Deep Learning

Fundamentos de la IA

DuocUC® | ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

El perceptrón

El núcleo de la IA

DuocUC®

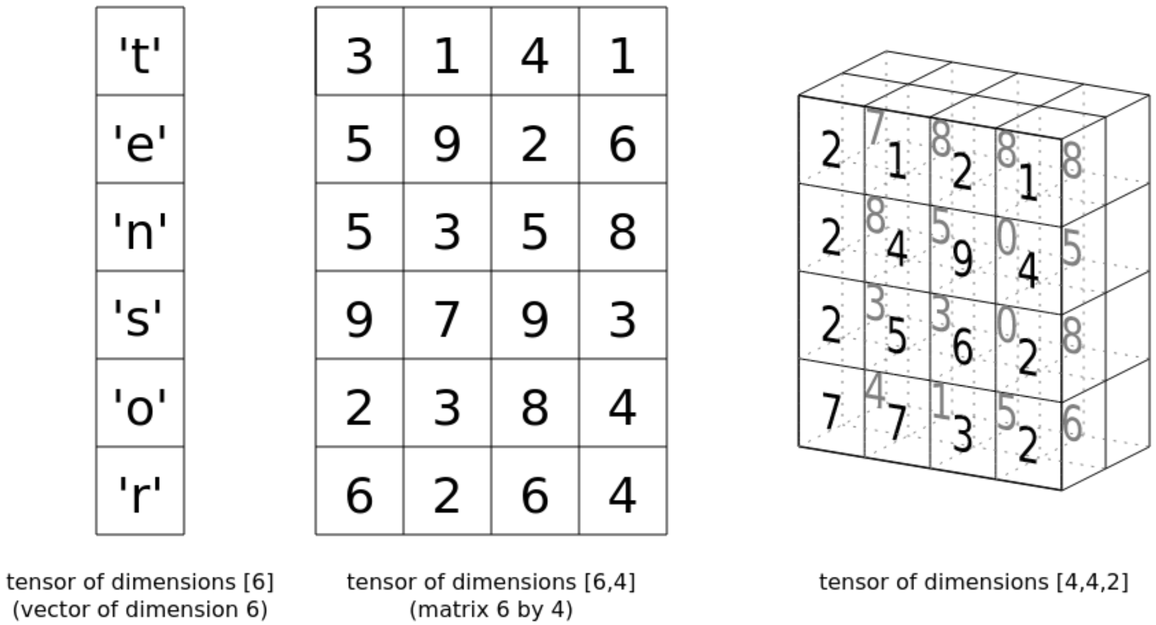
Índice

Tensores la base de los cálculos en redes neuronales

Resulta que los cálculos de la red neuronal son solo un montón de operaciones de álgebra lineal de tensores (una generalización de las matrices).

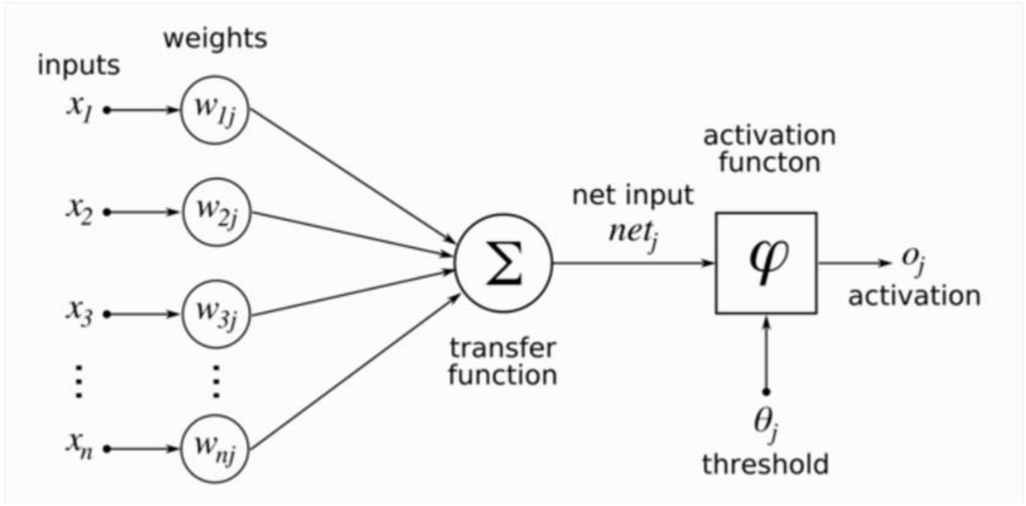
Un vector es un tensor unidimensional, una matriz es un tensor bidimensional, una matriz con tres índices es un tensor tridimensional (imágenes de color RGB, por ejemplo).

La estructura de datos fundamental para las redes neuronales son los tensores y PyTorch (así como casi cualquier otro framework de aprendizaje profundo) **se construye alrededor de los tensores**.

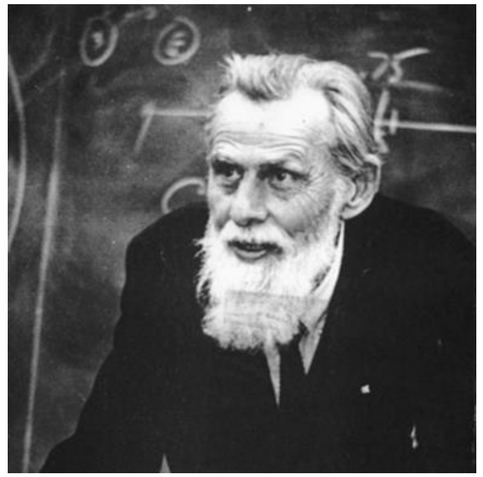


Historia y evolución de las redes neuronales

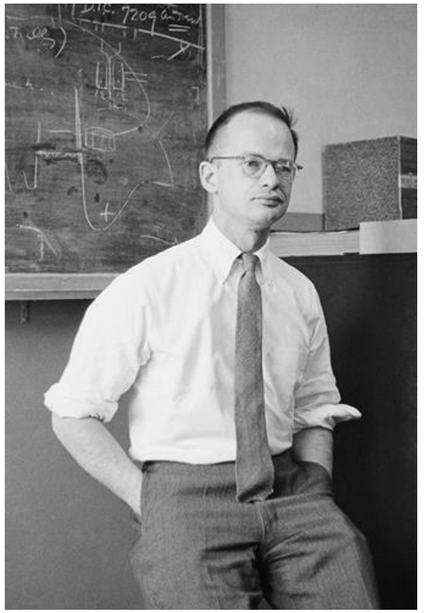
La idea de las redes neuronales comenzó como un modelo de cómo funcionan las neuronas en el cerebro, denominado "*conexionismo*" y utilizaba circuitos conectados para simular el comportamiento inteligente. En 1943, retratado con un circuito eléctrico simple por el neurofisiólogo Warren McCulloch y el matemático Walter Pitts.



Modelo de una neurona artificial de acuerdo con McCulloch and Pitts.



Warren McCulloch



Walter Pitts

Historia y evolución de las redes neuronales

Estos conceptos se trataron de llevar a la práctica en la década de 1950, cuando los investigadores comenzaron a intentar traducir estas redes en sistemas computacionales.

Durante esa época, Frank Rosenblatt estaba tratando entender el sistemas de decisión presentes en el ojo de una mosca y que determinan su respuesta de huida. Para comprender y cuantificar este proceso, propuso la idea de un **perceptrón** en 1958.

El perceptrón era un sistema con una relación entrada-salida simple, modelado en una neurona McCulloch-Pitts.

La idea es que a partir de varias entradas, toma una suma ponderada y devuelve "0" si el resultado está por debajo del umbral y "1" en caso contrario.

Los pesos se van “aprendiendo” al comparar el resultado obtenido con el deseado.

Frank Rosenblatt

El Perceptrón [Ojo, no es familiar de Megatrón]

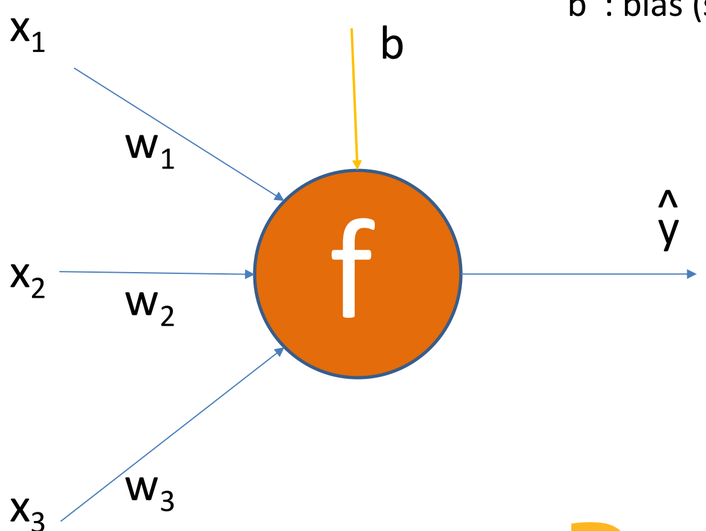
- Definido el año 1957 (F. Rosenblatt)
- Es una representación o modelo matemático y está inspirado en cómo funciona la neurona de los seres humanos:
  - En términos generales que una neurona recibe ciertos impulsos químicos y eléctricos externos (información), la procesa y luego la transmite a otra

x_i : entradas (input)

w_i : pesos

b : bias (sesgo)

Rosenblatt, diseñó un modelo matemático que permitiera recibir cierta información, procesarla aplicando una fórmula matemática y una función para luego transmitir el valor obtenido en este cálculo a la neurona o capa de neuronas siguiente



En otras palabras, y como resumen, el perceptrón o una neurona artificial es una función que recibe valores (x, w, b) y entrega un resultado (ŷ)

Los valores “w” y “b” reciben el nombre de parámetros

El Perceptrón

x_i : entradas (input)

w_i : pesos

b : bias (sesgo)

$\forall$ b, $w_i \in \mathbb{R}$

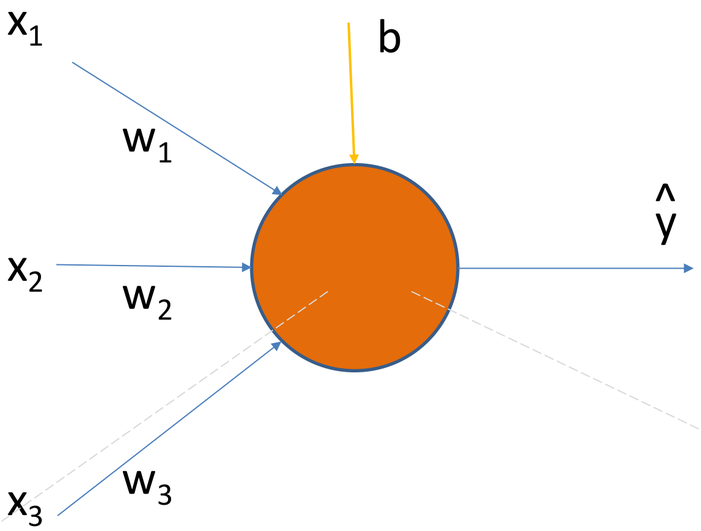
$0 <= x_i <= 1$

La operación que realiza esta función, es la suma ponderada de los datos ingresados (fórmula en el cuadro café)

A este resultado, que representamos con la letra “u”, se le aplica una función, llamada función de activación (que revisaremos en breve)

Los Pesos expresan la importancia de una entrada dada para generar una salida

Suma de las entradas por sus pesos + el sesgo (bias)



$$u = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$\hat{y} = f(u)$$

Función de activación

Función de Activación

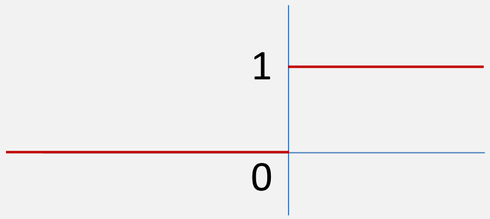
La función de activación originalmente intentaba emular lo que ocurre con las neuronas, es decir, llegado cierto nivel de acumulación de estimulación, la neurona lanza la información a la neurona siguiente. Entonces, si el valor de “u” es suficiente para “estimular” (dar un valor mayor a cero) la función de activación, entonces el dato pasa a la siguiente neurona

b : bias (sesgo)

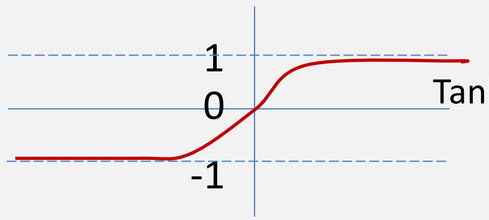
$\forall\ b, w_i \in \mathbb{R}$

$0 <= x_i <= 1$

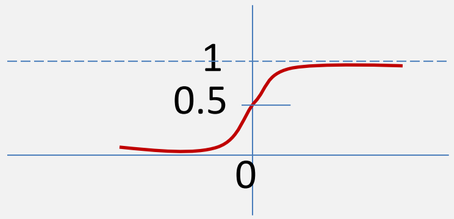
Función Escalón



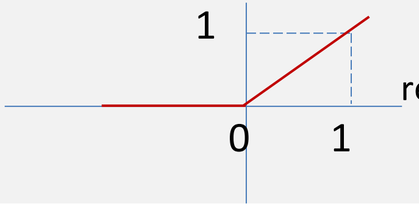
Función Tanh



Función Sigmoid o logística



Función ReLu

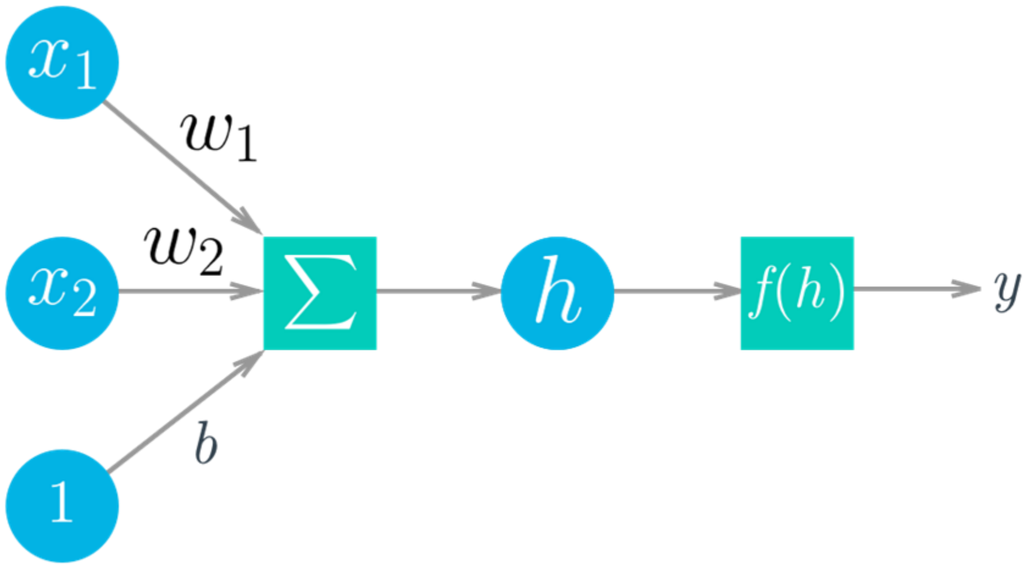


ReLu

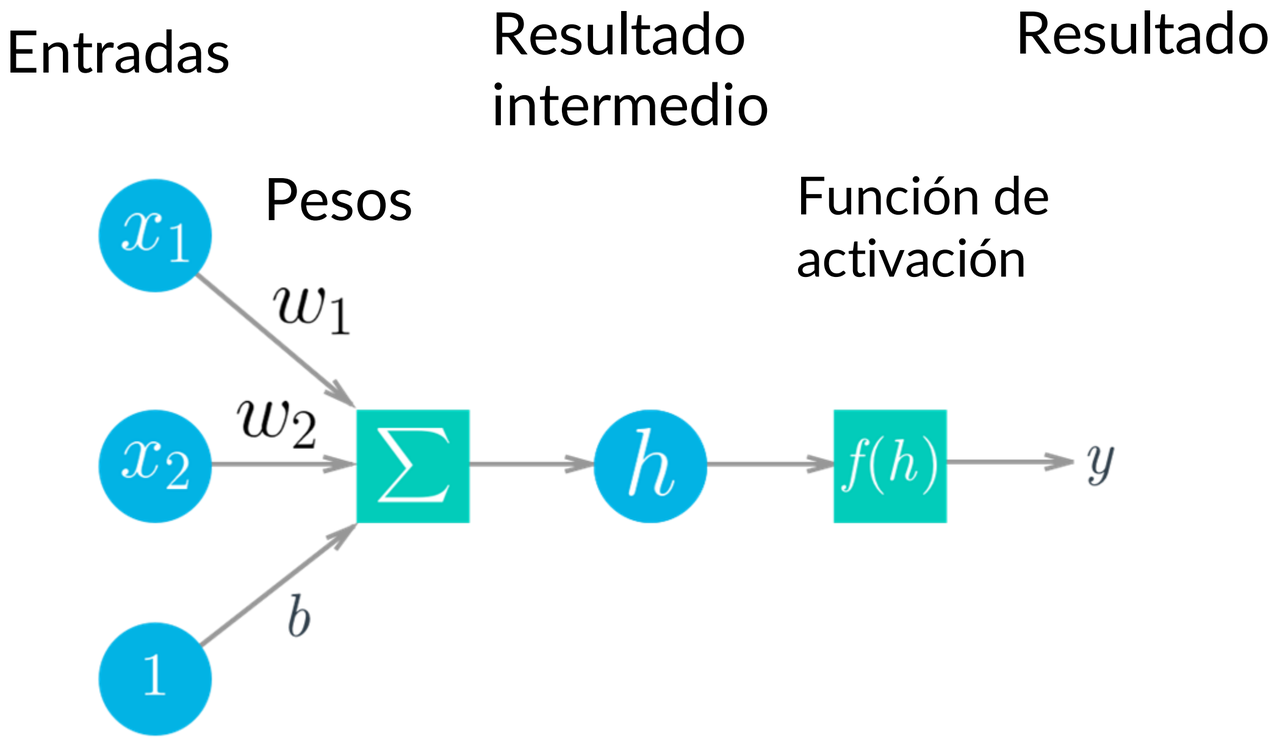
$$f(x) = \begin{cases} 0, if\ x < 0 \\ x, if\ x \geq 0 \end{cases}$$

DuocUC

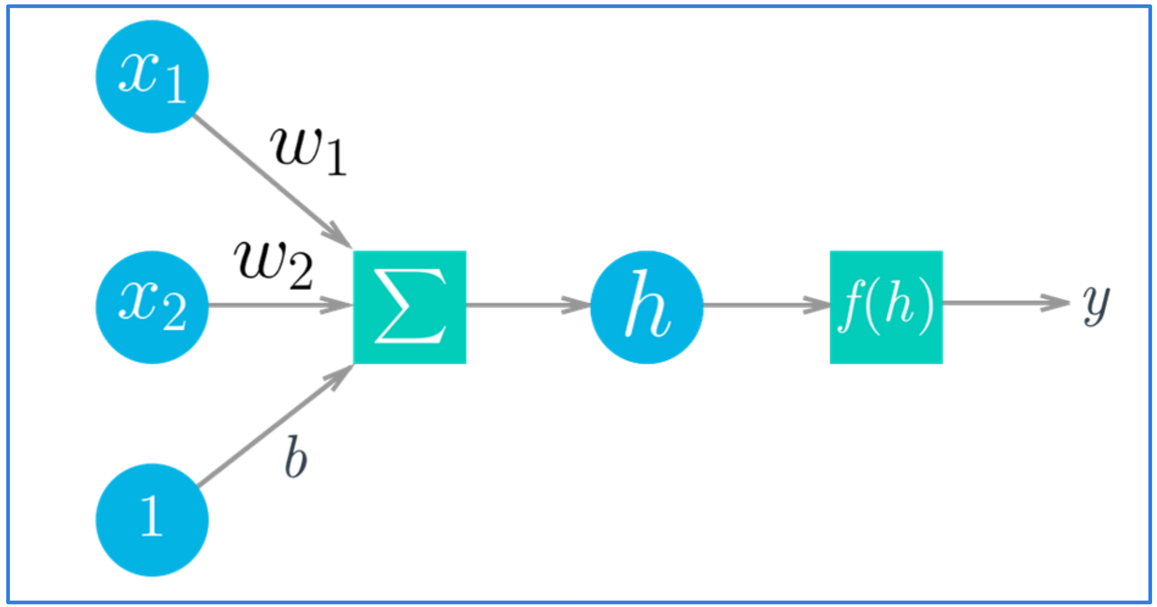
Ejercicio: Funcionamiento de un p**erceptrón**



Funcionamiento de un p**erceptrón**



Funcionamiento de un **perceptrón**

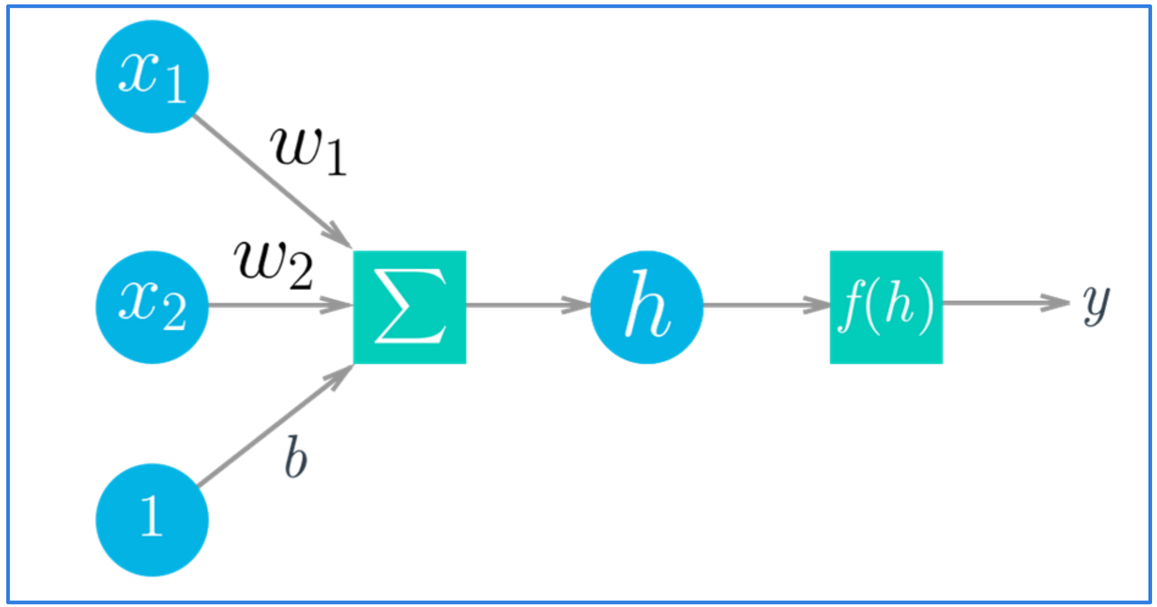


$$y = f(w_1 x_1 + w_2 x_2 + b)$$

TAREA:

Defina en Python una función que replique el funcionamiento de un perceptrón.

Funcionamiento de una **perceptrón**



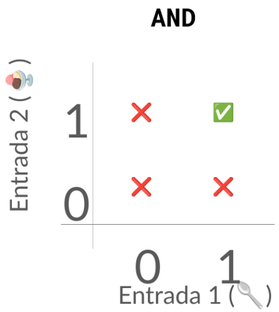
$$y = f(w_1 x_1 + w_2 x_2 + b)$$

$$h = \begin{bmatrix} x_1 & x_2 & \cdots & x_n & 1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}$$

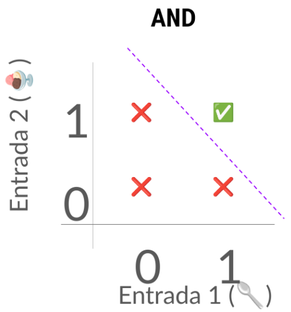
TAREA:

Modifique su código para que realice una operación con tensores.

Ejemplo perceptrón: el postre perfecto



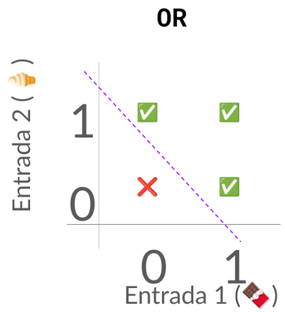
Ejemplo perceptrón: el postre perfecto



Ejemplo perceptrón: el postre perfecto

Ejemplo perceptrón: el postre perfecto

Ejemplo perceptrón: el postre perfecto



Ejemplo perceptrón: el postre perfecto

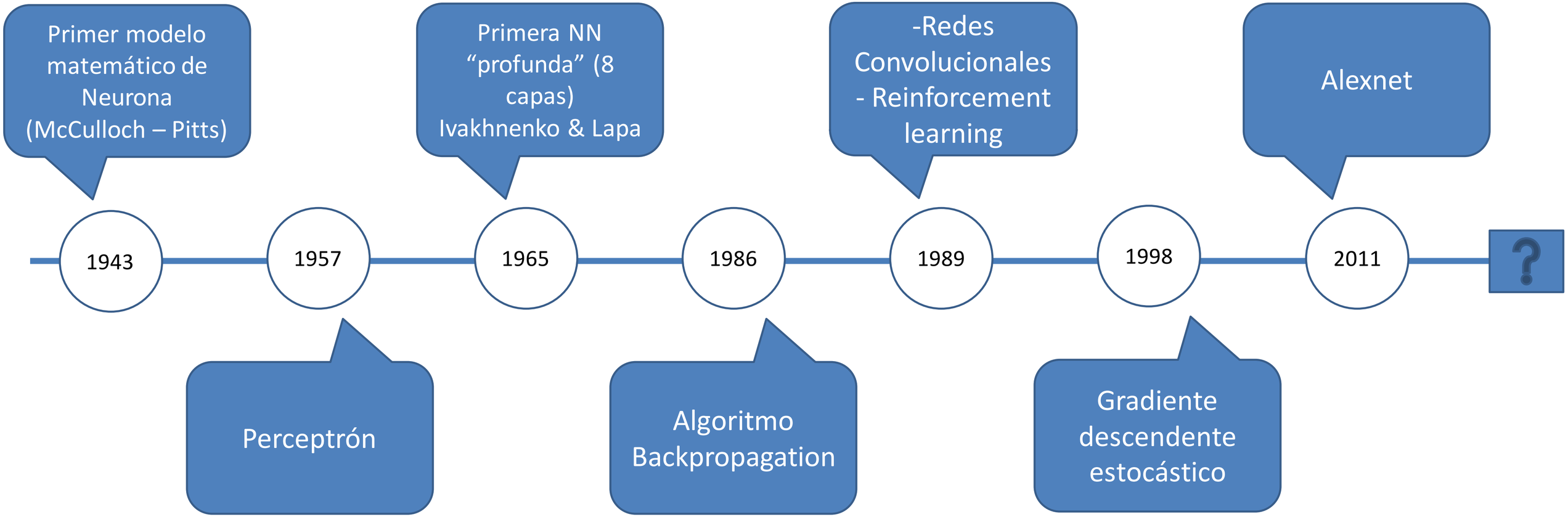
Ejemplo perceptrón: el postre perfecto

Ejemplo perceptrón: el postre perfecto

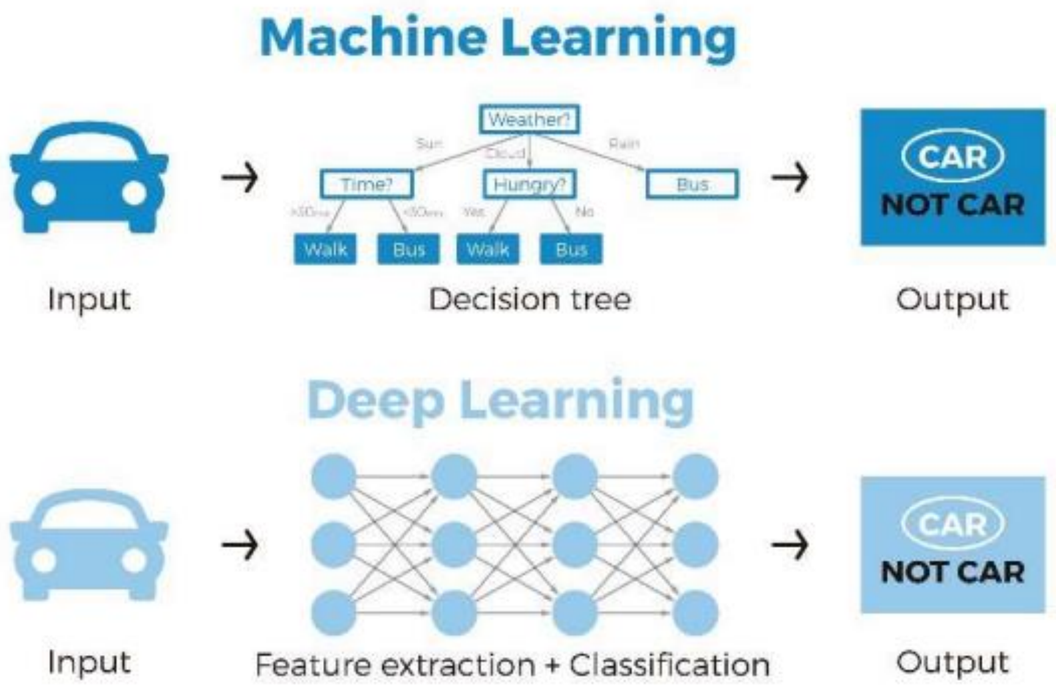
Ejemplo perceptrón: el postre perfecto

Ejemplo perceptrón: el postre perfecto

Timeline. Algunos de los grandes hitos en el campo de la IA



¿El Deep Learning es MachineLearning?

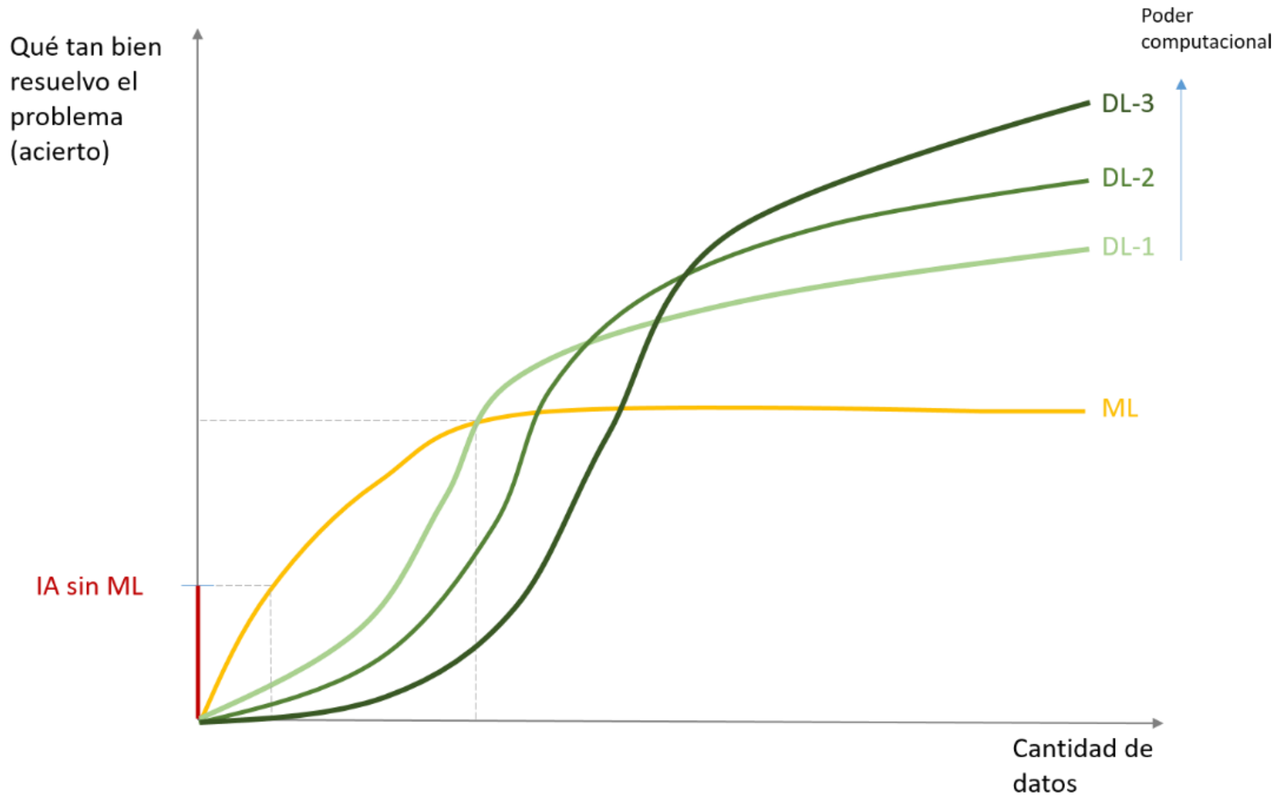


Área de machine learning que estudia modelos con muchas capas de abstracción para tomar decisiones.

En la práctica, los modelos son redes neuronales, una forma de modelo en capas compuesto de unidades de procesamiento que imitan las funcionalidades neuronales.

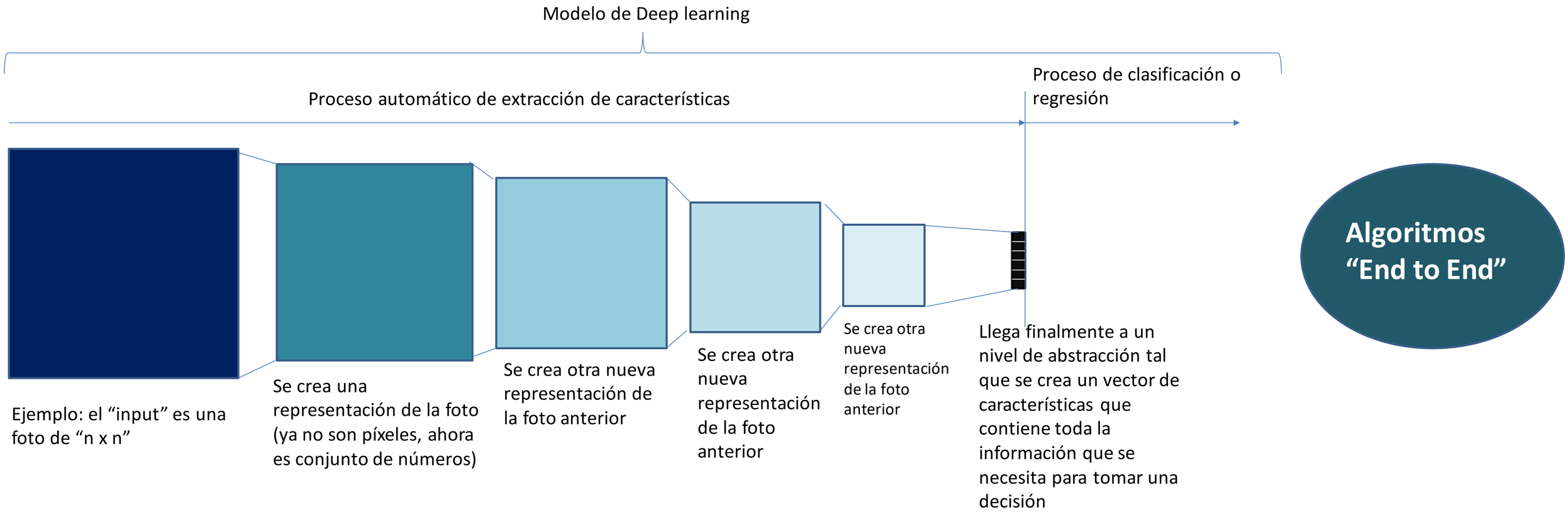
¿Por qué las técnicas de DL dan tan buenos resultados?

- Hoy: mejora en los algoritmos
- Hoy: cantidad de datos disponibles
- Hoy: capacidad de cómputo



Concepto: Abstracción

En el Deep learning se trabaja en base a niveles de abstracción: en cada uno de ellos se van construyendo las representaciones, **extrayendo las características** del nivel anterior, mejorando el nivel de representación y usando menos información que el anterior, pero cada vez **con una mayor abstracción**, una mejor representación y mejor capacidad de generalizar



En deep learning, este tipo de modelos se llaman algoritmos “End to End”, porque se entrena el proceso de extracción de características junto a un modelo de clasificación (o regresión)
En otras palabras y siguiendo con el ejemplo de la foto… a un algoritmo “end to end” le paso una imagen… y el algoritmo me dice si es un gato u otra cosa

https://arxiv.org/pdf/1206.5538)
https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4

DuocUC®

MultiLayer Perceptrón (MLP)

La red neuronal multi perceptrón, nos permitirán resolver problemas más complejos

Cuenta con una capa de entrada, que realmente no es una capa como tal, se le llama así, pero que representa los datos de entrada.

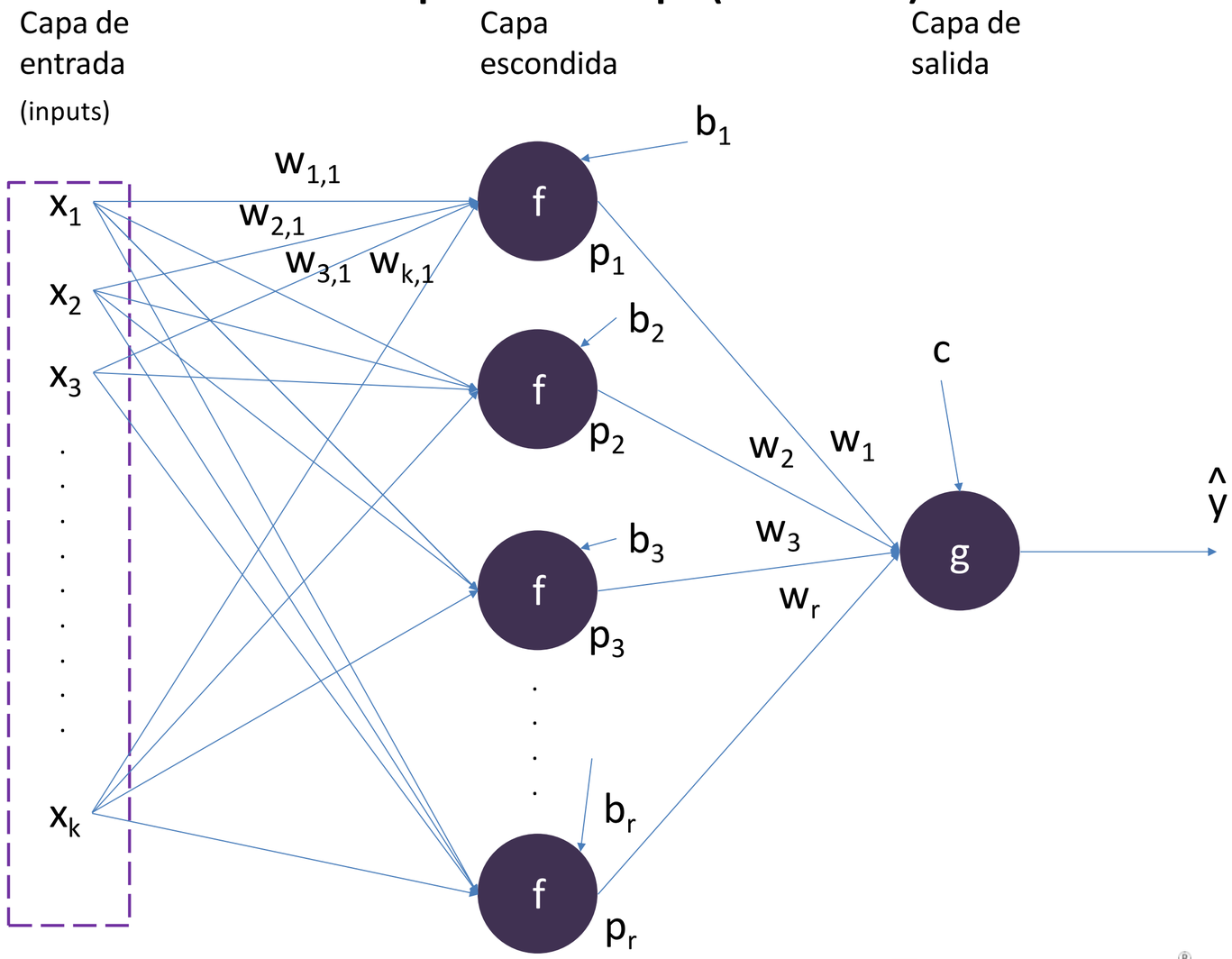
La segunda componente es o son las capas escondidas, en este ejemplo estamos usando 1 sola capa.
(1 o más capas escondidas = Red profunda)

¿Cuántas capas se deben usar?
¿Cuántos perceptrones por capa se deben usar?

Finalmente, la capa de salida, puede ser un vector o un escalar y representa una probabilidad cuando se está realizando una clasificación o un escalar cuando es una regresión (predicción)

~~Nótese que verán en la práctica que en la salida también se aplica una función, generalmente es softmax cuando se está realizando clasificación. Tema que también veremos más adelante en detalle.~~

Estructura de un Perceptrón de 1 capa (escondida)



MultiLayer Perceptrón (MLP)

Definiciones y conceptos:

Todas las entradas (x) , están conectadas con todas neuronas (p) (perceptrones) de la primera capa escondida

Todas las entradas reciben un peso (w), como veremos más adelante estos pesos son inicializados al azar siguiendo ciertas metodologías (x1 con w1, x2 con w2, x_i con w_i)

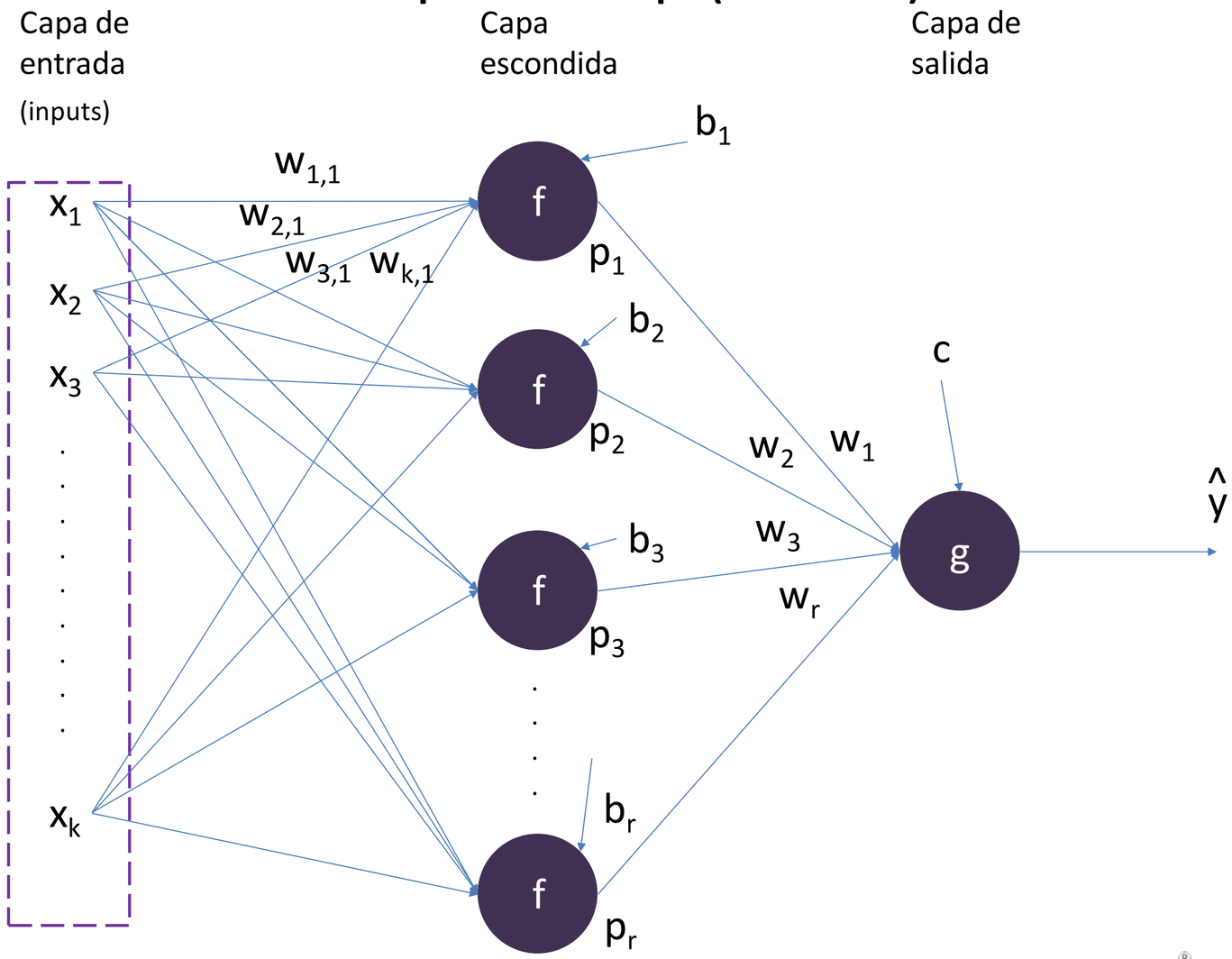
Cada perceptrón (p) tiene también su propio bias (b), que también son inicializados al azar

Todos los perceptrones de la misma capa escondida usan la misma función de activación. Como lo dijimos anteriormente, en la práctica, esta función es ReLu (la más usada)

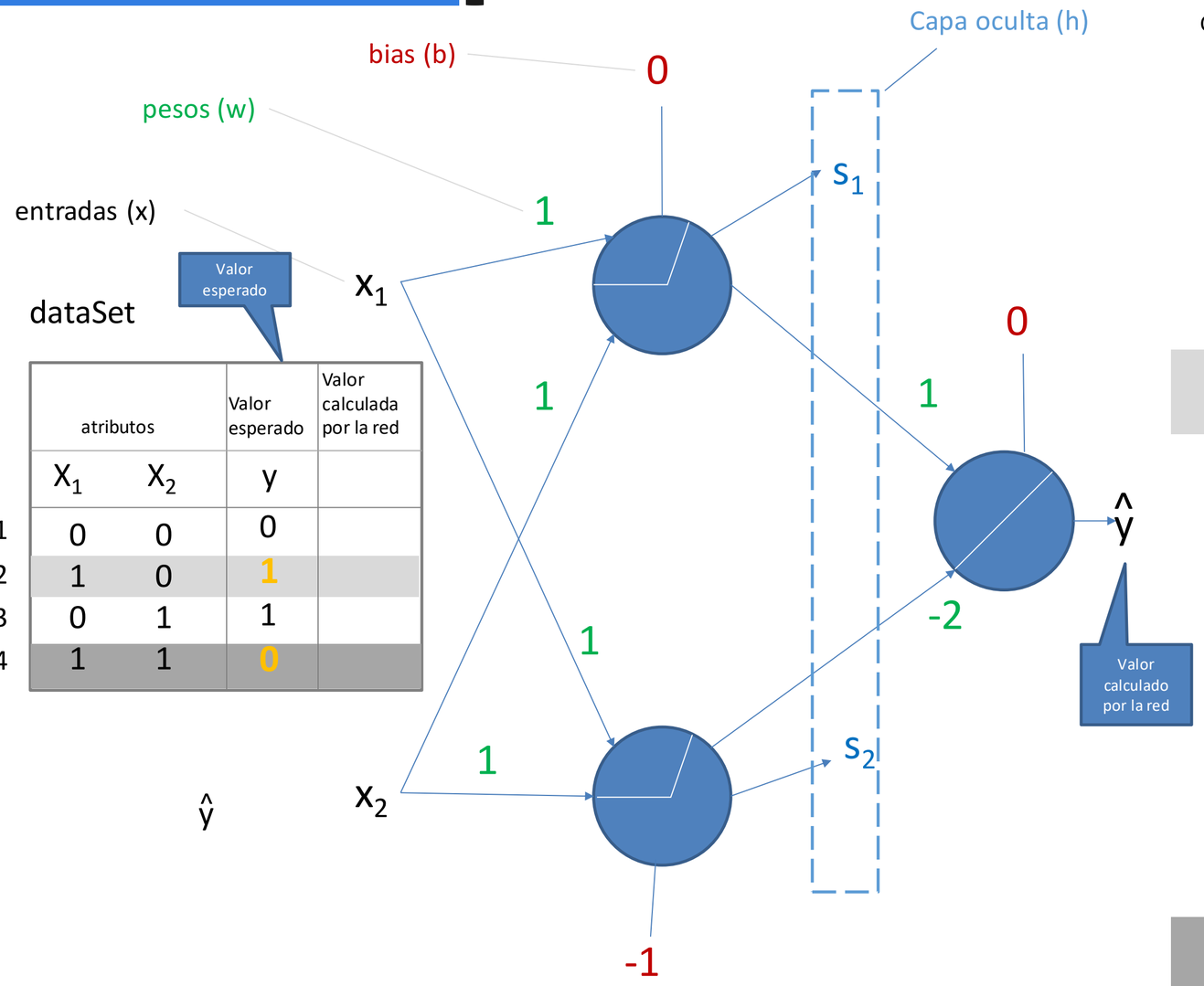
La última neurona, utiliza una función de salida “g” (no confundir con el concepto de función de activación) y también recibe su propio bias “c” generando un resultado ŷ

P son las salidas de la capa escondida, que a su vez son las entradas de la capa de salida.

Estructura de un Perceptrón de 1 capa (escondida)



MLP - Ejercicio



dataSet

	atributos		Valor esperado	Valor calculada por la red
	X_1	X_2	y	
Ejemplo 1	0	0	0	
Ejemplo 2	1	0	1	
Ejemplo 3	0	1	1	
Ejemplo 4	1	1	0	

Calculando para el ejemplo 2

$X_1 = 1$

$X_2 = 0$

$s_1 = Relu(1*1+0*1+0) = 1$

$s_2 = Relu(1*1+0*1+-1) = 0$

$\hat{y} = (1*1+ 0*-2 + 0) = 1$ **Ok**

Calculando para el ejemplo 4

$X_1 = 1$

$X_2 = 1$

$s_1 = Relu(1*1+1*1+0) = 2$

$s_2 = Relu(1*1+1*1+-1) = 1$

$\hat{y} = (2*1 + 1*-2 + 0) = 0$ **Ok**

- Ejecutaremos el cómputo utilizando solo 2 ejemplos de la tabla de verdad (de color gris)
- Se utiliza ReLu como función de activación
- Se usa una función lineal u (identidad) como función de salida
- s_1 y s_2 representan la salida de las neuronas de la capa escondida “h”
- “$\hat{y}$” representa la salida de la red neuronal de 1 capa que resuelve el XOR, este es el valor que calcula la red y debe ser comparado con el valor esperado. Si coinciden, implica que la red está bien entrenada

https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

DuocUC®

MLP – Generalizaciones matemáticas

$$h = ReLu(X*W + B)$$

Esta es la forma de representar la capa oculta de la red neuronal

generalizando la función ReLu, la fórmula general para la representación matemática de la capa h es

$$h = f(X*W + B)$$

Y generalizando la capa de salida, nos queda…

$$\hat{y} = h*U + c$$

** La mayúscula siempre representa el vector completo de una variable. Ejemplo. X es x_1, x_2, x_3 … x_n

Resumen

Qué es un perceptrón?
Una función matemática que intenta emular el comportamiento de una neurona biológica y que recibe valores numéricos muy chicos (reales), que aplica un cálculo aritmético y genera un resultado (otro real)

Qué es una **función de activación** en Deep learning?
una función matemática que transforma una entrada numérica en otro valor (escalar real) dentro de un cierto de rango de valores (excepto la función identidad)
Su rol es súper importante en la redes neuronales profundas dado que permite “quitarle” la linealidad a la función ponderada de los inputs (entradas)
Esto se traduce en que permite aumentar la capacidad de representación de las neuronas de la red

En la práctica, cuál es la función de activación recomendada hoy en día?
ReLu es la función altamente utilizada hoy en día, provocó una revolución cuando fue implementada (2012 en una competencia de reconocimiento de imágenes). Es una innovación que permitió dar un salto sustancial en el resultado de las redes neuronales artificiales

Desde esa fecha, se comenzó a utilizar Relu. Podrán ver que hoy en día existen algunas funciones basadas en ReLu, por ejemplo softplus

Notas

Repositorio temporal del curso:
https://github.com/jorgeanais/dly0100_deeplearning