

# Neurona Artificial General y Estándar

**Deisy Chaves**

Oficina 10, 4<sup>ro</sup> Piso, Edificio B13

[deisy.chaves@correounivalle.edu.co](mailto:deisy.chaves@correounivalle.edu.co)

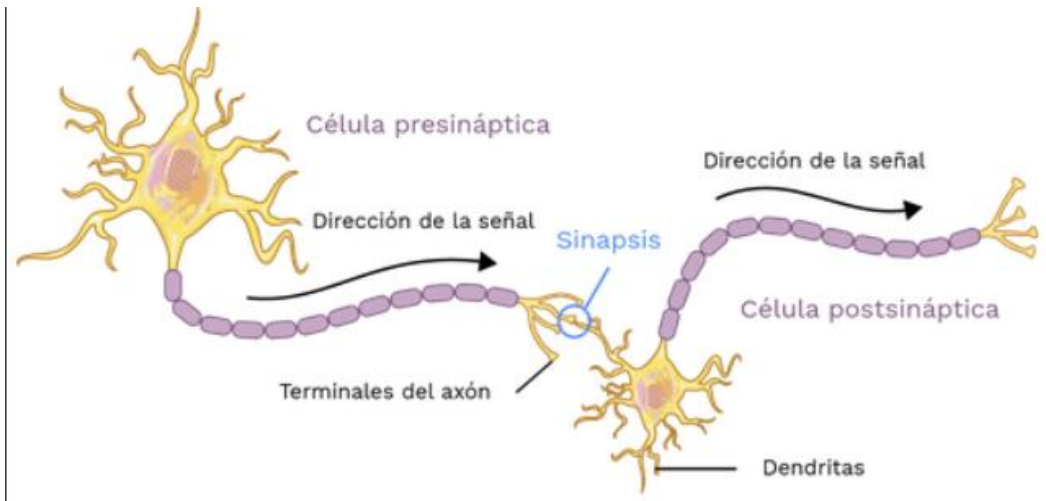
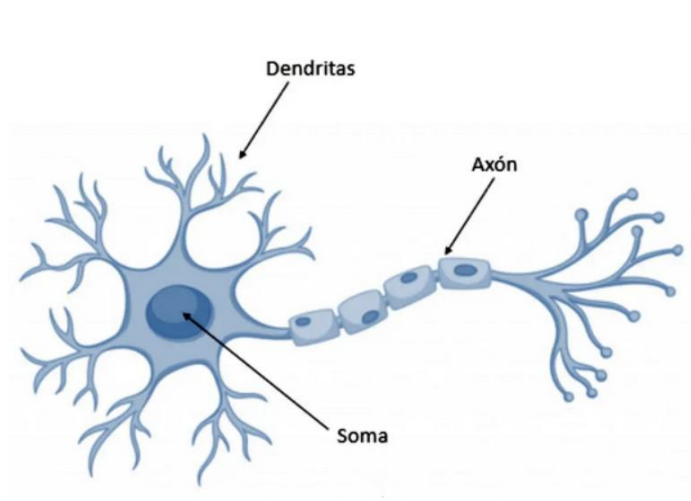
- Introducción
- Modelo de neurona artificial (general y estándar)
- Arquitectura de redes neuronales
- Modos de operación: recuerdo y aprendizaje
- Herramientas de software para implementación de redes neuronales

# Introducción biológica redes neuronales

- El cerebro es un ordenador (sistema de procesamiento de información) sumamente complejo, no lineal y paralelo.
- Organiza sus componentes estructurales, conocidos como **neuronas**, para realizar determinados cálculos
- Por ejemplo, reconocimiento de patrones, percepción y control motor

# Introducción biológica redes neuronales

- **Neurona biológica** es la base del funcionamiento del cerebro. Las neuronas constituyen procesadores de información sencillos



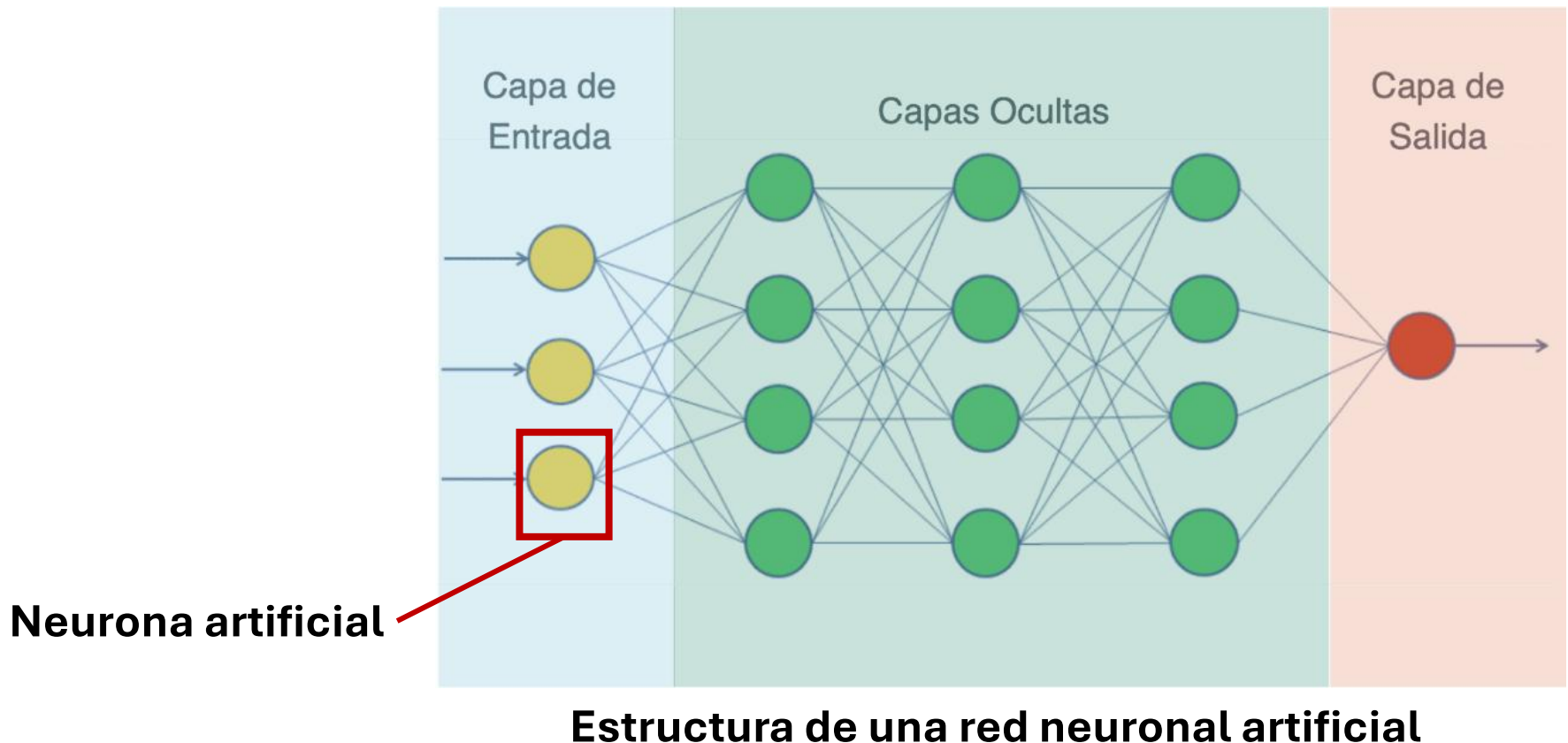
- **Elementos de que consta:** sinapsis, axón, dendritas y soma o cuerpo

# Introducción biológica redes neuronales

- **Aprendizaje** se produce a través de la comunicación entre neuronas. Las conexiones entre neuronas, que se fortalecen durante el aprendizaje, son la base de la memoria y la adquisición de nuevas habilidades
- El cerebro es altamente adaptable y puede cambiar su estructura en respuesta al aprendizaje e información que recibe (**neuroplasticidad**)
- Las señales nerviosas se pueden transmitir eléctrica o químicamente

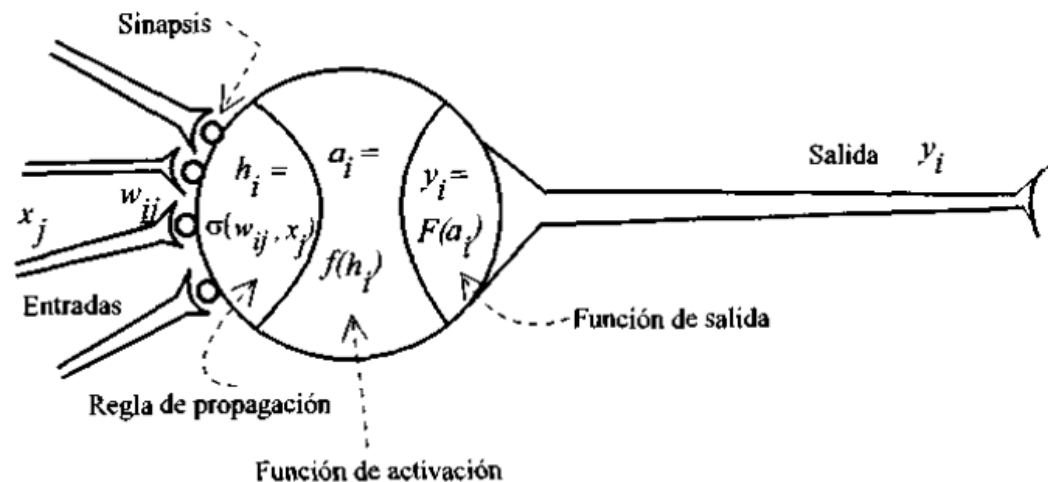
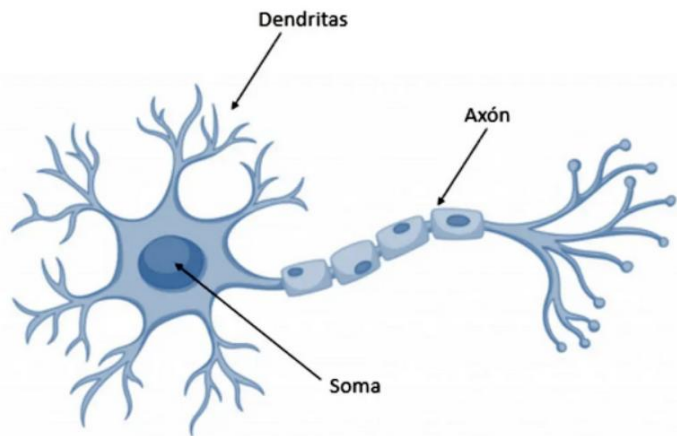
# Redes Neuronales Artificiales: Estructura

- Las redes neuronales artificiales “imitan” la estructura jerárquica de redes neuronales biológicas



# Modelo general de neurona artificial

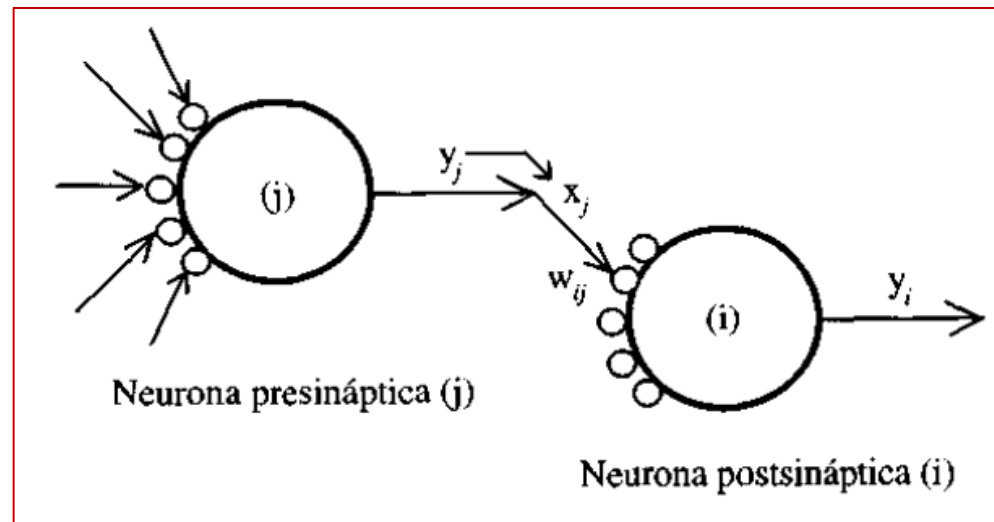
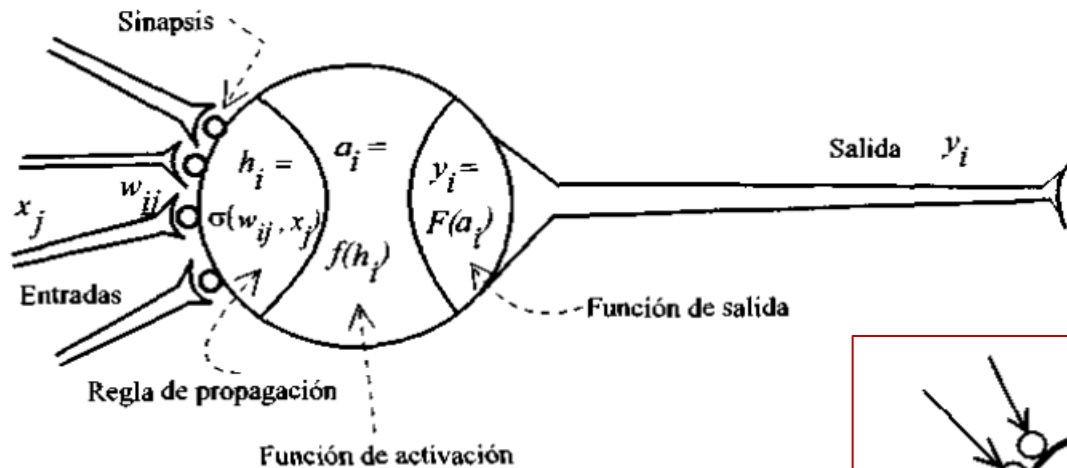
- Conjunto de entradas,  $x_j(t)$
- Pesos sinápticos de la neurona  $i$ ,  $w_{ij}$
- Regla de propagación  $h_i(t) = \sigma(w_{ij}, x_j(t))$
- Función de activación  $a_i(t) = f_i(a_i(t-1), h_i(t))$  que determina el estado de activación actual en función del estado anterior  $a_i(t-1)$
- Función de salida  $y_i(t) = F_i(a_i(t))$  que proporciona la salida actual de la neurona  $i$



# Modelo general de neurona artificial

- La operación de la neurona  $i$ , puede expresarse como:

$$y_i(t) = F_i(f_i[a_i(t-1), \sigma_i(w_{ij}, x_j(t))])$$





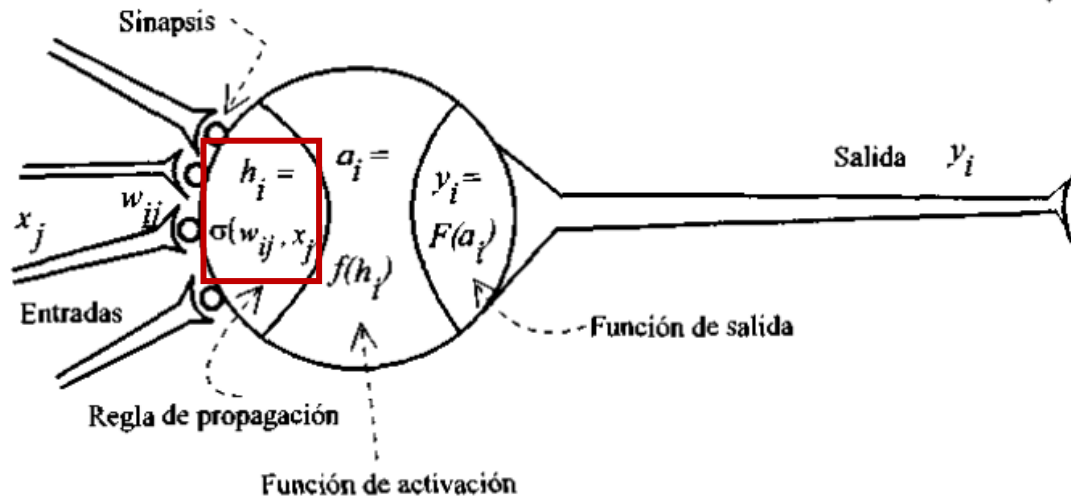
# Modelo general de neurona artificial

- **Regla de propagación**

- Permite obtener, a partir de las entradas y pesos, el valor del potencial  $h_i(t)$  de la neurona,  $h_i(t) = \sigma(w_{ij}, x_j(t))$

- La función más habitual es **lineal**  $h_i(t) = \sum_j w_{ij} x_j$

- Otra función habitual es la distancia euclidiana  $h_i^2(t) = \sum_j (x_j - w_{ij})^2$



# Modelo general de neurona artificial

- **Función de activación o de transferencia**

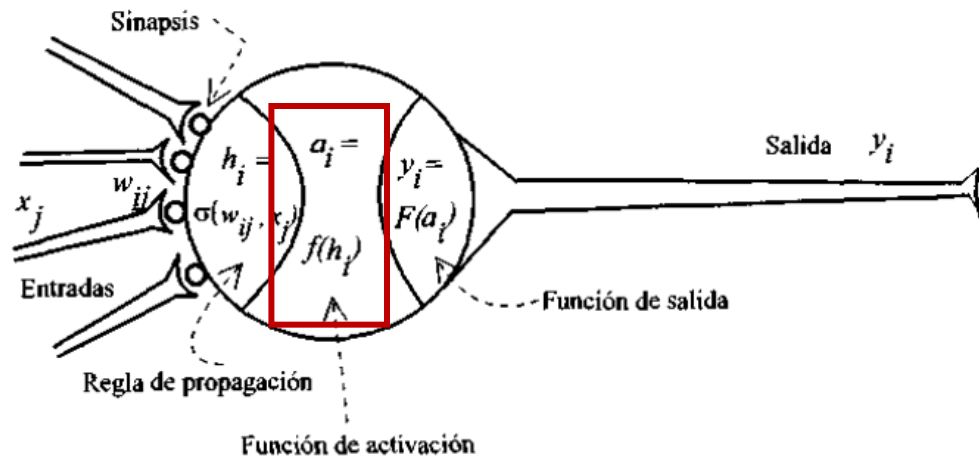
- Determina el estado de activación actual  $a_i(t)$  a partir del valor de propagación  $h_i(t)$  y de un estado de activación anterior  $a_i(t-1)$ :

$$a_i(t) = f_i(a_i(t-1), h_i(t))$$

- En muchos modelos de redes neuronales solo se considera el estado actual

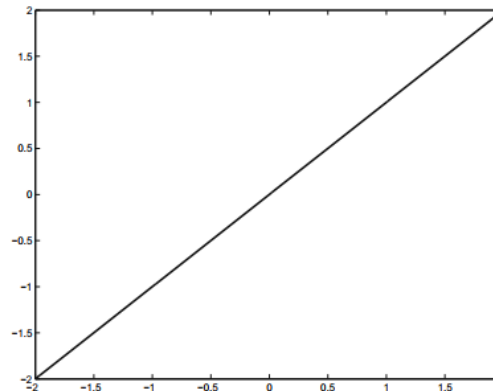
$$a_i(t) = f_i(h_i(t))$$

- Pueden ser **lineales** y **no lineales**



# Modelo general de neurona artificial

- Función de activación o de transferencia
  - **Función identidad (Adeline, regression lineal)**
    - $a(h) = h$
    - Esta función de activación lineal se usa si como salida se requiere una regresión lineal y de esta manera a la red neuronal que se le aplica la función va a generar un valor único
    - Rango de valores entre  $[-\infty, +\infty]$
    - Por ejemplo, se usa cuando se solicita predecir el valor de un número de ventas

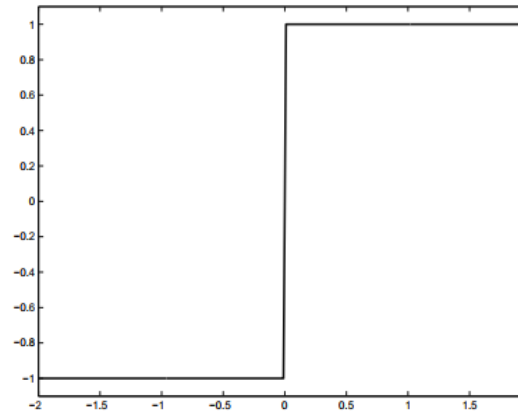


(a) Identity

# Modelo general de neurona artificial

- Función de activación o de transferencia
  - **Función umbral o escalón (Sign):**
    - Esta función se usa cuando se quiere clasificar o cuando se tiene salidas categóricas.
    - Por ejemplo, se puede usar para predecir si compro algo o no

$$a(h) = \begin{cases} 1 & \text{si } h \geq 0 \\ 0 & \text{si } h < 0 \end{cases}$$

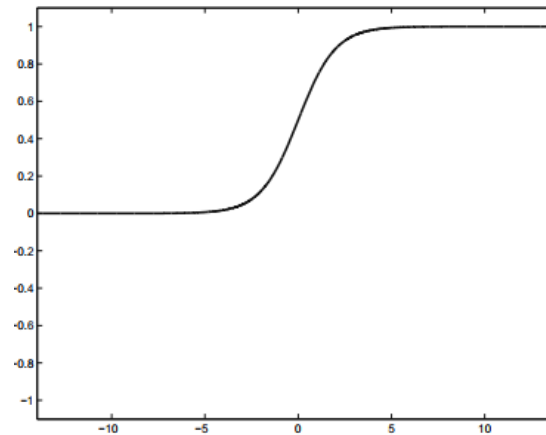


(b) Sign

# Modelo general de neurona artificial

- Función de activación o de transferencia
  - **Función Sigmoide (logística):**
    - Esta función se usa para clasificar datos en dos categorías (clasificación binaria) ya que transforma las entradas en probabilidades.
    - Rango de valores en el intervalo (0, 1)

$$a(h) = \frac{1}{1+e^{-h}}$$

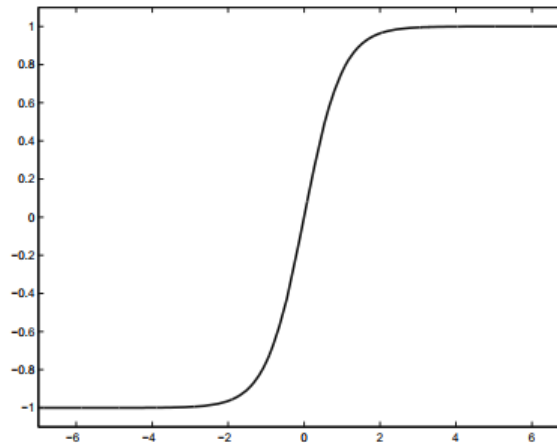


(c) Sigmoid

# Modelo general de neurona artificial

- Función de activación o de transferencia
- **Función tangente hiperbólica:**
  - Esta función puede manejar más fácilmente los números negativos
  - Se usa en las capas ocultas para introducir la no linealidad y normalizar los valores de entrada entre -1 y 1.

$$a(h) = \frac{2}{1+e^{-2h}} - 1$$

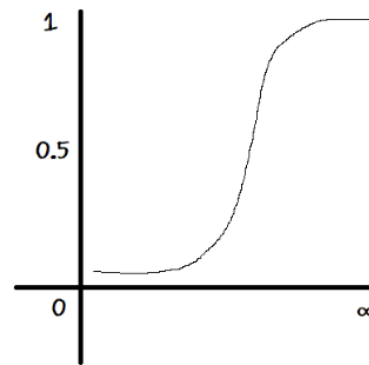


(d) Tanh

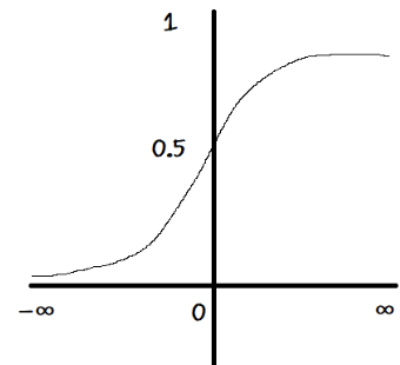
# Modelo general de neurona artificial

- Función de activación o de transferencia
- **Función softmax:**
  - Esta función transforma los valores de salida en probabilidades, estando cada valor comprendido entre 0 y 1 y siendo la suma de todas las salidas igual a 1
  - Esto permite determinar la clase a la que pertenece una entrada dada con un cierto grado de certeza

$$a(h) = \frac{e^h}{\sum_{j=0}^k e^{(h_i)}}$$



Sigmoid

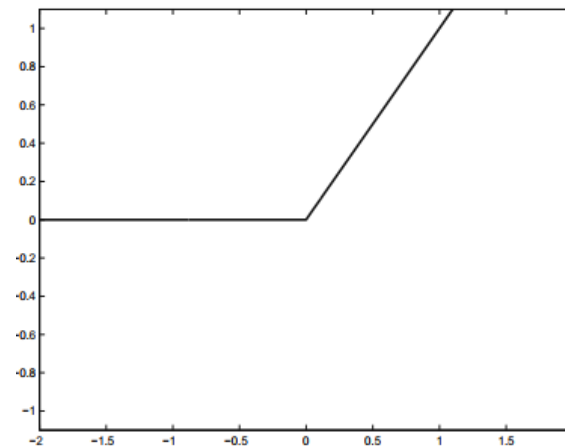


Softmax

# Modelo general de neurona artificial

- Función de activación o de transferencia
  - **Función ReLU o Rectificación lineal**
    - Esta función es computacionalmente eficiente porque implica un umbral en cero, introduciendo efectivamente la no linealidad en la red
    - Permite a las redes escalar a muchas capas sin un aumento significativo de la carga computacional

$$a(h) = \max(0, h)$$



(e) ReLU



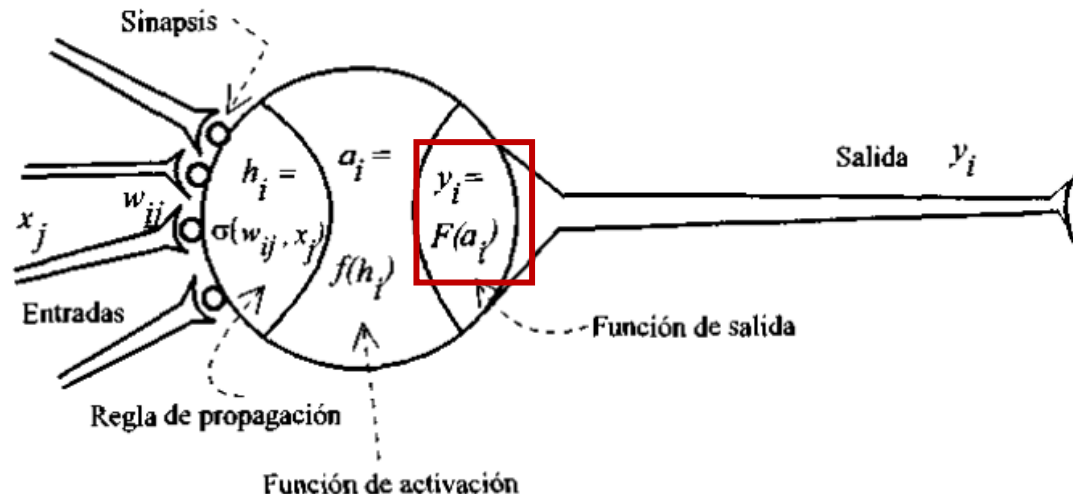
# Modelo general de neurona artificial

- **Función de salida**

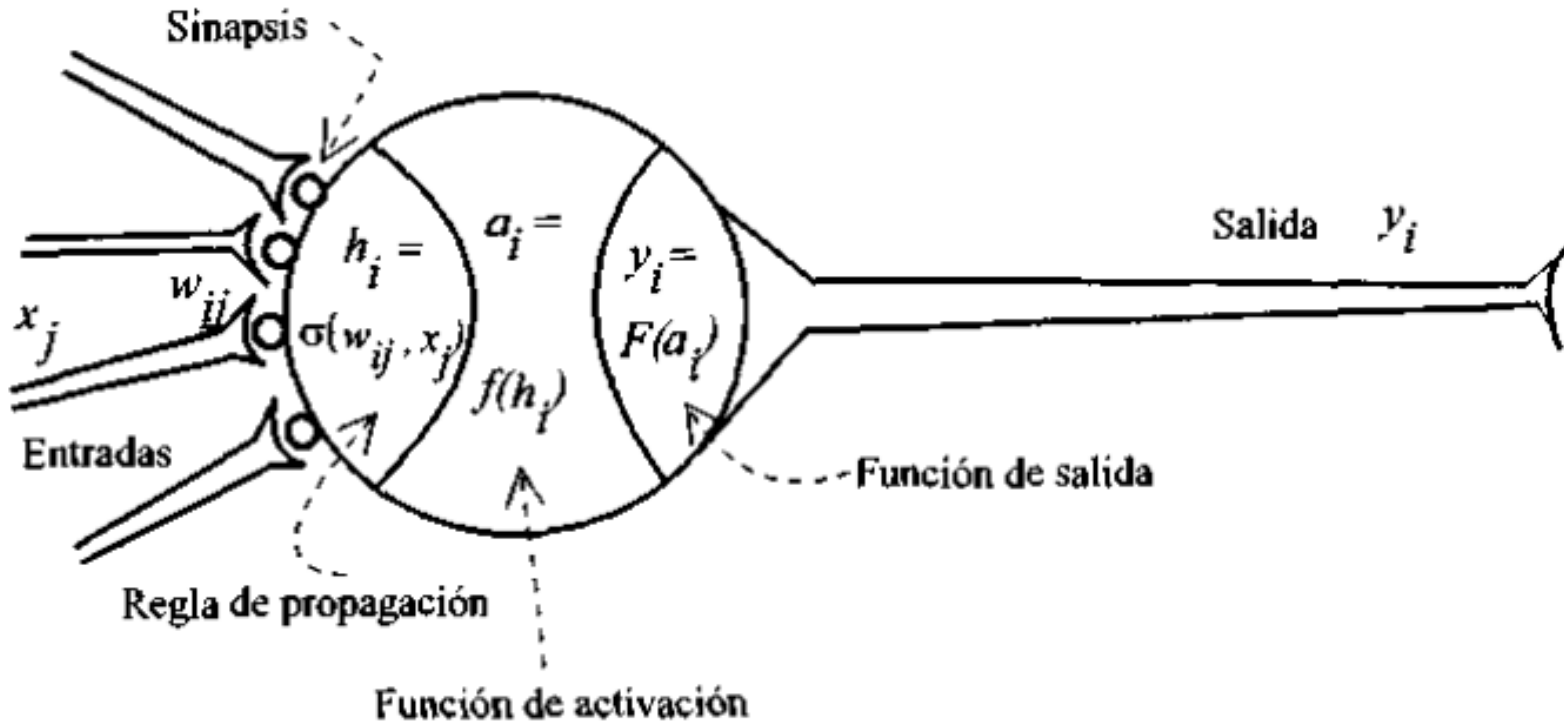
- Proporciona la salida global de la neurona en función del resultado de activación actual, como  $y_i(t) = F_i(a_i(t))$

- Se usa la función identidad  $F(x) = x$ , teniendo:

$$\mathbf{y_i(t) = F_i(a_i(t)) = a_i(t)}$$

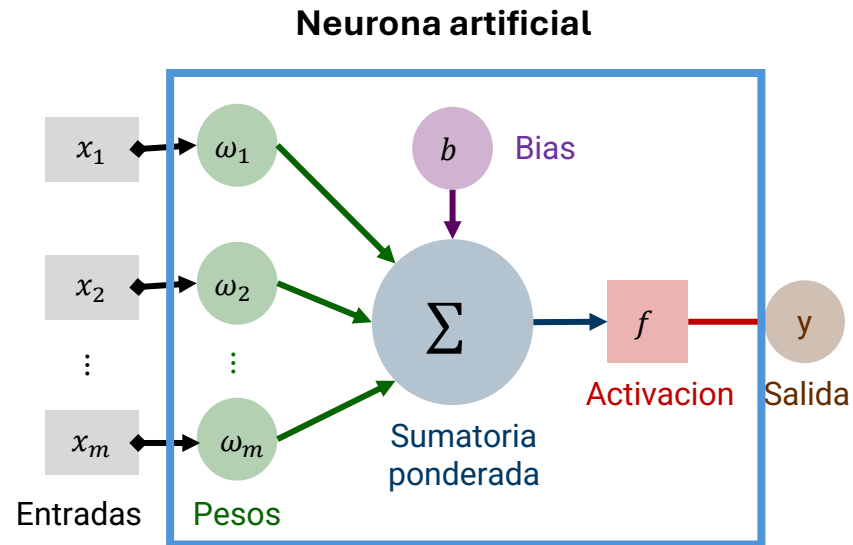
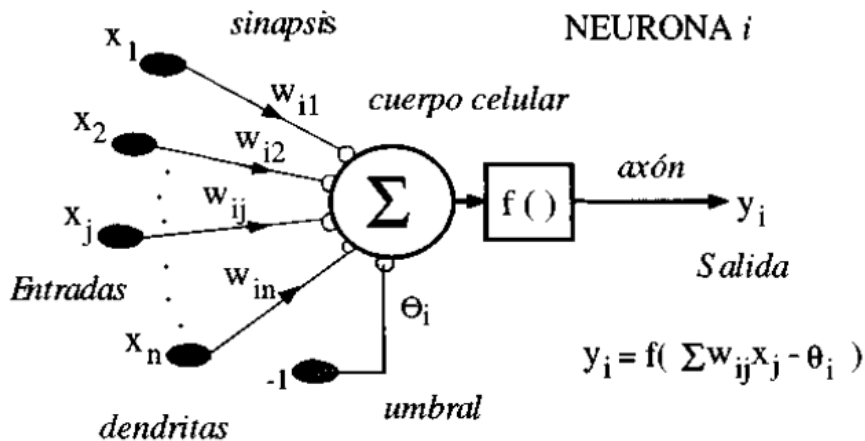


# Modelo general de neurona artificial



# Modelo estándar de neurona artificial

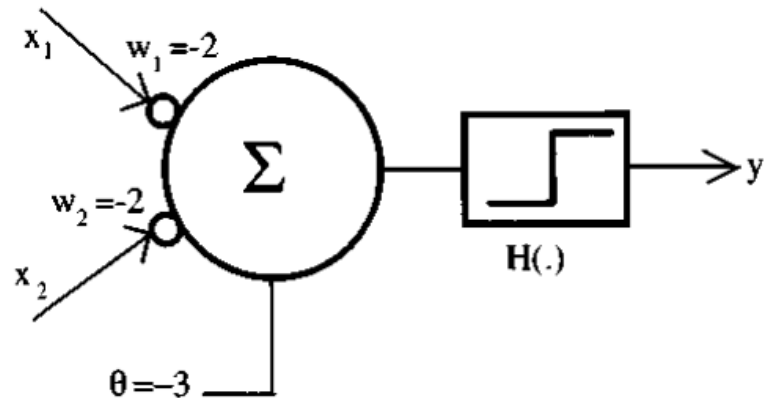
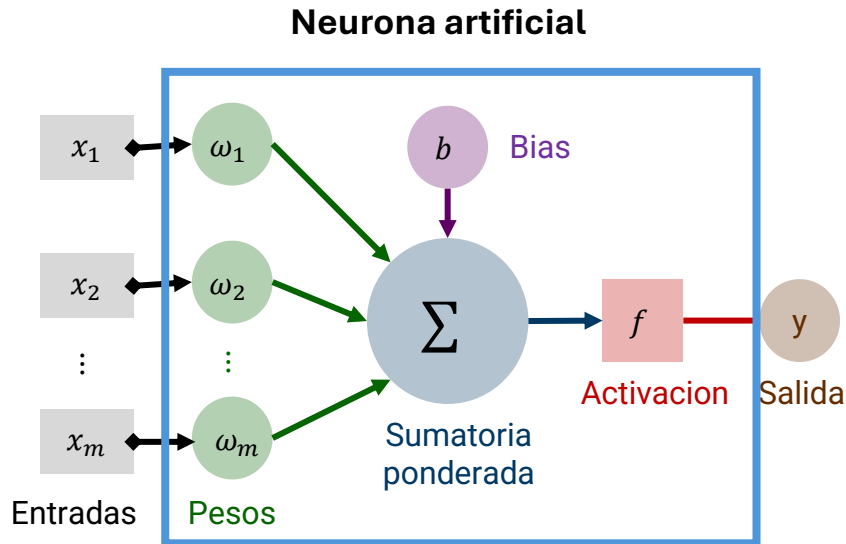
- Conjunto de entradas,  $x_j(t)$  y pesos sinápticos  $w_{ij}$
- Regla de propagación  $h_i(t) = \sigma(w_{ij}, x_j(t))$ ;  $h_i(t) = \sum_j w_{ij}x_j$
- Función de activación  $y_i(t) = f_i(h_i(t))$ , que representa tanto la salida de la neurona como su estado de activación
- **Umbral o bias:** peso que se introduce en la función activación y tiene el efecto de aumentar o disminuir la entrada neta de dicha función, dependiendo de su valor



# Modelo estándar de neurona artificial

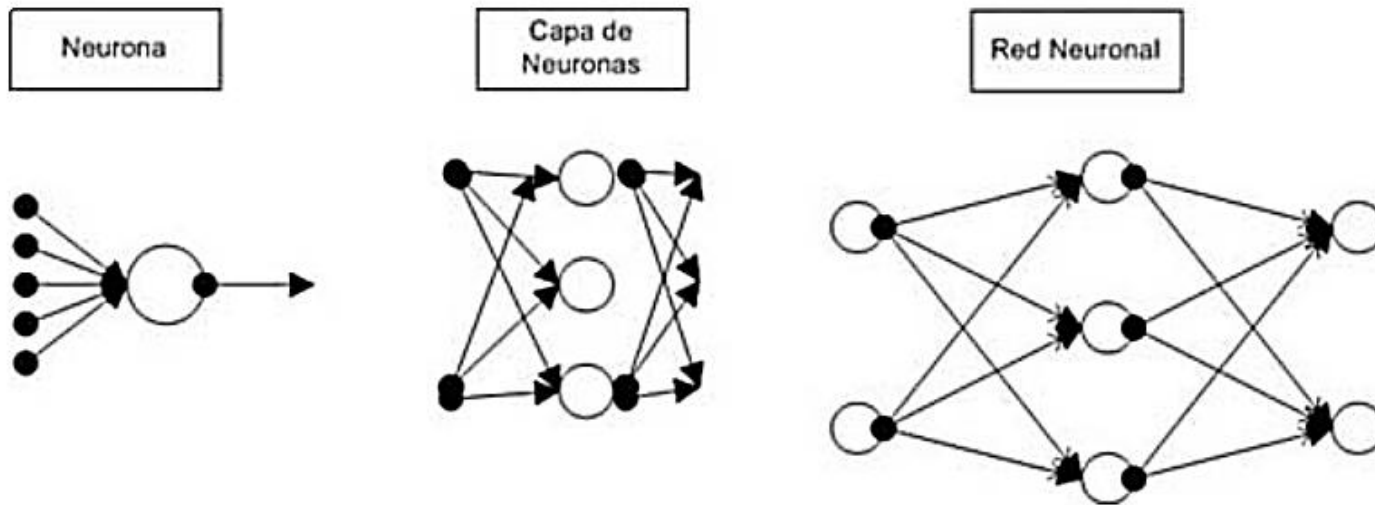
- **Neuronas todo-nada (umbral)**

- Entradas digitales, por ejemplo 0,1
- Función de **activación escalón**
- Modelo de la neurona de perceptrón original
- Este tipo de neuronas permiten representar cualquier función lógica
  - **Un nodo solo puede implementar funciones linealmente separables**



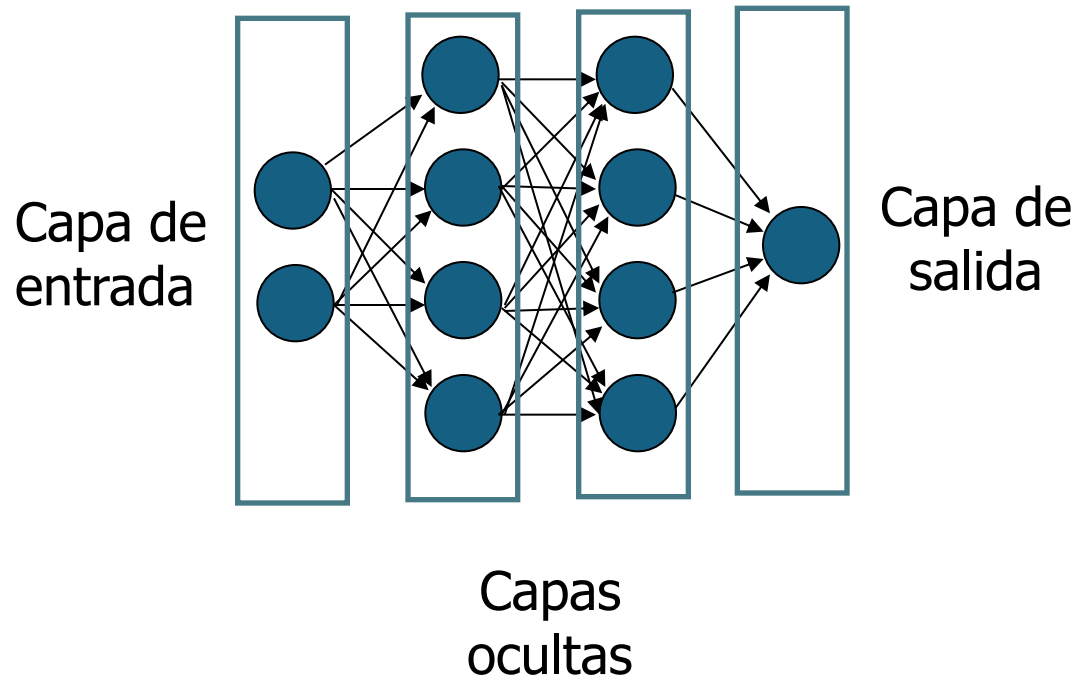
# Arquitectura de redes neuronales

- **Arquitectura:** de una red neuronal es su topología global más las funciones de activación utilizadas en cada neurona
- **Topología:** Manera en que se conectan las neuronas organizándose en capas



# Arquitectura de redes neuronales

- **Arquitectura:** de una red neuronal es su topología global más las funciones de activación utilizadas en cada neurona
- **Topología:** Manera en que se conectan las neuronas organizándose en capas



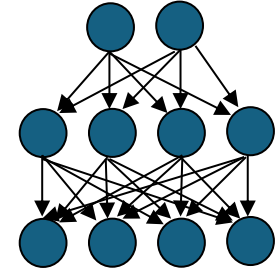
# Arquitectura de redes neuronales

- Características:
  - Cada capa puede tener un número de elementos (neuronas) distinto de las otras
  - Puede haber tantas capas como se desee
    - A mayor número de capas más complejo puede ser el problema a resolver
    - No siempre aumentar el número de capas mejora el resultado

# Arquitectura de redes neuronales

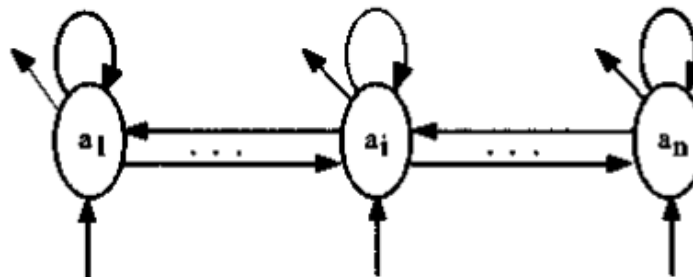
- Clasificación según el número de conexiones:

- Redes Totalmente Conectadas
- Redes Localmente Conectadas



- Clasificación según se transmite la información:

- **Redes *Feedforward* (unidireccional):** Las salidas de la capa  $i$  solo se conectan a las entradas de la capa  $i+1$
- **Redes *Feedbackward* (bidireccional):** : las salidas de una capa pueden ser entradas de capas anteriores
- **Redes Recurrentes:** puede haber lazos cerrados
- **Redes *Feedlateral*:** Las salidas de una capa pueden ser entradas de esa misma capa



Red monocapa  
recurrente



# Arquitectura de redes neuronales















- Características:
  - Redes *feedforward*:
    - Suelen ser más rápidas
    - Representan sistemas lineales
  - Redes *feedbackward* y *feedlateral*:
    - Son más lentas
    - Representan sistemas no lineales

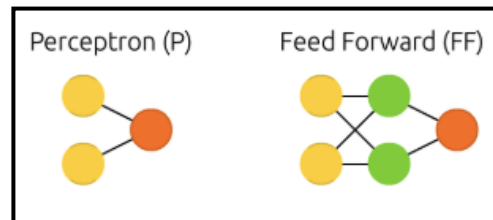
# Arquitectura de redes neuronales

<https://www.asimovinstitute.org/neural-network-zoo/>

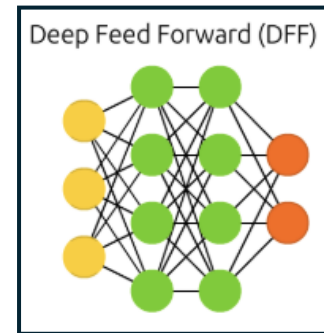
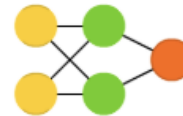
## A mostly complete chart of Neural Networks

©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org

-  Input Cell
-  Backfed Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Capsule Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Gated Memory Cell
-  Kernel
-  Convolution or Pool



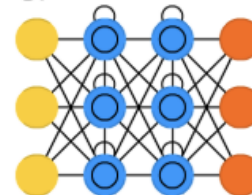
Radial Basis Network (RBF)



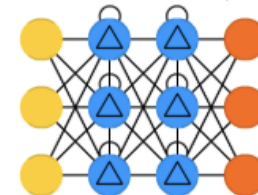
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)



Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)

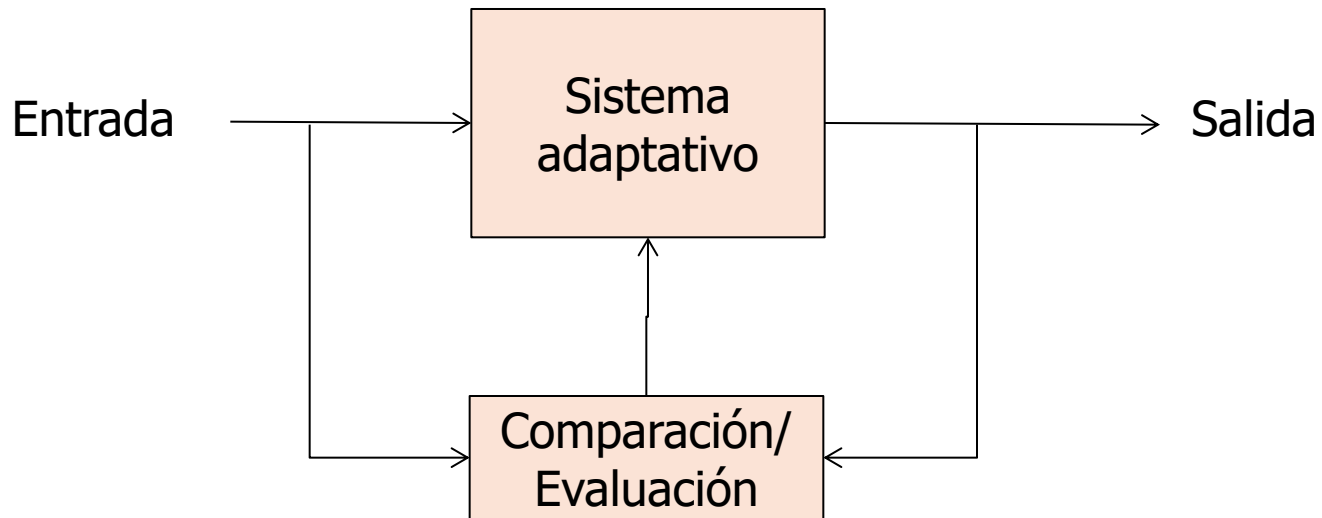


Sparse AE (SAE)



# Modos de operación: Recuerdo y Aprendizaje

- **Fase de aprendizaje o convergencia**
  - Proceso de **actualización de los pesos** de la red neuronal (y en algunos casos la arquitectura) con el fin de que la **red realice una tarea**
  - Este es un proceso **iterativo**

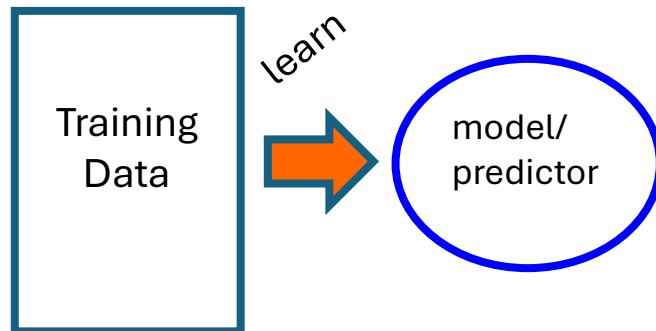


Estructura general de un sistema de aprendizaje

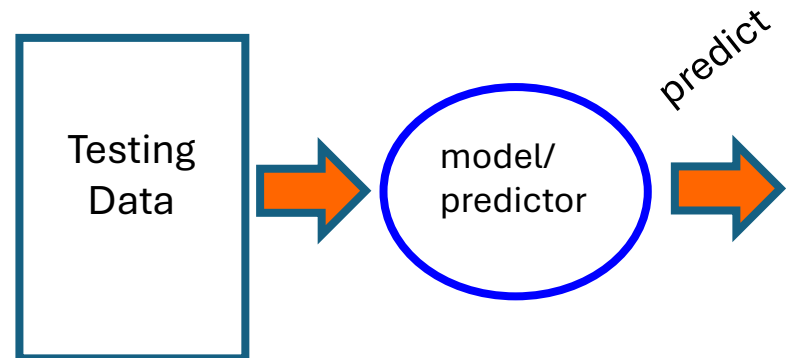
# Modos de operación: Recuerdo y Aprendizaje

- **Fase de aprendizaje o convergencia**
  - Proceso de **actualización de los pesos** de la red neuronal (y en algunos casos la arquitectura) con el fin de que la **red realice una tarea**
  - Este es un proceso **iterativo**

## Entrenamiento



## Evaluación



Estructura general de un sistema de aprendizaje

# Modos de operación: Recuerdo y Aprendizaje

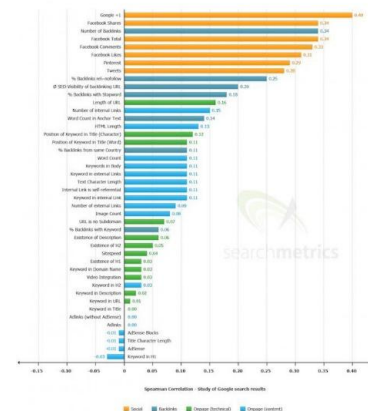
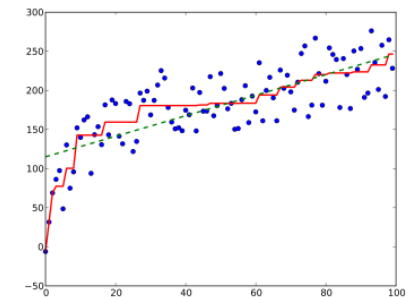
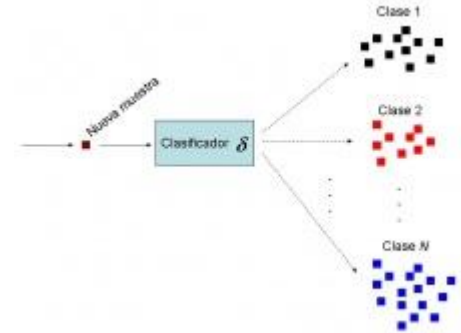
- **Fase de aprendizaje o convergencia**
  - Proceso de **actualización de los pesos** de la red neuronal (y en algunos casos la arquitectura) con el fin de que la **realice una tarea**
  - Este es un proceso **iterativo**
- **Conceptos clave:**
  - *Paradigma o tipo de aprendizaje*: información de la que dispone la red
  - *Regla de aprendizaje*: principios que gobiernan el aprendizaje, por ejemplo: optimización de la función de error
  - *Algoritmo de aprendizaje*: procedimiento o método numérico de ajuste de los pesos

# Tipos de tareas de aprendizaje

---

# Tipos de tareas de aprendizaje

- **Clasificación:** Predecir la clasificación sobre un conjunto de clases prefijadas
- **Regresión o predicción de valores:** Predecir un valor real
- **Ranking:** Predecir el orden óptimo de un conjunto de objetos según un orden de relevancia prefijado
- **Agrupamiento:** División de un conjunto de datos en grupos con propiedades comunes



# Paradigmas o tipos de aprendizaje





# Paradigmas o tipos de aprendizaje

- **Aprendizaje supervisado:** entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada (ejemplos)
  - **Supervisión:** Los datos (observaciones, medidas) son anotados con clases pre-definidas

# Paradigmas o tipos de aprendizaje

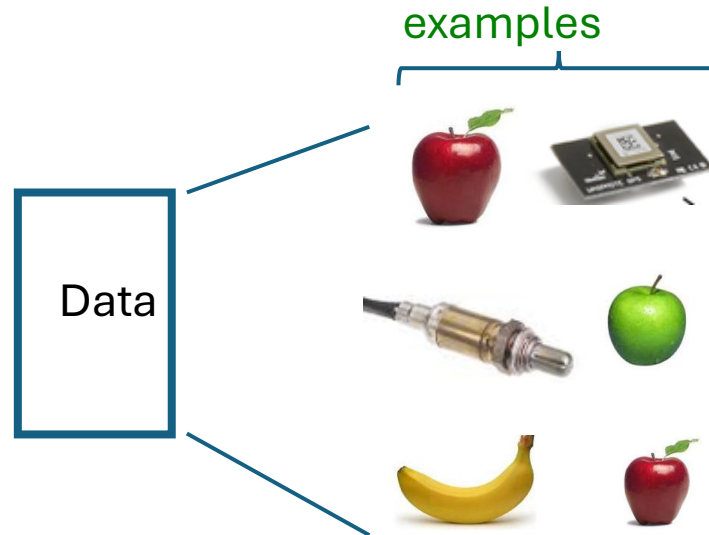
- **Aprendizaje supervisado:** entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada (ejemplos)
  - **Supervisión:** Los datos (observaciones, medidas) son anotados con clases pre-definidas
- **Aprendizaje no supervisado:** La red no recibe ninguna información que le indique si la salida generada en respuesta a una determinada entrada es o no correcta
  - Los datos NO se encuentran anotados

# Paradigmas o tipos de aprendizaje

- **Aprendizaje supervisado:** entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada (ejemplos)
  - **Supervisión:** Los datos (observaciones, medidas) son anotados con clases pre-definidas
- **Aprendizaje no supervisado:** La red no recibe ninguna información que le indique si la salida generada en respuesta a una determinada entrada es o no correcta
  - Los datos NO se encuentran anotados
- **Aprendizaje por refuerzo :** se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado, es decir, de no indicar durante el entrenamiento exactamente la salida que se desea que proporcione la red ante una determinada entrada.
  - **Supervisión:** Se indica con una señal de refuerzo si la salida obtenida en la red se ajusta a la deseada y en función de ello se ajustan los pesos

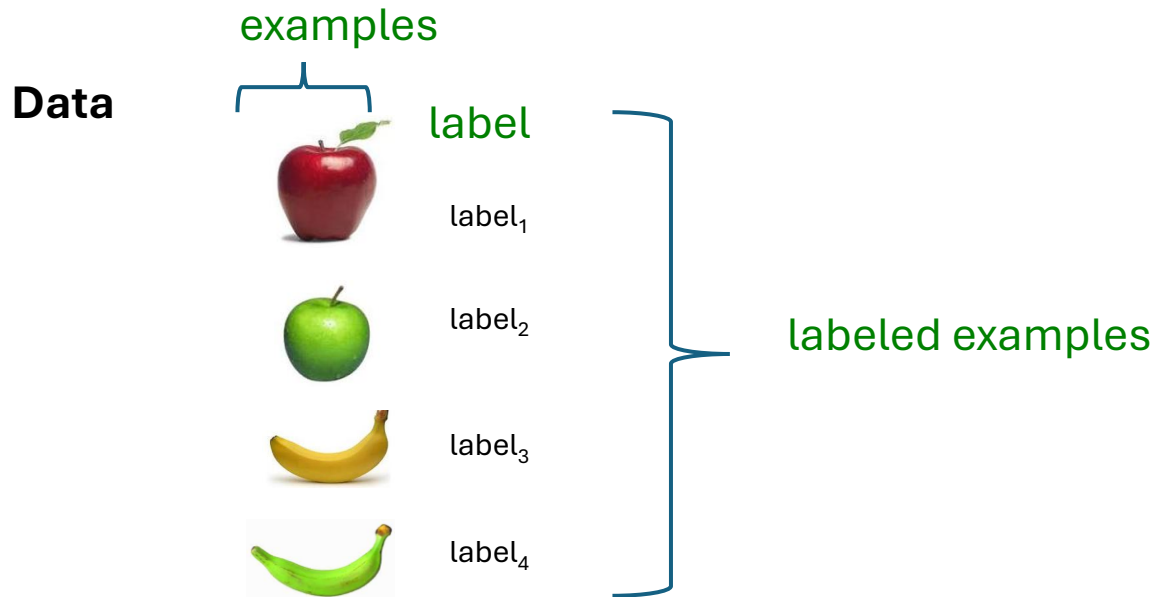
# Paradigmas o tipos de aprendizaje

- **Aprendizaje no supervisado:** Las etiquetas de los ejemplos son desconocidas



# Paradigmas o tipos de aprendizaje

- **Aprendizaje supervisado:** Se conocen las etiquetas de los ejemplos



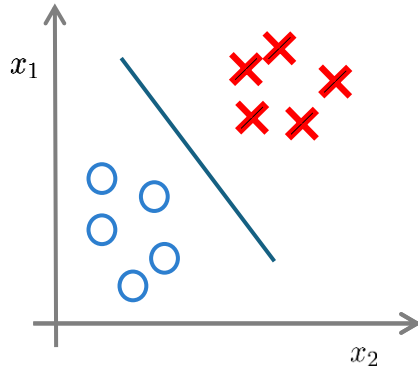
# Paradigmas o tipos de aprendizaje

- **Aprendizaje supervisado:** Se conocen las etiquetas de los ejemplos



# Paradigmas o tipos de aprendizaje

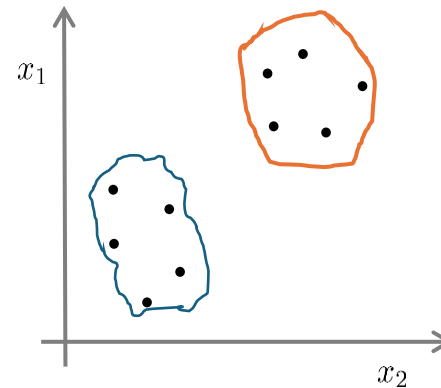
## Aprendizaje Supervizado vs Aprendizaje NO Supervizado



Dataset:

WITH labels

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots, (x^{(m)}, y^{(m)})\}$$



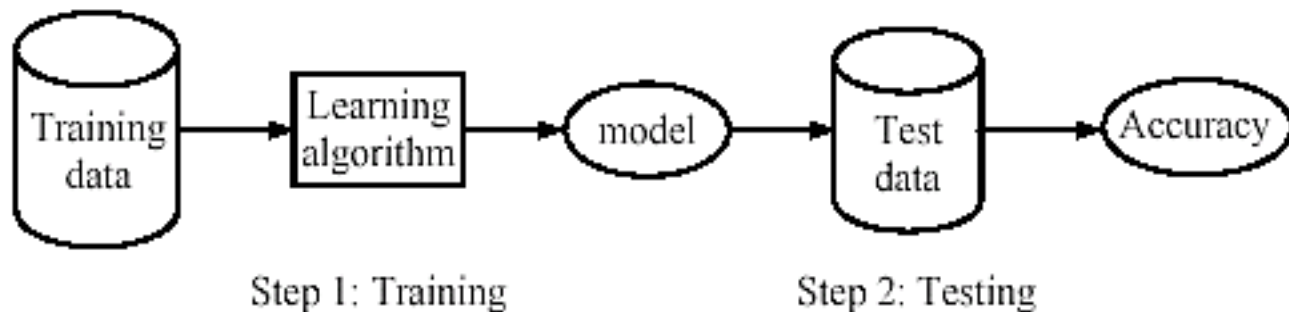
Dataset:

WITHOUT  
labels

$$\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)}\}$$

# Modos de operación: Recuerdo y Aprendizaje

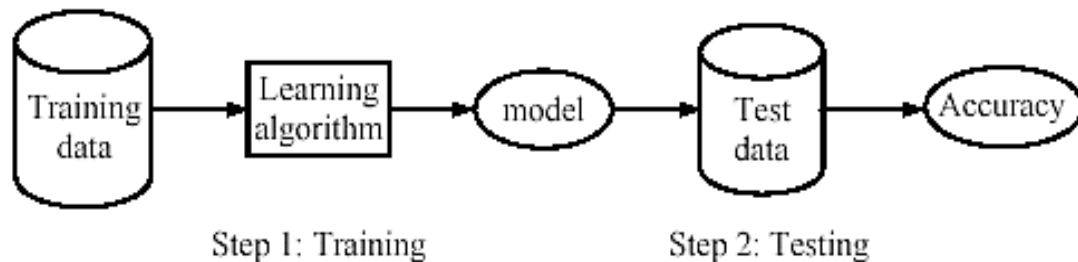
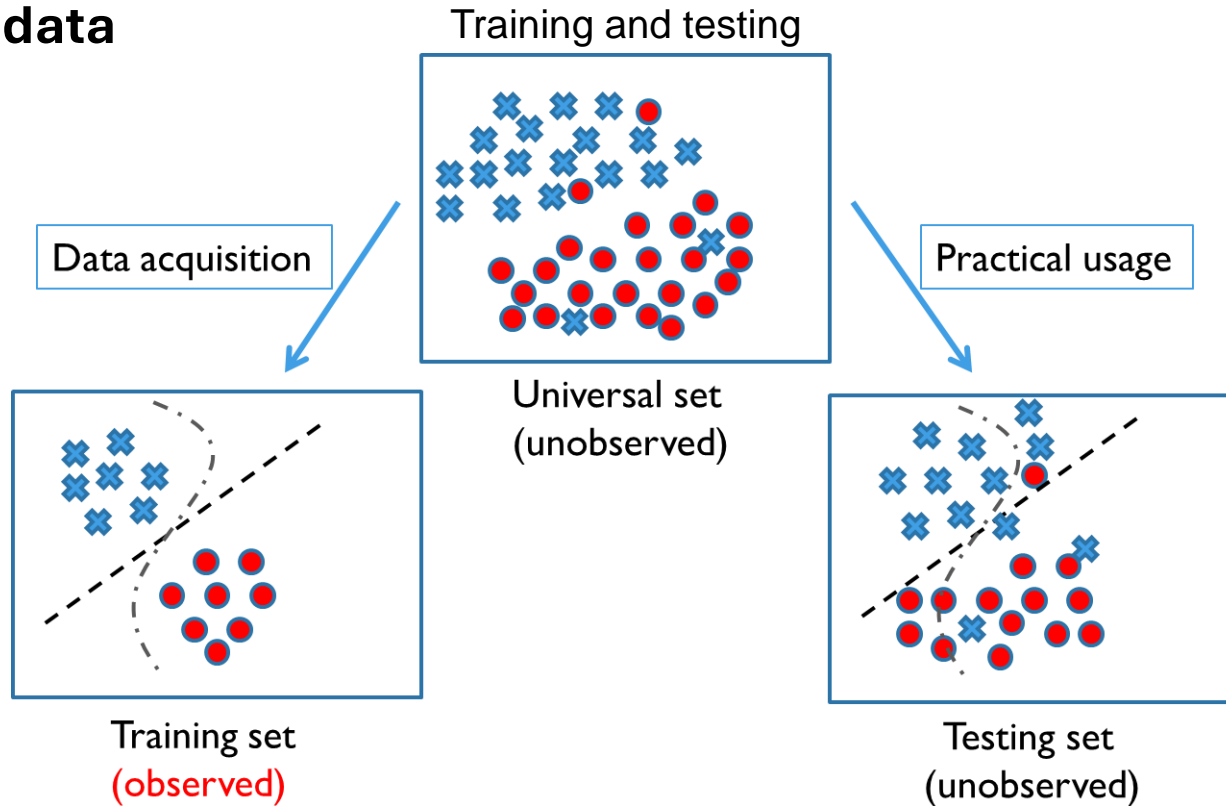
- El proceso de entrenamiento esta dividido en dos pasos, que tiene asociado un tipo de error:
  - **Etapas de entrenamiento:** Al final de la fase de aprendizaje el error obtenido se calcula para el conjunto de datos de entrenamiento (Training set)
  - **Etapas de pruebas:** error obtenido con la red ya entrenada al ser evaluada con patrones (datos) no utilizados en el entrenamiento (Testing set). Esto **determina la capacidad de generalización de la red**





# Modos de operación: Recuerdo y Aprendizaje

## Splitting data

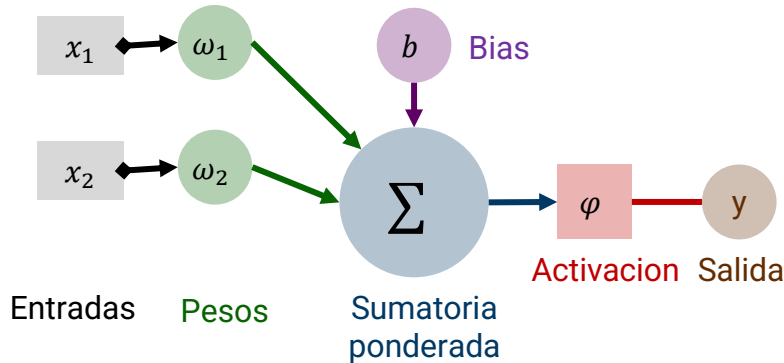


# Modos de operación: **Recuerdo y Aprendizaje**

- **Fase de recuerdo o ejecución (estabilidad)**
  - Una vez que el sistema ha sido entrenado, los pesos y la estructura de la red neuronal quedan fijos, estando la red lista para procesar datos
  - En las redes unidireccionales, ante un patrón de entrada, las neuronas responden proporcionando directamente la salida del sistema

# Ejercicio

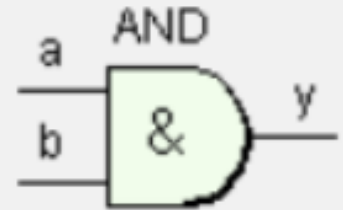
- Dada la siguiente neurona artificial verifique que permite calcular la función AND



## Función de activación

$$y = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

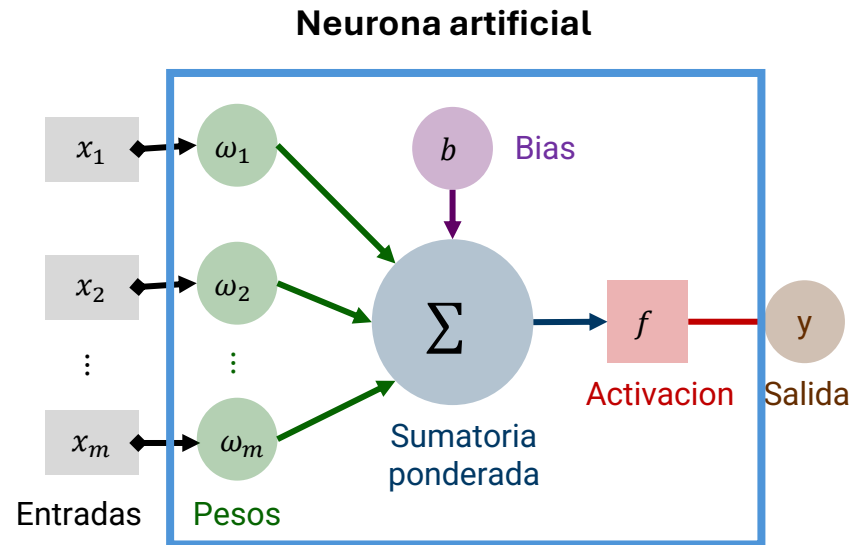
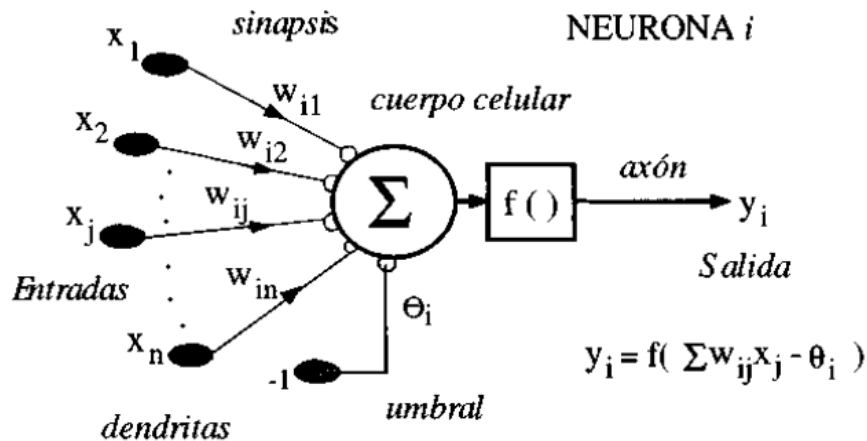
- Entradas:**  $x_1$  y  $x_2$
- Bias:**  $b = -1$
- Pesos**
  - $W_1 = 1$
  - $W_2 = 1$



a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

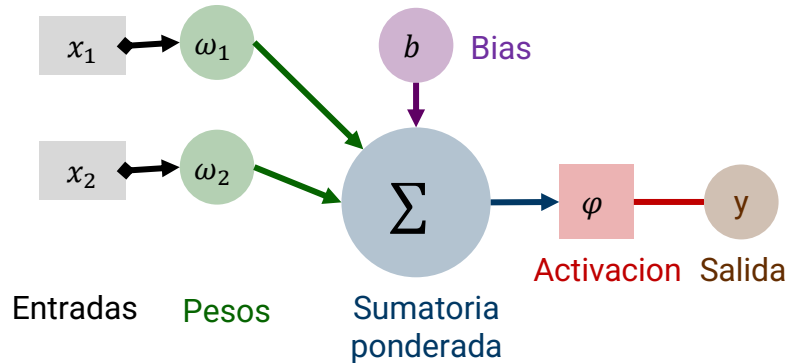
# Modelo estándar de neurona artificial

- Conjunto de entradas,  $\mathbf{x}_j(\mathbf{t})$  y pesos sinápticos  $\mathbf{w}_{ij}$
- Regla de propagación  $\mathbf{h}_i(\mathbf{t}) = \sigma(\mathbf{w}_{ij}, \mathbf{x}_j(\mathbf{t}))$ ;  $h_i(t) = \sum_j w_{ij}x_j$
- Función de activación  $\mathbf{y}_i(\mathbf{t}) = \mathbf{f}_i(\mathbf{h}_i(\mathbf{t}))$ , que representa tanto la salida de la neurona como su estado de activación
- Umbral o bias: peso que se introduce en la función activación y tiene el efecto de aumentar o disminuir la entrada neta de dicha función, dependiendo de su valor



# Ejercicio

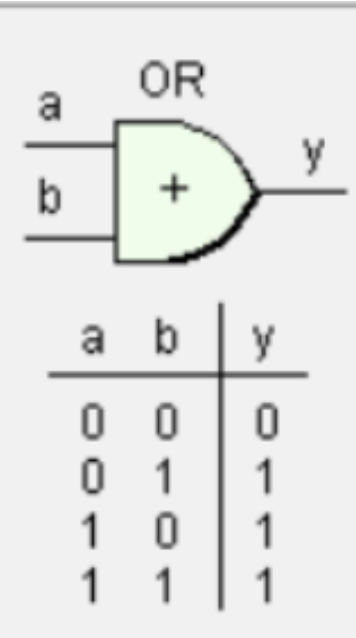
- Determine los parámetros necesarios para que la siguiente neurona calcule la función OR



## Función de activación

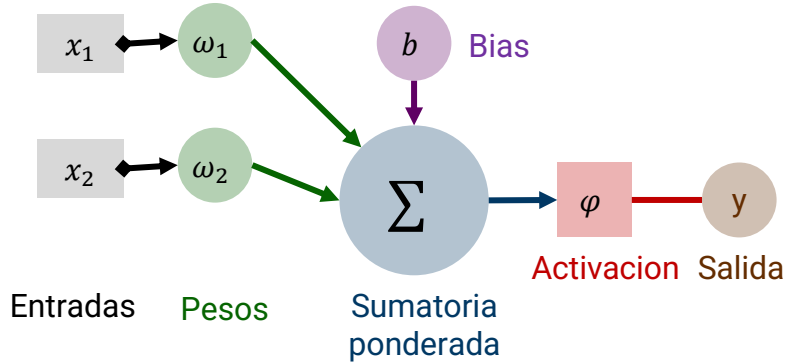
$$y = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

- Entradas:**  $x_1$  y  $x_2$
- Bias:**  $b = ?$
- Pesos**
  - $W_1 = ?$
  - $W_2 = ?$



# Ejercicio

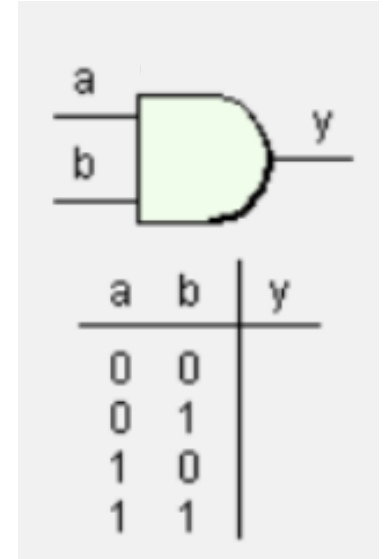
- Qué función booleana permite calcular el perceptrón con los siguientes parámetros?



## Función de activación

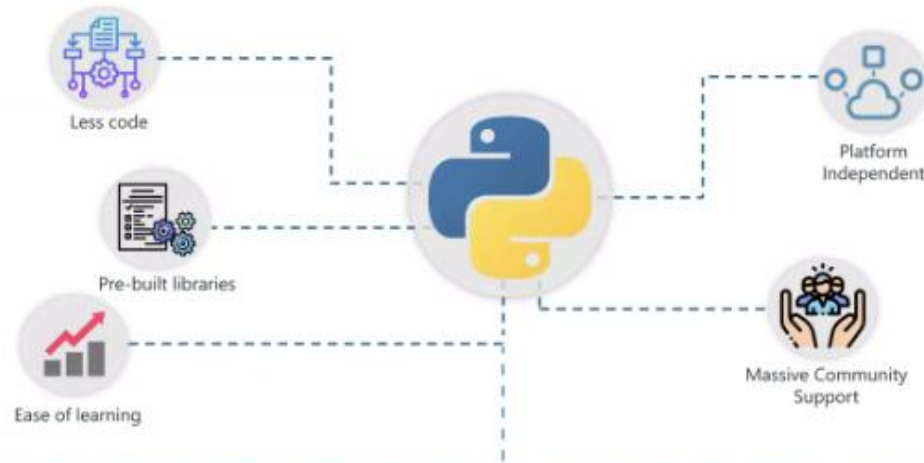
$$y = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

- **Entradas:**  $x_1$  y  $x_2$
- **Bias:**  $b = -1$
- **Pesos**
  - $W_1 = 1.5$
  - $W_2 = 1.5$



# Herramientas para implementar redes neuronales

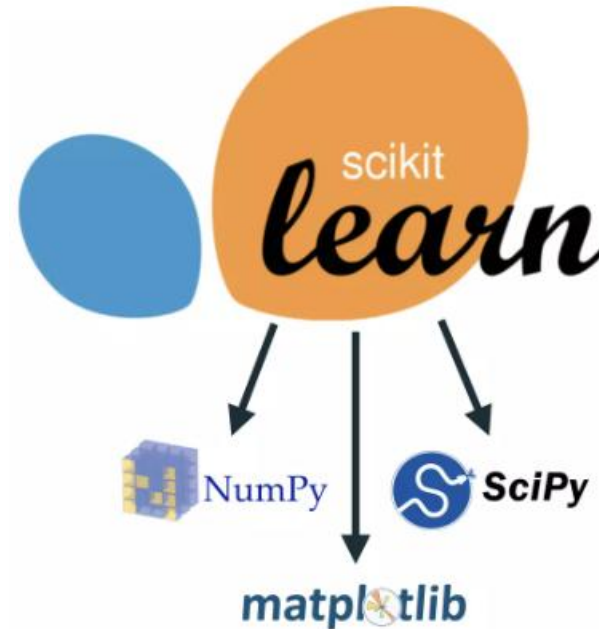
## ML and DL



# Herramientas para implementar redes neuronales

## Scikit-Learn

- Una de las herramientas más usadas para el entrenamiento de modelos de **machine learning** (supervisado y no supervisado).
- Extensión de SciPy con los algoritmos de ML más usados (SciKit). Creado sobre NumPy y SciPy, y algunos paquetes de C.
- Herramienta de código abierto.
- Probablemente la mejor herramienta para ML en la actualidad.
- Para clasificación, regresión, agrupación...
- Modelo lineal generalizado, SVM, kNN, árboles de decision, Naive Bayes.



<https://scikit-learn.org/1.4/tutorial/index.html>



# Herramientas para implementar redes neuronales

## TensorFlow

- Unas de las herramientas más usadas para el entrenamiento de modelos de **deep learning** usando redes neuronales.
- Escrito en C++ y Python, desarrollado por el **Google Brain** en 2015.
- Herramienta de código abierto que provee funciones para trabajar con tensores y grafos.
- Actualmente en su versión 2.0
- TensorFlow.js para correr modelos en el browser



## TensorFlow

<https://www.tensorflow.org/tutorials/quickstart/beginner>  
<https://www.geeksforgeeks.org/tensorflow/>

# Herramientas para implementar redes neuronales

## Keras

- Keras es una API de aprendizaje profundo escrita en Python, que se ejecuta sobre TensorFlow, pero también puede trabajar con otros backends (Theano, CNTK).
- Keras es la API de alto nivel con una interfaz accesible y altamente productiva
- Proporciona abstracciones y bloques de construcción esenciales para desarrollar soluciones de aprendizaje automático
- Podemos pensar en Keras como un conjunto de abstracciones que facilitan construir modelos de **deep learning**



<https://codificandobits.com/tutorial/keras-desde-cero/>  
<https://keras.io/examples/>

# Herramientas para implementar redes neuronales

## PyTorch

- Un contendiente fuerte para las otras herramientas de **deep learning** basado en Torch
- Herramienta de código abierto desarrollada por el departamento de investigación en AI de **Facebook**
- Entre los ejemplos más importantes de software que usa PyTorch se encuentra el “Tesla Autopilot”.



<https://pytorch.org/tutorials/beginner/basics/intro.html>

# Herramientas para implementar redes neuronales



vs



vs  PyTorch

	Keras	TensorFlow	PyTorch
API level	High	High and low	Low
Architecture	Simple, concise, readable	Complex, less readable	Not easy to use
Datasets	Smaller datasets	Large datasets, high performance	Large datasets, high performance
Debugging	Simple network, so debugging is not often needed	Difficult to conduct debugging	Good debugging capabilities
Does it have trained models?	Yes	Yes	Yes
Popularity	Most popular	Second most popular	Third most popular
Speed	Slow, low performance	Fast, high performance	Fast, high performance
Language	Python	C++, CUDA, Python	Lua

# Lecturas Complementarias

---

Martín del Brío, B. & Sanz Molina, A. Redes neuronales y sistemas borrosos,

- Capítulo 1: Fundamentos de las neuronales artificiales