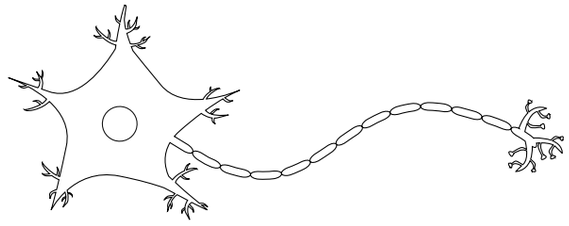
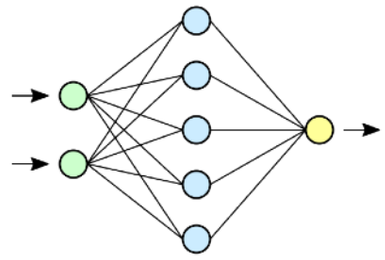
Inteligencia artificial aplicada a la astrofísica

Una introducción: Redes neuronales y vectores



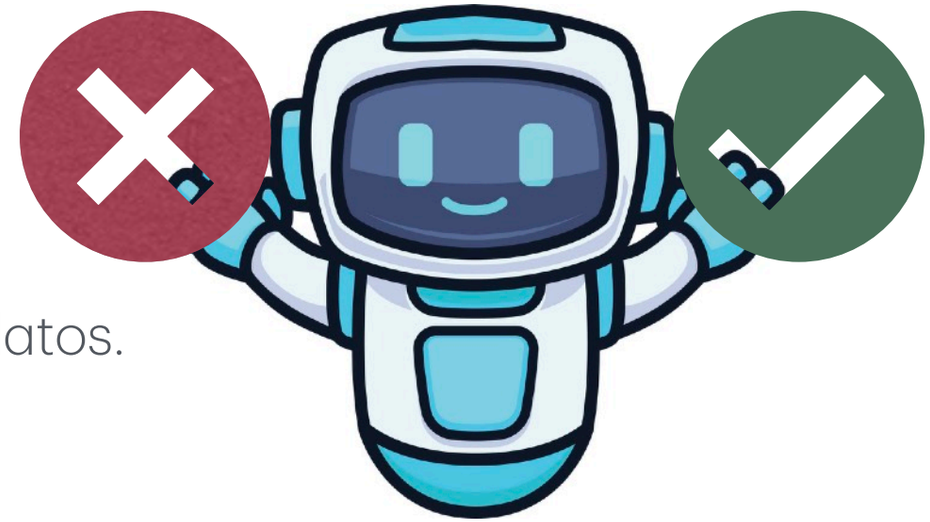


Pedro Elías Mirón Enríquez, TAACO, 18 de noviembre de 2024
pmiron@astro.unam.mx

Inteligencia Artificial:

¿Para qué?

- Clasificación correcta de información.
- Aprendizaje automatizado a partir de datos.



Aplicaciones

Aprendizaje Automático (Machine Learning)

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje por refuerzo
- Redes neuronales

Procesamiento de Lenguaje Natural

- Traducción automática
- Análisis de sentimientos
- Chatbots y asistentes virtuales
- Generación de texto

Visión por Computadora

- Reconocimiento facial
- Detección de objetos
- Procesamiento de imágenes médicas
- Vehículos autónomos

Robótica

- Robots industriales
- Robots de servicio
- Robots colaborativos
- Drones autónomos

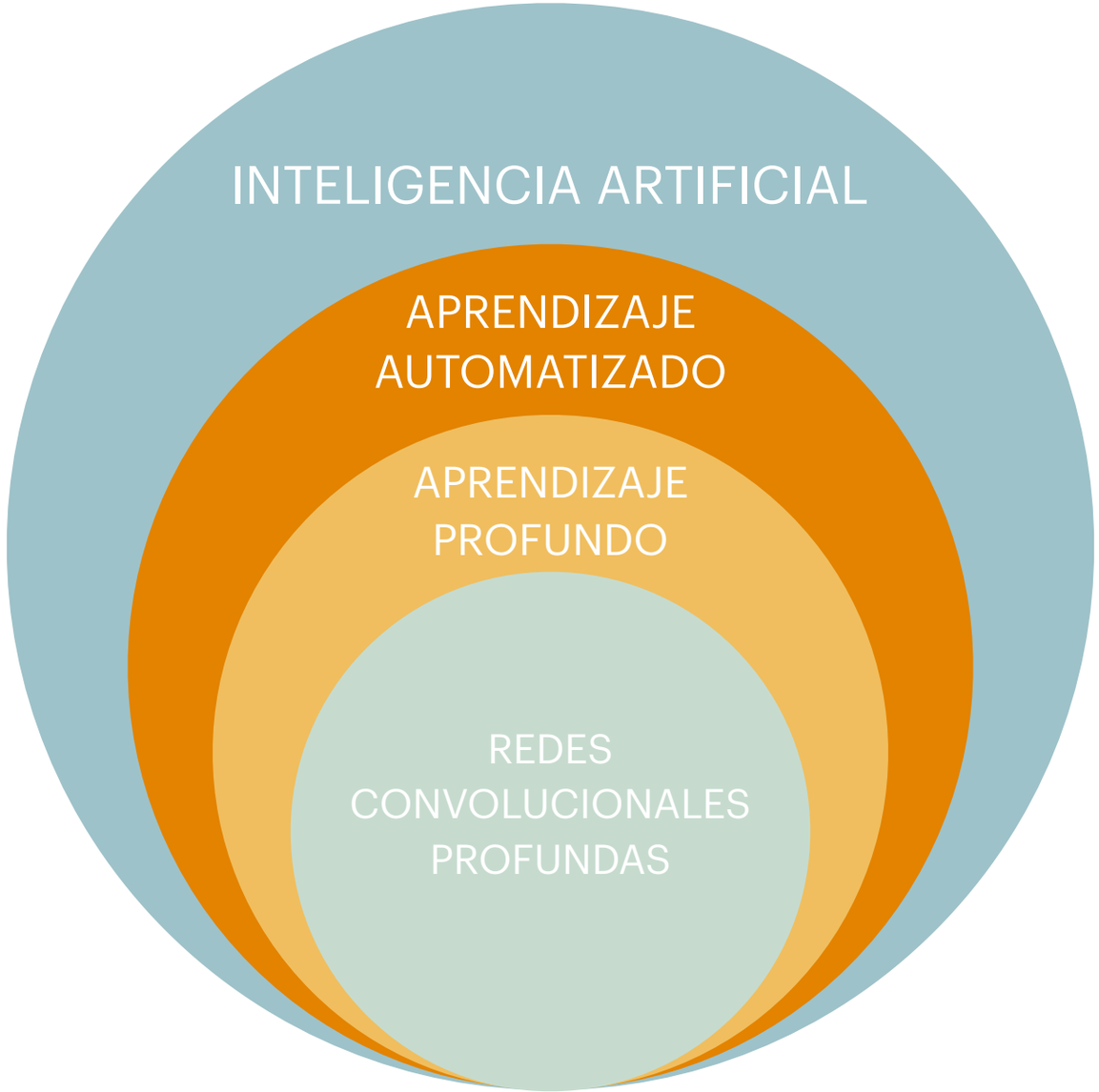
Aplicaciones Empresariales

- Análisis predictivo
- Sistemas de recomendación
- Detección de fraude
- Optimización de procesos

ETC..

Etc...

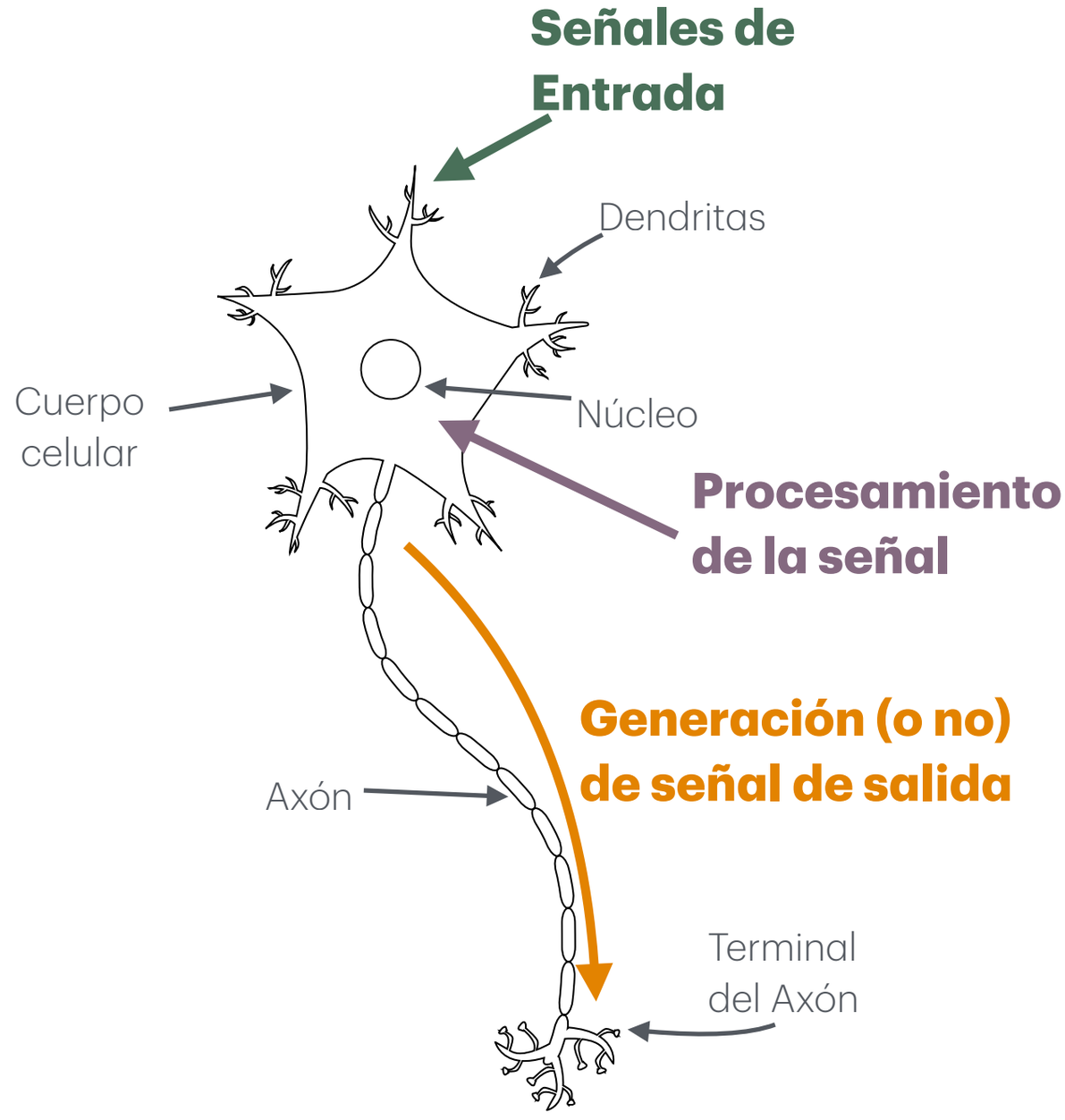
Una clasificación:



Introducción

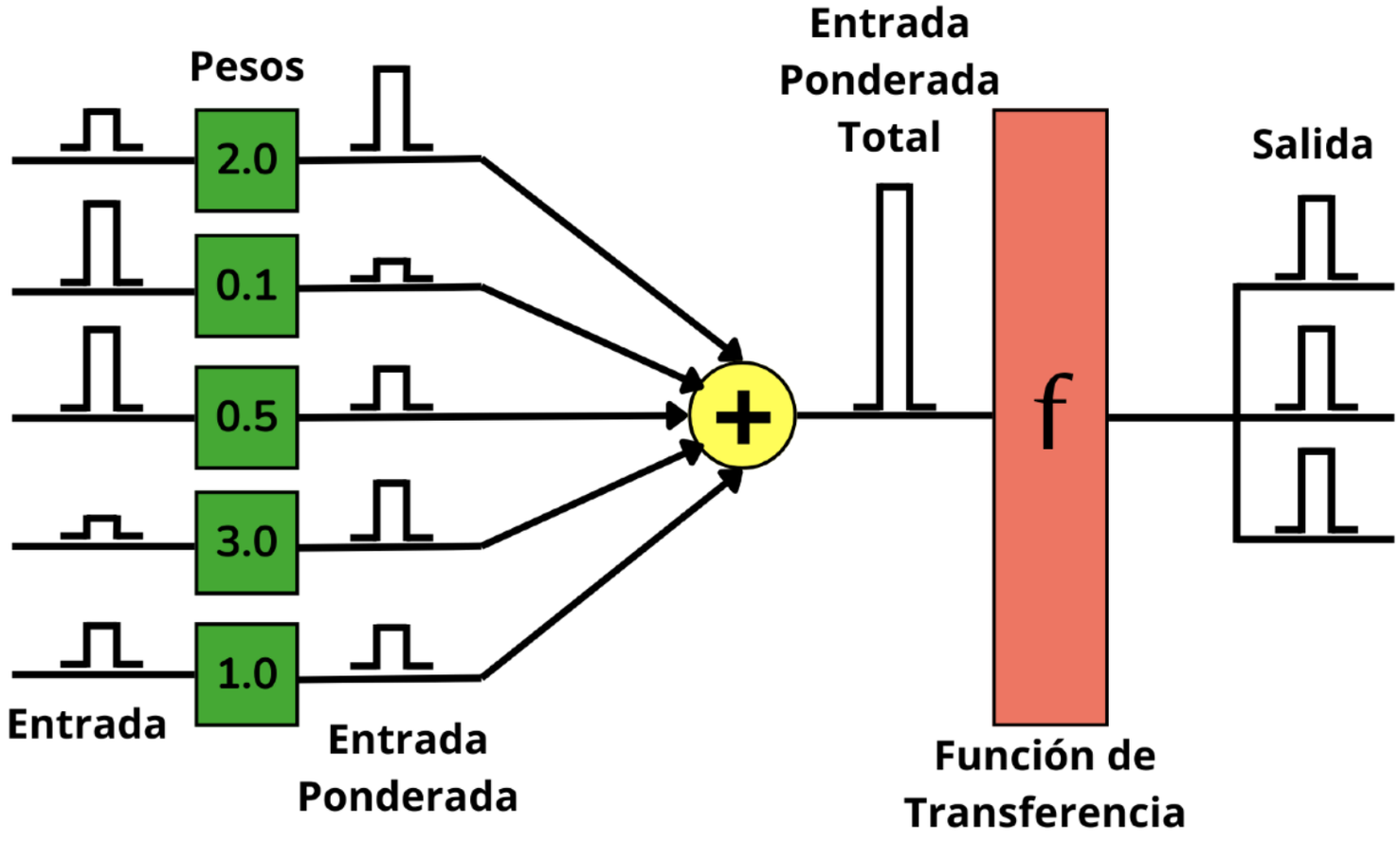
Neuronas:
Una inspiración biológica

- ¿Qué es la inteligencia?
- Neurona como elemento de la psique mínimo
- Modelo matemático inspirado de una simplificación de este proceso



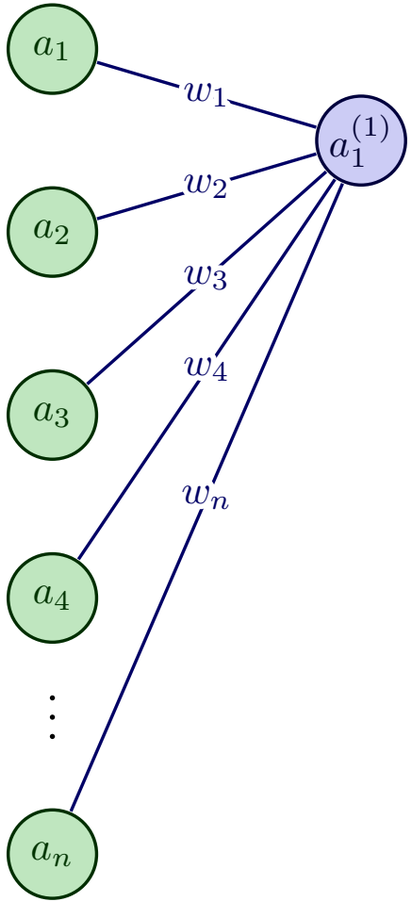
Primera abstracción

Entradas, pesos y función de transferencia



El modelo matemático

Combinación lineal compuesta con una función no lineal



$$= f\left(w_1 a_1 + w_2 a_2 + \ldots + w_n a_n + b\right)$$

$$= f\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

$$a_1^{(1)} = f\left[\begin{pmatrix} w_1 & w_2 & \ldots & w_n \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} + b\right]$$

$$= f\left(\mathbf{w}^T \mathbf{a} + b\right)$$

- Las entradas a_i representan las señales que vienen de otras neuronas (o el mundo exterior)
- Los pesos w_i son la intensidad que le da la neurona a las entradas
- f es la función umbral que la neurona debe sobrepasar para activarse
- b es la ganancia o sesgo (bias) de la neurona (o nodo)

Función de transferencia

Introduciendo no linealidad

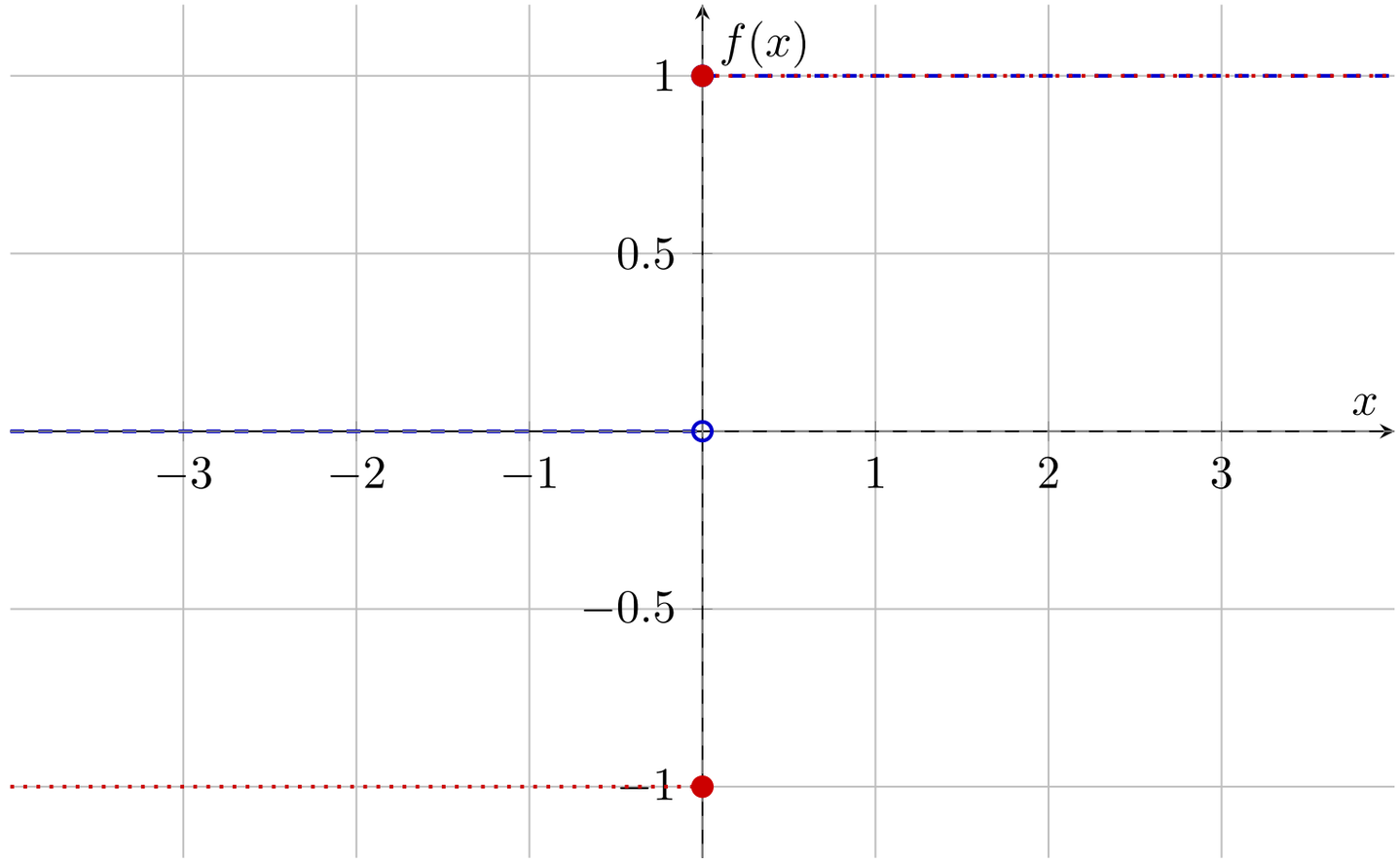
Funciones de Transferencia:

Hardlim:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Hardlims:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$



Función de transferencia

Introduciendo no linealidad

Funciones de Transferencia:

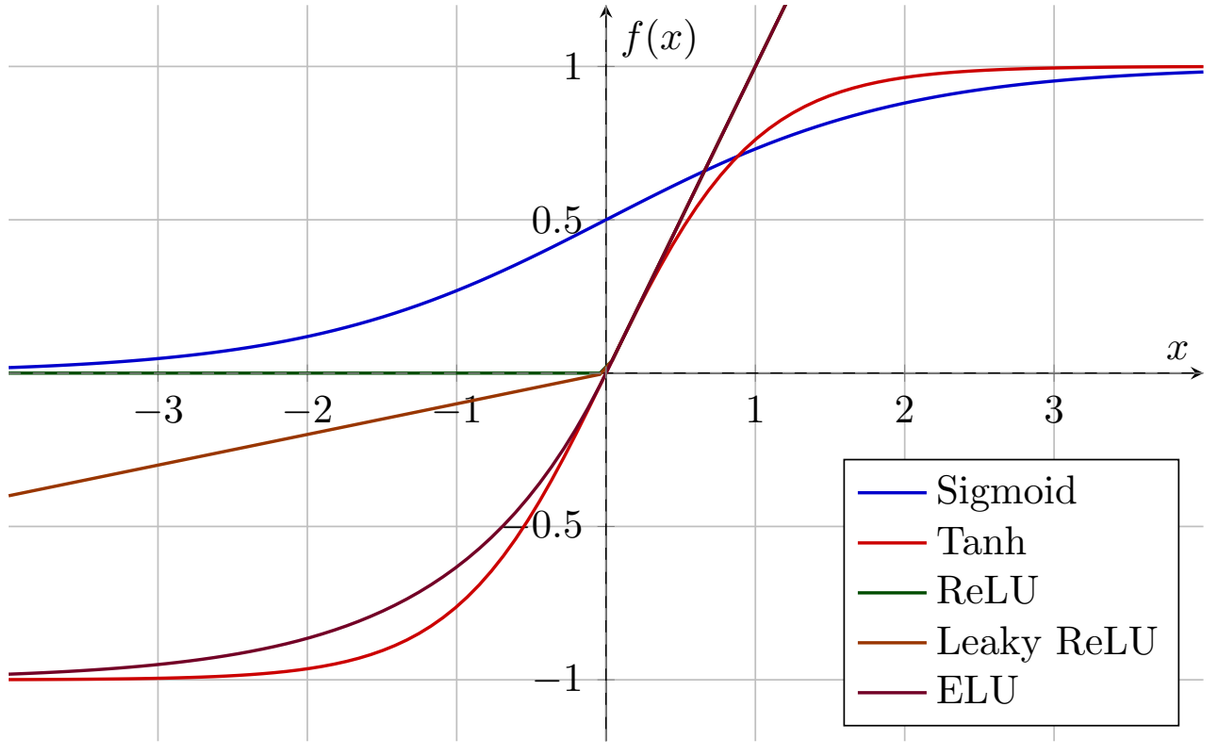
Sigmoid:
$$f(x) = \frac{1}{1+e^{-x}}$$

Tanh:
$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

ReLU:
$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

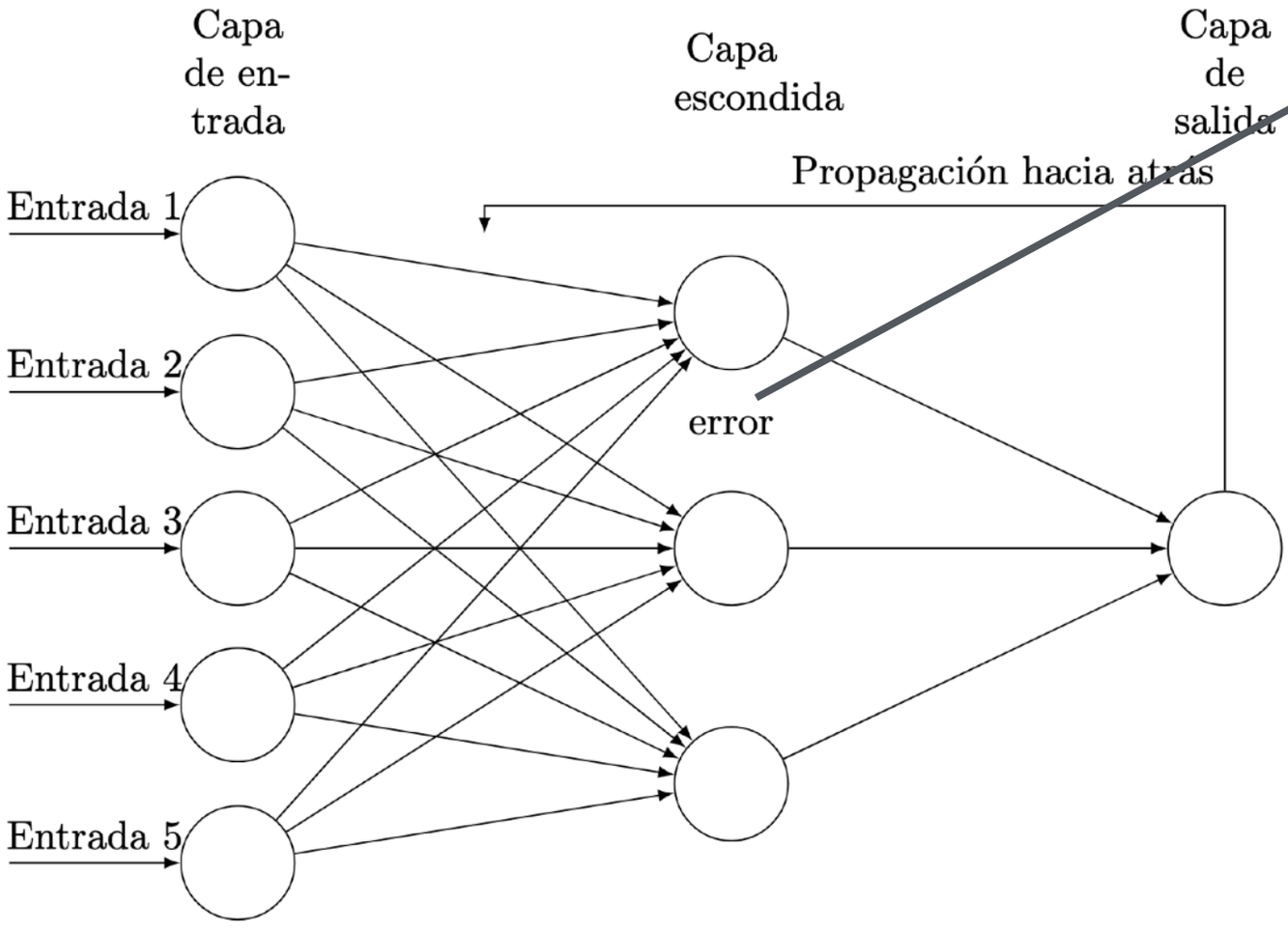
Leaky ReLU:
$$f(x) = \begin{cases} x & x \geq 0 \\ 0.1x & x < 0 \end{cases}$$

ELU:
$$f(x) = \begin{cases} x & x \geq 0 \\ e^{x} - 1 & x < 0 \end{cases}$$



Aprender: ¿Cómo?

Propagación hacia atrás



Donde E será nuestra función error

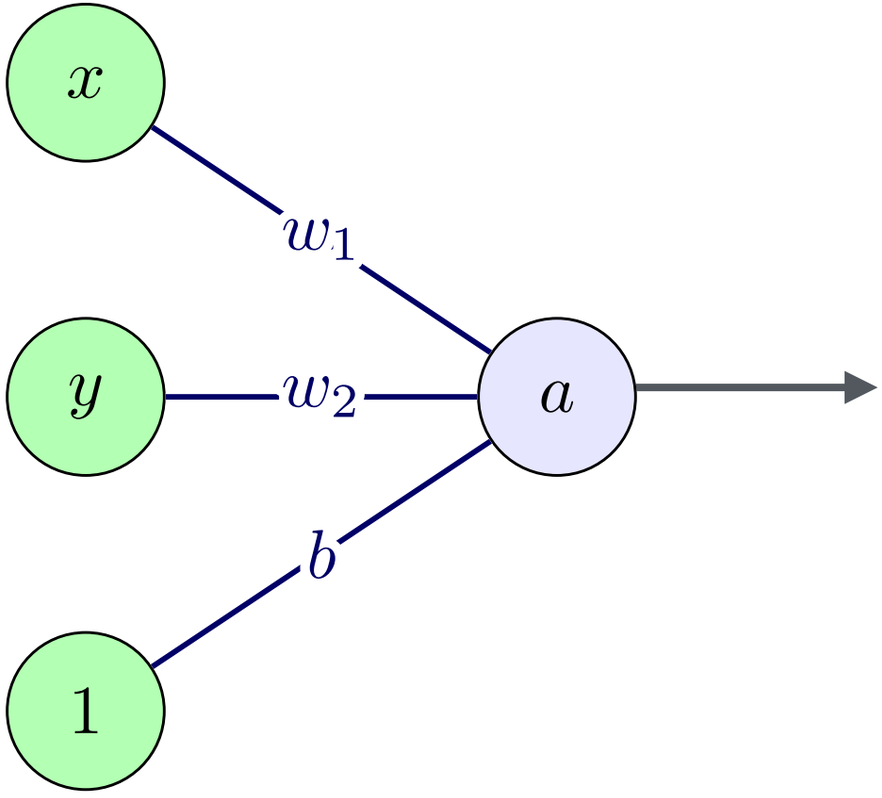
La red "aprende" actualizando sus pesos y sesgos de la siguiente manera:

$$W_{nuevo} = W_{anterior} - \alpha \frac{\partial E}{\partial W}$$

$$b_{nuevo} = b_{anterior} - \alpha \frac{\partial E}{\partial b}$$

Un ejemplo sencillo

Una recta como clasificador de puntos: el perceptrón



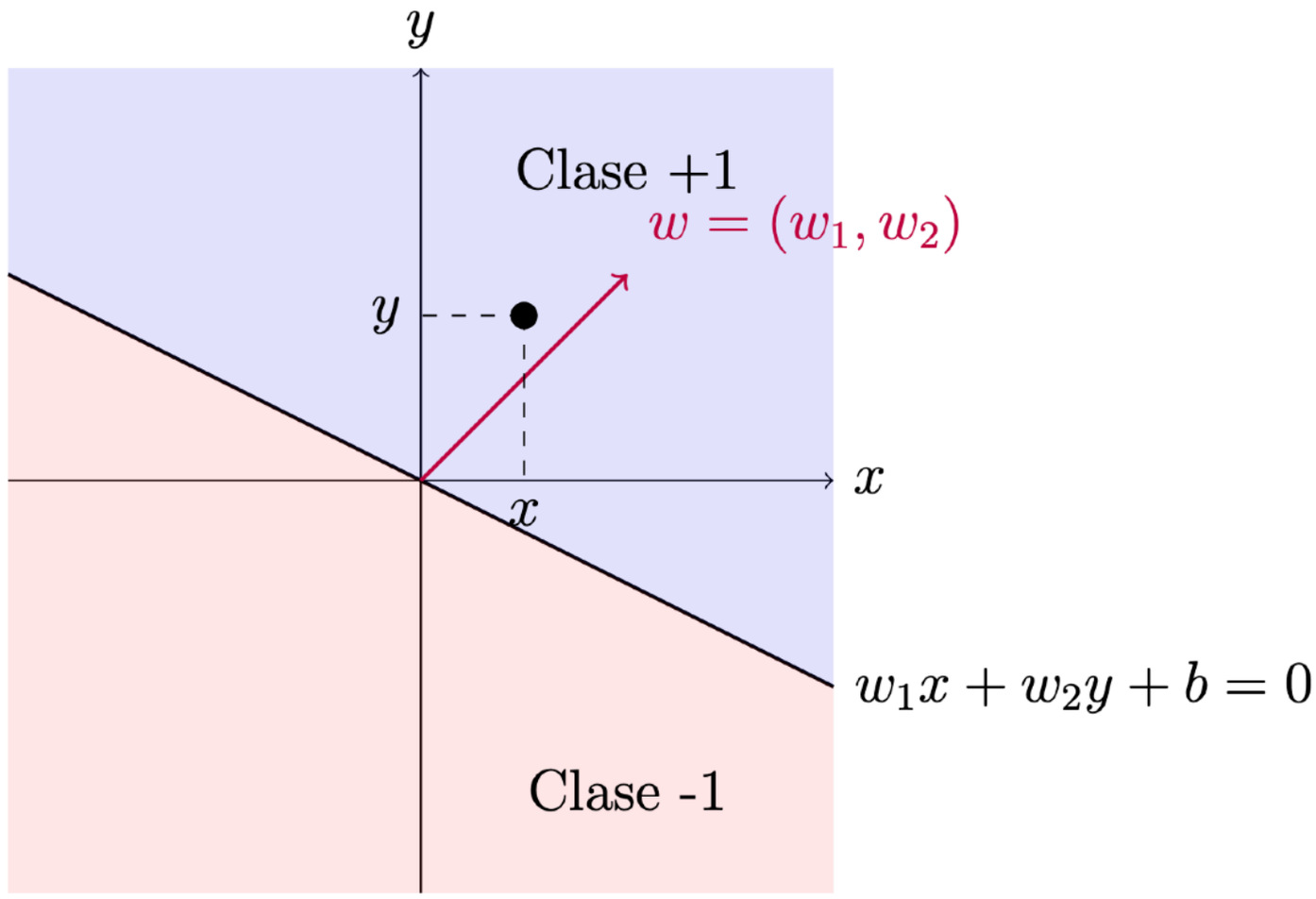
$$f(w_1 x + w_2 y + b)$$

Hardlims:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

Un ejemplo sencillo

Clasificador de puntos (linealmente separables):



Propagación hacia atrás (en este caso)

Mover una línea para cubrir/excluir puntos

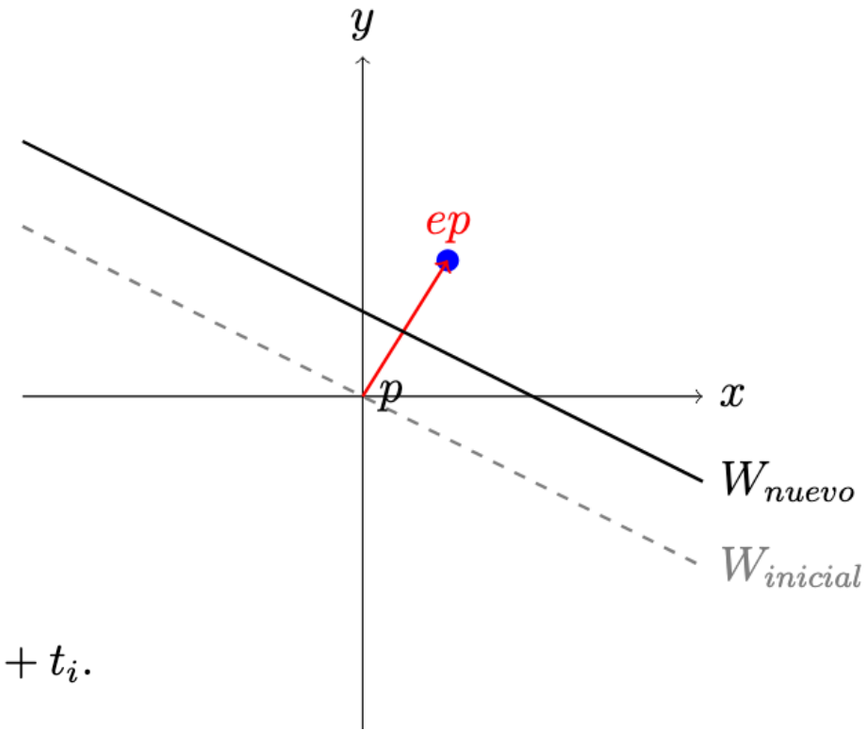
Considerando que:

- La salida a solo puede ser +1 o -1
- El target t solo puede ser +1 o -1
- El error $e = t - a$ solo puede ser 0, +2, o -2

Las actualizaciones se reducen a:

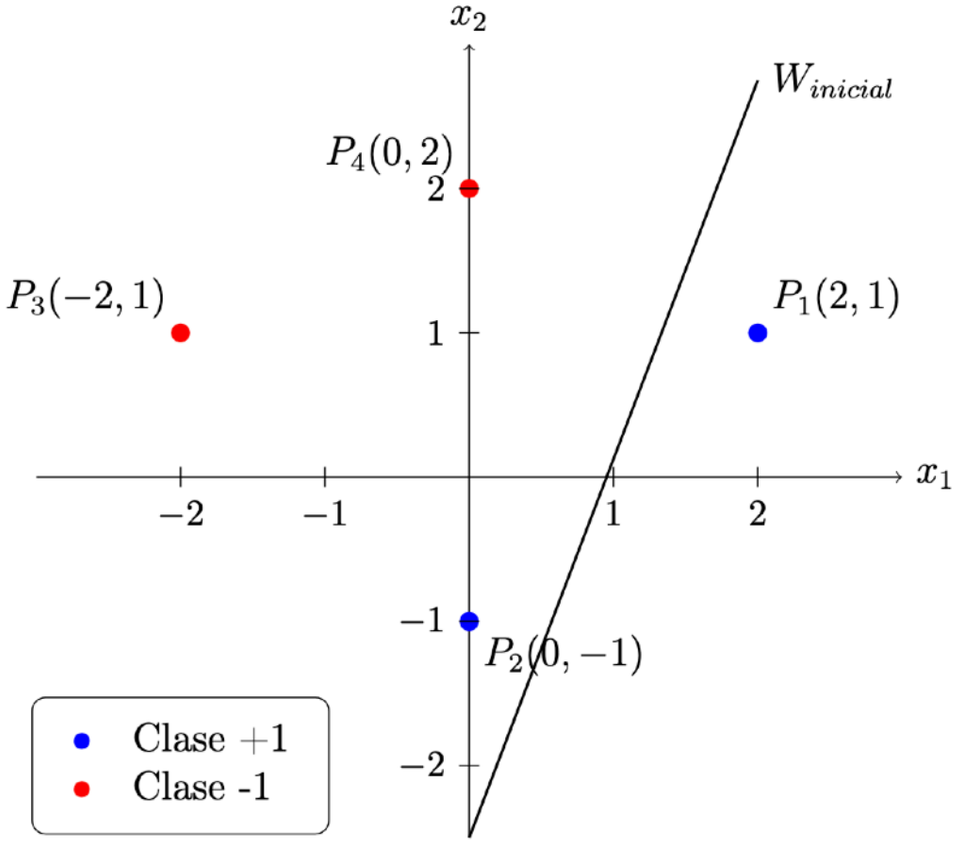
Usamos la regla de actualización del perceptrón:

$$W \leftarrow W + t_i P_i, \quad b \leftarrow b + t_i.$$



Paso cero:

Una estimación aleatoria



Puntos y etiquetas:

$$P_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, t_1 = +1, \quad P_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_2 = +1,$$

$$P_3 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, t_3 = -1, \quad P_4 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, t_4 = -1.$$

Pesos y sesgo iniciales:

$$W = \begin{bmatrix} -0.7 \\ 0.2 \end{bmatrix}, \quad b = 0.5.$$

Paso cero...

Todavía:

La salida se calcula como:

$$f(x) = \text{hardlims}(W^\top P + b)$$

- Para $P_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, t_1 = +1$:

$$f(P_1) = \text{hardlims}((-0.7)(2)+(0.2)(1)+0.5) = \text{hardlims}(-1.4+0.2+0.5) = \text{hardlims}(-0.7) = -1.$$

Incorrecto, $t_1 = +1$ pero se clasifica como -1.

- Para $P_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_2 = +1$:

$$f(P_2) = \text{hardlims}((-0.7)(0)+(0.2)(-1)+0.5) = \text{hardlims}(0-0.2+0.5) = \text{hardlims}(0.3) = +1.$$

Paso cero...

(Ya lo último del principio)

La salida se calcula como:

$$f(x) = \text{hardlims}(W^\top P + b)$$

- Para $P_3 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, t_3 = -1$:

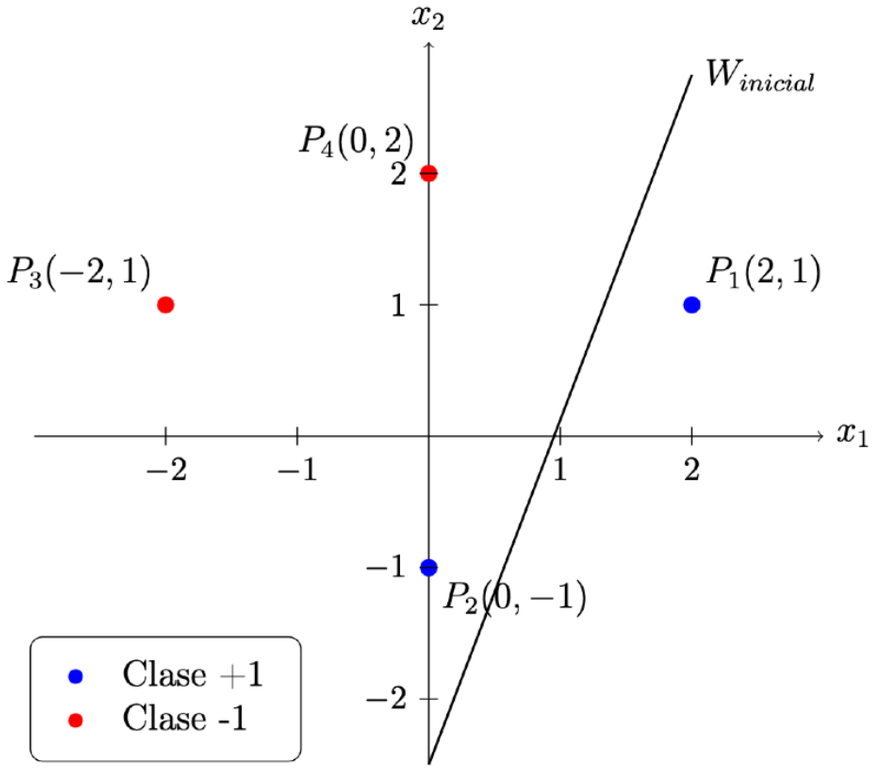
$f(P_3) = \text{hardlims}((-0.7)(-2)+(0.2)(1)+0.5) = \text{hardlims}(1.4+0.2+0.5) = \text{hardlims}(2.1) = +1.$ 

- Para $P_4 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, t_4 = -1$:

$f(P_4) = \text{hardlims}((-0.7)(0)+(0.2)(2)+0.5) = \text{hardlims}(0+0.4+0.5) = \text{hardlims}(0.9) = +1.$ 

Les mentí, ahora sí

Resumen:



Punto	Etiqueta Real	Salida $f(x)$	Clasificación
P_1	$+1$	-1	Incorrecta
P_2	$+1$	$+1$	Correcta
P_3	-1	$+1$	Incorrecta
P_4	-1	$+1$	Incorrecta

Paso 2:

Iteraciones

Iteración 1: Corregimos P_1:

$$W = \begin{bmatrix} -0.7 \\ 0.2 \end{bmatrix} + (+1) \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.3 \\ 1.2 \end{bmatrix}, \quad b = 0.5 + (+1) = 1.5.$$

Iteración 2: Corregimos P_3:

$$W = \begin{bmatrix} 1.3 \\ 1.2 \end{bmatrix} + (-1) \begin{bmatrix} -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.3 \\ 0.2 \end{bmatrix}, \quad b = 1.5 + (-1) = 0.5.$$

Iteración 3: Corregimos P_4:

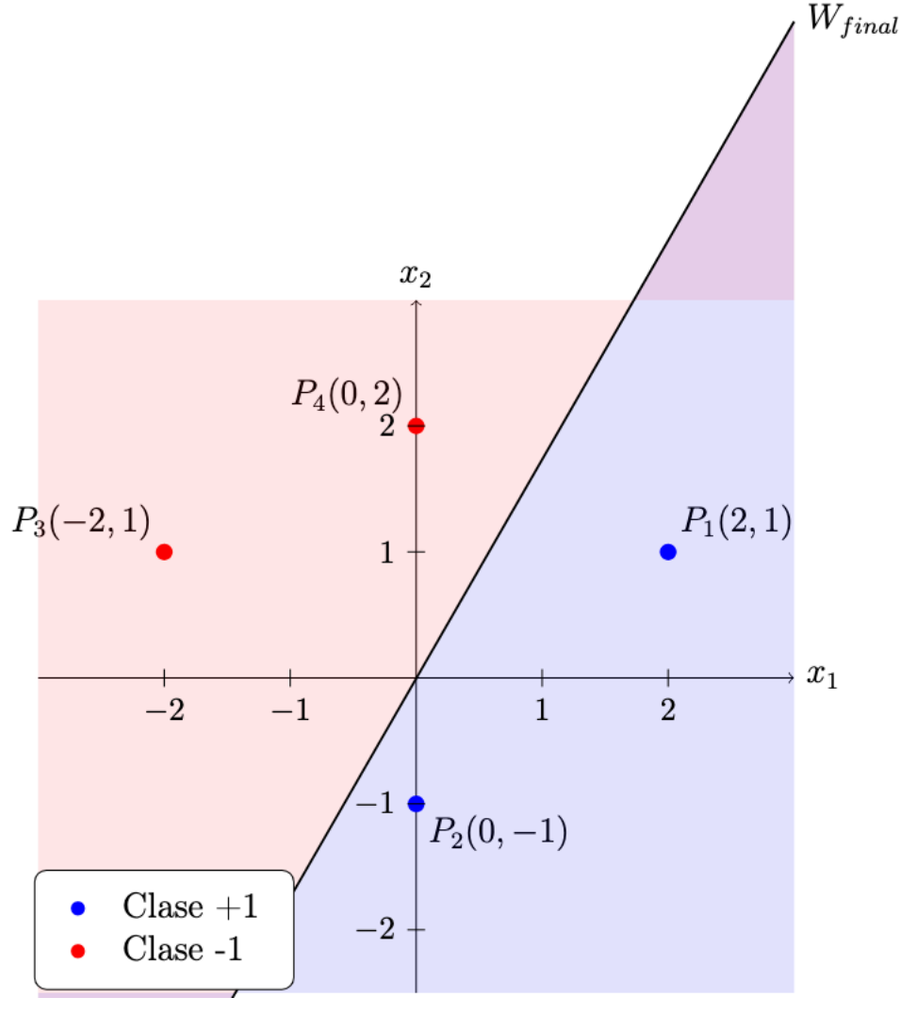
$$W = \begin{bmatrix} 3.3 \\ 0.2 \end{bmatrix} + (-1) \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.3 \\ -1.8 \end{bmatrix}, \quad b = 0.5 + (-1) = -0.5.$$

Verificación

Tarea:

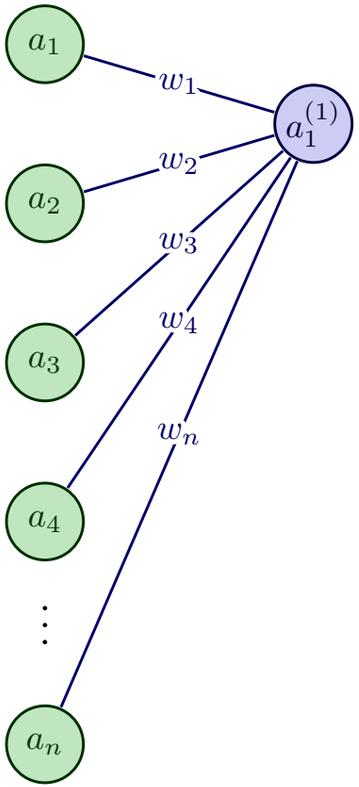
Verificar que la siguiente solución es válida para todos los puntos:

$$\text{Con } W = \begin{bmatrix} 3.3 \\ -1.8 \end{bmatrix} \text{ y } b = -0.5$$



Complicamos el asunto

Primero en variables... después en nodos

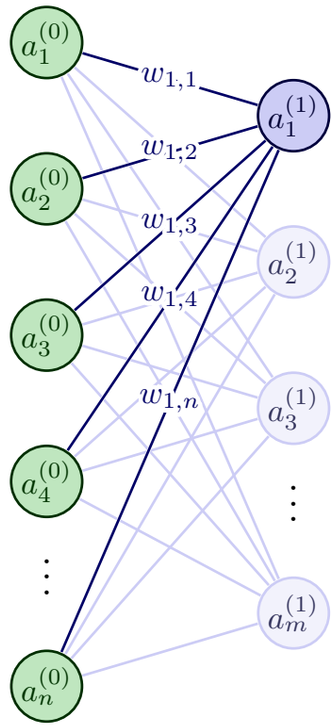


$$= f\left(w_1 a_1 + w_2 a_2 + \ldots + w_n a_n + b\right)$$

$$= f\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

$$a_1^{(1)} = f\left[\begin{pmatrix} w_1 & w_2 & \ldots & w_n \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} + b\right]$$

$$= f\left(\mathbf{w}^T\mathbf{a} + b\right)$$



$$= f\left(w_{1,1}a_1^{(0)} + w_{1,2}a_2^{(0)} + \ldots + w_{1,n}a_n^{(0)} + b_1^{(0)}\right)$$

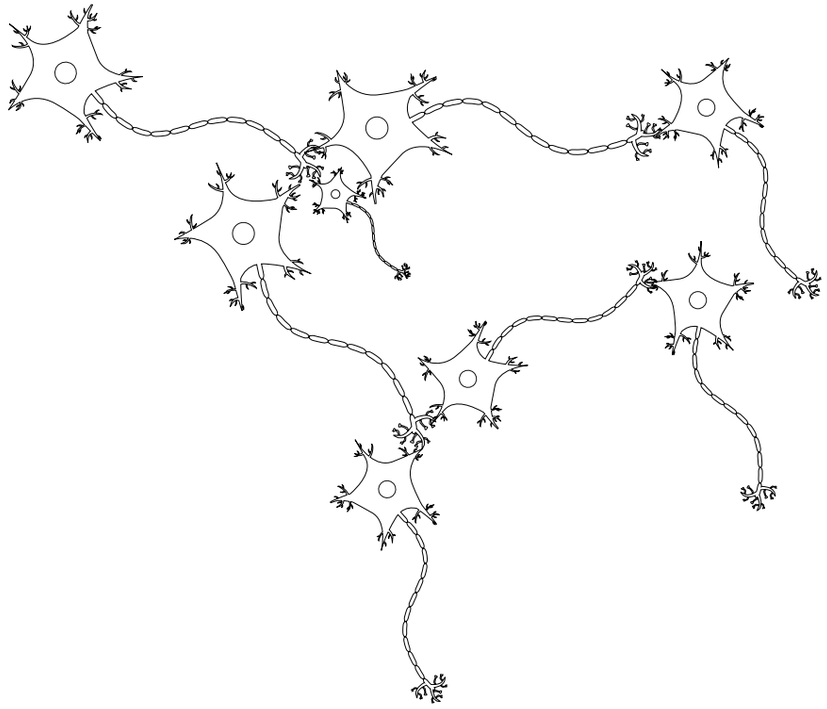
$$= f\left(\sum_{i=1}^{n} w_{1,i}a_i^{(0)} + b_1^{(0)}\right)$$

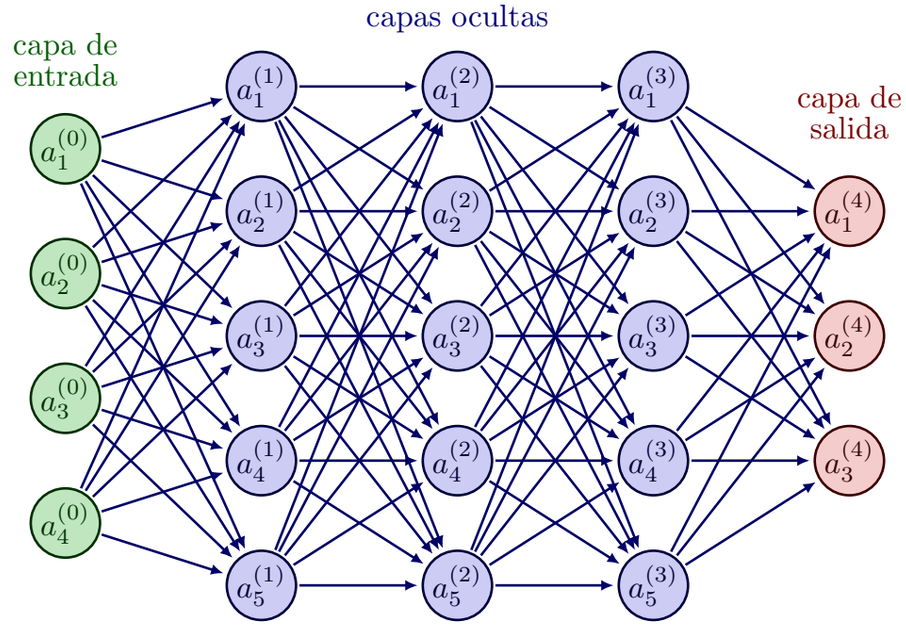
$$\begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_m^{(1)} \end{pmatrix} = f\left[\begin{pmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,n} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \ldots & w_{m,n} \end{pmatrix}\begin{pmatrix} a_1^{(0)} \\ a_2^{(0)} \\ \vdots \\ a_n^{(0)} \end{pmatrix} + \begin{pmatrix} b_1^{(0)} \\ b_2^{(0)} \\ \vdots \\ b_m^{(0)} \end{pmatrix}\right]$$

$$\mathbf{a}^{(1)} = f\left(\mathbf{W}^{(0)}\mathbf{a}^{(0)} + \mathbf{b}^{(0)}\right)$$

Una red de neuronas

Y luego en capas...





Recursos extra

Páginas, documentación, ejemplos

- El perceptrón desde cero
- Conceptos básicos de redes neuronales
- Una de las primeras propuestas del perceptrón:
- Documentación Python de PyTorch, Tensorflow (paquetería para hacer Aprendizaje Automatizado)
- Ejemplos sencillos