

SESIÓN 01: INTRODUCCIÓN A DEEP LEARNING

10 de Setiembre 2022

Índice

01 Motivación

02 Redes neuronales artificiales

2.1 Entendiendo una red neuronal

2.2 Funciones de activación

03 Descenso de gradiente

04 El problema del overfitting

01

Motivación

Motivación

Problema 1:

Un empresa dedicada a la piscicultura vende lotes de peces; no obstante, este proceso es manual y genera problemas al contar los peces, lo cual afecta sus ventas. ¿Qué solución podemos proponer?



Problema 2:

Sabemos que pocas personas saben lengua de señas y esto genera una barrera de comunicación. ¿Qué podemos Hacer?

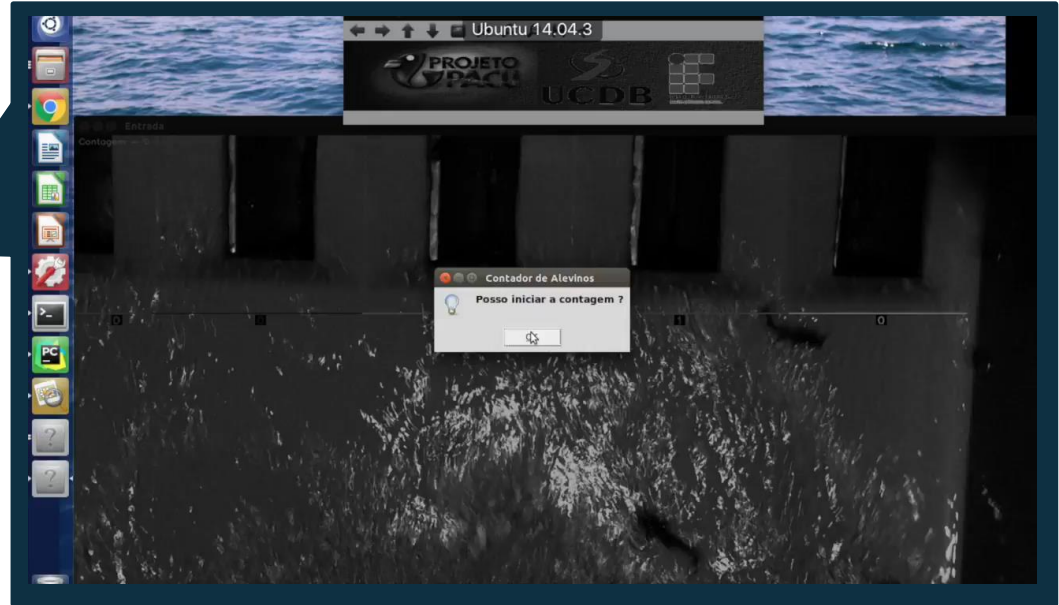


Motivación

Solución Problema 1

Data Product: API para el conteo automático de peces.

Error de 0.01%. Optimizó las ventas de la empresa.



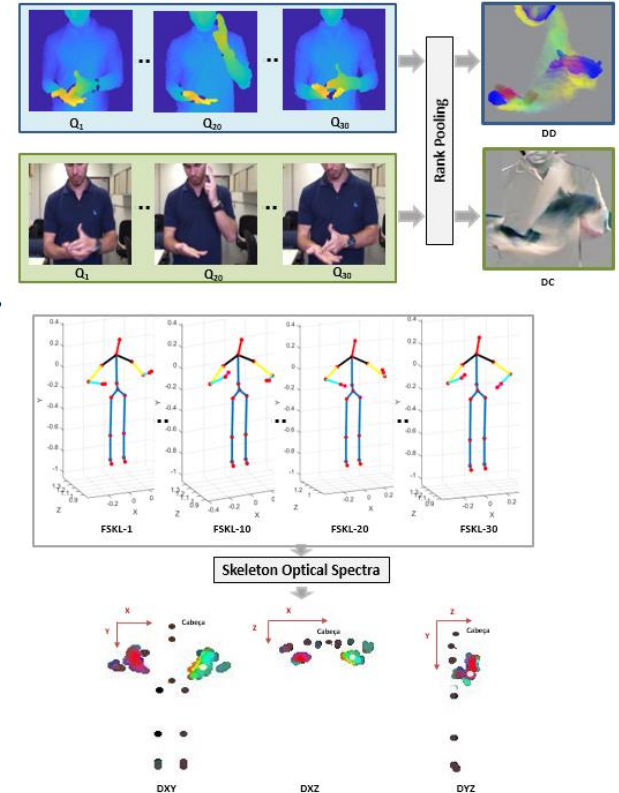
Motivación

Solución Problema 2

Proyecto de Inversión Social para mejorar la calidad de vida de personas sordomudas. Sistema de traducción automática de lengua de señas.

```
Editor - B:\EDWIN\DISERTACION\DOCUMENTOS COMPARTIDOS\MESTRAO\CODIGOS DISERTACION\LIBRAS KINET FOR WINDOWS\LoadPalavra.m

function LoadFrame()
2 -   close all;clear;clc;
3   endereco = pwd;
4   % endereco=uiinput('');
5   endereco = 'B:\EDWIN\DISERTACION\DOCUMENTOS COMPARTIDOS\Bases Libras Setembro 2013\01Primeira\01Pessoa\';pwd
6   endereco = 'B:\EDWIN\DISERTACION\DOCUMENTOS COMPARTIDOS\Bases Libras Setembro 2013\02Segunda\01Pessoa\';
7   %
8   calibparam.cx = 140;
9   calibparam.fy = 236.502;
10  calibparam.fx = 250.494;
11  %
12  options.calibparam = calibparam;
13  options.imgBorder = 150;
14  options.maxSize = 350;
15  %
16  [filename, pathname, filterindex] = uigetfile('*.mat', 'Escolha a palavra');
17  modelo = load(pathname filename);
18  disp(['Arquivo: ' filename]);
19  modelo = modelo.modelo;
20  [lin, col] = size(modelo);
21  %
22  disp('Carregando Video...');
23  cont = 0;
24  tamanho=iget(0,'ScreenSize');
25  figure('position',[tamanho(1) tamanho(2) tamanho(3) tamanho(4)]);
26  trajMacL = [];
27  trajHead = [];
28  trajCorvevelL = []; %Nodo L
29  trajPunhoL = []; %Nudeca L
30  trajCorvevelR = []; %Nodo R
31  trajPunhoR = []; %Nudeca R
32  trajMacR = [];
33  trajPeito = []; % pecho
34  lineWidth = [];
35  listRGB = [];
36  z = 480;
37  c = 640;
38  tic
39  %
40  for i = 1:lin
41      [imgRGB,imgDepth, pos, BW, area] = LoadQuadro(modelo(i), options);
```

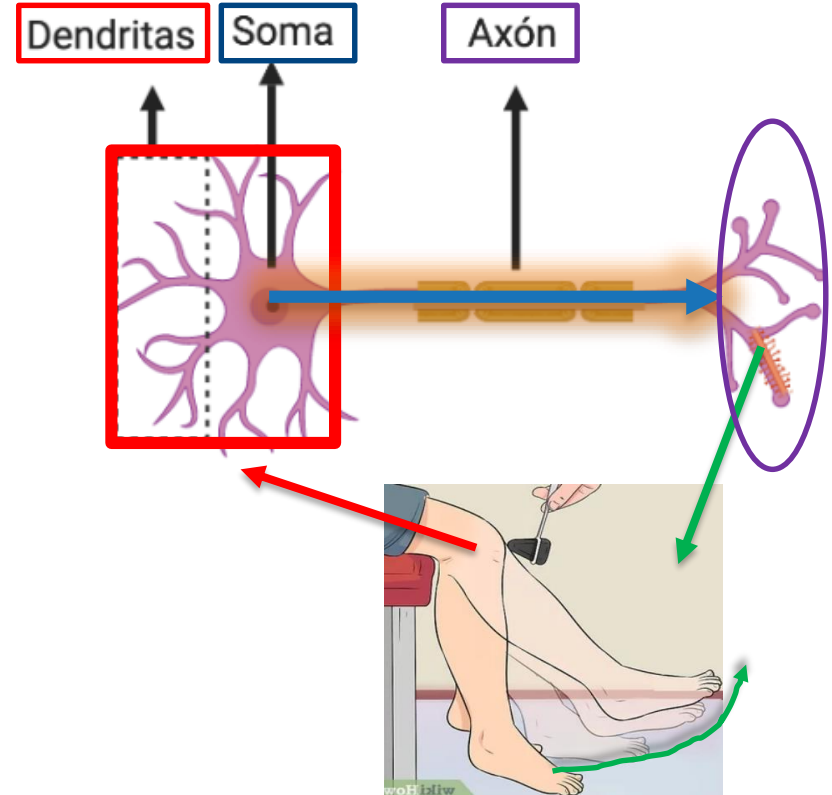


02

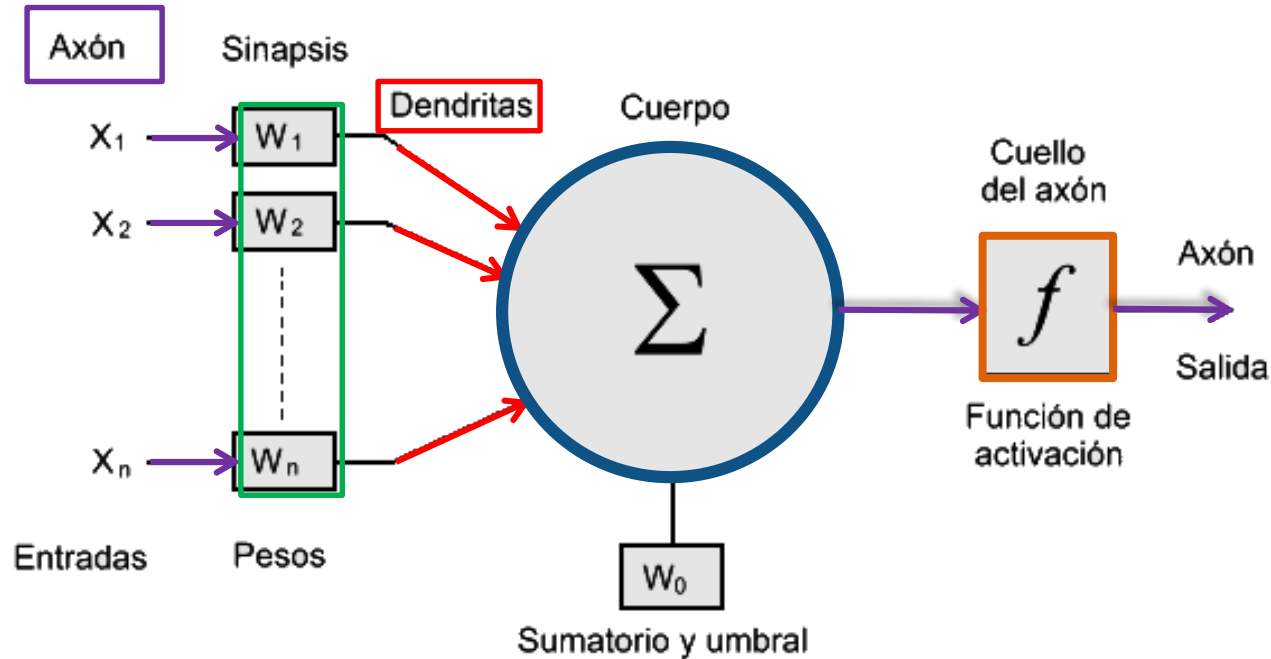
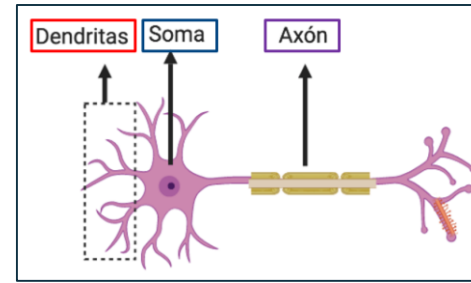
Redes Neuronales Artificiales

Neurona vs perceptrón

- La comunicación entre neuronas se llama **sinapsis**.
- Neuronas **simples** se combinan para generar neuronas más complejas.
- Una neurona posee tres partes: **dendritas**, cuerpo celular o **soma**, y el **axón**.
- Cuando la dendrita recibe un **estímulo** que **supera** cierto **umbral**, esta se **activa** y libera una **acción** que se transmite a través del axón.



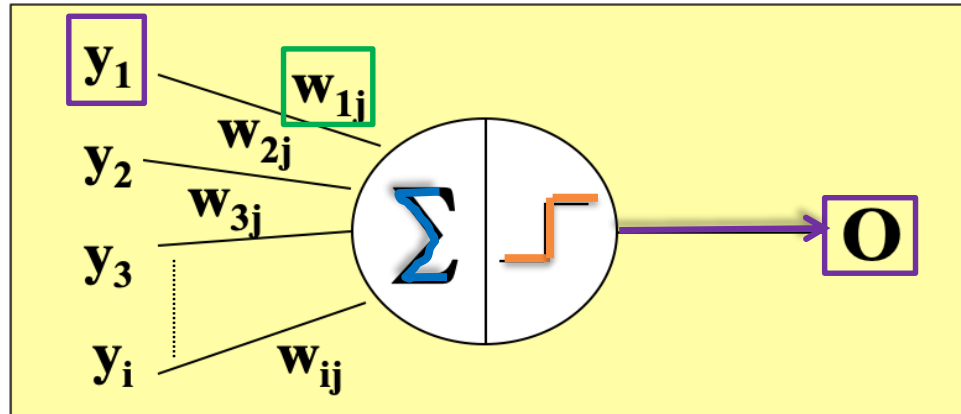
Neurona vs perceptrón



Redes Neuronales Artificiales

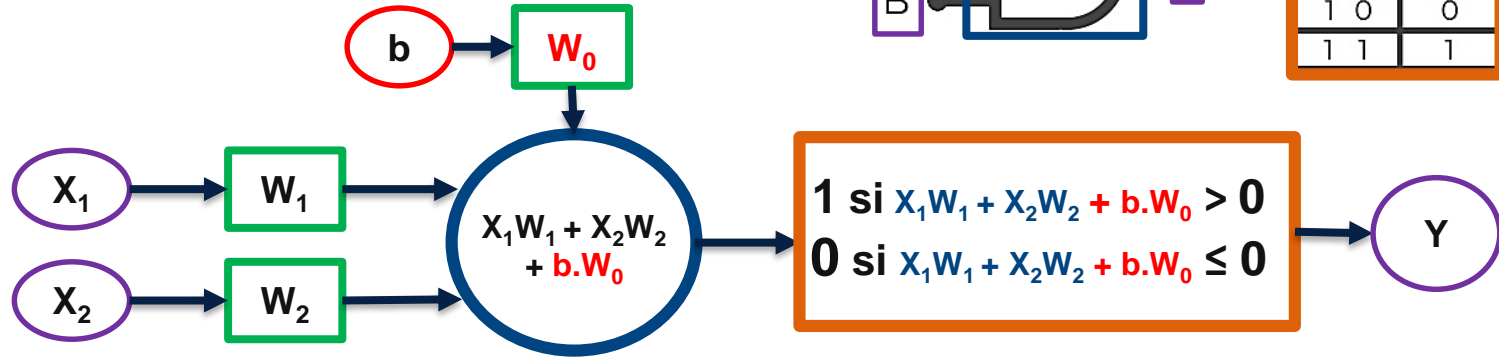
En las Redes Neuronales Artificiales (ANN) tenemos que:

- Cada neurona tiene un valor **umbral**
- Cada neurona tiene **entradas ponderadas** de otras neuronas
- Las señales de entrada forman una **suma ponderada**
- Si el nivel de activación excede el **umbral**, la neurona "**dispara**"



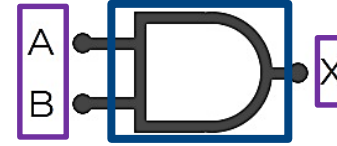
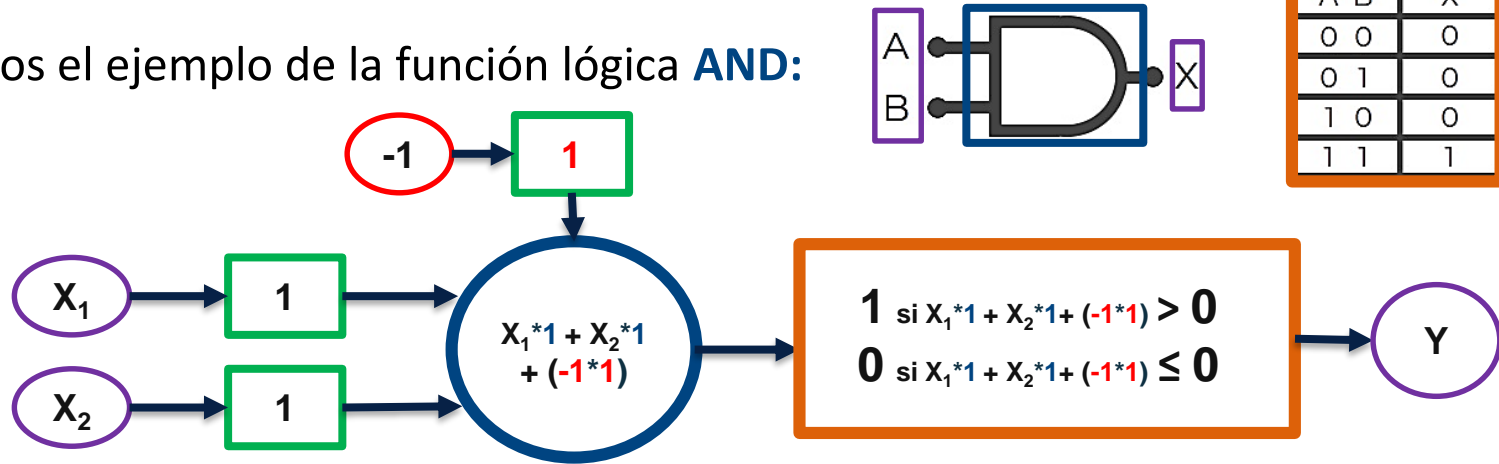
Redes Neuronales Artificiales

Veamos el ejemplo de la función lógica **AND**:



Redes Neuronales Artificiales

Veamos el ejemplo de la función lógica **AND**:



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

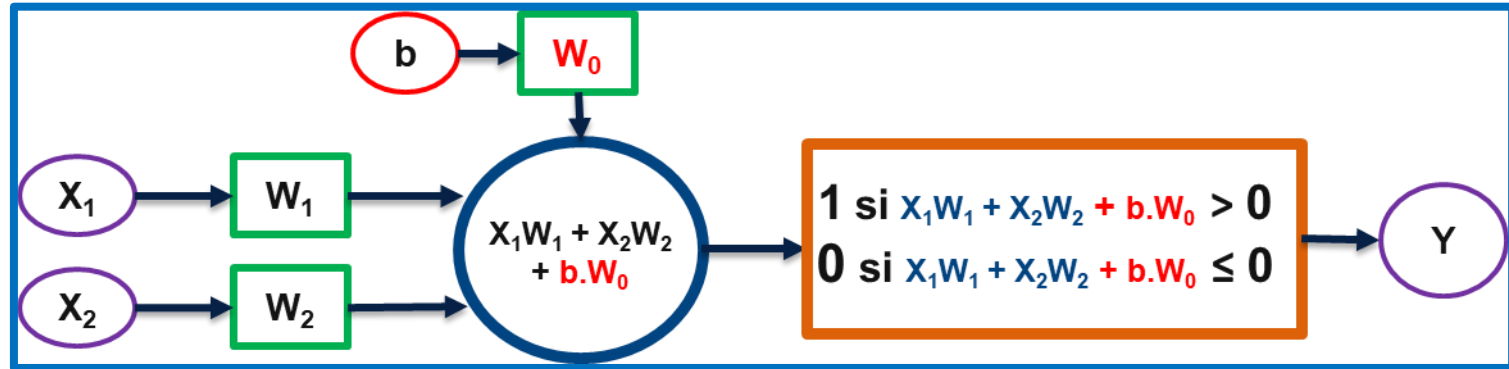
b	X_1	X_2	SUMA PONDERADA	SALIDA
-1	0	0	$(-1 * 1) + (0 * 1) + (0 * 1) = -1$	0
-1	0	1	$(-1 * 1) + (0 * 1) + (1 * 1) = 0$	0
-1	1	0	$(-1 * 1) + (1 * 1) + (0 * 1) = 0$	0
-1	1	1	$(-1 * 1) + (1 * 1) + (1 * 1) = 1$	1

Redes Neuronales Artificiales

Problemas simples, requieren de modelos simples. Pero, sabemos que nuestra **realidad es compleja** y su abstracción requiere de modelos más avanzados.

Es aquí donde recordamos el enunciado anterior:

“Neuronas simples se combinan para generar neuronas más complejas”

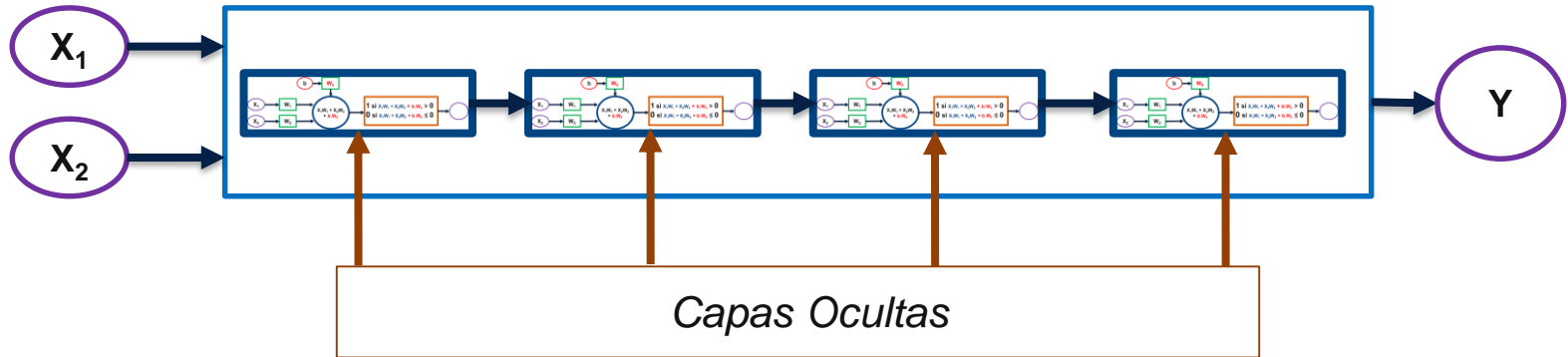


Redes Neuronales Artificiales

Problemas simples, requieren de modelos simples. Pero, sabemos que nuestra **realidad es compleja** y su abstracción requiere de modelos más avanzados.

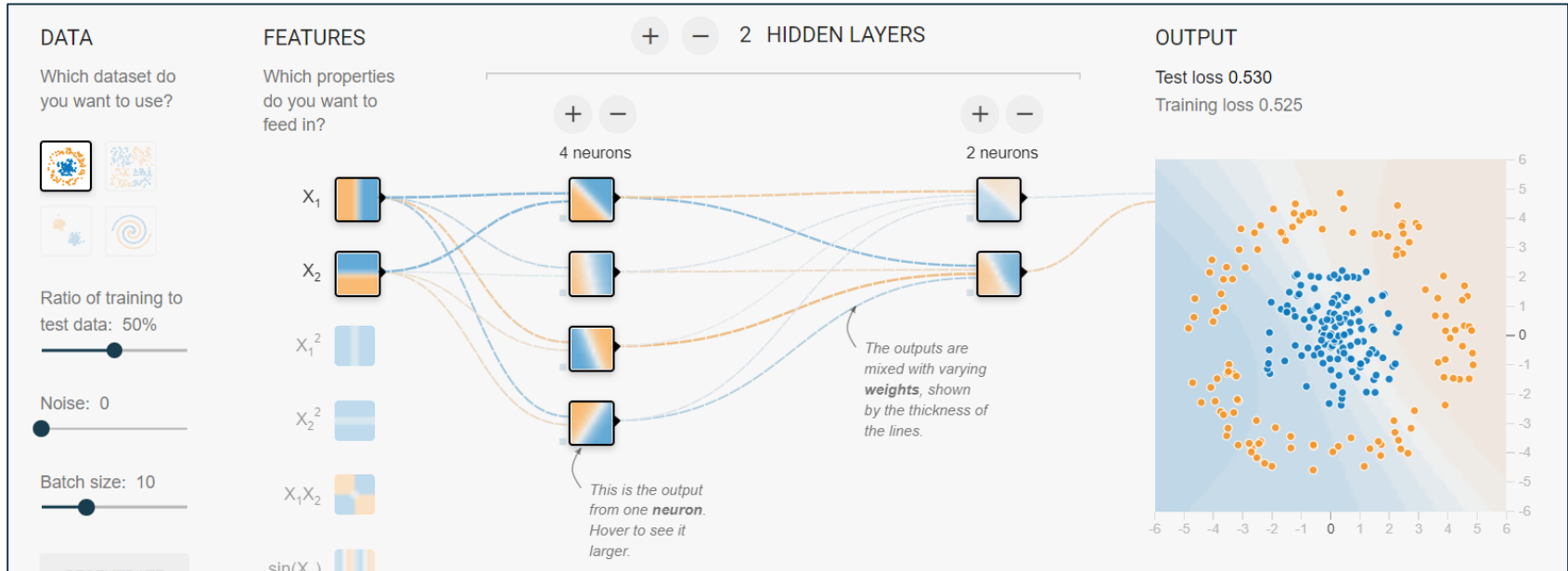
Es aquí donde recordamos el enunciado anterior:

“Neuronas simples se combinan para generar neuronas más complejas”



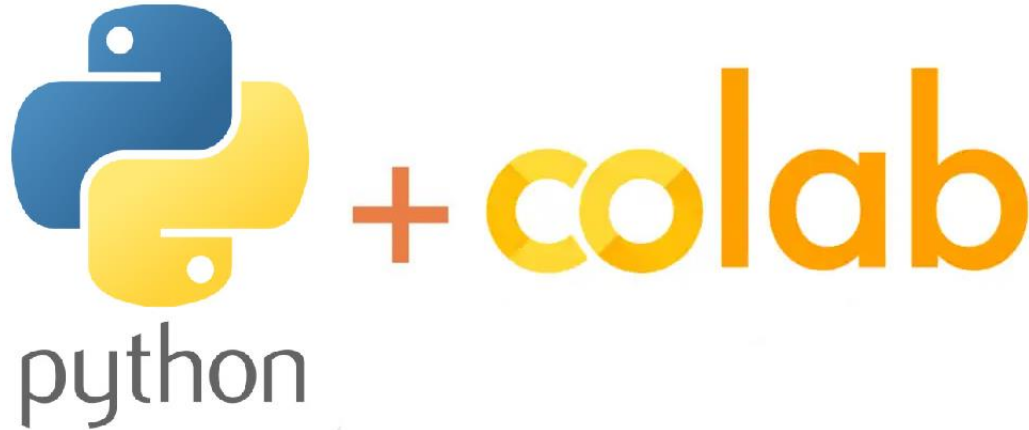
Redes Neuronales Artificiales

Con este nuevo concepto de **capas ocultas**, podemos resolver problemas más complejos. Observemos el siguiente [SIMULADOR](#) para analizar algunos ejemplos.



Redes Neuronales Artificiales

Programemos un poco en **Google colab**.

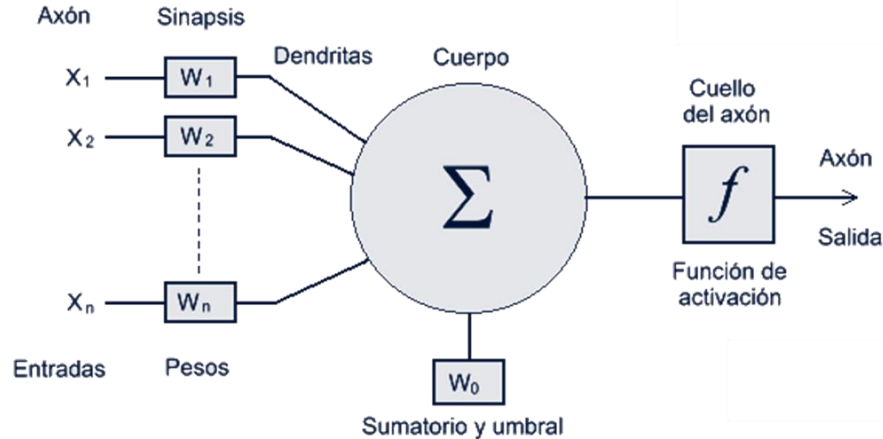


02

Funciones de Activación

Funciones de Activación

En este punto, hemos entendido el funcionamiento general de una ANN. Es el momento de profundizar un poco.

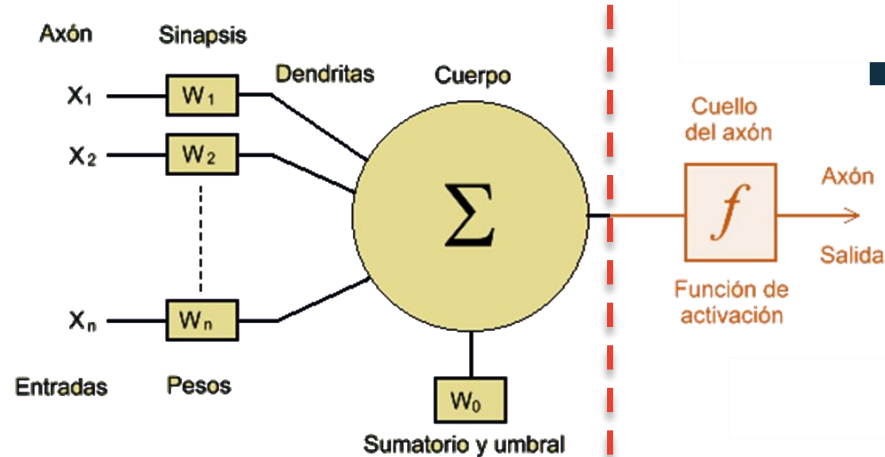
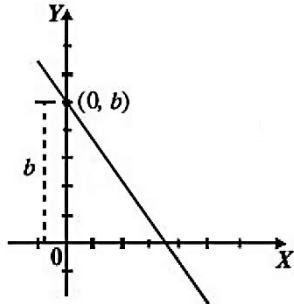


Funciones de Activación

En este punto, hemos entendido el funcionamiento general de una ANN. Es el momento de profundizar un poco.

$$X_1W_1 + X_2W_2 + b.W_0$$

MODELO LINEAL
 $Y = b + WX$



MODELO NO
LINEAL

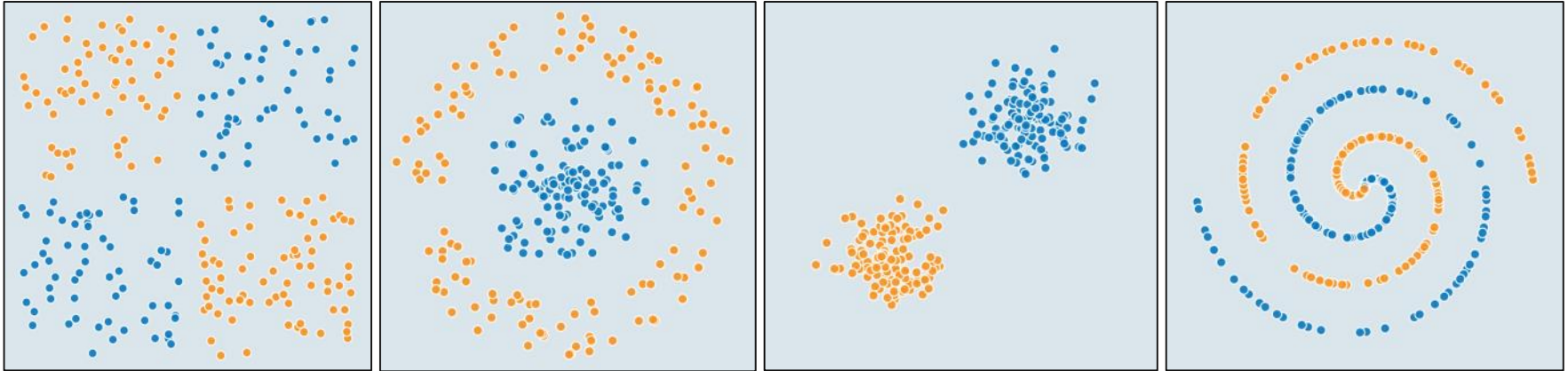
$$f(b + WX)$$

¿Por qué?

Funciones de Activación

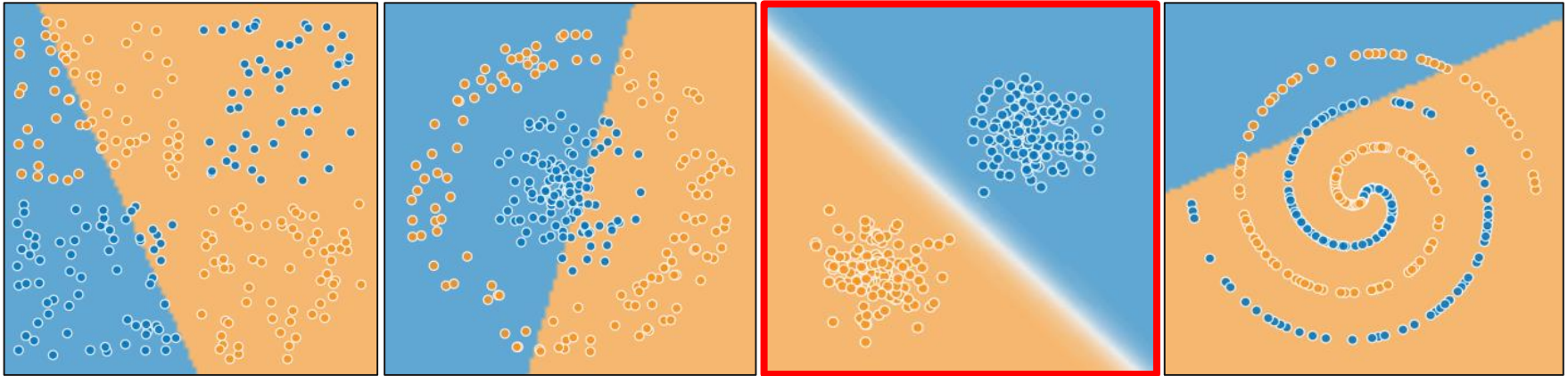
Sin la parte no lineal, la salida de una ANN simple o con muchas capas ocultas, será siempre otra función lineal.

Utilizar el [SIMULADOR](#) con una función de activación lineal e intentar separar los puntos.



Funciones de Activación

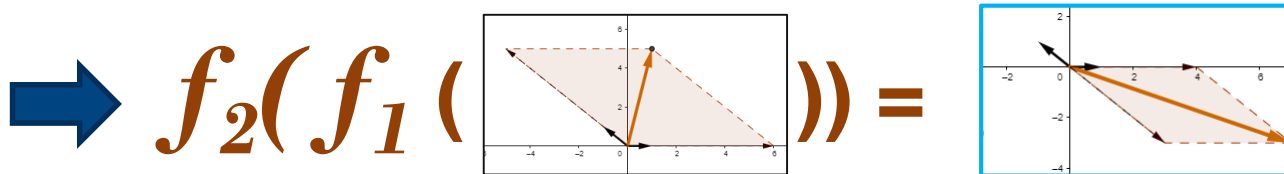
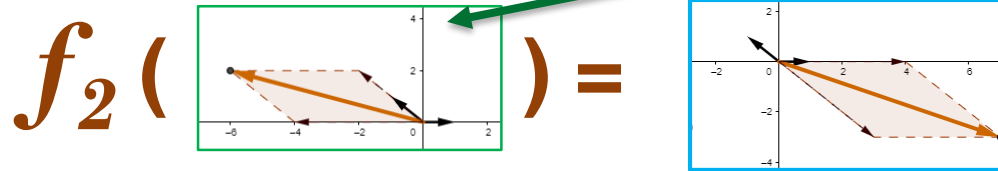
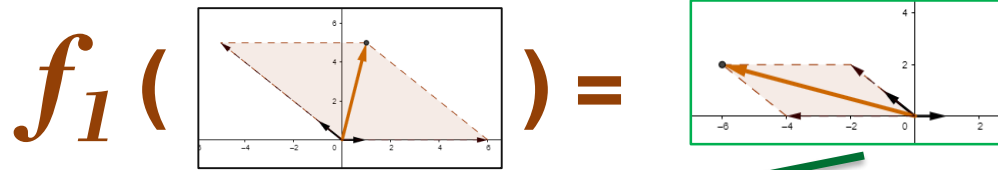
Sin la parte no lineal, la salida de una ANN simple o con muchas capas ocultas, será siempre otra función lineal. Los resultados son:



Solo el un caso pudo resolverse.
Pero **¿Por qué?**

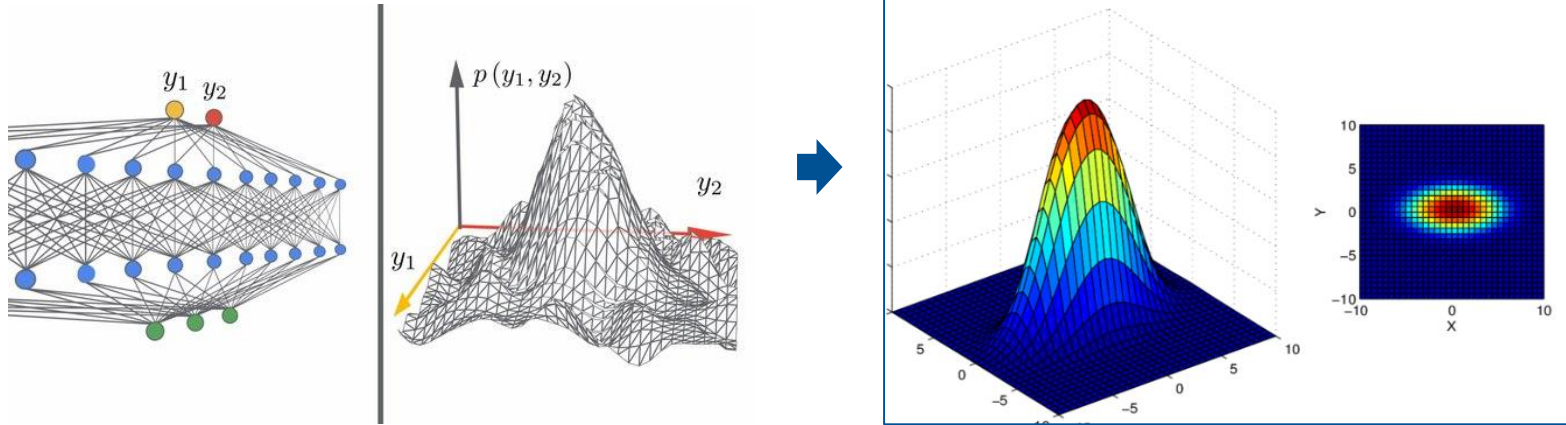
Funciones de Activación

Sin la parte no lineal, la salida de una ANN simple o con muchas capas ocultas, será siempre otra función lineal (Estamos haciendo **solo transformaciones lineales**).



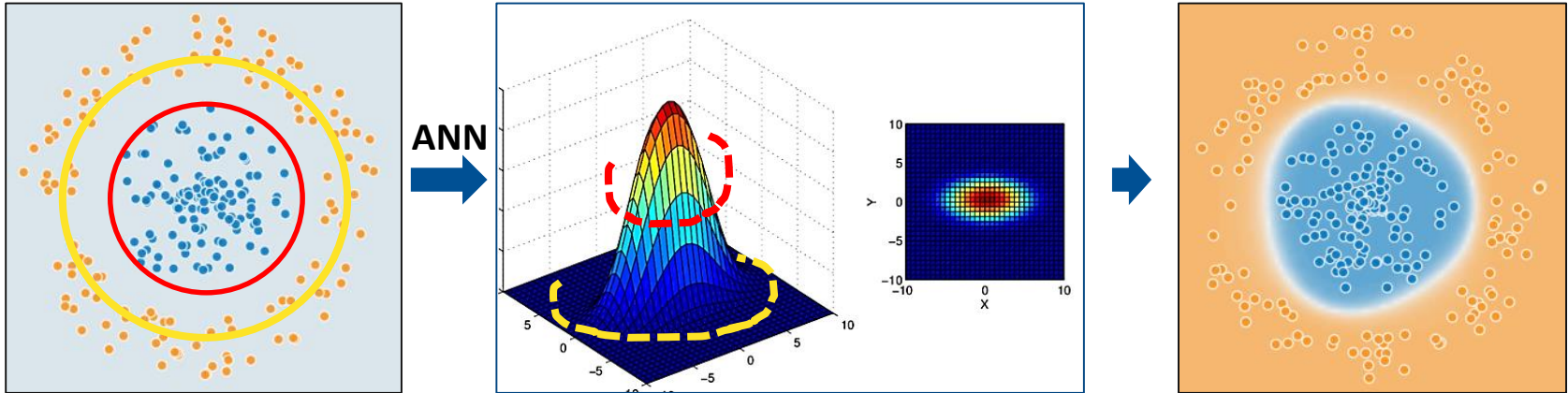
Funciones de Activación

El éxito de las redes neuronales es que ellas **deforman el espacio**, buscando una transformación donde los puntos sean separables. Pero esta transformación debe ser **no lineal**.



Funciones de Activación

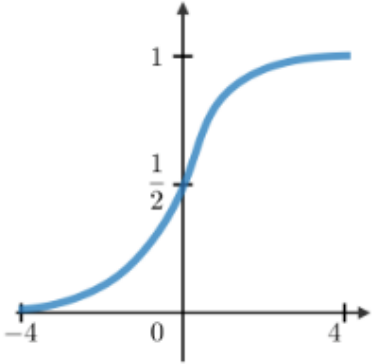
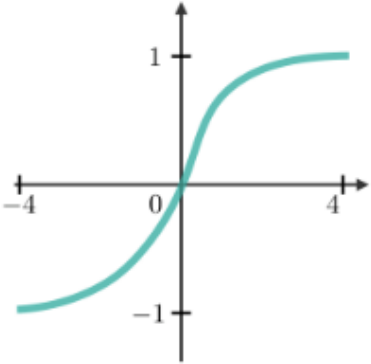
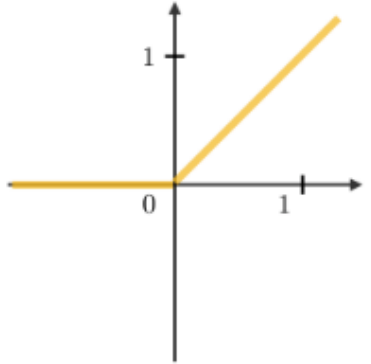
El éxito de las redes neuronales es que ellas **deforman el espacio**, buscando una transformación donde los puntos sean separables. Pero esta transformación debe ser **no lineal**.



**Espacio de Puntos
separables**

Funciones de Activación

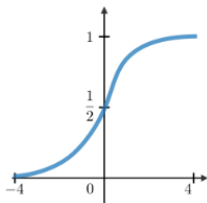
Las funciones no lineales son llamadas **FUNCIONES DE ACTIVACIÓN**. Se buscan funciones cuyas derivadas sean simples, para minimizar con ello el coste computacional. Las más conocidas son:

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

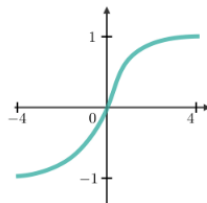
Funciones de Activación

Las funciones no lineales son llamadas **FUNCIONES DE ACTIVACIÓN**. Se buscan funciones cuyas derivadas sean simples, para minimizar con ello el coste computacional. Las más conocidas son:

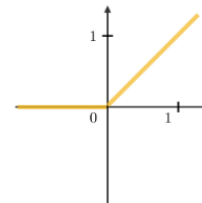
Sigmoid	Tanh	RELU
---------	------	------



- Satura y mata el **gradiente**
- Lenta Convergencia
- Acotada entre **[0, 1]**
- Buen rendimiento en la ultima capa. (usada como **clasificador binario**)



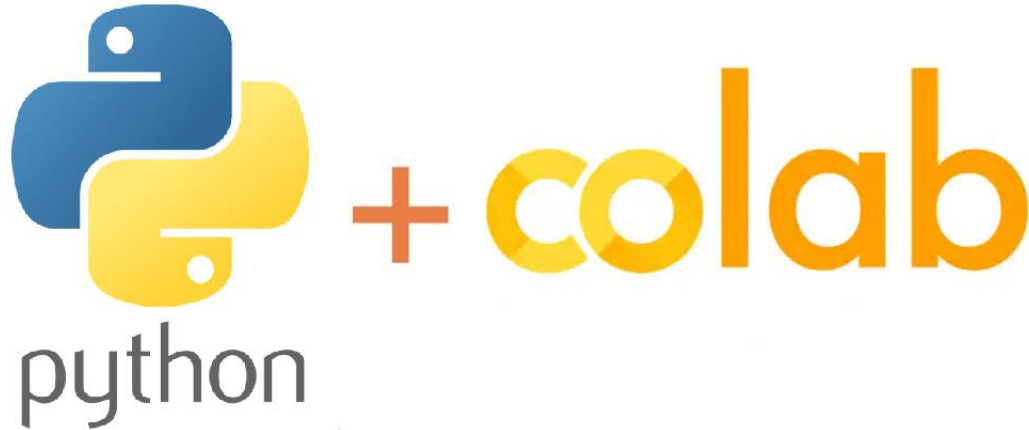
- Satura y mata el **gradiente**
- Lenta Convergencia
- Acotada entre **[-1, 1]**
- Se usa para decidir entre una opción y la contraria
- Buen rendimiento en las **redes recurrentes**



- Activación **Sparse**
- No está acotada
- Puede cancelar muchas neuronas
- Excelente desempeño en **redes convolucionales**.

Funciones de Activación

Programemos un poco en **Google colab**.



03

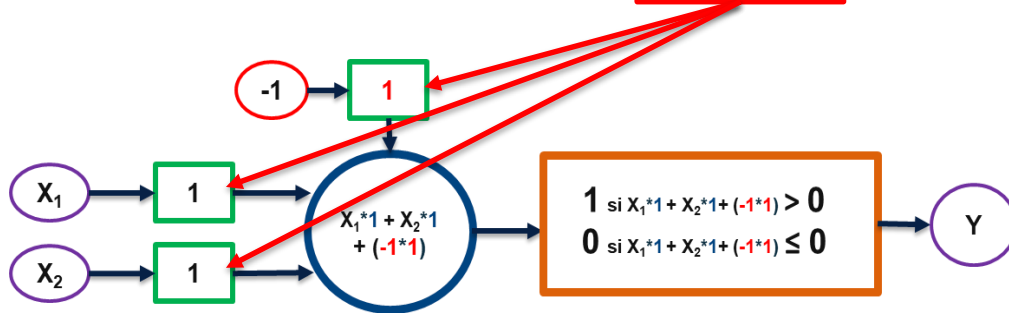
Descenso del Gradiente

Descenso del Gradiente

Ya entendimos como funciona una red neuronal en general, y la importancia de la función de activación. Ahora, surge una nueva interrogante:

En el ejemplo de la función **AND**,

¿Cómo llegamos a los valores finales para W_0 , W_1 y W_2 ?

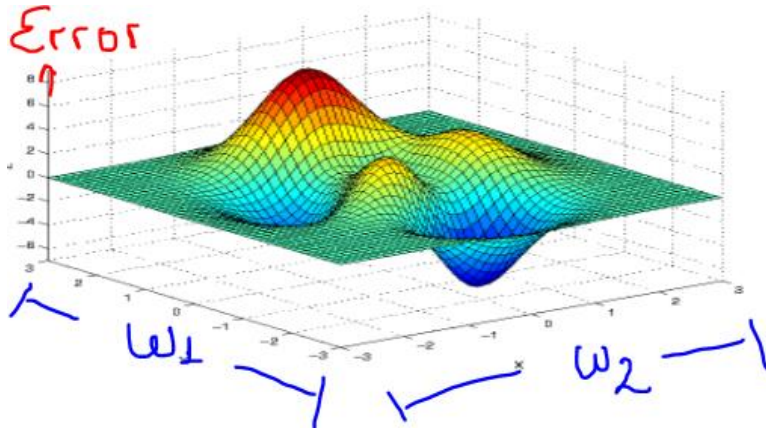


Los parámetros de la red, son los valores que debemos **aprender**.

Debemos ir ajustándolos mediante **entrenamiento** hasta obtener unos pesos que permitan solucionar el problema.

Descenso del Gradiente

Para comenzar, es necesario entender que el **error** está en función de los **pesos**.



$$\text{ERROR} = f(w_1, w_2)$$

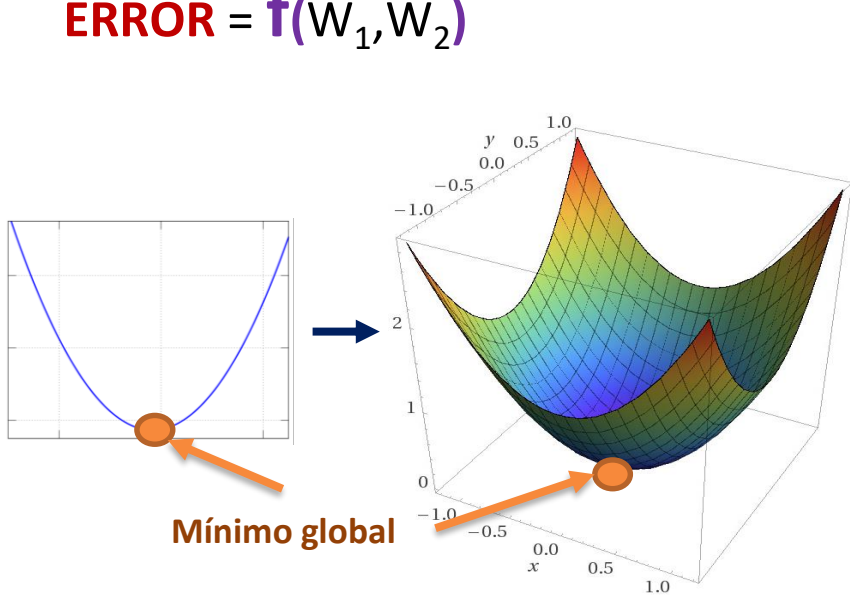
El objetivo del descenso del gradiente es encontrar el mínimo de una función objetivo.

Y nuestro objetivo es minimizar la función del error.

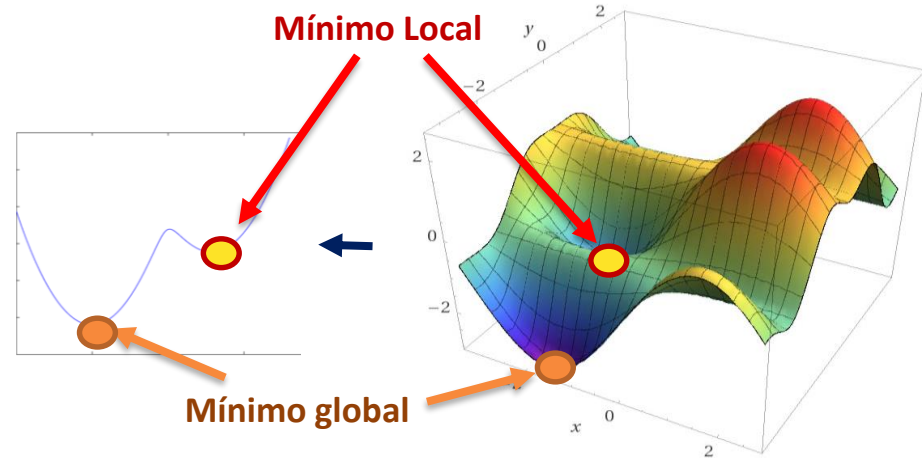
Descenso del Gradiente

El objetivo del descenso del gradiente es encontrar el mínimo de una función objetivo.

$$\text{ERROR} = f(W_1, W_2)$$



Convexo



No Convexo

Descenso del Gradiente

¿Cómo funciona este algoritmo?

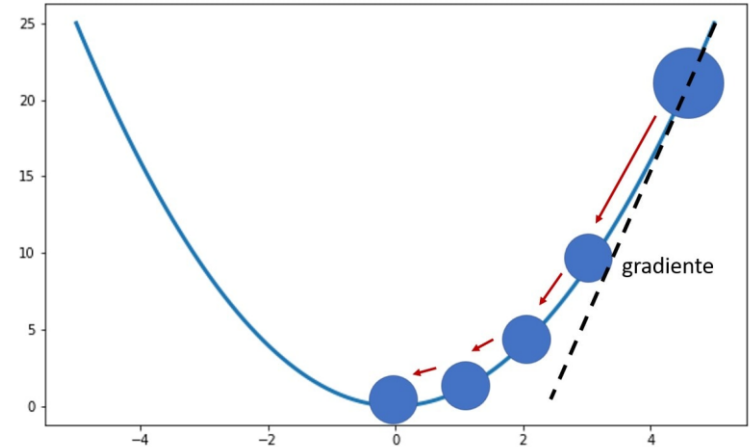
- 1) Definir un valor inicial para \mathbf{W}
- 2) Recibir los valores de entrada x^k
- 3) Definir \mathbf{lr} (tasa de aprendizaje) y \mathbf{n} (número de épocas)
- 4) Repetir por \mathbf{n} épocas o hasta tener un **error** aceptable

Calcular el error = $f(x^k)$

Escoger una dirección de descenso $d^k = -\nabla f(x^k)$

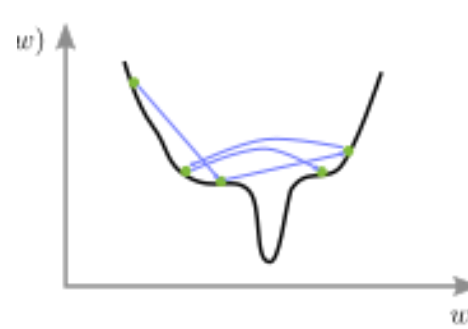
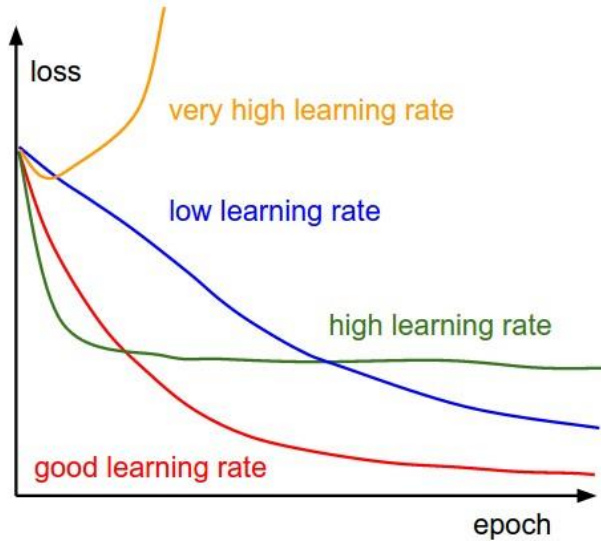
Actualizar $\mathbf{W} = \mathbf{W} + (d^k * \mathbf{lr})$

- 5) Retornar el modelo con \mathbf{W} ajustados

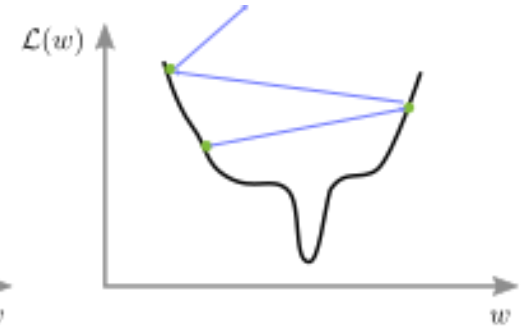


Descenso del Gradiente

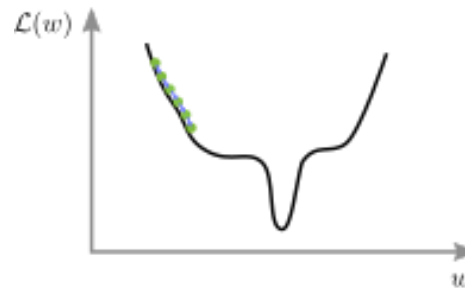
Tomar cuidado con el valor de la tasa de aprendizaje



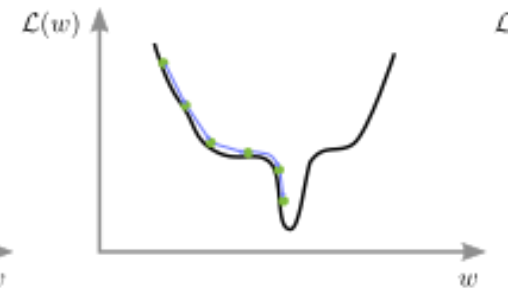
High learning rate



Learning rate much too high



Learning rate too low



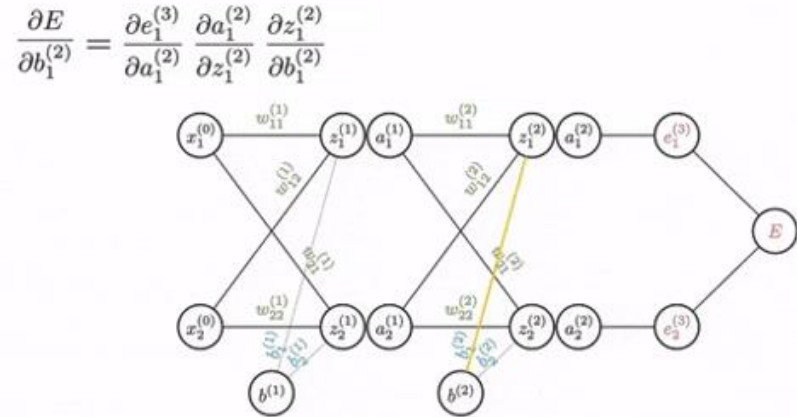
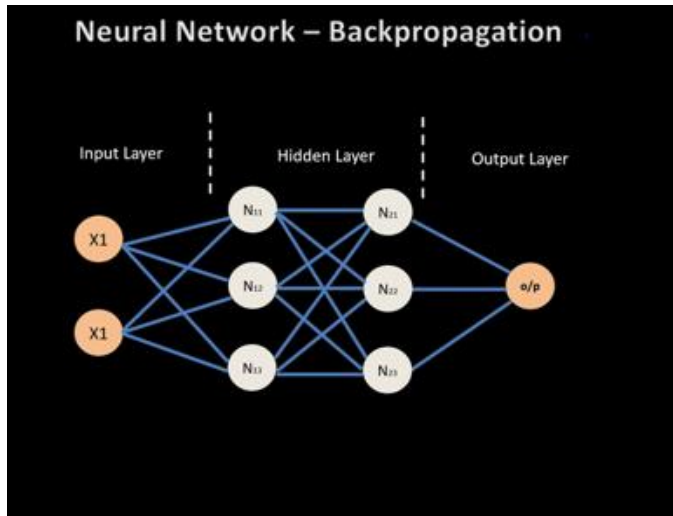
Good learning rate

Backpropagation

¿Qué ocurre si tenemos muchas neuronas?

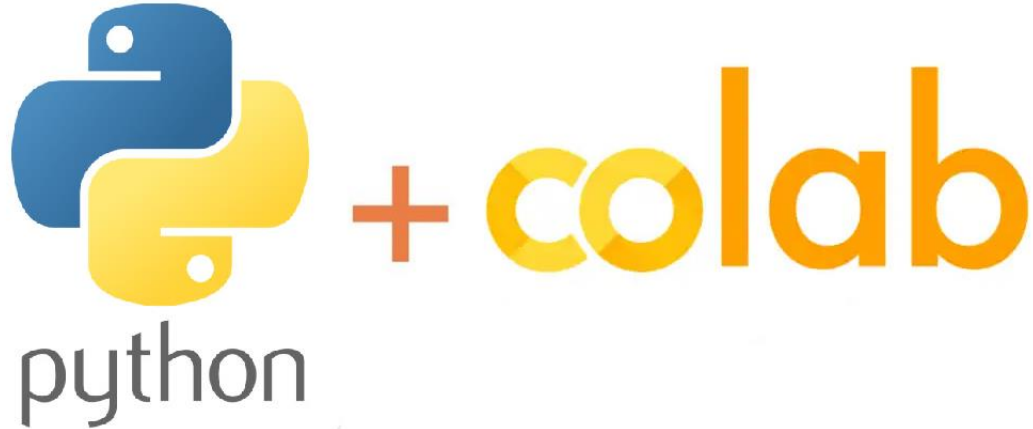
¿Cómo identificamos que neuronas aportan el mayor error al modelo?

El algoritmo backpropagation soluciona el problema.



Overfitting

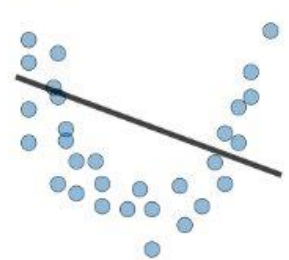


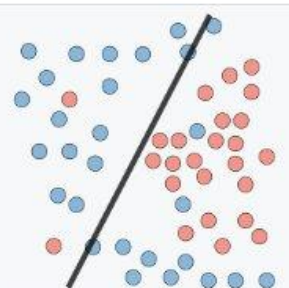
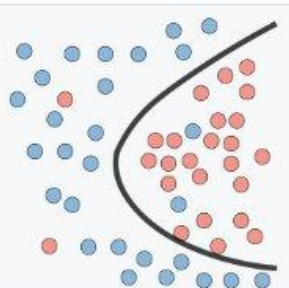
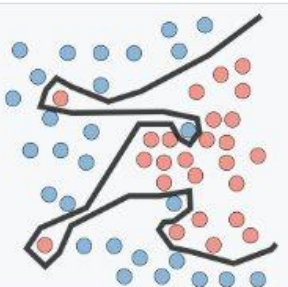
Programemos un poco en **Google colab**.



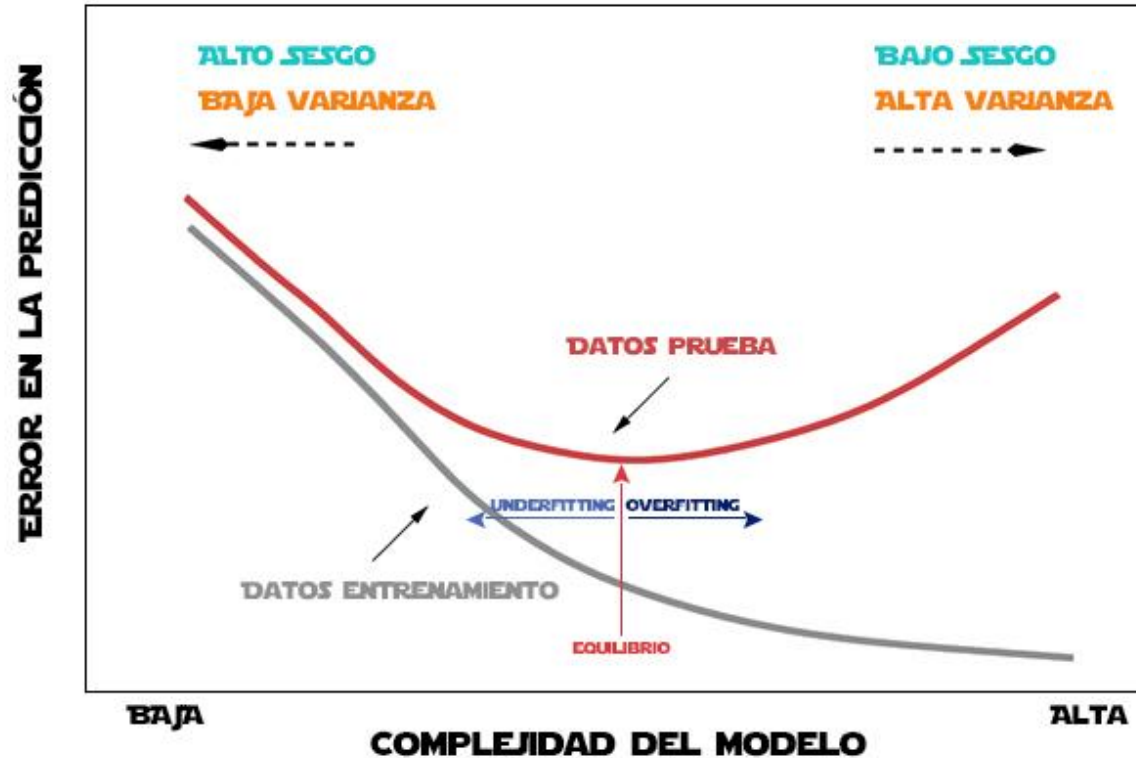
03

El problema del Overfitting

Overfitting

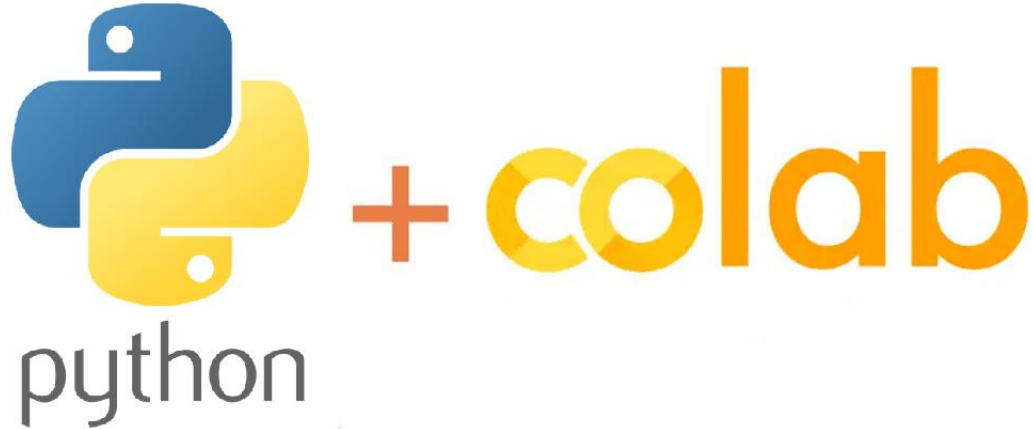
	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Regression illustration			
Classification illustration			

Overfitting



Overfitting

Programemos un poco en **Google colab**.



A dark blue background featuring a faint image of a robotic hand shaking a human hand. Two white horizontal lines with a right-angle bend and a dot at the end are positioned above the main text.

iGracias!



www.riskanalyticsperu.com



informes@riskanalyticsperu.com



902-679-682