

HENRY



Redes neuronales

Data Science





Agenda



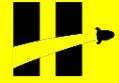
- Redes Neuronales
- Deep Learning
- Perceptrón y Perceptrón Multicapa
- Función de Costo
- Descenso por Gradiente
- Fordward Propagation y Back Propagation
- GPT



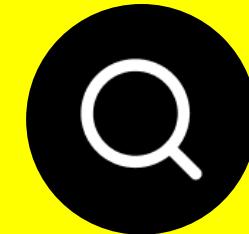
OBJETIVOS DE LA CLASE

Al finalizar esta lectura estarás en la capacidad de...

- Comprender el funcionamiento básico de las Redes Neuronales
- Entender los conceptos de Deep Learning, Forward Propagation, Back Propagation, Función de Costo y Funciones de Activación



Al **finalizar** cada uno de los temas,
tendremos un **espacio de consultas**.



Hay un **mentor** asignado para
responder el **Q&A**.

¡Pregunta, pregunta, pregunta! :D



Redes Neuronales





¿Qué es?

El modelo de neuronas artificiales inspirado en el comportamiento biológico de las neuronas y en cómo se organizan formando la estructura del cerebro, publicado por McCulloch y Pitts en 1943, se considera el primer trabajo en el campo de la Inteligencia Artificial.

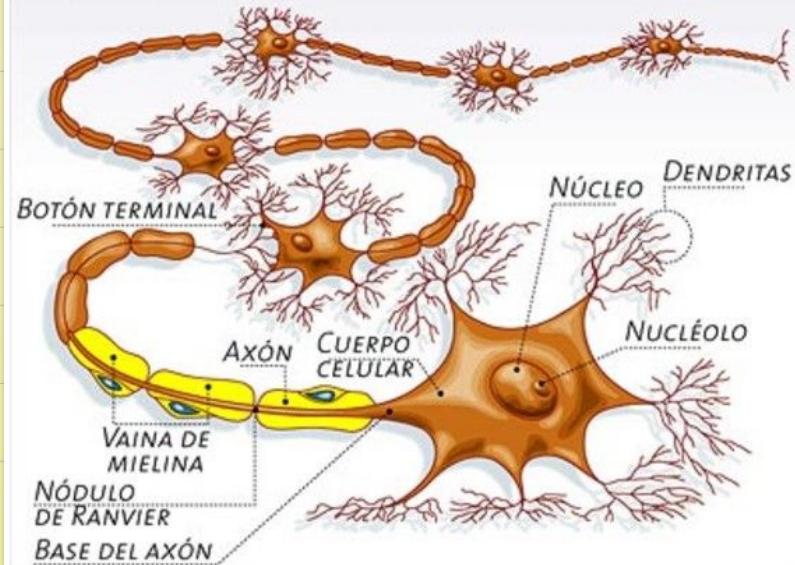


¿Qué es?

Partes:

- El **cuerpo central**, llamado soma, que contiene el núcleo celular.
- Una prolongación del soma, el **axón**.
- Una ramificación terminal, **dendritas**.
- Una zona de conexión entre una neurona y otra, conocida como **sinapsis**.

La neurona es la unidad funcional y estructural del sistema nervioso que produce y transmite el impulso nervioso. Se encuentra formada por tres partes: el **cuerpo neuronal** o **soma**, una prolongación larga y poco ramificada llamada **axón**, y otras prolongaciones muy ramificadas alrededor del soma llamadas **dendritas**.

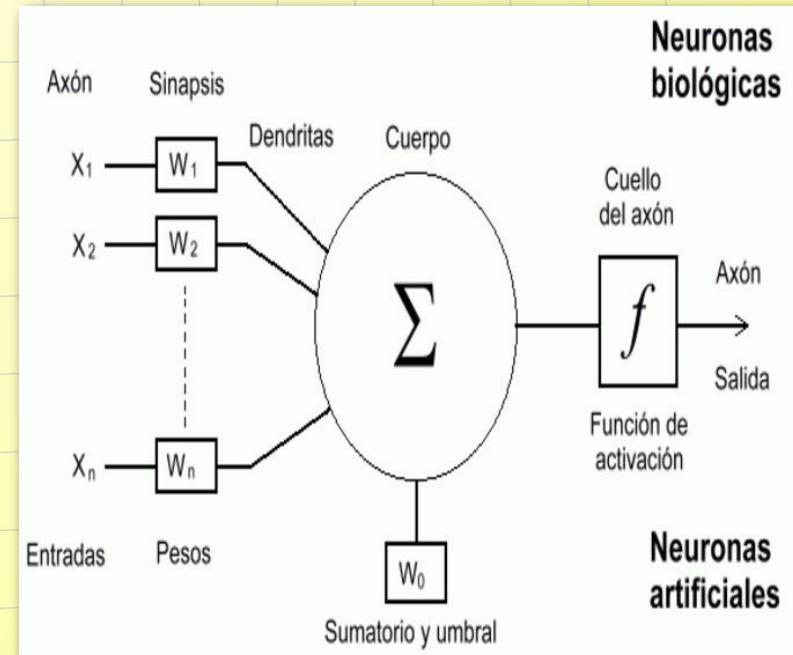


Modelo neuronal

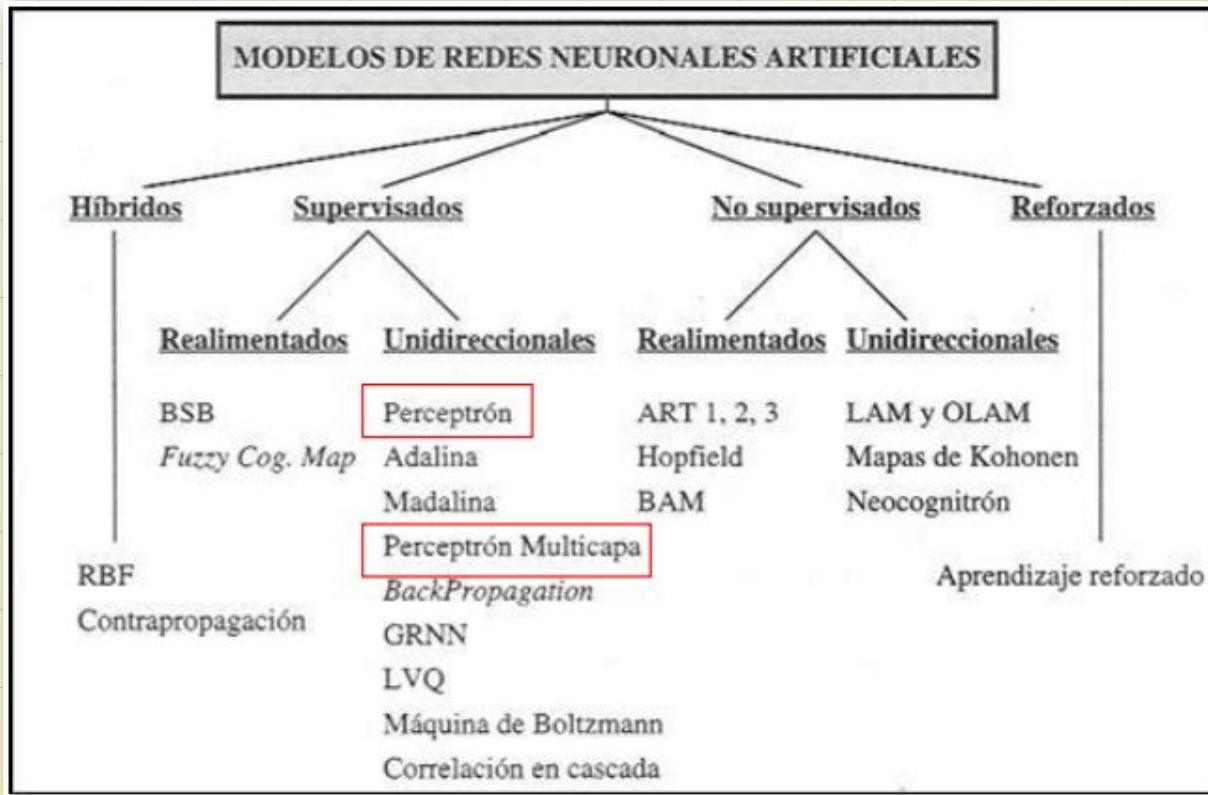
Modelo neuronal con n entradas, que consta de:

- Un conjunto de entradas x_1, \dots, x_n
- Los pesos sinápticos w_1, \dots, w_n , correspondientes a cada entrada
- Una función de agregación, Σ
- Una función de activación, f
- Una salida:

$$Y = f \left(\sum_{i=0}^n w_i x_i \right)$$



Redes neuronales





Deep Learning



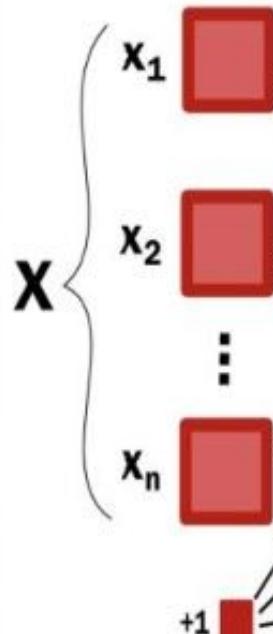


¿Qué es?

- Las neuronas se agrupan en **capas**.
- La primera capa (de entrada) tiene tantas neuronas como variables de entrada tiene el planteo del problema, mientras que la última capa tantas neuronas como salidas haya.
- Las capas que están entre la capa de entrada y la de salida, se llaman **capas ocultas**.
- Cada neurona tiene sus propios pesos/parámetros (suelen ser desde miles a millones de parámetros para toda la red).
- Esa obtención de conocimiento jerarquizado, es lo que da nombre al **Aprendizaje Profundo** o **Deep Learning**, y requiere de encontrar esos pesos **W** de manera eficiente, bajo la condición de realizar correctamente una tarea objetivo.



ENTRADAS:



Capa
oculta:

$$W^{(1)}$$

Capa de:
salida:

$$W^{(2)}$$

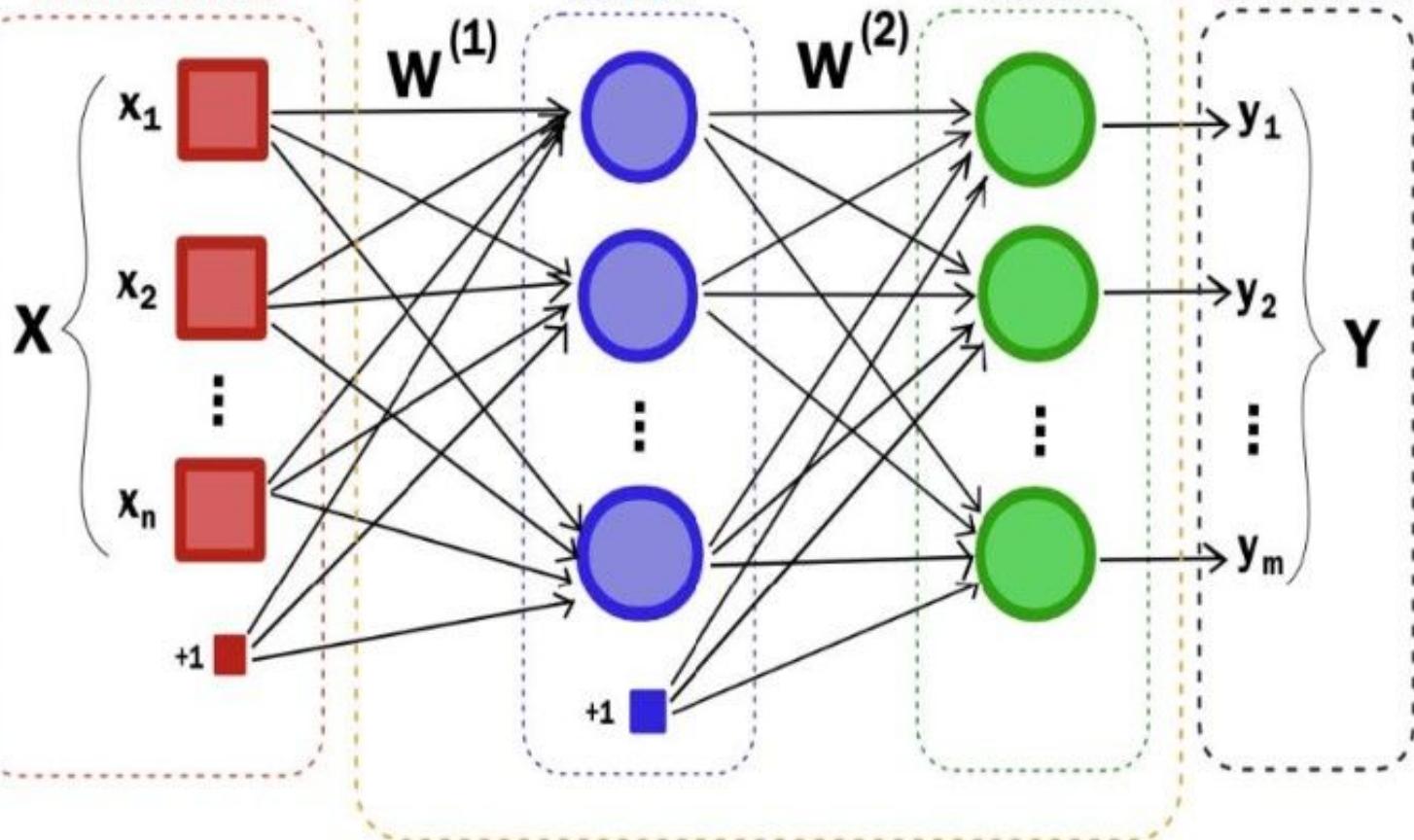
SALIDAS:

$$y_1$$

$$y_2$$

$$y_m$$

$$Y$$





Aprendizaje de Redes Neuronales

Es un problema de Regresión Lineal, por ese motivo buscamos entonces $Y = mx + b$ que mejor ajuste a los datos.

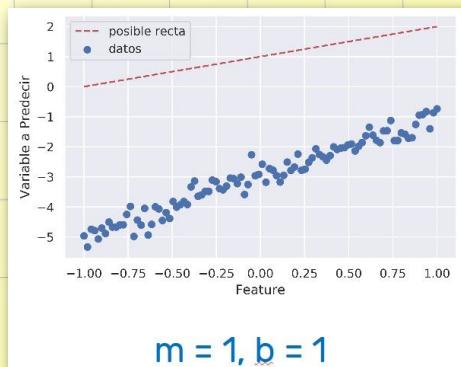
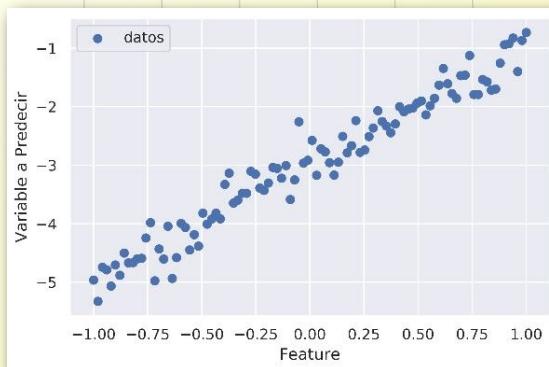
- m : pendiente
- b : ordenada al origen



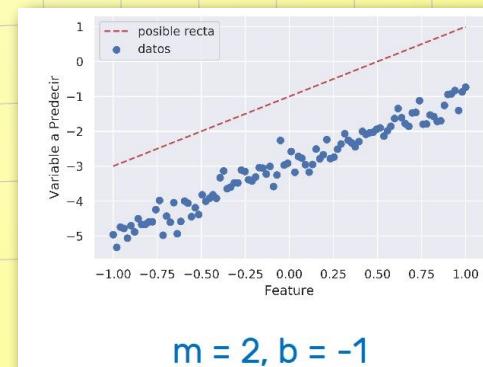
Aprendizaje de Redes Neuronales

Es necesario definir cuál es el mejor ajuste a los datos.

Se puede probar con distintos valores de m y b , y quedarse con el que mejor ajusta.



$$m = 1, b = 1$$



$$m = 2, b = -1$$

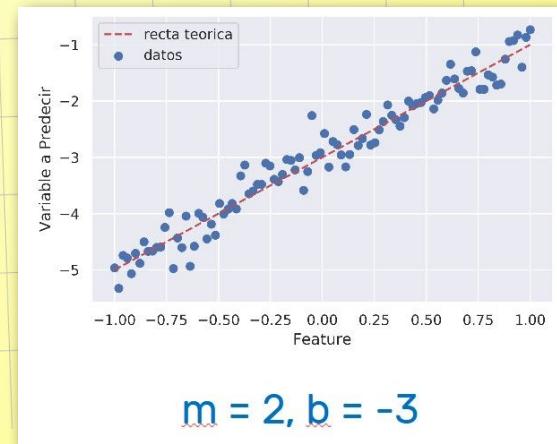


Aprendizaje de Redes Neuronales

Con un modelo paramétrico ($y = mx + b$), se puede hacer una búsqueda por fuerza bruta para encontrar el m y b (parámetros), y guiar la búsqueda con una función de costo.

Muy similar a GridSearch, pero en este caso es para buscar los parámetros del modelo, no sus hiperparámetros.

Ejemplo de hiperparámetro: **grado del polinomio**. Esta clase de problemas se conocen como **problemas de optimización**:

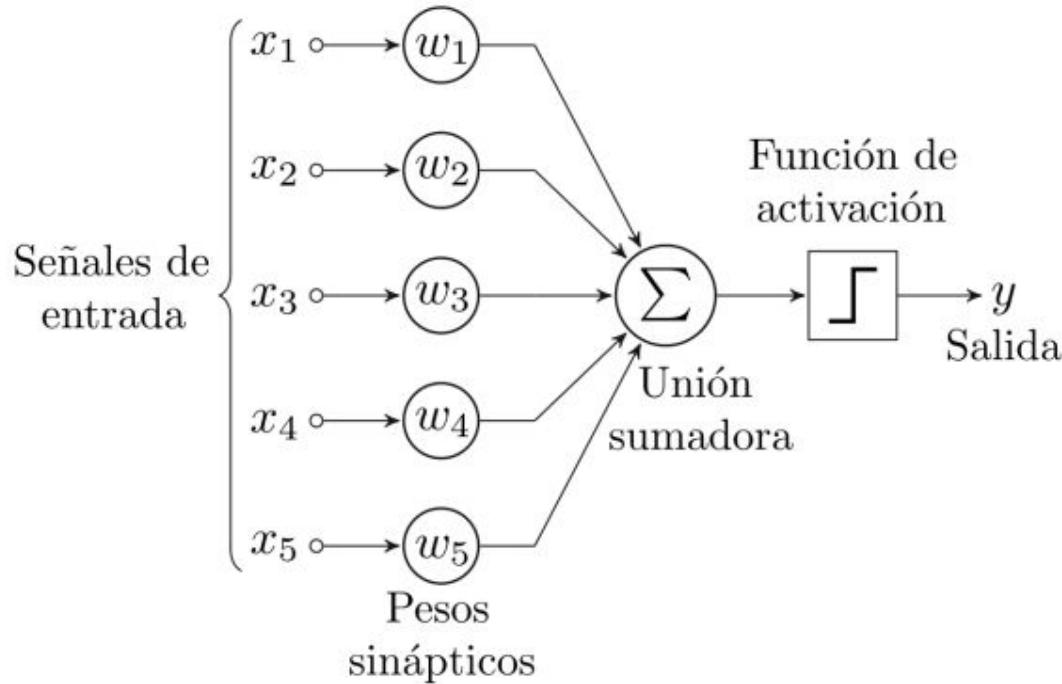




Perceptrón simple



Perceptrón simple



Entrada: Ingreso de señales de entrada (x_n) para ser procesada

- w_n : peso de la conexión de entrada y el nodo

Procesamiento:

- Suma de entradas:

$$Z = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_5 * w_5$$

$$O$$

$$Z = \sum_{j=1}^5 x_j w_j$$

Salida:

- Función de activación: $Y = \varphi(Z)$



Función de activación

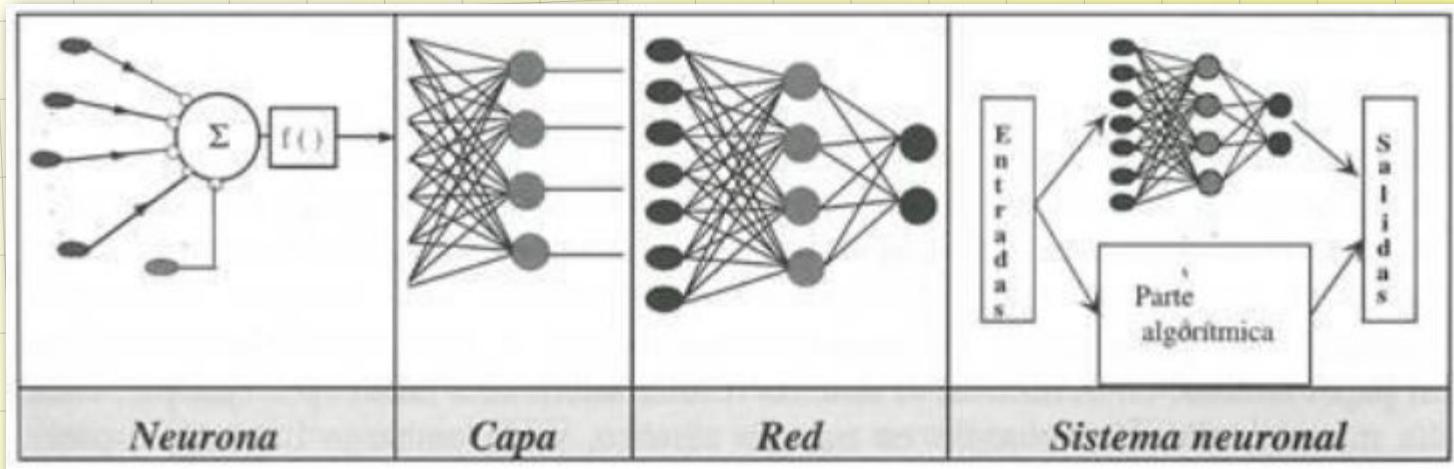
Las funciones de activación (φ) pueden ser lineales o no lineales.

Aquí algunos ejemplos:

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

Perceptrón simple

En la práctica no se suelen utilizar mucho porque matemáticamente son muy limitados, sin embargo, funcionan mejor cuando se agrupan con más unidades.



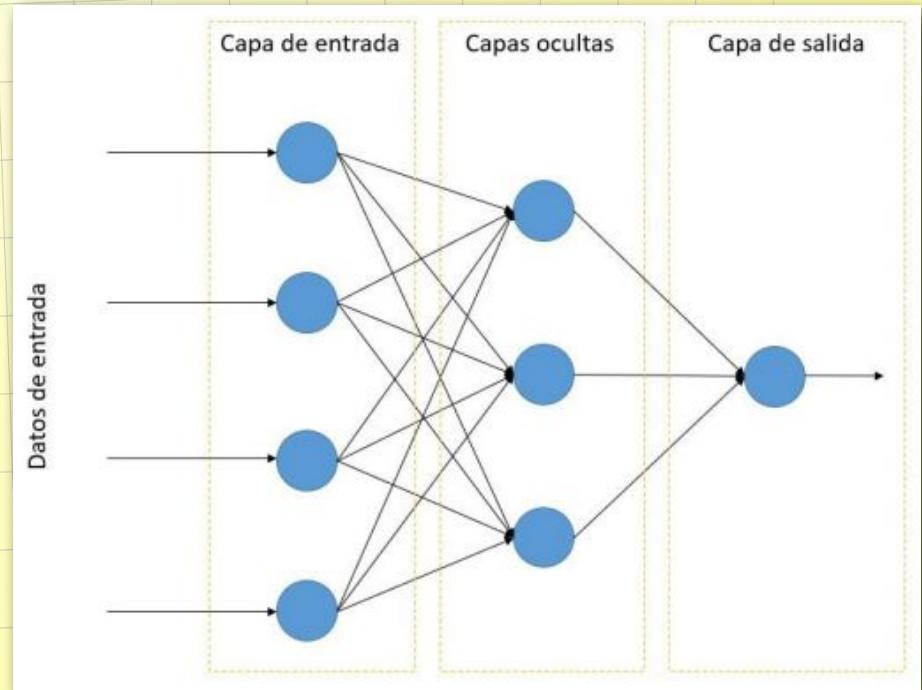


Perceptrón multicapa



Perceptrón Multicapa (MLP)

- **Capa de entrada:** Ingreso y transporte de información (x_n) a los nodos.
- **Capa oculta:** Procesamiento de información. Cada nodo una preponderación (z) y una función de activación $Y=\varphi(z)$. La salida de una neurona corresponderá a la entrada de otra.
- **Capa de salida:** Resultado final de la red neuronal.





Perceptrón Multicapa (MLP)

Dependiendo del número de Capas Ocultas la red recibe un nombre diferente:

- Si tienen 1 sola capa oculta serán nombradas Redes Neuronales Poco Profundas - **Shallow Neural Networks**.
- Si poseen más de 1 capa oculta se les llama Redes Neuronales Profundas - **Deep Neural Networks**.



Entrenamiento



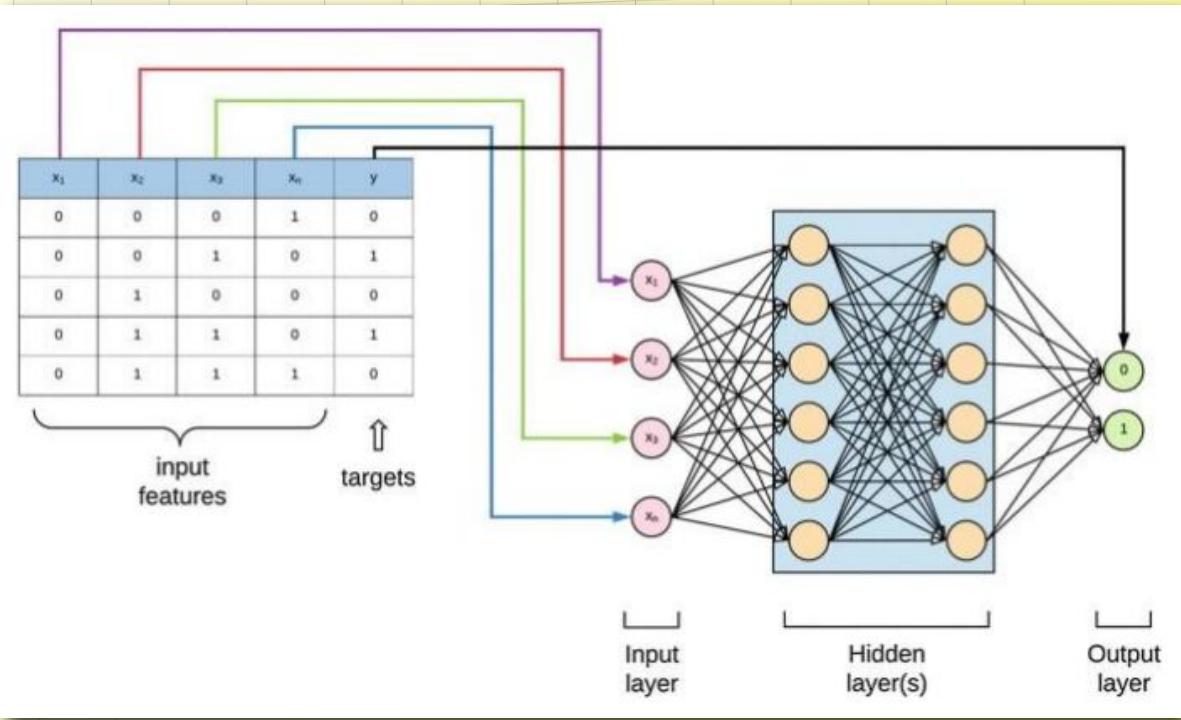


Entrenamiento

Se basa siempre en intentar buscar los pesos de las neuronas que ofrecan mejores resultados.

Condición de parada:

- Cuota de error suficientemente pequeña.
- Número máximo de iteraciones.
- Error cero.
- Punto de saturación.



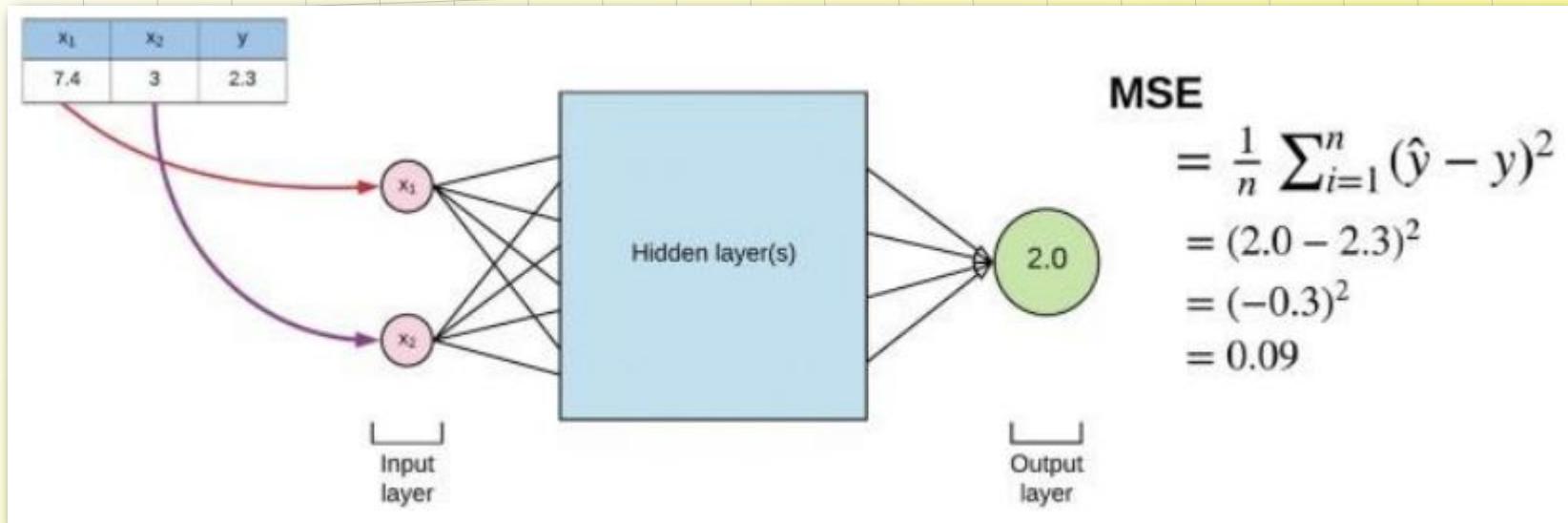


Funciones de costo



¿Cuáles?

Dos funciones de costo comúnmente utilizadas son la función de costo de error cuadrático para problemas de regresión y la función de costo de entropía cruzada softmax para problemas de clasificación.



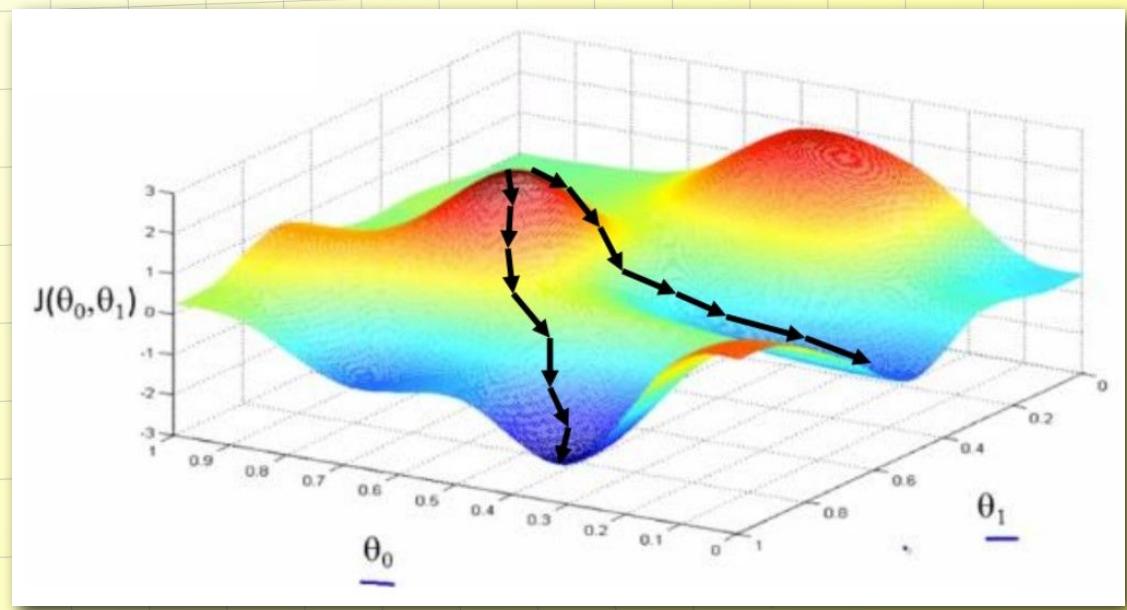


Descenso por Gradiente



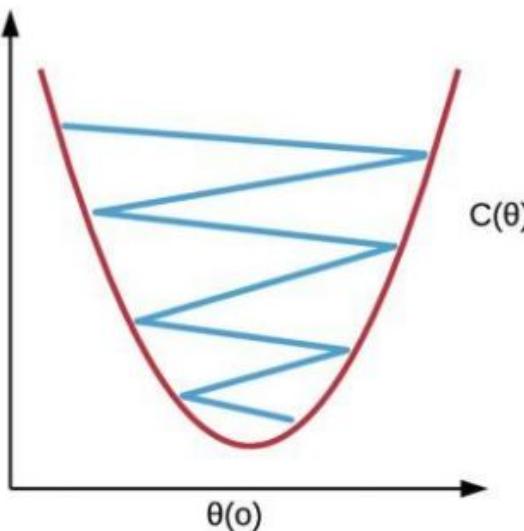
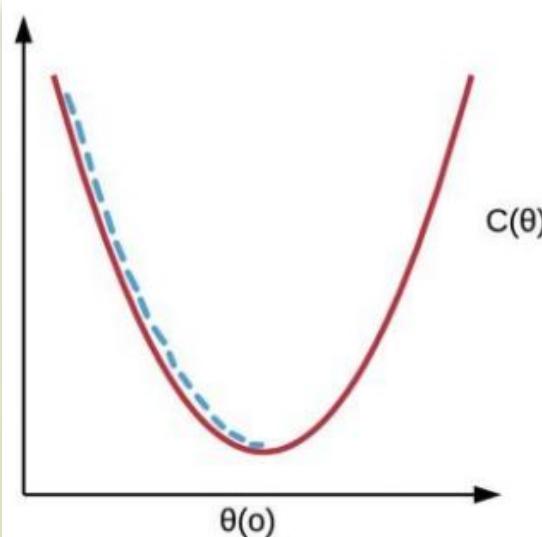
¿En qué consiste?

Descenso por
Gradiente consiste
en buscar un mínimo
global mediante
iteraciones en las
cuales se va
descendiendo en la
función de costo
 $J(\theta_0, \theta_1)$:



¿En qué consiste?

El método del gradiente descendente se basa en buscar la dirección en la que una pequeña variación del vector de pesos hace que el error decrezca más rápidamente.



$$W_i \leftarrow W_i + \Delta W_i$$

$$\Delta W_i = \propto \delta x$$

\propto = Tasa de aprendizaje

δ = Cambio del error

x = Entrada a la neurona



Forward Propagation

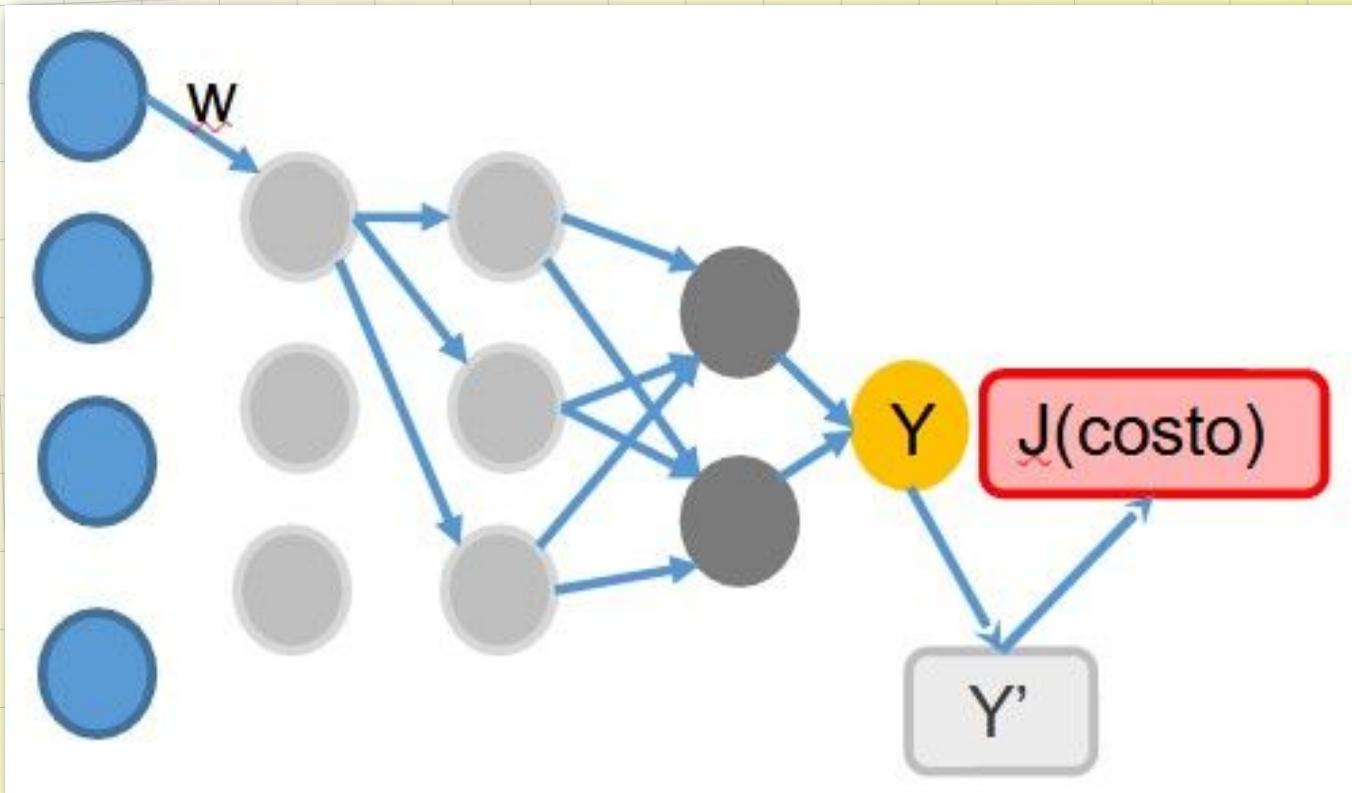




Forward propagation

- Cada valor w , afecta a la siguiente capa, y por ende, a todo el resto de la red neuronal, por tal motivo tiene parte de la responsabilidad en la función de Costo final.
- Esa función de Costo o de pérdida, determina cuán lejos está el resultado de la red contra el resultado esperado.
- El Descenso por gradiente calcula la derivada/gradiente del costo y con eso actualiza los parámetros. Este proceso lo va a hacer muchas veces hasta llegar al mínimo.
- En cada una de esas iteraciones, tiene que calcular el costo. El costo depende de las instancias de entrenamiento y de los parámetros que tengamos hasta ese momento.
- Calcular el costo con las instancias de entrenamiento es lo que se conoce como Forward propagation.

Forward propagation





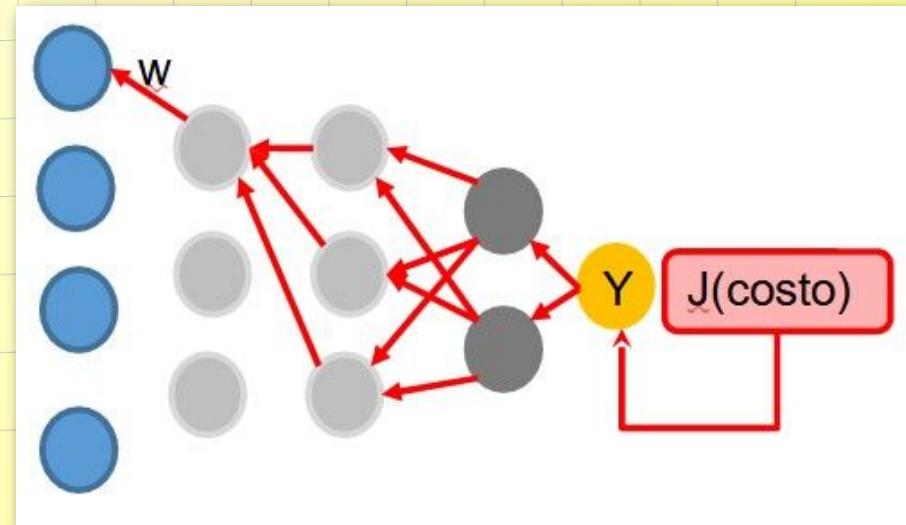
Back Propagation



Back propagation

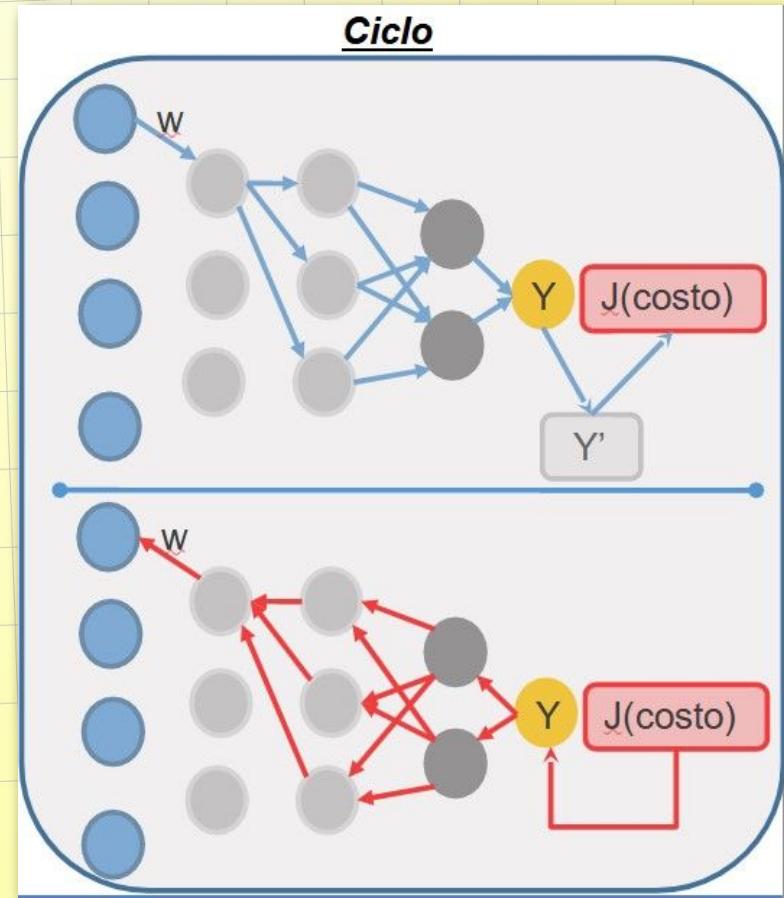
Con el costo calculado, se actualizan los valores de los parámetros ponderadores de las entradas, los w , que se inicializan con valores aleatorios.

- 1) Se deriva el costo con respecto a la función de activación ó predicción.
- 2) Se deriva la predicción con respecto a la función de agregación.
- 3) Se deriva la función de agregación, con respecto a los parámetros w .
- 4) Se actualizan los parámetros.



ciclo

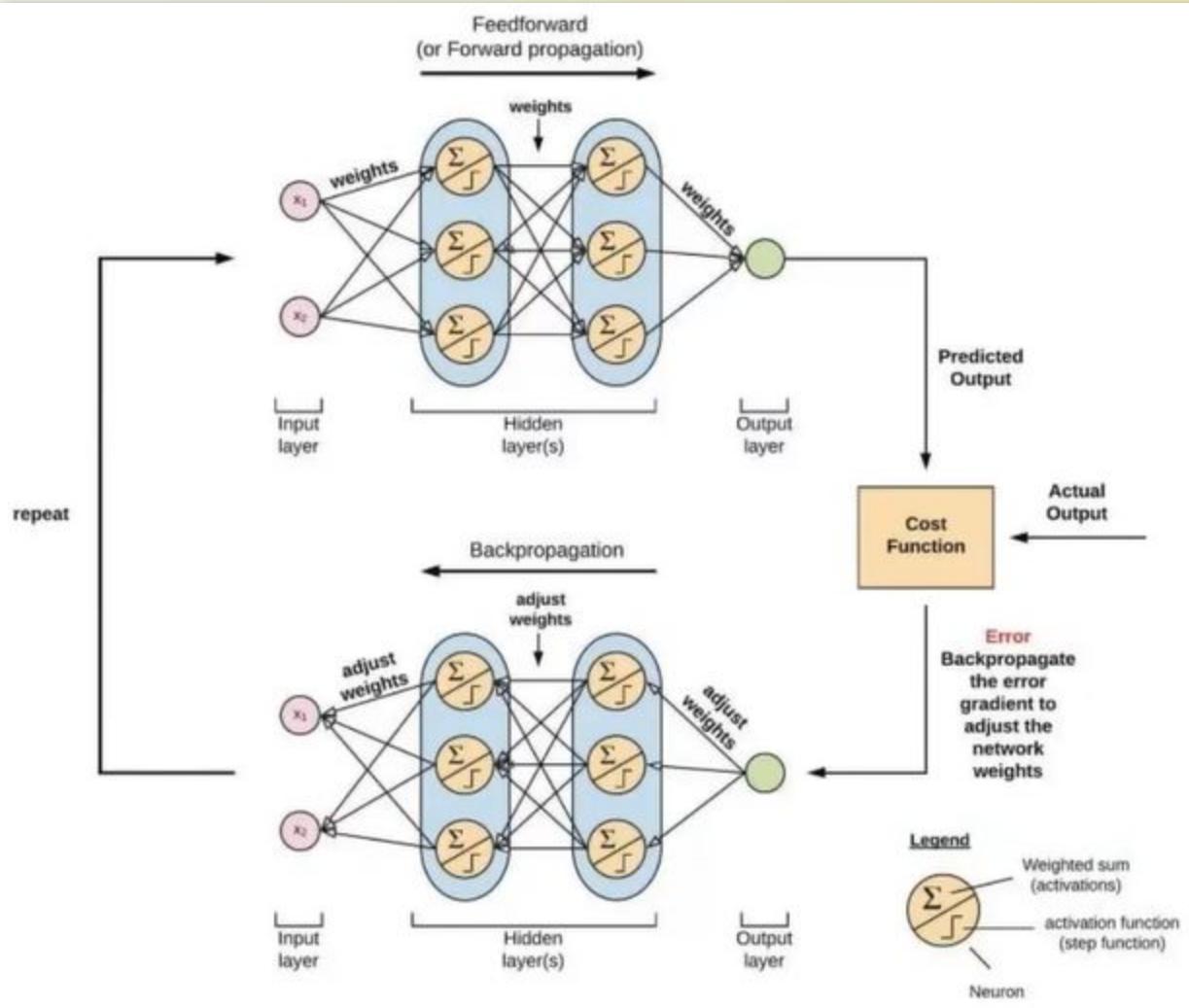
- con los datos de entrenamiento, la red ajusta sus parámetros, para adaptarse a la salida esperada.
- se llama **epoch** ó **ciclo**, al proceso completo desde un input de entrada, hasta la consecución de la función de coste, y posterior aplicación del algoritmo de Backpropagation de todos los ejemplos de entrenamiento.





Algoritmo “backpropagation”

- Inicializamos los pesos de manera aleatoria.
- Activamos cada neurona de nuestra red para hacer que la red produzca una salida.
- La salida producida en el paso anterior debe compararse con el conjunto de datos de entrenamiento real.
- calculamos el error cometido por la red.
- Ajustamos adecuadamente los pesos de cada neurona para reducir el error cometido por la red.



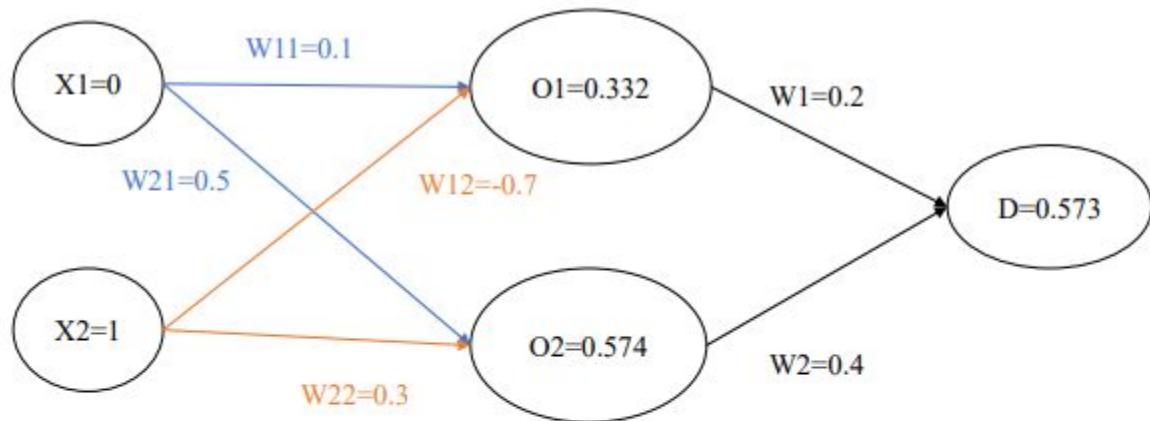


Ejemplos



Ejemplo 1

Entrada		Salida
X1	X2	D
0	1	1
1	0	1
1	1	0
0	0	0



$$Z = \sum_{j=1}^n x_j w_j$$

$$W_i \leftarrow W_i + \Delta W_i$$

$$y = \varphi(Z) = \frac{1}{1 + e^{-z}}$$

$$\Delta W_i = 0.25 \delta x$$



Ejemplo 1

Cálculos para obtener los resultados anteriores:

- $O_1 = \varphi(Z) = \varphi(x_1 w_{11} + x_2 w_{12}) = \varphi((0)(0.1) + (1)(-0.7)) = \varphi(-0.7) = 0.332$
- $O_2 = \varphi(Z) = \varphi(x_1 w_{21} + x_2 w_{22}) = \varphi(0.3) = 0.574$
- $D = \varphi(O_1 w_1 + O_2 w_2) = \varphi(0.2 \cdot 0.332 + 0.4 \cdot 0.574) = 0.573$
- $w_1 = w_1 + 0.25\delta O_1 = 0.2 + 0.25 \cdot 0.105 \cdot 0.332 = 0.209$
- $w_2 = w_2 + 0.25\delta O_2 = 0.4 + 0.25 \cdot 0.105 \cdot 0.574 = 0.415$
- $w_{11} = w_{11} + 0.25\delta x_1 = 0.1$
- $w_{21} = w_{21} + 0.25\delta x_1 = 0.5$
- $w_{12} = w_{12} + 0.25\delta x_2 = -0.7 + 0.25 \cdot 0.005 \cdot 1 = -0.699$
- $w_{22} = w_{22} + 0.25\delta x_2 = 0.3 + 0.25 \cdot 0.011 \cdot 1 = 0.303$



Ejemplo 1

En el ejemplo anterior, se retoma el algoritmo “backpropagation”, primero se inicializan los pesos de manera aleatoria, se activan las neuronas de la red (cálculos) y se compara el resultado con la salida.

Luego se calcula el error, se adecúan los pesos para reducir el error y se vuelve a realizar la iteración con los nuevos valores. En este caso se debe obtener un resultado más cercano al esperado, más cercano a 1.



Ejemplo 2

Si quisiéramos determinar si va a llover o no, y sabemos que esto depende esencialmente de la diferencia de temperatura entre el aire cercano a la superficie y el aire en la altura, que debe ser más frío; y por otra parte, también depende de la humedad, con lo que, si se dan ambas variables llueve.

Las dos variables son:

- x_1 : Diferencia de temperatura
- x_2 : Humedad

Ejemplo 2

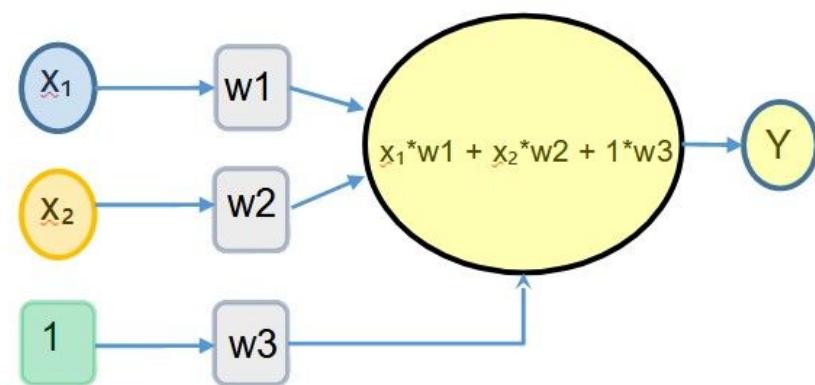
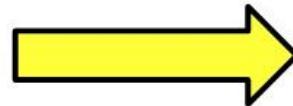
Teniendo en cuenta que cada una es binaria, es decir si hay la diferencia de temperatura necesaria o no, y si hay la suficiente humedad, entonces se puede armar la tabla:

X_1	X_2	
No	No	No
No	Sí	No
Sí	No	No
Sí	Sí	Sí

Ejemplo 2

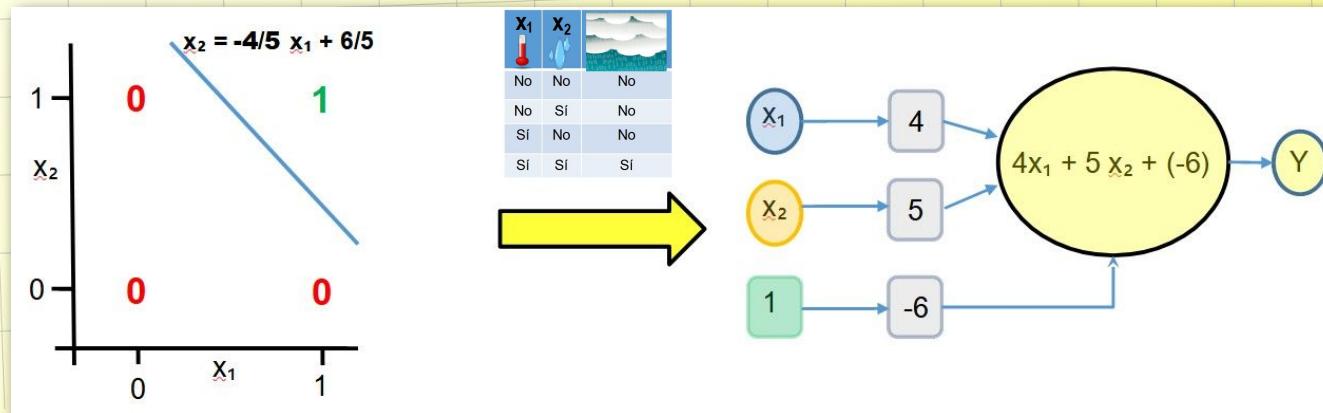
Convirtiendo los “Sí” y “No” en “1” y “0” respectivamente nos queda el siguiente esquema:

X ₁	X ₂	
No	No	No
No	Sí	No
Sí	No	No
Sí	Sí	Sí



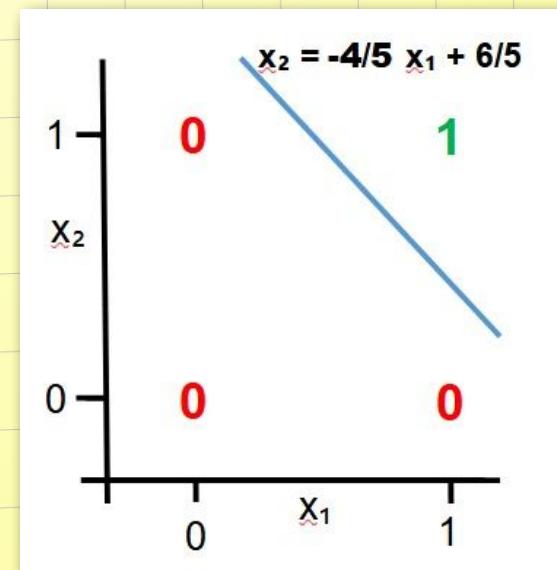
Ejemplo 2

- Se pueden ver las 2 entradas, más un valor de sesgo con su entrada que siempre vale 1.
- También están los pesos sinápticos, es decir, los ponderadores (w_1 , w_2 y w_3).
- El siguiente paso, es encontrar los valores w que separan correctamente las clases:



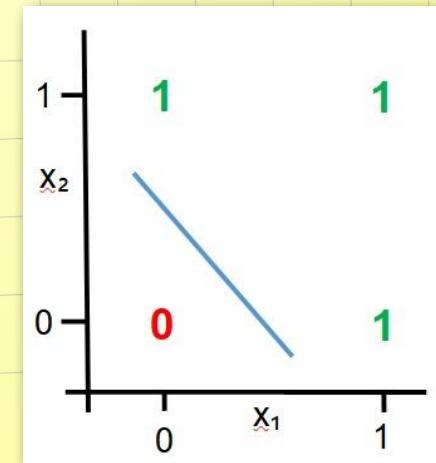
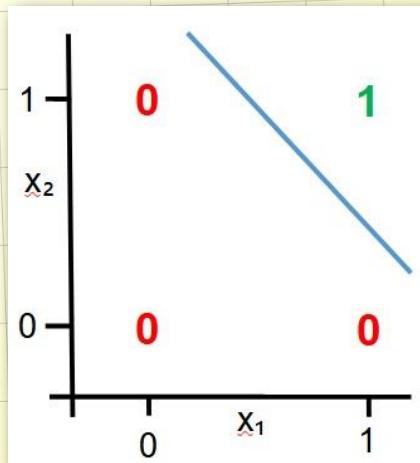
Ejemplo 2

La recta $x_2 = -4/5 x_1 + 6/5$ logra separar correctamente las clases. cuando x_1 y x_2 tienen el valor 1, la sumatoria es 3, es decir, mayor a 0. En cualquier otra combinación de valores de x_1 y x_2 el resultado es menor a 0. Una neurona artificial, en su núcleo, resuelve un problema de Regresión Lineal.



Ejemplo 2

Por analogía, podemos ver que esta gráfica, corresponde a la tabla de verdad de las compuertas lógicas AND y también con las compuertas lógicas OR.

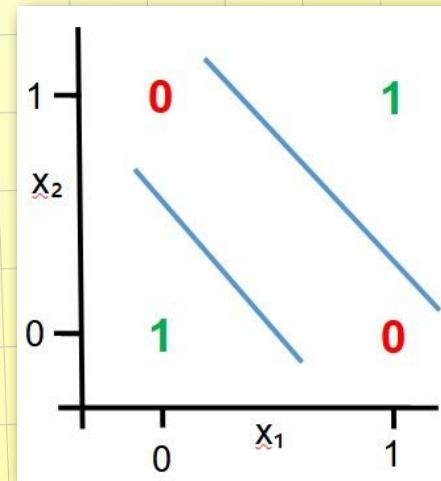




Ejemplo 2

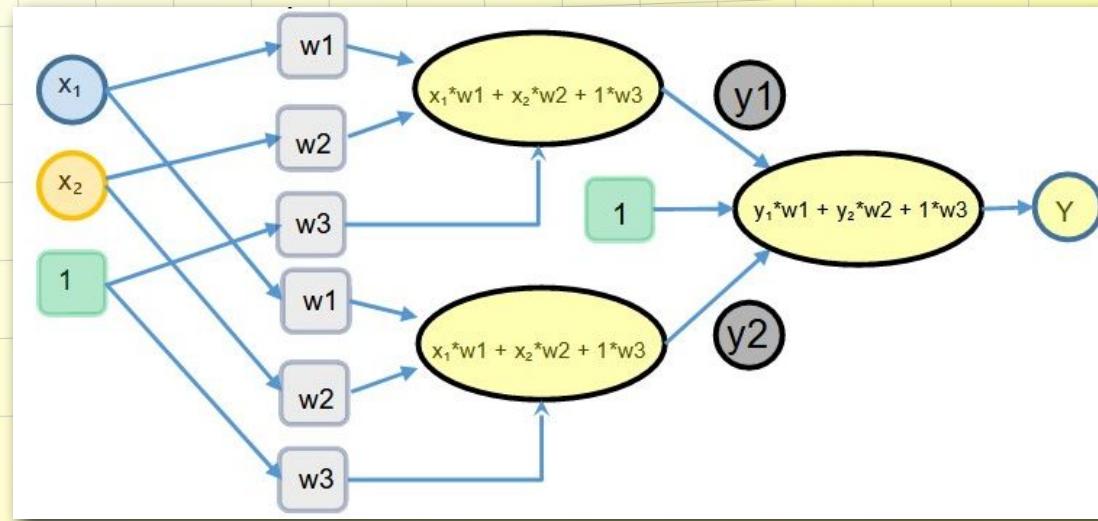
Pero, ¿qué pasa con la compuerta lógica XOR?

Mediante la gráfica, se puede ver que no es posible separar las clases con una sola recta.



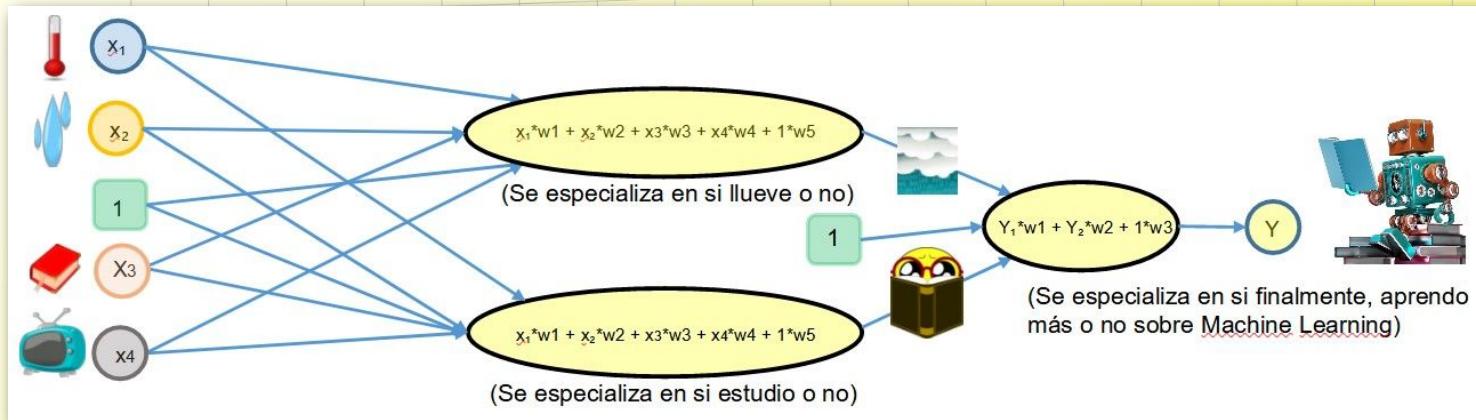
Ejemplo 2

Este problema existe desde que se formuló la neurona artificial, y dio origen a las redes neuronales, ya que, con dos neuronas, sí es posible resolverlo:



Ejemplo 2

- Agregar capas a la red, permite obtener conocimiento jerarquizado.
- Siguiendo el ejemplo, de si va a llover o no, podemos querer predecir información más compleja, como por ejemplo si podremos aprender más sobre Machine Learning, agregando dos entradas más:
 - Voy a leer el material sobre el tema que tengo disponible.
 - Me voy a quedar mirando tele.

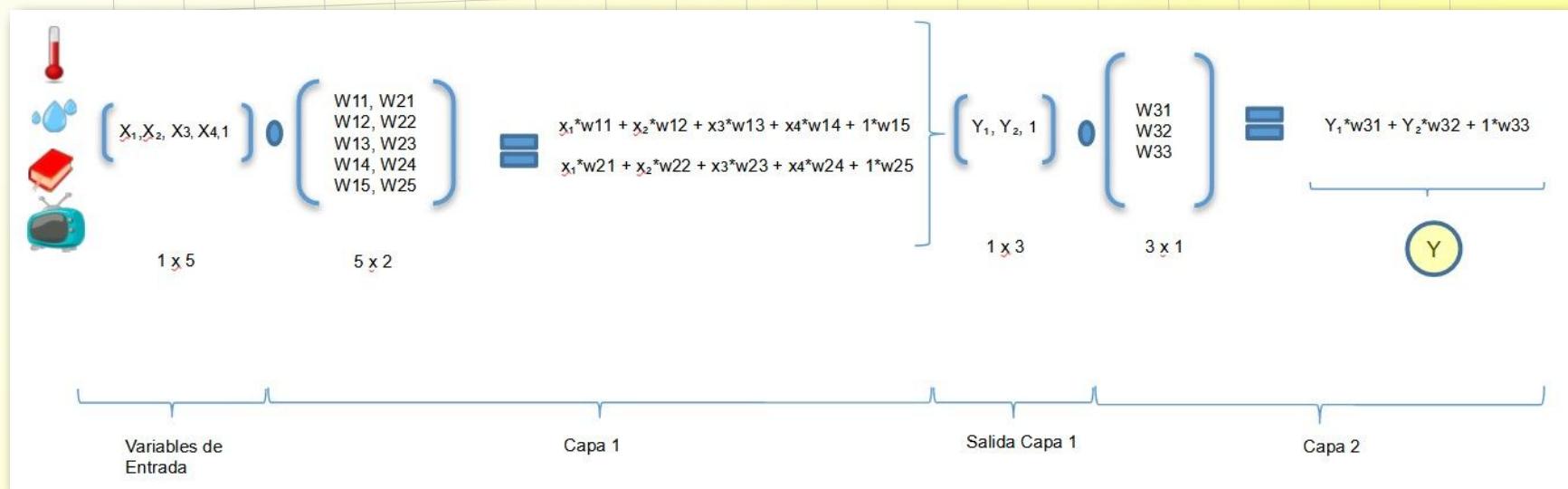


Ahora, es una red de 3 neuronas, con 13 w.



Ejemplo 2

A través del álgebra matricial, este proceso se puede representar como el producto escalar:





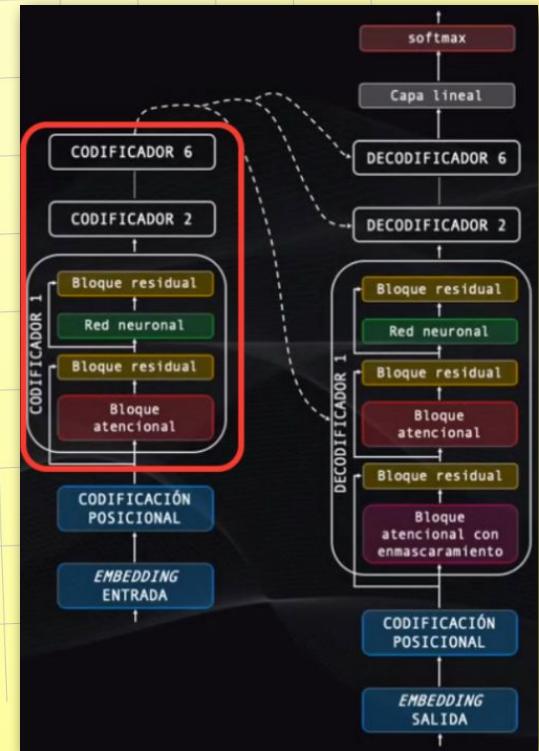
GPT



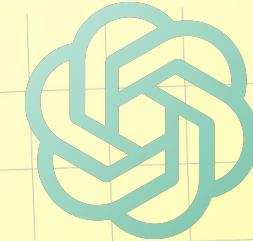


Redes transformers

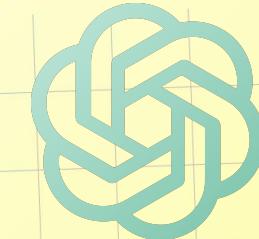
- Desarrolladas como una alternativa al problema de la traducción de texto de un idioma a otro.
- Analizan secuencias de palabras muy extensas usando un mecanismo que se llama atención.
- Procesan toda la secuencia en paralelo, y no en serie como ocurre con las Redes Recurrentes.



GPT...



- Las redes transformers han hecho posible un avance sustancial en la generación de texto, así, OpenIA desarrolló chatGPT, un modelo que además de generar texto, lo hace de una forma aplicable a una conversación por chat con un humano.
- Entonces, GPT es un **modelo generativo con entrenamiento no supervisado**, pero chatGPT funciona por entrenamiento por refuerzo, y está entrenado con datos residentes en plataformas como Wikipedia.
- Hay que tener en cuenta que **los resultados pueden no ser exactos**, por tanto, siempre vale la pena revisar la información que resulta del uso de esta herramienta.



GPT...

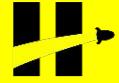
- La red neuronal de tipo GPT (Generative Pre-trained Transformer) utiliza una arquitectura de codificación bidireccional, que se entrena para **predecir** la siguiente palabra en una secuencia de texto dada.
- Una vez que se tiene entrenada la red GPT, se puede implementar un modelo de lenguaje conversacional como chatGPT, que utiliza la red GPT para **generar respuestas coherentes y contextuales** en una conversación en lenguaje natural.
- chatGPT se entrena en una gran cantidad de datos de conversación para **aprender** a generar respuestas que sean relevantes y coherentes con el contexto de la conversación.

¿PREGUNTAS?



¿Alguien dijo Homework?





¡Feedback!

Dispones de un **formulario** en:

shield icon Homeworks

shield icon Guías de clase

shield icon Slack

Click on me



HENRY

