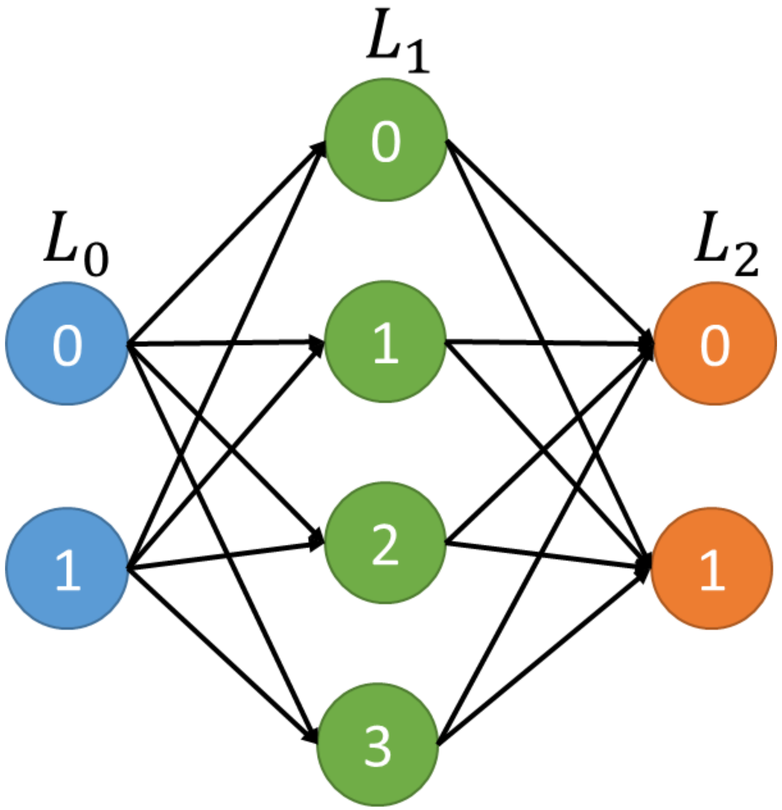
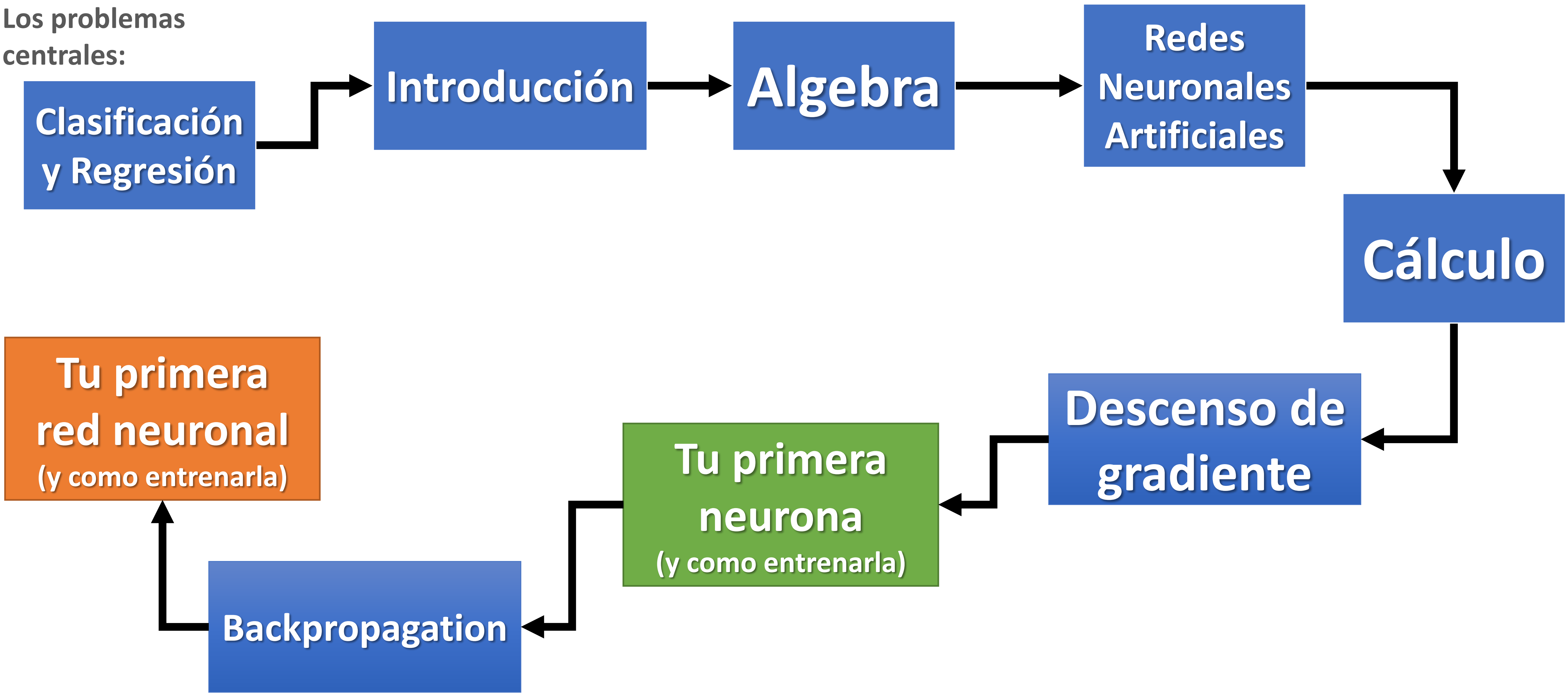


Redes Neuronales Desde 0

Teoría y Práctica

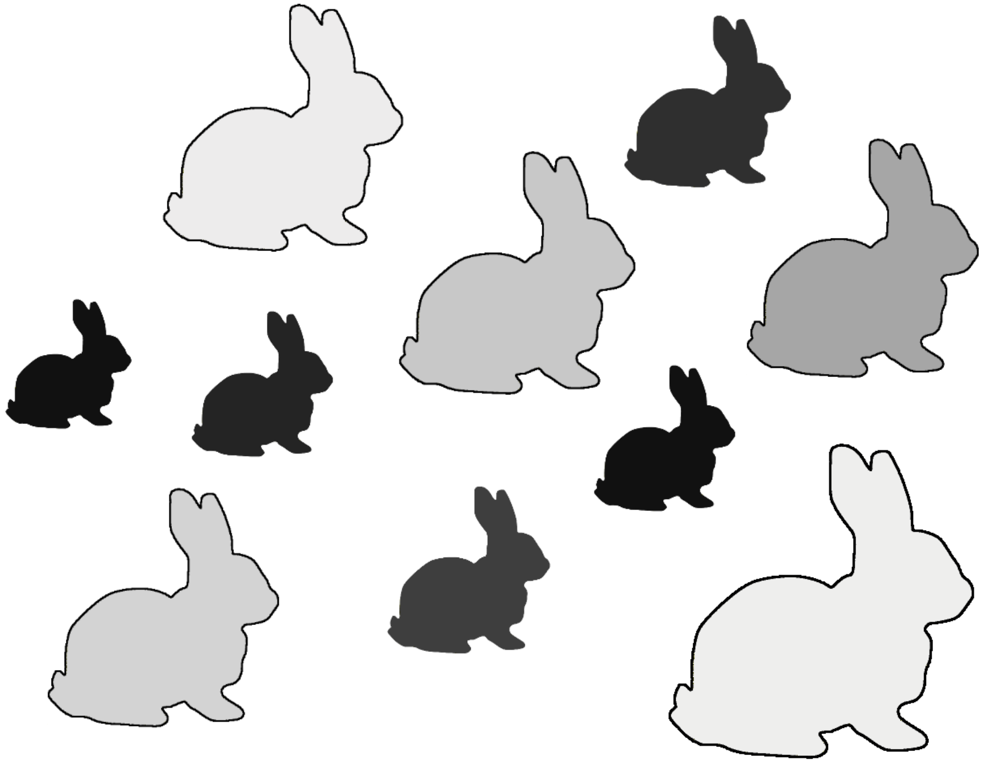
Rodolfo E. Escobar Uribe

The Roadmap


Los problemas centrales:
Clasificación y Regresión
Introducción
Algebra
Redes Neuronales Artificiales
Cálculo
Descenso de gradiente
Tu primera neurona (y como entrenarla)
Backpropagation
Tu primera red neuronal (y como entrenarla)

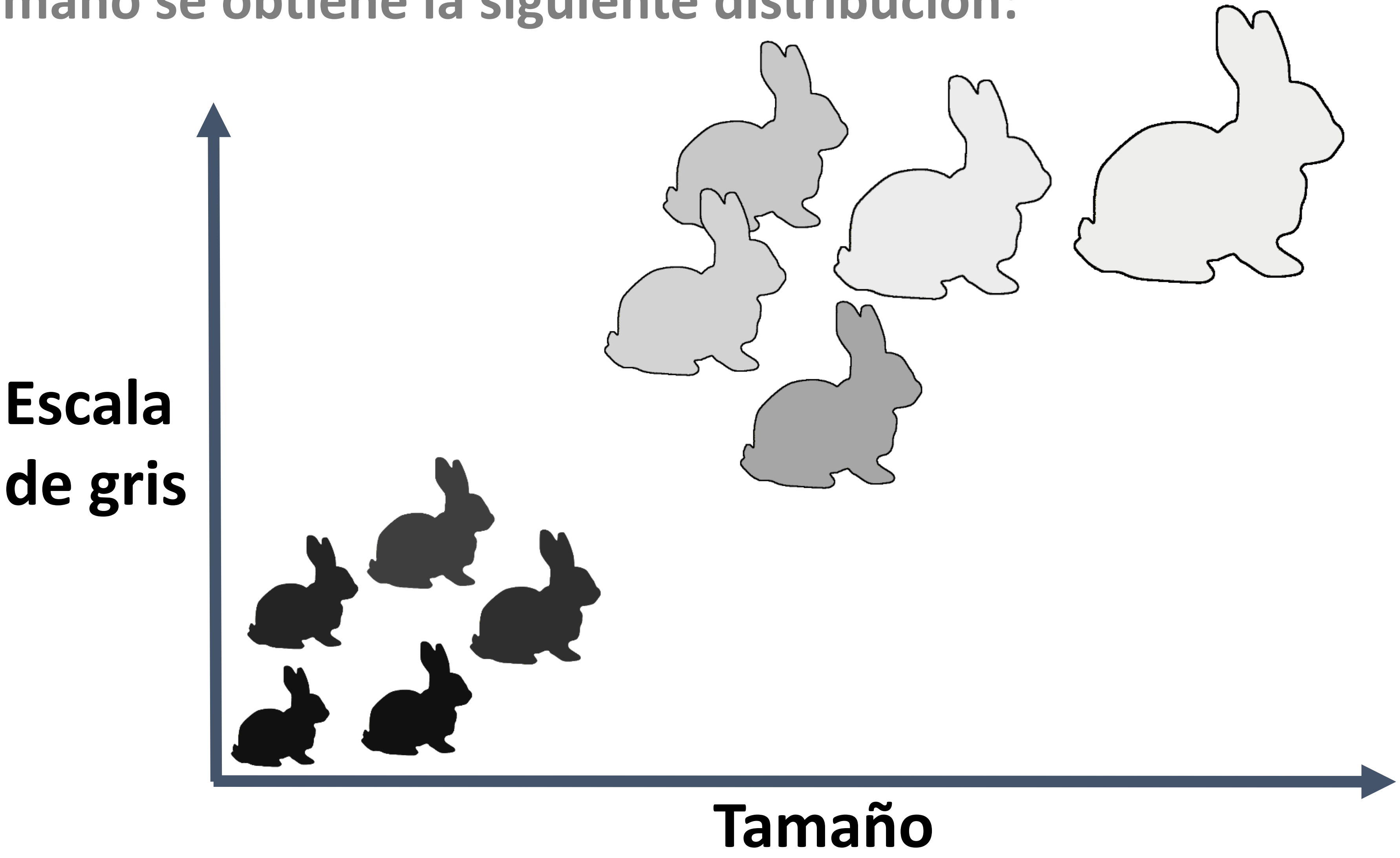
Clasificación: Ejemplo

En un criadero de conejos tenemos una población total de 10 conejos en edad adulta: 5 machos y 5 hembras.

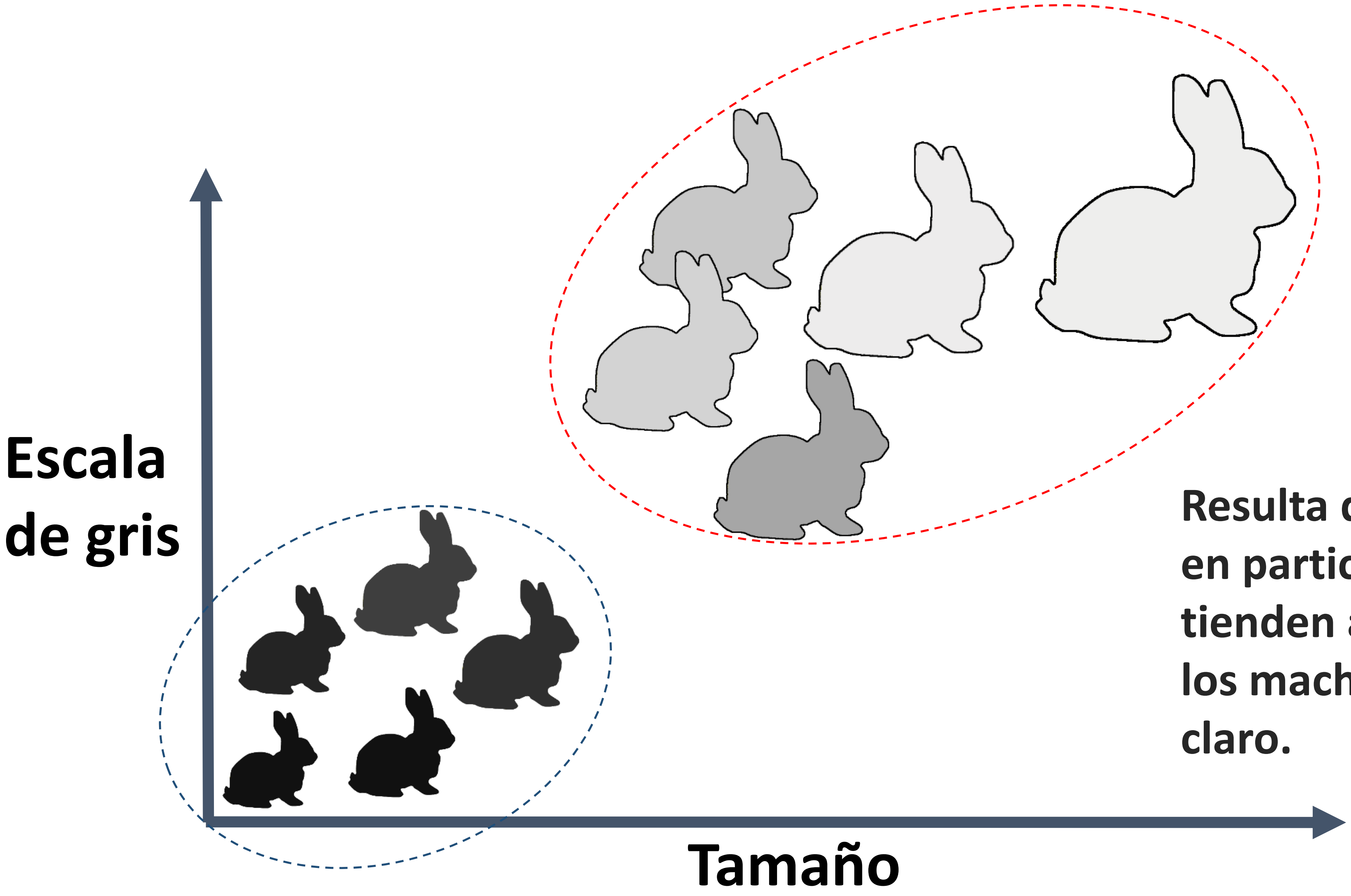


Clasificación: Ejemplo

Cuando el cuidador clasifica a los individuos de acuerdo a su tono de pelaje y tamaño se obtiene la siguiente distribución:

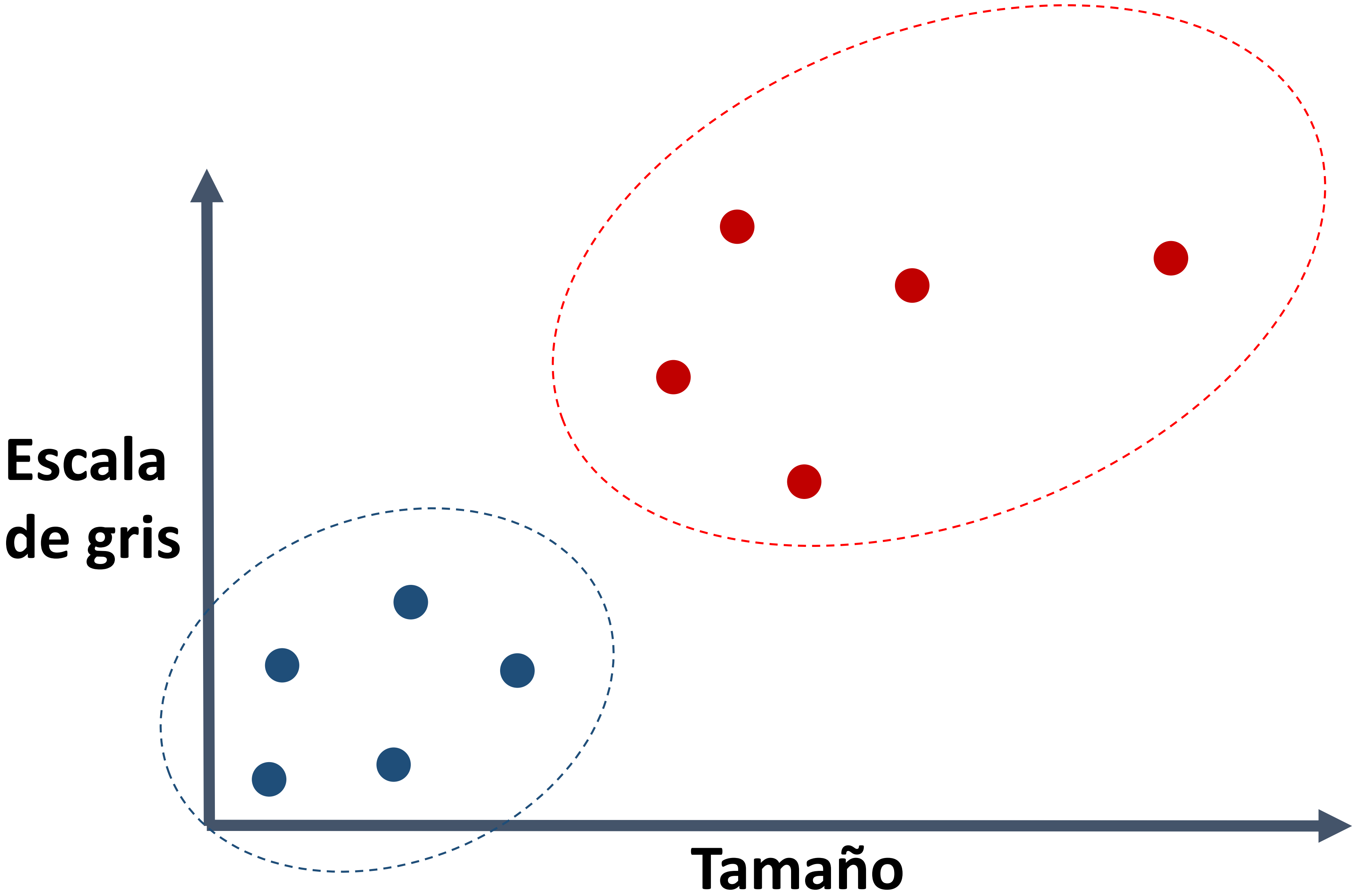


Clasificación: Ejemplo

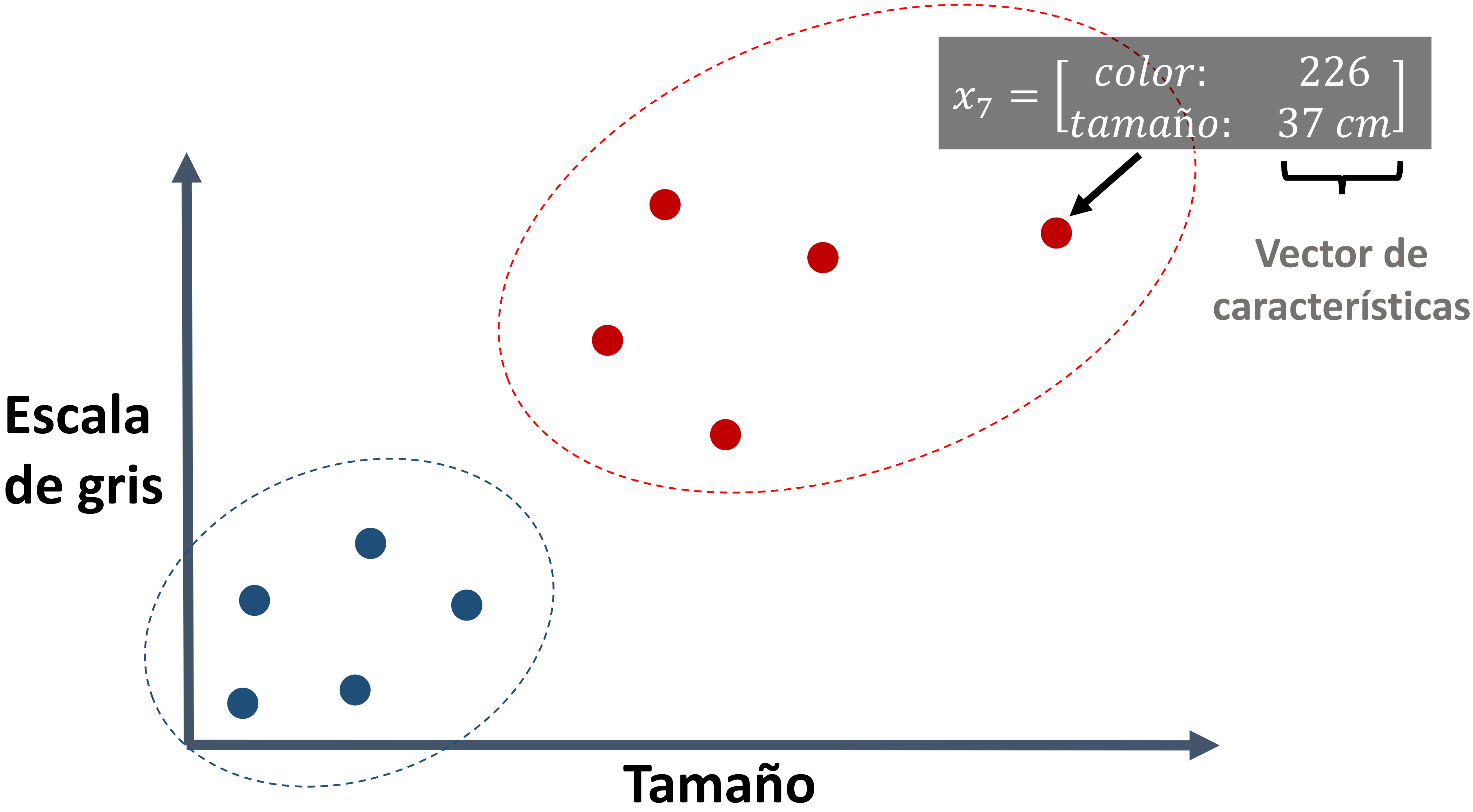


Resulta que para esta especie en particular, las hembras tienden a ser más grandes que los machos y con un pelaje más claro.

Clasificación: Feature space

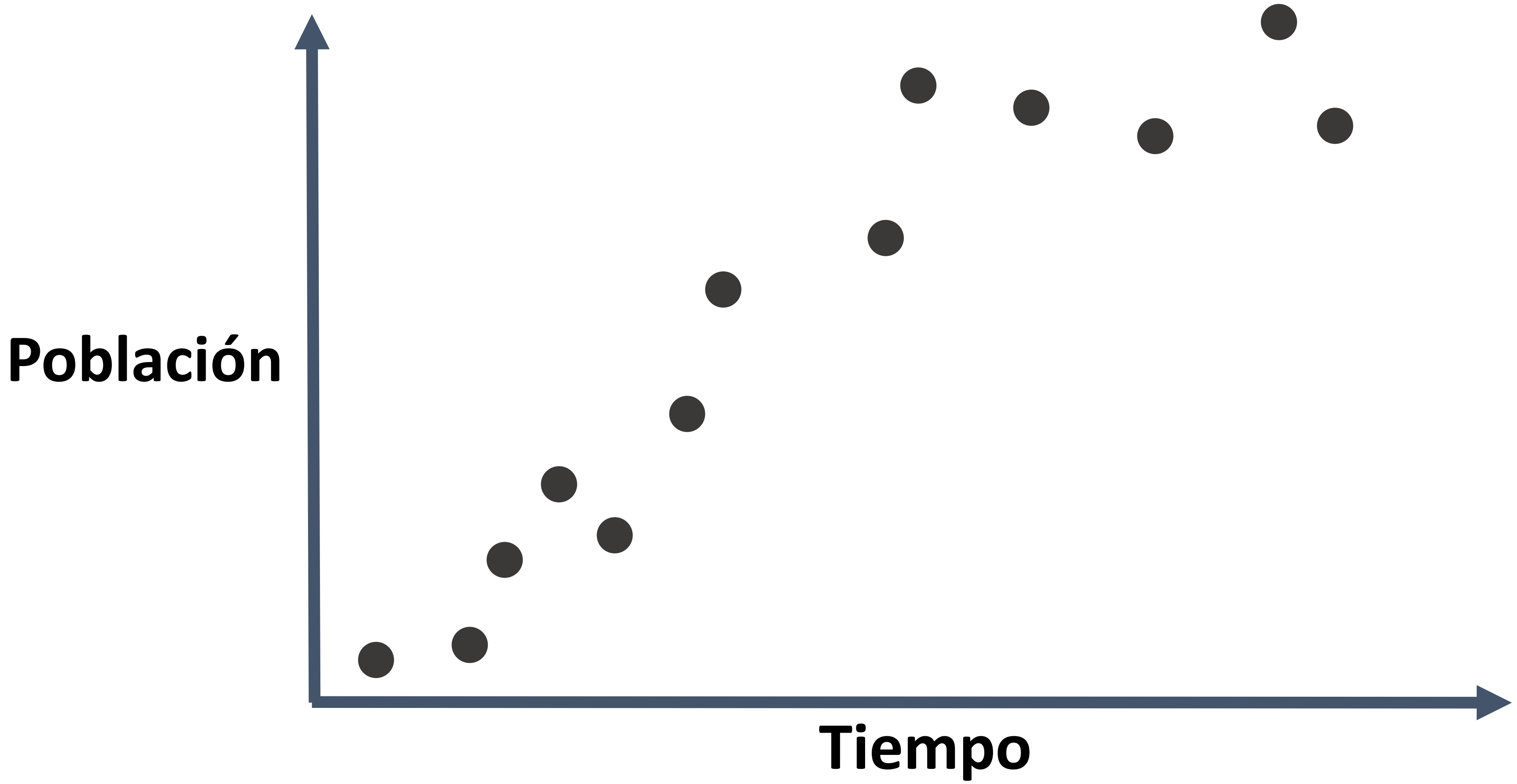


Clasificación: Feature space



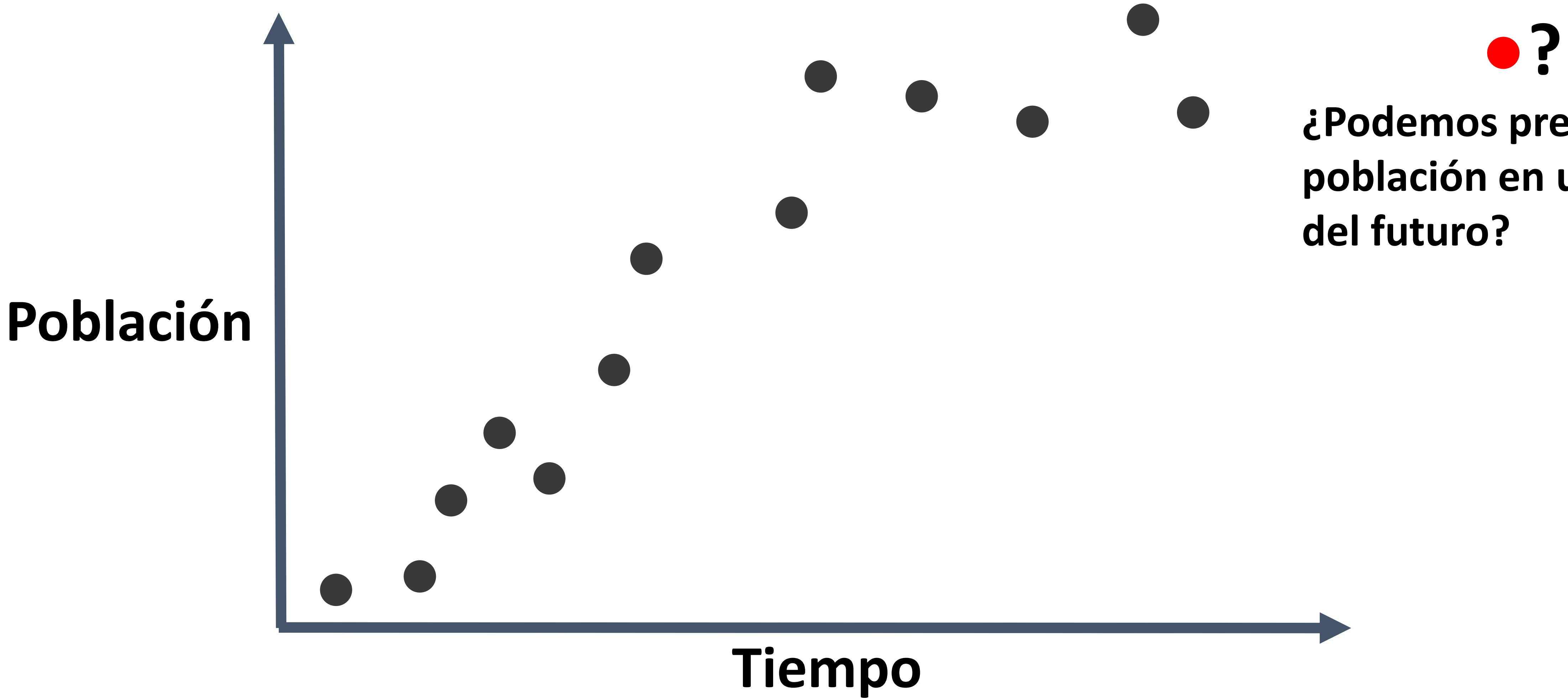
Regresión: Ejemplo

El dueño del mismo criadero de conejos llevó un registro de la población de conejos a lo largo del tiempo generando la siguiente gráfica:



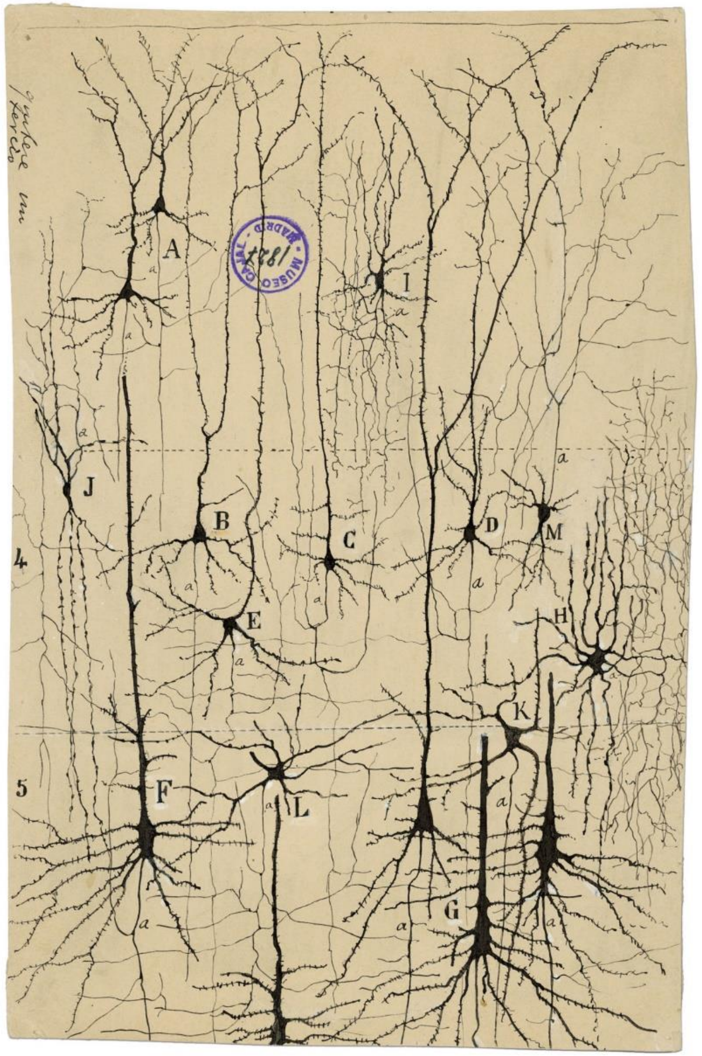
Regresión: Ejemplo

El dueño del mismo criadero de conejos llevó un registro de la población de conejos a lo largo del tiempo generando la siguiente gráfica:



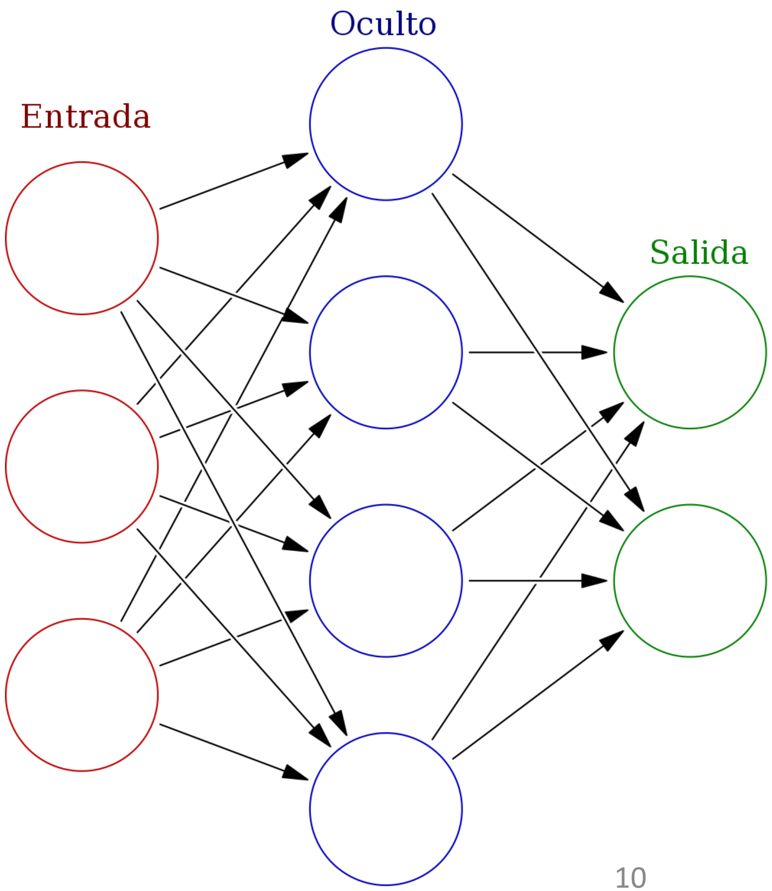
¿Podemos predecir la población en un momento del futuro?

Introducción: Redes Neuronales Artificiales

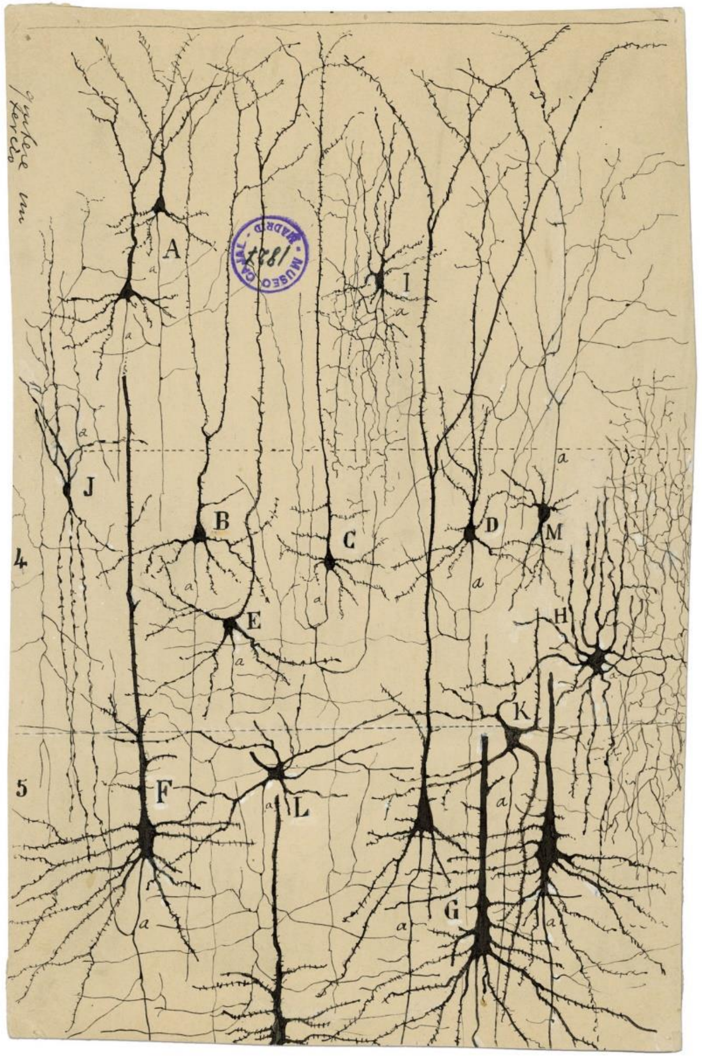


Dibujo de neuronas humanas, Santiago Ramón y Cajal

Las *redes neuronales artificiales* son **sistemas de computo conexionistas**. Esto significa que el computo final es el resultado de la ejecución de funciones matemáticas interconectadas por enlaces ponderados.



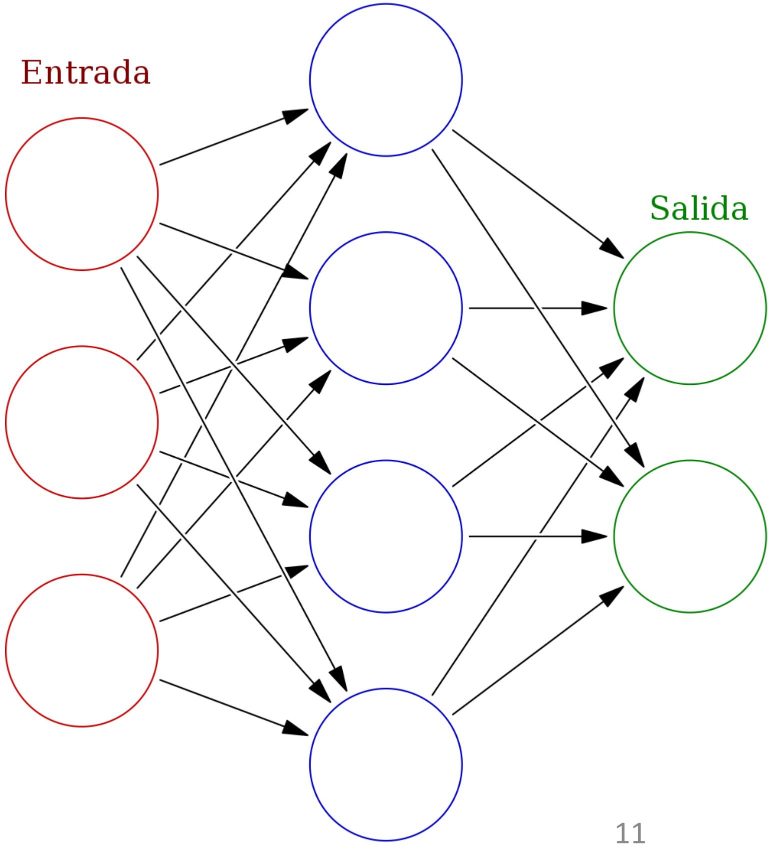
Introducción: Redes Neuronales Artificiales



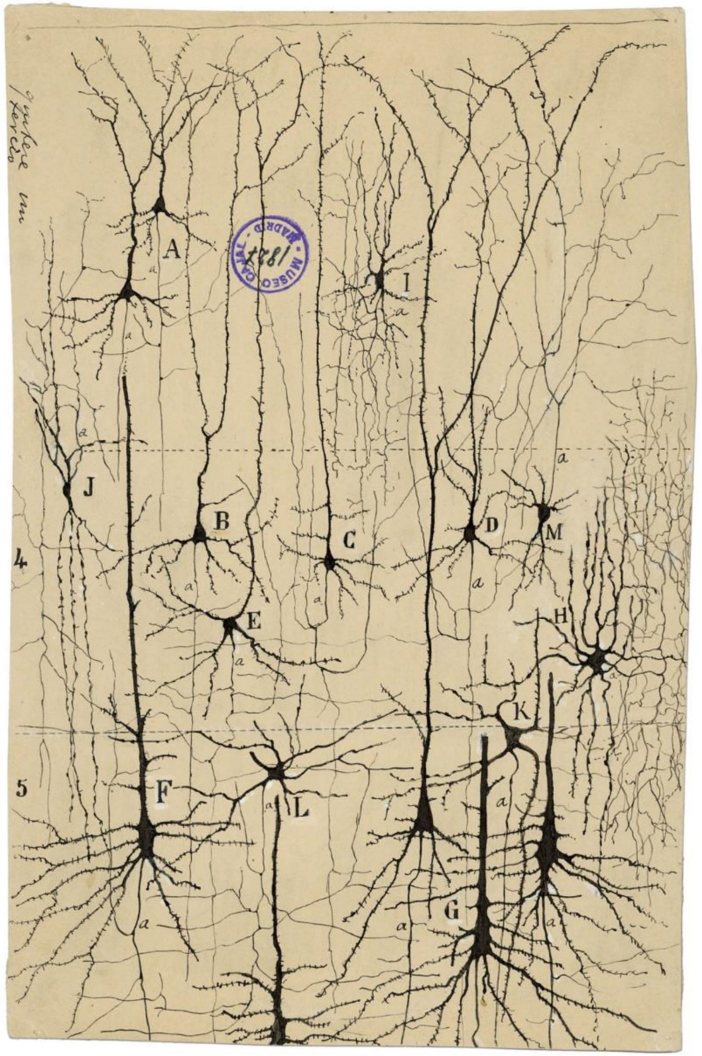
Dibujo de neuronas humanas, Santiago Ramón y Cajal

Las *redes neuronales artificiales* son **sistemas de computo conexionistas**. Esto significa que el computo final es el resultado de la ejecución de funciones matemáticas interconectadas por enlaces ponderados.

Su arquitectura está inspirada en las redes neuronales naturales.



Introducción: Redes Neuronales Artificiales



Dibujo de neuronas humanas, Santiago Ramón y Cajal

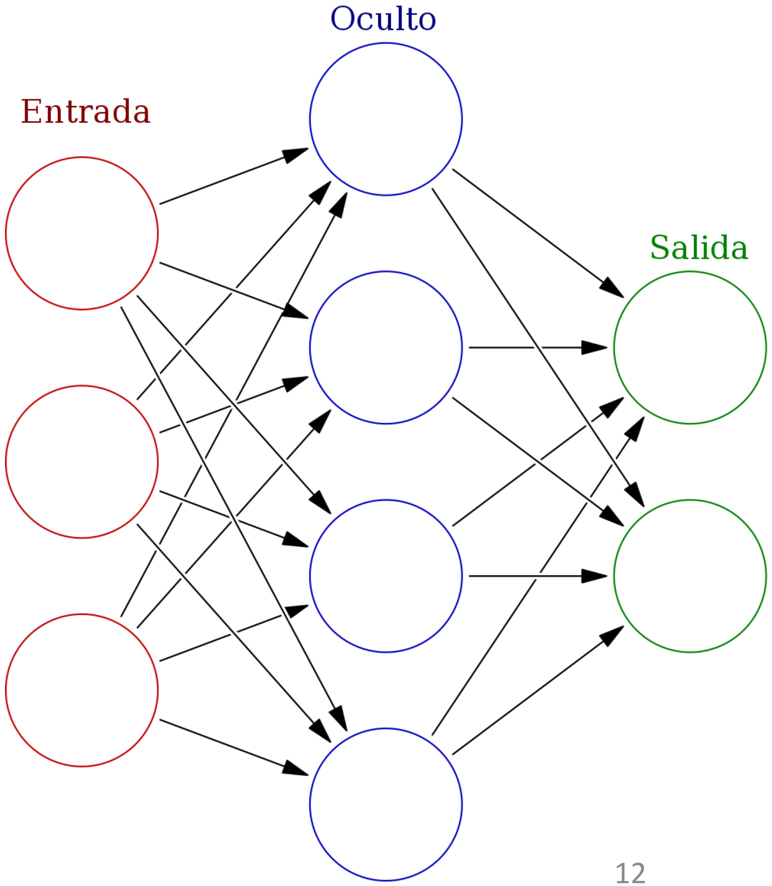
Las *redes neuronales artificiales* son **sistemas de computo conexionistas**. Esto significa que el computo final es el resultado de la ejecución de funciones matemáticas interconectadas por enlaces ponderados.

Su arquitectura está inspirada en las redes neuronales naturales.

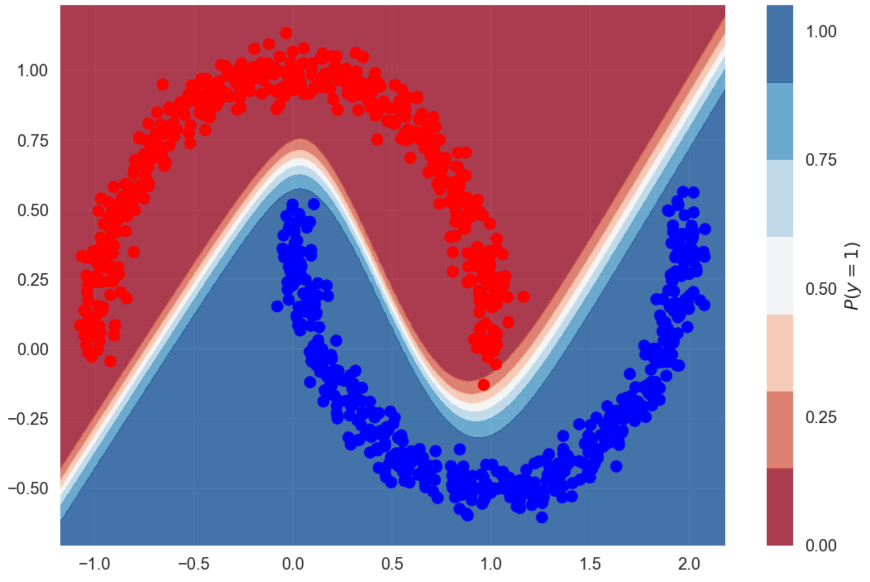
Por si solas sólo pueden hacer **dos** cosas:

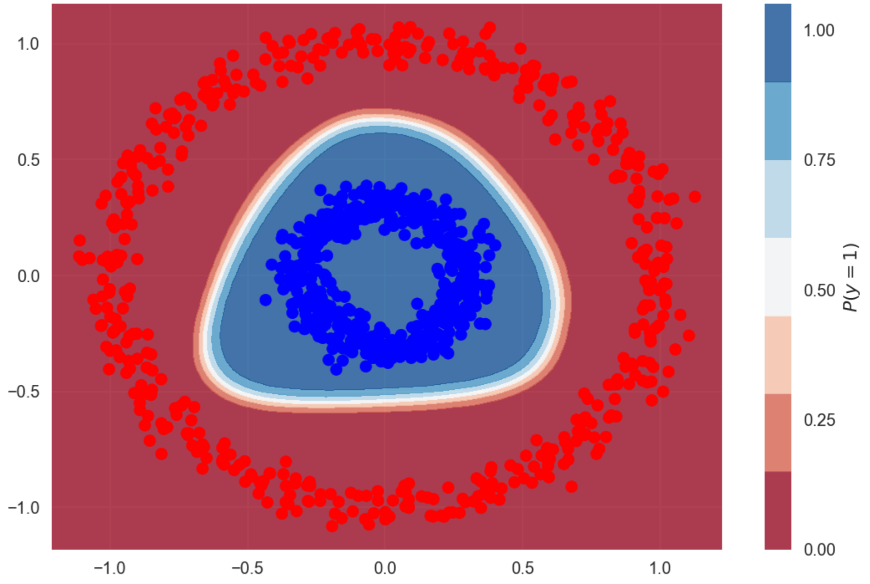
Clasificar vectores de características.

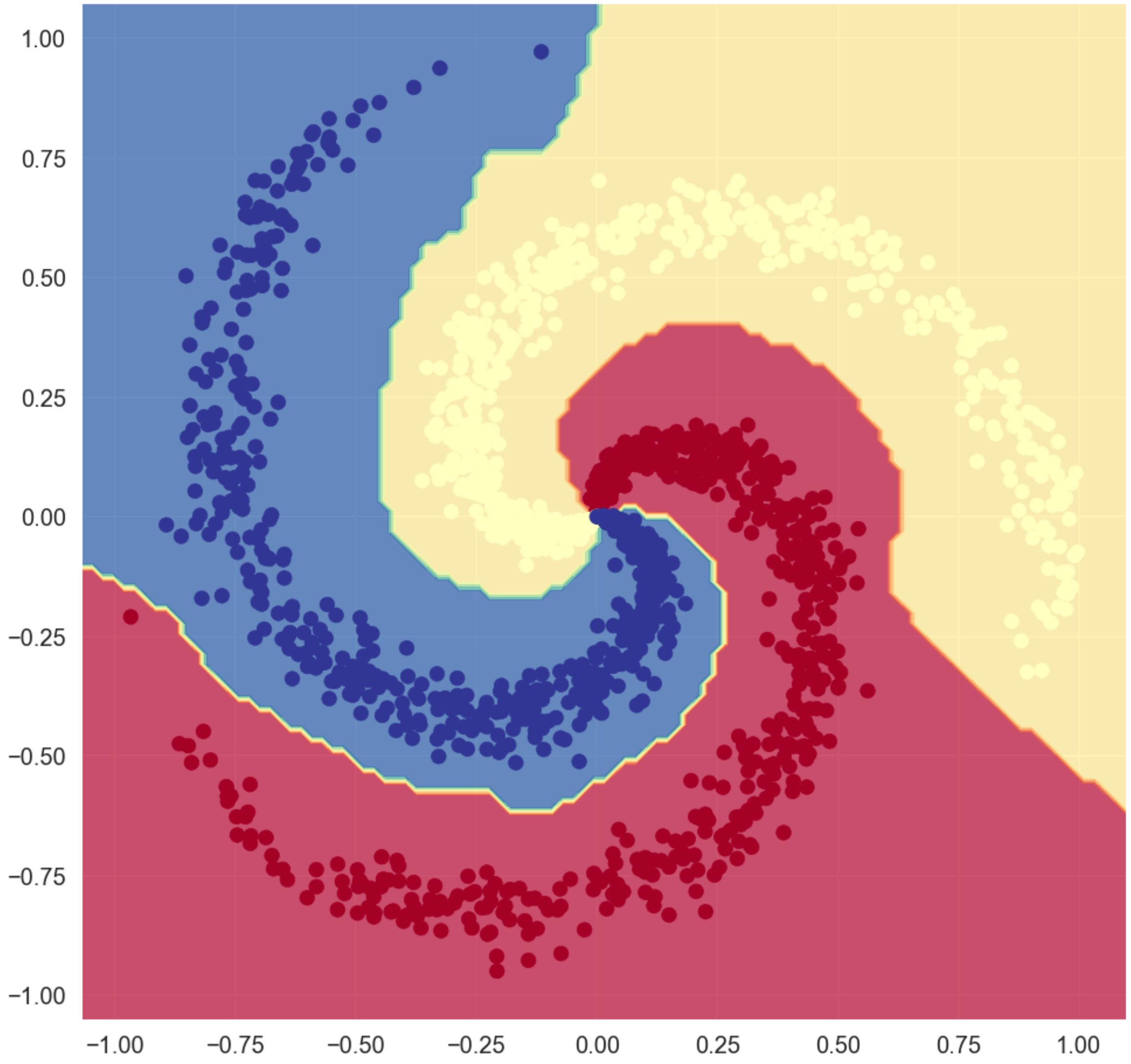
Aproximar funciones (modelar procesos).



Introducción: Redes Neuronales Artificiales







Algebra

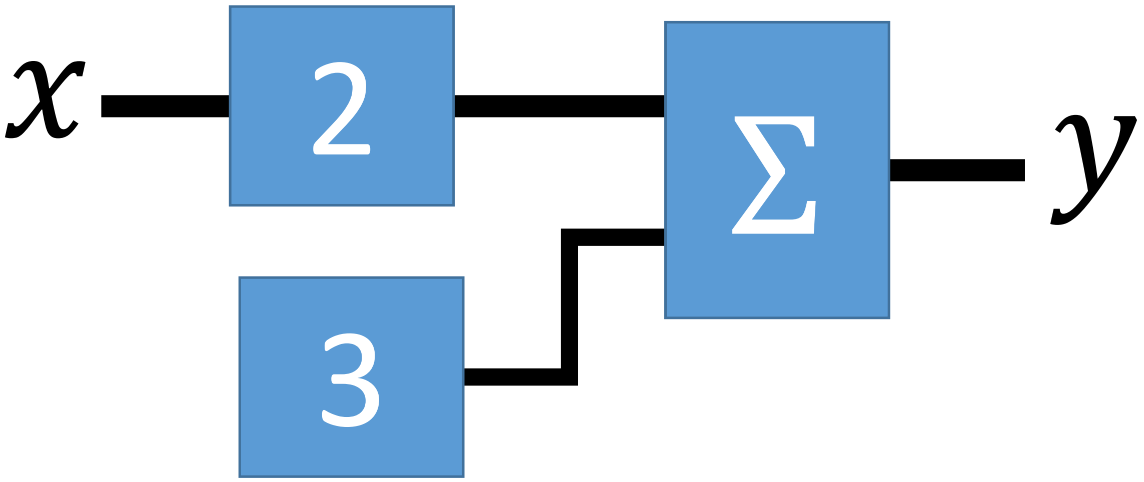
La estructura de las redes.

Algebra de bloques

Notación estándar

$$y = 2x + 3$$

Notación de bloques

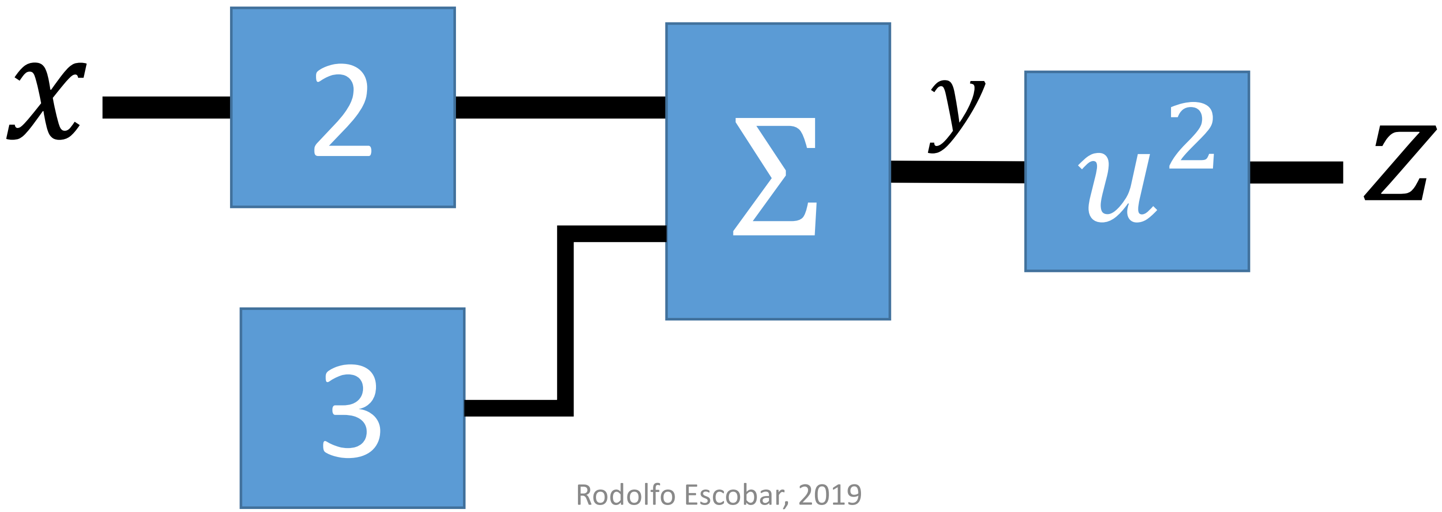


Algebra de bloques

Notación estándar

$$z = (2x + 3)^2$$

Notación de bloques

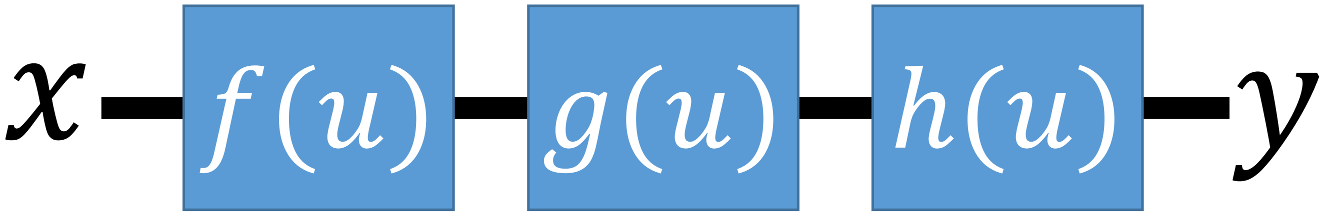


Algebra de bloques

Notación estándar

$$y = h\left(g\left(f(x)\right)\right)$$

Notación de bloques



Algebra lineal: aritmética vectorial

Vectores columna

$$\boldsymbol{u} = \begin{bmatrix} 0.6 \\ 1.6 \end{bmatrix}$$

$$\boldsymbol{v} = \begin{bmatrix} 2 \\ 2.2 \end{bmatrix}$$

La **dimensión** de un vector es el número de sus componentes. Sólo pueden sumarse/restarse vectores de la misma dimensión.

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}$$

Algebra lineal: aritmética vectorial

Vectores columna

$$\boldsymbol{u} = \begin{bmatrix} 0.6 \\ 1.6 \end{bmatrix}$$

$$\boldsymbol{v} = \begin{bmatrix} 2 \\ 2.2 \end{bmatrix}$$

Suma de vectores

$$\boldsymbol{u} + \boldsymbol{v} = \begin{bmatrix} 0.6 + 2 \\ 1.6 + 2.2 \end{bmatrix} = \begin{bmatrix} 2.6 \\ 3.8 \end{bmatrix}$$

Escalamiento

$$5\boldsymbol{v} = 5\begin{bmatrix} 2 \\ 2.2 \end{bmatrix} = \begin{bmatrix} (5)(2) \\ (5)(2.2) \end{bmatrix}$$

Algebra lineal: producto punto

Vectores columna

$$\boldsymbol{u} = \begin{bmatrix} 0.6 \\ 1.6 \end{bmatrix}$$

$$\boldsymbol{v} = \begin{bmatrix} 2 \\ 2.2 \end{bmatrix}$$

El **producto punto** es una operación que toma como argumento dos vectores y retorna un escalar.

$$\boldsymbol{v} \cdot \boldsymbol{u} = \sum_{i=0}^{n} v_i u_i$$

Algebra lineal: producto punto

Vectores columna

$$\boldsymbol{u} = \begin{bmatrix} 0.6 \\ 1.6 \end{bmatrix}$$

$$\boldsymbol{v} = \begin{bmatrix} 2 \\ 2.2 \end{bmatrix}$$

$$\boldsymbol{v} \cdot \boldsymbol{u} = (0.6)(2) + (1.6)(2.2)$$

$$\boldsymbol{v} \cdot \boldsymbol{u} = 4.7$$

Algebra lineal: matrices

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} \\ a_{21} & a_{22} & \cdots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} \end{bmatrix}$$

En matemáticas, una matriz es un arreglo bidimensional de números. Las dimensiones de una matriz suelen denotarse por $m \times n$ dónde m es el número de filas y n el de columnas.

Algebra lineal: aritmética matricial

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$

Suma de matrices

$$A + B = \begin{bmatrix} a_{00} + b_{00} & a_{01} + b_{01} \\ a_{01} + b_{10} & a_{11} + b_{11} \end{bmatrix}$$

Escalamiento

$$7A = \begin{bmatrix} 7a_{00} & 7a_{01} \\ 7a_{10} & 7a_{11} \end{bmatrix}$$

Algebra lineal: producto matricial

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$

$$AB = C: c_{ij} = \sum_{k=0}^{n} a_{ik} b_{kj}$$

Algebra lineal: producto matricial

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} a_{00}b_{00} + a_{01}b_{10} & \\ & \end{bmatrix}$$

Algebra lineal: producto matricial

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} \\ a_{10}b_{00} + a_{11}b_{10} \end{bmatrix}$$

Algebra lineal: producto matricial

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & a_{00}b_{01} + a_{01}b_{11} \\ c_{10} & \end{bmatrix}$$

Algebra lineal: producto matricial

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & a_{10}b_{01} + a_{11}b_{11} \end{bmatrix}$$

Algebra lineal: producto matricial

Sea A una matriz $m_A \times n_A$ y B una matriz $m_B \times n_B$, entonces el producto matricial AB puede realizarse **sólo si**: $n_A = m_B$

Algebra lineal: producto matricial

$$\underbrace{\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}}_{2 \times \boxed{2}} \overbrace{\begin{bmatrix} v_0 \\ v_1 \end{bmatrix}}^{\boxed{2} \times 1} = \begin{bmatrix} a_{00}v_0 + a_{01}v_1 \\ a_{10}v_0 + a_{11}v_1 \end{bmatrix}$$

Algebra lineal: notación matricial

$$x_0 + 2x_1 + 5x_2 = -17$$

$$2x_0 - 3x_1 + 2x_2 = -16$$

$$3x_0 + x_1 - x_2 = 3$$

Algebra lineal: notación matricial

$$x_0 + 2x_1 + 5x_2 = -17$$

$$2x_0 - 3x_1 + 2x_2 = -16$$

$$3x_0 + x_1 - x_2 = 3$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & 5 \\ 2 & -3 & 2 \\ 3 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -17 \\ -16 \\ 3 \end{bmatrix}$$

Algebra lineal: notación matricial

$$x_0 + 2x_1 + 5x_2 = -17$$

$$2x_0 - 3x_1 + 2x_2 = -16$$

$$3x_0 + x_1 - x_2 = 3$$

$$\begin{bmatrix} 1 & 2 & 5 \\ 2 & -3 & 2 \\ 3 & 1 & -1 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} -17 \\ -16 \\ 3 \end{bmatrix}$$

Algebra lineal: transpuesta

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

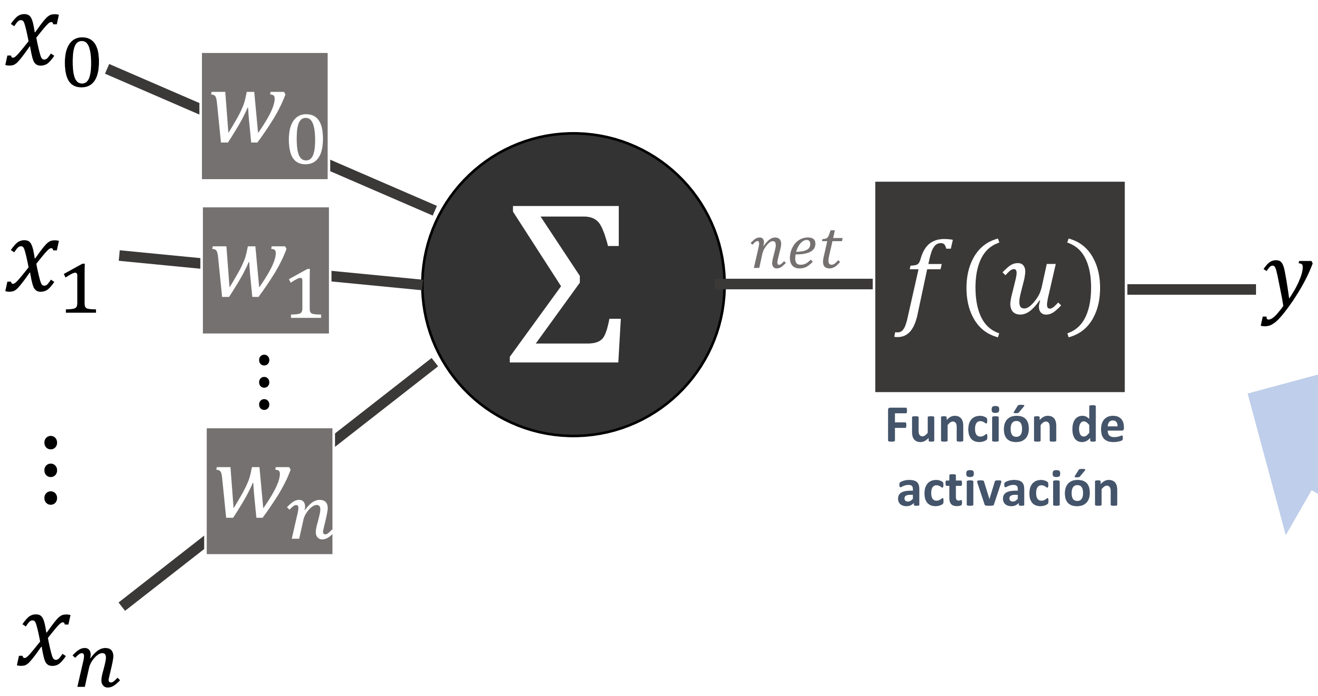
La transpuesta de una matriz es el intercambio de vectores por columnas:

$$A^T = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

La neurona artificial

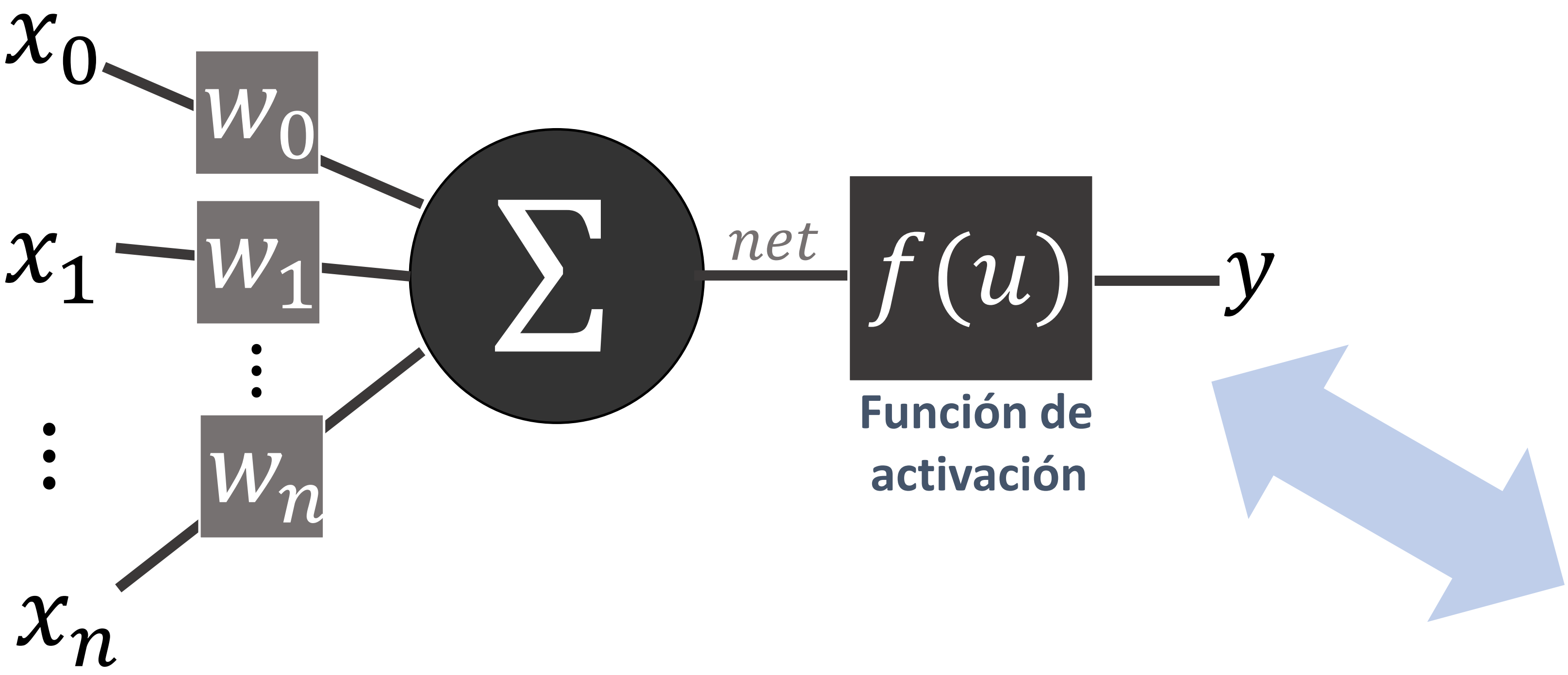
El bloque de construcción.

Neurona artificial



$$y = f\left(\sum_{i=0}^{n} w_i x_i\right)$$

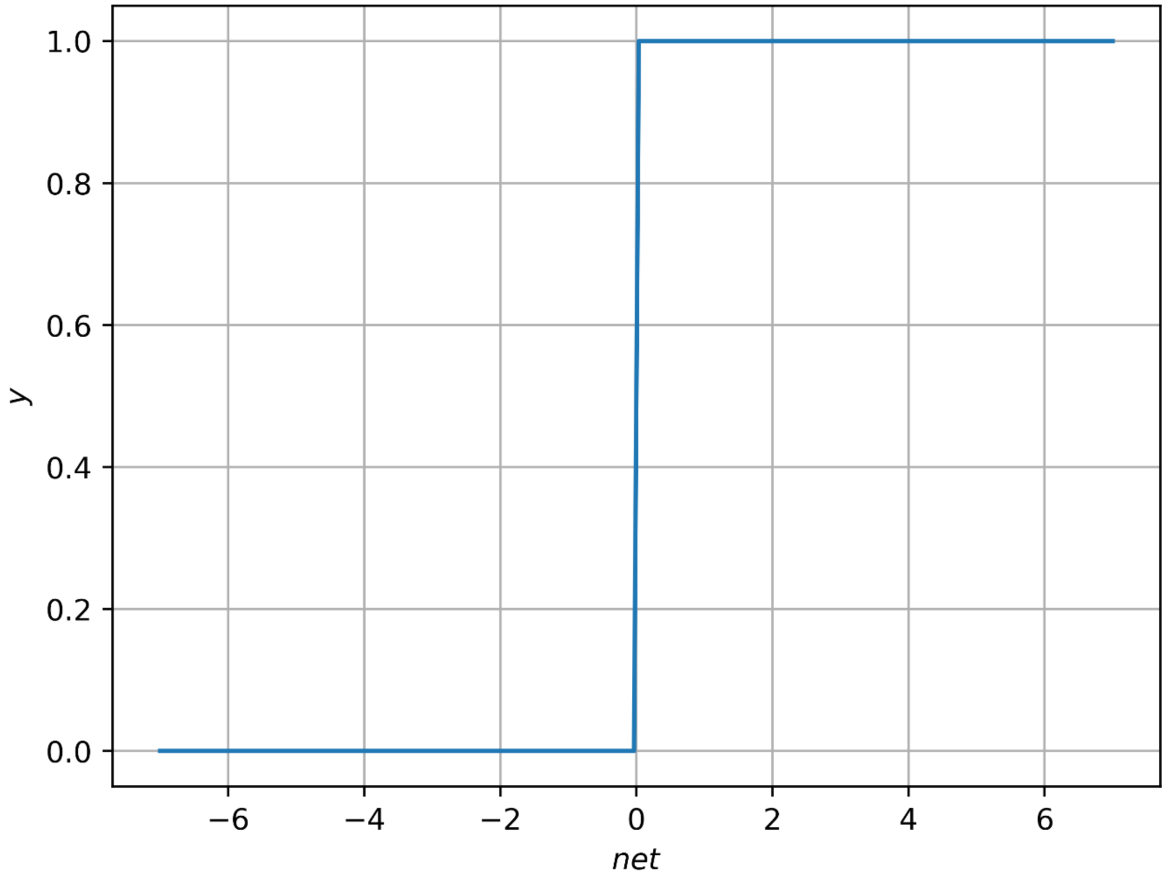
Neurona artificial



$$y = f(\boldsymbol{w} \cdot \boldsymbol{x})$$

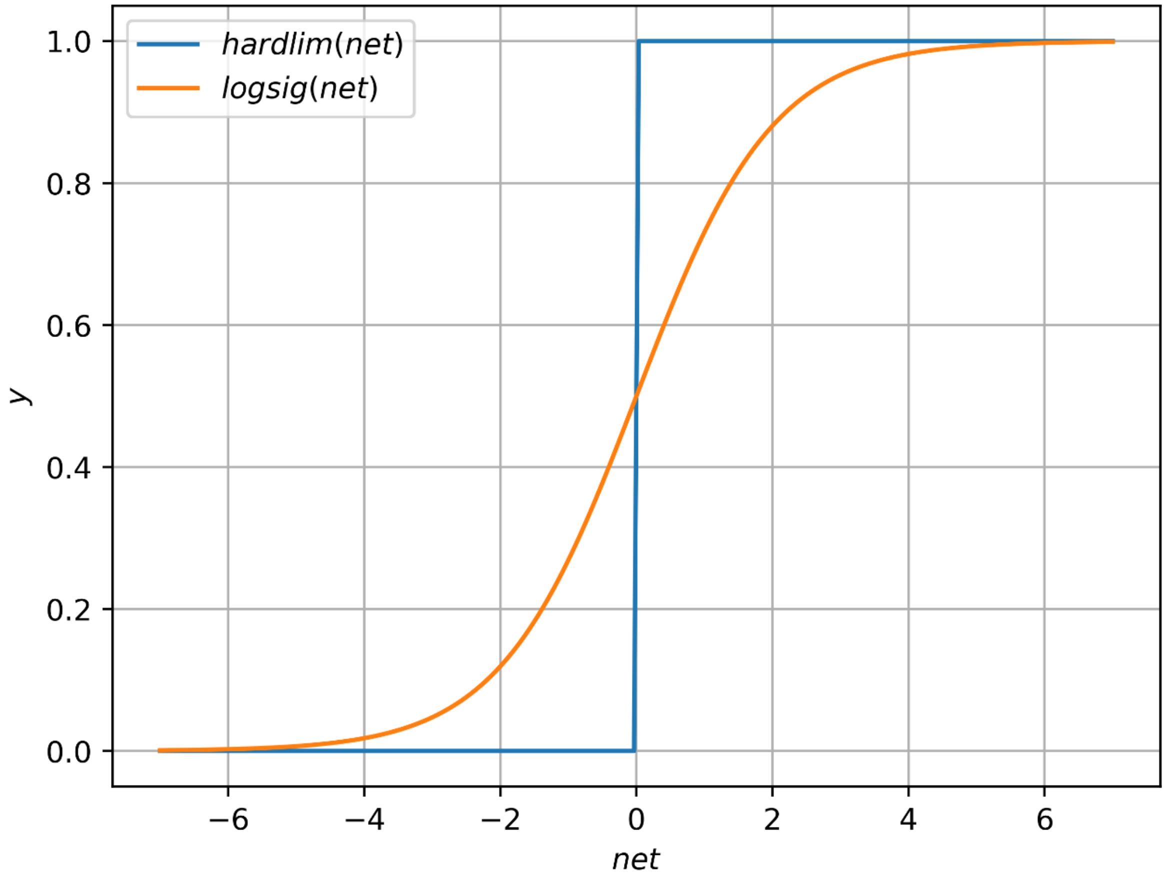
Funciones de activación: hardlim

$$hardlim(x) = \begin{cases} 1\,, x \geq 0 \\ 0 \;\; x < 0 \end{cases}$$

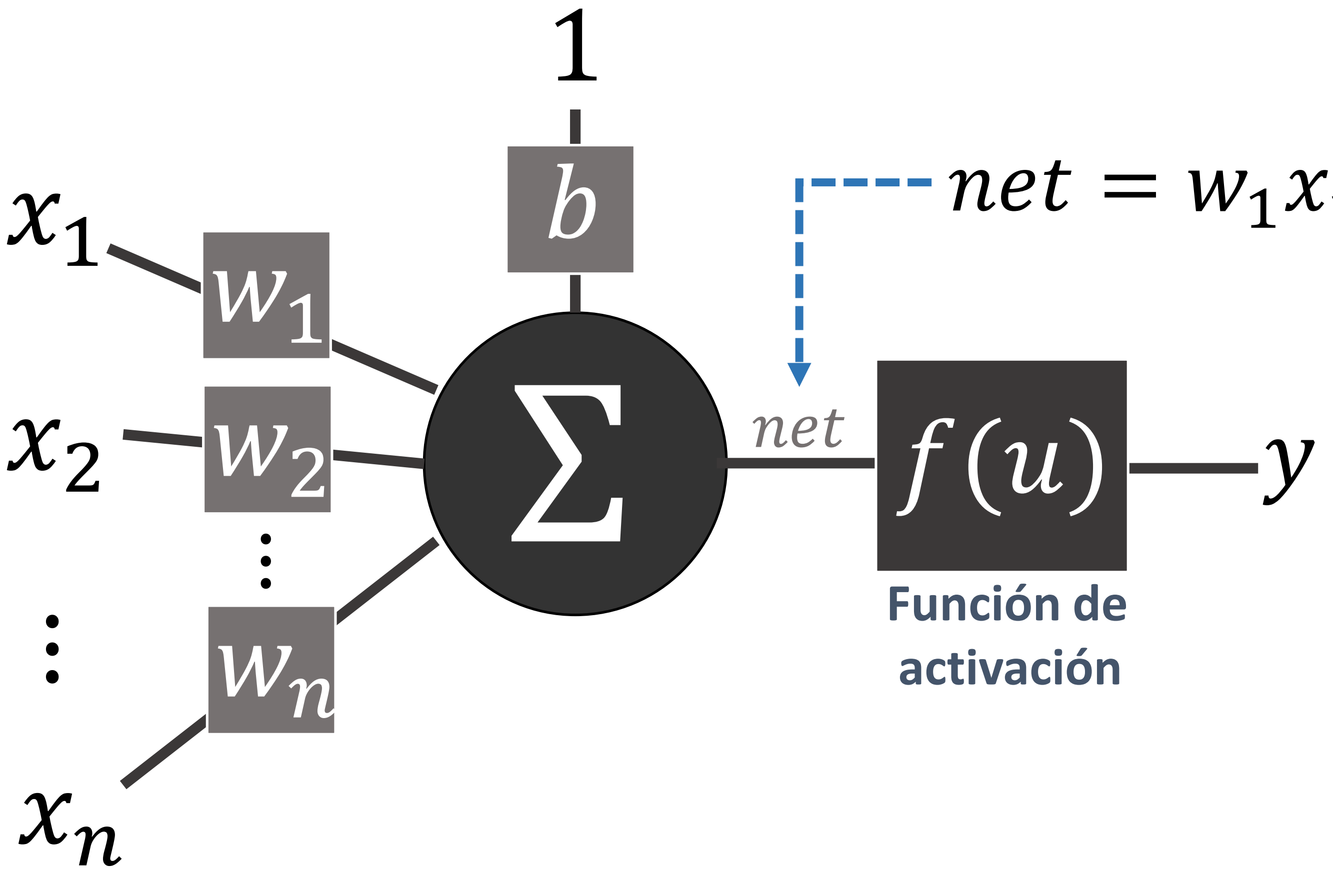


Funciones de activación: logsig

$$logsig(x) = \frac{1}{1 + e^{-x}}$$

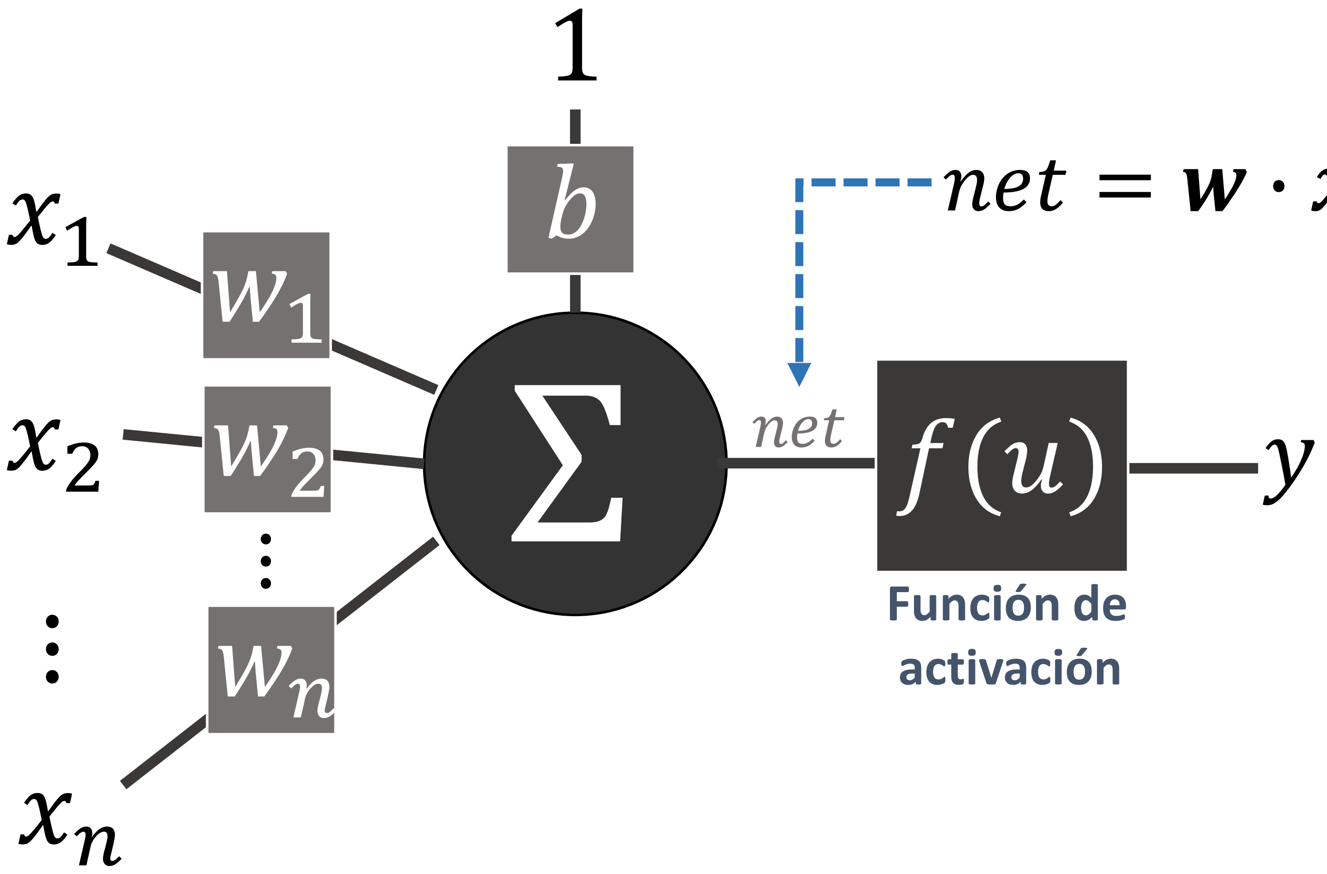


Umbral de activación



$$net = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Umbral de activación



$net = \boldsymbol{w} \cdot \boldsymbol{x} + b$

Practica #1 : Neurona artificial (no entrenada)

Escribe un código para una neurona con pesos y umbral ***aleatorios*** y una función de activación $hardlim$.

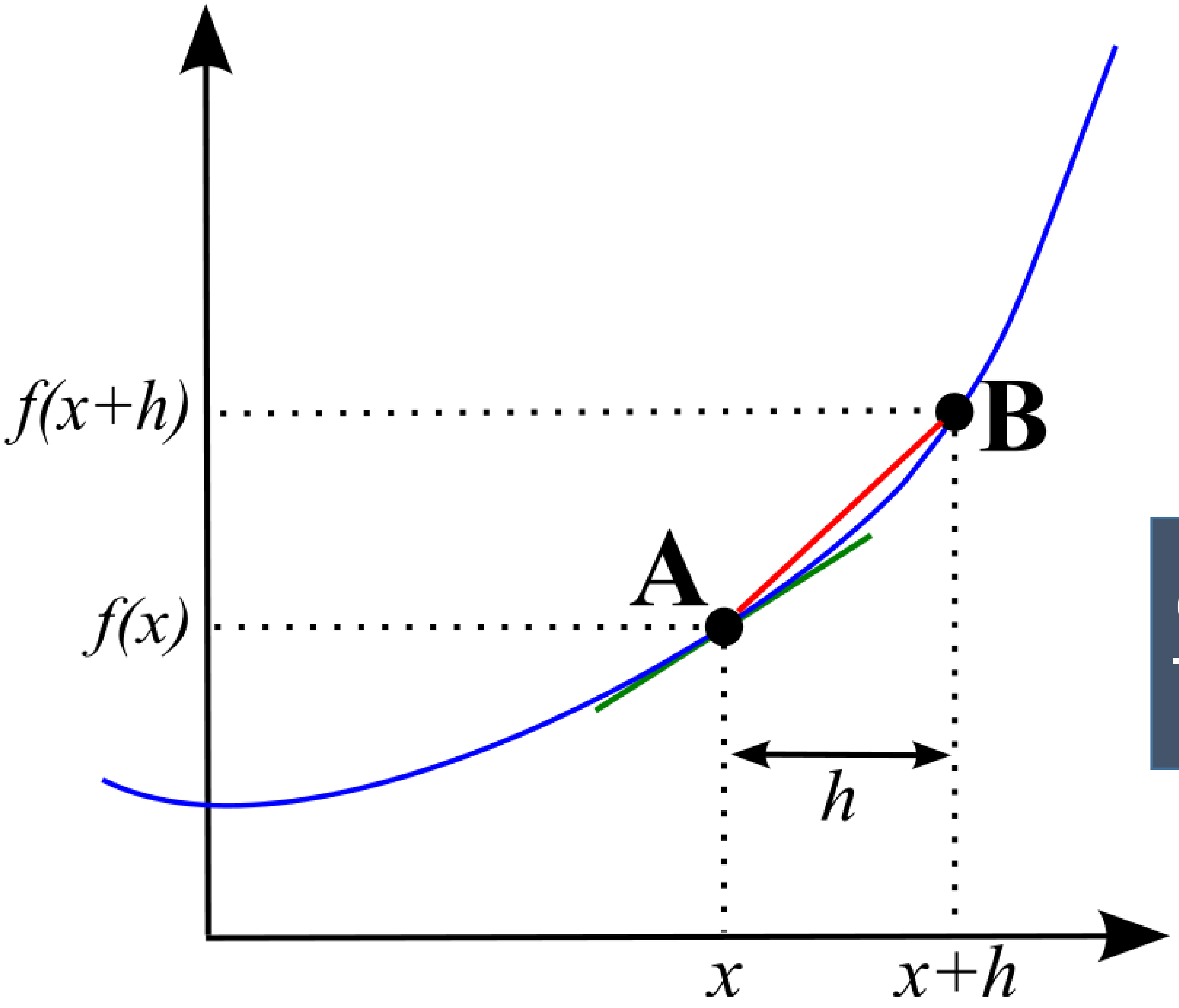
$$y = hardlim(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

$$hardlim(x) = \begin{cases} 1\,, x \geq 0 \\ 0 \;\; x < 0 \end{cases}$$

Cálculo

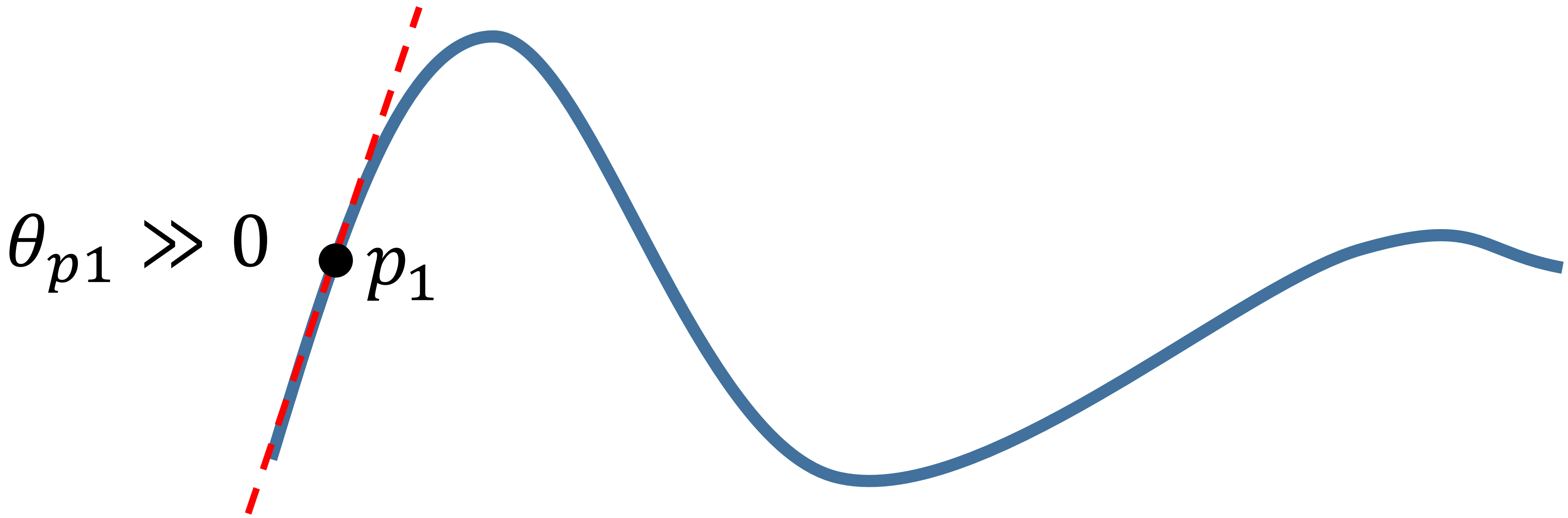
Como entrenar a tus redes neuronales.

La derivada

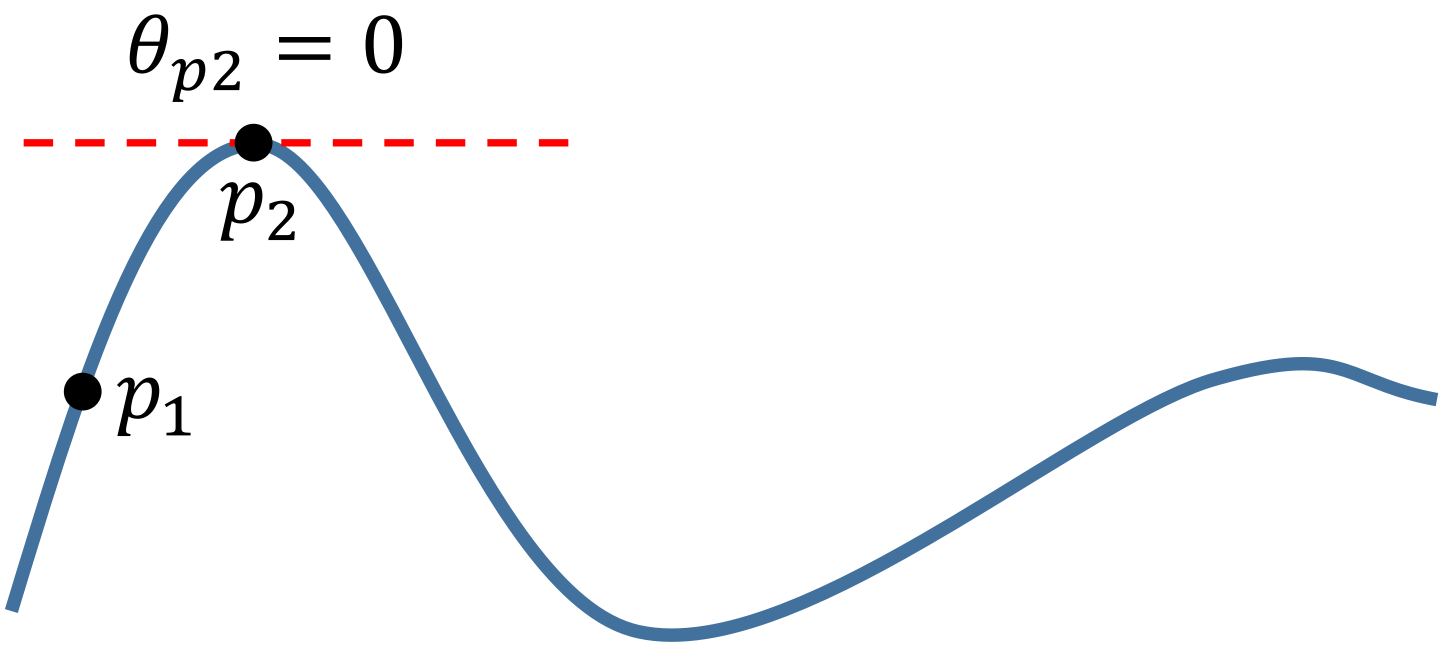


$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

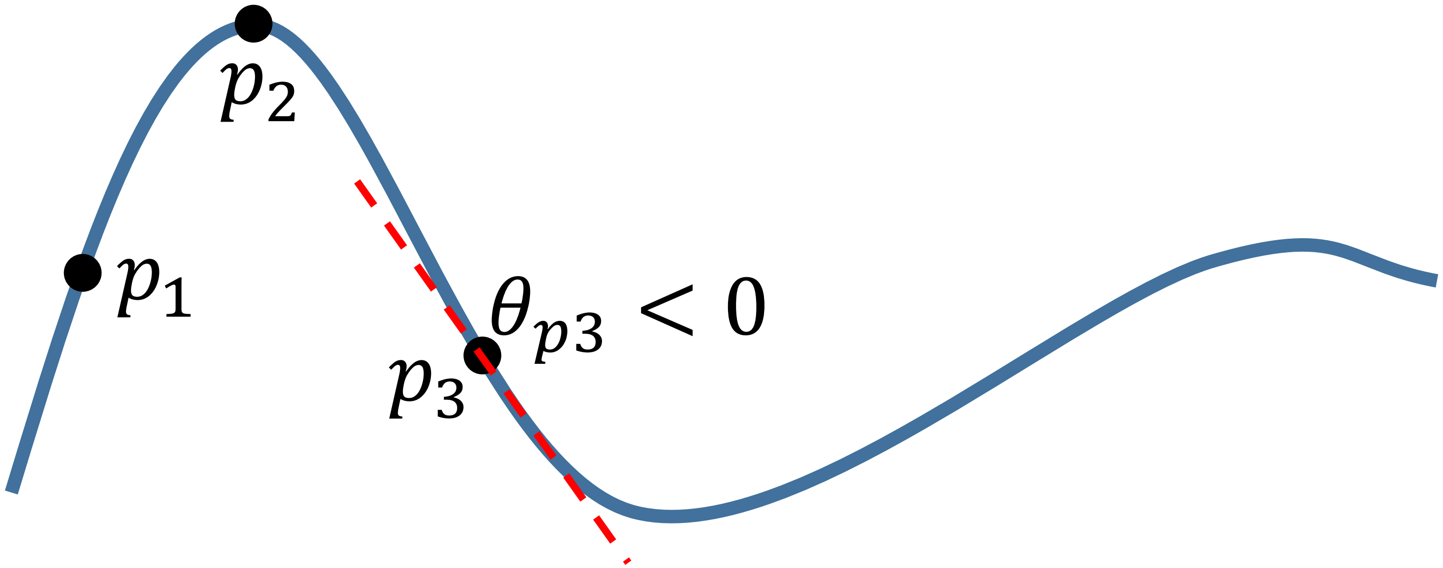
Ángulos de la recta tangente a una curva



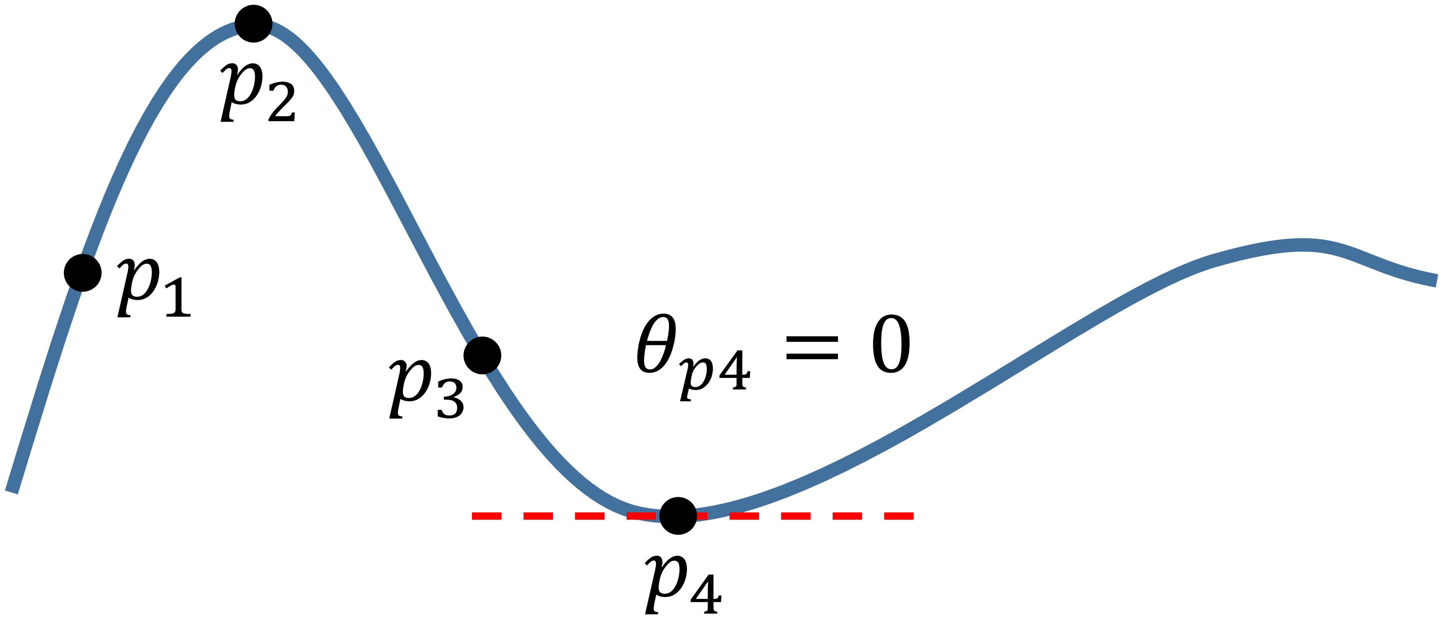
Ángulos de la recta tangente a una curva



Ángulos de la recta tangente a una curva

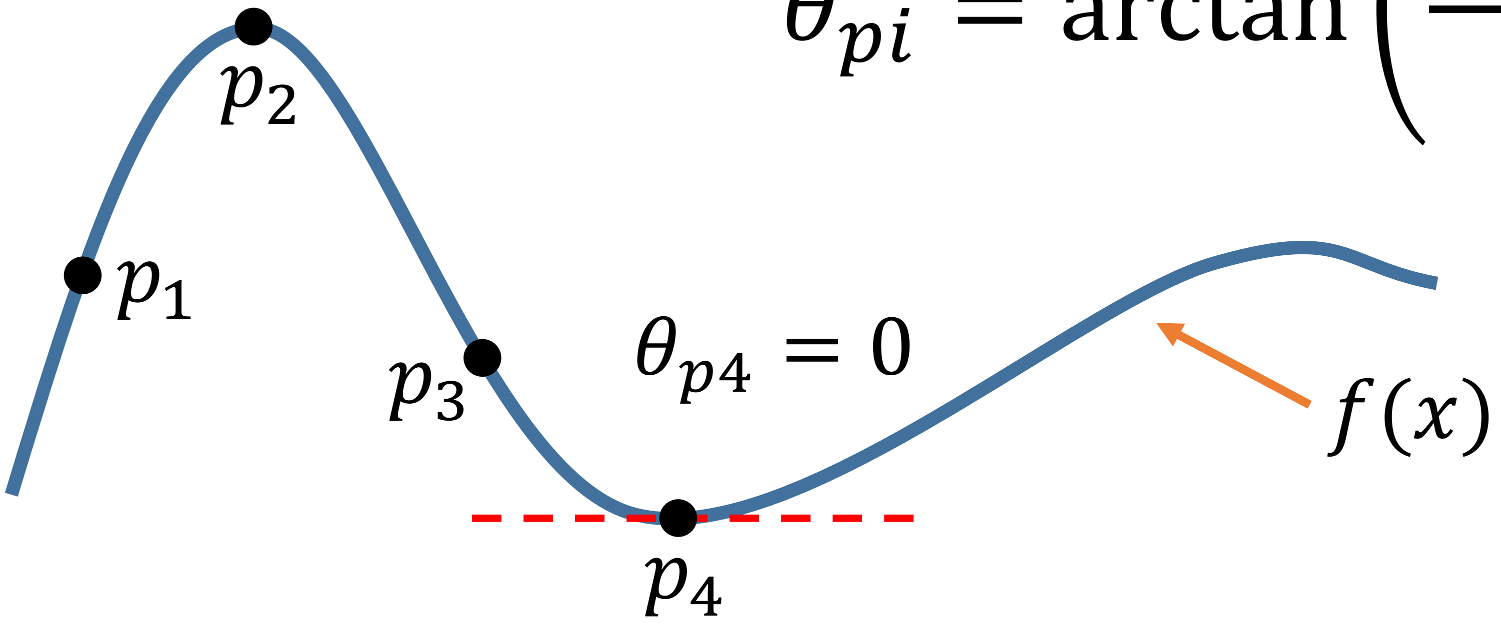


Ángulos de la recta tangente a una curva



Ángulos de la recta tangente a una curva

$$\theta_{pi} = \arctan\left(\left.\frac{df(x)}{dx}\right|_{pi}\right)$$



Derivada de funciones compuestas

$$f(x) = \tan\left(e^{\cos\left(x^3\right)}\right)$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

Derivada de funciones compuestas

$$f(x) = \tan\left(e^{\cos\left(x^3\right)}\right)$$

$$f(u) = \tan(u)$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

Derivada de funciones compuestas

$$f(x) = \tan\big(\underbrace{e^{\cos(x^3)}}_{u}\big)$$

$$\frac{df(u)}{du} = \sec^2(u)$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

Derivada de funciones compuestas

$$f(x) = \tan\big(\underbrace{e^{\cos(x^3)}}_{u}\big)$$

$$df(u) = \sec^2(u)du$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

Derivada de funciones compuestas

$$f(x) = \tan\big(\underbrace{e^{\overbrace{\cos(x^3)}^{v}}}_{u}\big)$$

$$df(u) = \sec^2(e^v)du$$

$$du = e^v dv$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

Derivada de funciones compuestas

$$f(x) = \tan\Big(\underbrace{e^{\overbrace{\cos(x^3)}^{v}}}_{u}\Big)$$

$$df(v) = \sec^2(e^v)e^v dv$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

Derivada de funciones compuestas

$$f(x) = \tan\Big(\underbrace{e^{\overbrace{\cos(\underbrace{x^3}_{w})}^{v}}}_{u}\Big)$$

$$df(w) = \sec^2\left(e^{\cos(w)}\right)e^{\cos(w)}dv$$

$$dv = -\text{sen(w)dw}$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

Derivada de funciones compuestas

$$f(x) = \tan\left(\underbrace{e^{\overbrace{\cos(\underbrace{x^3}_{w})}^{v}}}_{u}\right)$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

$$df(w) = -\sec^2\left(e^{\cos(w)}\right)e^{\cos(w)}\,sen(w)dw$$

Derivada de funciones compuestas

$$f(x) = \tan\Big(\underbrace{e^{\overbrace{\cos(\underbrace{x^3}_{w})}^{v}}}_{u}\Big)$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

$$df(x) = -3\sec^2\left(e^{\cos(x^3)}\right)e^{\cos(x^3)}\,sen(x^3)x^2dx$$

Derivada de funciones compuestas

$$f(x) = \tan\Big(\underbrace{e^{\overbrace{\cos(\underbrace{x^3}_{w})}^{v}}}_{u}\Big)$$

Función	Derivada
x^n	nx^{n-1}
e^x	e^x
$\cos(x)$	$-sen(x)$
$\tan(x)$	$\sec^2(x)$

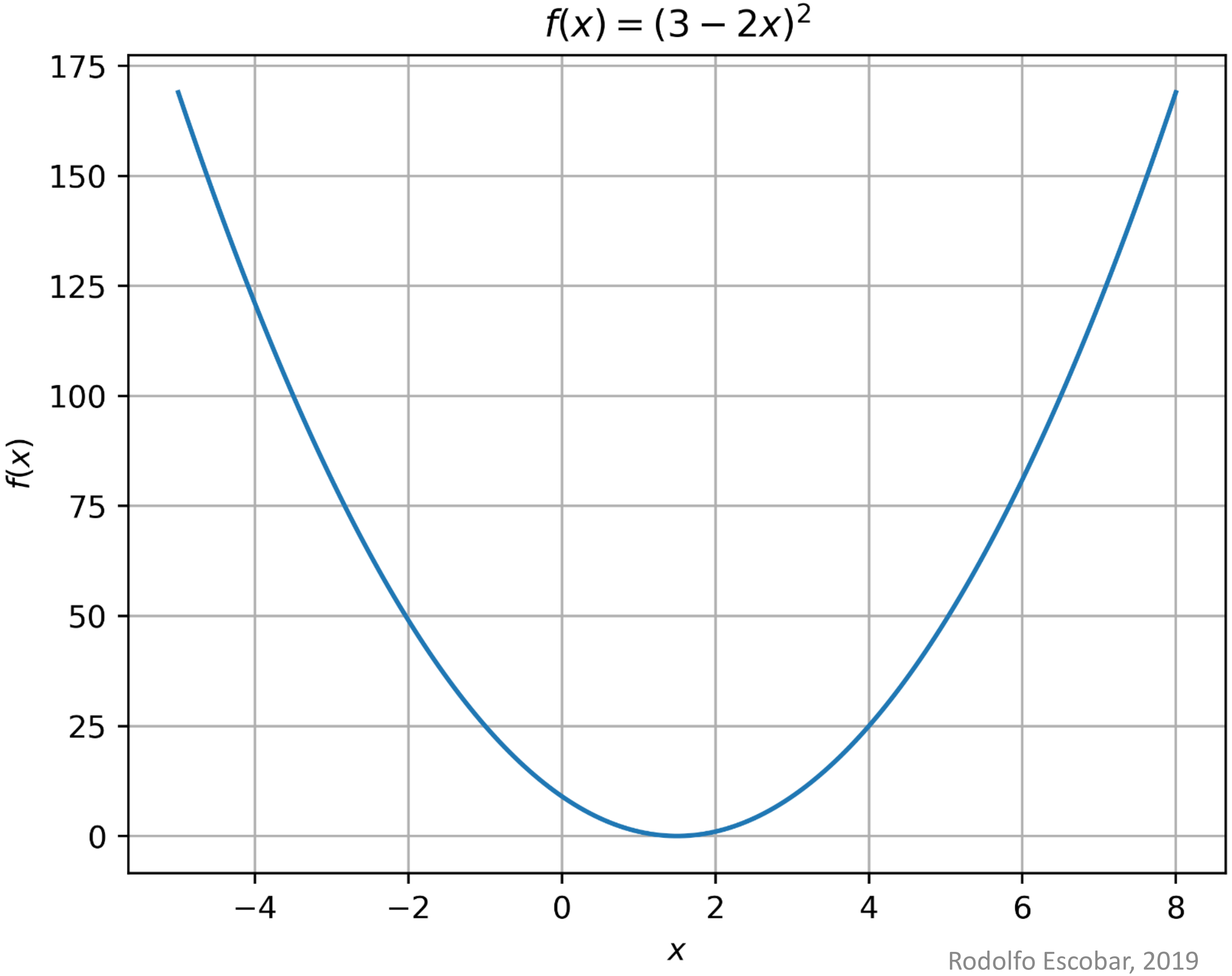
$$\frac{df(x)}{dx} = -3\sec^2\left(e^{\cos(x^3)}\right)e^{\cos(x^3)}\, sen(x^3)x^2$$

Regla de la cadena

$$\frac{f(x)}{dx} = \frac{df_1}{df_2}\frac{df_2}{df_3}\cdots\frac{df_n}{dx}$$

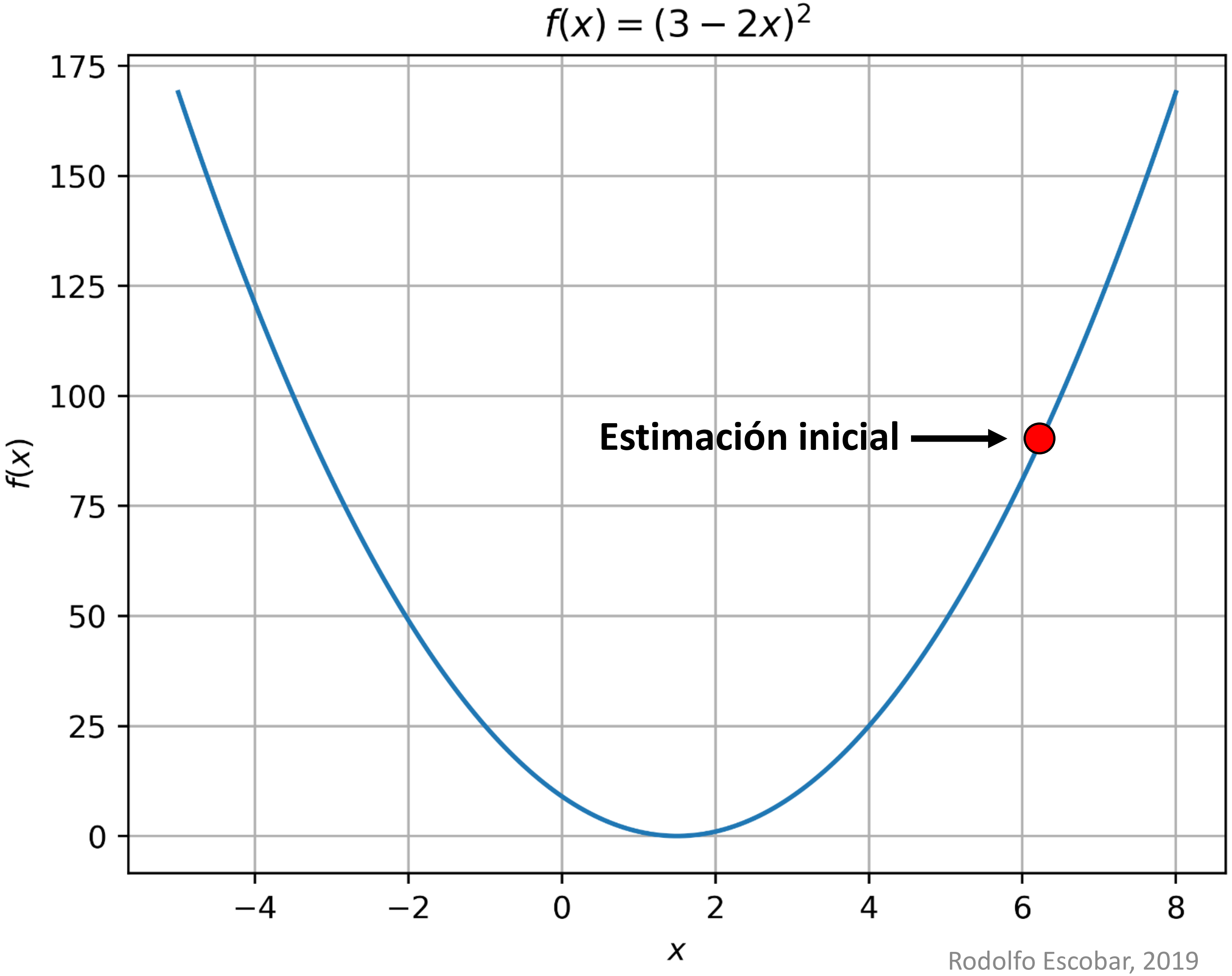
Minimización por descenso de gradiente

¿Qué valor de x minimiza la función $f(x) = (3 - 2x)^2$?



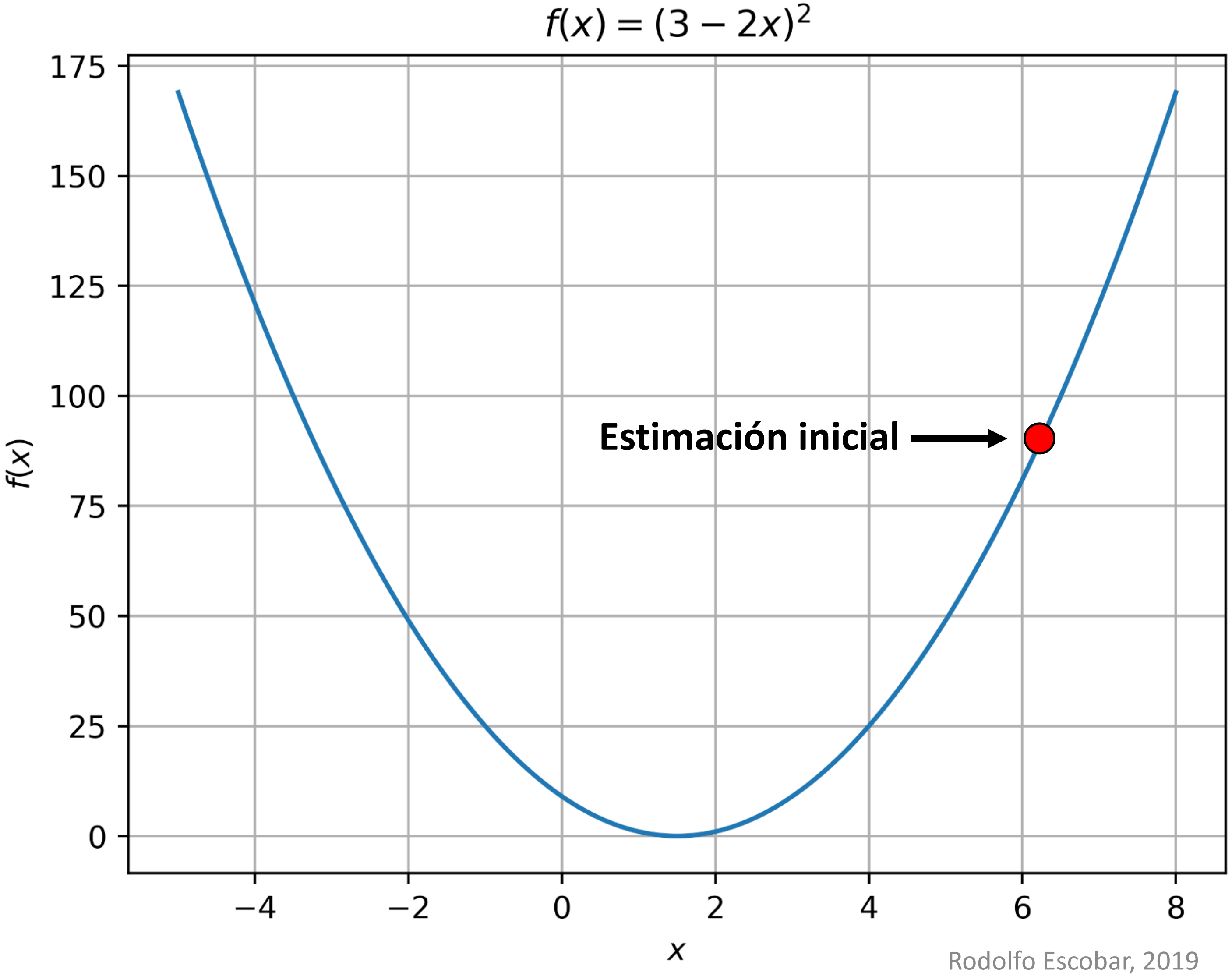
Minimización por descenso de gradiente

¿Qué valor de x minimiza la función $f(x) = (3 - 2x)^2$?



Minimización por descenso de gradiente

¿Qué valor de x minimiza la función $f(x) = (3 - 2x)^2$?

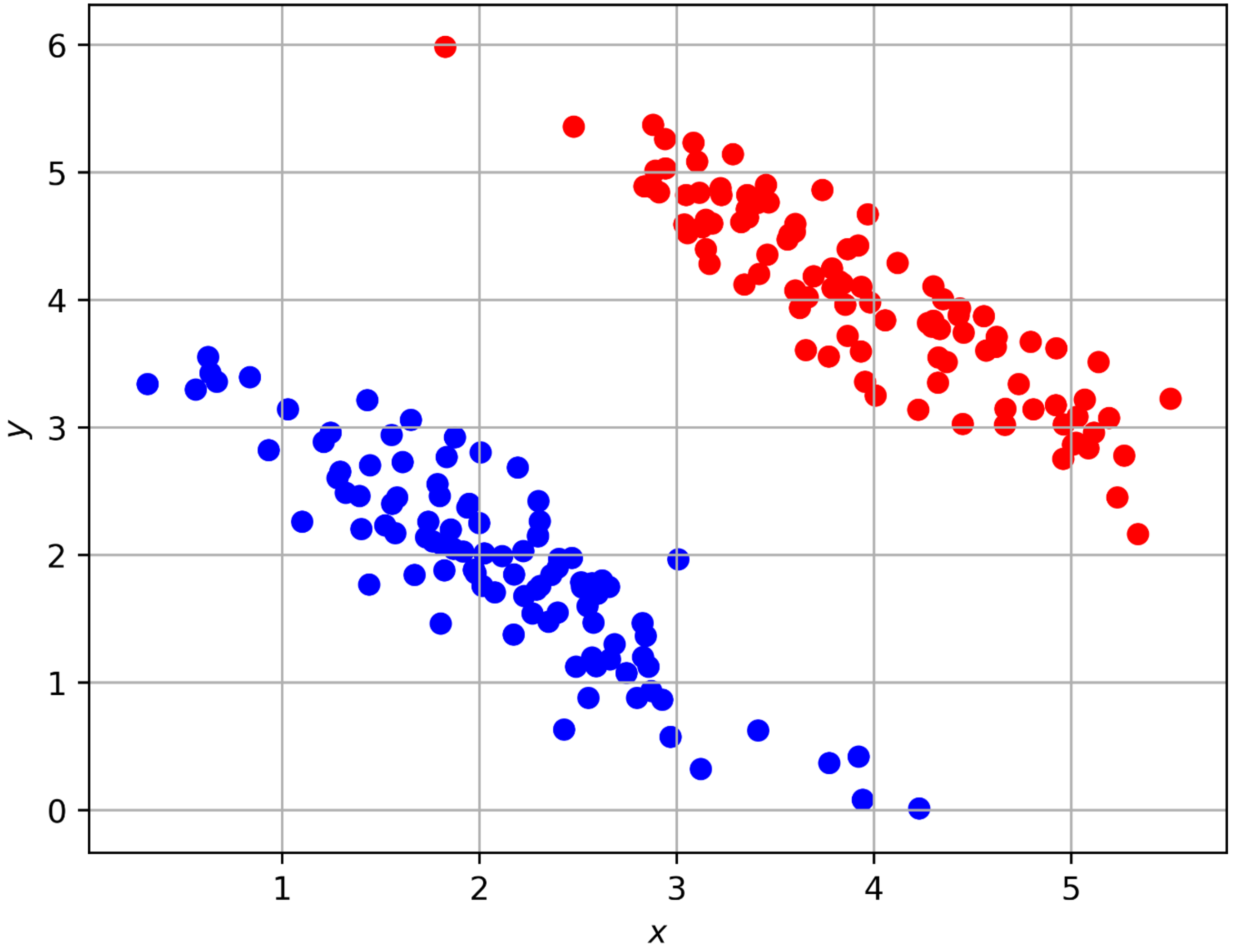


$$\boxed{x_{k+1} = x_k - a \left.\frac{df(x)}{dx}\right|_{x_k}}$$

Entrenamiento de redes neuronales

Neurona única.

Aprendizaje supervisado



dataset - Arreglo de NumPy

	0	1	2
108	2.82655	1.46985	0
109	3.92151	0.421043	0
110	1.02833	3.14651	0
111	4.11899	4.29137	1
112	3.14631	4.6287	1
113	1.80418	1.46429	0
114	2.5994	1.6996	0
115	3.16519	4.28628	1
116	2.00459	2.80665	0
117	4.73349	3.34389	1
118	2.90942	4.8445	1
119	1.80915	2.08437	0

Formato | Redimensionar | ☑ Color de fondo

Guardar y Cerrar | Cerrar

Aprendizaje supervisado

Python

```python
import numpy as np
import matplotlib.pyplot as plt

dataset = np.load(“desgrad_dataset.npy”)

#Grafica
plt.figure(1)
for i in range(len(dataset)):
    if(dataset[i,2]==1):
        plt.scatter(dataset[i,0],dataset[i,1],c='r')
    else:
        plt.scatter(dataset[i,0],dataset[i,1],c='b')

plt.xlabel("$x$")
plt.ylabel("$y$")
plt.grid(True)
```

Matlab/Octave

```matlab
load('desgrad_dataset.mat')

%Grafica
figure(1)
hold on
for i = 1:size(dataset,1)
    if(dataset(i,3)==1)
        scatter(dataset(i,1),dataset(i,2),'r')
    else
        scatter(dataset(i,1),dataset(i,2),'b')
    end
end

xlabel('x')
ylabel('y')
grid on
```

Aprendizaje supervisado: Función de costo

$$E = \frac{1}{2|P|} \sum_{k=0}^{\overbrace{|P|}^{\text{Número de patrones de entrenamiento}}-1} (t_k - y_k)^2$$

Target o clase a la pertenece el k-ésimo patrón

$P = \{p_0, p_1 \cdots, p_k\}$: Conjunto de *patrones de entrenamiento*
(vectores de características conocidos)

Aprendizaje supervisado: Función de costo

$$E_k' = \frac{1}{2}(t_k - y_k(\boldsymbol{w}, \boldsymbol{x}))^2$$

Para minimizar la función de costo debemos encontrar el valor de vector de pesos $\boldsymbol{w}$ que la vuelva lo más cercano a cero posible.

Aprendizaje supervisado: Regla Delta

Razón de aprendizaje

$$w_j^{nuevo} = w_j^{viejo} - \alpha \frac{\partial E(t_k, \boldsymbol{w}, \boldsymbol{x})}{\partial w_j}\bigg|_{\boldsymbol{w}^{viejo}, \boldsymbol{x}}$$

Nota. El número de pesos debe ser igual al número características de cada patrón. El índice *j* representa el peso asociado a cada característica. No confundir con el índice *k* que representa un elemento dado del conjunto de patrones.

Regla Delta: Derivada de la función de costo

$$E'_k = \frac{1}{2}(error)^2$$

$$\frac{\partial E'_k}{\partial w_j} = \big(t_k - f(\boldsymbol{w} \cdot \boldsymbol{x} + b)\big)\frac{d(error)}{df}$$

Regla Delta: Derivada de la función de costo

$$E'_k = \frac{1}{2}(t_k - f(net))^2$$

$$\frac{\partial E'_k}{\partial w_j} = \big(t_k - f(\boldsymbol{w} \cdot \boldsymbol{x} + b)\big)(-1)\frac{df}{dnet}$$

Regla Delta: Derivada de la función de costo

$$E'_k = \frac{1}{2}(t_k - f(\boldsymbol{w} \cdot \boldsymbol{x} + b))^2$$

$$\frac{\partial E'_k}{\partial w_j} = (t_k - f(\boldsymbol{w} \cdot \boldsymbol{x} + b))(-1)\left(\frac{df}{dnet}\right)\frac{d(net)}{\partial w_j}$$

Regla Delta: Derivada de la función de costo

$$E'_k = \frac{1}{2}(t_k - f(\boldsymbol{w} \cdot \boldsymbol{x} + b))^2$$

$$\boxed{\frac{\partial E'_k}{\partial w_j} = -(t_k - f(\boldsymbol{w} \cdot \boldsymbol{x} + b))\left(\frac{df}{dnet}\right)x_j}$$

Regla Delta: Derivada de la función de costo

$$f = logsig(net) \rightarrow \frac{df}{dnet} = \frac{e^{-net}}{(e^{-net}+1)^2}$$

$$\boxed{\frac{\partial E_k'}{\partial w_j} = -\left(t_k - f(\boldsymbol{w}\cdot\boldsymbol{x}+b)\right)\left(\frac{df}{dnet}\right)x_j}$$

Practica #2 : Entrenamiento por Regla Delta

Función de activación

$$f(net) = \frac{1}{1 + e^{-net}}$$

Derivada de la función de activación

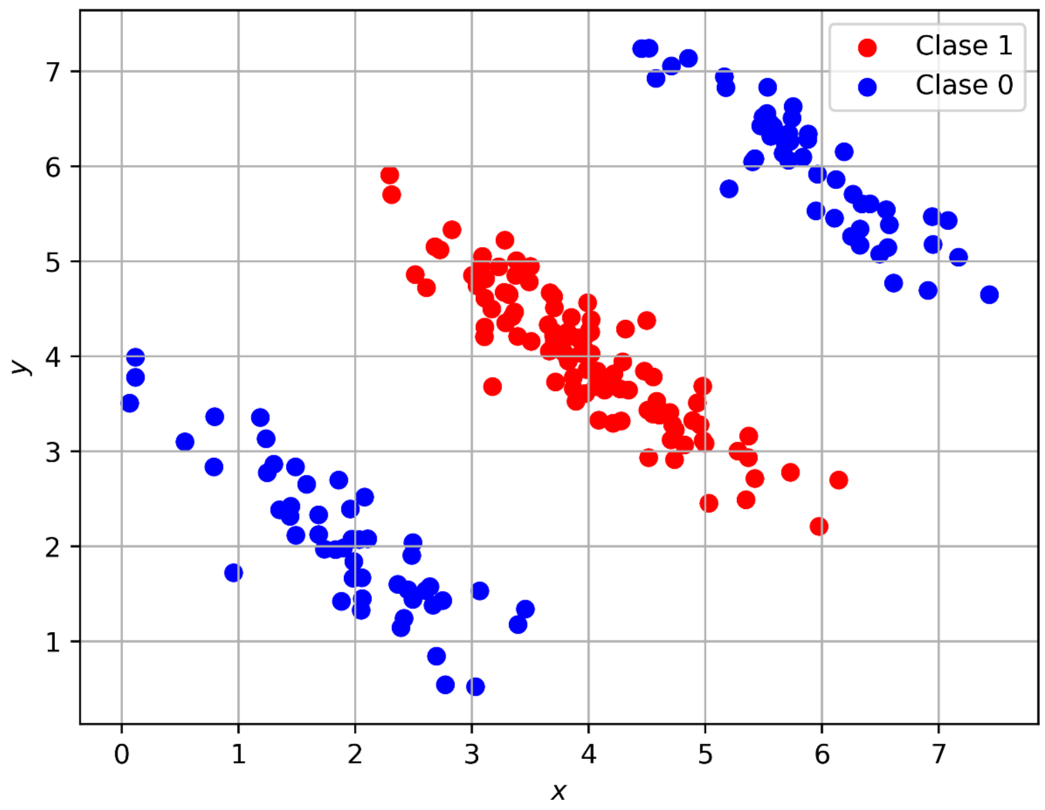
$$\frac{df(net)}{dnet} = f(net)(1 - f(net))$$

Regla Delta

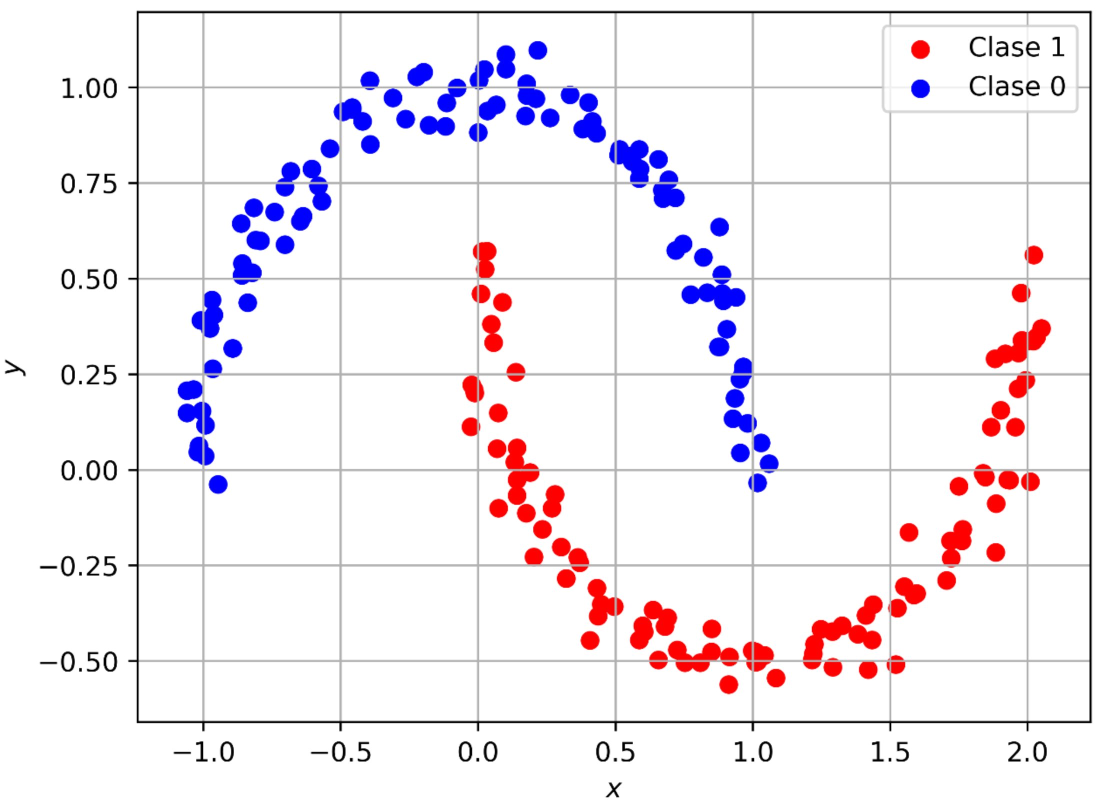
$$w_j^{nuevo} = w_j^{viejo} + \alpha\left(t_k - f(net)\right)\left(\frac{df(net)}{dnet}\right)x_j$$

¿Puede esta neurona clasificar los siguientes datasets?

dataset_BP_blobs.npy/mat/csv



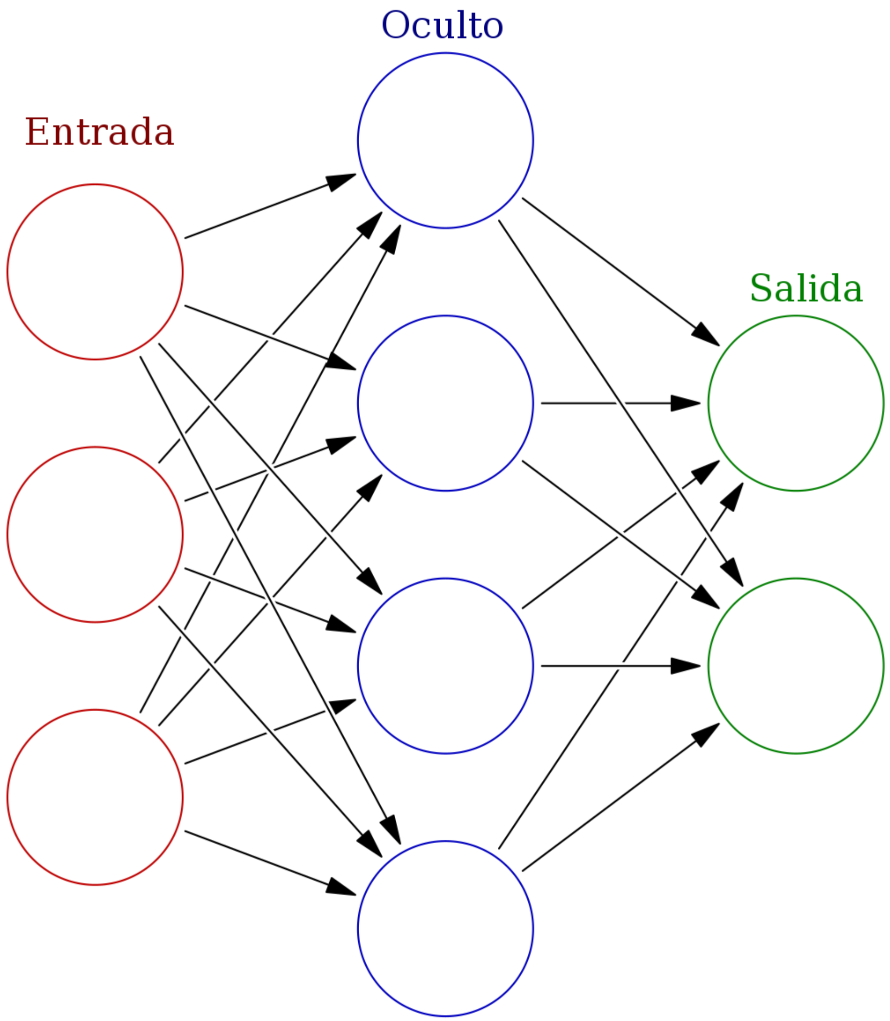
dataset_BP_moons.npy/mat/csv



Redes neuronales artificiales

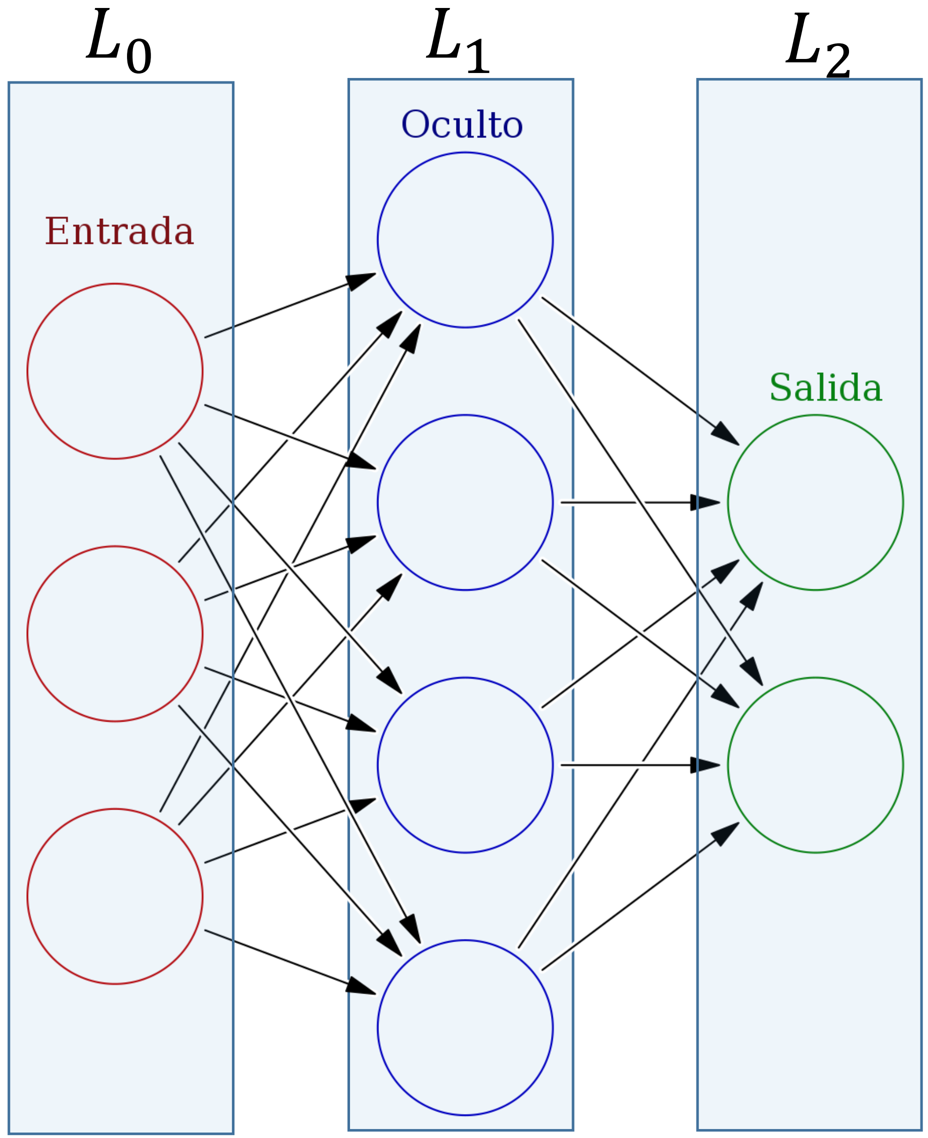
Conectando todo.

La red de neuronas



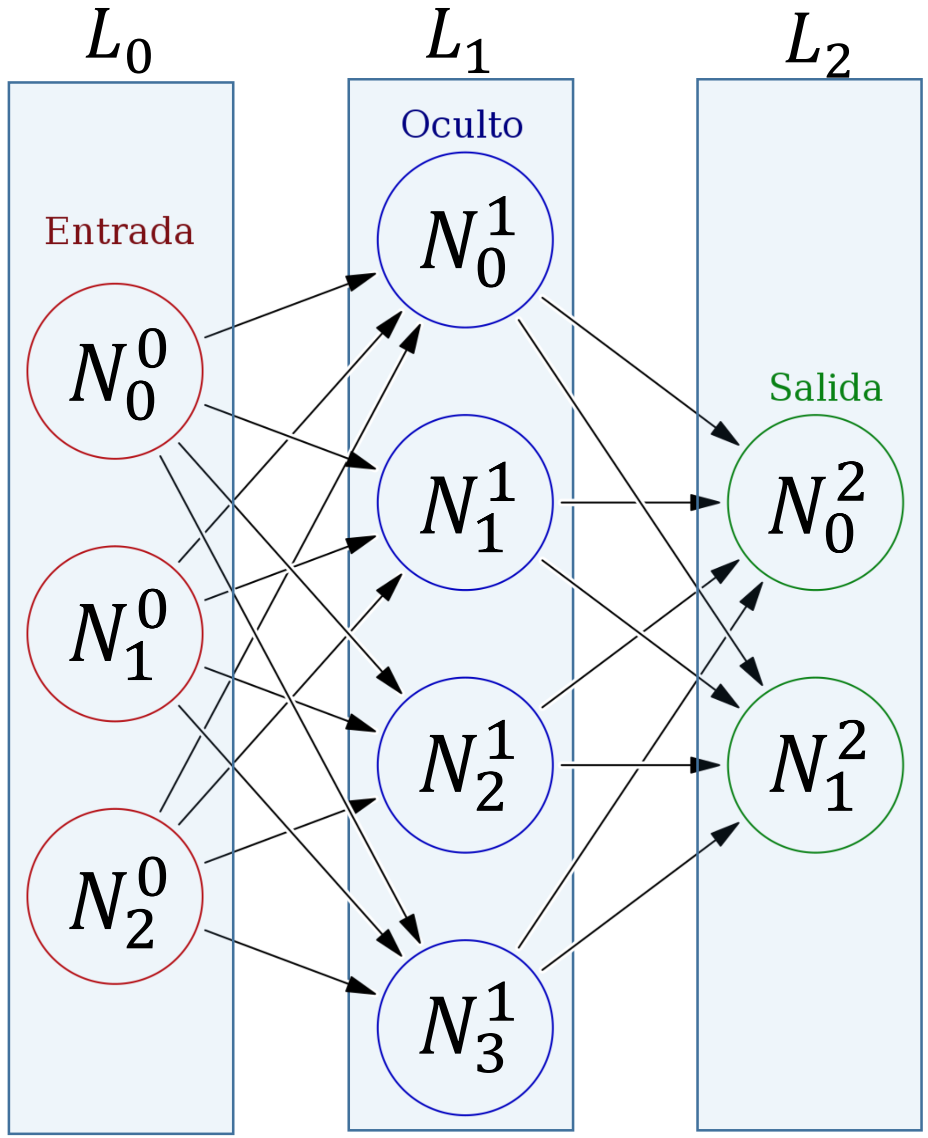
Habiendo entendido el funcionamiento interno de las neuronas artificiales, podemos abstraer la red para representarle solo a la través de **nodos** (neuronas), **conexiones** y **capas**.

La red de neuronas



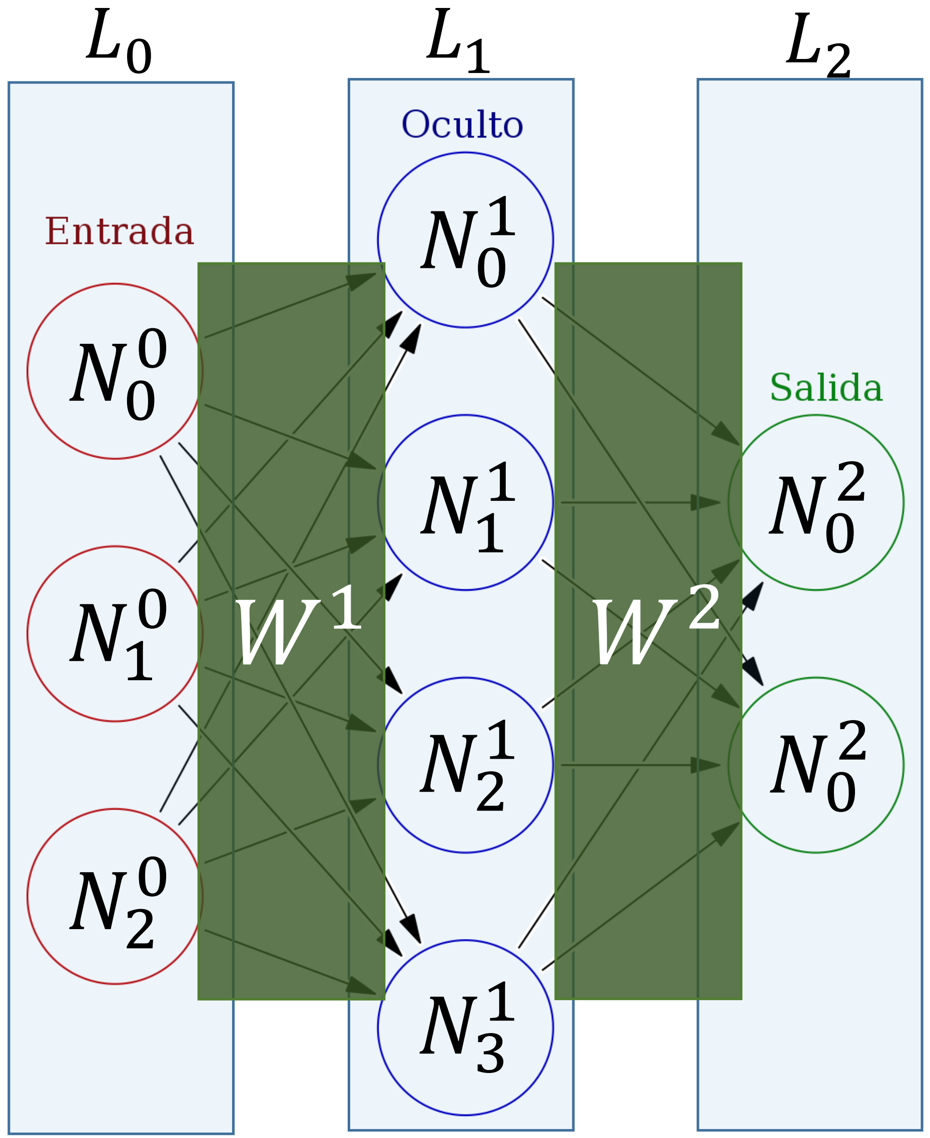
Se denota con L_l (*layer*) al conjunto de neuronas que forma la capa *l*-ésima

La red de neuronas



Las neuronas se denotan por N_j^L donde el superíndice L no denota una potencia sino la capa a la cual la neurona pertenece.

La red de neuronas



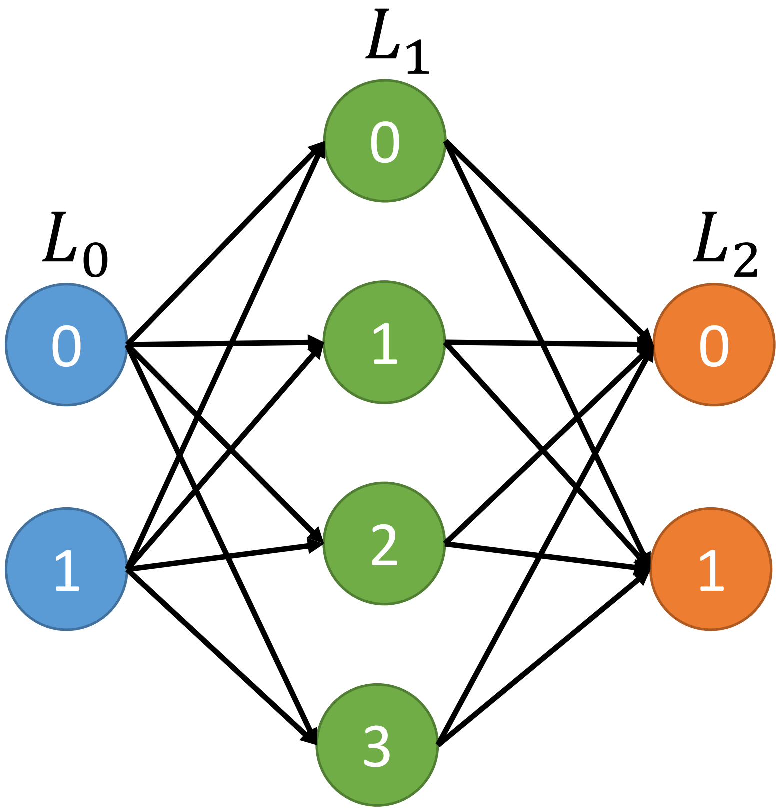
Cuando se tienen múltiples neuronas, los pesos ya no se representan como vectores sino como matrices W^L para cada capa que se construyen de la siguiente manera:

Número de neurona de destino (capa L)

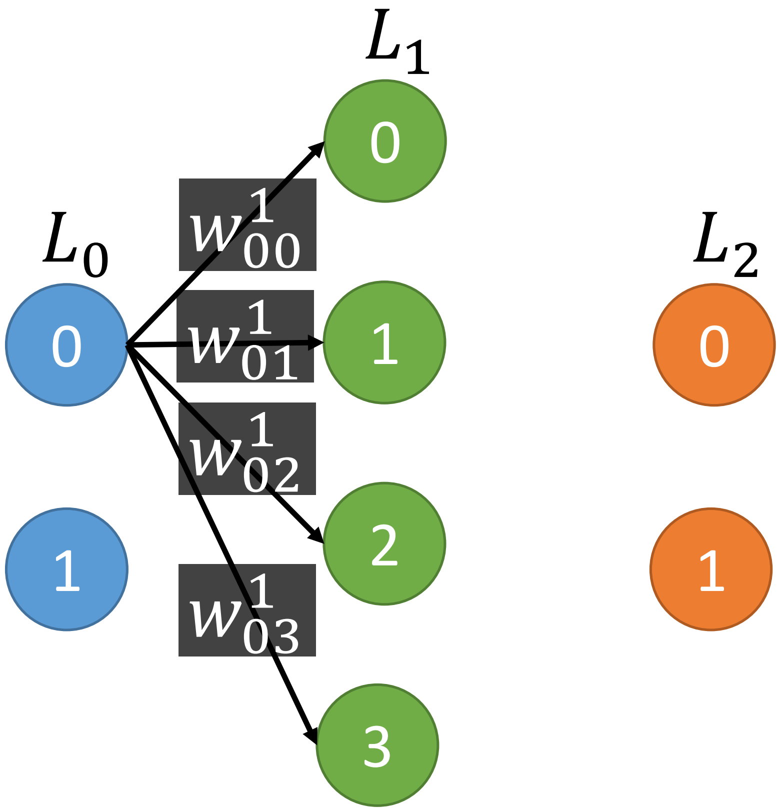
$$W^L\colon\ w_{ij}^L$$

Número de neurona de origen (capa $L-1$)

Ejemplo: Construcción de matrices de pesos

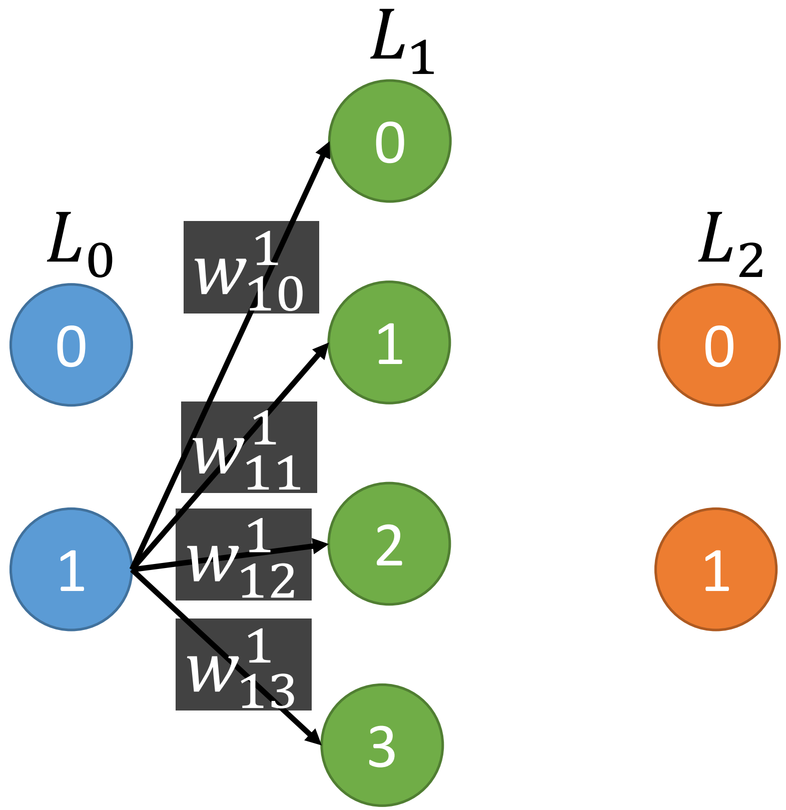


Ejemplo: Construcción de matrices de pesos



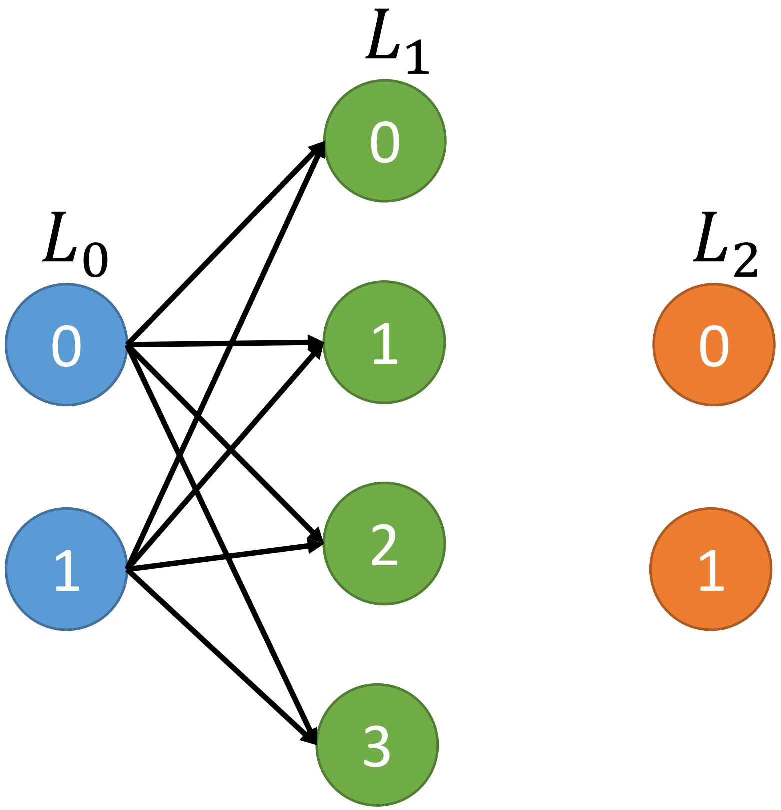
$$W^1 = \begin{bmatrix} w_{00}^1 & w_{01}^1 & w_{02}^1 & w_{03}^1 \\ & & & \end{bmatrix}$$

Ejemplo: Construcción de matrices de pesos



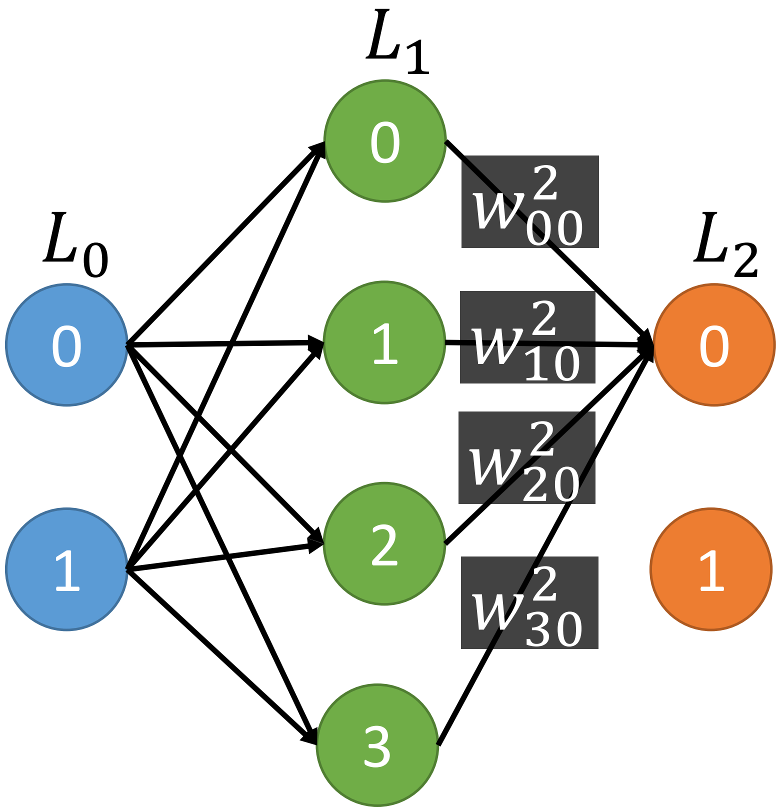
$$W^1 = \begin{bmatrix} w_{00}^1 & w_{01}^1 & w_{02}^1 & w_{03}^1 \\ w_{10}^1 & w_{11}^1 & w_{12}^1 & w_{13}^1 \end{bmatrix}$$

Ejemplo: Construcción de matrices de pesos



$$
\begin{array}{c}
\begin{array}{cccc} N_0^1 & N_1^1 & N_2^1 & N_3^1 \end{array} \\
W^1 = \begin{bmatrix} w_{00}^1 & w_{01}^1 & w_{02}^1 & w_{03}^1 \\ w_{10}^1 & w_{11}^1 & w_{12}^1 & w_{13}^1 \end{bmatrix}
\end{array}
$$

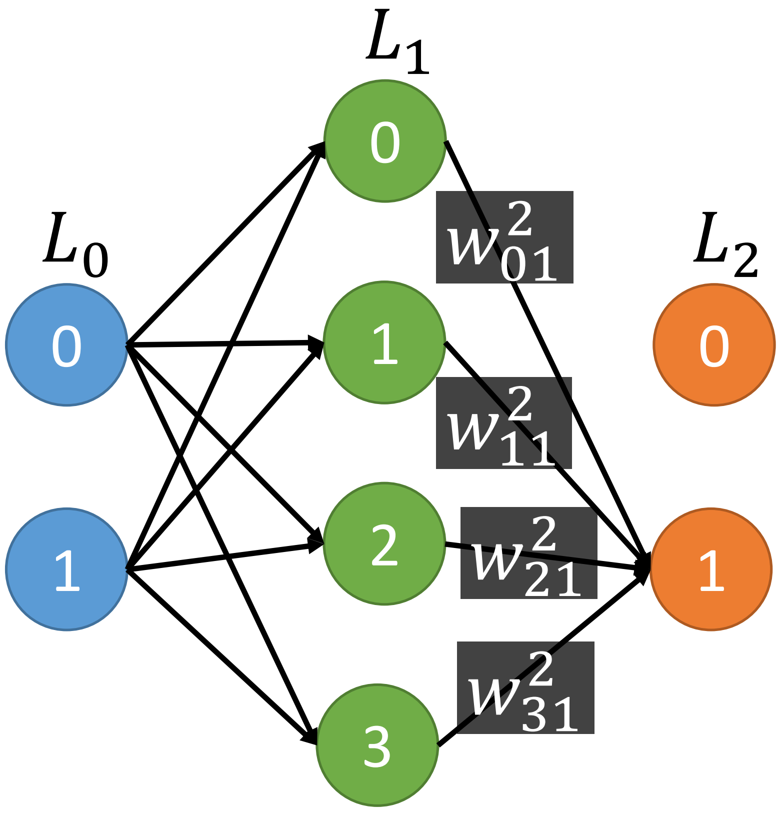
Ejemplo: Construcción de matrices de pesos



$$
\begin{array}{cccccc}
 & & N_0^1 & N_1^1 & N_2^1 & N_3^1 \\
W^1 & = & \multicolumn{4}{c}{\begin{bmatrix} w_{00}^1 & w_{01}^1 & w_{02}^1 & w_{03}^1 \\ w_{10}^1 & w_{11}^1 & w_{12}^1 & w_{13}^1 \end{bmatrix}}
\end{array}
$$

$$
W^2 = \begin{bmatrix} w_{00}^2 \\ w_{10}^2 \\ w_{20}^2 \\ w_{30}^2 \end{bmatrix}
$$

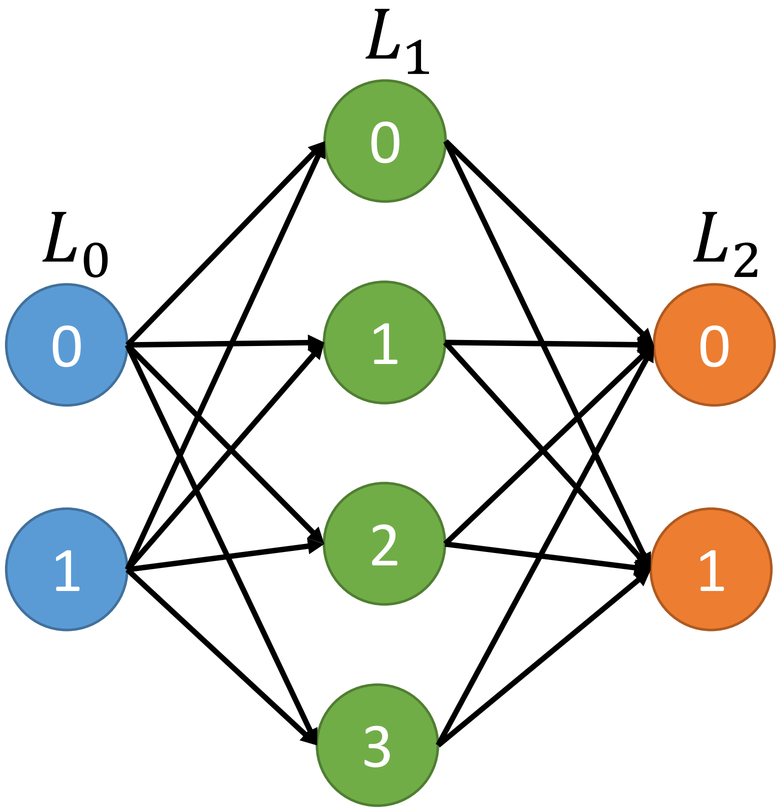
Ejemplo: Construcción de matrices de pesos



$$
\begin{array}{cc}
 & \begin{array}{cccc} N_0^1 & N_1^1 & N_2^1 & N_3^1 \end{array} \\
W^1 = & \begin{bmatrix} w_{00}^1 & w_{01}^1 & w_{02}^1 & w_{03}^1 \\ w_{10}^1 & w_{11}^1 & w_{12}^1 & w_{13}^1 \end{bmatrix}
\end{array}
$$

$$
W^2 = \begin{bmatrix} w_{00}^2 & w_{01}^2 \\ w_{10}^2 & w_{11}^2 \\ w_{20}^2 & w_{21}^2 \\ w_{30}^2 & w_{31}^2 \end{bmatrix}
$$

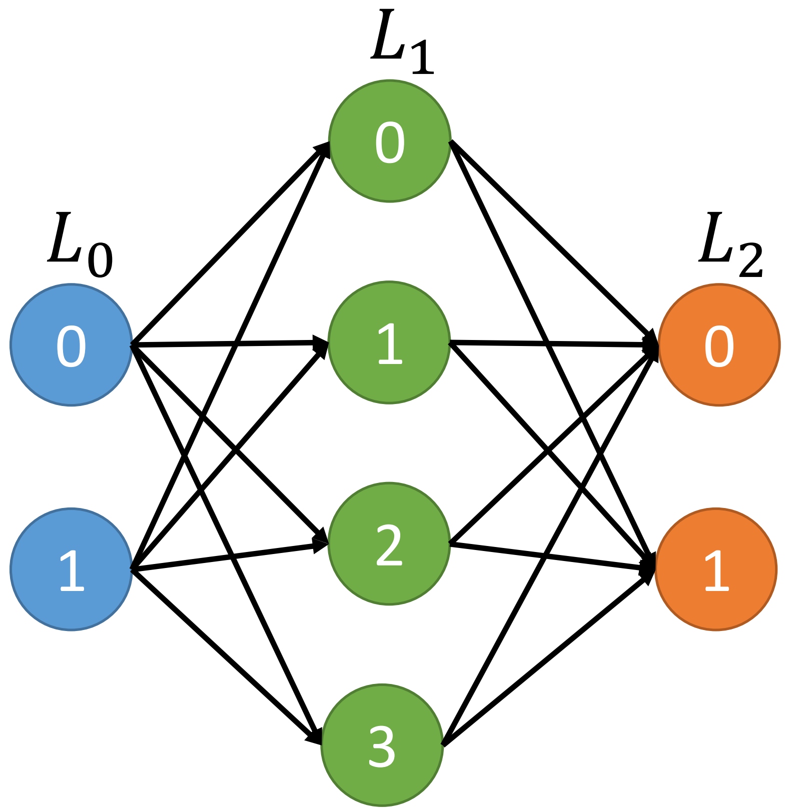
Ejemplo: Construcción de matrices de pesos



$$W^1 = \begin{array}{c} \begin{array}{cccc} N_0^1 & N_1^1 & N_2^1 & N_3^1 \end{array} \\ \begin{bmatrix} w_{00}^1 & w_{01}^1 & w_{02}^1 & w_{03}^1 \\ w_{10}^1 & w_{11}^1 & w_{12}^1 & w_{13}^1 \end{bmatrix} \end{array}$$

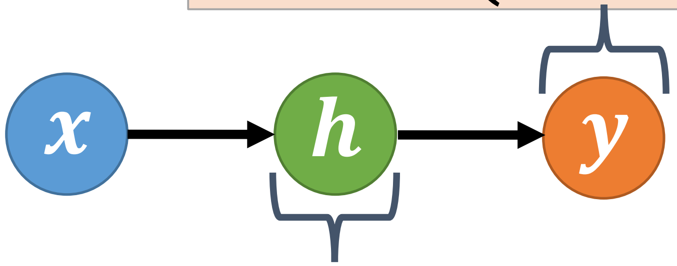
$$W^2 = \begin{array}{c} \begin{array}{cc} N_0^2 & N_1^2 \end{array} \\ \begin{bmatrix} w_{00}^2 & w_{01}^2 \\ w_{10}^2 & w_{11}^2 \\ w_{20}^2 & w_{21}^2 \\ w_{30}^2 & w_{31}^2 \end{bmatrix} \end{array}$$

Notación compacta de una red neuronal



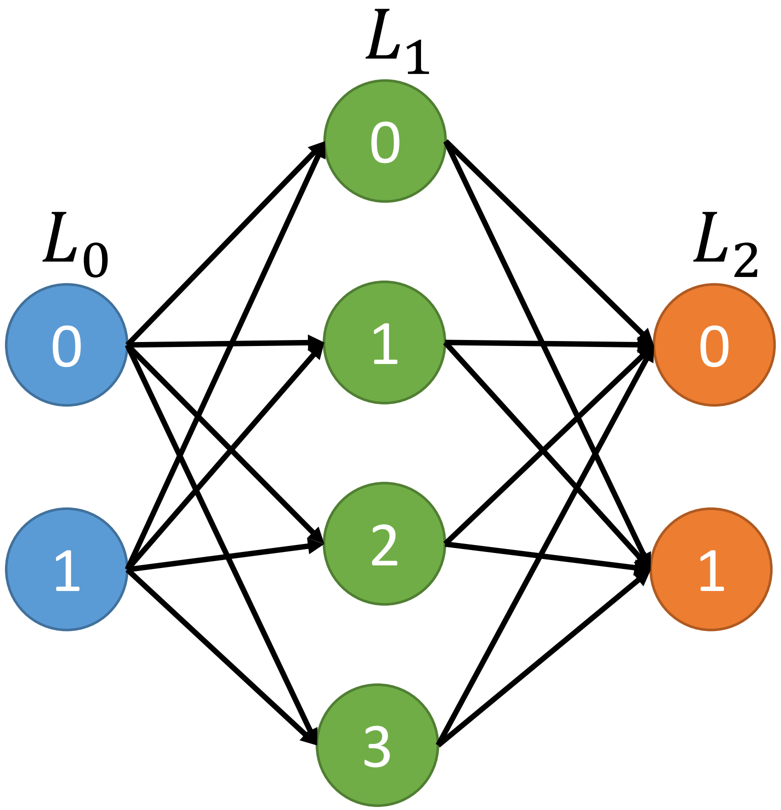
$=$

$$\boldsymbol{y} = f\left(\boldsymbol{W}^{2^T} \cdot \boldsymbol{h} + \boldsymbol{b}^2\right)$$



$$\boldsymbol{h} = f\left(\boldsymbol{W}^{1^T} \cdot \boldsymbol{x} + \boldsymbol{b}^1\right)$$

Practica #3 : Red neurona artificial



$$=$$

$$\boldsymbol{x} \to \boldsymbol{h} \to \boldsymbol{y}$$

$$\boxed{\boldsymbol{y} = f\left(\boldsymbol{W}^{2^T} \cdot \boldsymbol{h} + \boldsymbol{b}^2\right)}$$

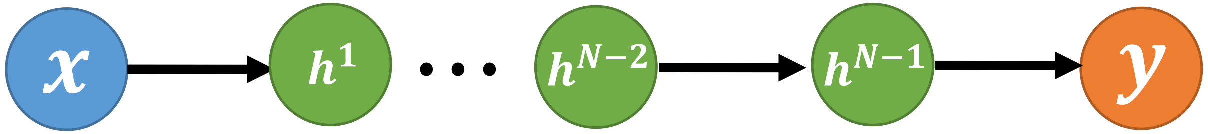
$$\boxed{\boldsymbol{h} = f(\boldsymbol{W}^{1^T} \cdot \boldsymbol{x} + \boldsymbol{b}^1)}$$

$$\begin{array}{c} N_0^1 \\ N_1^1 \\ \\ N_2^1 \\ N_3^1 \end{array} \quad \boldsymbol{net}^1 = \begin{bmatrix} w_{00}^1 & w_{10}^1 \\ w_{01}^1 & w_{11}^1 \\ w_{02}^1 & w_{12}^1 \\ w_{03}^1 & w_{13}^1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \begin{bmatrix} b_0^1 \\ b_1^1 \\ b_2^1 \\ b_3^1 \end{bmatrix} = \begin{bmatrix} w_{00}^1 x_0 + w_{10}^1 x_0 + b_0^1 \\ w_{01}^1 x_0 + w_{11}^1 x_0 + b_1^1 \\ w_{02}^1 x_0 + w_{12}^1 x_0 + b_2^1 \\ w_{03}^1 x_0 + w_{13}^1 x_0 + b_3^1 \end{bmatrix}$$

Entrenamiento de redes neuronales

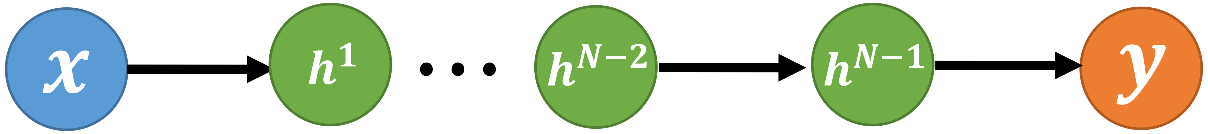
Entrenamiento multicapa: **Backpropagation**.

Regla delta generalizada (backpropagation)



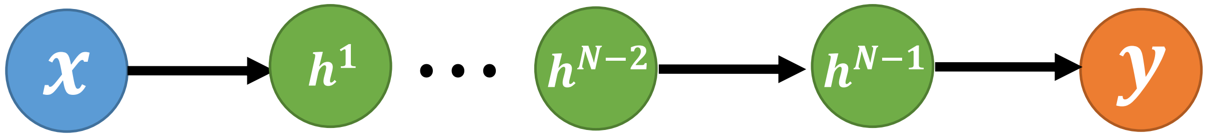
$$\frac{\partial E}{\partial w_{ji}} = S_j^L \frac{\partial net_i^L}{\partial w_{ji}}$$

Regla delta generalizada (backpropagation)



$$\frac{\partial E}{\partial w_{ji}} = S_j^L h_i^{L-1}$$

Regla delta generalizada (backpropagation)



$$\frac{\partial E}{\partial w_{ji}} = S_j^L h_i^{L-1}$$

$$S_j^L = \begin{cases} -((t_k)_i - y_i)\dfrac{df(net_i^L)}{dnet_i^L} \quad , & L = salida \\ (W^{L+1} \cdot S^{L+1})_j \dfrac{df(net_i^L)}{dnet_i^L} \quad , & L \neq salida \end{cases}$$

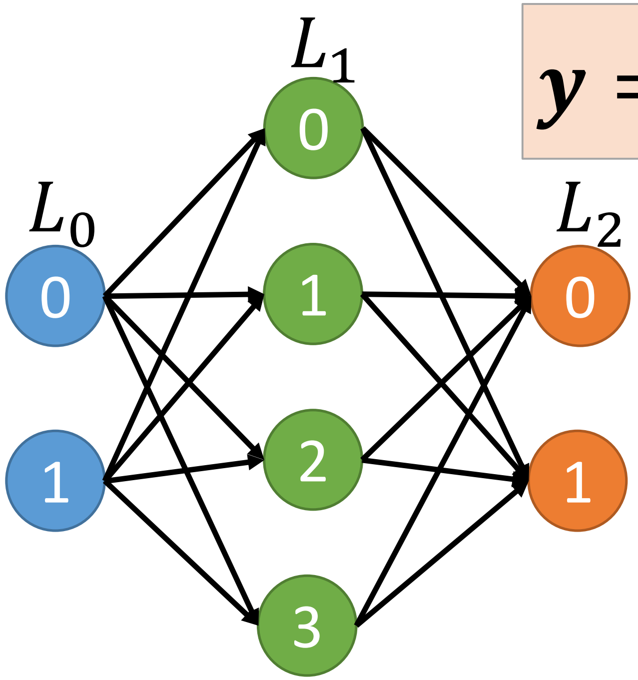
Regla delta generalizada (backpropagation)

$$S_j^L = \begin{cases} -((t_k)_j - y_j)\dfrac{df(net_j^L)}{dnet_j^L} & , \quad L = salida \\ (W^{L+1} \cdot S^{L+1})_j \dfrac{df(net_j^L)}{dnet_j^L} & , \quad L \neq salida \end{cases}$$

Regla Delta generalizada

$$\boxed{w_{ij}^{L(nuevo)} = w_{ij}^{L(viejo)} - \alpha S_j^L h_i^{L-1}}$$

Practica #3 : Red de 3 capas [2|4|2] entrenada por backpropagation



$$\boldsymbol{y} = f\left(\boldsymbol{W}^{2^T} \cdot \boldsymbol{h} + \boldsymbol{b}^2\right)$$

$$\boldsymbol{h} = f(\boldsymbol{W}^{1^T} \cdot \boldsymbol{x} + \boldsymbol{b}^1)$$

$$S_j^L = \begin{cases} -((t_k)_j - y_j)\dfrac{df(net_j^2)}{dnet_j^2} & , \quad L = 2 \\ (W^2 \cdot S^2)_j \dfrac{df(net_j^1)}{dnet_j^1} & , \quad L = 1 \end{cases}$$

Regla Delta generalizada

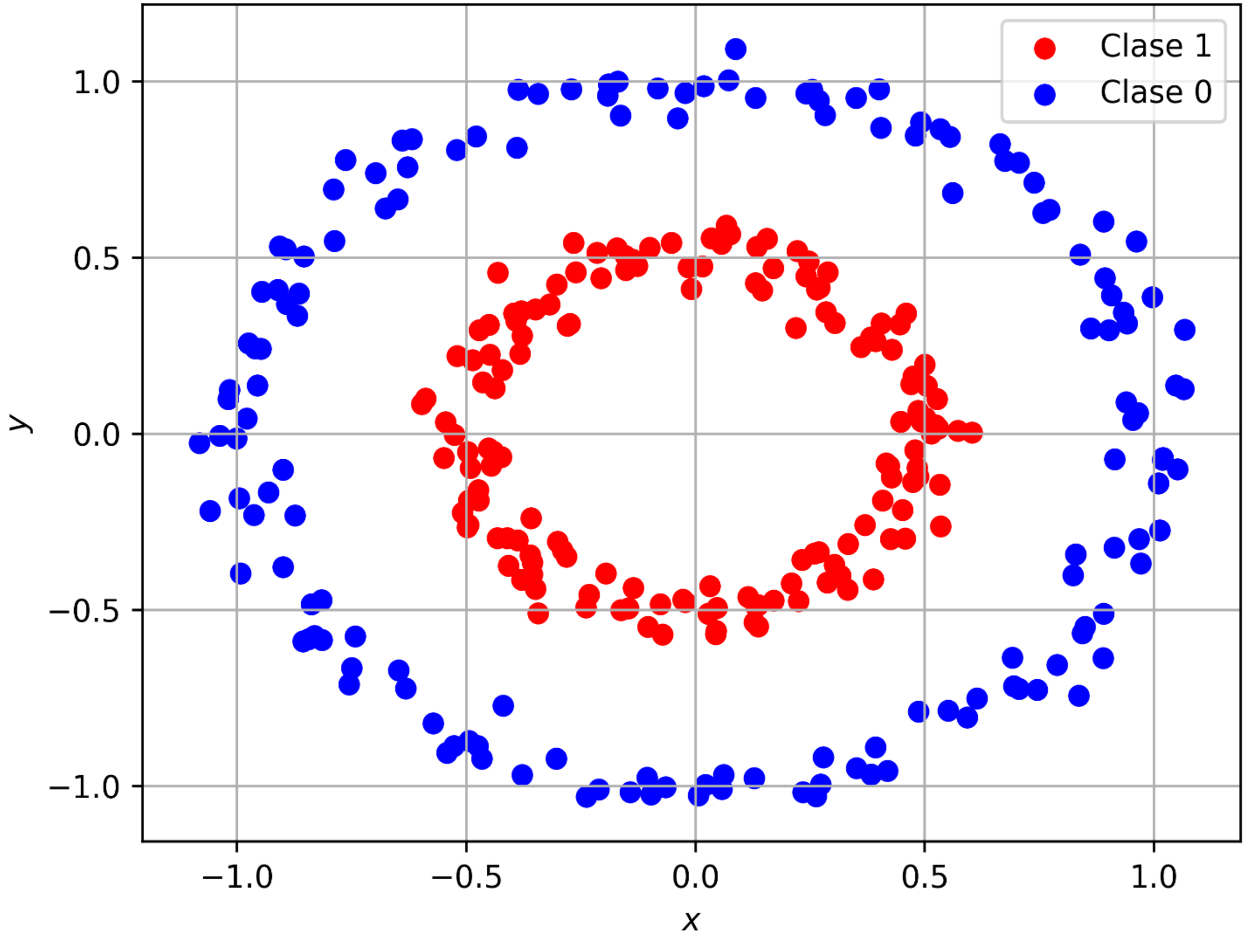
$$w_{ij}^{2(nuevo)} = w_{ij}^{2(viejo)} - \alpha S_j^2 h_i$$

$$w_{ij}^{1(nuevo)} = w_{ij}^{1(viejo)} - \alpha S_j^1 x_i$$

¿Puede esta red clasificar el siguiente dataset?

Spoiler ahead: No. Se requiere una red de 4 capas.

dataset_BP_circles.npy/mat/csv



Contacto

r.escurib@gmail.com

r.e.escobar.3