3ª Emisión

DIPLOMADO
Inteligencia Artificial Aplicada

Módulo 11: Introducción a las redes neuronales

1. Neuronas artificiales

Instructor: Blanca Vázquez

Objetivo de la sesión

- Aprender los conceptos básicos de las redes neuronales artificiales e identificar cómo se entrenan usando el algoritmo del descenso del gradiente.

Contenido

Generación automática de composiciones musicales

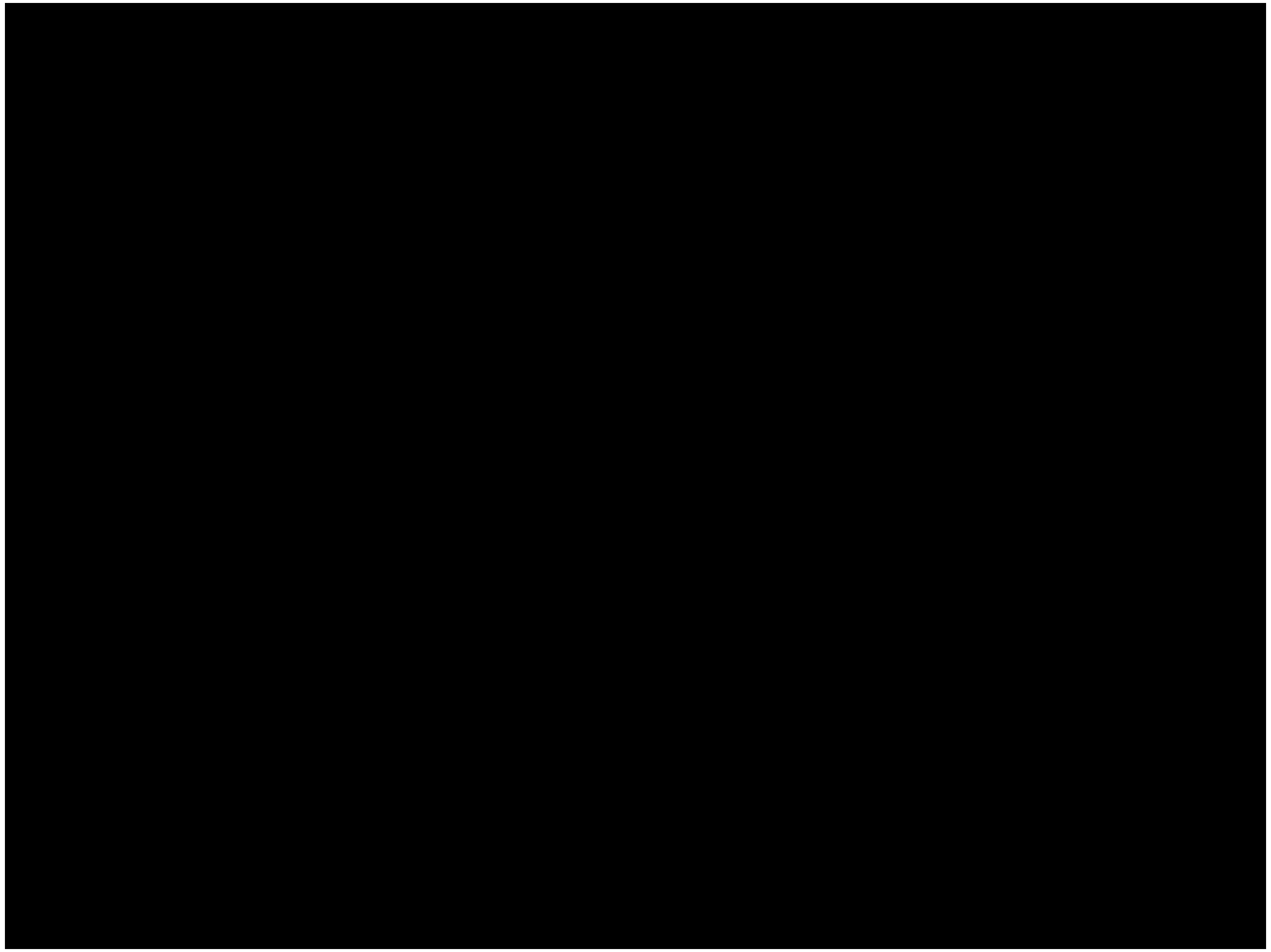
Composición en el estilo de Chopin iniciando con la canción de Adele *Someone Like You*

Composición en el estilo *Country* iniciando con la canción de Beethoven

Imagen y audio tomado de MuseNet, OpenIA, 2024.

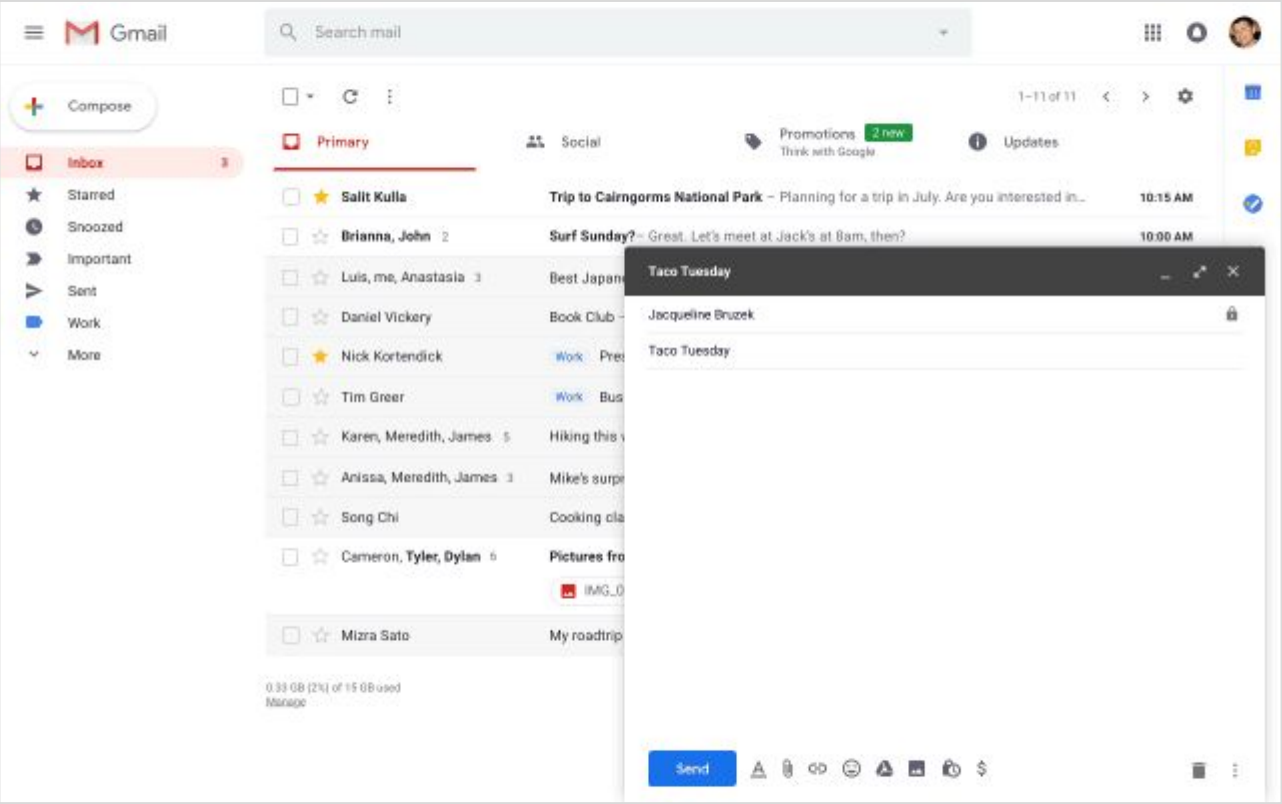
Generación de video a partir de texto

Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.

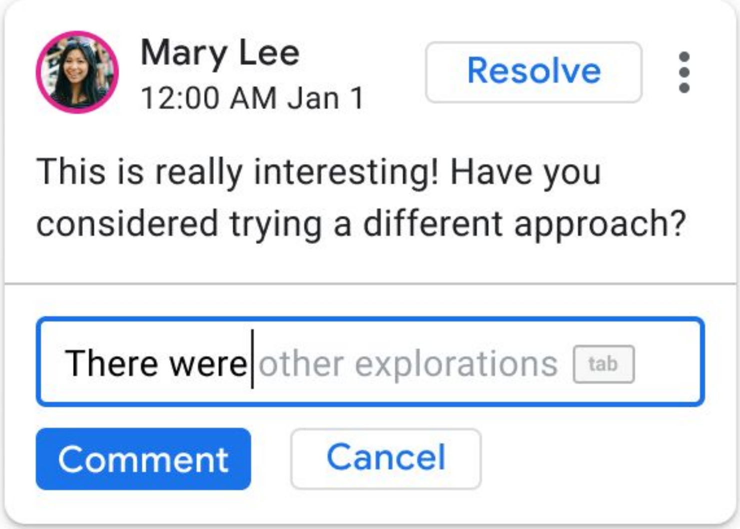


Texto y video tomado de Sora, OpenIA, 2024.

Generación de texto automático



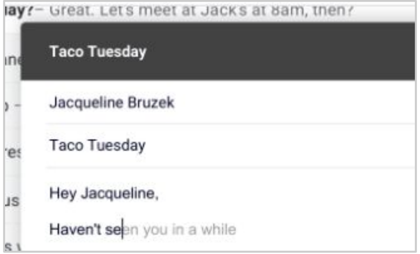
Video tomado de Smart Compose, Gmail, 2024.



Aplicaciones de la inteligencia artificial (IA)

Generating musical compositions

Smart compose



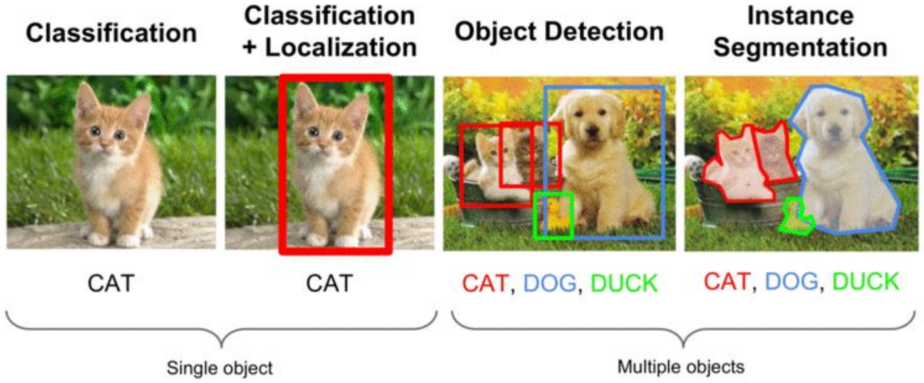
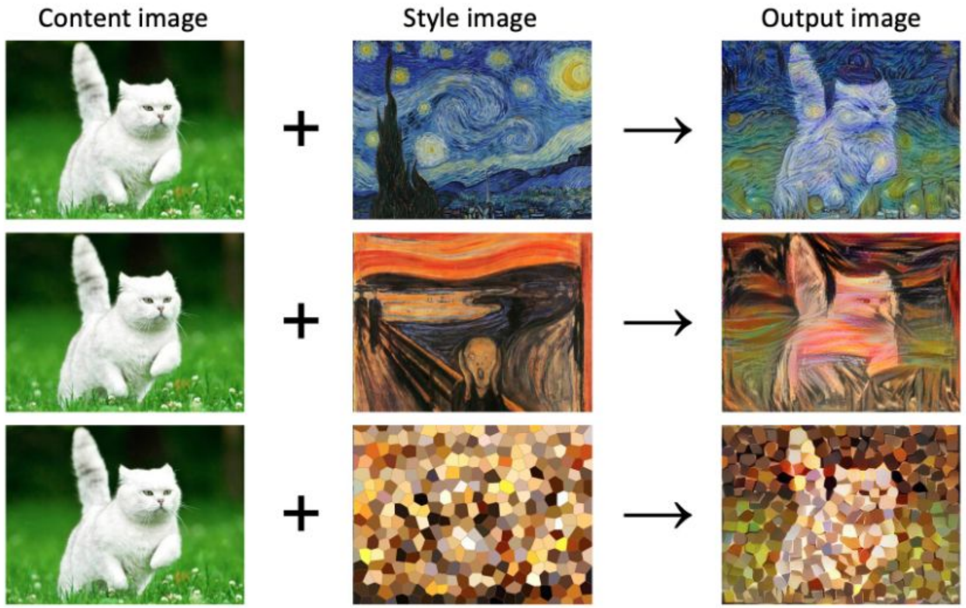


Imagen tomada de Adhane, 2019.

Transfer style



Gatys, Ecker, and Bethge, A Neural Algorithm of Artistic Style, 2015.
Imagen tomada de Bertens, 2019.

Creating video from text

Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.

¿Qué son las redes neuronales?

Una red neuronal es un modelo matemático **inspirado** en el comportamiento biológico de las neuronas y en la estructura del cerebro.

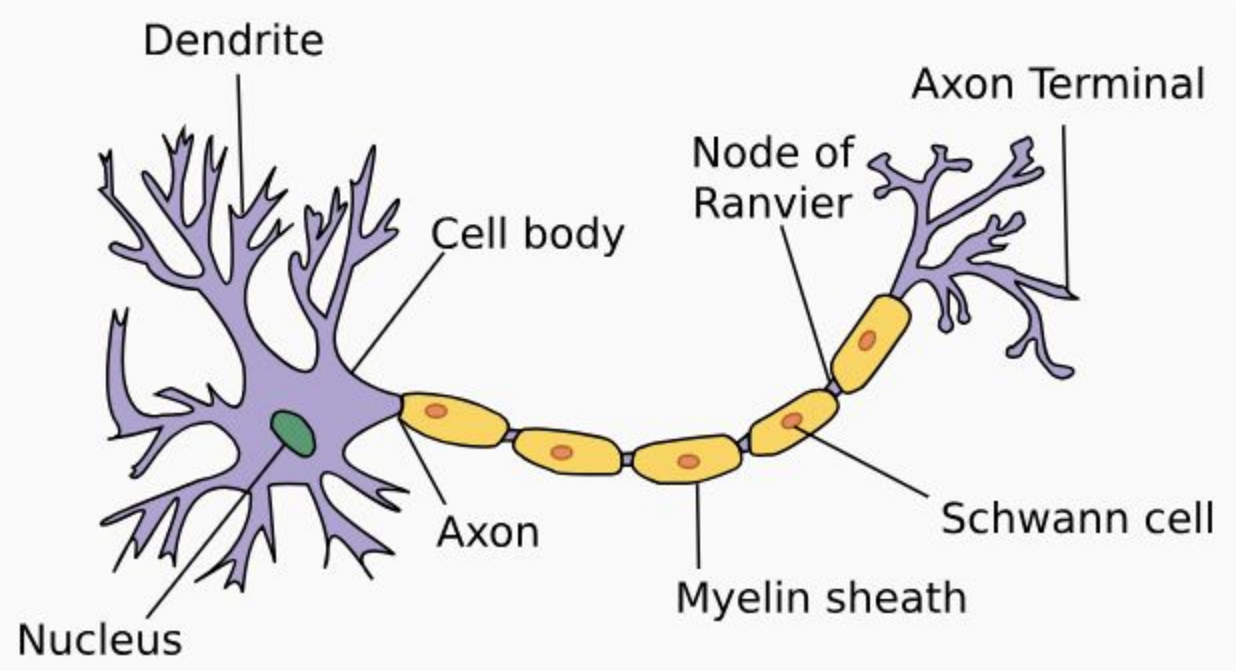


Imagen tomada de Anatomy and Physiology" by the US National Cancer Institute's Surveillance, Epidemiology and End Results (SEER) Program

Comunicación entre neuronas

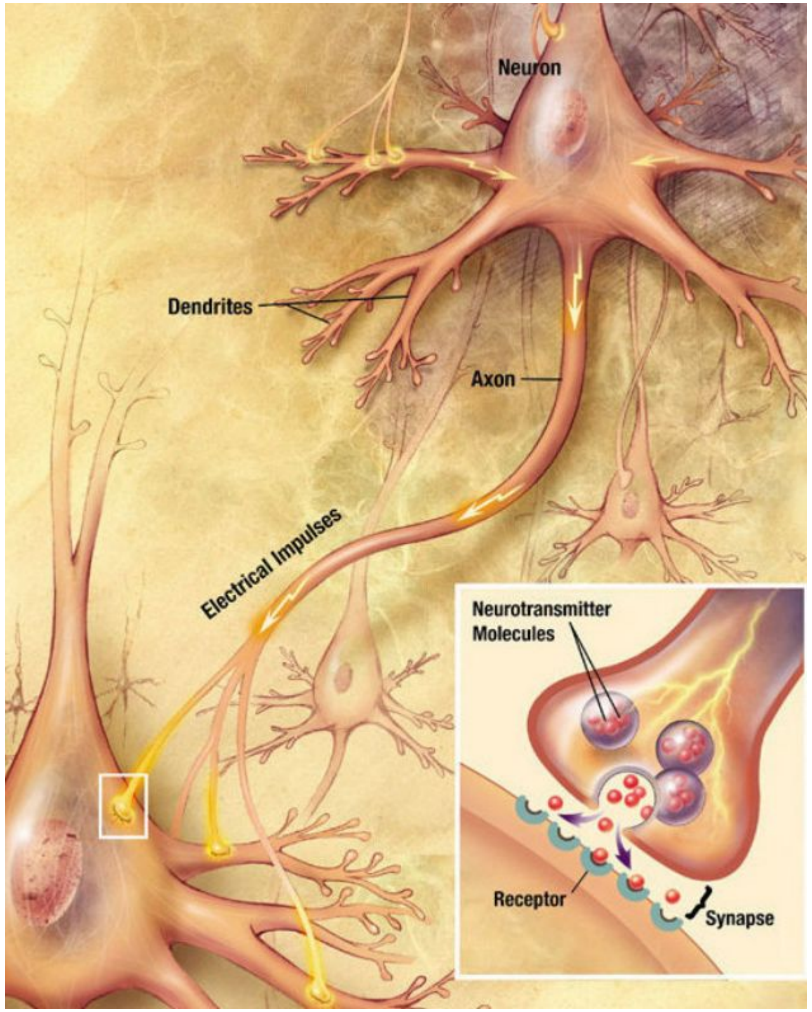
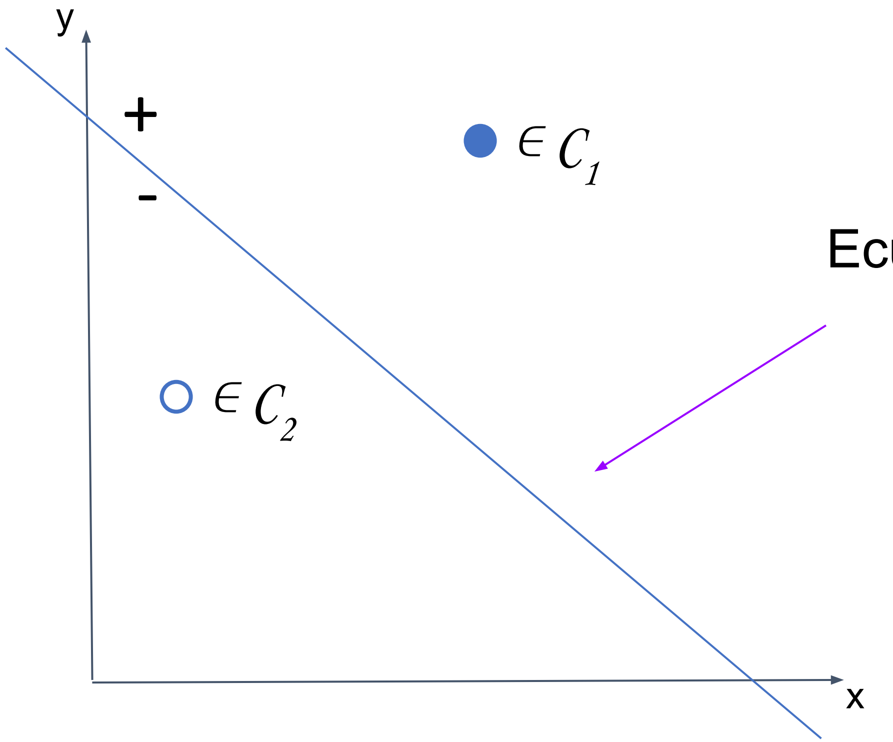


Imagen tomada de Wikipedia, 2020.

El perceptrón



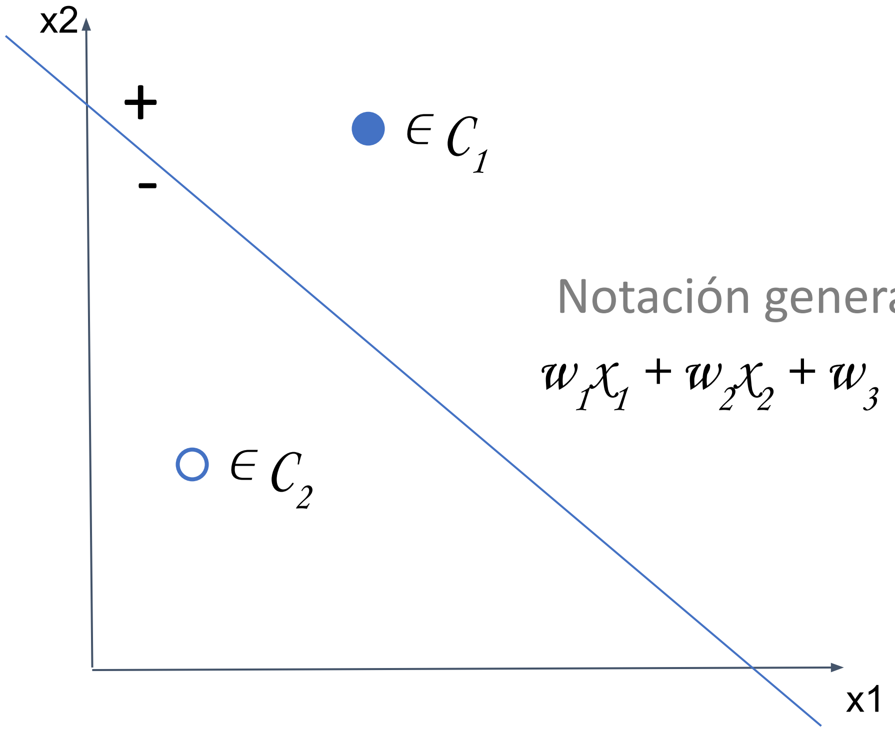
Ecuación de la recta

$$y = ax + b, \text{ o}$$

$$y - ax - b = 0$$

Es la unidad básica de una red neuronal

Perceptrón: modelo lineal binario



Notación general

$$w_1x_1 + w_2x_2 + w_3 = 0$$

Dado un punto arbitrario (x_1,x_2), va pertenecer al lado positivo de la recta cuando:

$$w_1x_1 + w_2x_2 + w_3 > 0$$

Hiperplanos

Para un punto con n dimensiones

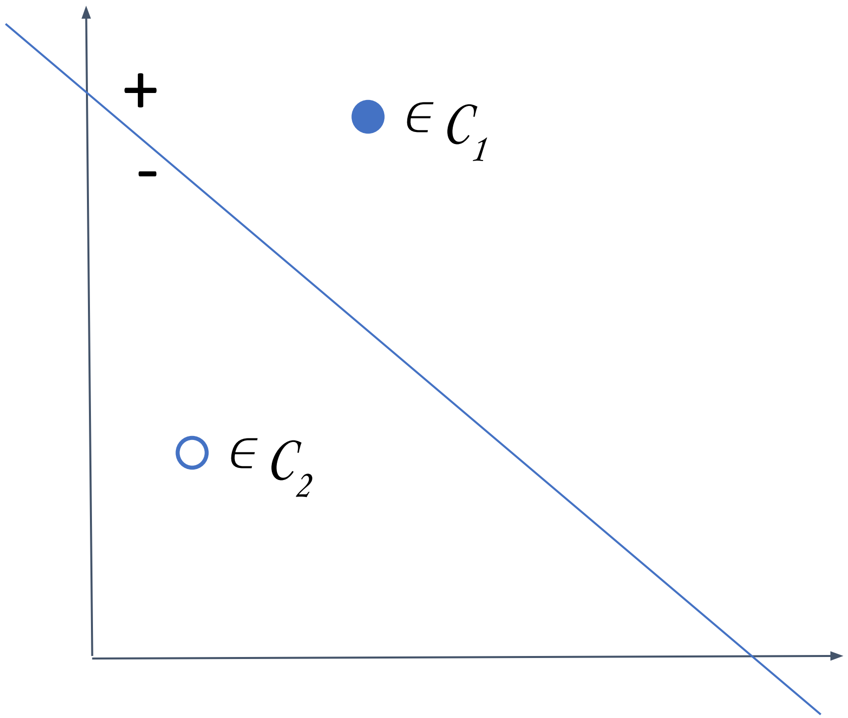
$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1} = 0$$

Esta ecuación se puede expresar en forma de suma:

$$\sum_{i=1}^{n} w_ix_i + w_{n+1} = 0$$

O en forma vectorial como:

$$\boldsymbol{w}^T\mathbf{x} + w_{n+1} = 0$$



Hiperplanos

Para un punto con n dimensiones

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1} = 0$$

Esta ecuación se puede expresar en forma de suma:

$$\sum_{i=1}^{n} w_ix_i + w_{n+1} = 0$$

O en forma vectorial como:

$$\mathbf{w}^T\mathbf{x} + w_{n+1} = 0$$

Dónde:

- *w* y *x* son vectores columnas n-dimensionales
- $w^T x$ es un producto punto de los 2 vectores
- *w* es el vector de pesos
- w_{n+1} es el sesgo

Problema de separación de clases

De forma general, dado cualquier vector *x* buscamos encontrar un conjunto de pesos dónde se cumpla:

$$\boldsymbol{w}^T\mathbf{x} + w_{n+1} = \begin{cases} > 0 & \text{if } \mathbf{x} \in c_1 \\ < 0 & \text{if } \mathbf{x} \in c_2 \end{cases}$$

Objetivo: encontrar una línea que separe dos clases de patrones *linealmente separables*

Problema de separación de clases

De forma general, dado cualquier vector *x* buscamos encontrar un conjunto de pesos dónde se cumpla:

$$\boldsymbol{w}^T\mathbf{x} + w_{n+1} = \begin{cases} > 0 & \text{if } \mathbf{x} \in c_1 \\ < 0 & \text{if } \mathbf{x} \in c_2 \end{cases}$$

A través de un proceso iterativo:

- Iniciamos con pesos y sesgos arbitrarios.
- Después de *n* iteraciones, se observará una convergencia (si las clases son linealmente separables).

Objetivo: encontrar una línea que separe dos clases de patrones *linealmente separables*

Algoritmo del perceptrón

Algorithm Perceptron learning algorithm

Input:

A set of training examples $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$
Learning rate $0 < \alpha < 1$
Number of epochs $epochs$

Initialize the weight vector $\mathbf{w}$ with random values
Initialize the bias $b \leftarrow 0$
for $i \leftarrow 1$ to $epochs$ **do**
    $err \leftarrow 0$ ▷ The number of misclassifications
    for each training example $(\mathbf{x}_i, y_i) \in D$ **do**
        $z_i \leftarrow \mathbf{w}^T\mathbf{x}_i + b$
        $o_i \leftarrow \begin{cases} 1 & z_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$ Compute the prediction
        if $y_i \neq o_i$ **then** Compute the error
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha(y_i - o_i)\mathbf{x}_i$ Update the weights
            $b \leftarrow b + \alpha(y_i - o_i)$
            $err \leftarrow err + 1$
    if $err = 0$ **then break**

Imagen tomada de Towards Data Science, 2023.

Esquema de un perceptrón

$$\boldsymbol{w}^T\mathbf{x} + w_{n+1} = \begin{cases} > 0 & \text{if } \mathbf{x} \in c_1 \\ < 0 & \text{if } \mathbf{x} \in c_2 \end{cases}$$

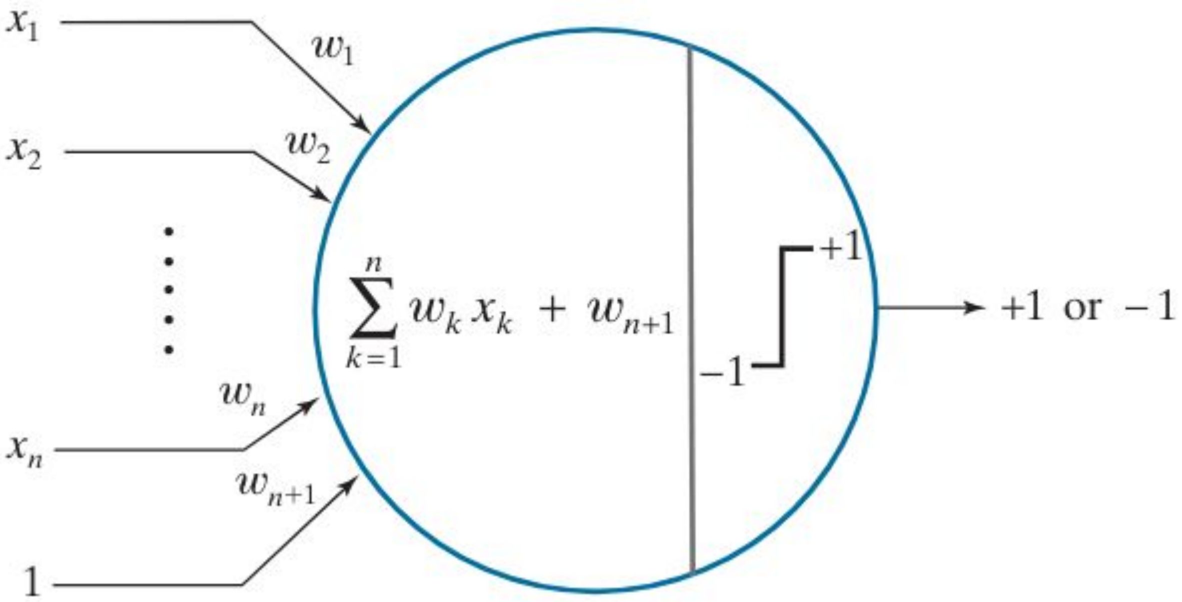


Imagen tomada de González, Woods, Digital Image Processing, 2018.

Esquema de un perceptrón

$$\boldsymbol{w}^T\mathbf{x} + w_{n+1} = \begin{cases} > 0 & \text{if } \mathbf{x} \in c_1 \\ < 0 & \text{if } \mathbf{x} \in c_2 \end{cases}$$

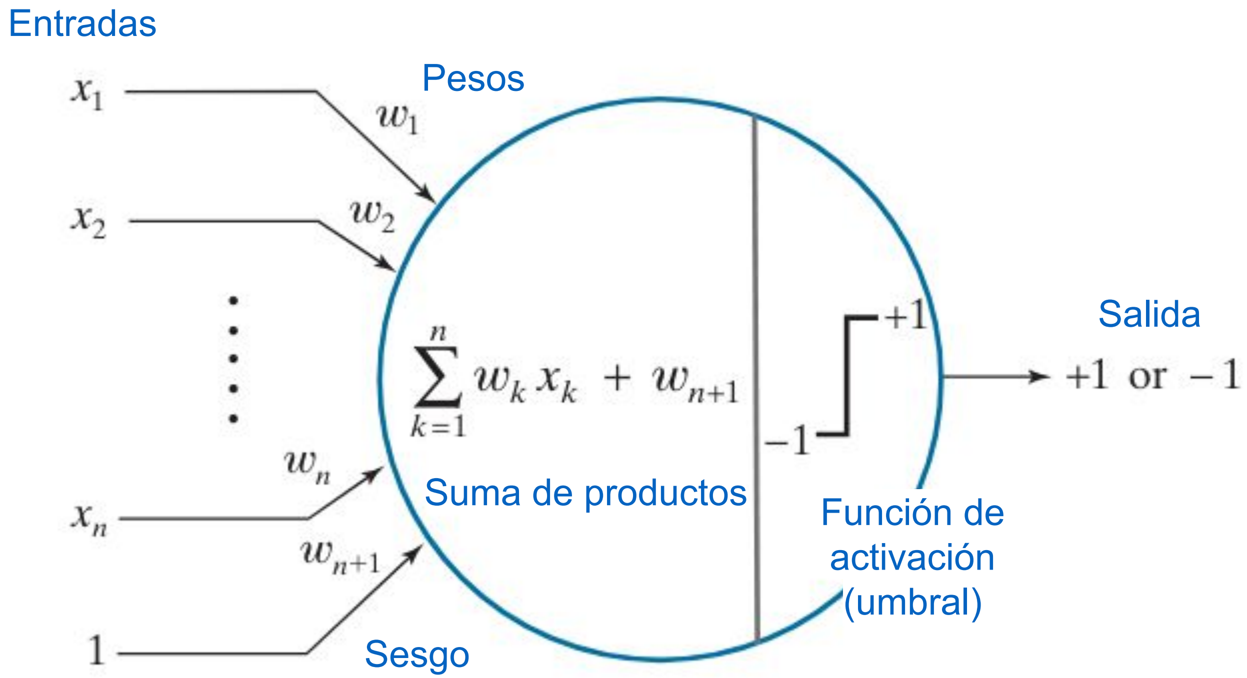


Imagen tomada de González, Woods, Digital Image Processing, 2018.

Neurona artificial vs neurona biológica

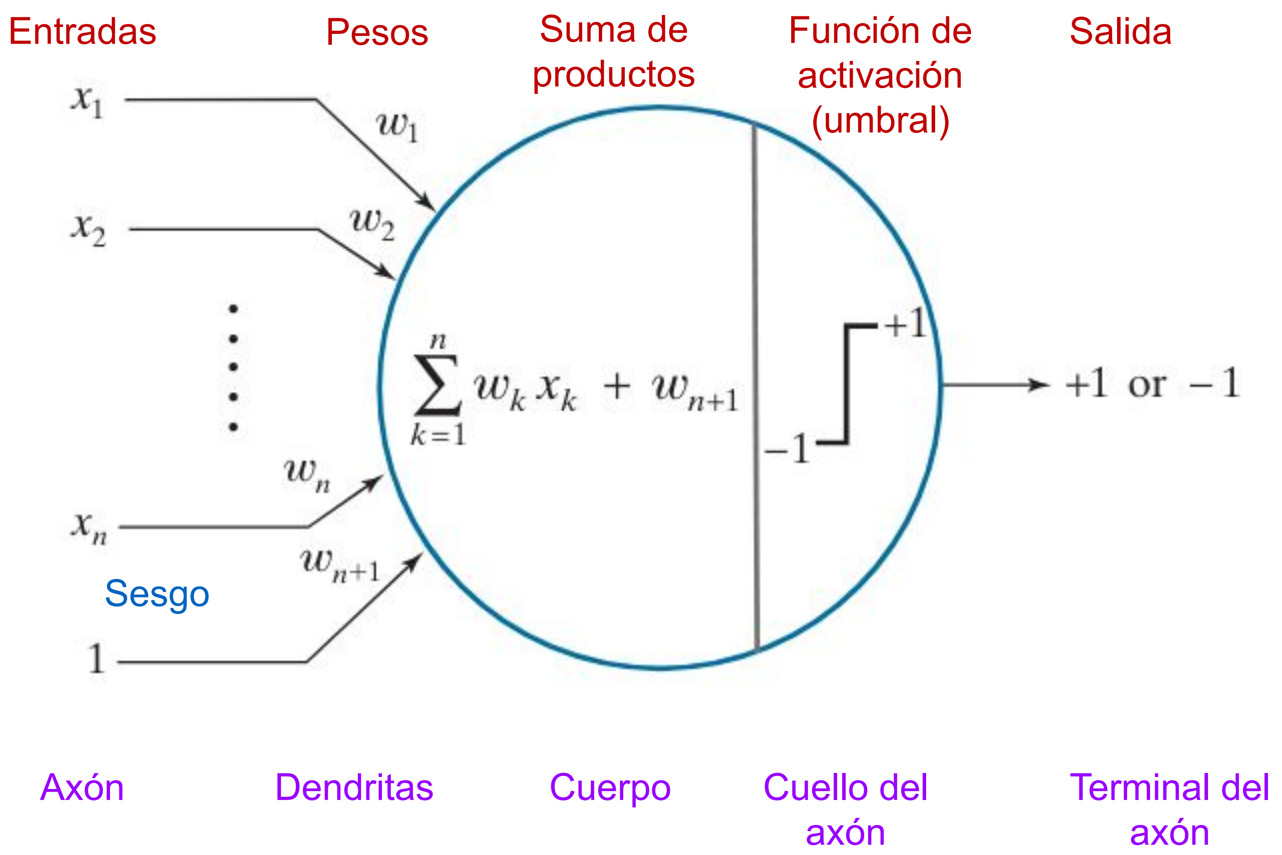
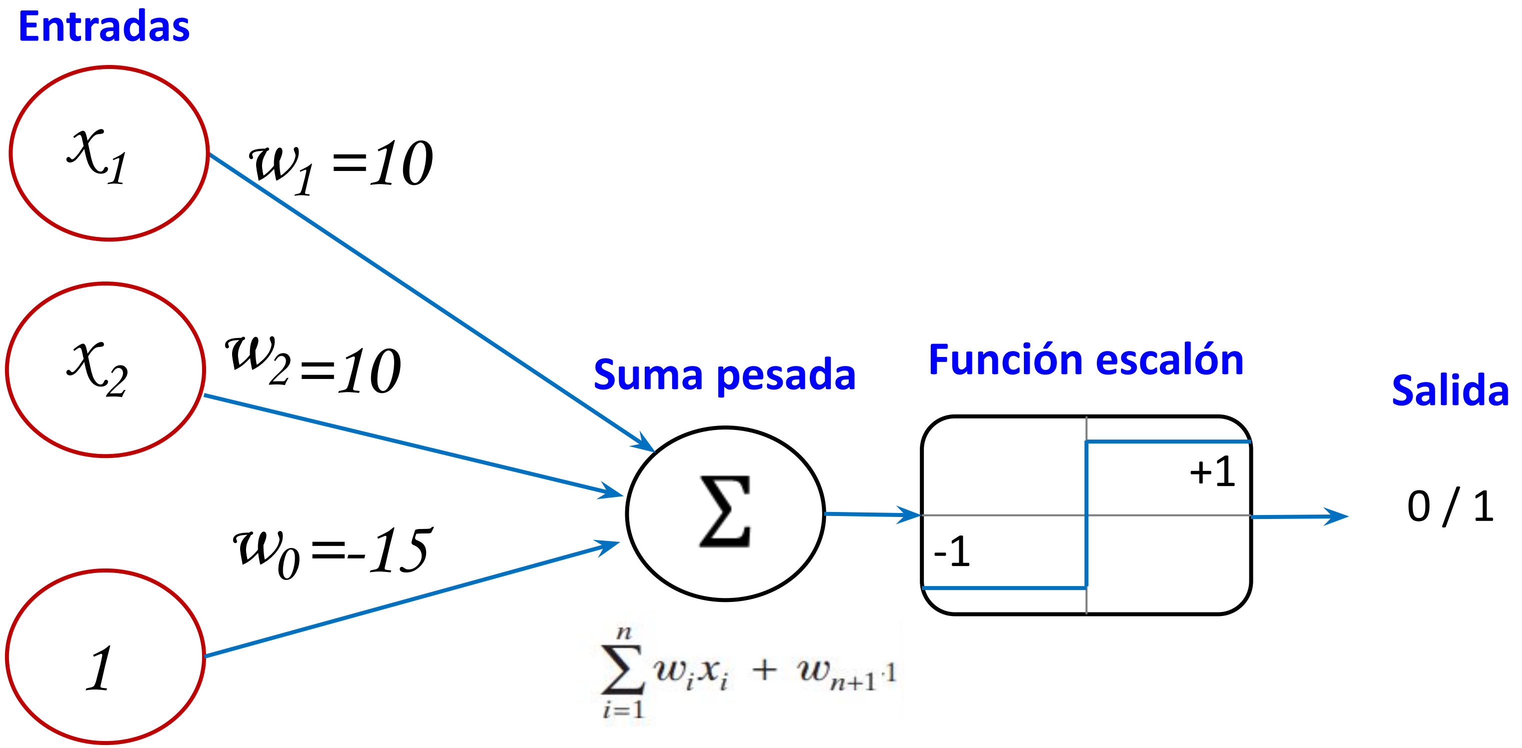


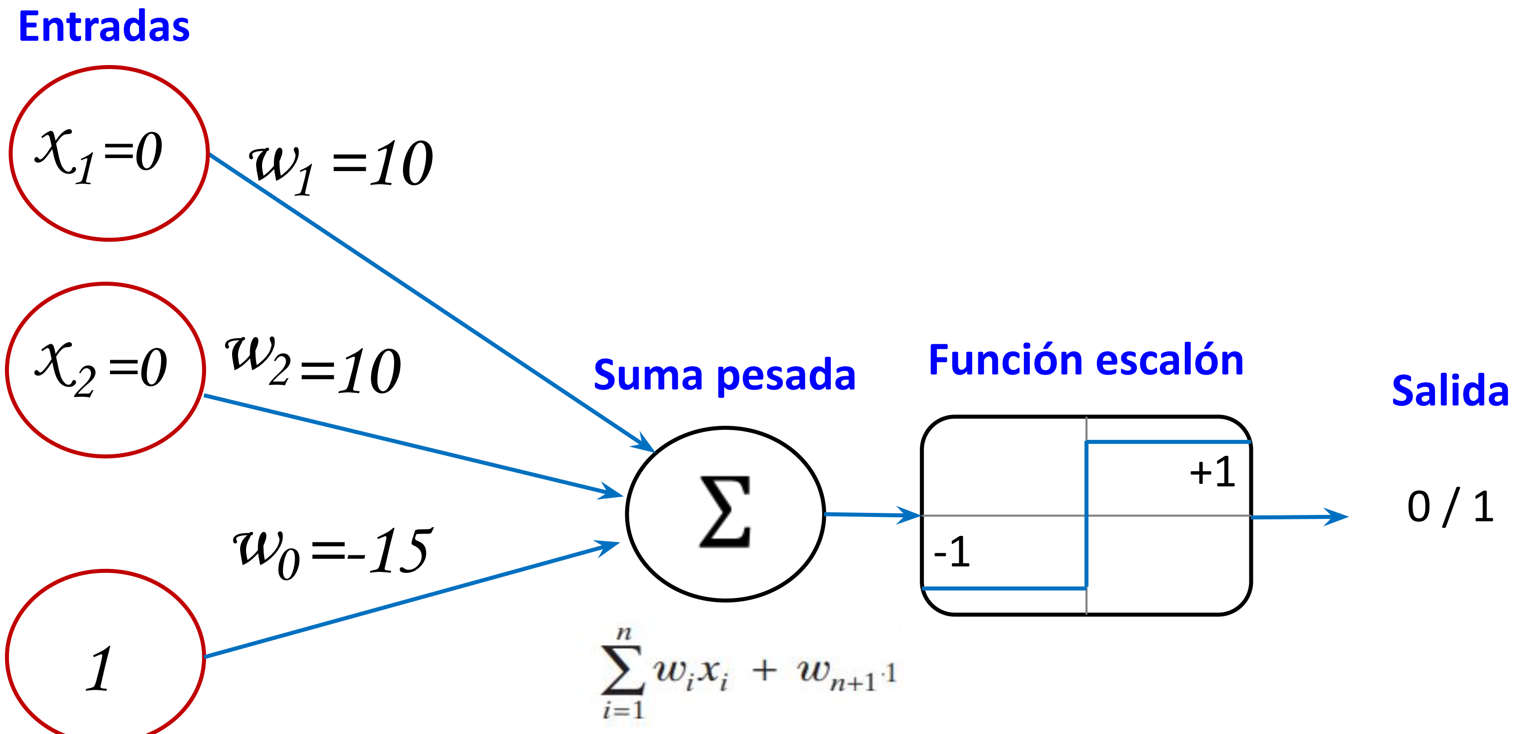
Imagen tomada de González, Woods, Digital Image Processing, 2018.

Compuerta AND



x_1	x_2	x_1 AND x_2
0	0	
0	1	
1	0	
1	1	

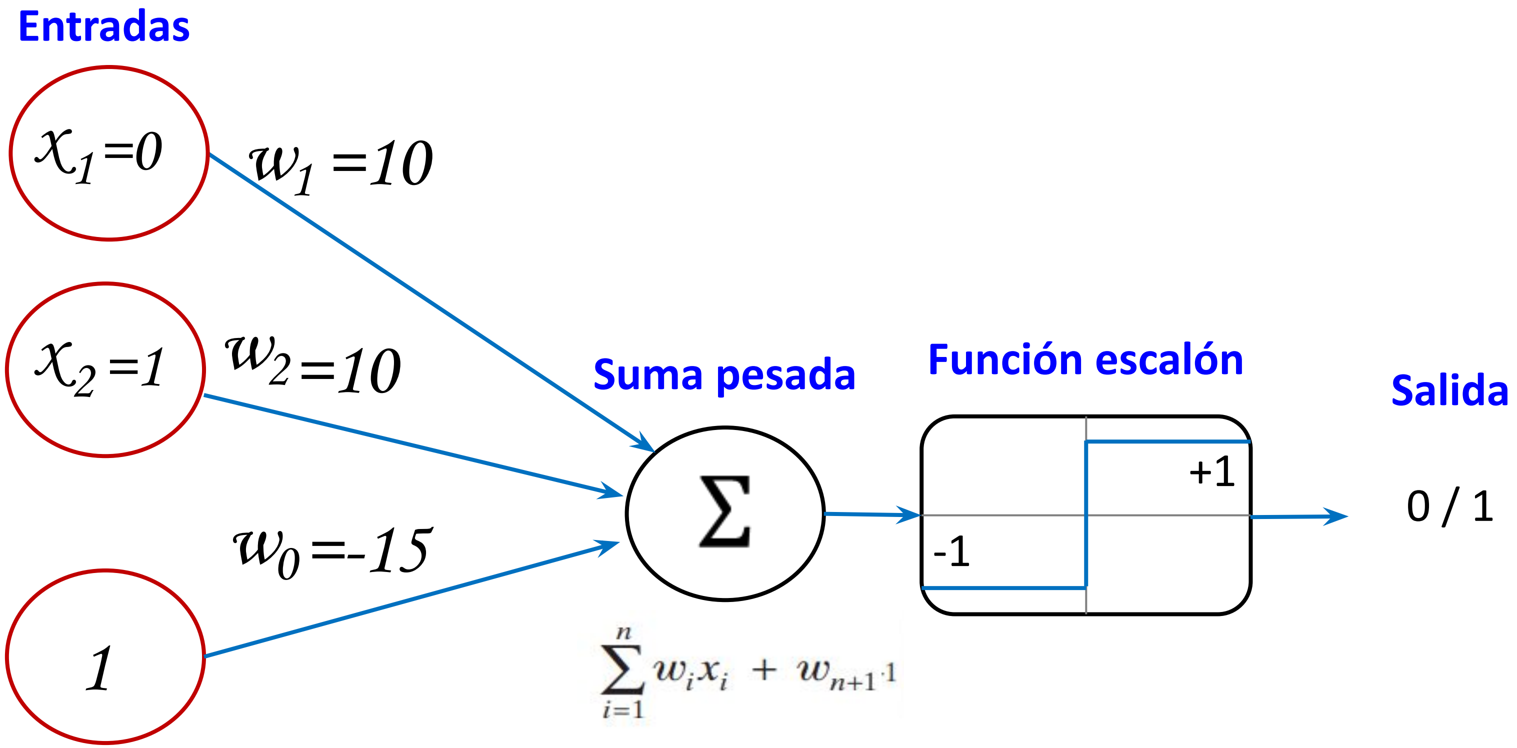
Compuerta AND



x_1	x_2	x_1 AND x_2
0	0	0
0	1	
1	0	
1	1	

$$h((0 \times 10) + (0 \times 10)\,(1 \times -15)\,) = h(-15) = 0$$

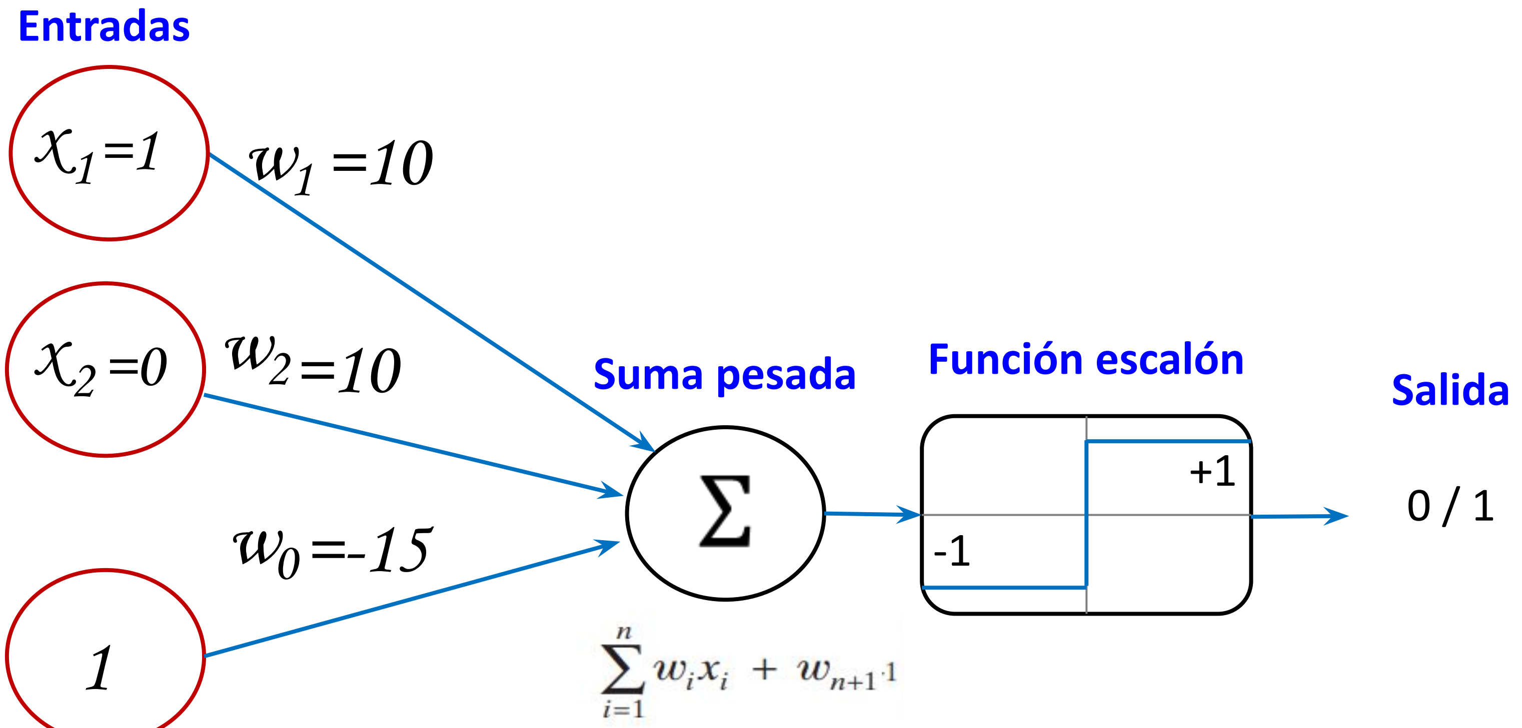
Compuerta AND



x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	
1	1	

$$h((0 \times 10) + (1 \times 10)\,(1 \times -15)\,) = h(-5) = 0$$

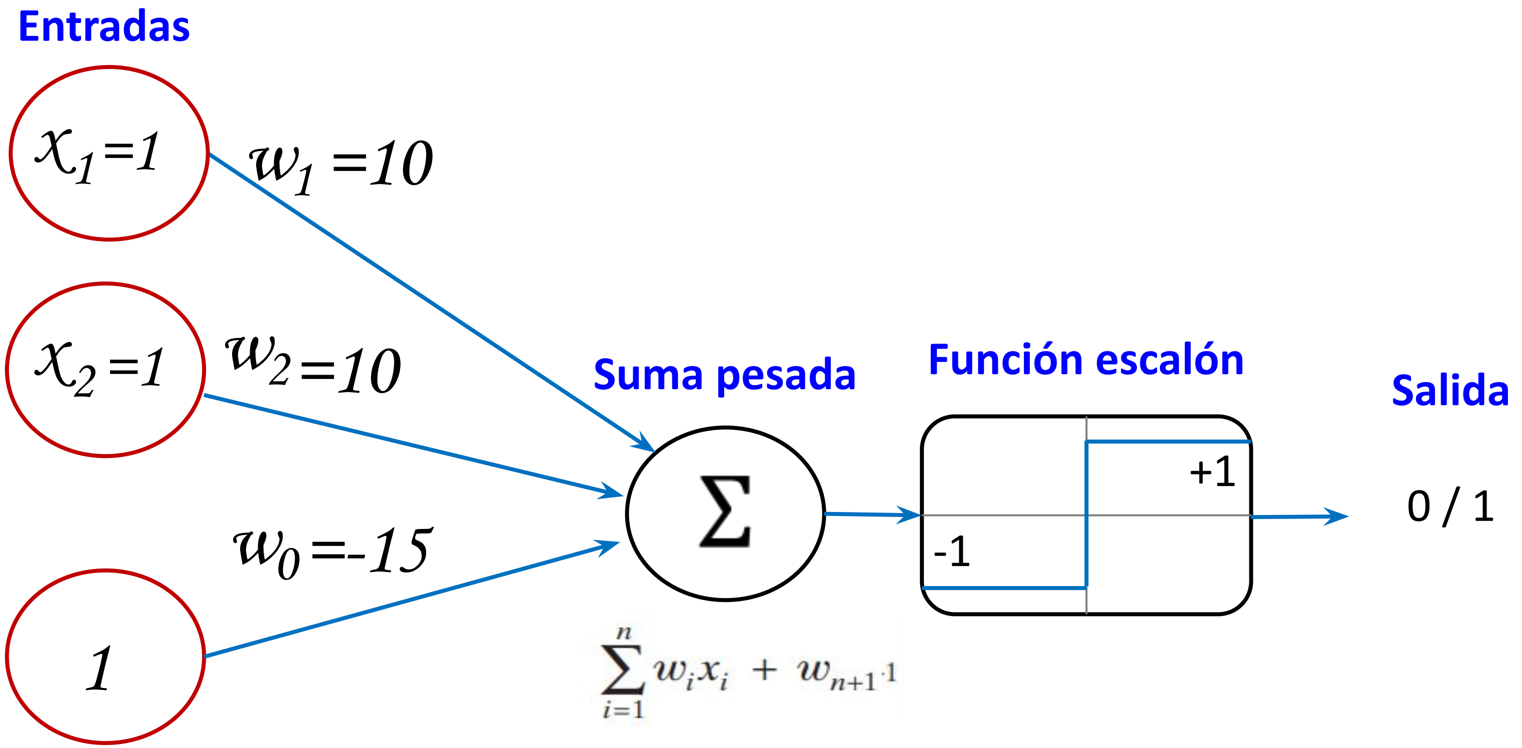
Compuerta AND



$$h((1 \times 10) + (0 \times 10)\,(1 \times -15)\,) = h(-5) = 0$$

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	

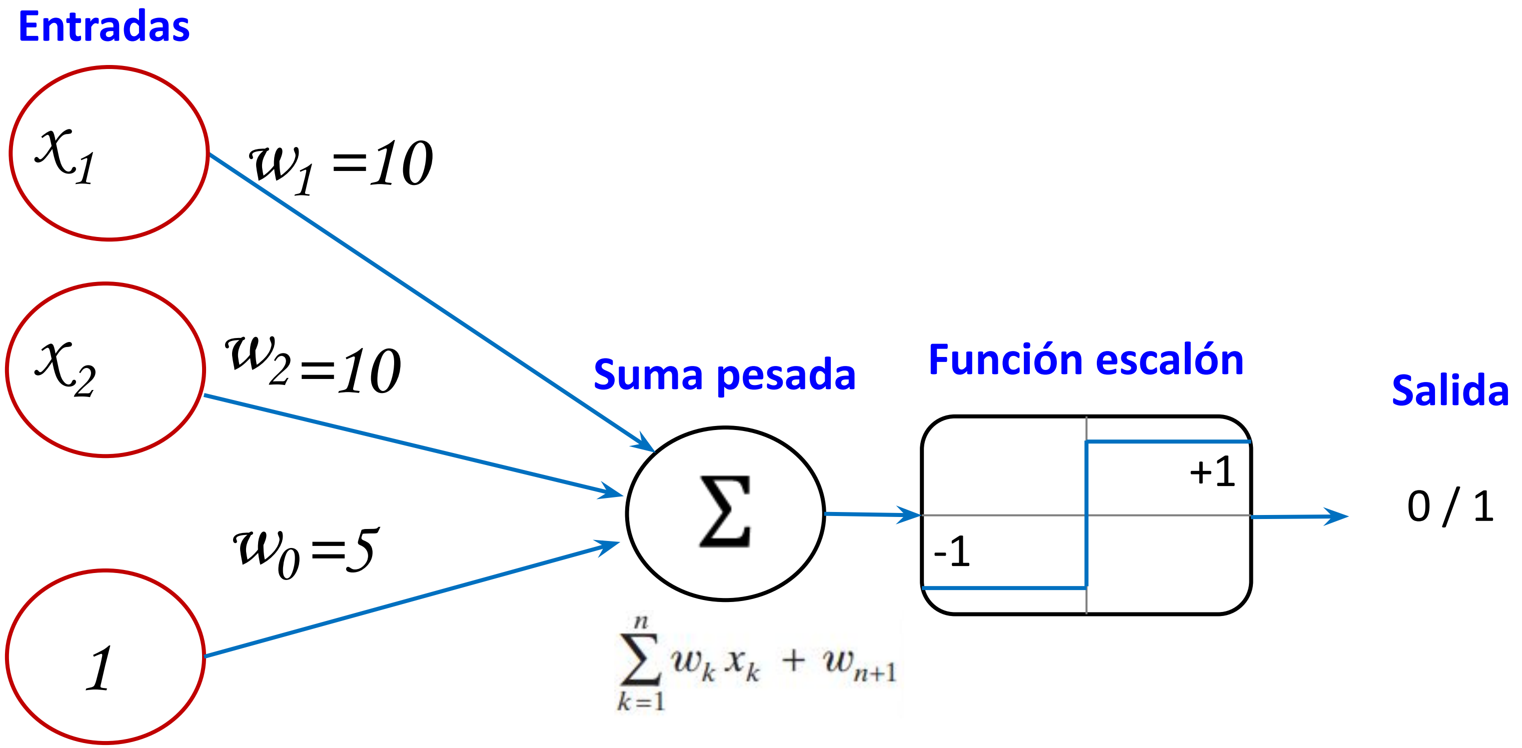
Compuerta AND



x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

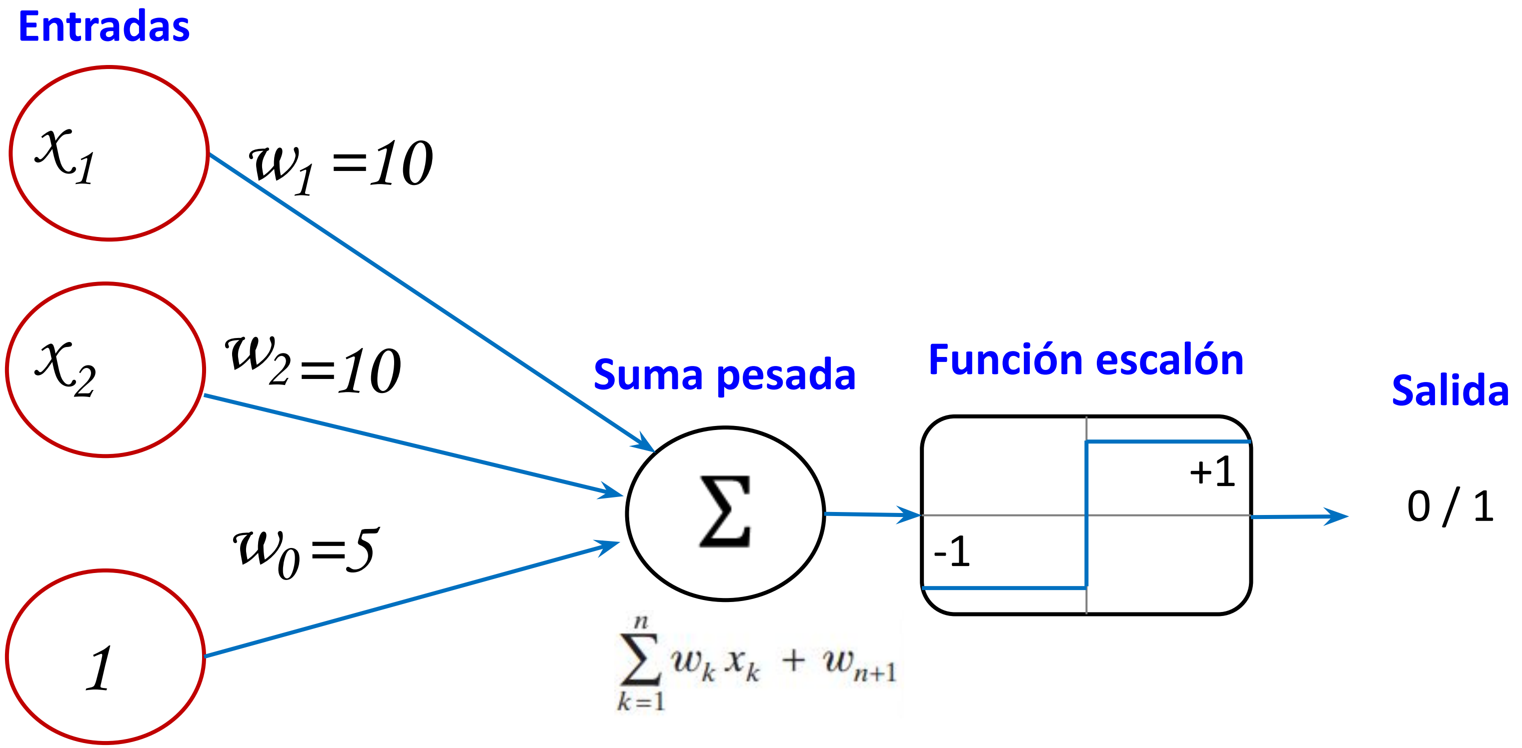
$$h((1 \times 10) + (1 \times 10)\,(1 \times -15)\,) = h(5) = 1$$

Compuerta NOR



x_1	x_2	x_1 NOR x_2
0	0	
0	1	
1	0	
1	1	

Compuerta NOR



x_1	x_2	x_1 NOR x_2
0	0	1
0	1	0
1	0	0
1	1	0

Análisis

- ¿Qué observas en común con las compuertas AND y NOR?

Análisis

- ¿Qué observas en común con las compuertas AND y NOR?
- ¿Conoces algún escenario que tenga ese comportamiento?

Problemas no lineales

¿Cómo modelar una compuerta XOR?

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

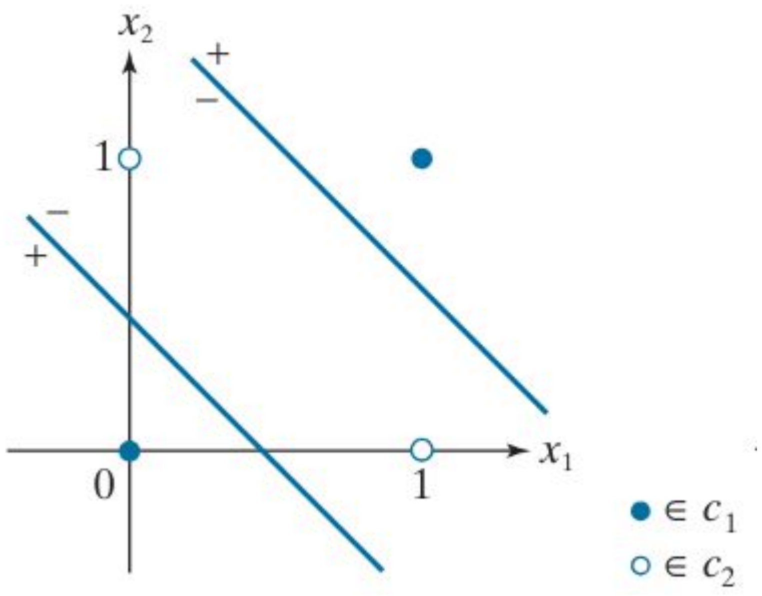
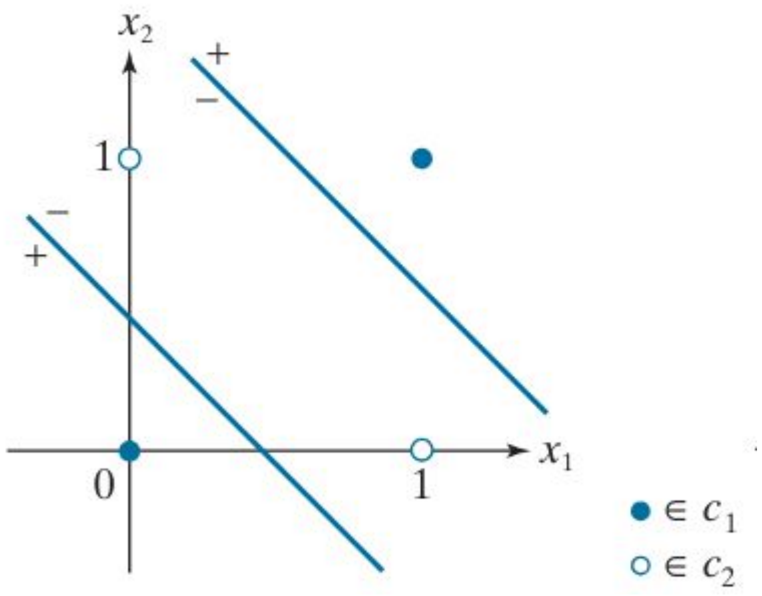


Imagen tomada de González, Woods, Digital Image Processing, 2018.

Problemas no lineales

¿Cómo modelar una compuerta XOR?

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

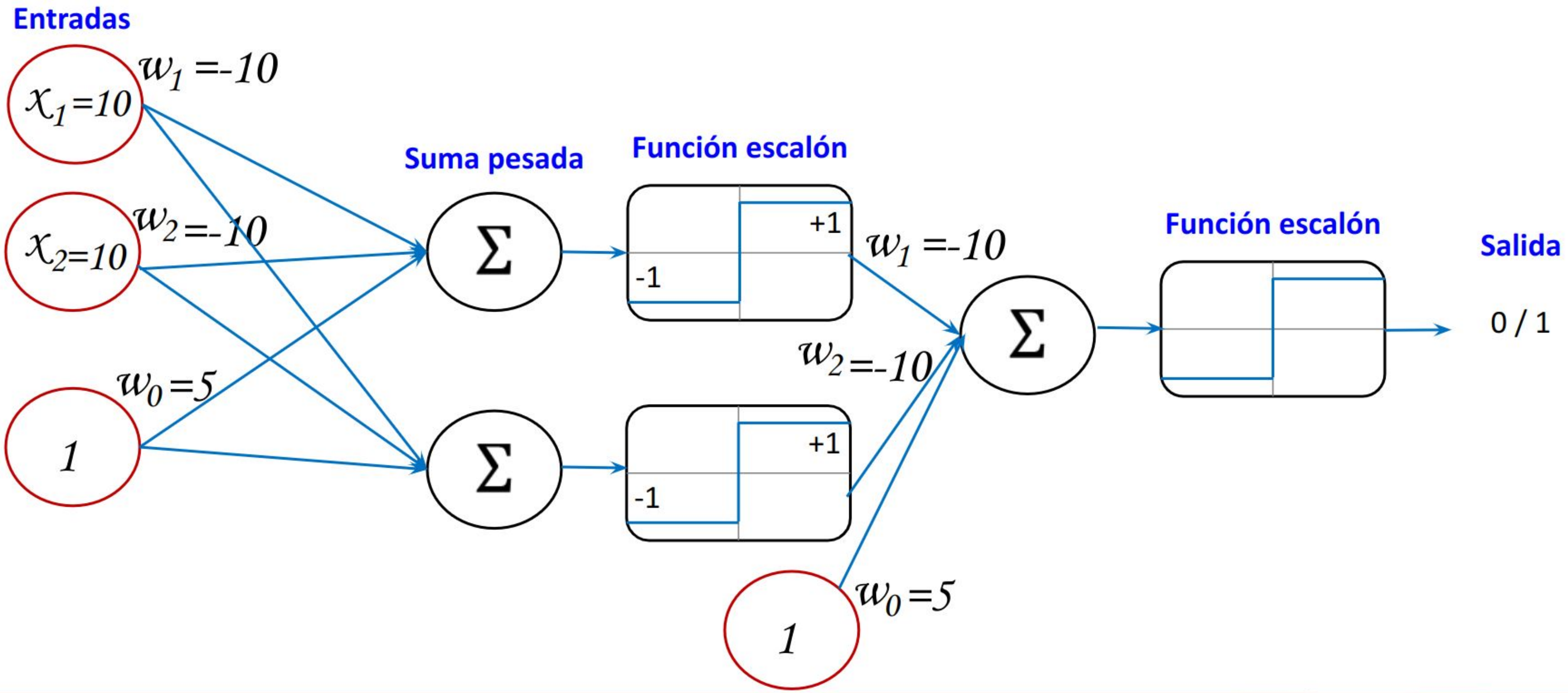


Minsky y Papert demostraron que era imposible aprender la compuerta XOR con perceptrones (1969).

Imagen tomada de González, Woods, Digital Image Processing, 2018.

Compuerta XOR (múltiples capas)

X_1 XOR X_2 = (X_1 AND X_2) NOR (X_1 NOR X_2)



Perceptrón multicapa

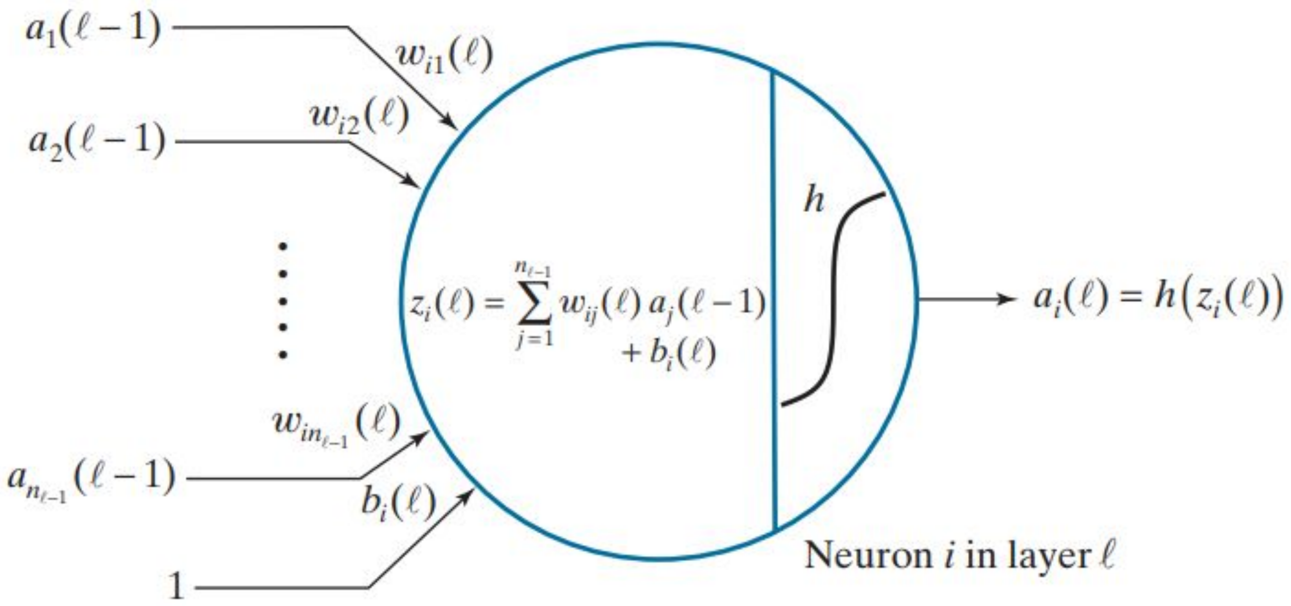


Imagen tomada de González, Woods, Digital Image Processing, 2018.

Función de activación

- Son operadores diferenciables para transformar señales de entrada en salidas, mientras que la mayoría de ellos añaden no linealidad.
- Se les conoce como umbrales.
- **Sin** funciones de activación, las redes neuronales solo se enfocarían en operaciones lineales.

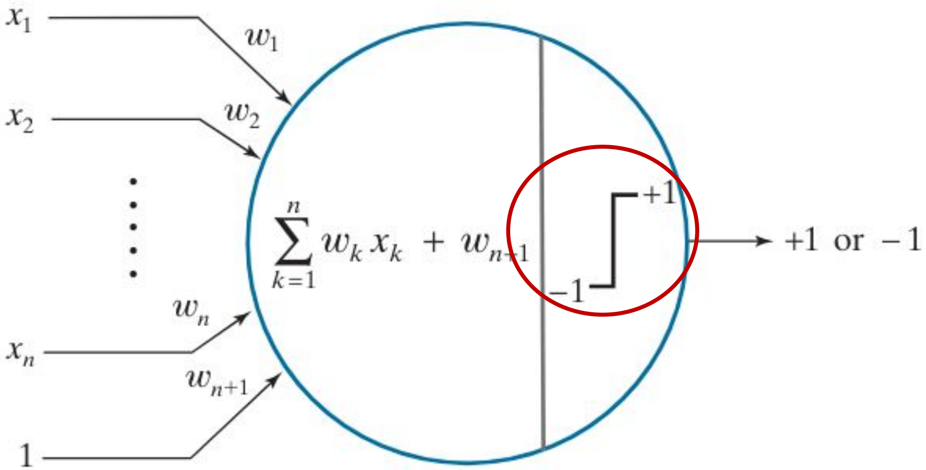
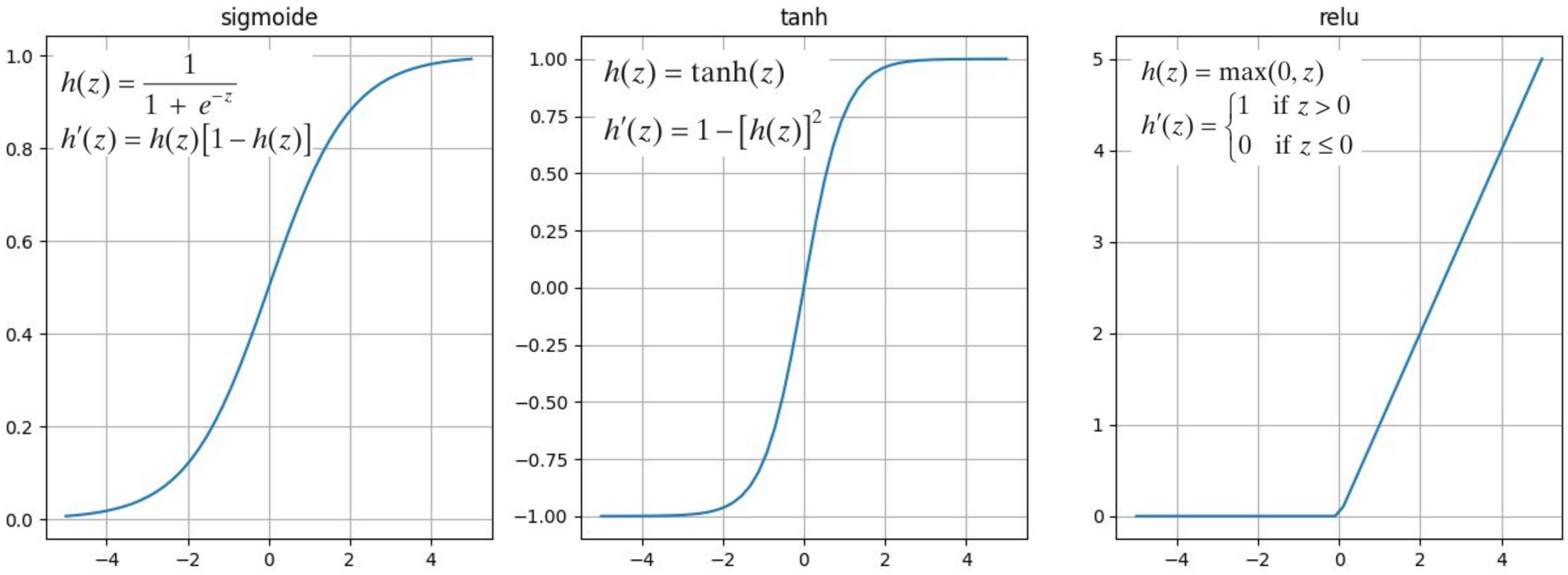


Imagen tomada de González, Woods, Digital Image Processing, 2018.

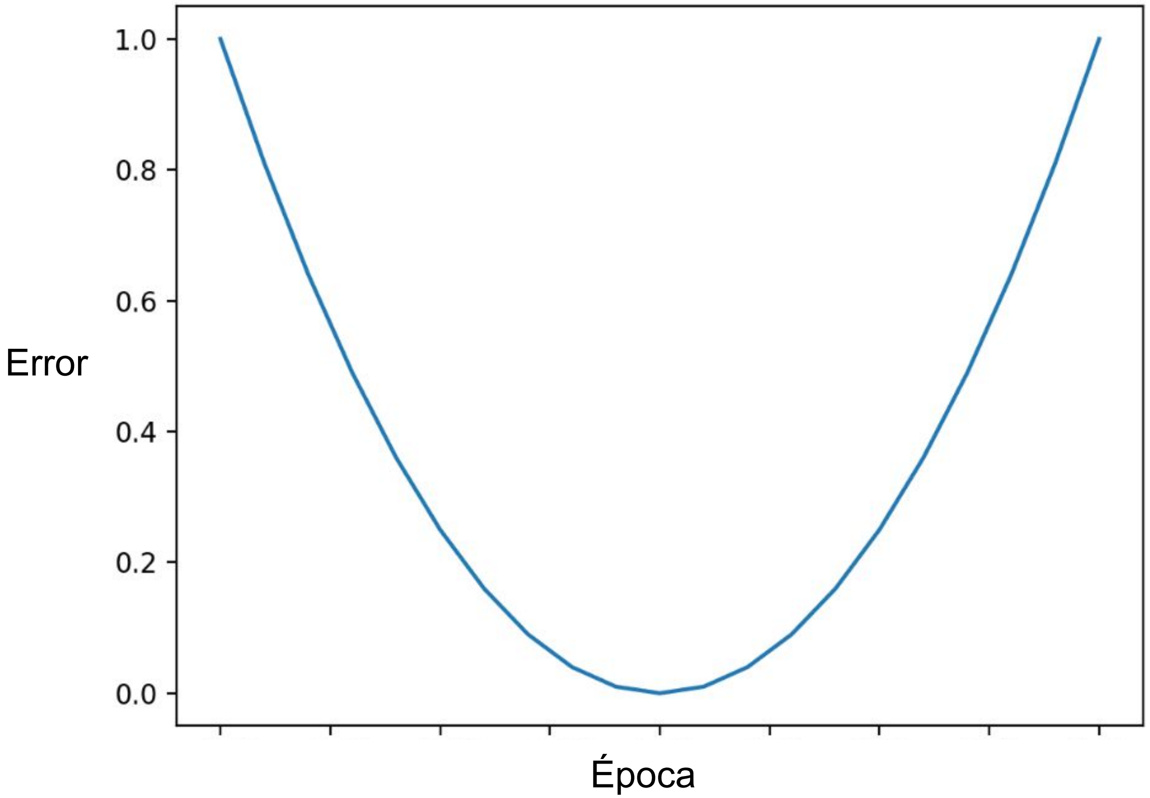
Funciones de activación más comunes



Time to Code

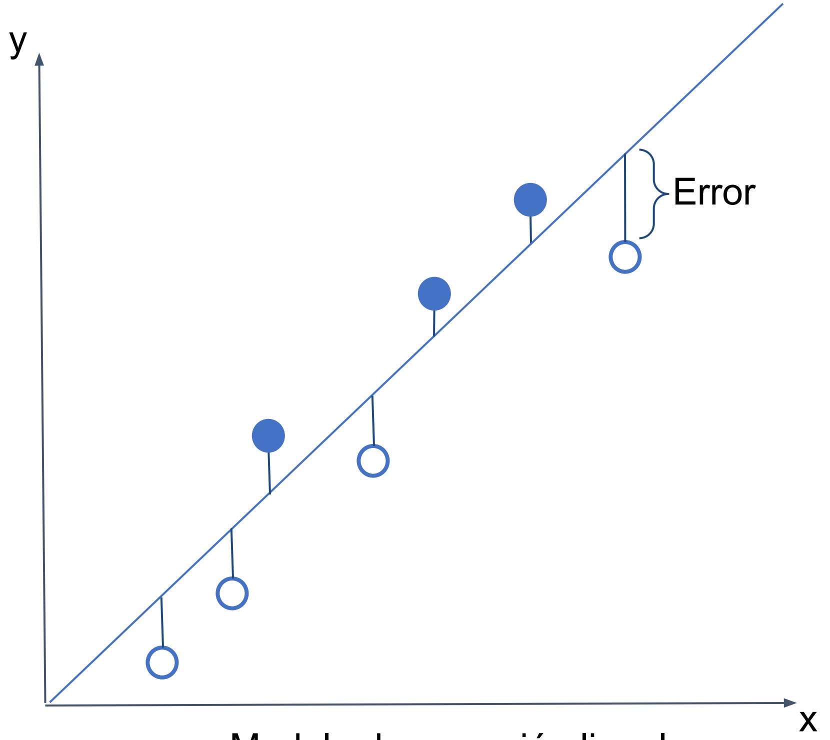
Función de pérdida

Cuando entrenamos redes neuronales se busca encontrar los pesos y sesgos que minimicen una función de pérdida



Función de pérdida

- Las funciones de pérdida **cuantifican** la distancia entre los valores reales y predichos del objetivo.
- La pérdida normalmente será un número no negativo donde los valores más pequeños son mejores y las predicciones perfectas incurren en una pérdida de 0.

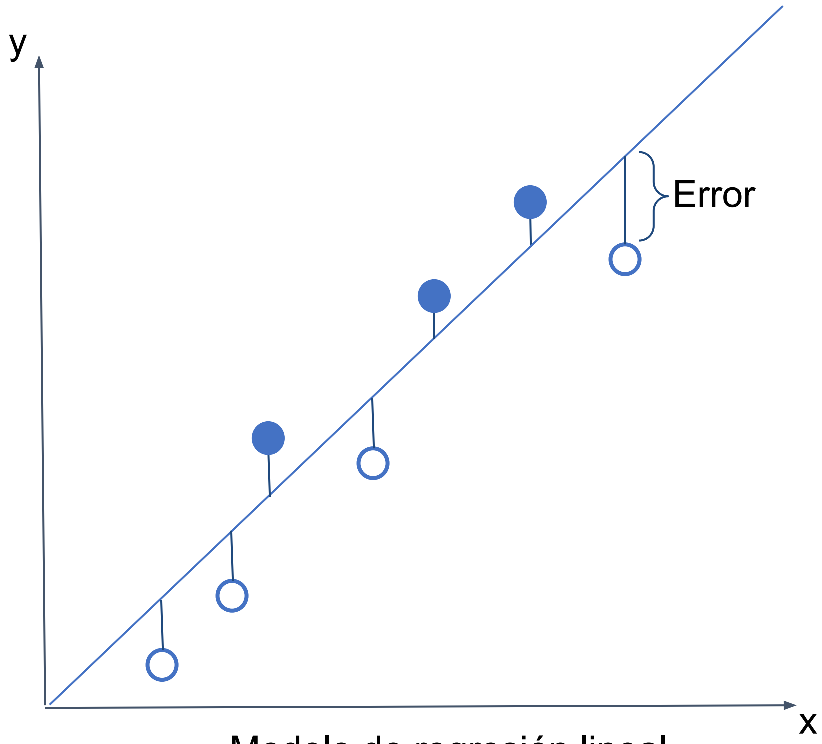


Modelo de regresión lineal

Regresión lineal

La regresión lineal modela la relación entre una variable dependiente y una o más variables independientes, asumiendo que dicha relación es lineal.

Es decir, las variables independientes tienen una relación **directa** con la variable dependiente y no tienen ninguna relación con las demás variables independientes.



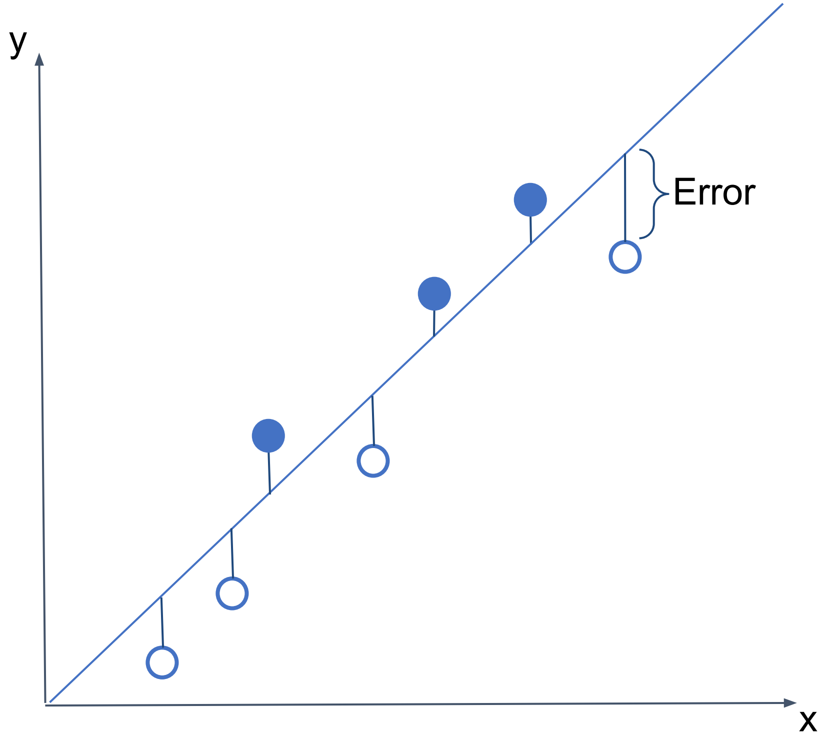
Modelo de regresión lineal

- Función de activación: lineal
- Función de pérdida: error cuadrático medio
- Salida: contínua

Regresión lineal y el error cuadrático medio

Para problemas de regresión, la función de pérdida más común es el error cuadrático medio (MSE, por las siglas en inglés de *mean square error*).

$$MSE = \frac{1}{N}\sum_{i}^{N}(Y_i - \hat{Y}_i)^2$$

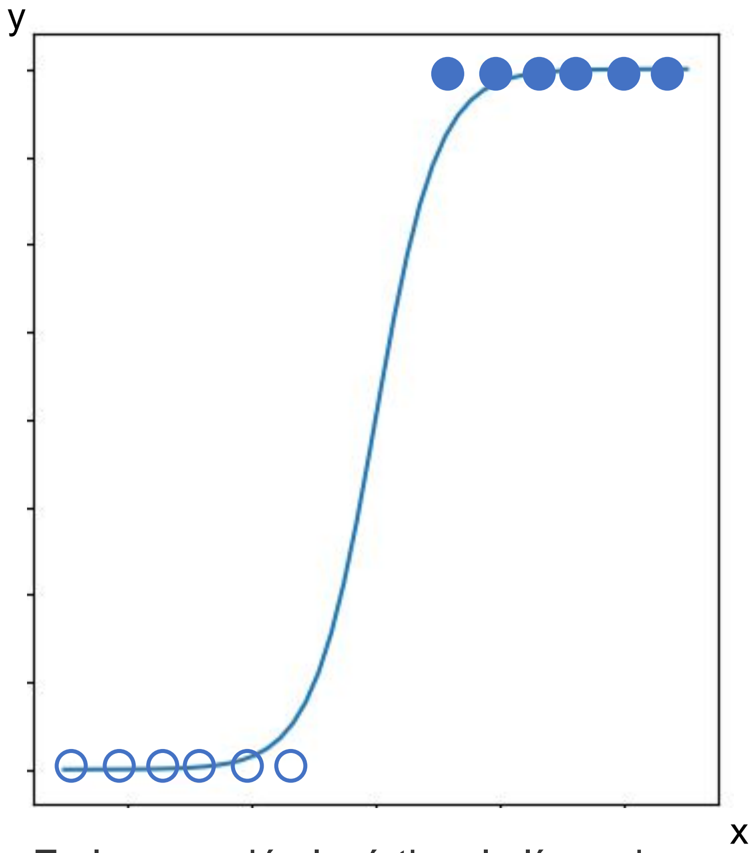


En la regresión lineal, la línea de regresión es recta. Cualquier cambio en una variable independiente tiene un efecto directo en la variable dependiente.

Regresión logística

La regresión logística modela la probabilidad de que ocurra un evento binario en función de una o más variables independientes.

- Función de activación: *sigmoide* o logística
- Función de pérdida: entropía cruzada binaria
- Salida: categórica

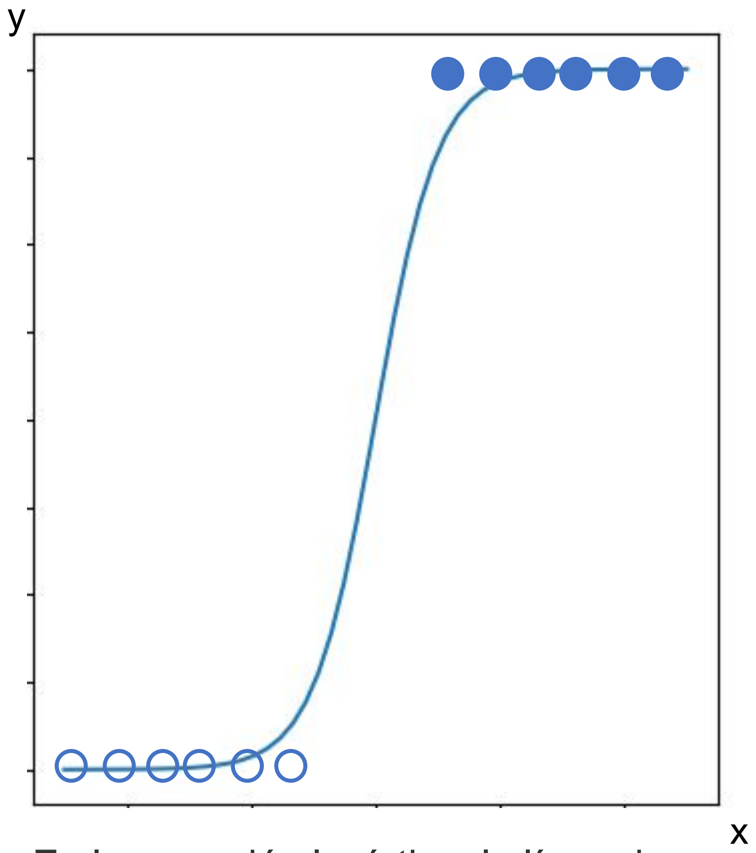


En la regresión logística, la línea de regresión es una curva en forma de S, también conocida como curva sigmoidea.

Regresión logística y la entropía cruzada binaria

La entropía cruzada binaria aplica una transformación logit, o el logaritmo natural de las probabilidades, a la probabilidad de éxito o fracaso de una variable categórica concreta.

$$ECB(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{N} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$



En la regresión logística, la línea de regresión es una curva en forma de S, también conocida como curva sigmoidea.

Tipos de clasificación

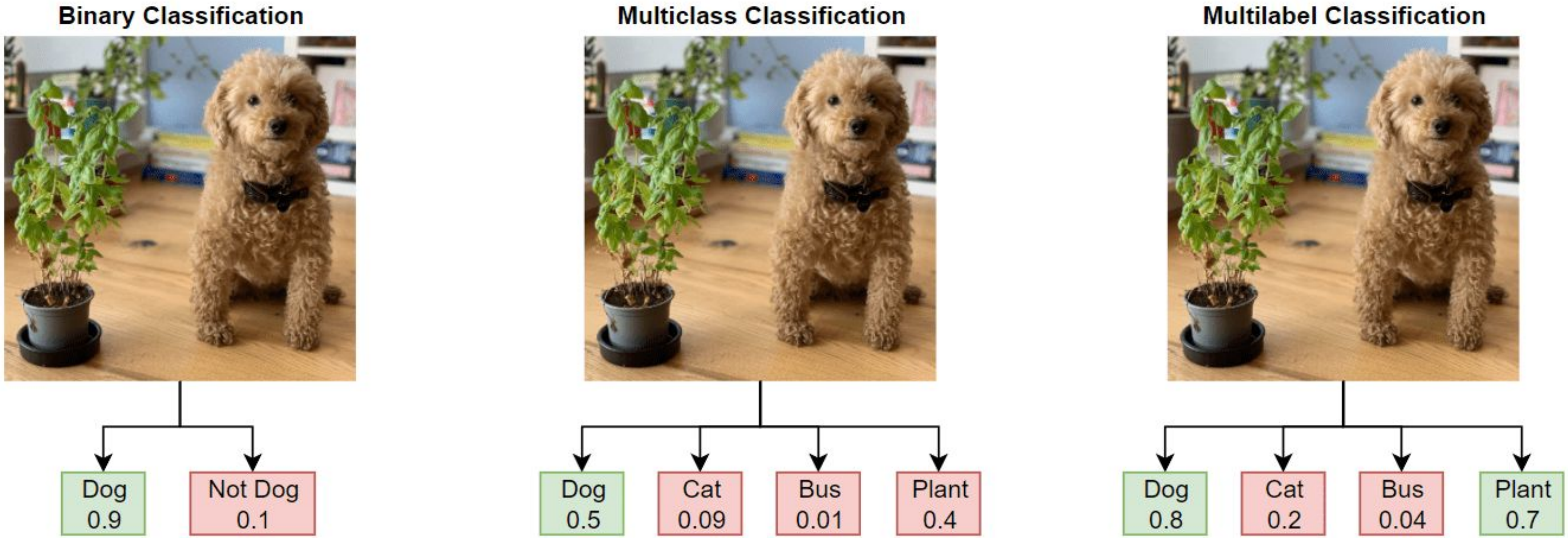
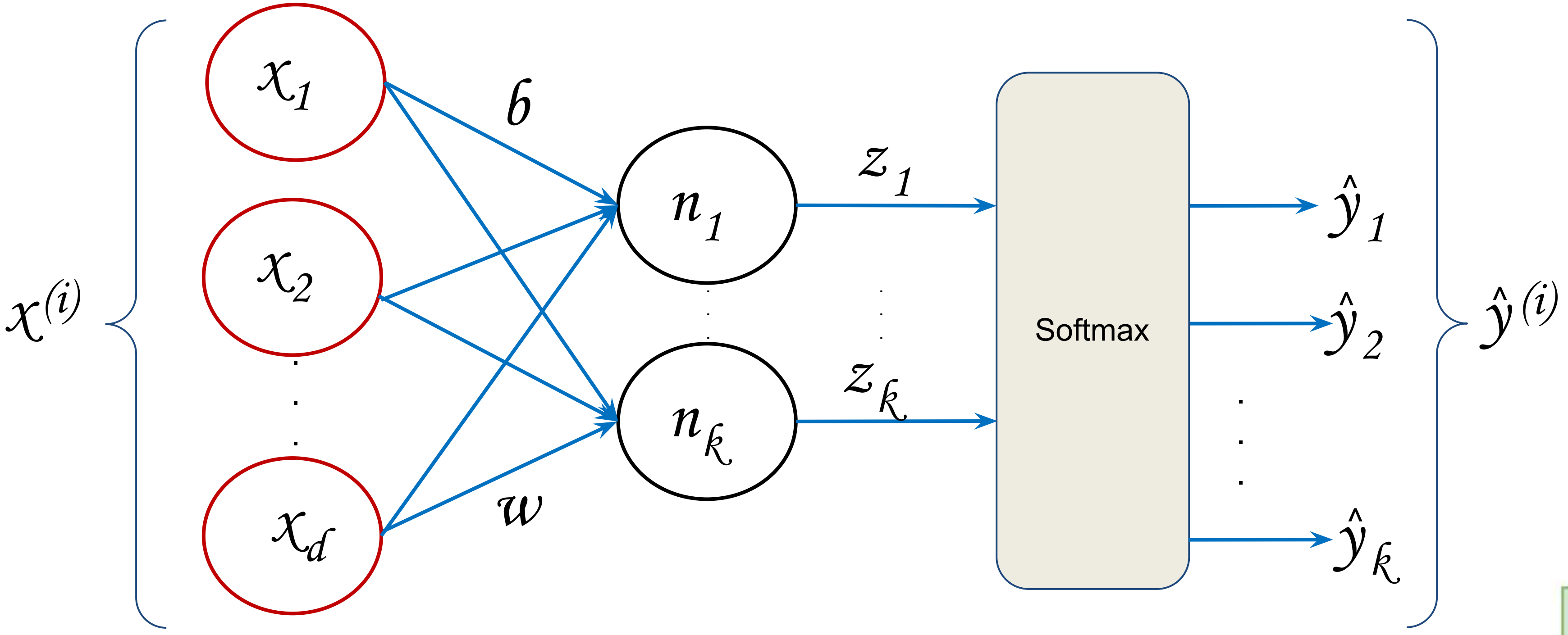


Imagen tomada de MathWorks, 2024.

Regresión logística multinomial o *softmax*



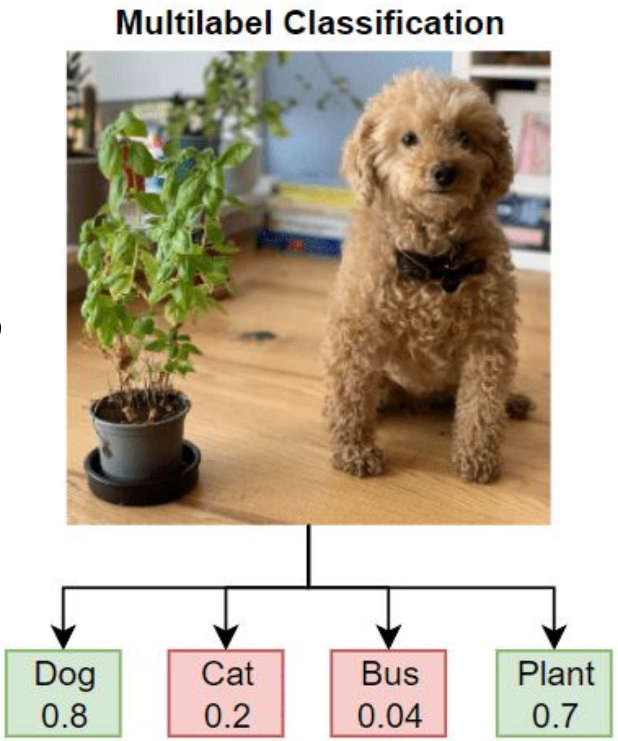


Imagen tomada de MathWorks, 2024.

- Función de activación: *softmax*
- Función de pérdida: entropía cruzada categórica
- Salida: categórica

Regresión logística multinomial y entropía cruzada categórica

Función de activación softmax

- Convierte un vector de *K* números reales en una distribución de probabilidad de *K* resultados posibles.
- Es una generalización de la función logística a múltiples dimensiones.

$$\text{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, i = 1, \ldots, K$$

Regresión logística multinomial y entropía cruzada categórica

Función de activación softmax

- Convierte un vector de K números reales en una distribución de probabilidad de K resultados posibles.
- Es una generalización de la función logística a múltiples dimensiones.

$$\mathrm{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, i = 1, \ldots, K$$

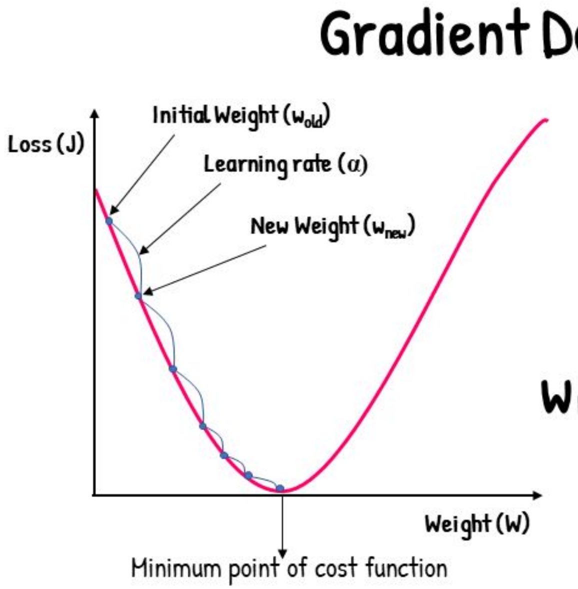
Función de pérdida de entropía cruzada categórica (ECC)

$$ECC(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i=1}^{n}\sum_{c=1}^{k}\left[y_c^{(i)} \cdot \log\left(\overbrace{\frac{e^{z_c^{(i)}}}{\sum_j e^{z_j^{(i)}}}}^{\hat{y}_c^{(i)}} \right) \right]$$

Time to Code

Descenso del gradiente

- Es un algoritmo de optimización para minimizar una función (función de pérdida).
- La meta del algoritmo es encontrar los parámetros del modelo: pesos (w) y sesgo(b).
- Hasta que la función sea cercana o igual a cero, el modelo continuará ajustando sus parámetros para reducir el error.



$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

Imagen tomada de *Analytics vidhya*, 2024.

Algoritmo del descenso del gradiente

Algoritmo iterativo que va moviendo los pesos w y sesgos b hacia donde la pérdida descienda más rápido en el vecindario.

1. En t=0 se inicializan los parámetros $\theta^{[0]}$
2. Se actualizan los parámetros $\theta^{[0+1]}$ usando la siguiente regla:

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}^{[t]})$$

donde

$$\boldsymbol{\theta} = \{\mathbf{w}, \mathbf{b}\}$$

$$\nabla \mathcal{L}(\boldsymbol{\theta}^{[t]}) = \left[\frac{\partial \mathcal{L}}{\partial \theta_0^{[t]}}, \cdots, \frac{\partial \mathcal{L}}{\partial \theta_d^{[t]}} \right]$$

3. Se repite el paso 2 hasta que se cumpla algún criterio de convergencia

A α se le conoce como tasa de aprendizaje.

Imagen tomada de *Analytics vidhya*, 2024.

Ejemplo de clasificación de correo electrónico

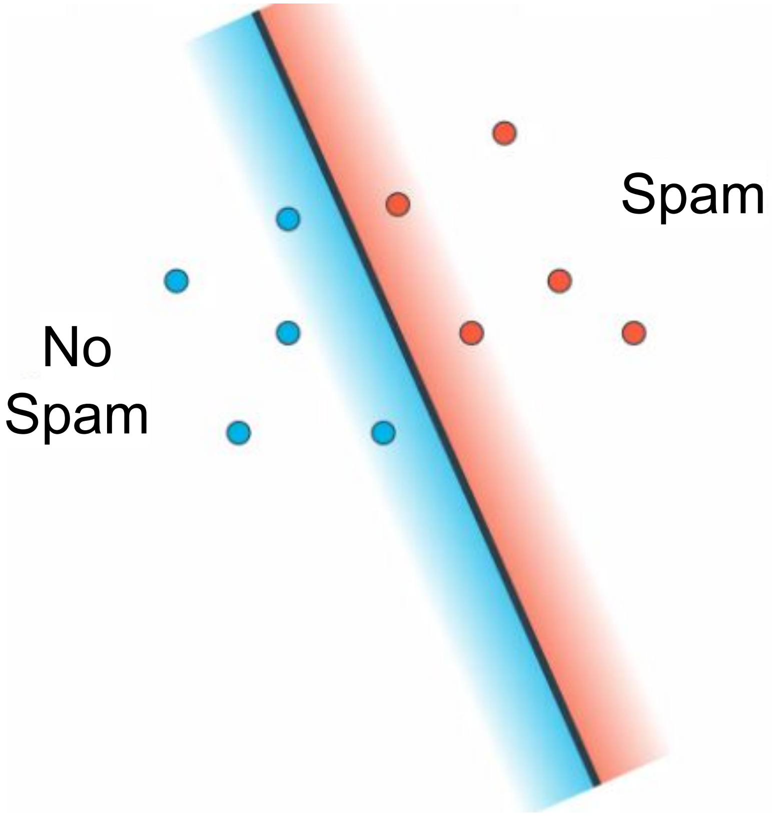


Imagen tomada de Serrano, 2020.

Pasos de optimización mediante gradiente descendente

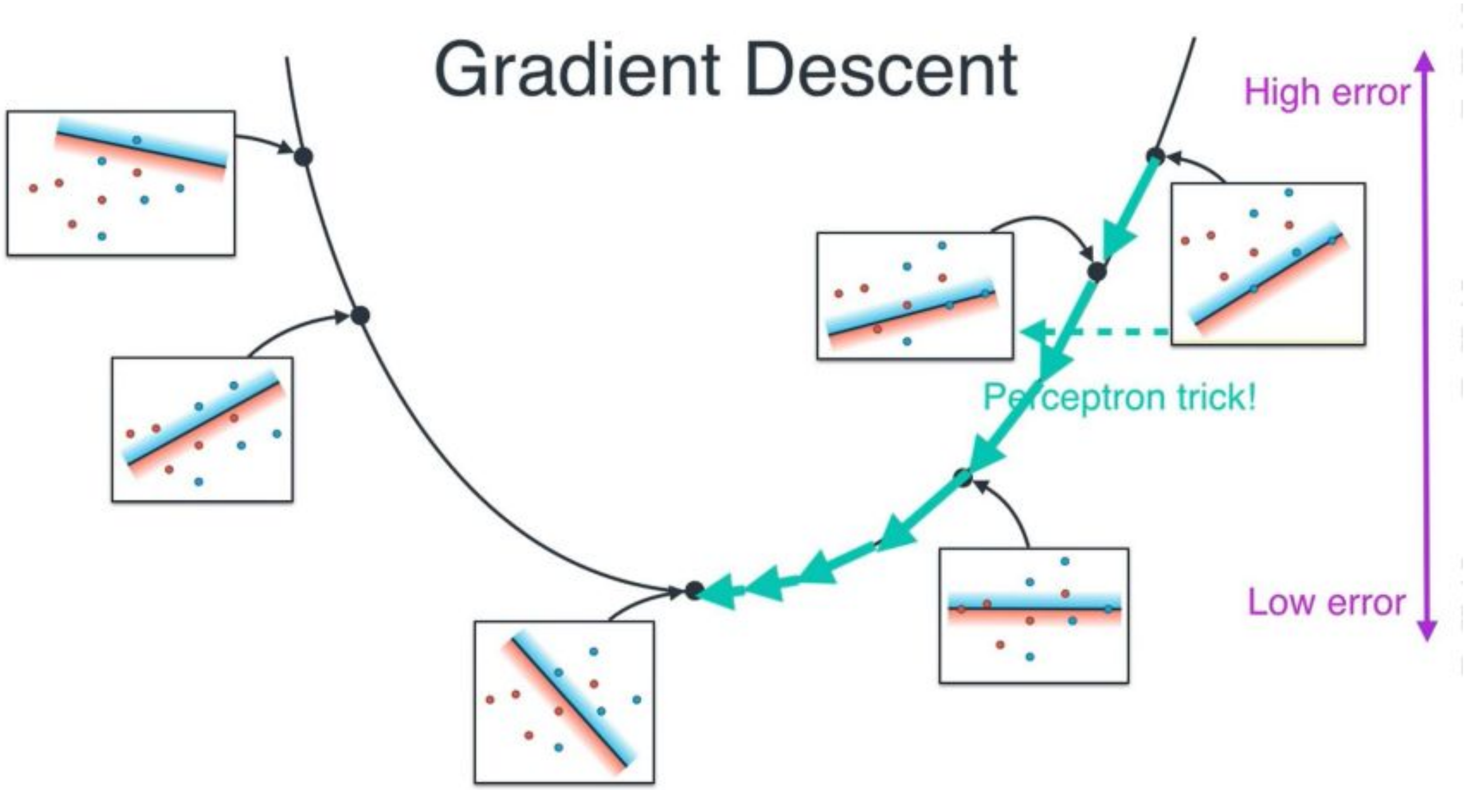


Imagen tomada de Serrano, 2020.

Efecto de la tasa de aprendizaje en la función de pérdida

Si el learning rate es muy grande, se darán pasos grandes y se puede saltar a través de el mínimo de la función.

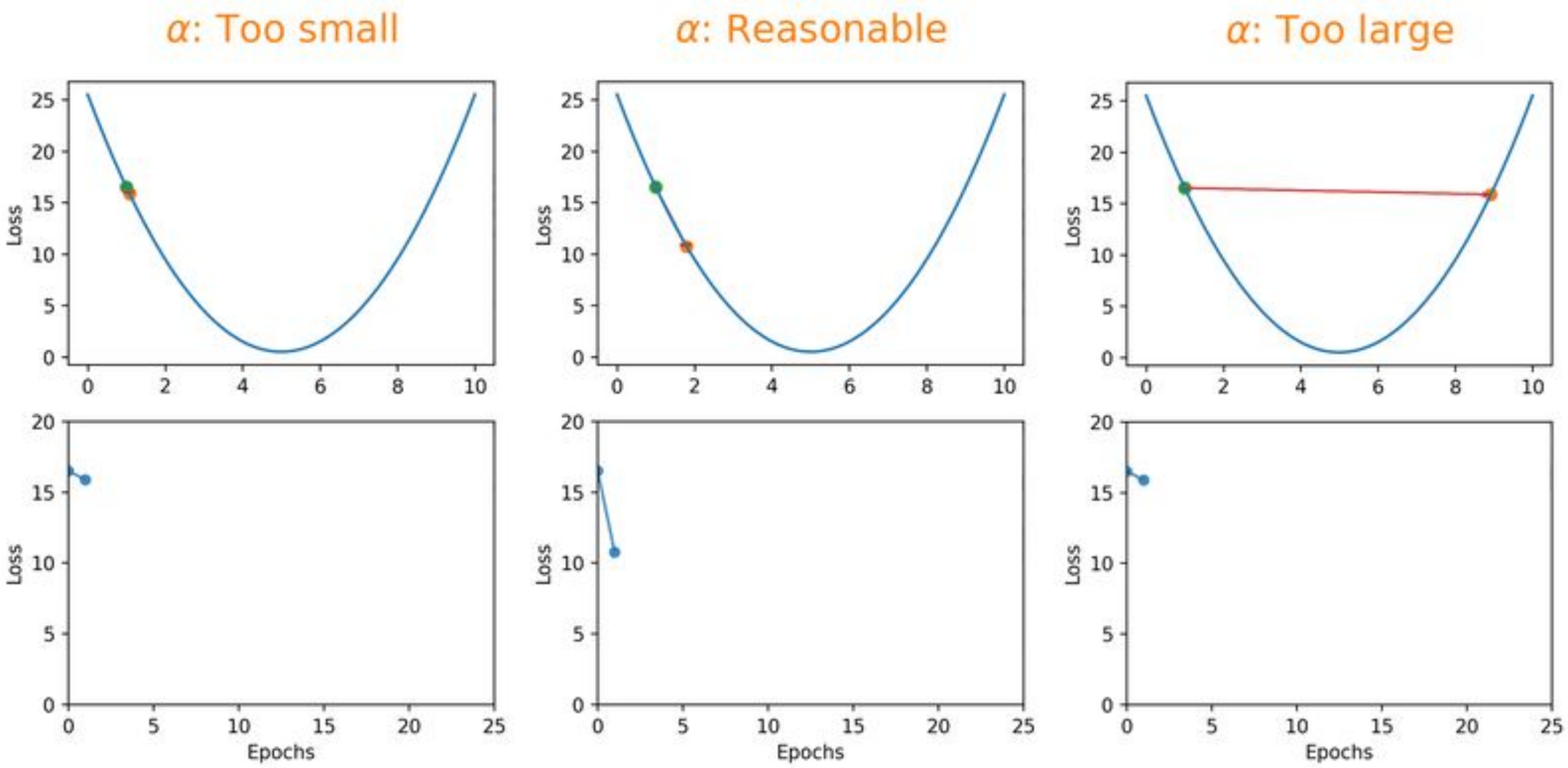


Imagen tomada de Andrew D. White, 2020.

Time to Code

Repaso

- Aprendimos a incorporar no linealidades en las redes neuronales.
- Identificamos diferentes funciones de activación y la diferencia entre ellas.
- Estudiamos la regresión lineal, logística y Softmax.
- Analizamos el algoritmo del descenso del gradiente

Referencias

- Zhang A, Lipton Z, Li M, and Smola J. Dive into Deep Learning. 2020. Disponible en https://d2l.ai/
- Murphy, K. P. (2022). Probabilistic Machine Learning: An introduction. MIT Press. Capítulo 8, 10 y 11. Disponible en https://probml.github.io/pml-book/book1.html
- Nielsen, M. (2019). Neural Networks and Deep Learning. Capítulo 1. Disponible en http://neuralnetworksanddeeplearning.com/index.html
- Rafael C. Gonzalez, Richard Eugene Woods (2018). Digital Image Processing. Capítulo 12. Disponible en https://dl.icdst.org/pdfs/files4/01c56e081202b62bd7d3b4f8545775fb.pdf

Contacto

Dra. Blanca Vázquez

Investigadora Postdoctoral

Unidad Académica del IIMAS

en el estado de Yucatán, UNAM.

Correo: blanca.vazquez@iimas.unam.mx

Github: https://github.com/blancavazquez

ArBio

Artificial Intelligence in Biomedicine Group (ArBio)

https://iimas.unam.mx/arbio