

# Curso Inteligencia Artificial Clásica y Cuántica

Nodos de Inovación Especializados. Ruta N. Medellín, Colombia

## ¿Cómo Funciona la “Inteligencia Artificial”?

Jorge Mahecha Gómez

Grupo de Física Atómica y Molecular

Instituto de Física, Universidad de Antioquia

Medellín, Colombia

[jorge.mahecha@udea.edu.co](mailto:jorge.mahecha@udea.edu.co)

Agosto 31, 2023



El conocimiento  
es de todos

Minciencias



UNIVERSIDAD  
DE ANTIOQUIA

# RESUMEN

La llamada “**inteligencia artificial**” (IA), es un campo interdisciplinario que involucra la estadística, problemas matemáticos de optimización, las ciencias de la computación, algunas ramas de la ingeniería, ciencias de la cognición y el lenguaje, y ramas de la filosofía tales como la lógica y la epistemología. Consta de diferentes métodos para realizar inferencias a partir de ciertos datos, la máquina debe “aprender” de los datos de entrada y luego “tomar decisiones”. Puede decirse que la IA se implementa con ayuda de los métodos del aprendizaje automático (AA) o machine learning (ML).

Se presentan los conceptos básicos del aprendizaje automático. Se comentan los métodos llamados **Redes Neuronales**, con énfasis en las redes neuronales de muchas capas. Se ilustra lo anterior con algunos ejemplos. Se comentan algunos paquetes de redes neuronales como el Pytorch y se ilustra su uso con algunos ejemplos, entre ellos el reconocimiento de imágenes.

# COMPUTACIÓN

La computación digital se basa en la idea de utilizar máquinas universales que pueden resolver cualquier problema lógico o matemático. Una de esas máquinas es **la máquina universal de Turing**. Otro tipo de máquina que utiliza **los circuitos digitales** se basa en puertas lógicas, con las únicas puertas lógicas de un bit, la identidad y la NOT. Además, existen muchas puertas lógicas de dos bits, incluidas AND, OR, NAND, NOR, XOR y XNOR. Estas puertas lógicas se implementan utilizando dispositivos electrónicos como transistores y diodos, y ópticos. Su escalabilidad permite la implementación de lógica y de funciones de complejidad arbitraria. Se ha demostrado que AND, OR, NOT y COPY forman un conjunto de puertas universales que se pueden utilizar para computar cualquier función booleana.

# COMPONENTES DE LA MÁQUINA DE TURING

Una **Cinta** dividida en celdas. Cada celda puede contener un símbolo perteneciente a un alfabeto.

Una **Cabeza** ubicada frente a la cinta que se puede mover a la izquierda o la derecha y sirve para leer los símbolos grabados en la cinta.

Un **Registro de Estados** que almacena el estado de la máquina.

Una **Tabla de Instrucciones**, que contiene instrucciones predeterminadas dependientes del Estado de la máquina y del valor de la celda ubicada al frente, una de las cuales se ejecuta dependiendo de dicho estado y del valor de la celda.

Ejemplos de instrucciones son escribir un símbolo en la celda, mover la cabeza a la derecha o a la izquierda, permanecer al frente de la misma celda, etc.

Una máquina de Turing tiene 7 características: Un conjunto finito y no vacío de estados; un conjunto finito y no vacío de símbolos; el símbolo en blanco; la función de transición (L o desplazamiento a la izquierda, R o desplazamiento a la derecha, N o no desplazamiento); el estado inicial; el conjunto de estados finales o aceptables; el conjunto de instrucciones.



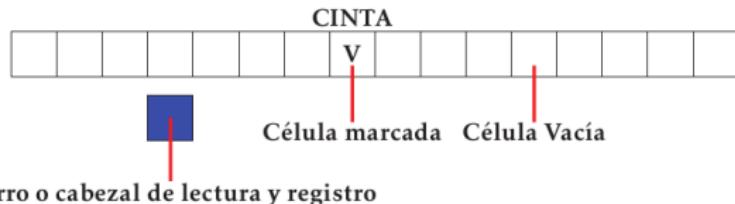
Mán Castro

## **El Arte de Razonar, La Matemática como diversión**

2º Edición

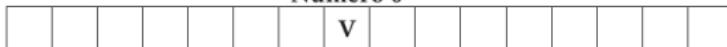
## 9.1. ¿Qué es una Máquina de Post?

Es una cinta infinita integrada por celdas que pueden estar marcadas o vacías y además también tiene una celda externa llamada “*carro*” o “*cabezal de lectura y registro*”.

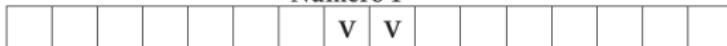


Los números naturales se señalan en una Máquina de Post de la siguiente manera:

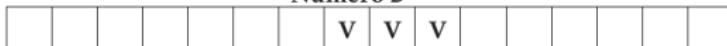
Número 0



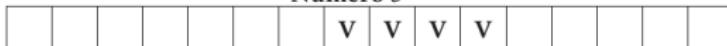
Número 1



Número 2



Número 3



Y así sucesivamente.

En 1936 Emil L. Post y Alan M. Turing publicaron, independientemente y por caminos distintos, sendos artículos en donde anticipándose a la aparición de las computadoras universales, presentaban en forma teórica las características fundamentales que deben tener las máquinas que estén en capacidad de Calcular.

Una función se dice *CALCULABLE*, si viene dada por un algoritmo cualquiera, cuyo dominio es el dominio de aplicabilidad del algoritmo y a cada elemento del dominio le hace corresponder el elemento resultante de la aplicación de este algoritmo a dicho elemento.



Emil Post (1897-1954)

Funciones usuales de la aritmética como la función siguiente de que calcula el número siguiente de un número natural, la función que calcula la suma, la que calcula el producto, la que calcula el cociente, la que calcula el máximo común divisor de dos números, la que calcula del mínimo común múltiplo de dos números, la que calcula la parte entera de un número, y muchas más son calculables mientras que la función:



Alan Turing (1912-1954)

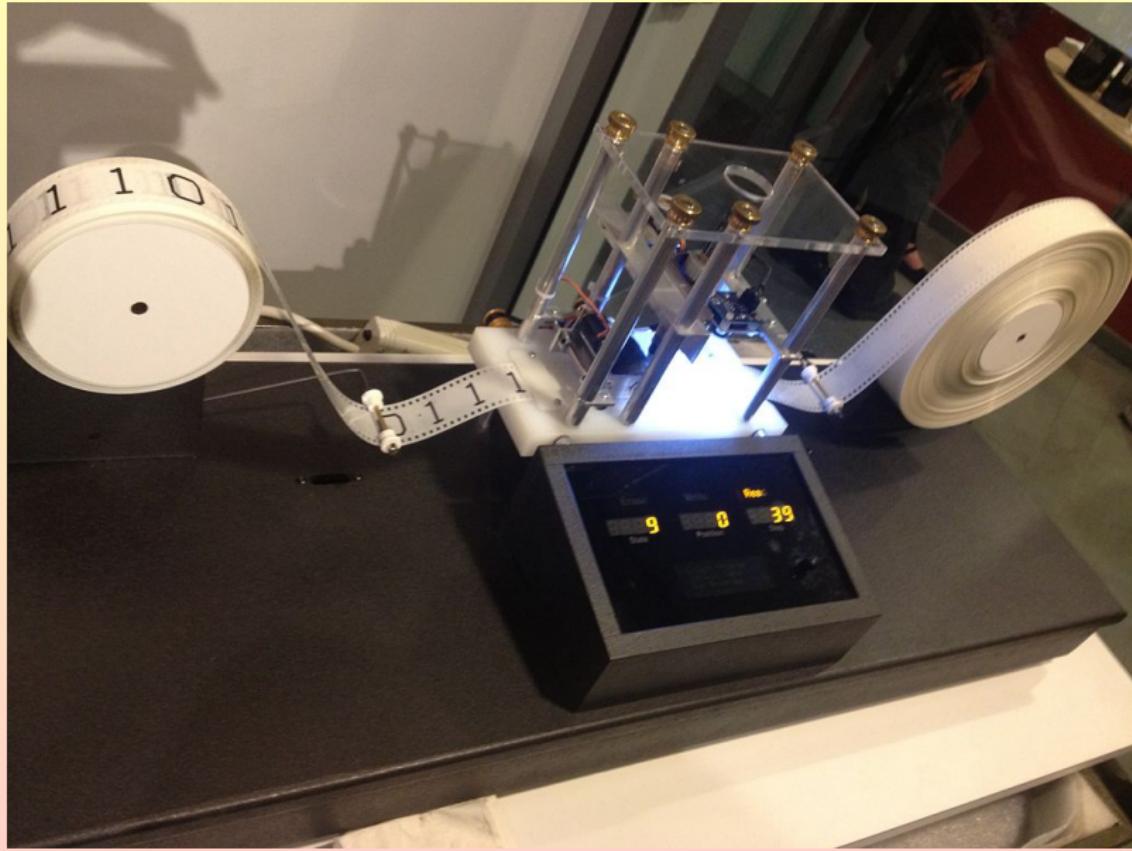


Imagen de Google

## ChatGPT

Please state the differences between a Post machine and a Turing machine.

Please cite some references about physical implementations of a Post machine or a Turing machine.

The “Game of Life” can be simulated in a Turing machine, or viceversa?

A Neural Network can simulate a Turing machine and viceversa?

Beginner's Guide to Universal Approximation Theorem. Khushee Upadhyay. 2021. <https://www.analyticsvidhya.com/blog/2021/06/beginners-guide-to-universal-approximation-theorem/>

# PUERTAS LÓGICAS

Dos modelos equivalentes para cualquier computador: Máquina de Turing. Modelo de circuito.

Un circuito está formado por puertas y conexiones entre puertas. Las “Conexiones” transportan información y las “Puertas” transforman la información.

Ejemplos de puertas de un bit ( $f : \{0, 1\} \rightarrow \{0, 1\}$ ) son la *Identidad* (llamada también puerta de Buffer), y la *NOT*.

Puerta Identidad

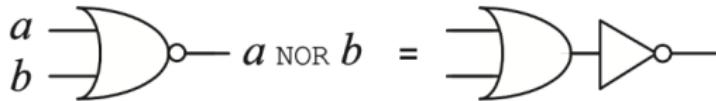
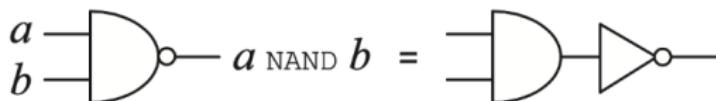
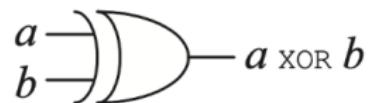
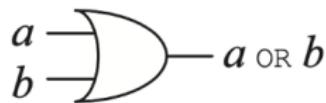
a	a
0	0
1	1

Puerta NOT

a	a'
0	1
1	0

Relación entre lógica y aritmética:  $\text{NOT}(a) = a' = 1 - a$

Ejemplos de puertas de 2 Bits ( $f : \{0, 1\}^2 \rightarrow \{0, 1\}$ ) son *OR*, *XOR* (OR exclusivo), *NAND* (AND negado) and *NOR* (OR negado). Esas puertas tienen 2 bits de entrada y 1 bit de salida. Por lo tanto son no invertibles.

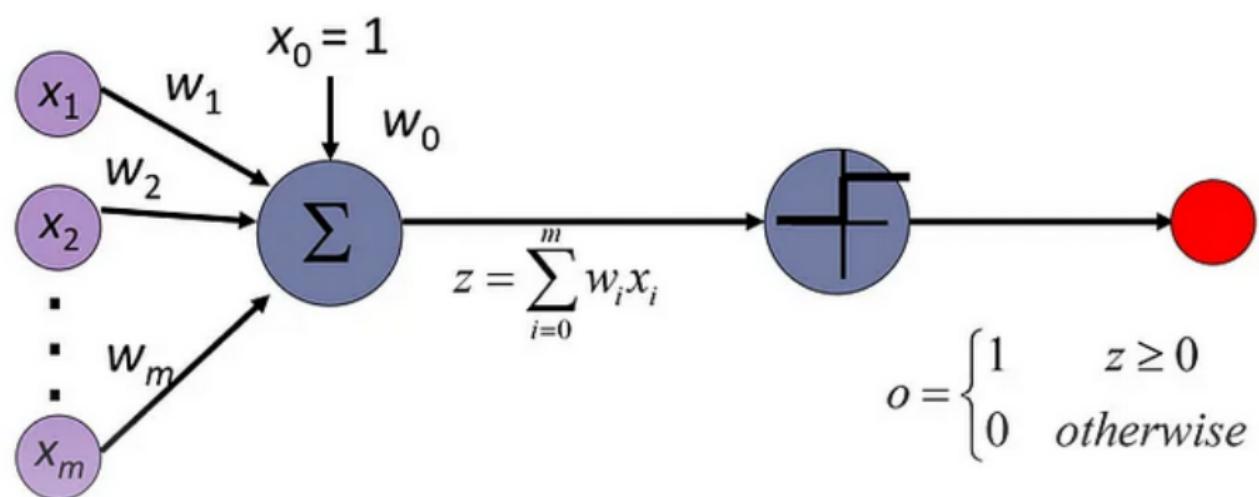


Relación entre lógica y aritmética:

$$\begin{array}{ll}
 a \text{ AND } b = a \wedge b = ab, & a \text{ OR } b = a \vee b = a + b - ab, \\
 a \text{ XOR } b = a \oplus b = a + b, & a \text{ NAND } b = a \uparrow b = (a \wedge b)' = (ab)' = 1 - ab, \\
 a \text{ NOR } b = a \downarrow b = (a \vee b)' = (a + b - ab)' = 1 - a - b + ab
 \end{array}$$

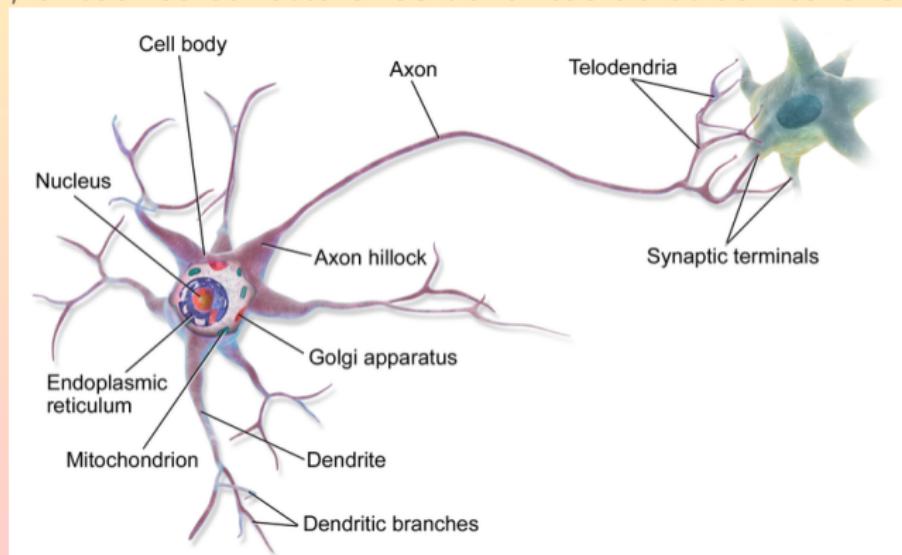
## REDES NEURONALES

**El perceptrón.** La neurona  $\Sigma$  tiene entradas de intensidad  $x_i$  a través de enlaces que tienen peso  $w_i$ . La suma de todas las entradas vale  $z$ . Se muestra la “función de activación” y la salida única (binaria)  $o$ .



# REDES NEURONALES BIOLÓGICAS

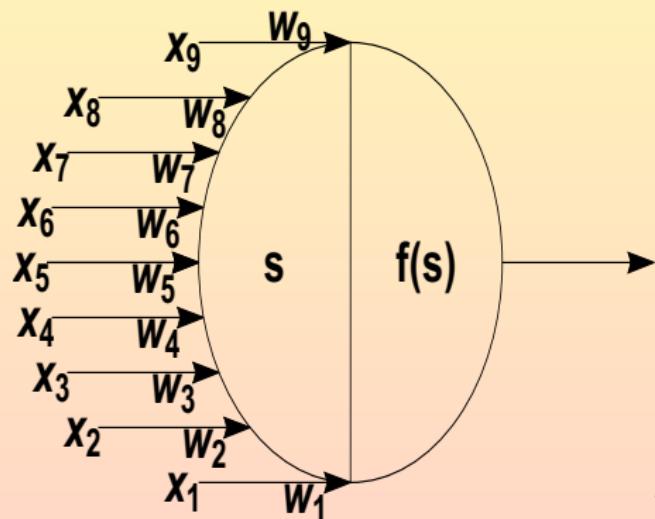
El perceptrón es una caricatura de una neurona biológica (llamada regla de Hebb, 1949). A través de las **dendritas** la neurona recibe “impulsos eléctricos”. Se dice que cuando la suma de tales impulsos supera cierto valor umbral, la neurona produce “impulsos eléctricos” que salen a través de su **axón**, el cual se conecta a las dendritas de otras neuronas.



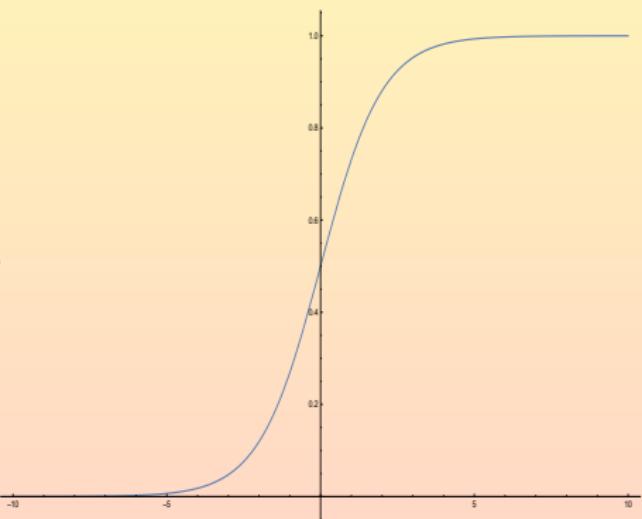
# PREGUNTAS A CHATGPT

1. En una neurona biológica se identifican las dendritas, el núcleo y el axón. Se dice que cada dendrita aporta cierto “potencial eléctrico” con un peso determinado como entrada al núcleo, en donde se realiza una suma de todas las contribuciones; si dicha suma supera cierto valor umbral entonces se emite una señal eléctrica por el axón. Esta descripción es usada para definir las llamadas “redes neuronales” de las ciencias de datos. ¿En la actualidad se considera válida esta descripción del funcionamiento de las neuronas biológicas, o hay descripciones alternativas? ¿Hay soporte experimental de este modelo?
2. Dices que “la realidad biológica es mucho más compleja y sutil que esta simplificación.” ¿Podrías dar algunos de los detalles de dicha complejidad que hacen incompleto el modelo de “redes neuronales” para describir las redes neuronales biológicas?

# PERCEPTRÓN Y FUNCIÓN DE ACTIVACIÓN SIGMOIDE



Perceptrón de una sola capa.



Sigmoide

# REGRESIÓN LOGÍSTICA

Se define

$$h_{\mathbf{w}}(\mathbf{x}) = f(\mathbf{w} \cdot \mathbf{x}),$$

donde  $f(z)$  es la función de activación logística  $[1/(1 + e^{-z})]$  y  $\mathbf{x}$  y  $\mathbf{w}$  son vectores de dimensión  $D + 1$ , con  $x_0 = 1$ ,

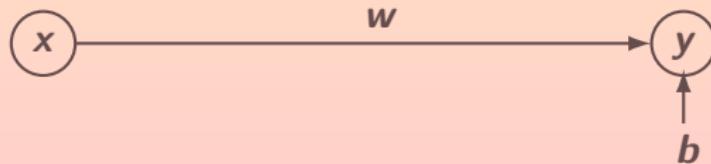
$$\mathbf{w} \cdot \mathbf{x} = w_0 + \sum_{i=1}^D w_i x_i.$$

El problema de regresión consiste en ajustar  $\mathbf{w}$  conociendo  $N$  series de datos  $\mathbf{x} = \{1, x_1, \dots, x_D\}$ . Una manera simple es usar el ajuste de **mínimos cuadrados**. Otra es maximizar una función de **plausibilidad**. Llamamos  $x_i$  a las componentes del vector  $\mathbf{x}$  y  $\mathbf{x}_\lambda$ , con  $\lambda = 1, \dots, N$  las  $N$  realizaciones del mismo.

Suponemos que a la entrada  $\mathbf{x}_\lambda$  le corresponde una salida del perceptrón  $y_\lambda$ , la cual puede ser  $y = 0$  o  $y = 1$  (perceptrón binario).

# UN “PERCEPTRÓN” QUE PRODUCE LA PUERTA NOT

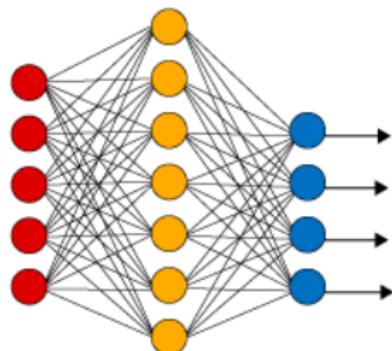
La entrada es  $x$  y la salida  $y$ , binarias. La salida deseable es el NOT de la entrada. Es decir, que si  $x = 0$  entonces  $y = 1$  y si  $x = 1$  entonces  $y = 0$ . Si se hace  $w = 1$ , entonces  $s = x$ . Si se toma como función de activación la identidad entonces  $f(s) = s$ . Resulta así la puerta lógica identidad. Sin embargo, si se suma cierta cantidad  $b$ , llamada en inglés “bias” (que se puede traducir como “sesgo”, “polarización” o “desplazamiento”), puede lograrse la puerta lógica NOT.  $s = w \cdot x + b$ ,  $y = f(s)$ . Tomando  $f = Id$ ,  $w = 1$ ,  $b = 1$  se obtiene  $f(s) = w \cdot x + b = 1 \cdot x + 1 = x + 1$ , lo cual, con suma en la base binaria, da  $0 + 1 = 1$  y  $1 + 1 = 0$ .



# “DEEP LEARNING” O APRENDIZAJE PROFUNDO

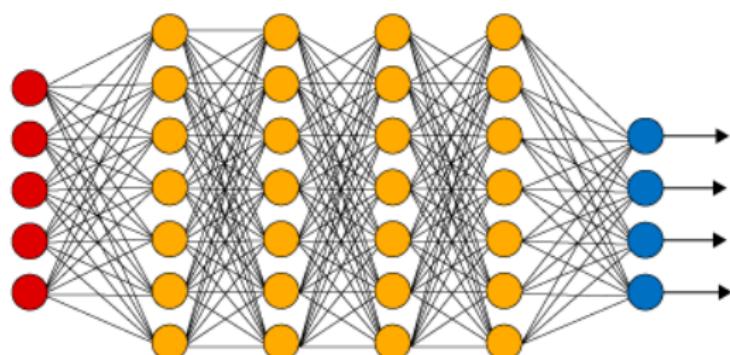
Preguntar a Bard: Explique el deep learning o aprendizaje profundo.

**Simple Neural Network**



● Input Layer

**Deep Learning Neural Network**



● Hidden Layer

● Output Layer

Imagen de Google

# EL MÉTODO DE LOS MÍNIMOS CUADRADOS

La *media aritmética* o *valor medio aritmético* se define mediante la relación

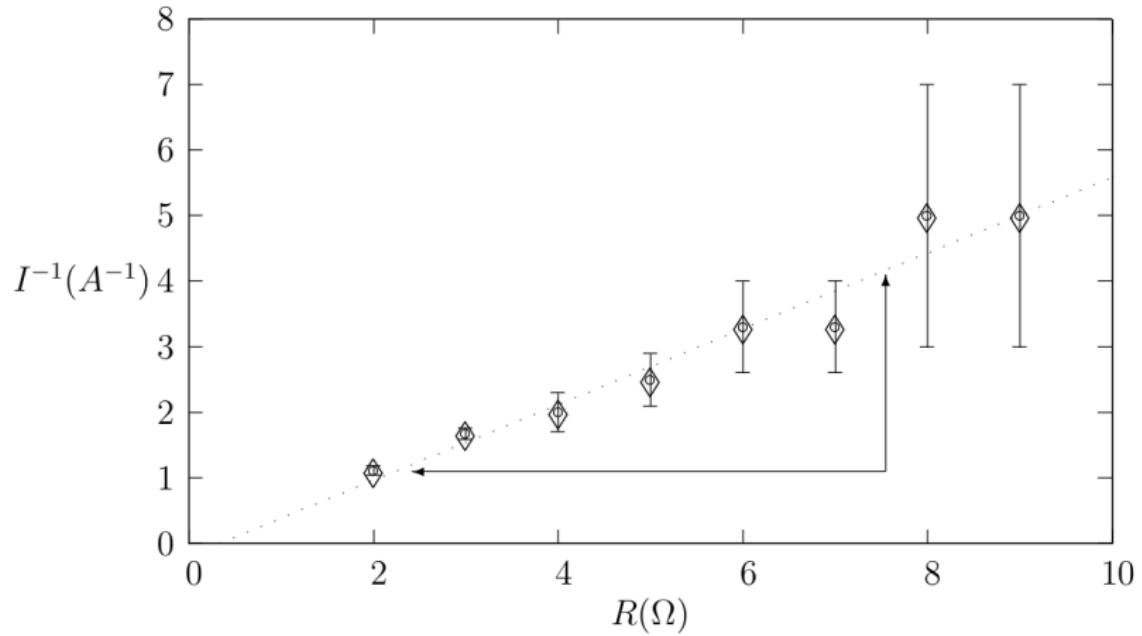
$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i.$$

La *desviación estándar* es una función de las desviaciones que sirve de indicador de la dispersión. Se define como

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}.$$

$R (\Omega)$	$I (A)$	$\Delta I (A)$	$1/I (A^{-1})$	$(1/I^2) \Delta I (A^{-1})$
2	0.9	0.06	1.11	0.07
3	0.6	0.03	1.67	0.09
4	0.5	0.08	2.0	0.3
5	0.4	0.06	2.5	0.4
6	0.3	0.06	3.3	0.7
7	0.3	0.06	3.3	0.7
8	0.2	0.08	5	2
9	0.2	0.08	5	2

Datos de las medidas de corriente y resistencia en un circuito eléctrico.



Gráfica de  $1/I$  contra  $R$  elaborada con los datos de la tabla anterior. Las líneas verticales alrededor de cada punto representan el error en  $1/I$ . El cambio de variable,  $1/I$  en vez de  $I$ , llevó a la no uniformidad del margen de error.

A la línea recta de la figura anterior le corresponde una ecuación de la forma:

$$y = m \cdot x + b,$$

en donde “ $y$ ” es la variable dependiente,  $1/I$  en este caso, “ $x$ ” es la variable independiente,  $R$  en este caso, “ $m$ ” es la **pendiente** y “ $b$ ” es el **intercepto** de la recta con el eje  $y$ .

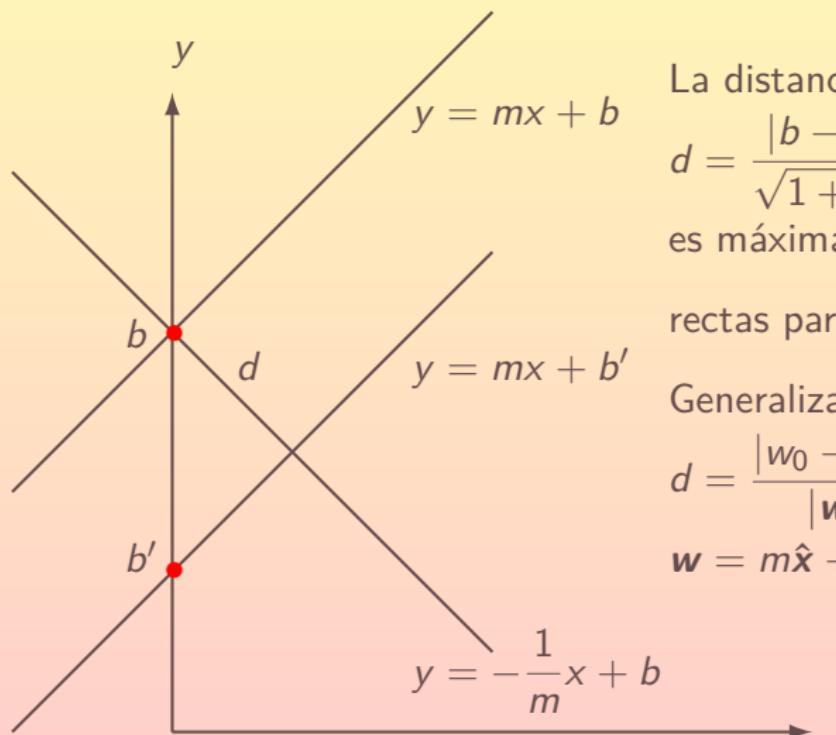
En los estudios de los datos se acostumbra escribir dicha recta en la forma

$$y = w \cdot x + b,$$

donde a “ $w$ ” lo llaman el **peso** (weight) y a “ $b$ ” el **sesgo** (bias). Sin embargo, se pueden definir 2 vectores bidimensionales, el vector de “puntos”  $\mathbf{x} = \{x, 1\}$  y el vector de “pesos”  $\mathbf{w} = \{m, b\}$ , con lo cual

$$y = \mathbf{w} \cdot \mathbf{x}.$$

# RECTAS DE LA FORMA $mx - y + b = 0$



La distancia vale  
$$d = \frac{|b - b'|}{\sqrt{1 + m^2}},$$
  
es máxima si  $m = 0$ ,

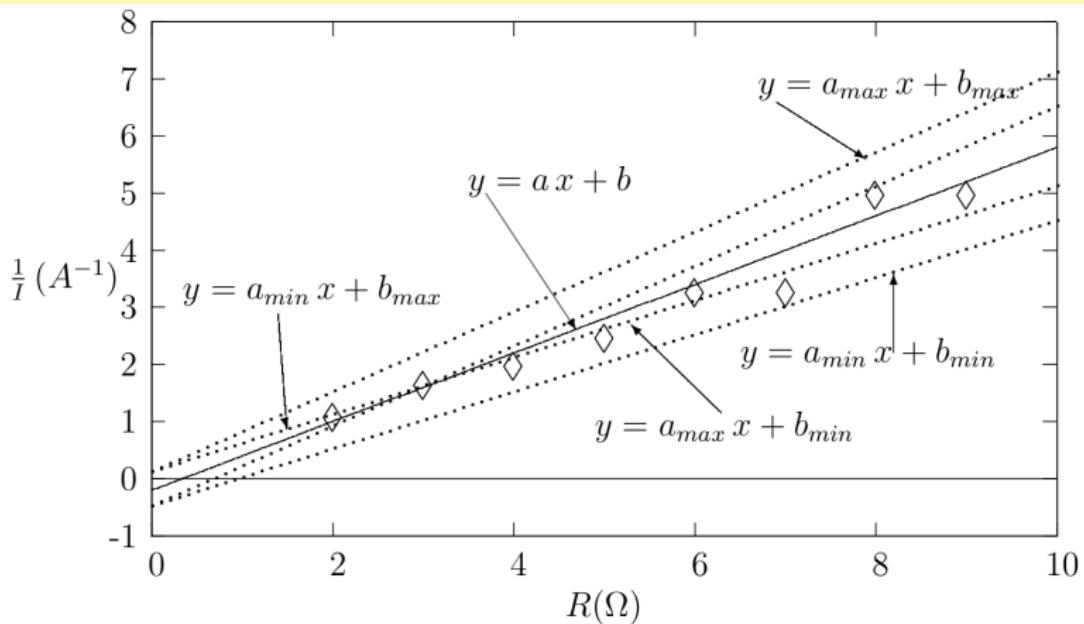
rectas paralelas.

Generalización:

$$d = \frac{|w_0 - w'_0|}{|\mathbf{w}|}.$$

$$\mathbf{w} = m\hat{\mathbf{x}} - 1\hat{\mathbf{y}}, w_0 = b.$$

# LA RECTA ÓPTIMA



Gráfica de  $1/I$  contra  $R$  elaborada con las datos de la tabla, de acuerdo a la teoría de mínimos cuadrados. Nótense las dos líneas límites entre las cuales deben estar todos los datos, determinadas por las barras de error.

## LA SOLUCIÓN OBTENIDA APLICANDO EL LSM

Las variables de ese ejemplo están relacionadas por la ecuación de la línea recta,  $y = m \cdot x + b$ . Si se conocen los parámetros  $m$  y  $b$  (pendiente e intercepto con el eje de las ordenadas), podemos trazar fácilmente la recta. Como se espera una única recta experimental, los datos experimentales han de dar lugar a valores únicos de  $m$  y  $b$ . Para hallar por el método de los mínimos cuadrados tales parámetros es necesario calcular la suma de los cuadrados de las desviaciones verticales de los puntos experimentales respecto a la recta buscada y luego minimizar tal suma respecto a los parámetros  $m$  y  $b$ .

Consideremos que  $(x_i, y_i)$ ,  $i = 1, 2, 3, \dots, n$ , son las  $n$  parejas de datos obtenidos experimentalmente. Se quiere encontrar la recta óptima, cuya coordenada vertical  $y$  es una función de  $x$  dada por  $y = mx + b$ . Se deben encontrar **valores óptimos** de  $a$  y  $b$ . Para ello se define la “función de costo”

$$Q = \sum_{i=1}^n [y_i - (mx_i + b)]^2 = \sum_{i=1}^n (y_i^2 + m^2x_i^2 + b^2 - 2mx_iy_i - 2by_i + 2mbx_i)$$

y se requiere que los parámetros  $a$  y  $b$  sean tales que  $Q$  sea mínimo,

$$\frac{\partial Q}{\partial m} = 0, \quad \frac{\partial Q}{\partial b} = 0.$$

Si se define un vector bidimensional  $\mathbf{u}$  con componentes  $m$  y  $b$ , la minimización de la función de costo se puede escribir en términos del **gradiente** respecto al vector  $\mathbf{u}$ ,

$$\nabla_{\mathbf{u}} Q = \mathbf{0}.$$

Explícitamente,

$$\sum_{i=1}^n (2mx_i^2 - 2x_iy_i + 2bx_i) = 0, \quad \sum_{i=1}^n (2b - 2y_i + 2mx_i) = 0.$$

Estas dos ecuaciones se pueden escribir como

$$m \sum_{i=1}^n 2x_i^2 + b \sum_{i=1}^n 2x_i = \sum_{i=1}^n 2x_iy_i, \quad m \sum_{i=1}^n 2x_i + b \sum_{i=1}^n 2 = \sum_{i=1}^n 2y_i,$$

lo cual se expresa como un sistema algebraico de 2 ecuaciones lineales con 2 incógnitas,

$$\begin{pmatrix} 2 \sum_{i=1}^n x_i^2 & 2 \sum_{i=1}^n x_i \\ 2 \sum_{i=1}^n x_i & 2n \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 2 \sum_{i=1}^n x_iy_i \\ 2 \sum_{i=1}^n y_i \end{pmatrix}.$$

$$\begin{pmatrix} 2n\bar{x}^2 & 2n\bar{x} \\ 2n\bar{x} & 2n \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 2n\bar{x} \cdot \bar{y} \\ 2n\bar{y} \end{pmatrix}.$$

$n$  es el número de puntos de la gráfica,  $(\bar{x}, \bar{y})$  es la coordenada del “centro de gravedad” del conjunto de datos y  $\bar{-}$  denota la media aritmética.

Con ayuda del determinante de la matriz  $2 \times 2$  se calcula la inversa y luego se halla la siguiente solución para las incógnitas  $m$  y  $b$ .

$$m = \frac{\bar{x} \cdot \bar{y} - \bar{x} \bar{y}}{\bar{x}^2 - \bar{x}^2}, \quad b = \frac{\bar{y} \bar{x}^2 - \bar{x} \bar{x} \cdot \bar{y}}{\bar{x}^2 - \bar{x}^2} = \bar{y} - m \bar{x}.$$

Los errores de estas cantidades se calculan mediante las fórmulas siguientes:

$$s_m^2 = \left| \frac{\bar{y}^2 + m^2 \bar{x}^2 + b^2 - 2m \bar{x} \cdot \bar{y} - 2b \bar{y} + 2m b \bar{x}}{(n-2)(\bar{x}^2 - \bar{x}^2)} \right|^2,$$

$$s_b^2 = s_m^2 \cdot \bar{x}^2.$$

En la siguiente tabla están transcritos los datos del experimento, llamando  $x_i$  a los valores de la resistencia y  $y_i$  a los inversos de las corrientes.

$x_i$	$y_i$	$x_i^2$	$y_i^2$	$x_i \cdot y_i$	
2	1.1	4	1.2	2.2	$n = 8$
3	1.7	9	2.9	5.1	
4	2.0	16	4	8.0	
5	2.5	25	6.2	12.5	
6	3.3	36	10.9	19.8	$\bar{x} = 5,5 \quad \bar{y} = 3,0$
7	3.3	49	10.9	23.1	
8	5.0	64	25	40.0	$\bar{x} \cdot \bar{y} = 19,5$
9	5.0	81	25	45.0	
SUMA	44	23.9	284	86.1	$\bar{y^2} = 10,8$
				155.7	$\bar{x^2} = 35,5$

Datos para el cálculo de la pendiente y el intercepto usando las fórmulas del método de mínimos cuadrados.

Cuando se reemplazan los valores apropiados de esta tabla, se obtiene

$$m = 0,6 \text{ } (A^{-1} \cdot \Omega^{-1}) \quad y \quad b = -0,2 \text{ } (A^{-1}).$$

Estos valores definen la recta óptima  $y = 0,6 \cdot x - 0,2$ .

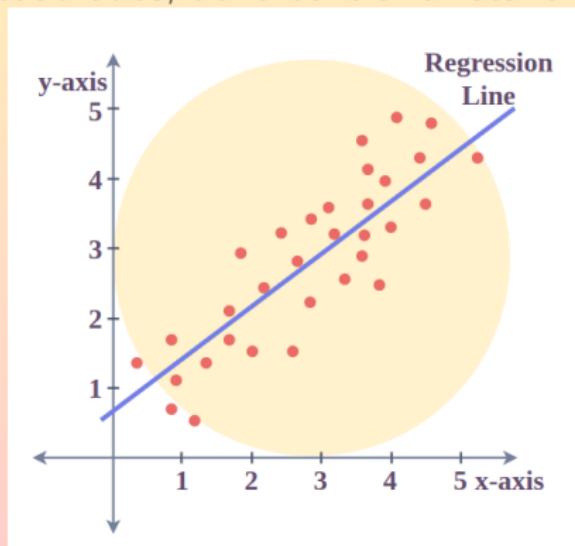
Físicamente se espera que el valor del parámetro  $b$  sea cero (corriente infinita cuando la resistencia es cero), pero el cálculo da  $b = -0,2$ . No podemos imponer que la recta trazada pase por el punto  $(0, 0)$ , el cual tampoco puede considerarse como punto experimental porque no es posible medir una corriente infinita. Respecto al hecho de no pasar la curva por el origen sólo nos quedaría evaluar el error en el cálculo de  $b$  y verificar si se cumple o no que

$$b - s_b \leq 0 \leq b + s_b.$$

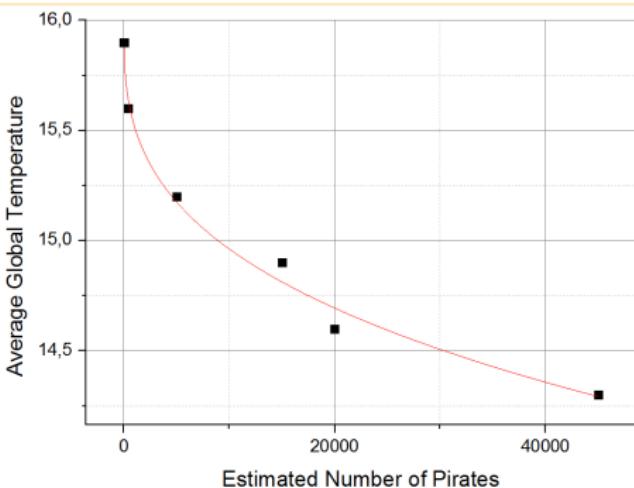
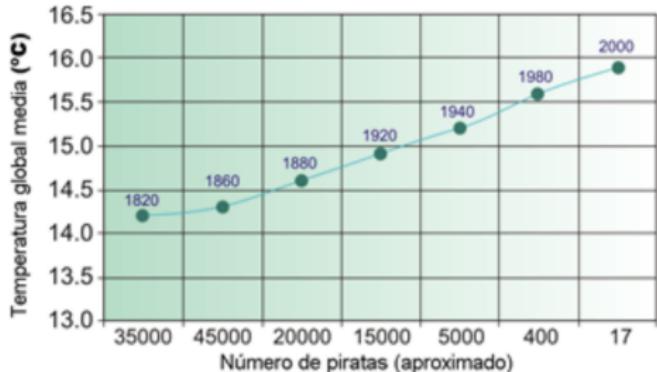
Reemplazando en las fórmulas correspondientes los valores requeridos, que se obtienen de la tabla, obtenemos  $s_m = 0,1$ ,  $s_b = 0,3$ , y por lo tanto  $-0,2 - 0,3 \leq 0 \leq -0,2 + 0,3$ , con lo cual concluimos que el valor  $b = -0,2$  efectivamente es atribuible a los errores experimentales, y no a una discrepancia respecto a la **ley de Ohm**.

# LA LÍNEA DE REGRESIÓN

Si cada punto rojo representara una masa de igual magnitud, la línea azul fuera una varilla rígida, cada masa estuviera unida a la varilla azul con una varillita perpendicular sin masa, y el campo de gravedad actuara perpendicular al plano de la gráfica, el sistema quedaría estático si la línea azul es de mínimos cuadrados, de lo contrario rotaría.



## Temperatura global vs. N° de piratas



With a decrease in the number of pirates,  
there has been an increase in global  
warming over the same period. Therefore,  
global warming is caused by a lack of pirates.

Even more compelling: Somalia has the  
highest number of Pirates AND the lowest  
Carbon emissions of any country.

Coincidence?

— Tim Ferriss —



Imagen de Google

# QUIROMANCIA

EL DESTINO Y LA SUERTE EN LA PALMA DE LA MANO

Francis King



Imagen de Google

[https://en.wikipedia.org/wiki/Church-Turing\\_thesis](https://en.wikipedia.org/wiki/Church-Turing_thesis)

**La tesis de Church-Turing** dice que cualquier función para cuya evaluación se siguen reglas matemáticas definidas puede evaluarse mediante una máquina de Turing. Dado que existe equivalencia entre la máquina de Turing y el modelo de circuitos de la computación, se sigue de dicha tesis que cualquier función puede evaluarse con un computador. Y si las redes neuronales se pueden implementar con un modelo de circuitos, se seguiría una forma análoga de la tesis de Church-Turing con redes neuronales. Similarmente con el juego de la vida y otras equivalencias con la máquina de Turing.

#### Acerca de correlación y causalidad:

<https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.95.244101>

<https://journals.aps.org/pre/abstract/10.1103/PhysRevE.92.022126>

<https://arxiv.org/pdf/1403.6496.pdf>

Scalar



0D tensor

Vector



1D tensor

Matrix



2D tensor

# TENSORES EN MACHINE LEARNING

Son matrices multidimensionales que pueden almacenar cualquier tipo de datos, incluyendo números, textos, imágenes y videos. Los tensores también se pueden utilizar para transformar datos de una forma a otra, como rotar una imagen o traducir un texto.

Los tensores de 2 índices se utilizan para representar texto. El primer índice del tensor representa la posición del carácter en la palabra, y el segundo índice representa el carácter en sí. O también, el primer índice del tensor se puede usar para representar la posición de la palabra en el texto, y el segundo índice para representar los caracteres de la palabra.

Un tensor puede representar una imagen como una matriz de píxeles. Una imagen se puede representar como un tensor de 3 índices, con un índice para cada fila, columna y canal de color.

Un ejemplo de tensor de 4 índices que se usa en inteligencia artificial es un tensor que representa una imagen 3D. Este tensor tendría un índice para cada fila, columna, profundidad y canal de color.

Por ejemplo, una imagen 3D de 200x200x100x3 píxeles tendría un tensor de 4 índices con la siguiente forma:

$$(\text{filas}, \text{columnas}, \text{profundidad}, \text{canales}) = (200, 200, 100, 3)$$

Cada píxel del tensor representaría la intensidad de un color en una ubicación específica de la escena.

Otro ejemplo de tensor de 4 índices que se usa en inteligencia artificial es un tensor que representa un video. Este tensor tendría un índice para cada fotograma, fila, columna y canal de color.

Por ejemplo, un video de 100 fotogramas de 200x200x3 píxeles tendría un tensor de 4 índices con la siguiente forma:

$$(\text{fotogramas}, \text{filas}, \text{columnas}, \text{canales}) = (100, 200, 200, 3)$$

Aprendizaje automático: Los tensores se utilizan en una variedad de algoritmos de aprendizaje automático, como las redes neuronales. Por ejemplo, una red neuronal se puede representar como un conjunto de tensores que se conectan entre sí.

Una red neuronal se puede representar como un tensor de 3 índices, con un índice para cada capa, neurona y entrada. Los tensores de 2, 3 o más índices se utilizan para representar datos de entrada, pesos y salida de redes neuronales.

Los tensores de 2 índices son suficientes para representar datos unidimensionales y bidimensionales, como listas y matrices. Para representar datos tridimensionales o más, se necesitan tensores de 3 índices o más.

Los tensores de 5 índices también se pueden utilizar para representar datos más complejos, como redes neuronales de varias capas.

Otro ejemplo de tensor de 5 índices que se usa en inteligencia artificial es un tensor que representa un conjunto de datos de audio. Este tensor tendría un índice para cada muestra, canal, frecuencia, tiempo y dimensión. Por ejemplo, un conjunto de datos de audio de 1000 muestras de 16 bits, 44,1 kHz, estéreo tendría un tensor de 5 índices con la siguiente forma: (muestras, canales, frecuencia, tiempo, dimensiones) = (1000, 2, 44100, 1, 2)

Para convertir un NumPy array a TensorFlow tensor se usa

```
tensor = tf.convert_to_tensor(array)
```

La conversión de tensores de Numpy en tensores de Pytorch se puede realizar utilizando la función

```
torch.from_numpy()
```

## CARACTERÍSTICAS Y ETIQUETAS

Puede decirse que una red neuronal es un procedimiento para evaluar una “función” complicada,  $f$ , que tiene en su capa de entrada un conjunto de tensores  $X$  llamados Features (características) y en la capa de salida otro conjunto  $y$  con los Labels (etiquetas). La creencia básica consiste en la conexión entre la entrada y la salida.

$$y = f(X)$$

# CÁLCULOS DE LA PENDIENTE Y EL INTERCEPTO

## Algunos Notebooks

Ajuste de mínimos cuadrados con Numpy

[https://colab.research.google.com/drive/131Lbb2LS2gXe8VkJToYg0msQzm-NTU\\_A](https://colab.research.google.com/drive/131Lbb2LS2gXe8VkJToYg0msQzm-NTU_A)

Ajuste de mínimos cuadrados con Sklearn

[https://colab.research.google.com/drive/1SbDFJ3\\_bGmcjBPjNR0o1DYWckm5EpoPz](https://colab.research.google.com/drive/1SbDFJ3_bGmcjBPjNR0o1DYWckm5EpoPz)

Cálculo de la pendiente y el intercepto con Tensorflow

[https://colab.research.google.com/drive/1IDI8KB\\_xo7qh-VWO\\_qy\\_6Qs-k7m4BkTi](https://colab.research.google.com/drive/1IDI8KB_xo7qh-VWO_qy_6Qs-k7m4BkTi)

Cálculo de la pendiente y el intercepto con Tensorflow usando PGNN

[https://colab.research.google.com/drive/1DK1HI7HTuegTW36QYHJFmvnTb\\_wvjPhx#scrollTo=I44HQEv7tjOU](https://colab.research.google.com/drive/1DK1HI7HTuegTW36QYHJFmvnTb_wvjPhx#scrollTo=I44HQEv7tjOU)

Cálculo de la pendiente y el intercepto con Pytorch. También usando PGNN

<https://colab.research.google.com/drive/>

# EL CONJUNTO DE DATOS DE LA FLOR IRIS

Las tablas de Fisher contienen los datos morfológicos de la planta iris. Son 50 datos de cada una de los 3 tipos de la flor iris (setosa, virginica y versicolor). Cada registro contiene 5 entradas: **longitud y ancho de los cépalos** y **longitud y ancho de los pétalos**, en cm, y el **nombre** del correspondiente tipo de flor.

Fisher calculó un discriminante lineal para distinguir un tipo de otro. El correspondiente problema matemático forma parte de la clasificación estadística. Se puede implementar en aprendizaje automático, o ML, con diferentes algoritmos, uno de ellos llamado de Máquina de Soporte Vectorial (SVM).

[https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set](https://en.wikipedia.org/wiki/Iris_flower_data_set)



Versicolor

Virginica

Setosa

# ESPAZO TETRADIMENSIONAL

Algunos Datos de Ronald Fisher del Iris (1936)

Largo de sépalo	Ancho de sépalo	Largo de péntalo	Ancho de péntalo	Especie
5,0	2,0	3,5	1,0	<i>I. versicolor</i>
6,2	2,2	4,5	1,5	<i>I. versicolor</i>
6,0	2,2	5,0	1,5	<i>I. virginica</i>
6,0	2,2	4,0	1,0	<i>I. versicolor</i>
6,3	2,3	4,4	1,3	<i>I. versicolor</i>
5,5	2,3	4,0	1,3	<i>I. versicolor</i>
5,0	2,3	3,3	1,0	<i>I. versicolor</i>
4,5	2,3	1,3	0,3	<i>I. setosa</i>
5,5	2,4	3,8	1,1	<i>I. versicolor</i>
5,5	2,4	3,7	1,0	<i>I. versicolor</i>
4,9	2,4	3,3	1,0	<i>I. versicolor</i>
6,7	2,5	5,8	1,8	<i>I. virginica</i>

+++++ **xxxxxx**ooooooo

Ancho de pétalo

++++++ + x x xxxxxxxxooooxxxoooooo oo oo o

Largo de pétalo

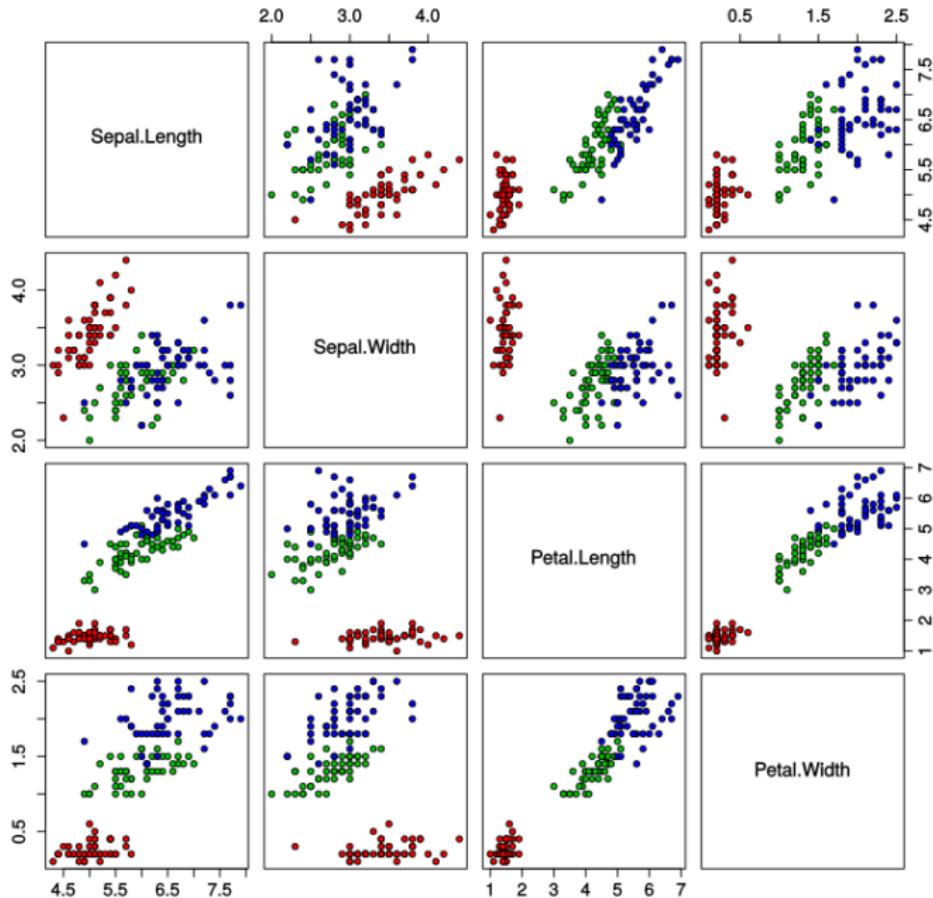
x xxxxooooooxxxx+ +

Ancho de sépalo

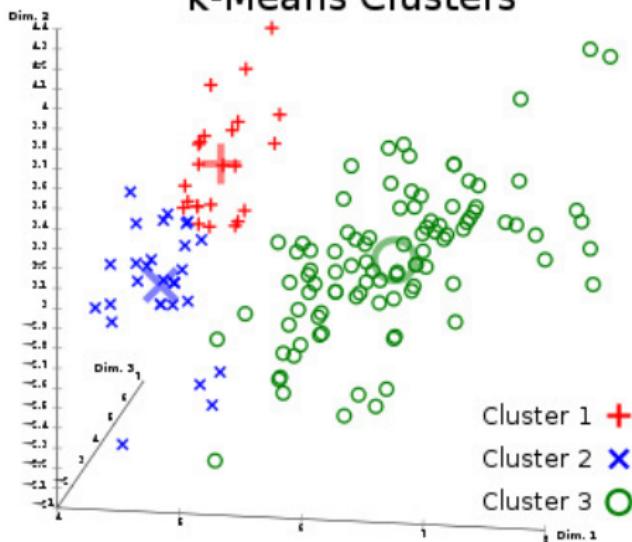
+++++xxxxxoooooxxxxxoooo oo o Largo de sépalo



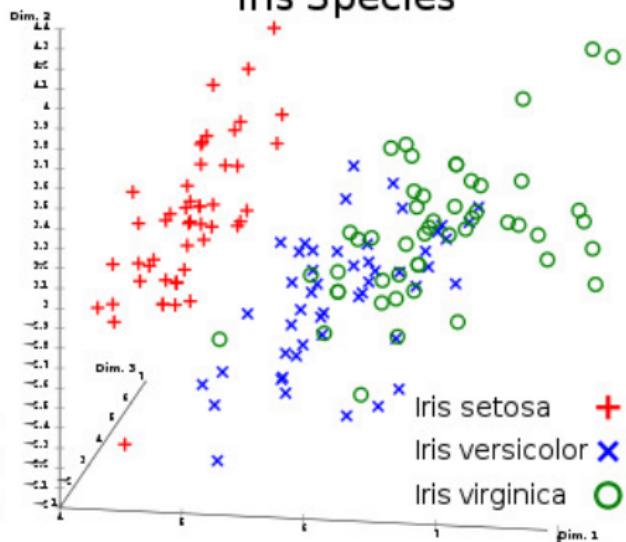
Iris Data (red=setosa,green=versicolor,blue=virginica)



## k-Means Clusters



## Iris Species



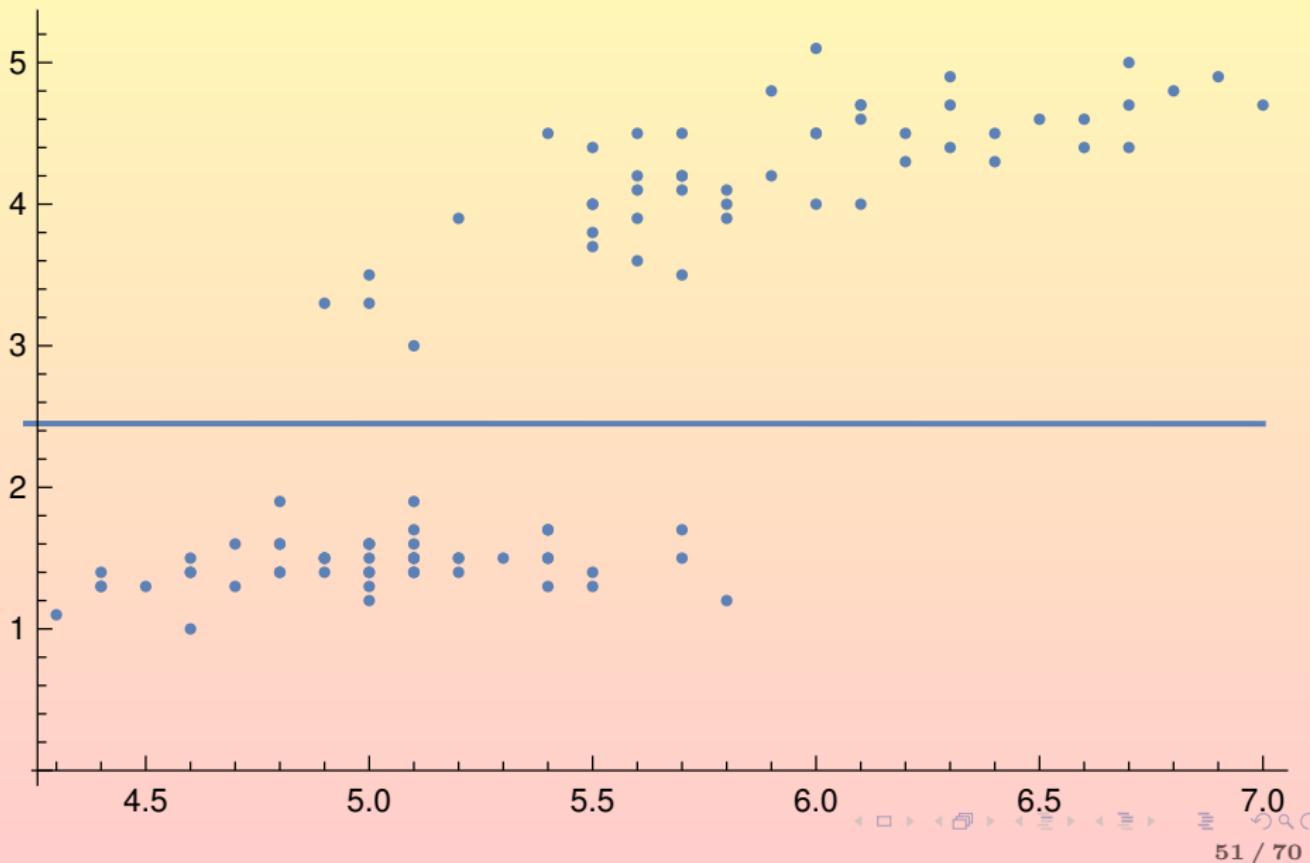
El problema: Encontrar dos hiperplanos (superficies 3D inmersas en el espacio 4D) que produzcan una separación de los puntos de los 3 tipos de iris. Lo anterior marcará fronteras de decisión para futuras tareas de clasificación.

Caso simple: Datos en 1D con sólo 2 clases. Se trata de hallar un punto  $x$  que se sitúe exactamente entre los miembros de las clases 1 y 2, tal que todos los puntos a la izquierda de  $x$  pertenecen a una clase y los de la derecha a otra.

Caso más general en  $D$  dimensiones con sólo 2 clases. Se trata de hallar un hiperplano  $D - 1$  dimensional que se sitúe exactamente entre los miembros de las clases 1 y 2, tal que todos los puntos a un lado pertenecen a una clase y los del otro lado a la otra clase. Se supone que las dos clases son disjuntas. El mejor hiperplano discriminador tiene la máxima distancia a los puntos de datos más próximos de cada clase, los cuales se llaman “vectores de soporte”.

Se tiene el siguiente problema matemático de optimización: Hallar el hiperplano de “máximo margen” o de “separación óptima”, el “más alejado” de los datos de entrenamiento.

## PUNTOS $(x_1, x_2)$ DE DOS CLASES



# HIPERPLANO DE SEPARACIÓN

Obedece la ecuación

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_D x_D = 0,$$

donde  $(x_1, x_2, \dots, x_D)$  es un punto situado sobre el hiperplano.

Las clases están conformadas por puntos a cada uno de los 2 lados del hiperplano. Si  $(x_1, x_2, \dots, x_D)$ , por fuera del hiperplano, es tal que

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_D x_D > 0,$$

decimos que ese punto pertenece a la clase  $y = 1$ . Si

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_D x_D < 0,$$

decimos que el punto  $(x_1, x_2, \dots, x_D)$ , por fuera del hiperplano, al “otro lado” del anterior, pertenece a la clase  $y = 2$ .

Por lo tanto la ecuación del hiperplano sirve de clasificador.

Los puntos del “soporte” son tales que pertenecen a las dos clases y están ubicados en hiperplanos paralelos al separador. Caracterizan la SVM. Definamos dos vectores  $D$  dimensionales  $\mathbf{r}$  y  $\mathbf{w}$  y un escalar  $w_0$ ,

$$\mathbf{r} = (x_1, x_2, \dots, x_D); \quad \mathbf{w} = (\beta_1, \beta_2, \dots, \beta_D); \quad w_0 = \beta_0.$$

Los puntos ubicados sobre el separador satisfacen

$$\mathbf{w} \cdot \mathbf{r} + w_0 = 0.$$

Los puntos ubicados sobre uno de los dos planos de soporte satisfacen relaciones similares.  $\mathbf{w}$  es común a los tres planos, pero el  $w_0$  obviamente es diferente, porque determina el sesgo de cada uno.

En el caso 2D, el “hiperplano” separador tiene una dimensión menor, es una línea recta.

# CLASIFICACIÓN DE LOS TIPOS DE FLOR IRIS

book [http://localhost:8888/notebooks/Iris\\_flower\\_classification.ipynb](http://localhost:8888/notebooks/Iris_flower_classification.ipynb)

# RECONOCIMIENTO DE IMÁGENES

[ab.research.google.com/drive/1bexwbro5XFGsPpZncFc\\_TdDv8E8-Nmh0#s](https://ab.research.google.com/drive/1bexwbro5XFGsPpZncFc_TdDv8E8-Nmh0#s)

# RN CONVOLUCIONALES

Son similares a las RN en general, pero están adaptadas a situaciones en las cuales la entrada contiene información “redundante” o que contiene detalles que pueden omitirse sin graves consecuencias.

Es el caso de las imágenes, las cuales admiten diferentes procedimientos de compresión. Una imagen 2D se puede representar con un número de bits igual a largo  $\times$  ancho  $\times$  profundidad, donde largo y ancho son los pixeles y profundidad representa otros atributos como colores y tonos de gris.

Imágenes de baja resolución como las de CIFAR-10 requieren  $32 \times 32 \times 3 = 3072$  bits. Una imagen de resolución moderada  $200 \times 200 \times 3 = 120000$  bits. Cuando se tengan muchas de estas últimas ciertamente habrá dificultades computacionales.

La “correlación” o correlación cruzada de dos funciones de una variable discreta y de dos variables discretas, se define por

$$y(n) = (x * h)(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k).$$

$$(x * h)(m, n) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x(j, k)h(m-j, n-k).$$

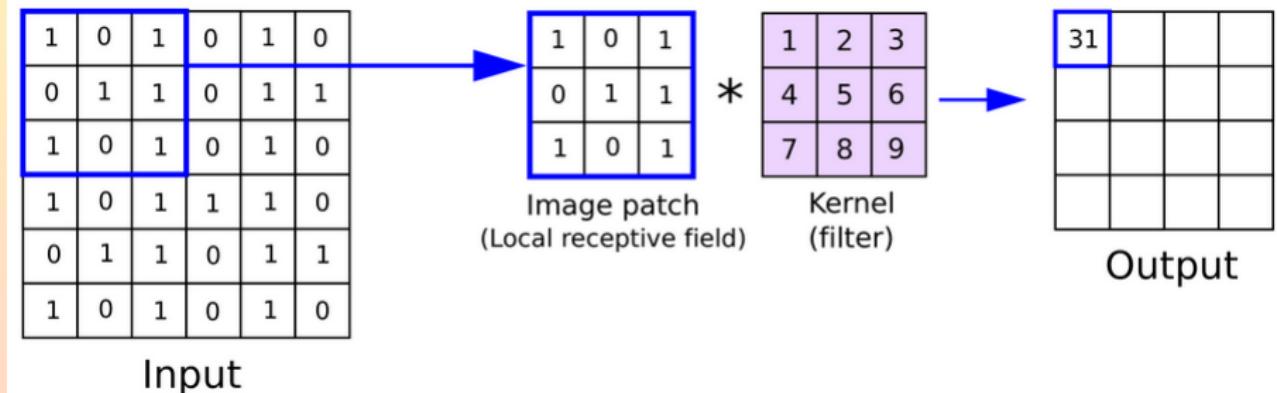
Es una suma de productos de valores de dos funciones punto a punto, sometidas a un desajuste.

<http://cs231n.stanford.edu/> es un curso sobre CNN.

<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets> contiene una explicación intuitiva acerca de las redes neuronales convolucionales. También es interesante

<http://cs231n.github.io/convolutional-networks/>

# REDES NEURONALES CONVOLUCIONALES



La técnica de CNN opera sobre los mencionados “volúmenes”. Utiliza un “filtro” de dimensión menor que se mueve a lo largo de toda la imagen original. En el camino se realiza el producto entre la imagen y el filtro, lo cual da lugar a una serie de trozos más pequeños. Por eso se supone que las CNN son apropiadas para extraer propiedades que no dependan drásticamente de la posición.

Por ejemplo, si la imagen original es  $32 \times 32 \times 3$  y el filtro es  $5 \times 5 \times 3$ , la operación la convierte en  $28 \times 28 \times 1$ . En efecto, hay  $28 \times 28$  posiciones únicas donde el filtro puede colocarse sobre la imagen original. Otro ejemplo: con una imagen  $4 \times 4 \times 3$  y un filtro  $2 \times 2 \times 3$ , el cual se puede deslizar de  $3 \times 3 = 9$  maneras distintas sobre la imagen, da lugar a  $3 \times 3 \times 1$ . La palabra convolución representa la acción de deslizar el filtro y sumar los productos de las respectivas unidades en coincidencia.

# ENTRENAR UN CLASIFICADOR

[https://pytorch.org/tutorials/beginner/blitz/cifar10\\_tutorial.html](https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html)

Allí se considera el problema de reconocimiento de imágenes mediante una red neuronal con muchas capas ocultas.

El problema empieza con el entrenamiento de un clasificador. Requiere: Disponer de los datos de entrenamiento, definir la red neuronal, calcular funciones de pérdida y actualizar los pesos de la red.

Considera el conjunto de imágenes dados en CIFAR10. Allí se tienen las clases 'avión', 'automóvil', 'pájaro', 'gato', 'cabra', 'perro', 'rana', 'caballo', 'barco', 'camión'.

Del enlace

[https://pytorch.org/tutorials/beginner/blitz/cifar10\\_tutorial.html](https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html)  
se extrae el notebook

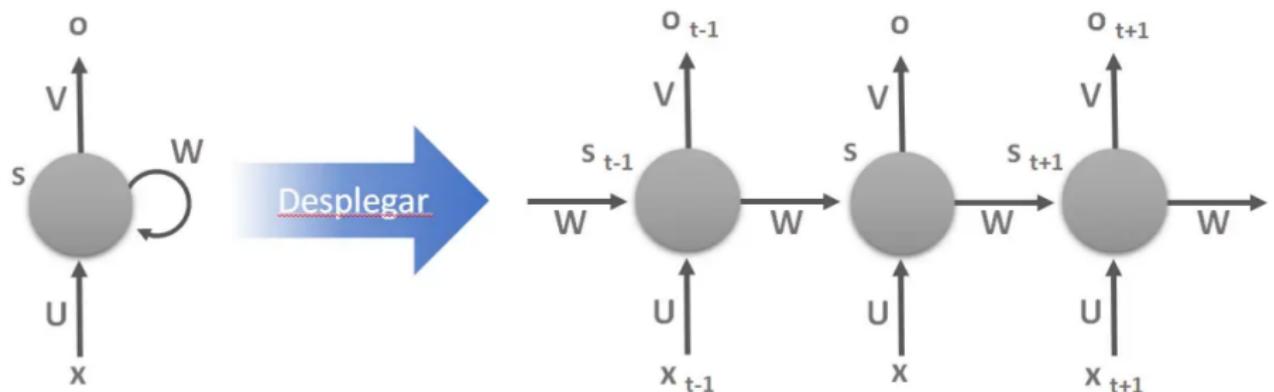
EnsayoPytorchVision.ipynb

# REDES NEURONALES RECURRENTES

Las redes neuronales recurrentes (RNN) son un tipo de RN que se usan para procesar datos que guarden una estructura secuencial, como series de tiempo y los textos. Permiten llevar un registro de las relaciones estructurales entre los datos, por ejemplo en una serie de tiempo capturar información del pasado y usarla para definir la salida actual.

Se usan en procesamiento del lenguaje natural (NLP), reconocimiento de voz, predicción de series temporales incluyendo la captura de patrones y dependencias a lo largo del tiempo, generación de música y artes plásticas, análisis de sentimientos o juicios de valor.

Las RNN son adecuadas para problemas con datos secuenciales y dependencias a largo plazo, mientras que las CNN son más apropiadas para problemas que involucran datos con estructura espacial, como imágenes y videos. Las CNN sirven para captar patrones globales repetitivos, en tanto que las RNN se enfocan en las especificidades locales de los datos. Se pueden usar como una “memoria interna” o “estado oculto”.



Red Neuronal Recurrente

Imagen de Google

# REDES NEURONALES CON ATENCIÓN

Las redes neuronales con atención, los encoders-decoders y los transformadores son tipos de redes neuronales recurrentes que se utilizan para el procesamiento del lenguaje natural (NLP).

La **atención** permite a las redes neuronales recurrentes centrarse en las partes más relevantes de una secuencia de datos. Se usa en traducción automática, respuesta a preguntas y otras tareas de NLP.

Las redes neuronales con atención son redes neuronales recurrentes que utilizan una capa de atención para asignar diferentes pesos a diferentes partes de una secuencia de datos. Una capa con atención es una capa adicional que se coloca en el medio de una red neuronal recurrente. Esta capa toma como entrada la salida de la capa anterior, que es una representación vectorial de la secuencia de datos. A continuación, la capa con atención calcula un vector de atención, que asigna diferentes pesos a diferentes partes de la secuencia. De esta manera le asigna un peso o importancia a la salida de la capa anterior. Las partes de la secuencia que tienen un peso alto se consideran más relevantes y, por lo tanto, tienen un mayor impacto en la salida de la red neuronal.

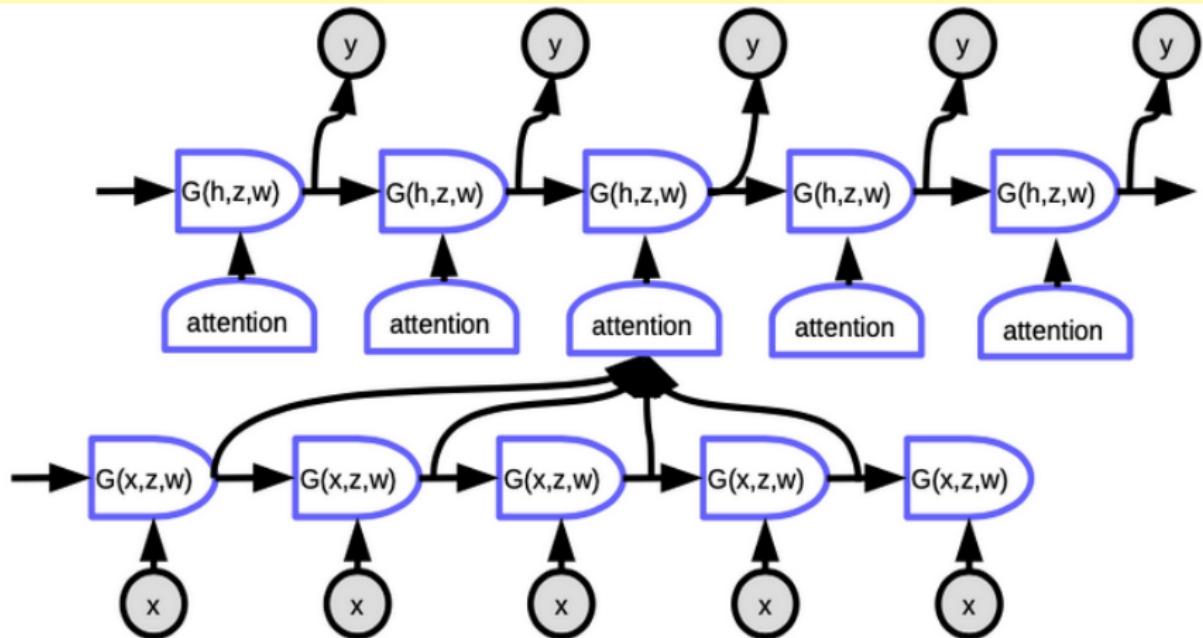


Imagen de Google

Con la atención, en lugar de procesar toda la secuencia de entrada de manera uniforme, la red aprende a enfocar y dar más peso a las partes relevantes o informativas de los datos.

Hay diferentes maneras de implementar una capa con atención. Una forma común es utilizar una función de atención que calcula el producto escalar entre el vector de estado de la capa anterior y un vector de consulta dado. El vector de consulta se puede inicializar con un vector de valores constantes, o se puede aprender durante el entrenamiento de la red neuronal.

Dicho vector de consulta puede representar un **prompt**. Un prompt es una cadena de texto que se utiliza para proporcionar información adicional a una red neuronal. Por ejemplo, un prompt se podría utilizar para proporcionar el contexto de una oración o para indicar el tipo de respuesta que se espera.

Otra forma de implementar una capa con atención es utilizar una función de atención que calcula la probabilidad de que cada palabra de la secuencia sea importante. La probabilidad de cada palabra se puede calcular utilizando una función de activación, como la función sigmoide.

# CODIFICADORES Y DECODIFICADORES. TRANSFORMERS

Los **encoders-decoders** son una arquitectura de red neuronal recurrente en la cual el encoder convierte una secuencia de entrada en una representación vectorial, y el decoder utiliza esta representación para generar una secuencia de salida. Esta arquitectura es especialmente adecuada para tareas de NLP que implican la traducción automática, la generación de texto y la respuesta a preguntas.

Los **transformadores** son un tipo de red neuronal recurrente que utiliza una capa de atención para mapear una secuencia de entrada a una secuencia de salida. Los transformadores se han demostrado que son muy eficaces en una variedad de tareas de NLP, incluidas la traducción automática, la generación de texto y la respuesta a preguntas.

Las redes neuronales con atención son especialmente eficaces en tareas de NLP que requieren comprender el orden de las palabras en una oración. Se utilizan a menudo para la traducción automática, la respuesta a preguntas y la generación de texto creativo.

La principal ventaja de las redes neuronales con atención es su capacidad para manejar secuencias de longitud variable y seleccionar automáticamente las partes más relevantes de la secuencia. Esto es particularmente útil en tareas donde se necesita una comprensión contextual de la secuencia, como el procesamiento del lenguaje natural (NLP). Al enfocarse en partes específicas, la red puede capturar dependencias a largo plazo y extraer características relevantes, lo que conduce a un mejor rendimiento en diversas aplicaciones.

Algunas aplicaciones de las redes neuronales con atención incluyen traducción automática, resumen de texto, preguntas y respuestas, reconocimiento de voz, análisis de sentimientos, patrones inusuales en los datos, generación de música, etc.

La fortaleza principal de los transformers radica en su mecanismo de atención, que permite a la red enfocarse en diferentes partes de la secuencia de entrada mientras procesa la información.

## TERMINOLOGÍA

En los transformers, los “codificadores” y “decodificadores” se refieren a los bloques de capas que realizan operaciones de atención y feed-forward. Estos bloques se repiten en la arquitectura para procesar y transformar las secuencias de datos.

El “**one-hot encoding**” es una técnica utilizada para representar datos categorizados o categóricos en forma de vectores binarios. Cada categoría se representa como un vector binario con una dimensión igual al número total de categorías. Todas las dimensiones del vector son cero, excepto la correspondiente a la categoría que se está representando, que se establece en uno.

En los transformers, las secuencias de entrada no se representan utilizando “one-hot encoding”. En su lugar, se utilizan representaciones vectoriales densas que contienen información contextual y semántica sobre las palabras o elementos de la secuencia. Estas representaciones vectoriales se aprenden mediante el proceso de entrenamiento de la red neural, en el que se ajustan los pesos y se optimizan para capturar características y relaciones relevantes en los datos.

Los transformers tienen la propiedad de atención global, con la cual pueden acceder a todas las posiciones de la secuencia de entrada. Así capturan relaciones a largo plazo y comprenden el contexto global de la secuencia. Con la atención, el modelo puede asignar diferentes pesos las partes relevantes de la entrada, lo que mejora su capacidad para procesar información contextual.

Los transformers pueden procesar todas las posiciones de la secuencia en paralelo, lo cual los hace útiles cuando se usa hardware con GPUs o TPUs. Los transformers son escalables, así pueden manejar secuencias de longitud variable. En NLP se tienen secuencias de diferentes longitudes, como la traducción automática o el resumen de texto.

Los transformers preentrenados en grandes cantidades de datos pueden usarse como modelos de transferencia de aprendizaje para tareas específicas. Así, se cuenta con buena aproximación inicial a los pesos de la red neuronal.

<https://gpt4all.io/index.html>

<https://github.com/amaiya/ktrain>

Enlace a los notebooks mencionados en la charla:

<https://github.com/JorgeEMahecha/Notebooks-Intro-ML>