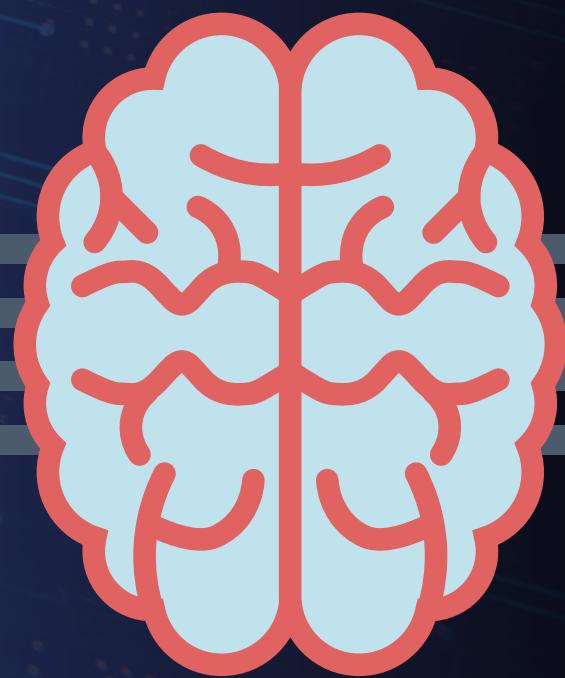


Universidad  
Rafael Landívar

# Redes Neuronales **Artificiales**

*Inteligencia Artificial*





# Agenda

## *Sesión 1: Perceptrón*

- *Historia*
- *Estructura*
- *Algoritmo*
- *Ejemplos y Ejercicios*
- *Limitaciones*

Sesión 2: Perceptrón Multiclasa y Multicapa

Sesión 3: Redes Neuronales



# Perceptrón

*En el campo del aprendizaje automático, el perceptrón es un algoritmo utilizado para el aprendizaje supervisado de clasificadores binarios, es decir, funciones que determinan si una entrada (representada como un vector de números) pertenece a una clase u otra. Se trata de un clasificador lineal, lo que significa que realiza sus predicciones basándose en una función lineal que combina un conjunto de pesos con las características de entrada.*



**El algoritmo del perceptrón se remonta a finales de la década de 1950. Su primera implementación se realizó en hardware especializado, y fue una de las primeras redes neuronales artificiales construidas en la historia.**



Perceptron

# Breve Introducción Histórica



BIOINSPIRACION

*El perceptrón fue propuesto por Frank Rosenblatt en 1958, inspirado por el funcionamiento de las neuronas biológicas.*



DESARROLLO

*Desarrollado durante la Guerra Fría con financiamiento militar (US Office of Naval Research).*



PRIMER INVIERNO DE LA IA

*El perceptrón fue criticado duramente por Minsky y Papert en 1969, lo que detuvo el progreso de las redes neuronales por décadas.*



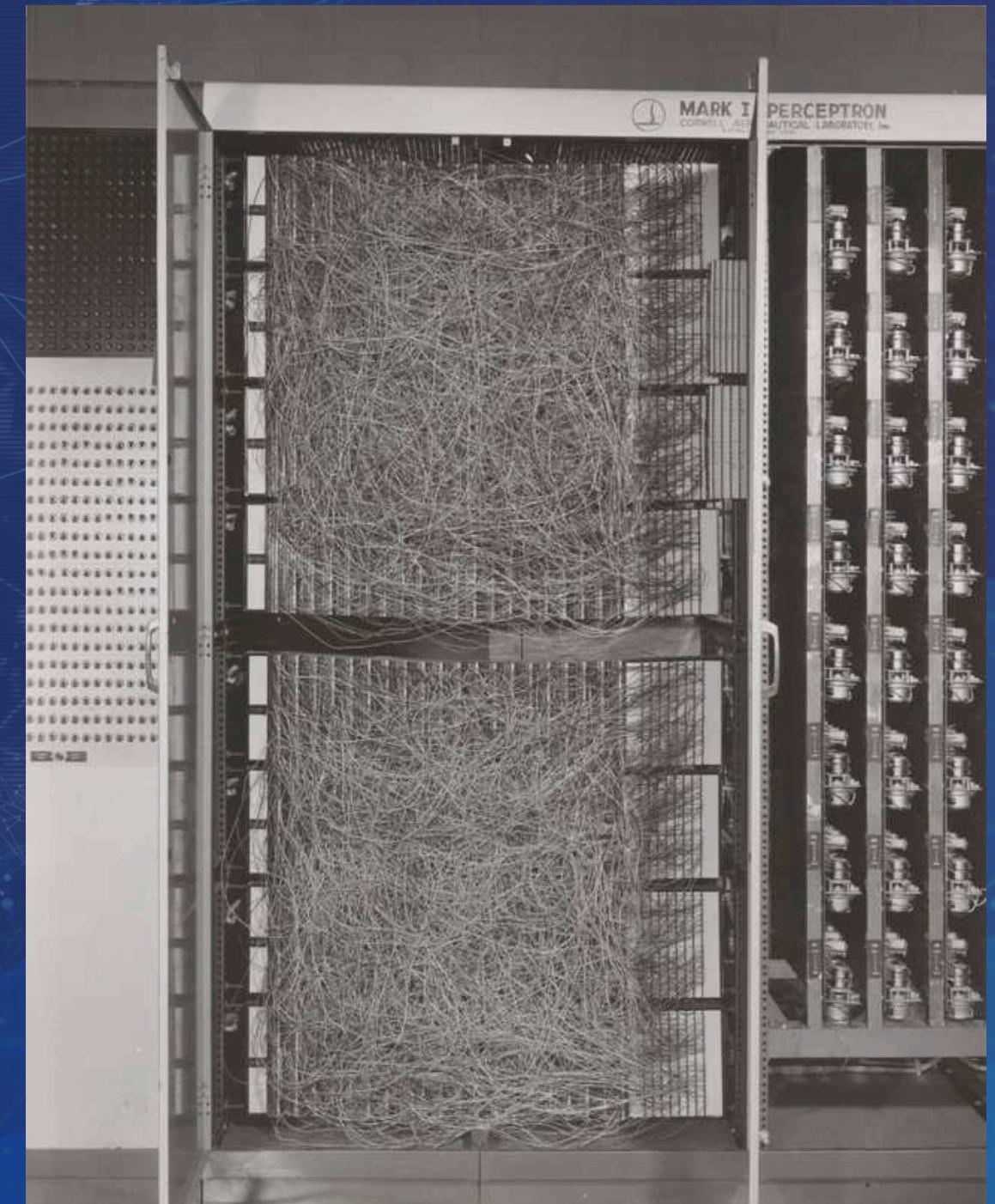
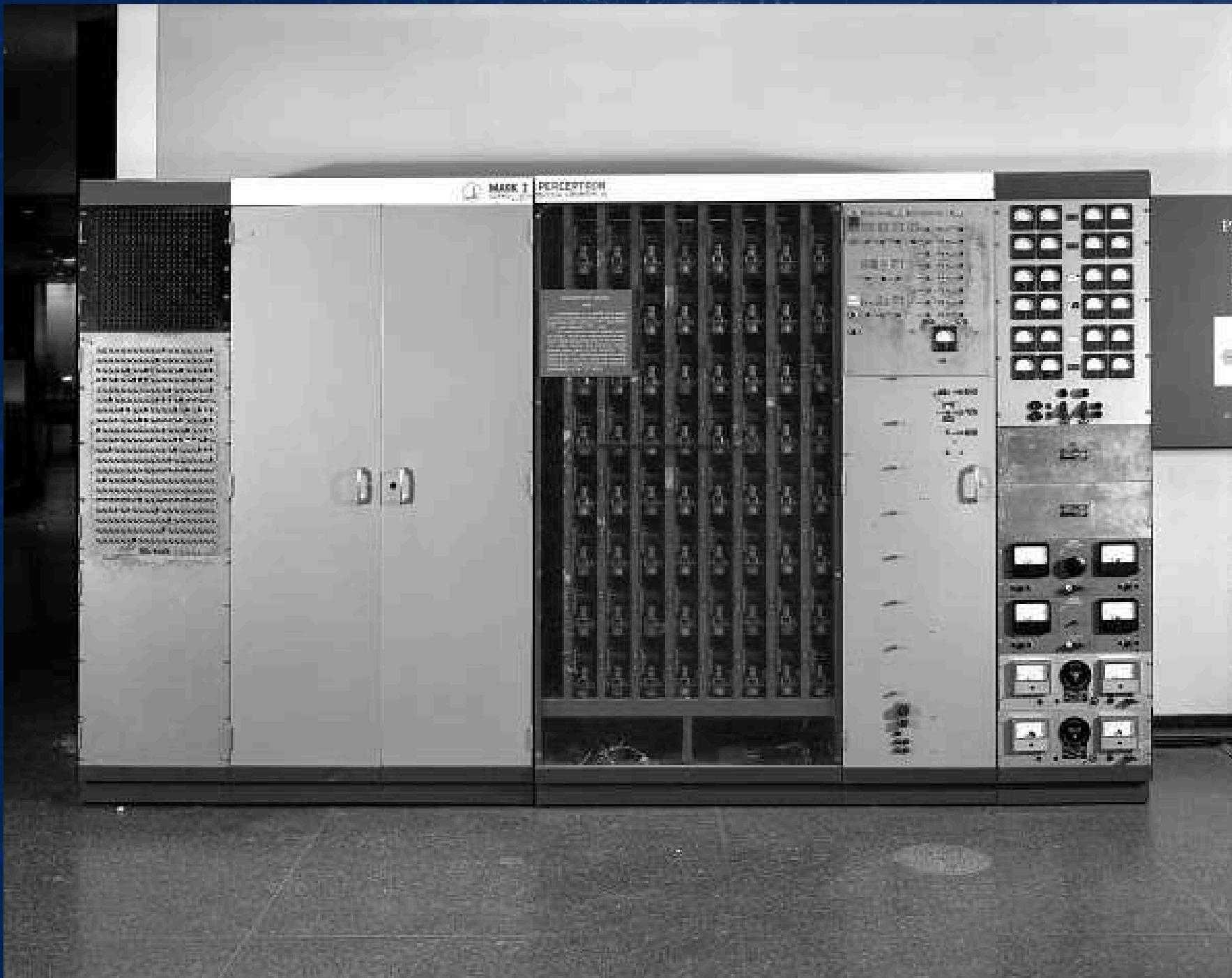
**Video unavailable**  
[Watch on YouTube](#)





Perceptron

# Breve Introducción Histórica





Perceptron

# Breve Introducción Histórica

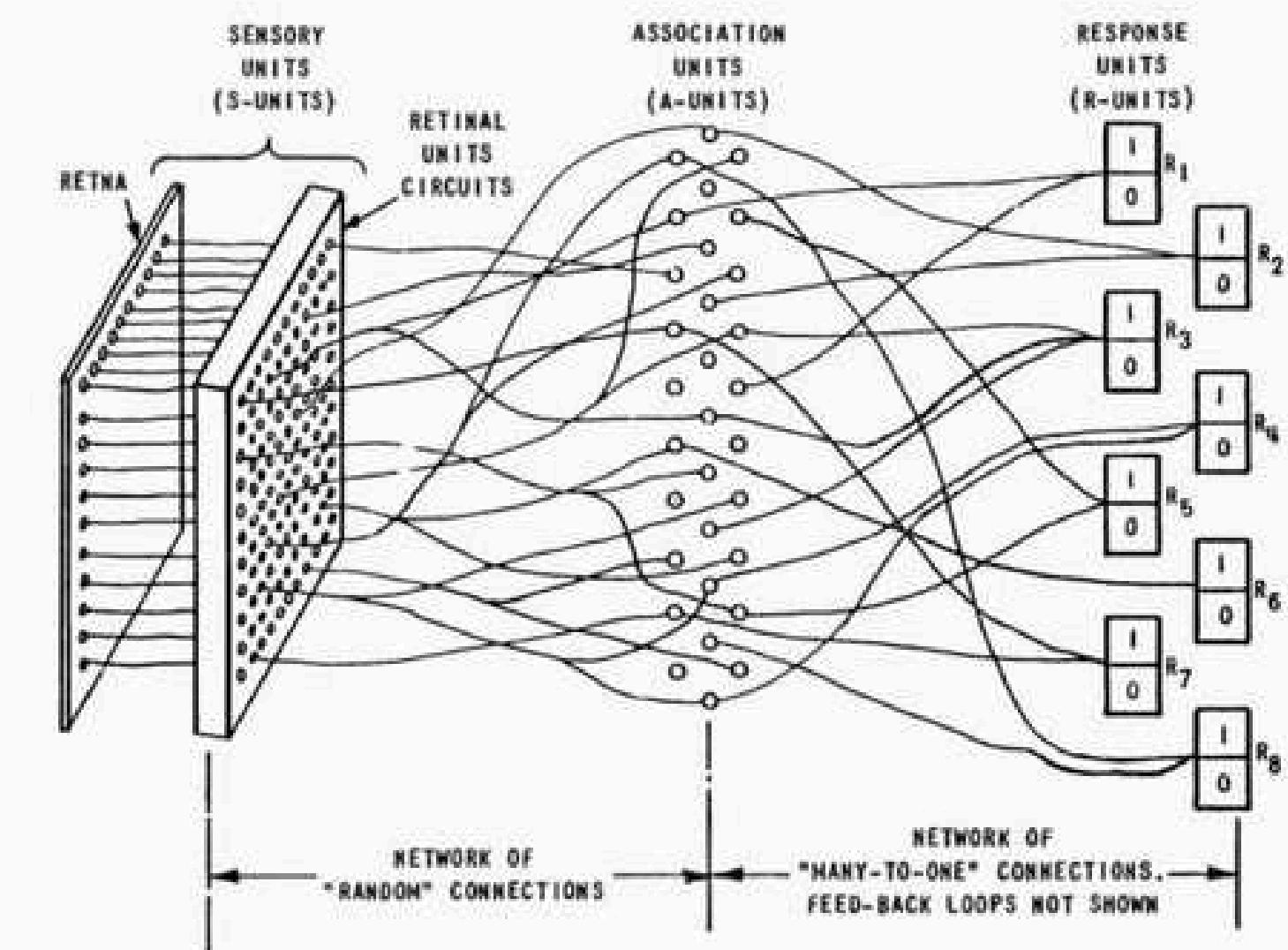
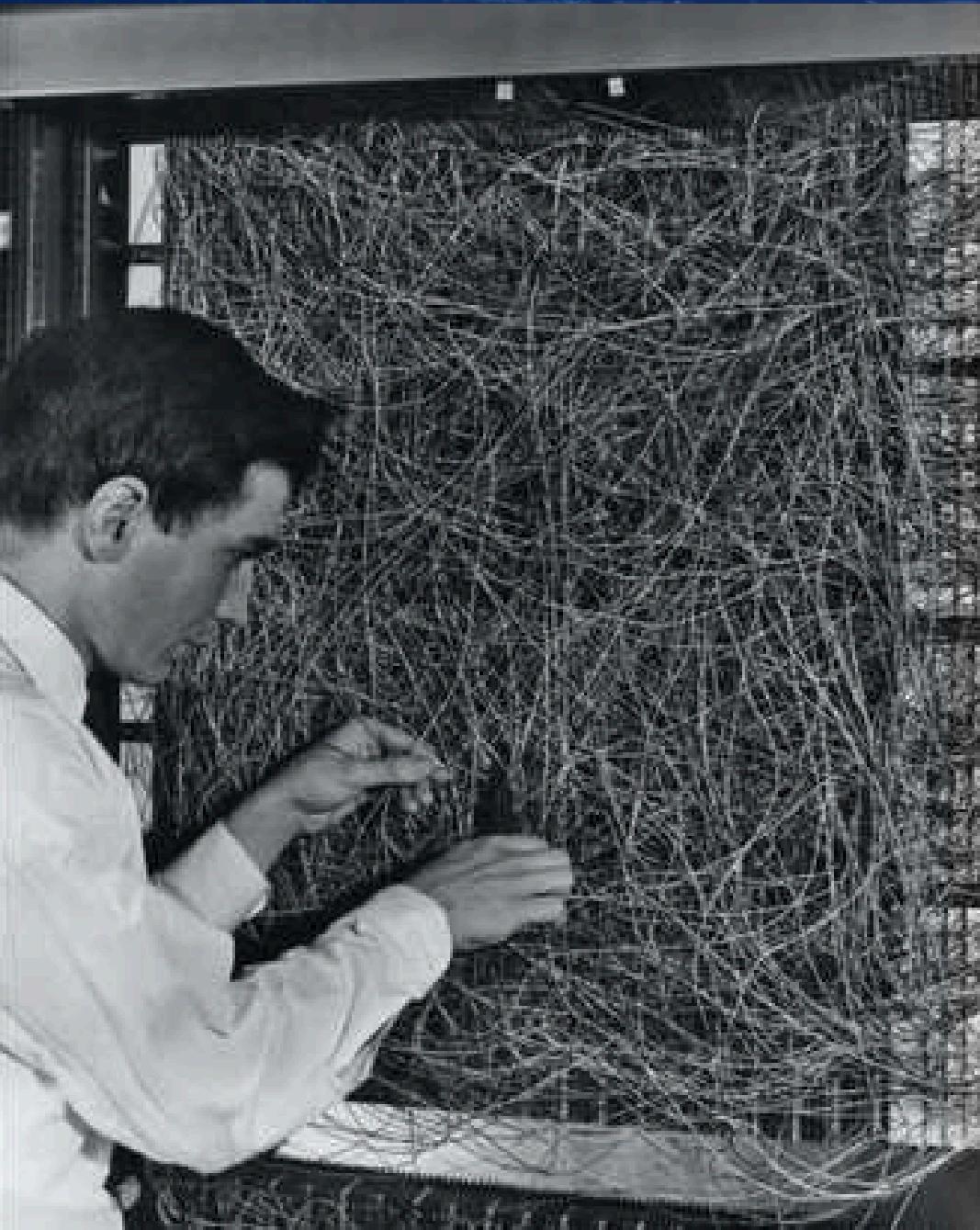


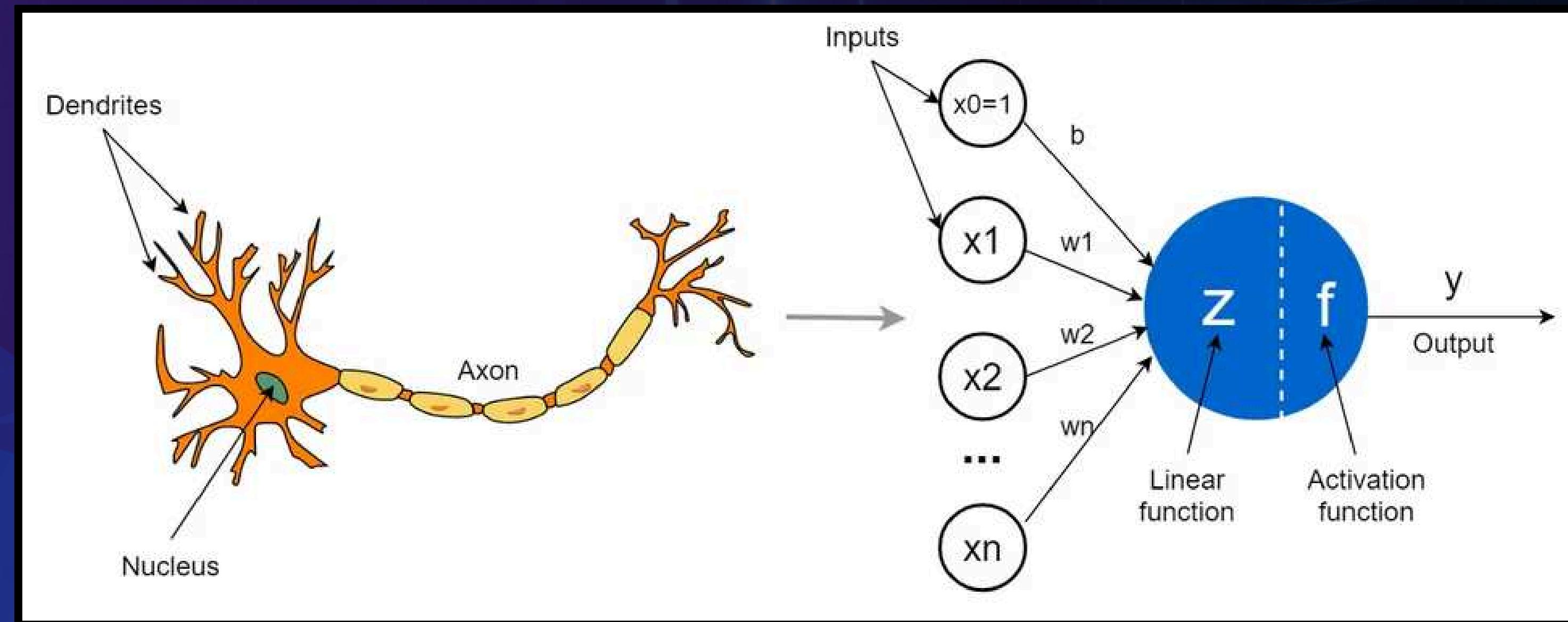
Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

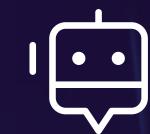


Perceptron

# Estructura

- Una neurona artificial toma varias entradas numéricas, cada una con un peso asociado.
- Se calcula una suma ponderada, a la que se le aplica una función de activación para producir una salida.



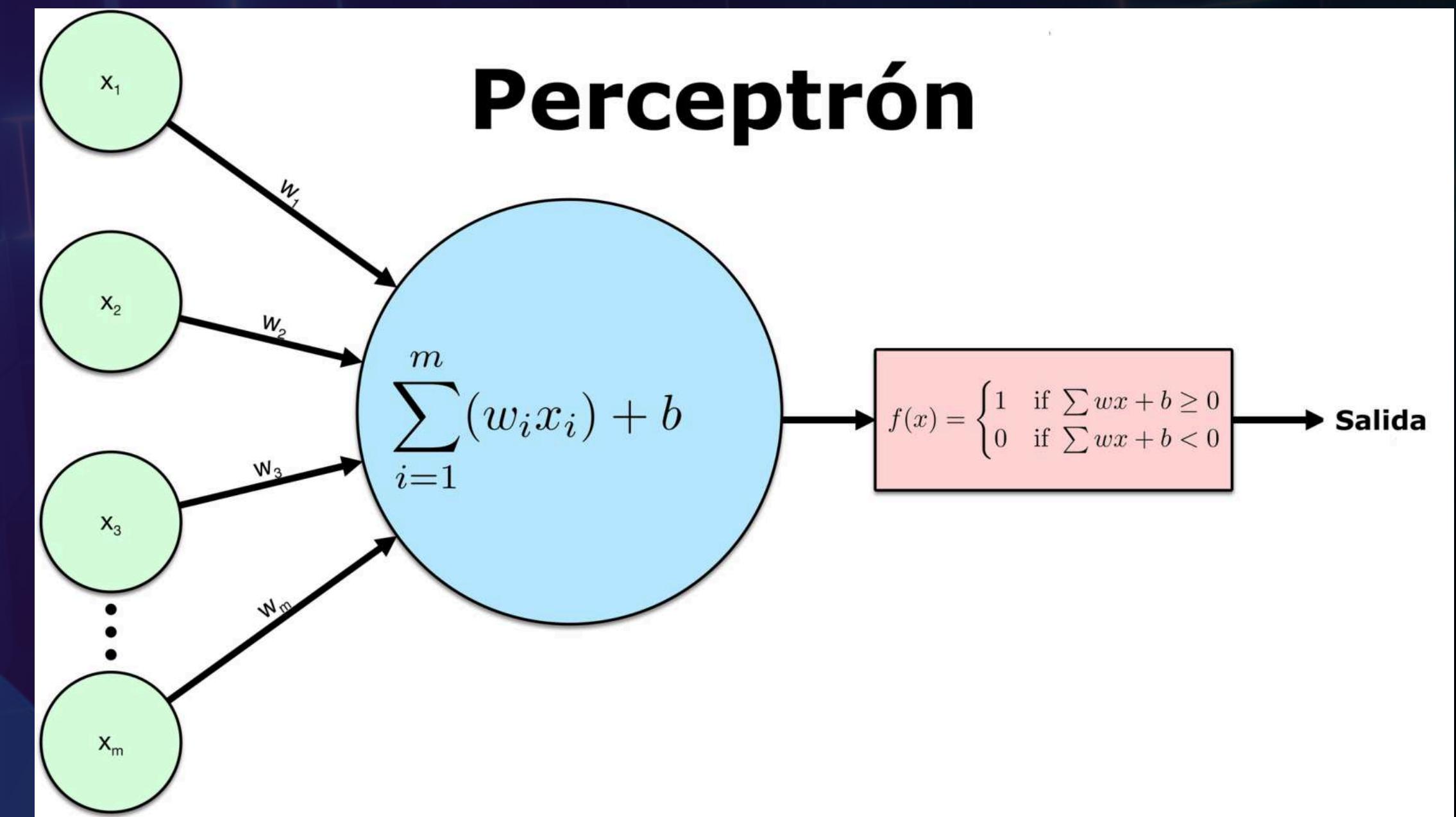
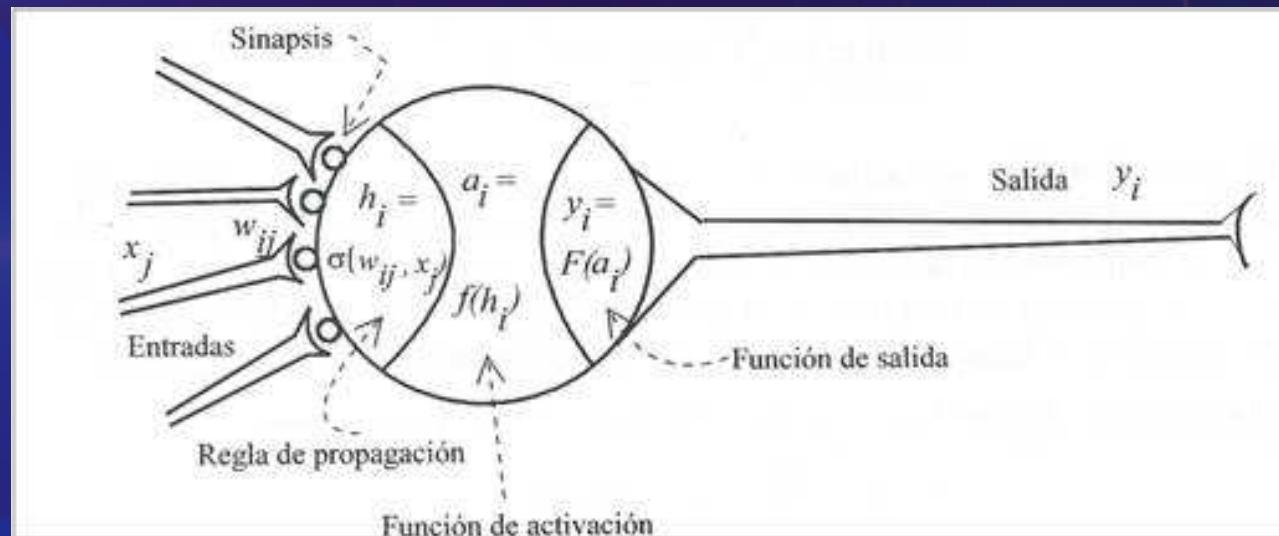


## Perceptron

# Estructura

Componentes clave:

1. **Entradas ( $x_1, x_2, \dots, x_n$ ):** Datos o características
2. **Pesos ( $w_1, w_2, \dots, w_n$ ):** Parámetros ajustables
3. **Suma ponderada**
4. **Función de activación**

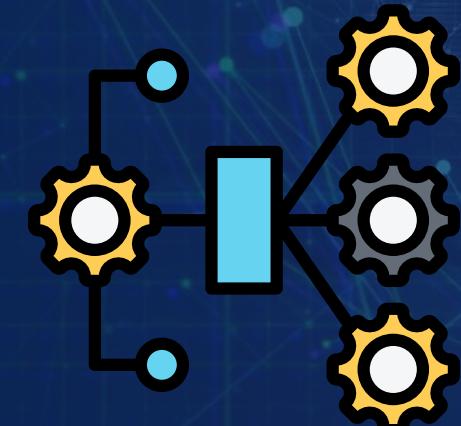




Perceptron

# Algoritmo

- Entrenamiento
  - *Ajuste de pesos (Parámetros)*
- Hiperparámetros
  - *Tasa de aprendizaje  $\eta$*
  - *Épocas*



- Entrenamiento
  - Se compara la salida deseada  $y_i$  con la salida estimada  $\hat{y}_i$
  - Si hay error, se ajustan los pesos
  - El sesgo  $b$  se ajusta igual
- Hiperparámetros
  - *El error promedio <  $\eta$* , ó
  - Se alcanza el máximo de épocas definido



Perceptron

# Algoritmo

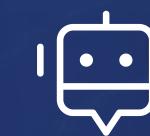
## Pasos del entrenamiento:

- *Inicialización:* Pesos aleatorios pequeños (ej. [-0.5, 0.5]) y sesgo (ej.  $b=0.1$ ).
- *Para cada muestra:*
  - Calcular salida  $y$
  - Error:  $e = y - y_{\text{pred}}$
- *Regla de actualización:*

$$w_i^{\text{nuevo}} = w_i^{\text{viejo}} + \eta \cdot e \cdot x_i$$

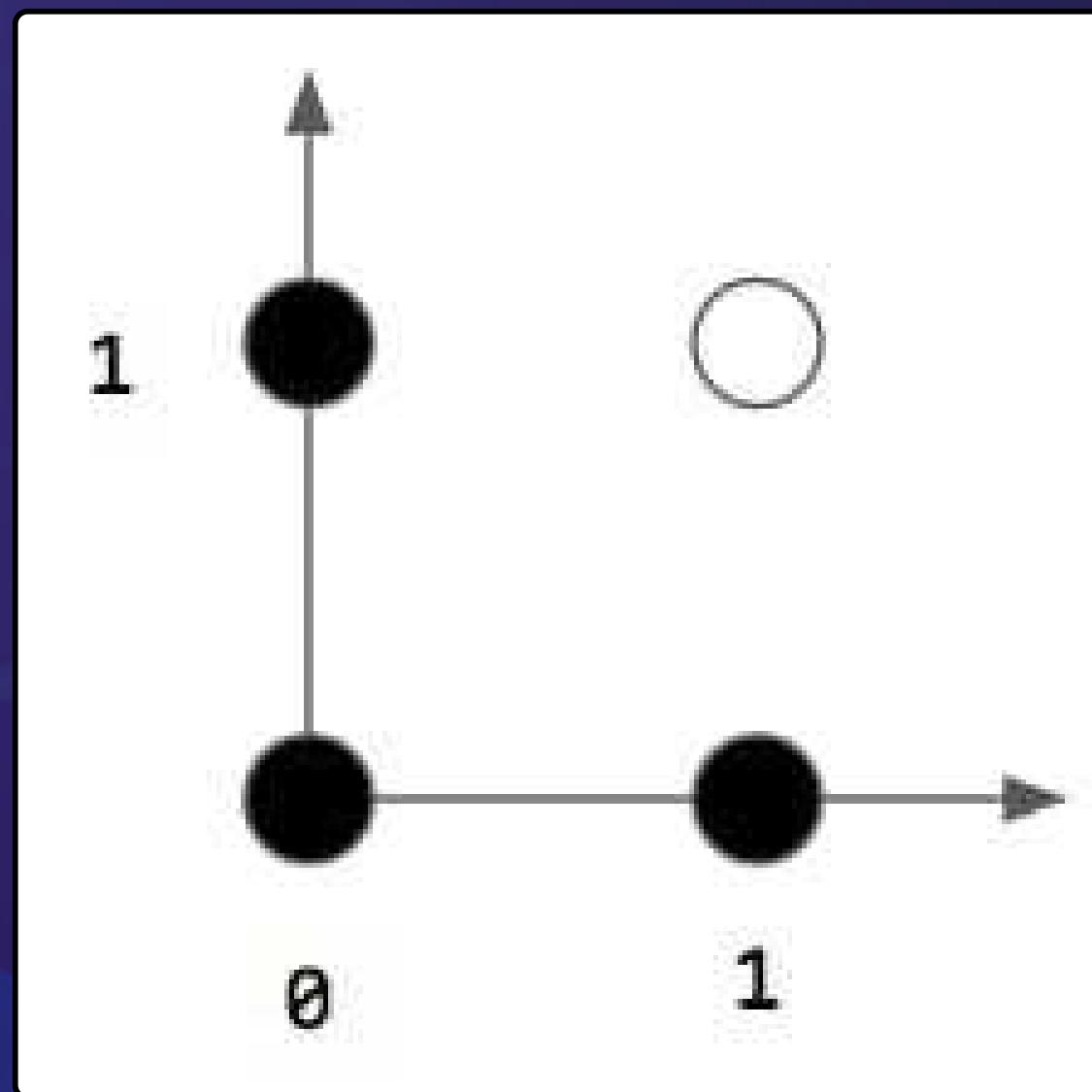
$$b_i^{\text{nuevo}} = b_i^{\text{viejo}} + \eta \cdot e \cdot 1$$

- Repetir hasta  $\text{error promedio} < \text{umbral}$  o máximo épocas.



Perceptron

# Ejemplo



ENTRADA $X_1$	ENTRADA $X_2$	SALIDA ESPERADA (Y)
0	0	0
0	1	0
1	0	1
1	1	1



Perceptron

# Ejemplo

Para cada entrada del conjunto de entrenamiento:

Calcular:

$$z = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

Obtener la salida con la función escalón:

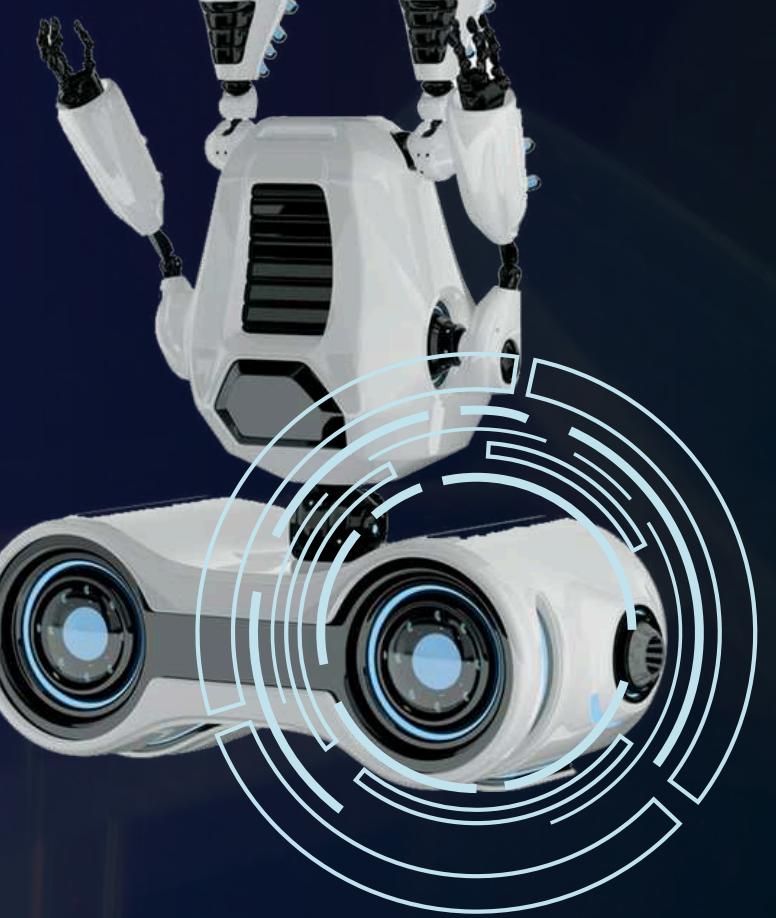
$$y_{\text{pred}} = \text{step}(z)$$

Calcular error:

$$\text{error} = y - y_{\text{pred}}$$

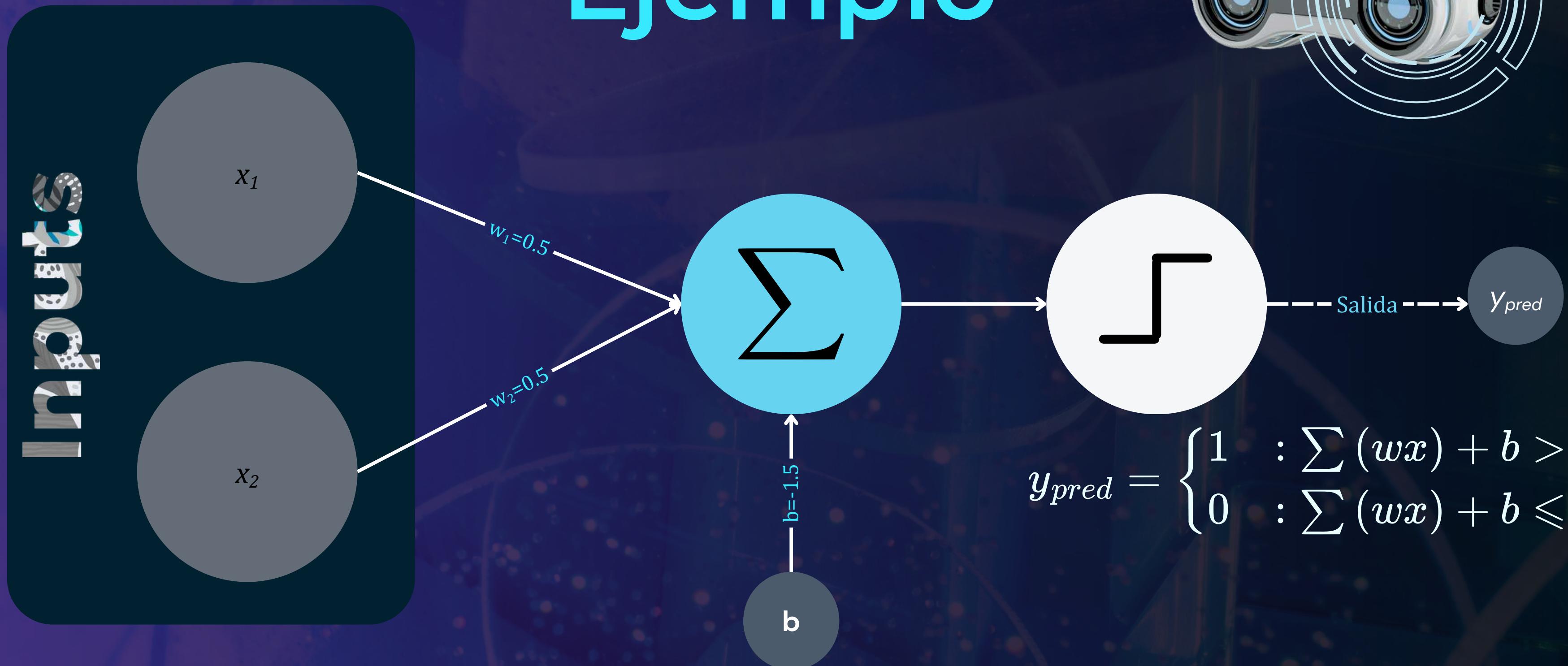
Actualizar pesos y bias:

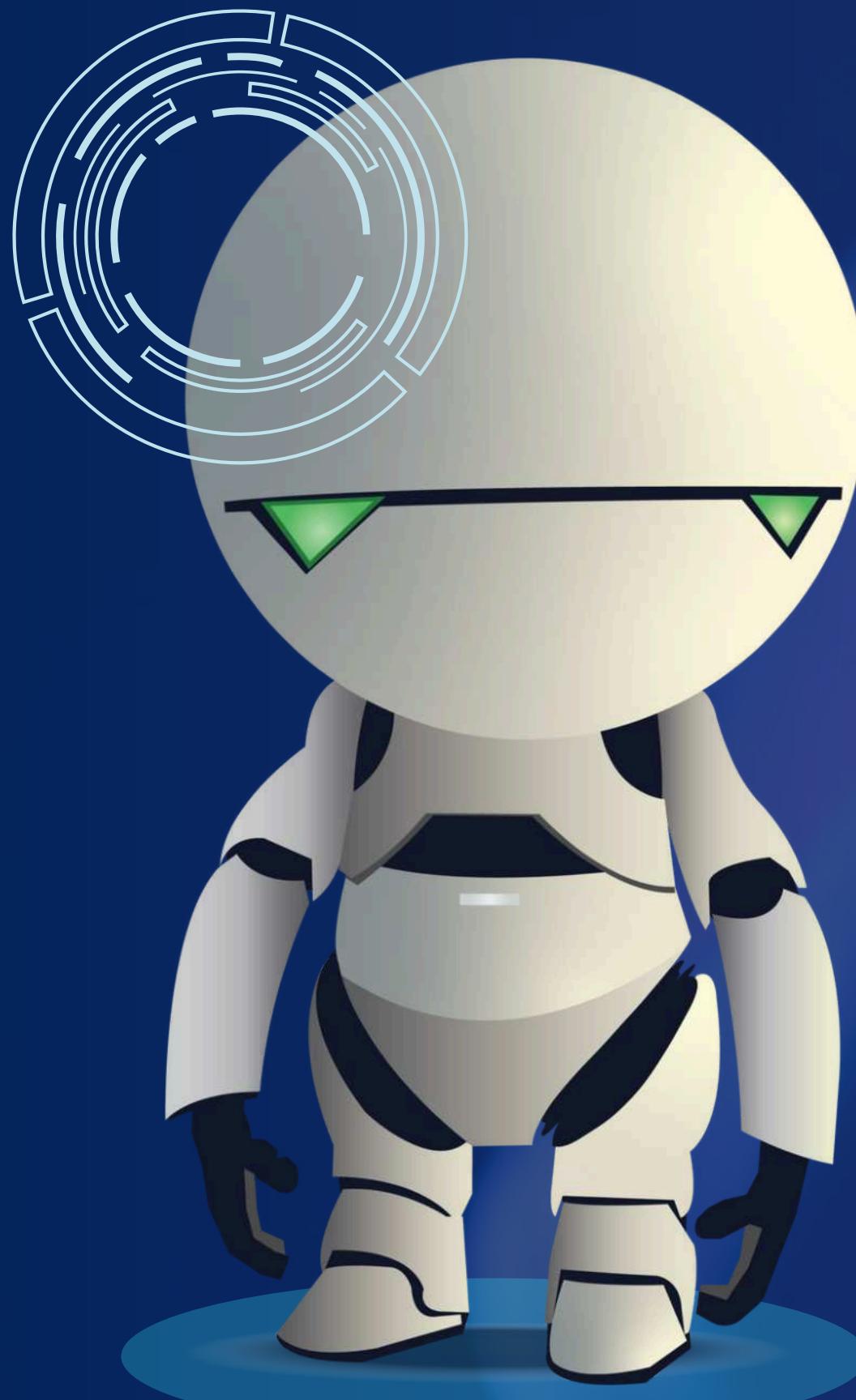
$$\begin{aligned}w_i &= w_i + \eta \cdot \text{error} \cdot x_i \\b &= b + \eta \cdot \text{error}\end{aligned}$$



Perceptron

# Ejemplo





# Ejemplo: Operador Lógico AND

- Pesos:  $w_1=0.5, w_2=0.5$
- Bias (sesgo):  $b = -1.5$
- Tasa de aprendizaje:  $\eta=1$
- Función de activación:
  - Escalón estricto

$$y_{pred} = \begin{cases} 1 & : z > 0 \\ 0 & : z \leq 0 \end{cases}$$



Perceptron

# Ejemplo

## Época 1

1) Ejemplo (0, 0)     $y = 0$

$$z=0.5(0)+0.5(0)+(-1.5)=-1.5 \quad y_{pred}=0 \quad \text{Error} = 0-0 = 0 \rightarrow \text{No se actualiza}$$

2) Ejemplo (0, 1)     $y = 0$

$$z=0.5(0)+0.5(1)+(-1.5)=-1.0 \quad y_{pred}=0 \quad \text{Error} = 0-0 = 0 \rightarrow \text{No se actualiza}$$

3) Ejemplo (1, 0)     $y = 0$

$$z=0.5(1)+0.5(0)+(-1.5)=-1.0 \quad y_{pred}=0 \quad \text{Error} = 0-0 = 0 \rightarrow \text{No se actualiza}$$

4) Ejemplo (1, 1)     $y = 1$

$$z=0.5(1)+0.5(1)+(-1.5)=-0.5 \quad y_{pred}=0 \quad \text{Error} = 1-0 = 1 \rightarrow \text{(error)}$$



Perceptron

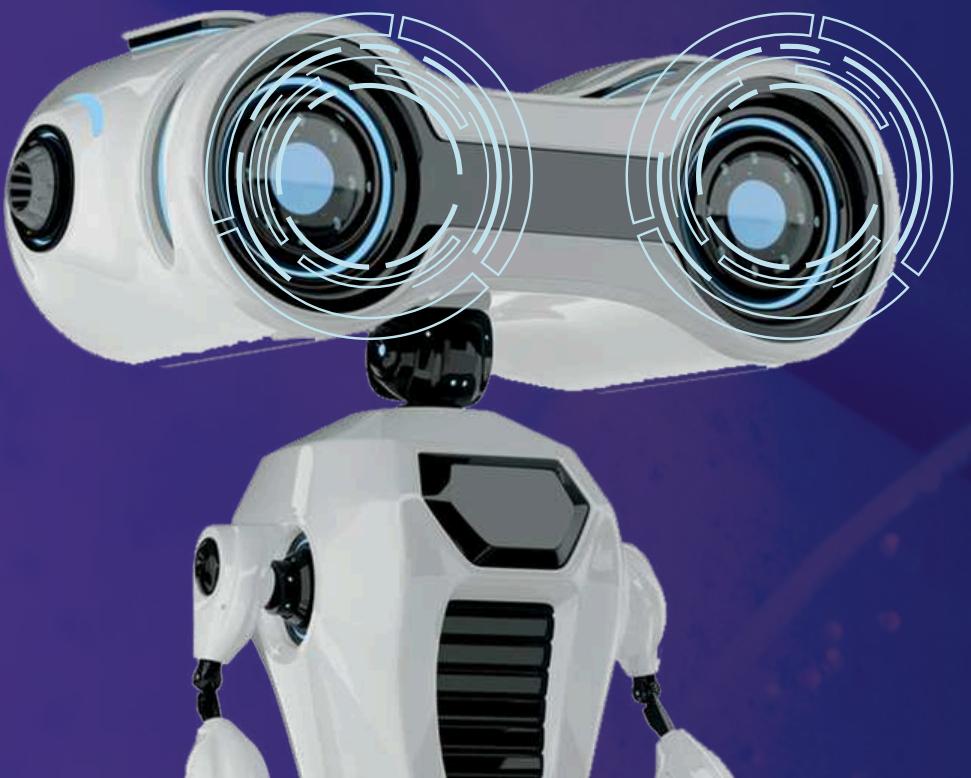
# Ejemplo

Actualización:

$$w_1 = 0.5 + 1(1 - 0)(1) = 1.5$$

$$w_2 = 0.5 + 1(1 - 0)(1) = 1.5$$

$$b = -1.5 + 1(1 - 0) = -0.5$$





Perceptron

# Ejemplo

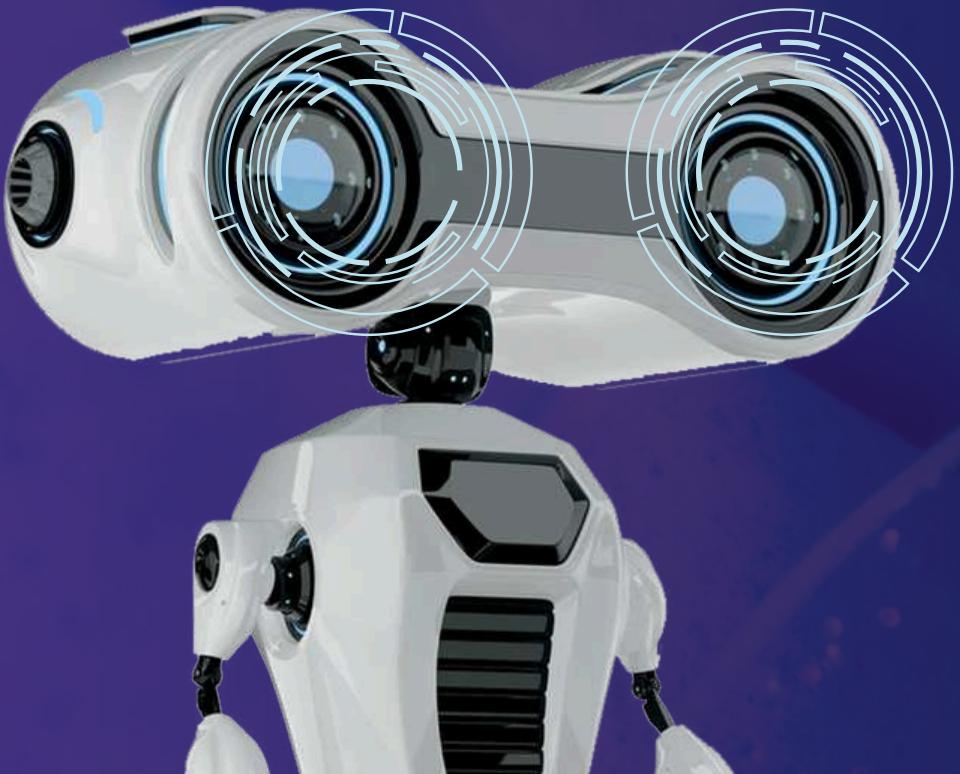
## Época 2

1) Ejemplo (0, 0)     $y = 0$

$$z=1.5(0)+1.5(0)+(-0.5)=-0.5 \quad y_{pred}=0 \quad Error = 0-0 = 0 \rightarrow \text{No se actualiza}$$

2) Ejemplo (0, 1)     $y = 0$

$$z=1.5(0)+1.5(1)+(-0.5)=1.0 \quad y_{pred}=1 \quad Error = 0-1 = -1 \rightarrow \underline{\text{(error)}}.$$





Perceptron

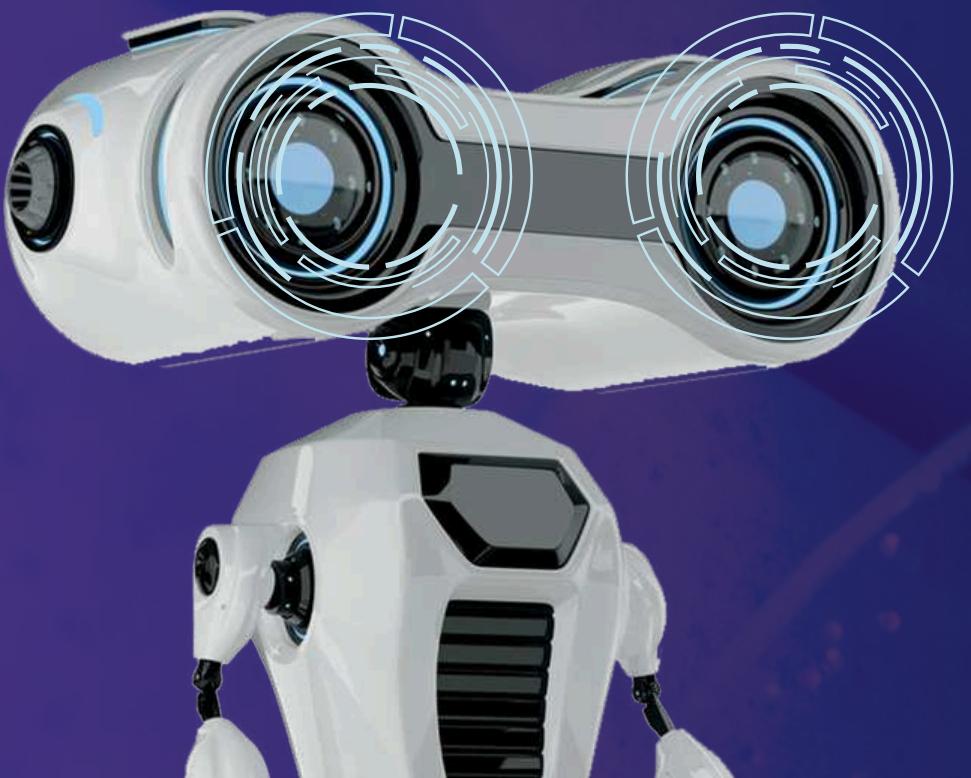
# Ejemplo

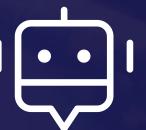
Actualización:

$$w_1 = 1.5 + 1(0-1)(0) = 1.5$$

$$w_2 = 1.5 + 1(0-1)(1) = 0.5$$

$$b = -0.5 + 1(0-1) = -1.5$$





Perceptron

# Ejemplo

## Época 2

1) Ejemplo (0, 0)     $y = 0$

$$z=1.5(0)+1.5(0)+(-0.5)=-1 \quad y_{pred}=0 \quad Error = 0-0 = 0 \rightarrow \text{No se actualiza}$$

2) Ejemplo (0, 1)     $y = 0$

$$z=1.5(0)+1.5(1)+(-0.5)=-0.5 \quad y_{pred}=1 \quad Error = 0-1 = -1 \rightarrow (\text{error})$$

Nuevos pesos:

$$w_1=1.5$$

$$w_2=0.5$$

$$b=-1.5$$

3) Ejemplo (1, 0)     $y = 0$

$$z=1.5(1)+0.5(0)+(-1.5)=0 \quad y_{pred}=0 \quad Error = 0-0 = 0 \rightarrow \text{No se actualiza}$$

4) Ejemplo (1, 1)     $y = 1$

$$z=1.5(1)+0.5(1)+(-1.5)=0.5 \quad y_{pred}=1 \quad Error = 1-1 = 0 \rightarrow \text{No se actualiza}$$



Perceptron

# Funciones Step

Variante	Fórmula	Valor en z=0	Descripción
Clásica	$f(x) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$	f(0)=1	Activa en cero y valores positivos
Strict step	$f(x) = \begin{cases} 0 & z \leq 0 \\ 1 & z > 0 \end{cases}$	f(0)=0	Solo activa con valores estrictamente positivos



Perceptron

# Ejercicio

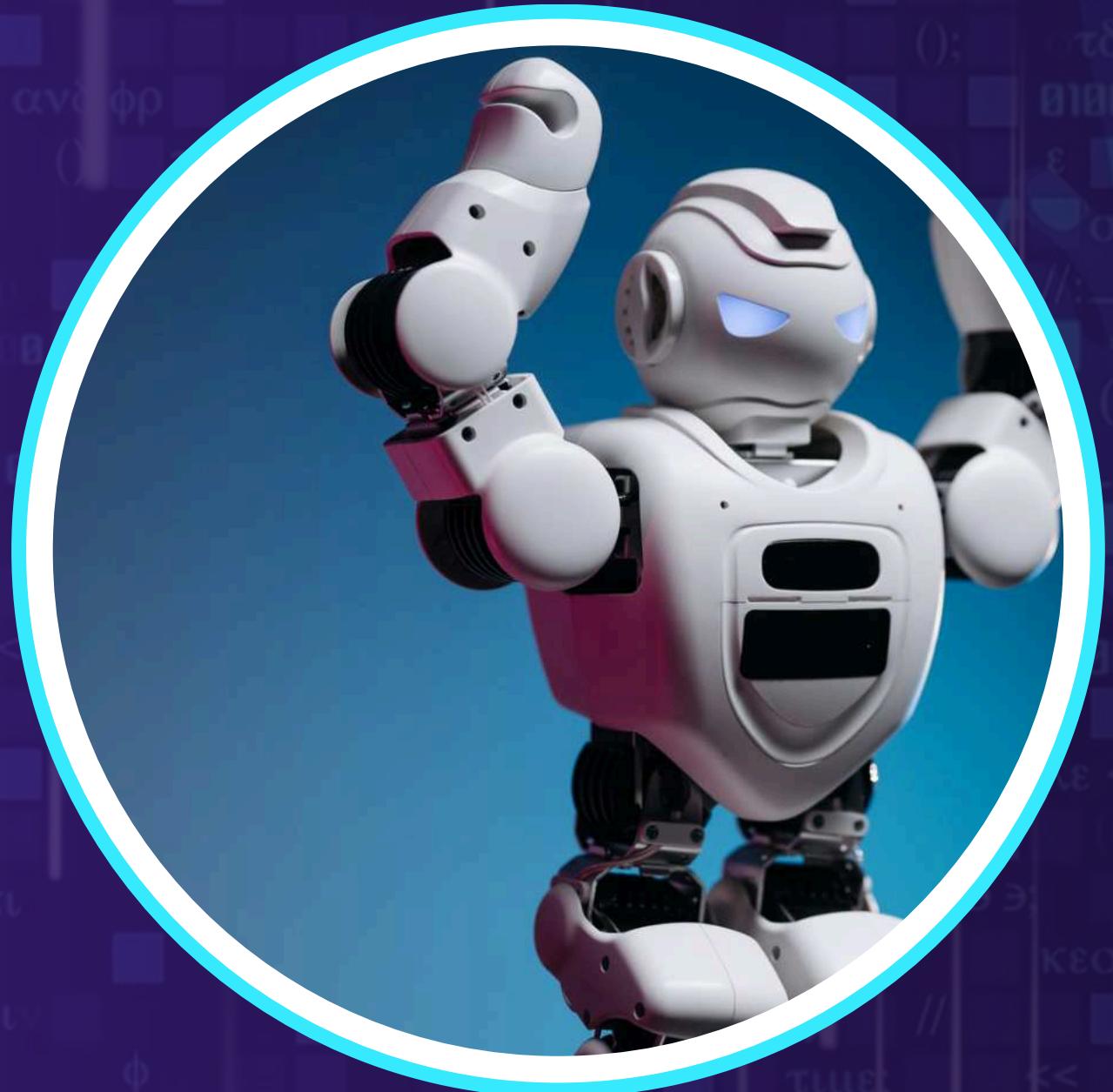
ENTRADA X <sub>1</sub>	ENTRADA X <sub>2</sub>	SALIDA ESPERADA (Y)
0	0	0
0	1	0
1	0	0
1	1	1

- Pesos:  $w_1=0.5, w_2=0.5$
- Bias (sesgo):  $b = -1.5$
- Tasa de aprendizaje:  $\eta=1$
- Función de activación:
  - Escalón Clásico

$$y_{pred} = \begin{cases} 1 & : z \geqslant 0 \\ 0 & : z < 0 \end{cases}$$



# Variantes



## Perceptrón Online (Aprendizaje Secuencial o Incremental)

- Se actualizan los pesos cada vez que se procesa un ejemplo de entrenamiento.
- Ventajas:
  - Reacciona rápidamente a nuevos datos.
  - Ideal para flujos de datos o aprendizaje en tiempo real.

## Perceptrón Offline (Aprendizaje por Lotes o Batch)

- Se procesa todo el conjunto de entrenamiento antes de actualizar los pesos.
- Ventajas:
  - Puede ser más estable.
  - Reduce ruido de datos individuales.
- En el caso del perceptrón: se suele simular ejecutando una época completa (pasar por todos los datos) y luego realizar una actualización basada en el conjunto de errores.

# Variantes



## ¿Cuál usar?

- En la práctica educativa y con datasets pequeños (como los de OR, AND, XOR), el modo online es el más común y más fácil de entender.
- En problemas más grandes o si hay mucha variabilidad en los datos, el modo offline o mini-batch puede ser preferido.





# Limitaciones Clave

1. **Separabilidad lineal:** Solo resuelve problemas con frontera decisional recta.
2. **XOR problem:** Falla completamente (motivó redes multicapa).
3. **Datos ruidosos:** Sensible a outliers (no tiene regularización).
4. **Clases desbalanceadas:** Tiende a favorecer clases mayoritarias.



# Ejercicio en Casa:

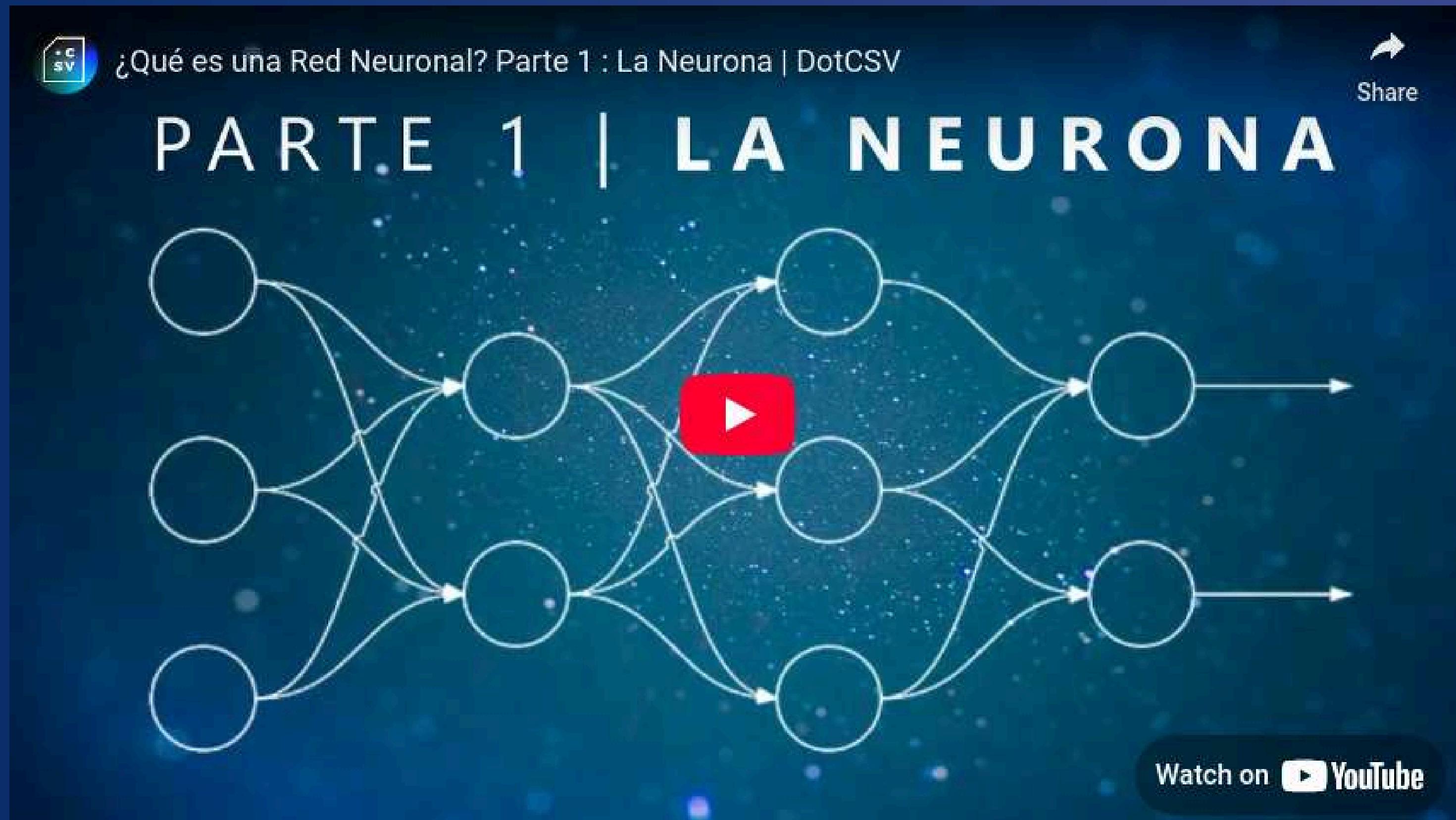
ENTRADA X <sub>1</sub>	ENTRADA X <sub>2</sub>	SALIDA ESPERADA (Y)
0	0	0
0	1	1
1	0	1
1	1	0

- Pesos:  $w_1=0.5, w_2=0.5$
- Bias (sesgo):  $b = -1.5$
- Tasa de aprendizaje:  $\eta=1$
- Función de activación:
  - Escalón Clásico

$$y_{pred} = \begin{cases} 1 & : z \geqslant 0 \\ 0 & : z < 0 \end{cases}$$



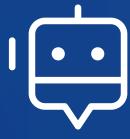
# Resumen





# Conclusión

El perceptrón sentó las bases para el deep learning, pero su incapacidad con problemas no lineales llevó al desarrollo de arquitecturas más complejas (MLP, CNN). Su simplicidad lo hace ideal para introducir conceptos de aprendizaje automático supervisado.



# Próxima sesión:

*Perceptrón Multiclasificación*

*Perceptrón Multicapa:*

- *Topología*
- *Algoritmo: Forward Propagation*

*Funciones de Activación*