

# Redes Neuronales

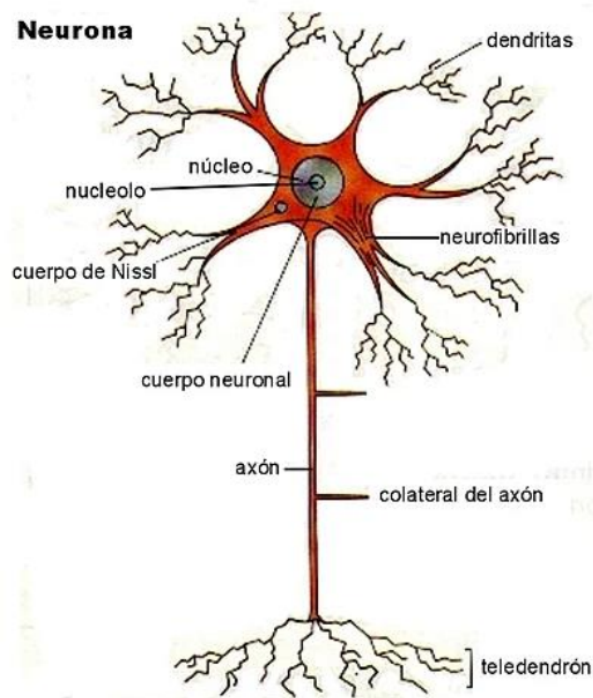
## Teóricas

- ☐ <https://drive.google.com/file/d/1sVlfxGLzAD2LQXowb8lz7IA2IWSk6YvR/view>
- ☐ [https://drive.google.com/file/d/18GhcgB\\_\\_p9D8cFLLoyycgFzgDedb0DUQ/view](https://drive.google.com/file/d/18GhcgB__p9D8cFLLoyycgFzgDedb0DUQ/view)

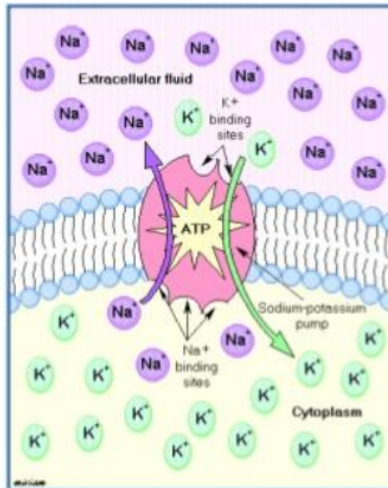
## Prácticas

- ☐ <https://drive.google.com/file/d/1UYVuij5qTbLCKplX2CdfQAAiuDIbZNFx/view?usp=sharing>

## ▼ Intro de que son las neuronas

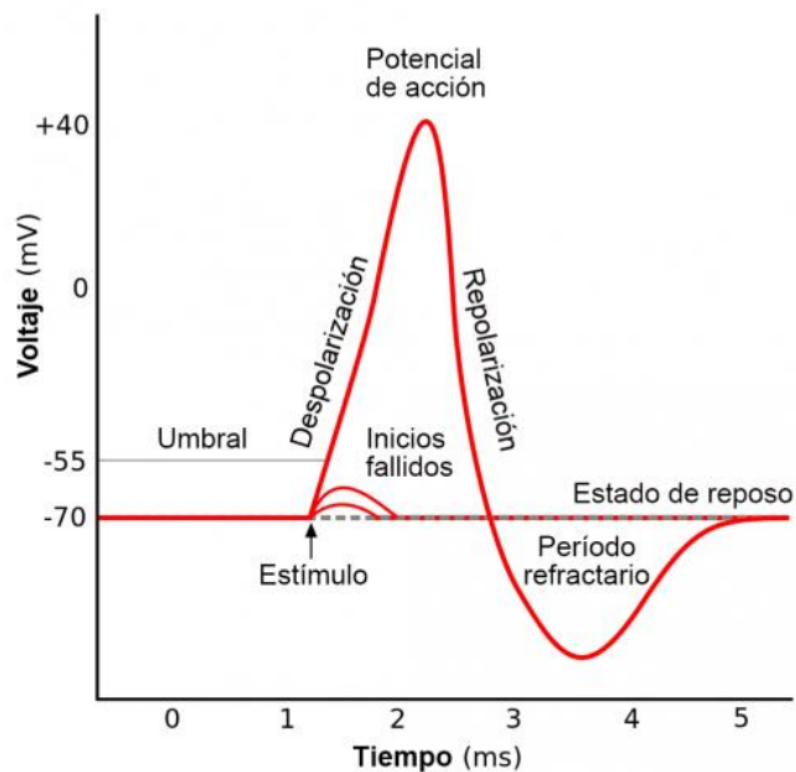


Las neuronas permiten modelizar el conocimiento y el aprendizaje en el cerebro.

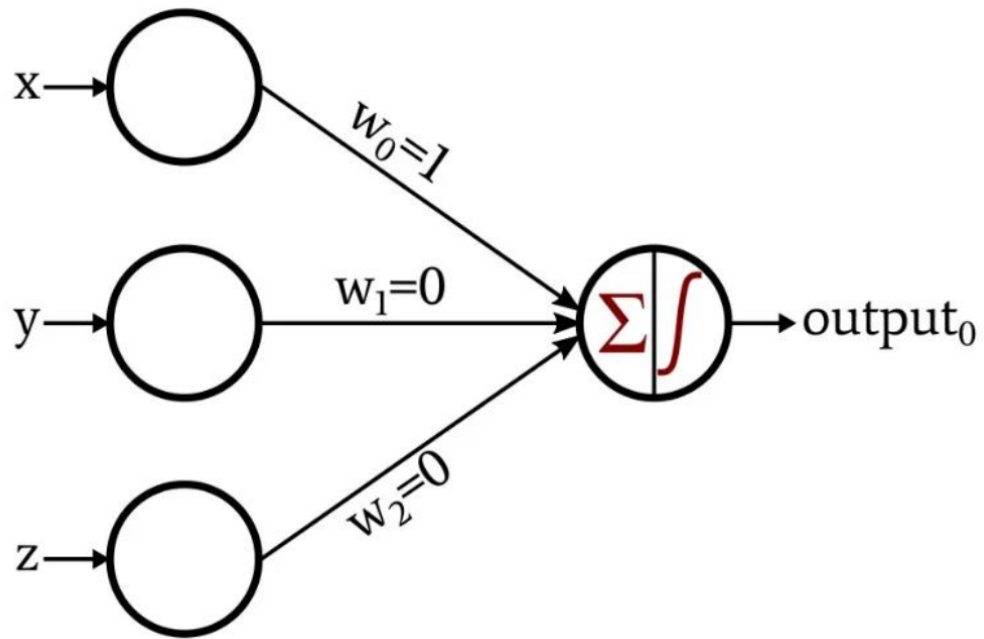


- Lo anterior permite que haya diferencias de cargas entre el exterior (+) y el interior (-) de la neurona: **POLARIDAD**.
- La diferencia de carga está dada por la concentración de iones.
- Hay mayor concentración de  $\text{Na}^+$  fuera de la membrana y mayor concentración de  $\text{K}^+$  dentro de la misma
- Esto es posible gracias a la **bomba de sodio-potasio (transporte activo)**.

## EL GRADIENTE IONICO LO LOGRA GRACIAS A LA BOMBA DE SODIO-POTASIO



## Redes neuronales artificiales



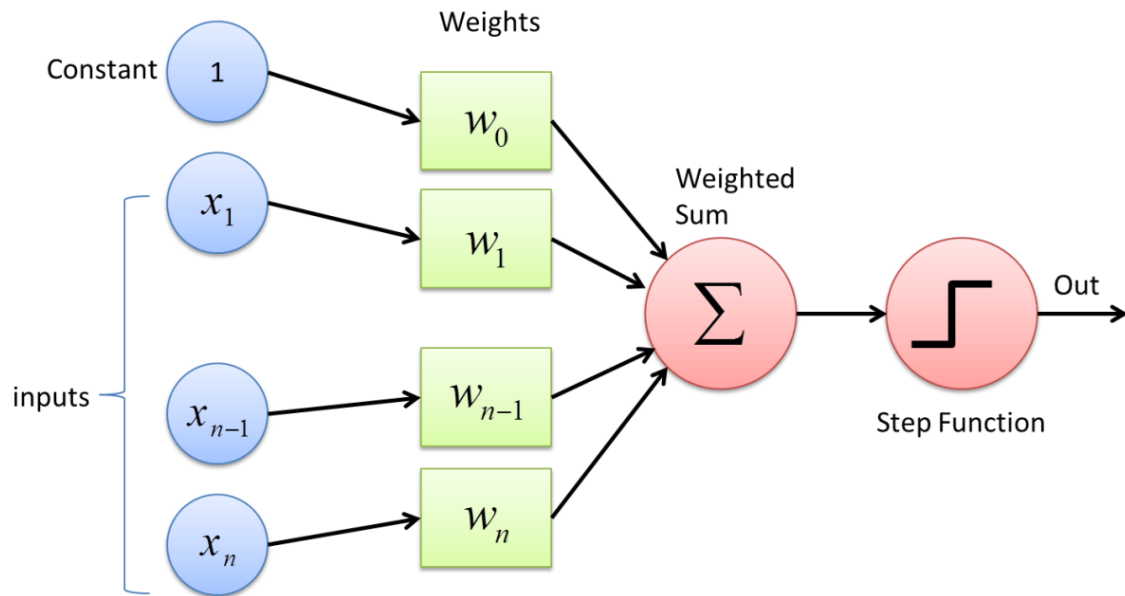
Inputs:  $x, y, z$

Pesos:  $w_0, w_1, w_2$

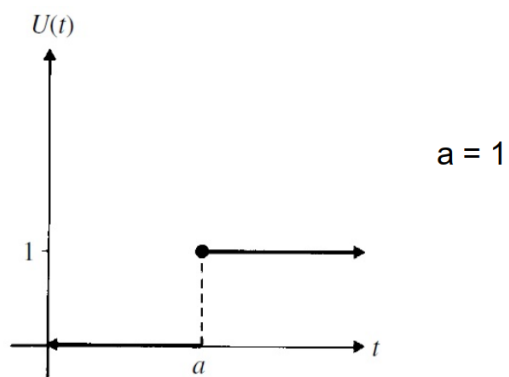
Funcion de activacion

Output

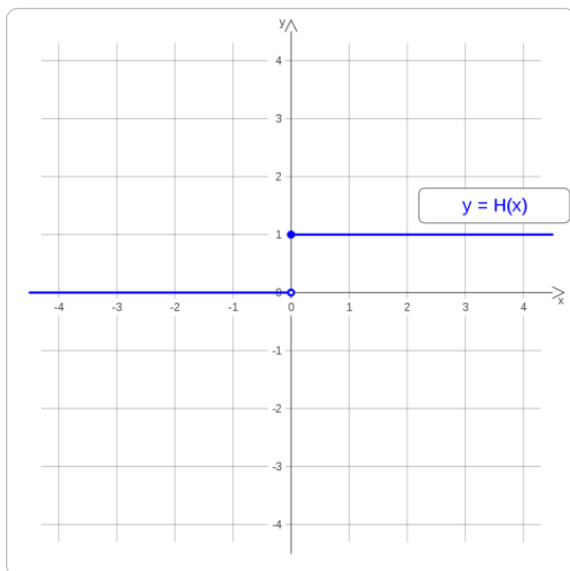
**Neurona individual → Perceptron**



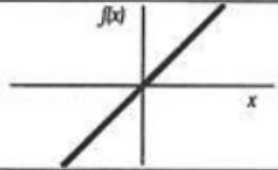
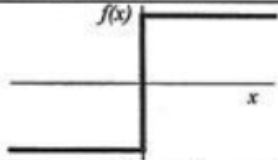
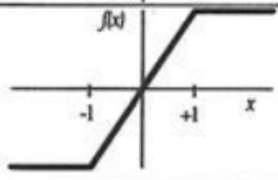
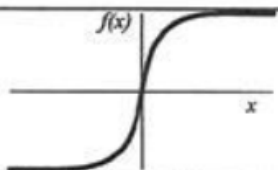
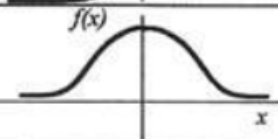
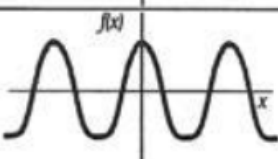
Función escalón



## Heaviside - Función escalón

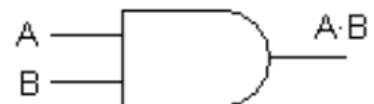


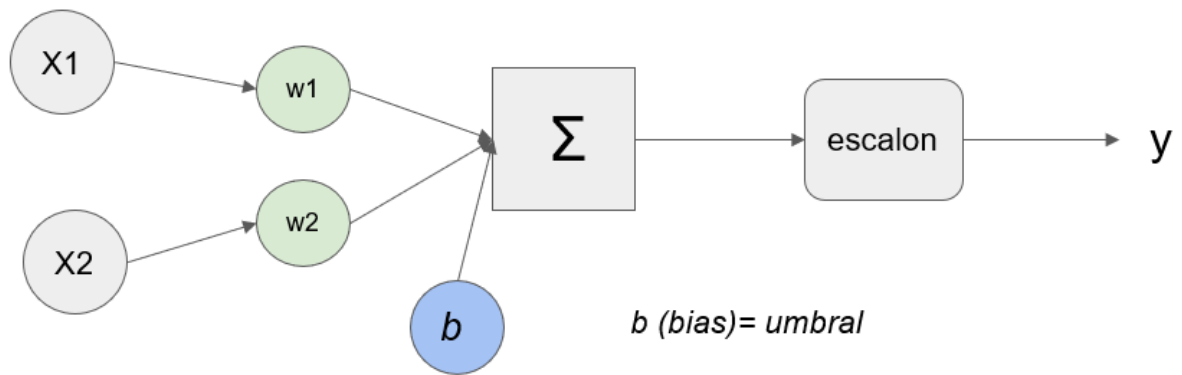
$$f(x) \left\{ \begin{array}{l} \bullet \text{ 1 si } x \geq 0 \\ \bullet \text{ 0 para todo otro valor} \end{array} \right.$$

	Función	Rango	Gráfica
<b>Identidad</b>	$y = x$	$[-\infty, +\infty]$	
<b>Escalón</b>	$y = \text{sign}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
<b>Lineal a tramos</b>	$y = \begin{cases} -1, & \text{si } x < -l \\ x, & \text{si } -l \leq x \leq +l \\ +1, & \text{si } x > +l \end{cases}$	$[-1, +1]$	
<b>Sigmoidea</b>	$y = \frac{1}{1+e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
<b>Gaussiana</b>	$y = Ae^{-Bx^2}$	$[0, +1]$	
<b>Sinusoidal</b>	$y = A \text{sen}(\omega x + \varphi)$	$[-1, +1]$	

## Perceptrón simple - AND

COMPUERTA AND		
A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1





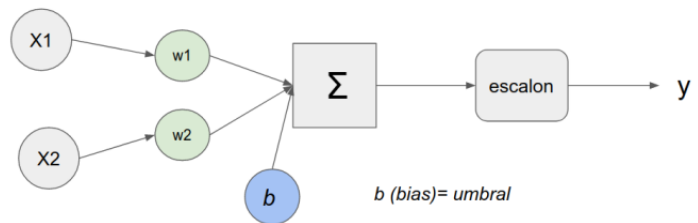
$$X1 * w1 + x2 * w2 + b \geq 0 \Rightarrow y = 1$$

$$X1 * w1 + x2 * w2 + b < 0 \Rightarrow y = 0$$

La tabla de verdad vendría a ser mi conjunto de entrenamiento, donde tengo dos valores posibles  $x1$  y  $x2$

y vendría a ser la variable dependiente que quiero calcular

X1	X2	y
0	0	0
0	1	0
1	0	0
1	1	1



$$X1 * w1 + x2 * w2 + b \geq 1 \Rightarrow y = 1$$

$$X1 * w1 + x2 * w2 + b < 1 \Rightarrow y = 0$$

## Perceptrón simple - Conjunto etiquetado

Datos de entrenamiento

X1	X2	y
0	0	0
0	1	0
1	0	0
1	1	1

Etiquetas. Valores esperados de salida

X1	X2	$x1*w1+x2*w2 + b$	z	y
0	0	b	$< 0$	0
0	1	$w2 + b$	$< 0$	0
1	0	$w1 + b$	$< 0$	0
1	1	$w1 + w2 + b$	$\geq 0$	1

En z es lo que debería pasar para obtener esos valores de y

### ¿Como calculo w1, w2 y b?

Los seteamos de forma aleatoria y lo vamos refinando

### Perceptrón simple - Primera Iteración



- $W1 = 0.3$
- $W2 = 0.2$
- $b = -1$

X1	X2	$x1*w1+x2*w2 + b$	z	y'	y	Error ( y-y' )
0	0	-1	-1	0	0	0
0	1	$w2 + b$	-0.8	0	0	0
1	0	$w1 + b$	-0.7	0	0	0
1	1	$w1 + w2 + b$	-0.5	0	1	1

Tenemos que mejorar estos valores

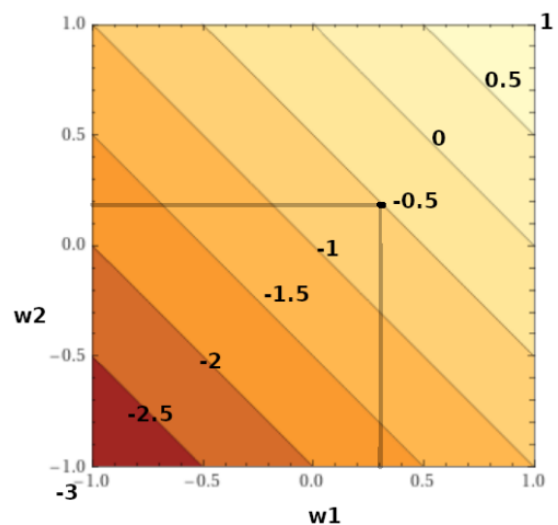
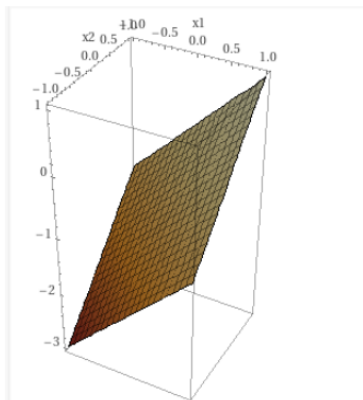
La ultima hace que este perceptron no represente una compuerta AND

## Función del error

- $W1 = 0.3$
- $W2 = 0.2$
- $b = -1$

$$\text{Escalon}(w1 + w2 - 1) - 1 = \text{error}$$

$$\text{Si } \begin{cases} w1 + w2 - 1 < 0 \Rightarrow 0 \\ w1 + w2 - 1 \leq 0 \Rightarrow \end{cases}$$



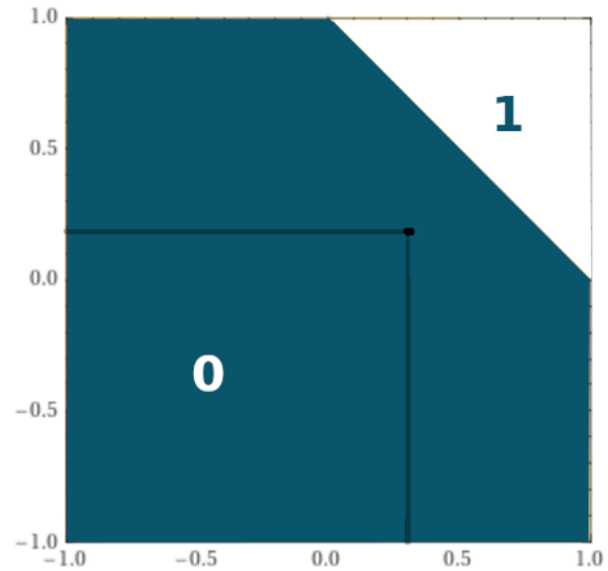
La funcion escalon va a devolver cero para todo valor  $\leq 0$

## Función del error: Escalón

- $W1 = 0.3$
- $W2 = 0.2$
- $b = -1$

$\text{Escalon}(w1 + w2 - 1) - 1 = \text{error}$

$$\text{Si } \begin{cases} w1 + w2 - 1 < 0 \Rightarrow 0 \\ w1 + w2 - 1 \leq 0 \Rightarrow 1 \end{cases}$$

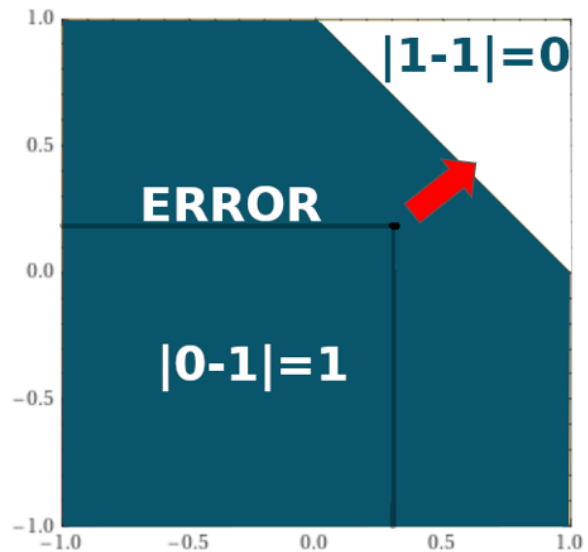


## Función del error:

- $W1 = 0.3$
- $W2 = 0.2$
- $b = -1$

**Error:**

$$\text{Si } \begin{cases} w1 + w2 - 1 < 0 \Rightarrow |0 - 1| = 1 \\ w1 + w2 - 1 \leq 0 \Rightarrow |1 - 1| = 0 \end{cases}$$



**Dirección de decrecimiento**

# Actualización de los pesos

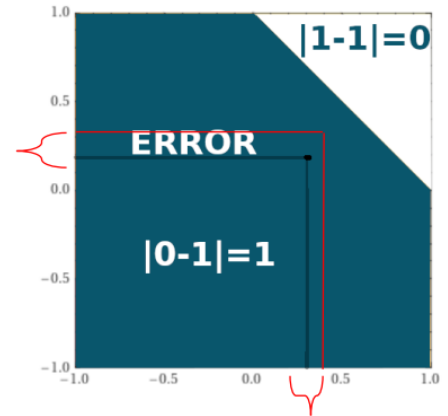
$\alpha$  = tasa de crecimiento.

Valor entre 0 y 1.

Suele ser 0.5

$\alpha = 0.2$

- $W1 = 0.3 + 0.2 * \text{error}$
- $W2 = 0.2 + 0.2 * \text{error}$
- $b = -1$



A medida que se acerque a cero, más pequeña será la actualización

## Perceptrón simple - Segunda Iteración

- $W1 = 0.3 + 0.2 = 0.5$
- $W2 = 0.2 + 0.2 = 0.4$
- $b = -1$

X1	X2	$x1*w1+x2*w2 + b$	z	$y'$	y	Error ( $ y-y' $ )
0	0	-1	-1	0	0	0
0	1	$w2 + b$	-0.6	0	0	0
1	0	$w1 + b$	-0.5	0	0	0
1	1	$w1 + w2 + b$	-0.1	0	1	1

## Perceptrón simple - Tercera Iteración

- $W1 = 0.5 + 0.2 = 0.7$

- $W2 = 0.4 + 0.2 = 0.6$
- $b = -1$

X1	X2	$x1*w1+x2*w2 + b$	z	y'	y	Error ( $ y-y' $ )
0	0	-1	-1	0	0	0
0	1	$w2 + b$	-0.4	0	0	0
1	0	$w1 + b$	-0.3	0	0	0
1	1	$w1 + w2 + b$	0.3	1	1	0

## Perceptrón simple - AND en producción

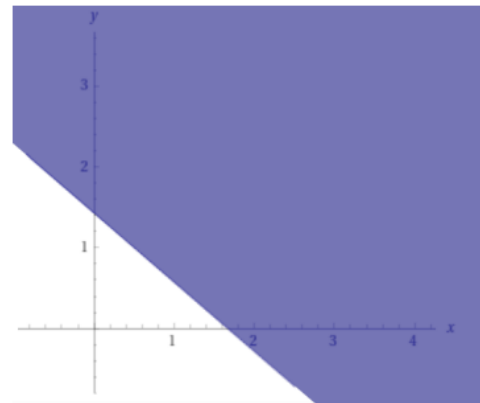
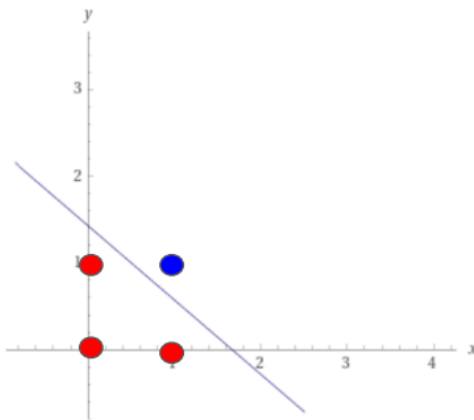
- $W1 = 0.7$
- $W2 = 0.6$
- $b = -1$

Una vez entrenada, lo pruebo con otros valores de entrada, que estan fuera de mi conjunto de entrenamiento

X1	X2	z	AND= y'
0.5	0.5	-0.35	0
0.5	1	-0,05	0
0.9	0.9	0.17	1

Con 0.9 que son valores cercanos a 1, devuelve 1  $\Rightarrow$  esta generalizando el conocimiento.

$$x_1 * 0.7 + x_2 * 0.6 - 1$$



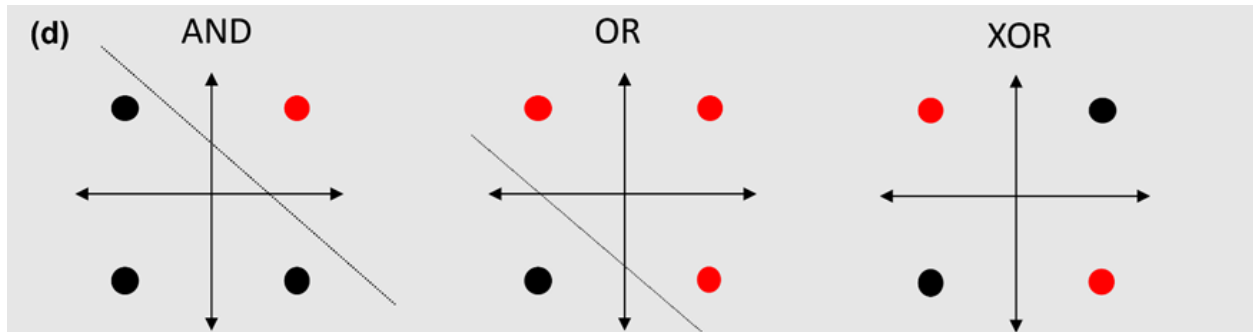
## Perceptrón simple - Almacenamiento

$$\begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = x_1 * w_1 + x_2 * w_2 + b$$

Modelo entrenado = matriz de números flotantes

$$\begin{bmatrix} 0.7 \\ 0.6 \\ -1 \end{bmatrix}$$

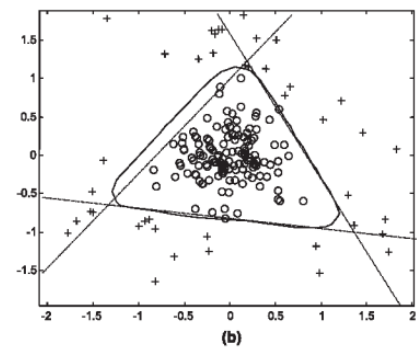
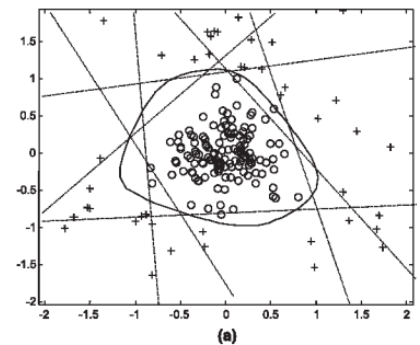
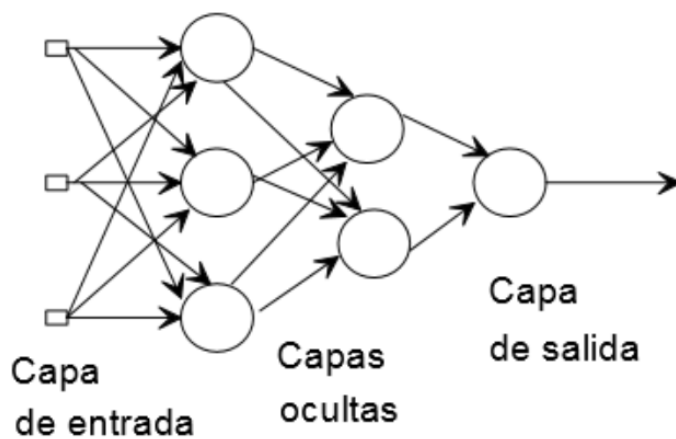
## Perceptrón simple - Limitaciones



Los problemas que no son linealmente separables no los puedo resolver con un perceptron simple.

Necesito agregar otra neurona.

## Perceptrón Multicapa



- Las redes neuronales se optimizan mediante descenso de gradiente. \_\_V\_\_

Si todas se optimizan con el descenso de gradiente

- Las redes neuronales son sensibles a la escala de los datos. \_\_V\_\_

**Las redes neuronales necesitan un mayor preprocesamiento de los datos**, siendo bastante sensibles a las distintas escalas de las variables. Suelen necesitar mayor volumen de datos para el entrenamiento del modelo y requieren de alta capacidad de recursos computacionales.

- Un perceptrón simple no puede modelar compuertas NAND. \_\_F\_\_

Si puede modelar compuertas NAND

Porque una puerta NAND es igual que la puerta AND, pero con las mitades intercambiadas.

No lo que no puede modelar es una compuerta XOR

- Las redes neuronales muy complejas pueden sub-ajustar, esto se soluciona con métodos de

regularización como Early stopping. \_\_F\_\_

Puede sobreajustar (overfitting)

- a) El perceptrón simple no permite dividir el espacio linealmente: \_\_F\_\_

Si lo permite, lo que no permite es dividir xor que no es linealmente separable, pero and y or que si lo son si les puede dividir.

- b) Las redes SOM necesitan un conjunto de datos balanceado y correctamente etiquetado

para su entrenamiento: \_\_\_\_F\_\_\_\_

No necesita eso

- c) Para poder entrenar una red neuronal con el algoritmo Backpropagation, las funciones

de activación de las neuronas deben si o si ser derivables: \_\_\_\_V\_\_\_\_

SI

d) Backpropagation es un método alternativo al método de descenso por gradiente: \_\_\_\_F\_\_\_\_

Aplica descenso

---

## Ventajas

- **Obtienen resultados con una alta precisión.**
- **El procesamiento de la información es local**, es decir que al estar compuesto por unidades individuales de procesamiento, dependiendo de sus entradas y pesos, y de que todas las neuronas de una capa trabajan en forma paralela, proporcionan una respuesta al mismo tiempo.
- **Los pesos son ajustados basándose en la experiencia**, lo que significa que se le tiene que enseñar a la red lo que necesita saber antes de ponerla en funcionamiento.
- **Las neuronas son tolerantes a fallos**, si parte de la red no trabaja, solo dejará de funcionar la parte para que dicha neurona sea significativa, el resto tendrá su comportamiento normal.
- **Las neuronas pueden reconocer patrones que no han sido aprendidos**, sólo deben tener cierto parecido con el conocimiento previo que tenga la red. Dicho de otra forma: si la entrada presenta alguna alteración la red podrá identificarla siempre y cuando se mantenga cierto grado de similitud entre lo aprendido y lo mostrado en la entrada de la red.

## Desventajas

- **Las redes neuronales necesitan un mayor preprocesamiento de los datos**, siendo bastante sensibles a las distintas escalas de las variables. Suelen necesitar



mayor volumen de datos para el entrenamiento del modelo y requieren de alta capacidad de recursos computacionales.

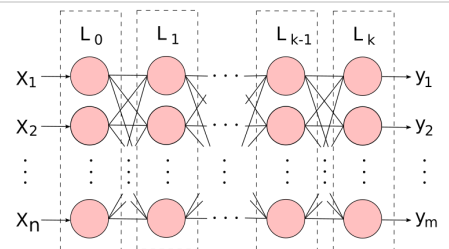
- **Complejidad de aprendizaje para grandes tareas**, cuanto más se necesite que aprenda una red, más complicado será enseñarle.
- **Tiempo de aprendizaje elevado**. Esto depende de dos factores: primero si se incrementa la cantidad de patrones a identificar o clasificar, y segundo, si se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos, se deberá invertir más tiempo en lograr que la red converja a valores de pesos que representen lo que se quiera enseñar.
- **No son fácilmente explicables**. Conocer las reglas o motivos por los que la red devuelve esos resultados no suele ser fácil y precisa de otras analíticas.

## Links

inmensia

De igual forma que las puertas lógicas se conectan para crear circuitos lógicos, las neuronas artificiales se pueden conectar para crear redes neuronales artificiales. Al conectar una neurona con

 <https://inmensia.com/page/2/>



Redes neuronales

<https://albertotb.com/curso-ml-R/Rmd/12-nn/12-nn.html#1>

## Practica

**Descenso por gradiente:** Es un metodo para hallar parametros del modelo, buscando que siempre se minimice el error (el error de una funcion de costo). Cuanto se separa el valor predicho del real

**Capas densas:** son capas donde la salida de una van a parar a todas las neuronas de la capa siguiente

### **SGD descenso por gradiente estocastico**

Consiste en hacer descenso por gradiente tomando un muestreo de las observaciones

### **Epocas**

cantidad de iteracion de entrenamiento

cantidad de iteraciones de pasadas que le hago al dataset completo

En cada epoca el dataset o separamos de a baches, de a lotes