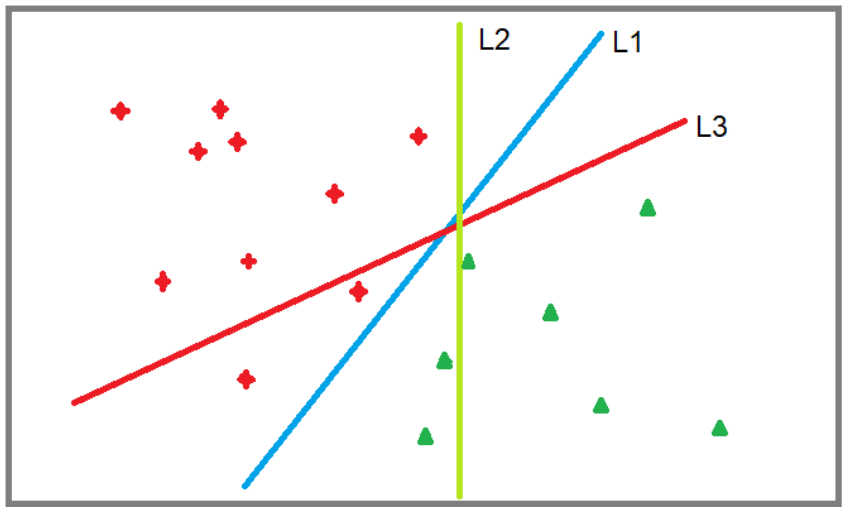


Figura 7: Separador óptimo

Problema de optimización

Función Objetivo

Estamos buscando un conjunto de parámetros $\Theta \in \{\theta_0, \theta_1, \ldots \theta_n\}$ que minimicen la función objetivo (Loss) $J(\Theta)$:

$$\Theta^* = \arg\min_{\Theta} J(\Theta)$$

La forma mas común para J es:

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h(x_i;\Theta), y_i) + \lambda R(\Theta)$$

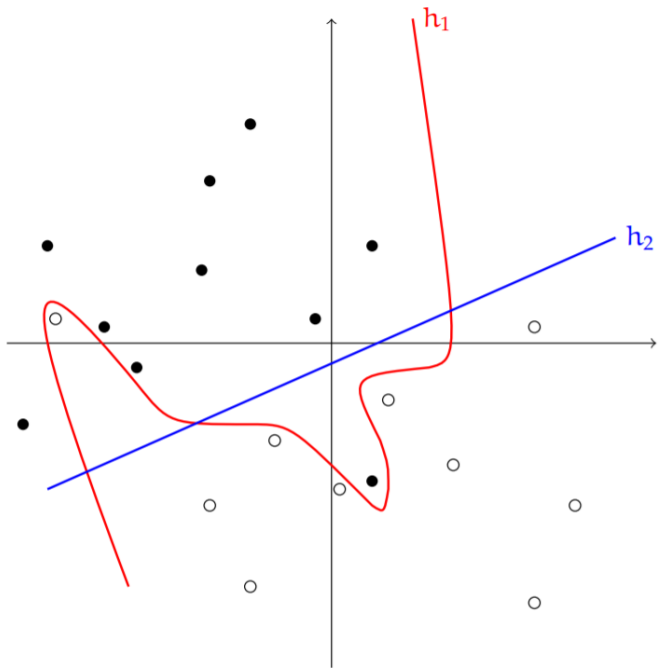


Figura 8: Overfitting

Overfitting

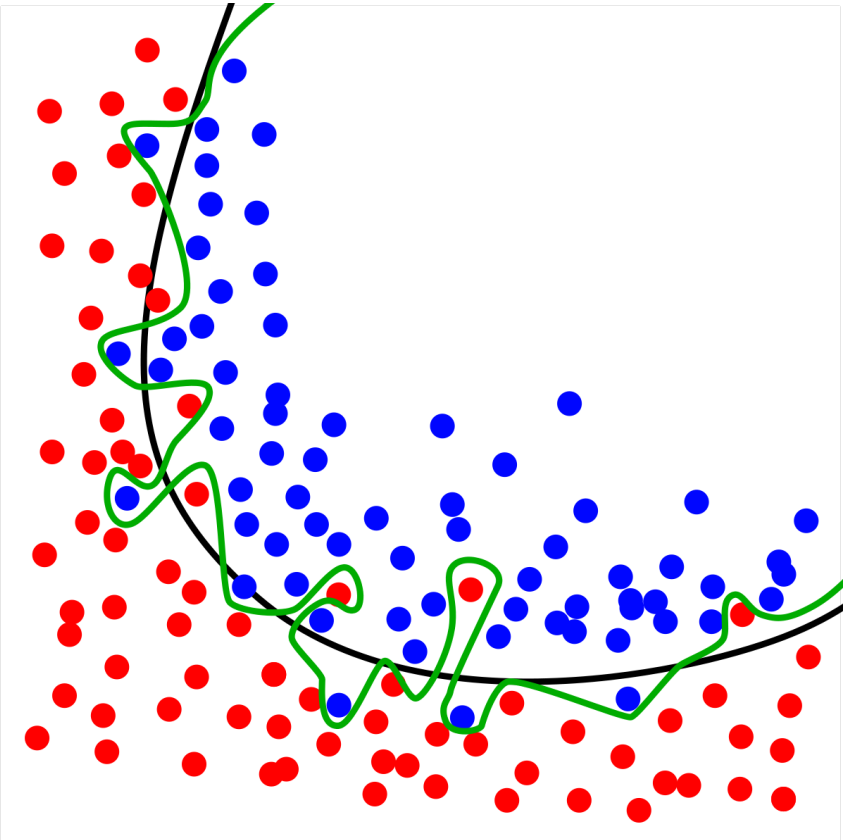


Figura 9: Overfitting

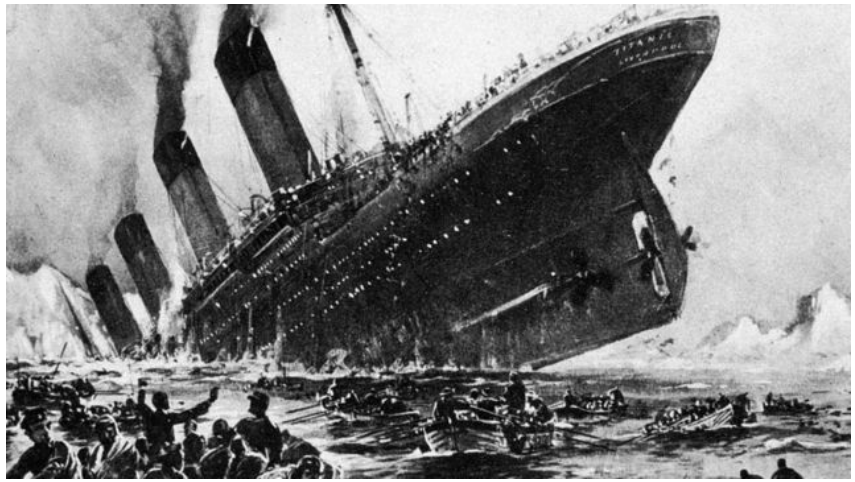


Figura 10: Sobrevivientes del Titanic

Clasificador Lineal Logístico

Para el algoritmo del perceptrón:

$$h(\theta, \theta_0) = sign(\theta^T x_i + \theta_0)$$

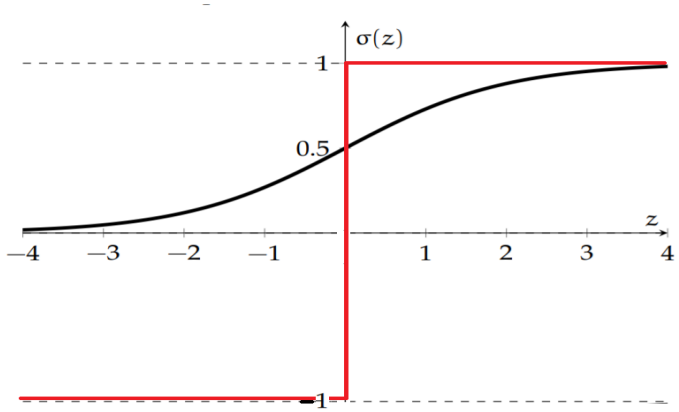


Figura 11: Clasificador Lineal binario vs logístico

Clasificador Lineal Logístico

Para el Clasificador Lineal Logístico:

$$h(\theta, \theta_0) = \sigma(\theta^T x_i + \theta_0)$$

Donde

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

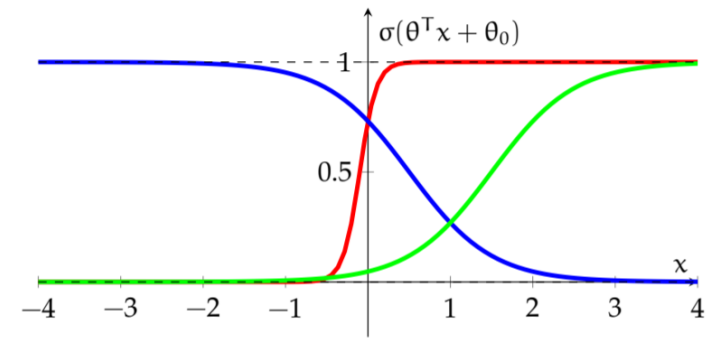


Figura 12: **Función sigmodide para distintos $\theta^T x + \theta_0$**

Sigmoide en 2 dimensiones

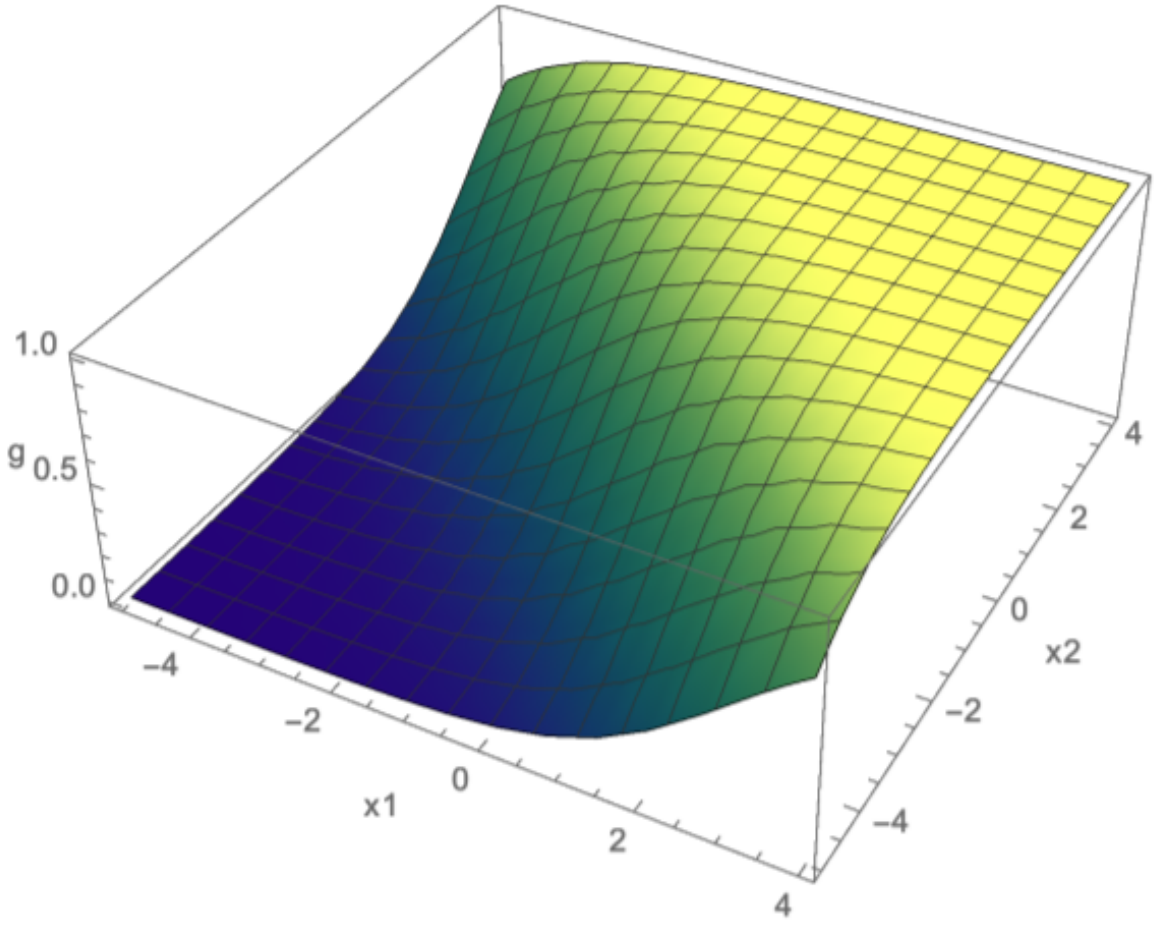


Figura 13: Sigmoide como superficie de probabilidad

Función objetivo del Clasificador Lineal Logístico

Sea g_i la probabilidad de pertenecer a la clase 1 es $g_i = \sigma(\theta^T x_i + \theta_0)$

$$\prod_{i=1}^{n} \begin{cases} g_i & \text{si } y_i = 1 \\ 1 - g_i & \text{si } y_i = 0 \end{cases}$$

suponiendo que las predicciones son independientes, se puede escribir:

$$\prod_{i=1}^{n} g_i^{y_i} (1 - g_i)^{(1-y_i)}$$

Se quiere maximizar esta función, y tomando en cuenta que la función $\log(x)$ es monótona y continua:

$$\frac{1}{n} \sum_{i=1}^{n} y_i \log(g_i) + (1 - y_i) \log(1 - g_i)$$

Método del Descenso del Gradiente (una dimensión)

Función de costo (Loss):

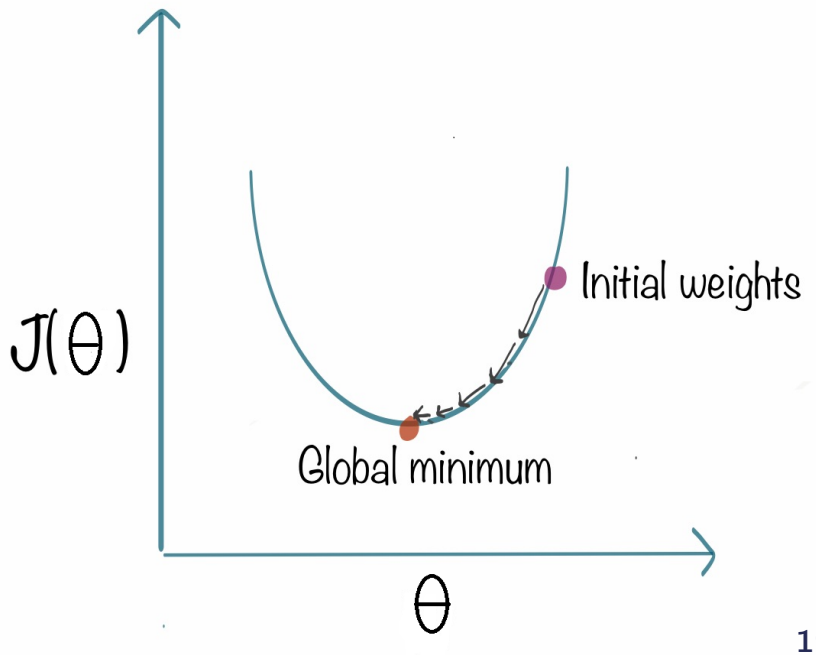
$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(g_i) + (1 - y_i)\log(1 - g_i)$$

donde

$$g_i = \sigma(\theta^T x_i)$$

La derivada de J es:

$$\frac{\partial}{\partial\theta} J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(g_i - y_i)x_i$$



Pseudocódigo

1D-GRADIENT-DESCENT$(\Theta_{init}, \eta, f, f', \epsilon)$

1 $\Theta^{(0)} = \Theta_{init}$
2 $t = 0$
3 **repeat**
4 $\quad t = t + 1$
5 $\quad \Theta^{(t)} = \Theta^{(t-1)} - \eta\, f'(\Theta^{(t-1)})$
6 **until** $|f(\Theta^{(t)}) - f(\Theta^{(t-1)})| < \epsilon$
7 **return** $\Theta^{(t)}$

Figura 15: Pseudocodigo Descenso del Gradiente

Método del Descenso del Gradiente (múltiples variables)

El gradiente de la función de costo indica la dirección del ajuste de los parámetros:

$$\nabla_{\Theta} J(\Theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \\ \frac{\partial J}{\partial \theta_2} \\ \\ \vdots \\ \\ \frac{\partial J}{\partial \theta_m} \end{bmatrix}$$

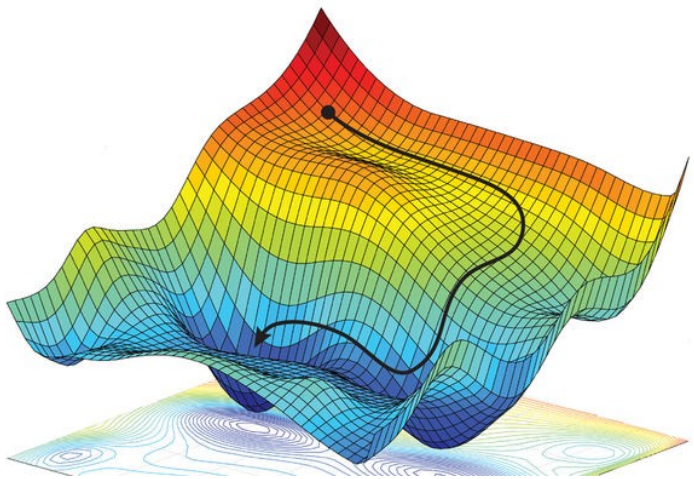


Figura 16: Función de costo para dos variables

$$\Theta(t) = \Theta(t-1) - \eta \nabla J(\Theta(t-1))$$

Aplicación en 2D

$$g_i = \sigma(\theta^T x_i + \theta_0)$$

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(g_i) + (1 - y_i)\log(1 - g_i)$$

$$\nabla J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n}(g_i - y_i)x_i + \lambda\theta$$

$$\frac{\partial}{\partial\theta_0} J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n}(g_i - y_i)$$

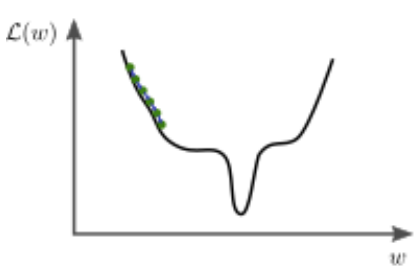
Descenso de Gradiente estocástico

STOCHASTIC-GRADIENT-DESCENT($\Theta_{init}, \eta, f, \nabla_\Theta f_1, \ldots, \nabla_\Theta f_n, T$)

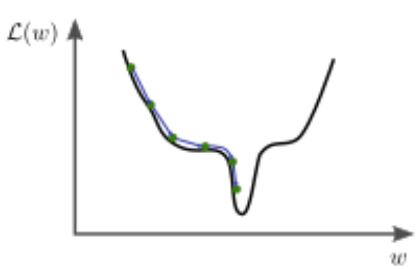
1 $\Theta^{(0)} = \Theta_{init}$
2 **for** $t = 1$ **to** T
3 randomly select $i \in \{1, 2, \ldots, n\}$
4 $\Theta^{(t)} = \Theta^{(t-1)} - \eta(t) \nabla_\Theta f_i(\Theta^{(t-1)})$
5 **return** $\Theta^{(t)}$

Figura 17: Pseudocódigo descenso estocástico del gradiente

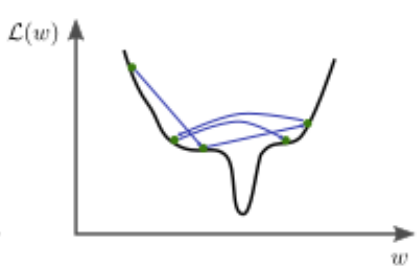
Dificultades del descenso de gradiente



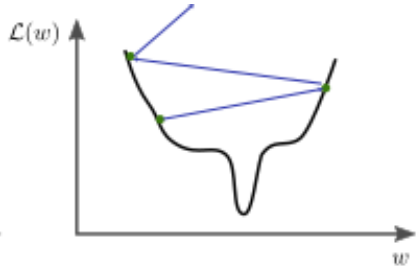
Paso muy pequeño



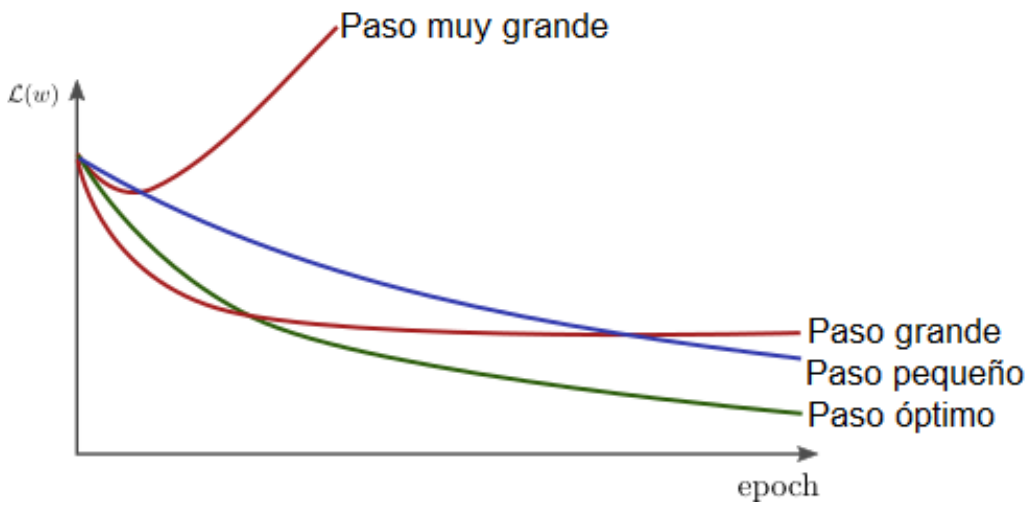
Paso óptimo

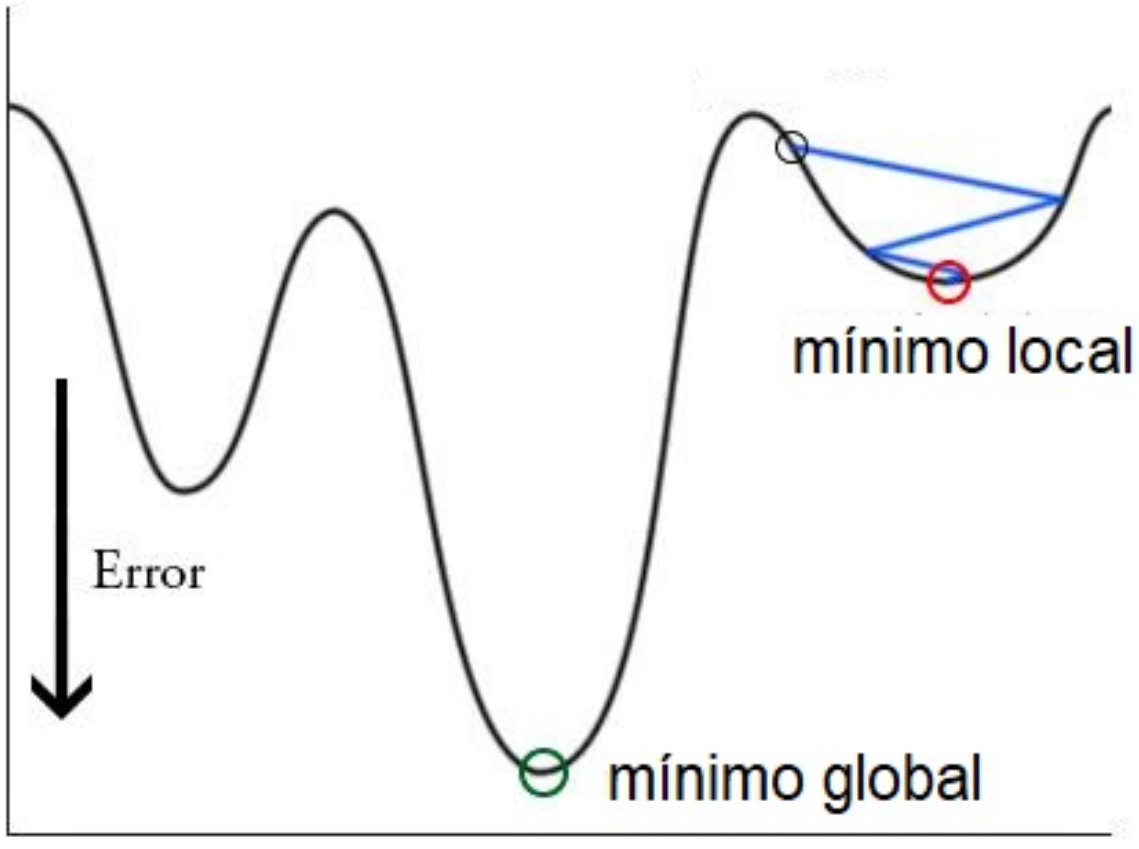


Paso grande



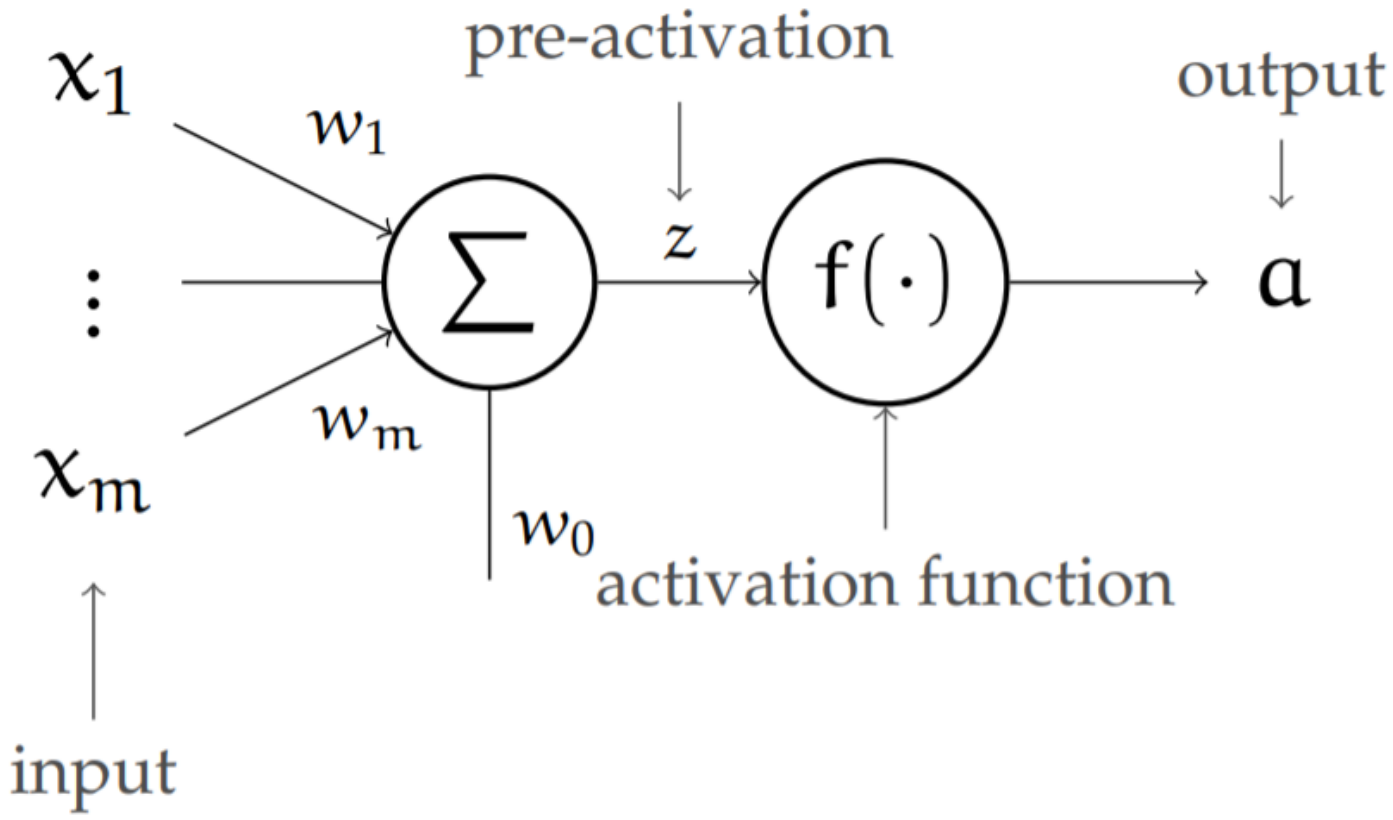
Paso muy grande




mínimo local
Error
mínimo global

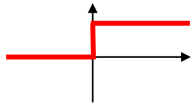
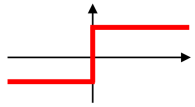
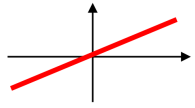
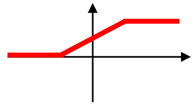
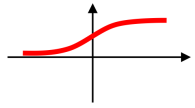
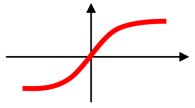
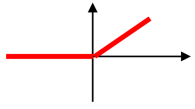
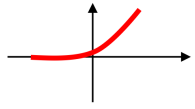
Redes Neurales

La neurona



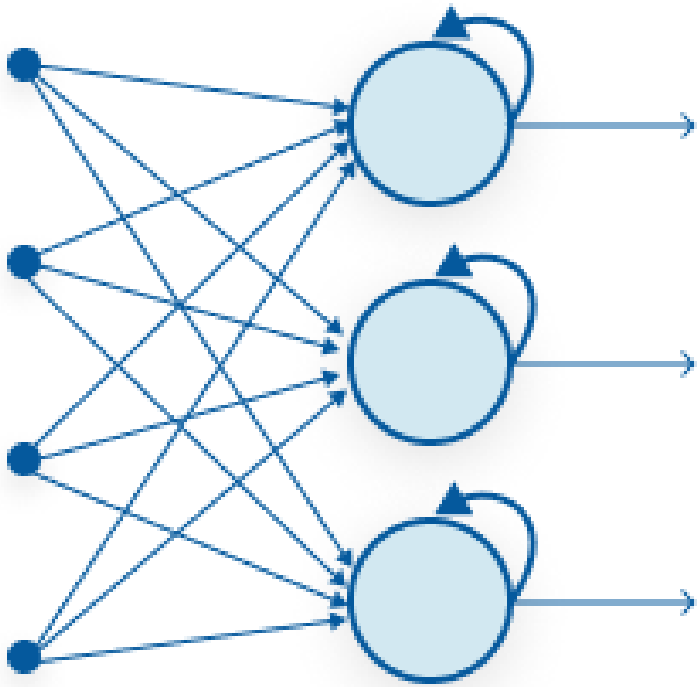
Donde f es una función de $\mathbb{R}^d \to \mathbb{R}$

Función de activación

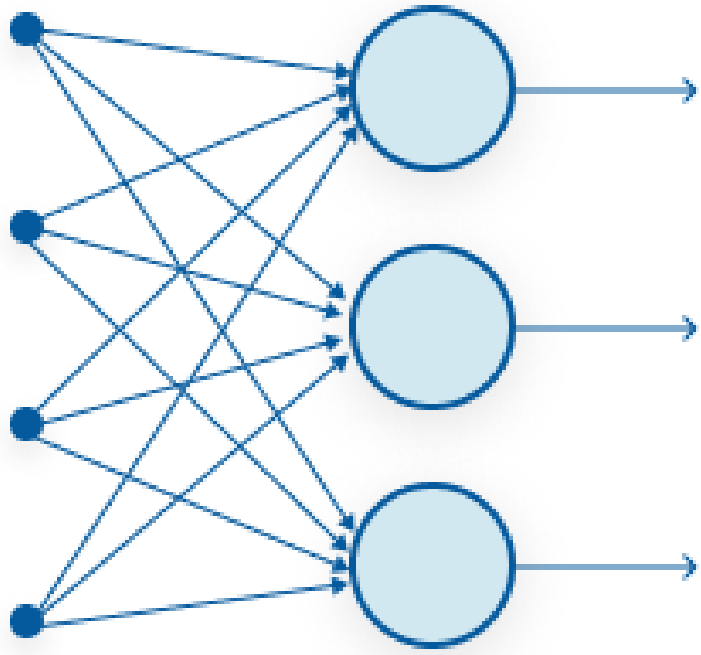
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016 (http://sebastianraschka.com)

Feed Forward

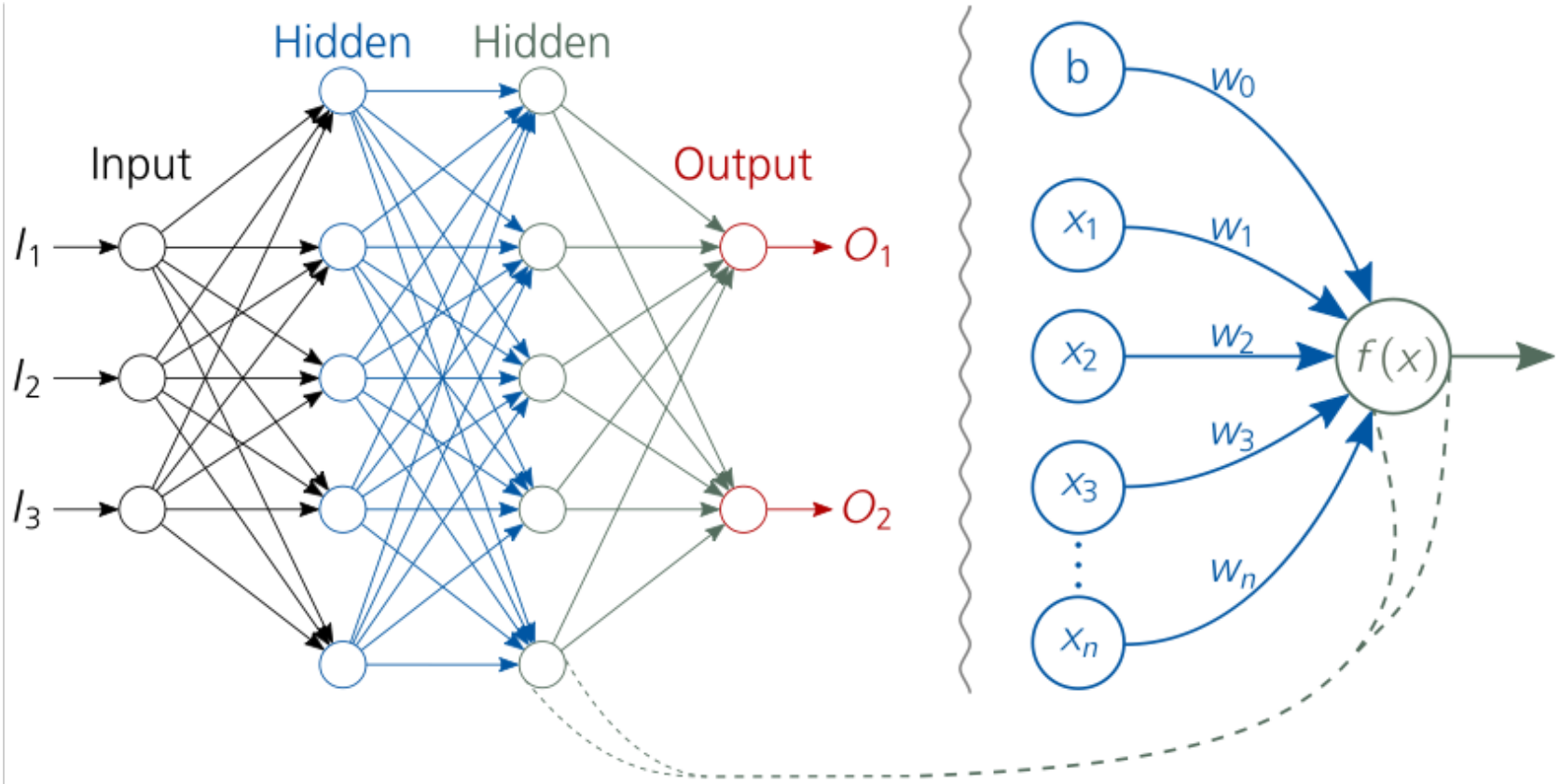


Recurrent Neural Network

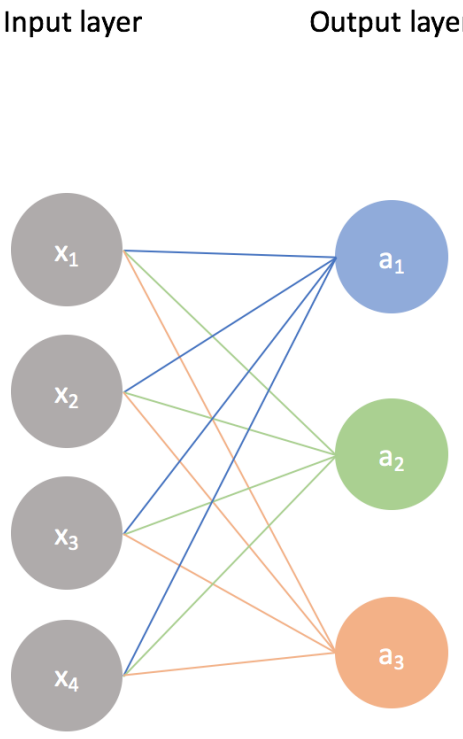


Feed-Forward Neural Network

NN Densamente Conectada



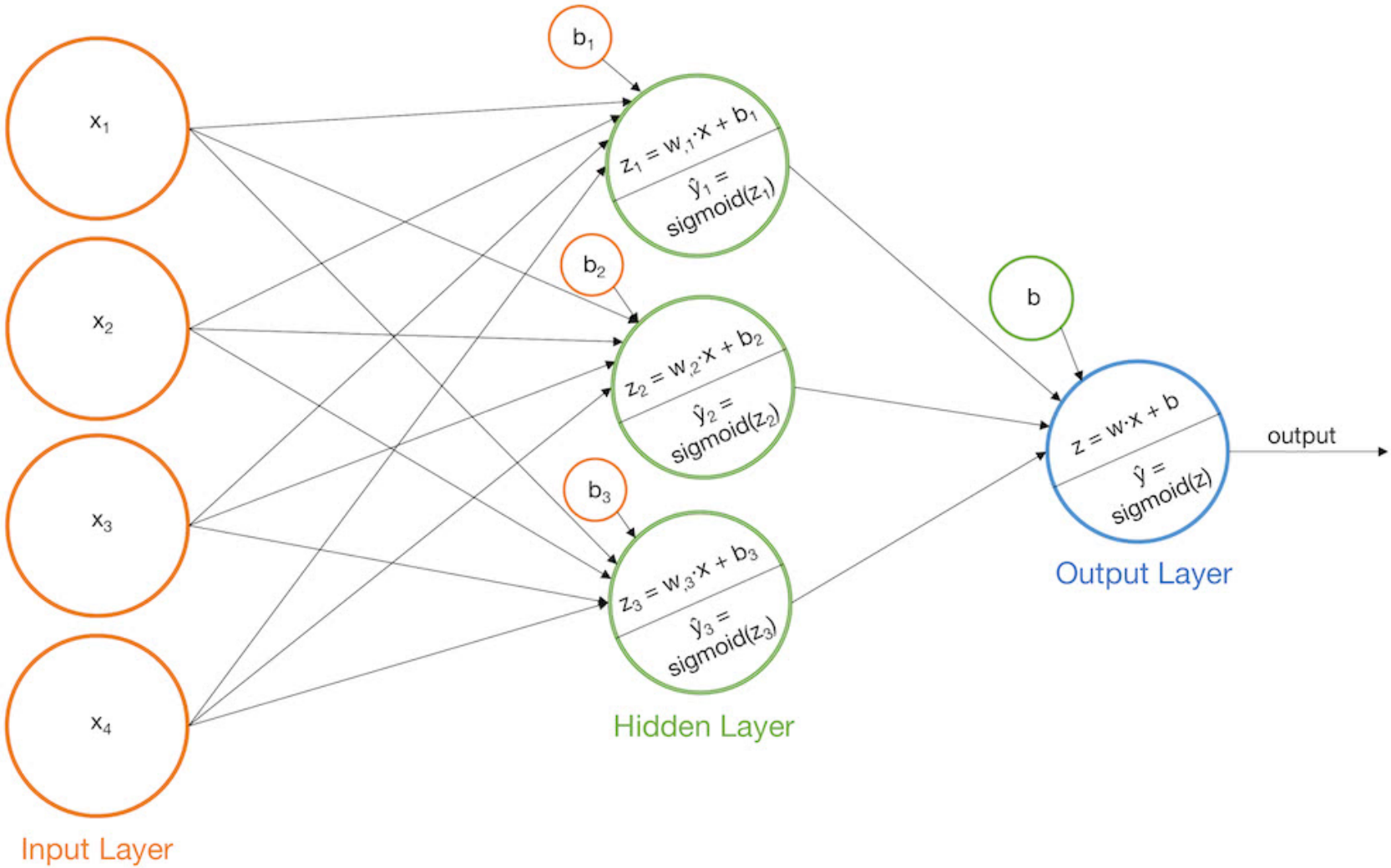
Algebra de la Red Neural



A simple neural network

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow{activation} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Red Neural Multicapa

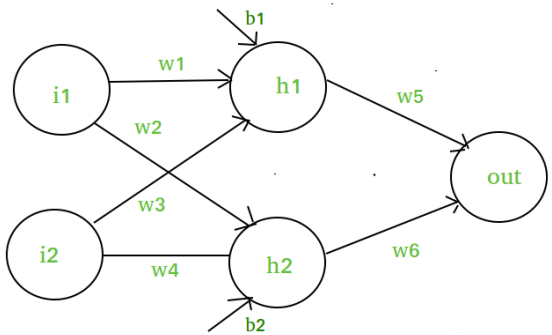


¿Activación no lineal?

En la Red Neural las activaciones de las neuronas internas **deben** ser no lineales. En este caso si las activaciones son lineales $f(x) = x$ tendríamos:

$$\begin{cases} h_1(i_1, i_2) = f(w_1 i_1 + w_3 i_2) = w_1 i_1 + w_3 i_2 \\ h_2(i_1, i_2) = f(w_2 i_1 + w_4 i_2) = w_2 i_1 + w_4 i_2 \end{cases}$$

$$\begin{aligned} out &= f(w_5 h_1 + w_6 h_2) = w_5 h_1 + w_6 h_2 \\ &= w_5(w_1 i_1 + w_3 i_2) + w_6(w_2 i_1 + w_4 i_2) \\ &= (w_5 w_1 + w_6 w_2) i_1 + (w_5 w_3 + w_6 w_4) i_2 \\ &= W_a i_1 + W_b i_2 \end{aligned}$$



Back Propagation

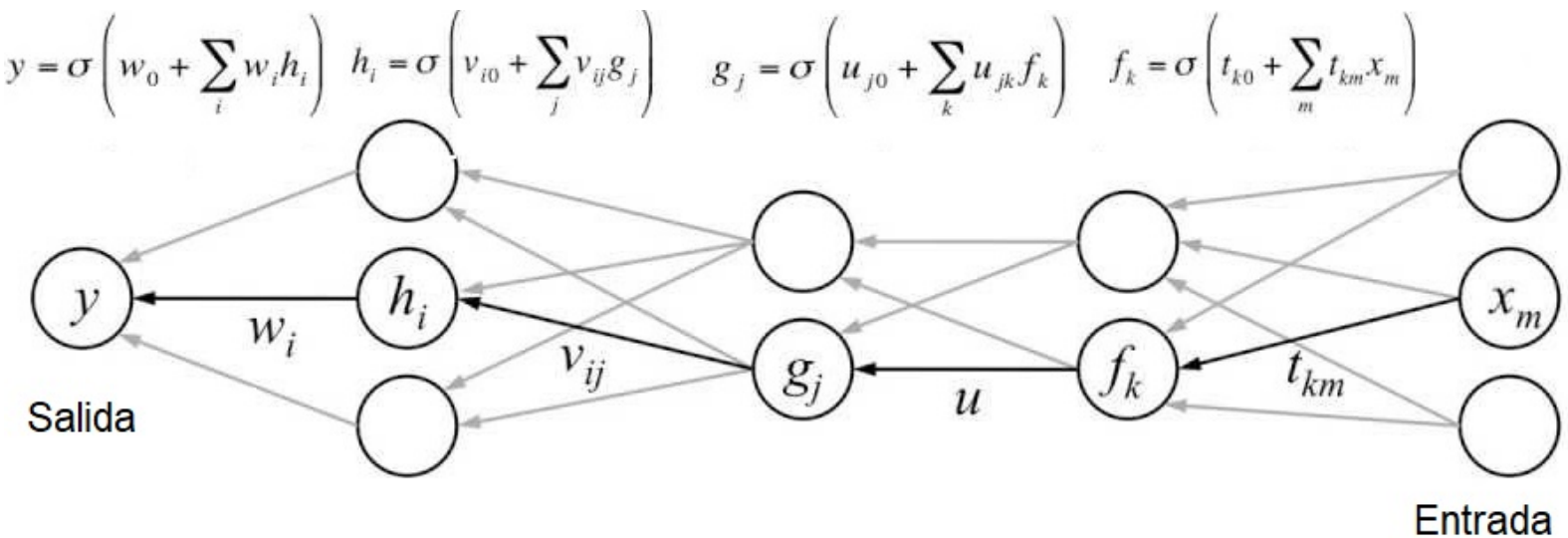
Regla de la cadena

Recordemos la regla de la cadena de las derivadas:

$$a = f(g(x))$$

$$\frac{\partial a}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

Esta regla se aplica a las redes neurales, ya que la salida es una función compuesta de todas las operaciones f_i en las neuronas:



Gradiente del Loss

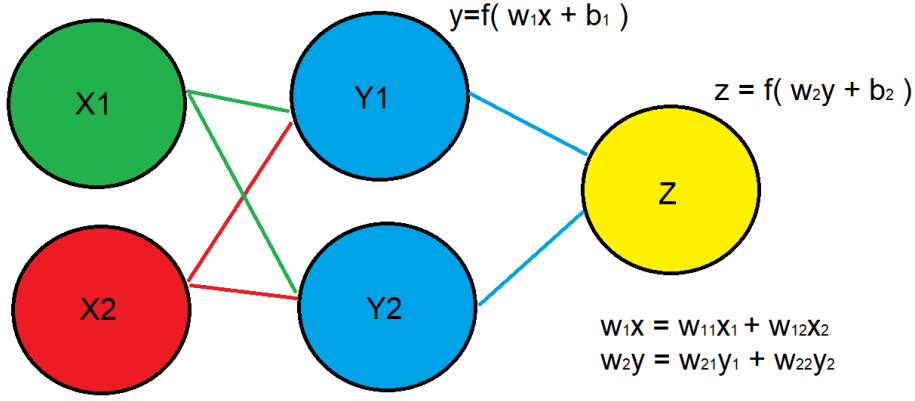
Supongamos $f(x) = \sigma(x)$ la función de activación en todas las neuronas y $f'(x) = (1 - \sigma)\sigma$. Recordemos, que para un punto de entrenamiento:

$$L = -(a \log(\sigma) + (1 - a) \log(1 - \sigma))$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \sigma} \frac{\partial \sigma}{\partial w_2}$$

$$\begin{cases} \frac{\partial L}{\partial \sigma} = \frac{\sigma - a}{\sigma(1-\sigma)} \\ \frac{\partial \sigma}{\partial w_2} = \sigma(1 - \sigma)y \end{cases}$$

$$\frac{\partial L}{\partial w_2} = (\sigma - a)y$$



Para el término independiente:

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial b_2}$$

$$\frac{\partial \sigma}{\partial b_2} = \sigma(1-\sigma)$$

$$\frac{\partial L}{\partial b_2} = (\sigma - a)$$

Para una neurona de la capa interna:

$$\frac{\partial L}{\partial w_{1,1}} = \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial y}\frac{\partial y}{\partial w_{1,1}}$$

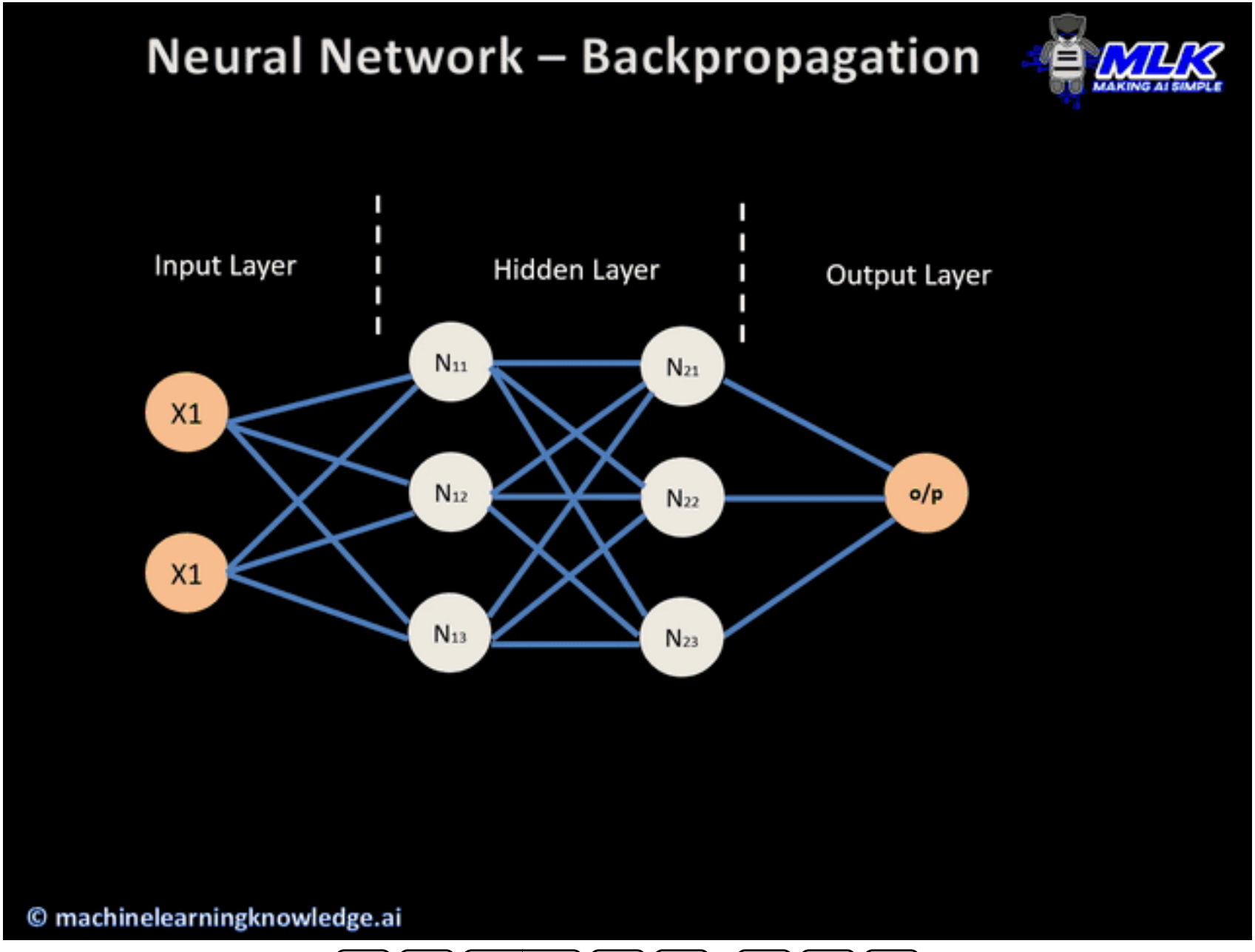
$$\begin{cases} \frac{\partial \sigma}{\partial y} = \sigma(1-\sigma)W_2 \\ \frac{\partial y}{\partial W_{1,1}} = \sigma(1-\sigma)x \\ \frac{\partial L}{\partial \sigma} = \frac{\sigma - a}{\sigma(1-\sigma)} \end{cases}$$

$$\frac{\partial L}{\partial w_{1,1}} = \sigma(1-\sigma)(\sigma - a)W_2 x$$

Para el termino independiente:

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial y}\frac{\partial y}{\partial b_1}$$

$$\frac{\partial L}{\partial b_1} = \sigma(1-\sigma)(\sigma - a)W_2$$


Neural Network – Backpropagation
MLK
MAKING AI SIMPLE
Input Layer
Hidden Layer
Output Layer
X1
X1
N11
N12
N13
N21
N22
N23
o/p
© machinelearningknowledge.ai