Introducción a Redes Neuronales Artificiales

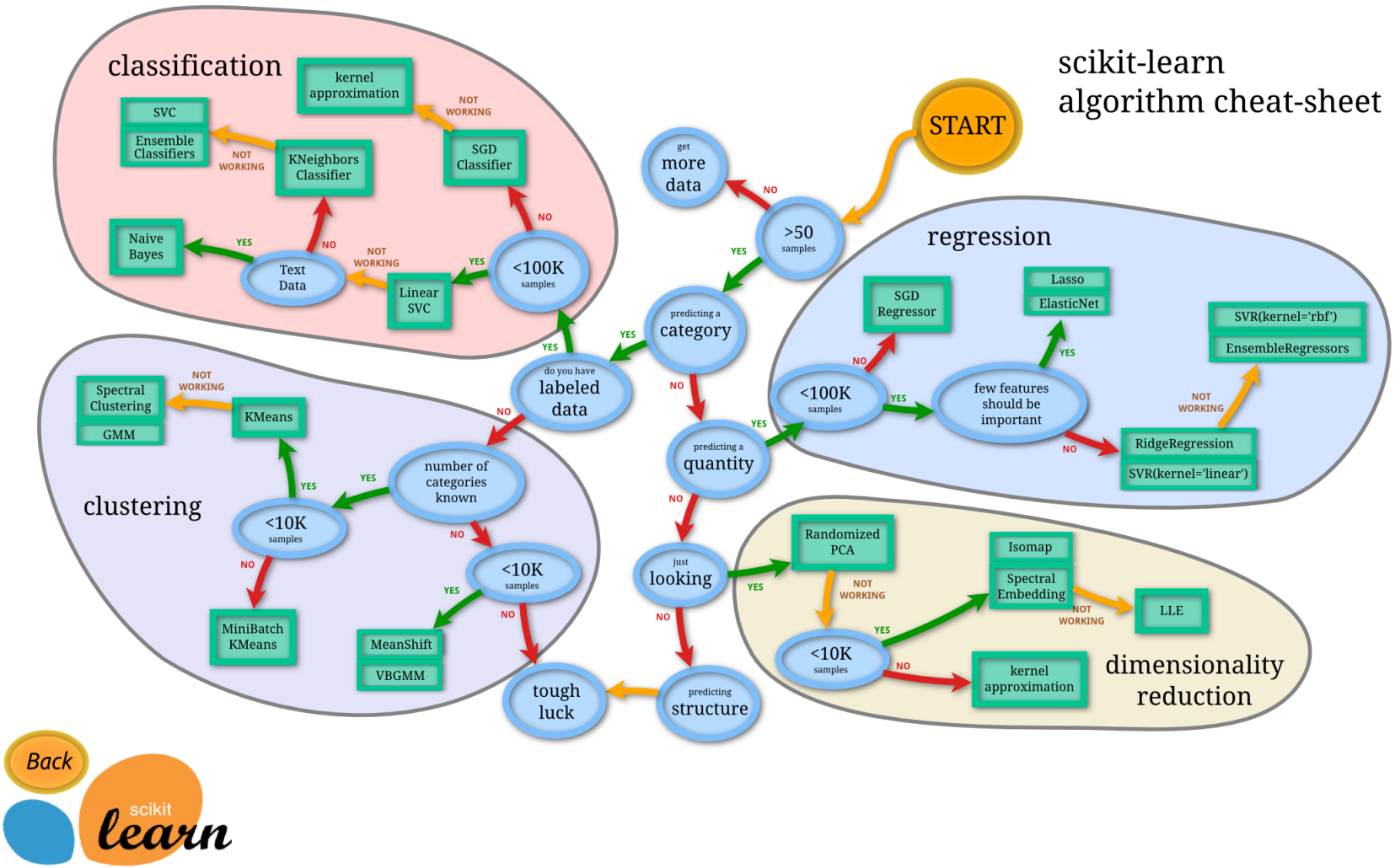
A. M. Alvarez-Meza, Ph.D.
D. F. Collazos-Huertas, Ph.D(c)
amalvarezme@unal.edu.co, dfcollazos@unal.edu.co

Analítica de datos
Departamento de ing. eléctrica, electrónica y computación
Universidad Nacional de Colombia-sede Manizales

Contenido

Contenido

Métodos clásicos de aprendizaje de máquina - Scikit-learn



Artificial neural networks (ANNs): Motivación

Google Images

YouTube

Hey Siri

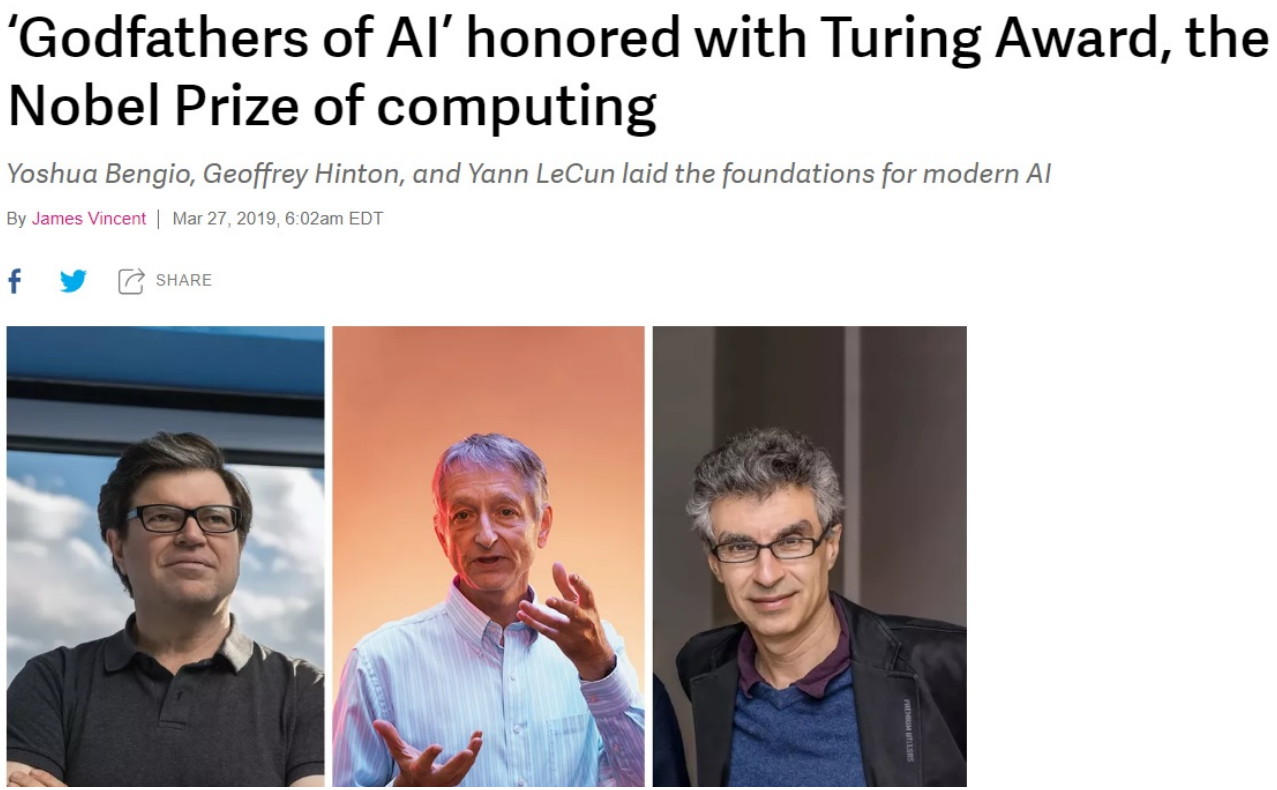
Las ANNs están en el centro del aprendizaje automática actual, son versátiles, potentes y escalables, lo que las hace ideales para abordar tareas sobre grandes cantidades de datos.

Aprendizaje de máquina

- En una frase: *aprendizaje de máquina* es el conjunto de los **algoritmos** y las **técnicas** que se usan para diseñar sistemas que aprendan a partir de los datos.
- Los fundamentos del *aprendizaje de máquina* se basan en las **matemáticas** y la **estadística**.
- De forma general, no tienen en cuenta el conocimiento del dominio y el pre-procesamiento de los datos.
- El aprendizaje de máquina es el eje central de la ciencia de datos y la inteligencia artificial - (IA).
- Primeros avances serios en IA:

  https://www.youtube.com/watch?v=FwFduRA$_L$6Qab$_c$hannel = YannLeCun.

El renacer de la inteligencia artificial (Premio Turing 2019)



'Godfathers of AI' honored with Turing Award, the Nobel Prize of computing

Yoshua Bengio, Geoffrey Hinton, and Yann LeCun laid the foundations for modern AI

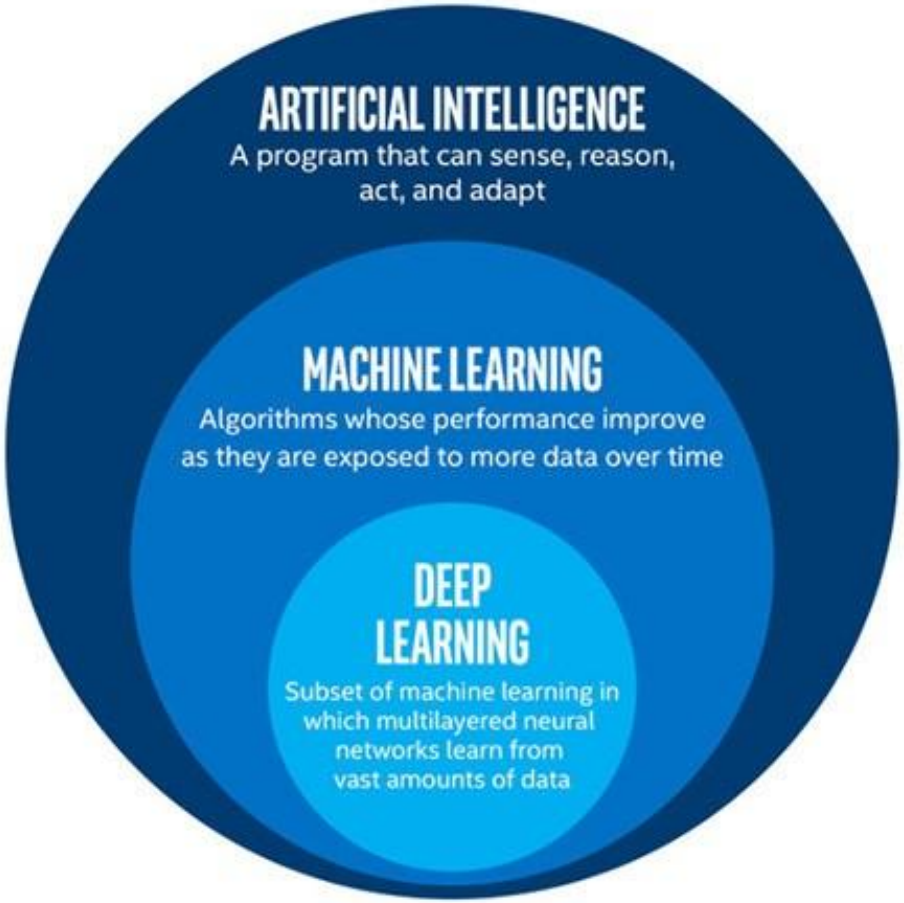
By James Vincent | Mar 27, 2019, 6:02am EDT

SHARE

En 2006, Geoffrey Hinton et al. publicaron un artículo [1] que mostraba como un algoritmo de aprendizaje profundo podía reconocer digitos a mano con una precisión $> 98\,\%$, llamándolo Deep Learning.

[1] ver `http://www.cs.toronto.edu/~hinton/`

Inteligencia artificial, Aprendizaje de máquina y Aprendizaje profundo



El renacer de la inteligencia artificial (Aprendizaje de máquina)

- Entrenar un modelo de deep learning era considerado imposible en los 90s.
- Hinton y los demás investigadores en redes neuronales empezaron a destronar a los algoritmos clásicos de aprendizaje de máquina.
- En la actualidad: aprendizaje de máquina como corazón de muchos productos de tecnología de punta (búsqueda web, teléfonos inteligentes, reconcomiento de habla, autos que se conducen solos, etc...)
- La clave: mucho poder de cómputo y muchos datos.

De las neuronas biológicas a las artificiales

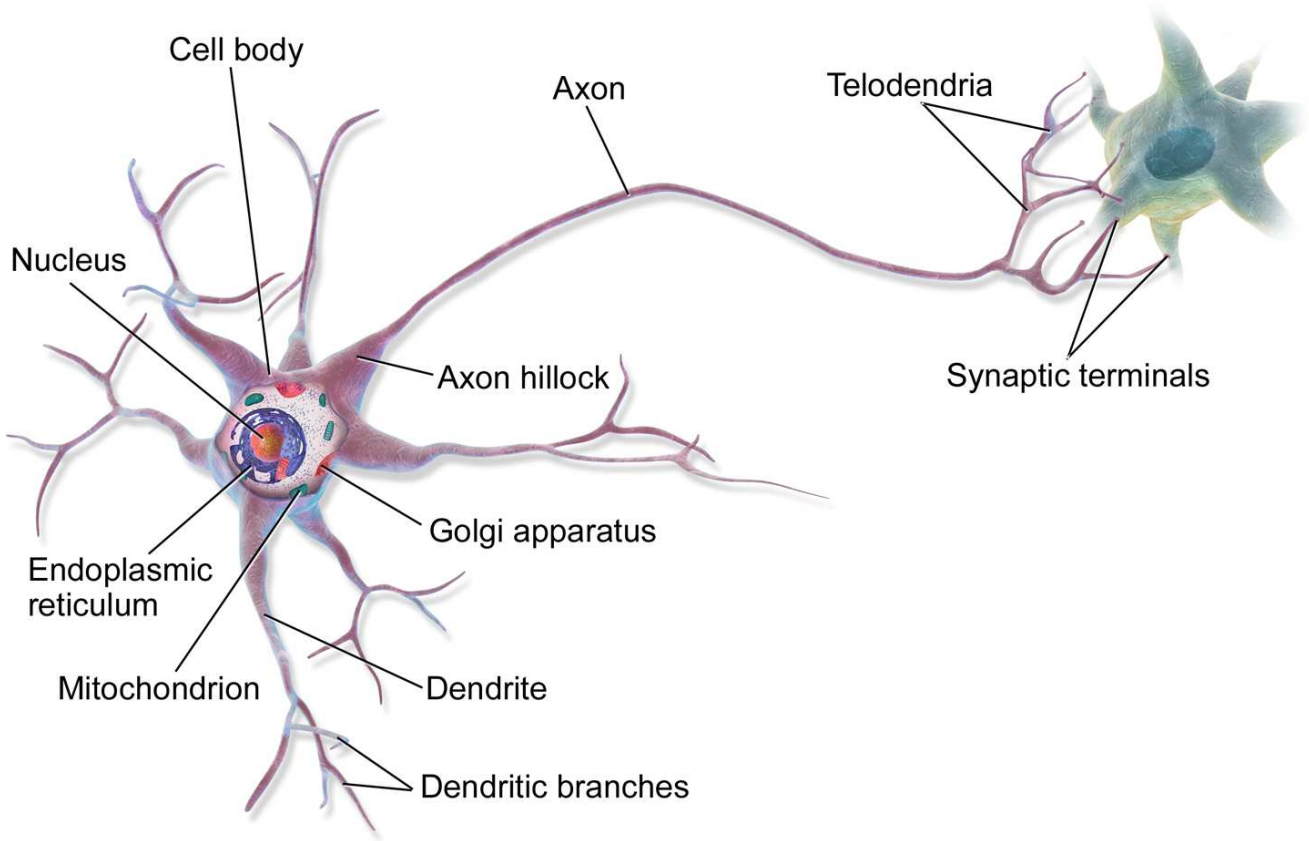
- Introducidos en **1943** por el *Warren McCulloch* y *Walter Pitts*.Modelo computacional de cómo las neuronas biológicas podrían trabajar juntas en los cerebros de los animales.
- En la década **1960** surgieron los primeros éxitos de las ANNs. Sin embargo, los fondos volaron a otra parte y las ANNs entraron en un largo invierno.
- A principios de los **80's** hubo un resurgimiento del interés por el **conexionismo**, se inventaron nuevas arquitecturas y se desarrollaron mejores técnicas de entrenamiento.
- En la década de 1990 se inventaron otras poderosas técnicas de *Machine Learning*, como SVMs, por lo que una vez más el estudio de las redes neuronales entró en un largo invierno.

Por qué esta ola es diferente?

- **Gran cantidad de datos** disponibles.
- El aumento de la potencia informática (Cantidad vs. Tiempo).
- Producción de **tarjetas GPU** potentes.
- Los **algoritmos de entrenamiento** han sido mejorados.
- Algunas limitaciones teóricas de los ANN han resultado ser benignas en la práctica (problema de óptimos locales).
- Las ANNs parecen haber entrado en un círculo virtuoso de financiación y progreso.

Contenido

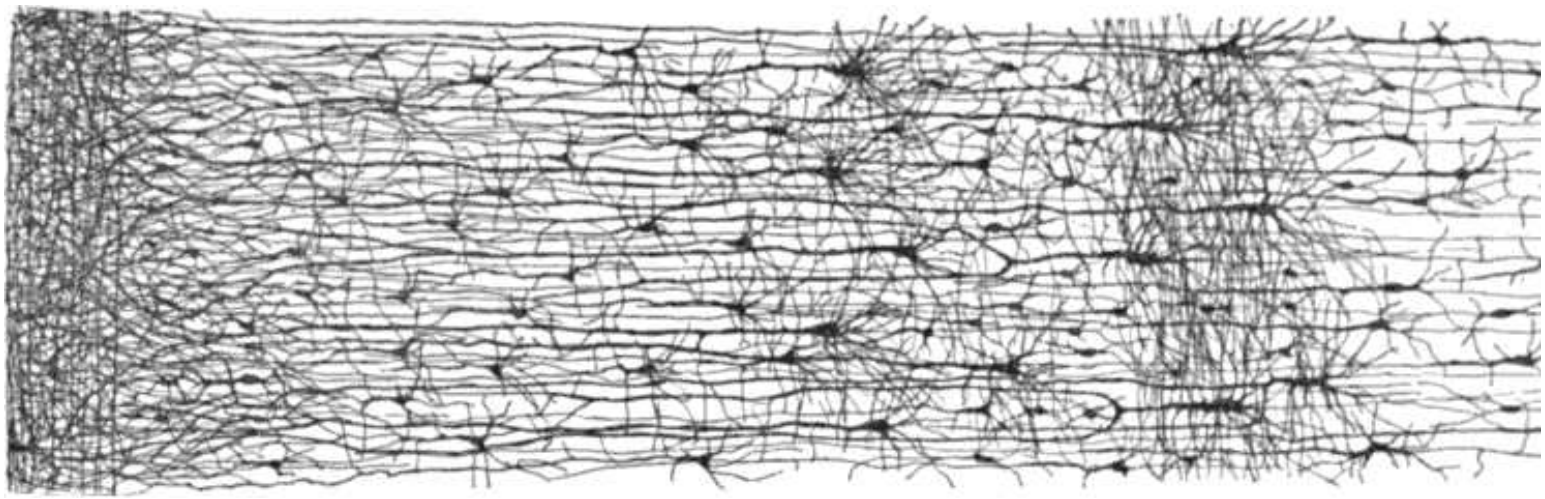
Neurona I



Es una célula que se encuentra principalmente en la corteza cerebral animal, compuesta por: el **núcleo** y muchas extensiones ramificadas llamadas **dendritas**, más una extensión muy larga llamada **axón**.

Neurona II

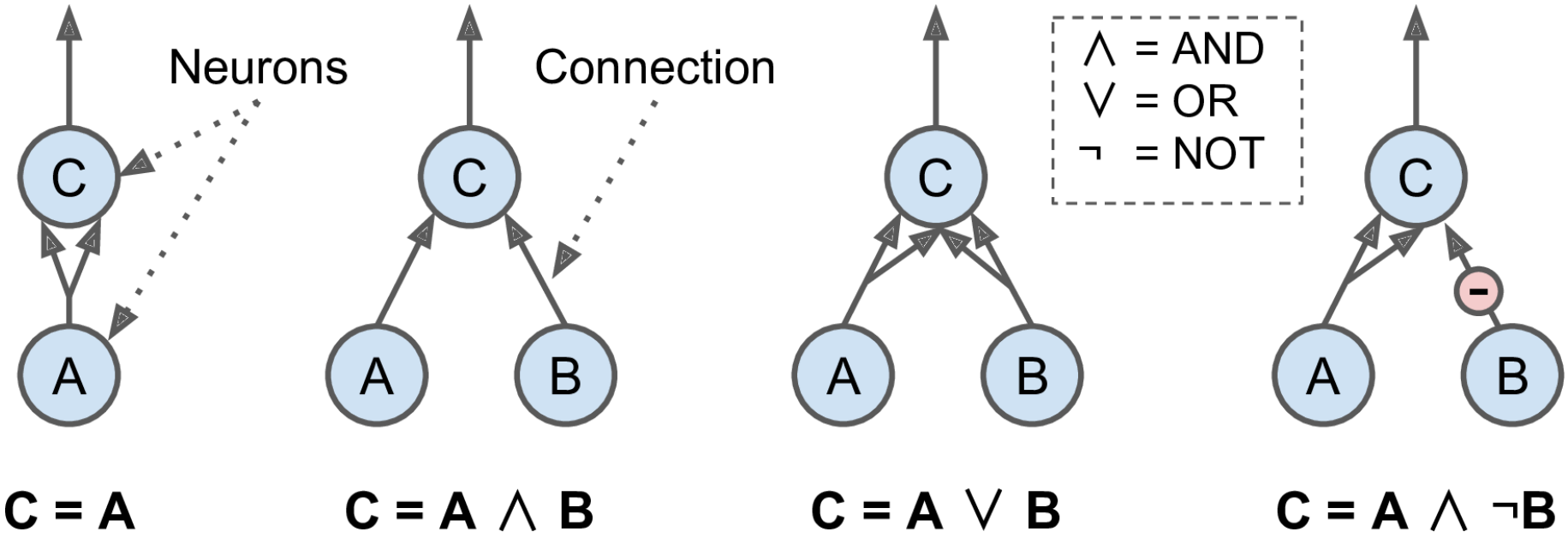
- Las neuronas biológicas individuales parecen comportarse de una manera bastante simple, pero están organizadas en una vasta red de miles de millones de neuronas, cada neurona típicamente **conectada a miles de otras neuronas**.



Contenido

Cálculos lógicos con neuronas I

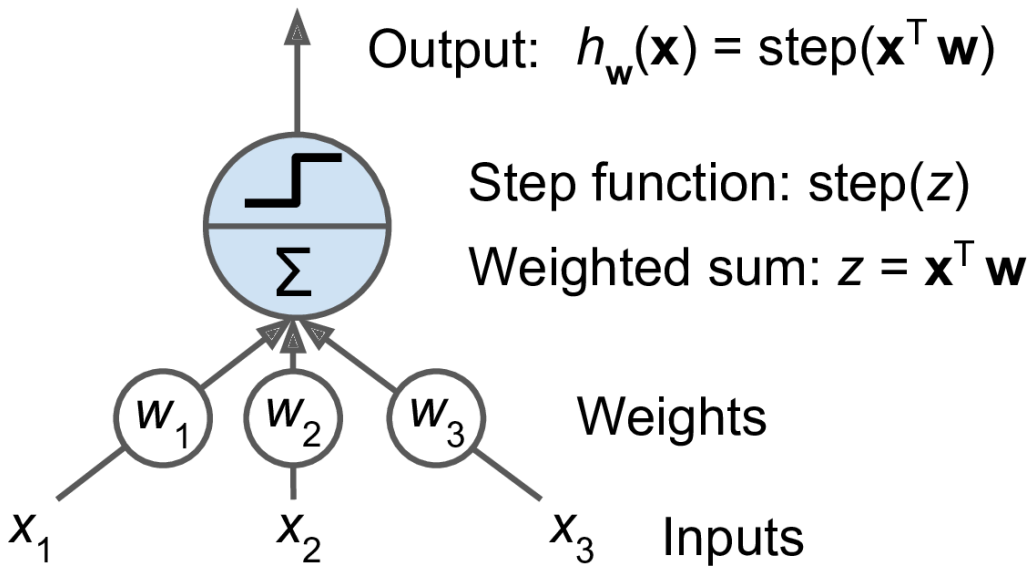
Modelo de neurona artificial: Tiene una o más entradas binarias (encendido/apagado) y una salida binaria. La neurona artificial simplemente activa su salida cuando más de un cierto número de sus entradas están activas.



Se supone que una neurona se activa cuando al menos dos de sus entradas están activas.

Contenido

Perceptrón I



- El **Perceptrón** es una de las arquitecturas ANN más simples, inventada en 1957 por Frank Rosenblatt.
- Se basa en una neurona artificial llamada unidad lógica de umbral (TLU) o, a veces, una unidad de umbral lineal (LTU).

Las entradas y salidas pueden ser **números reales** (en lugar de valores binarios de activación/desactivación) y cada conexión de entrada está asociada a un **peso**.

Perceptrón II

- La TLU calcula una suma ponderada de sus entradas $z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \boldsymbol{X}^T \boldsymbol{W}$, luego aplica una *step function* a esa suma: $h_w(\boldsymbol{X}) = step(z)$, donde $z = \boldsymbol{X}^T \boldsymbol{W}$.
- La *step function* más común utilizada en Perceptrons es la **Heaviside**.

$$heaviside(z) = \begin{cases} 0 & if \quad z < 0 \\ 1 & otherwise \end{cases}$$

- A veces se usa la función de signo en su lugar.

$$sgn(z) = \begin{cases} -1 & if \quad z < 0 \\ 0 & if \quad z = 0 \\ +1 & if \quad z > 0 \end{cases}$$

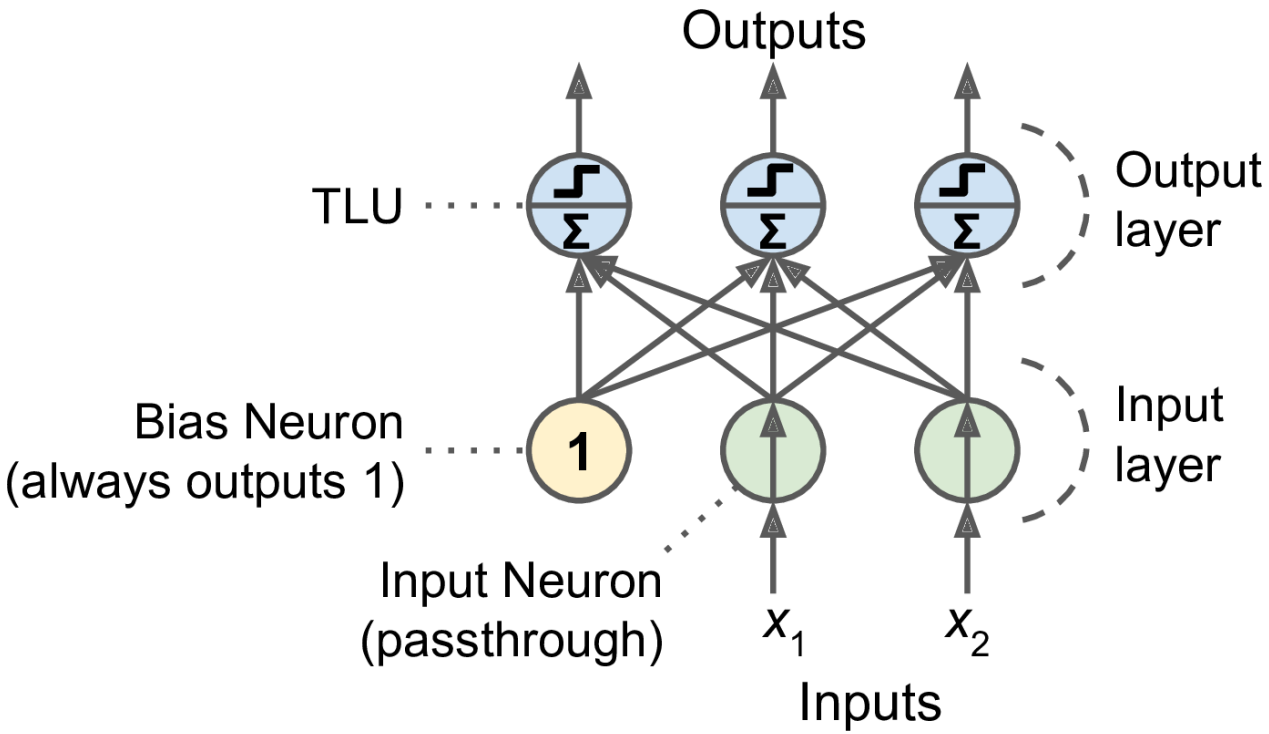
Se calcula una combinación lineal de las entradas y, si el resultado excede un umbral, genera la clase positiva o la clase negativa

Perceptrón III

- Un **perceptrón** está compuesto por una sola capa de TLU conectada a todas las entradas.
- Cuando todas las **neuronas** en una capa **están conectadas** a cada neurona en la **capa anterior**, se define como una capa completamente conectada o densa.
- Todas las neuronas de entrada forman la **capa de entrada**.
- Generalmente se agrega una característica de sesgo adicional ($x_0 = 1$), representada por la **neurona de sesgo**.

Perceptrón IV

Ejemplo: Un Perceptrón con dos entradas y tres salidas.



Este Perceptrón puede clasificar instancias simultáneamente en tres clases binarias diferentes **(clasificador de salida múltiple)**.

Perceptrón V

Gracias a la magia del álgebra lineal, es posible calcular eficientemente los resultados de una capa de neuronas artificiales para varias instancias:

$$h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{X}) = \phi\left(\boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}\right) \tag{1}$$

* $\boldsymbol{X}$ representa la matriz de características de entrada.
* La matriz de peso $\boldsymbol{W}$ contiene todos los pesos de conexión.
* El vector bias $\boldsymbol{b}$ contiene todos los pesos de conexión entre la neurona de bias y las neuronas artificiales.
* La función $\phi\left(\cdot\right)$ se conoce como función de activación.

Perceptrón VI: Cómo se entrena?

El algoritmo de entrenamiento de Perceptrón se inspiró en gran medida en la **regla de Hebb**.

- "El peso de conexión entre dos neuronas aumenta cuando tienen la misma salida."
- Se **tiene en cuenta el error cometido por la red**; refuerza las conexiones que ayudan a reducir el error.
- Específicamente, el perceptrón se alimenta de una instancia de entrenamiento a la vez, y **para cada instancia hace sus predicciones**.

Para cada neurona de salida que produjo una predicción incorrecta, **se refuerzan los pesos de conexión de las entradas que habrían contribuido a la predicción correcta**.

Perceptrón VII: Cómo se entrena?

Regla de aprendizaje de perceptrón **(actualización de peso)**:

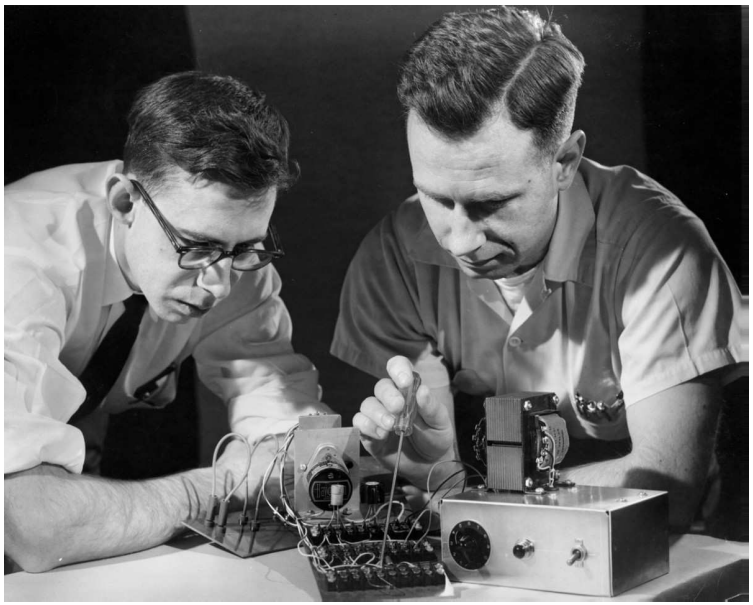
$$w_{i,j}^{next\_step} = w_{i,j} + \eta \left(y_j - \hat{y}_j\right) x_i \tag{2}$$

- wi, j es el peso de conexión entre la i-ésima neurona de entrada y la j-ésima neurona de salida.
- x_i es el i-ésimo valor de **entrada** de la instancia de entrenamiento actual.
- $\hat{y}_j$ es la **salida** de la neurona de salida j para la instancia de entrenamiento actual.
- y_j es la salida objetivo de la neurona de salida j para la instancia de entrenamiento actual.
- η es la tasa de aprendizaje.

Teorema de convergencia del perceptrón

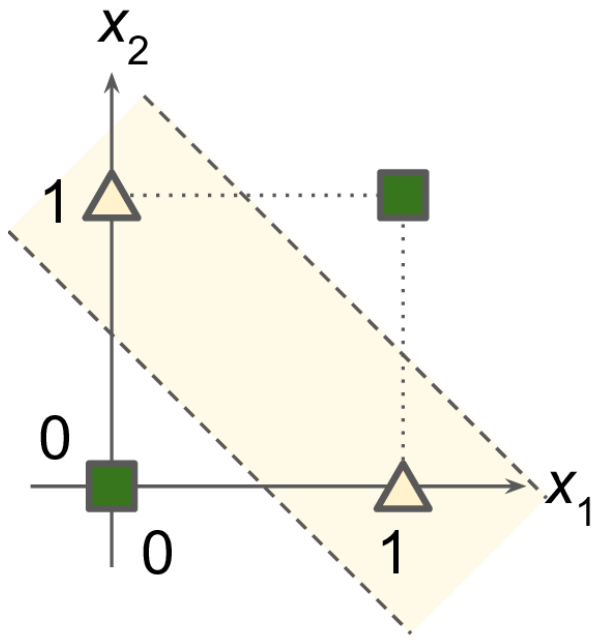
El **límite de decisión** de cada neurona de salida es lineal, por lo que los perceptrones son incapaces de aprender patrones complejos.

Sin embargo, si las instancias de entrenamiento son **linealmente separables**, *Rosenblatt* demostró que este algoritmo convergería en una solución. Esto se denomina teorema de convergencia de perceptrón.



Perceptrón: Debilidades

- *Marvin Minsky* y *Seymour Papert* 1969: "**Perceptrons**".
- Los perceptrones son incapaces de resolver algunos problemas triviales (por ejemplo, el problema de clasificación Exclusivo OR (XOR).



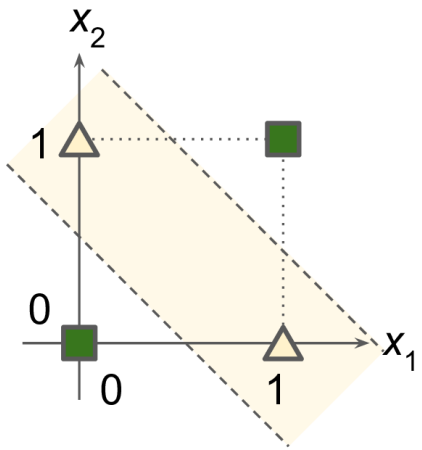
- Por supuesto, esto también es cierto para cualquier otro modelo de clasificación lineal **(como los clasificadores de regresión logística)**.

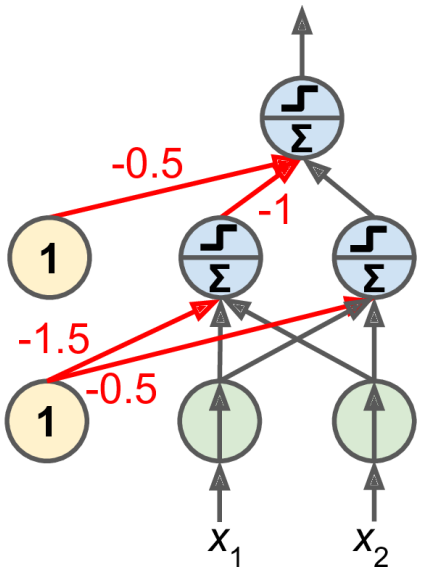
Muchos investigadores abandonaron las redes neuronales por completo.

Contenido

MLP

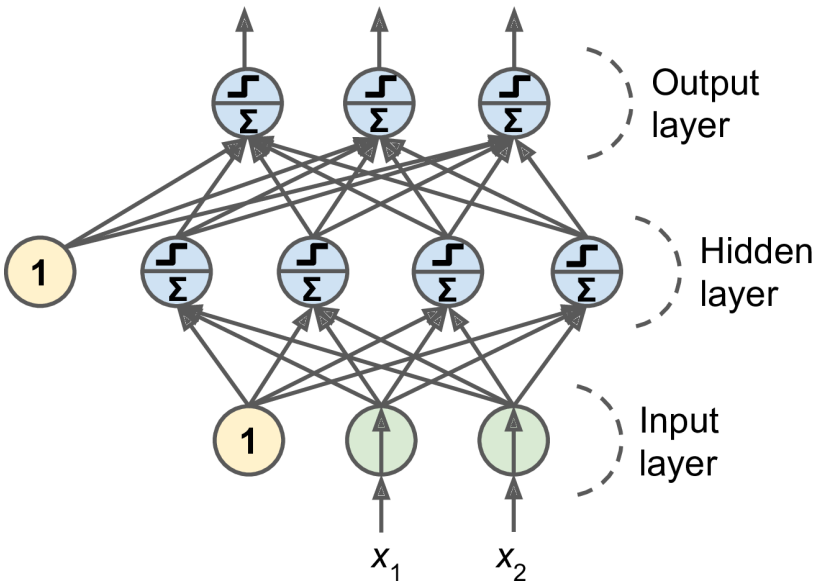
- Se pueden apilar múltiples perceptrones.
- La ANN resultante se llama un
- **perceptrón multicapa (MLP)**.
- Un MLP **puede resolver el problema XOR**: con las entradas (0, 0) o (1, 1) la salida 0, y con las entradas (0, 1) o (1, 0) emite 1.
- Ejercicio: Demostrar la salida de la XOR con MLP.





Todas las conexiones tienen un peso igual a 1, excepto las cuatro conexiones donde se muestra el peso.

MLP and backpropagation I



- Un MLP se compone de una capa de entrada, una o más capas de TLU llamadas capas ocultas, y una capa final de TLU llamada capa de salida.
- Cada capa, excepto la de salida, **incluye una neurona bias** y está completamente conectada a la siguiente.
- Ver https://playground.tensorflow.org

La señal fluye solo en una dirección, desde las entradas a las salidas: Feedforward Neural Network (FNN).

MLP and backpropagation II

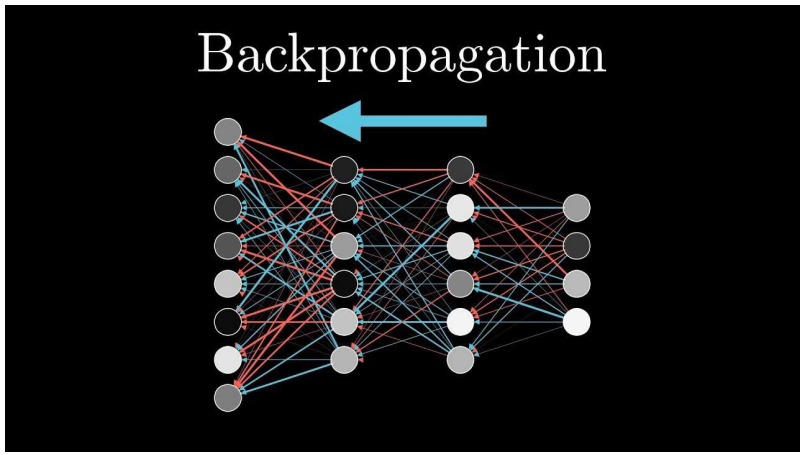
* Durante muchos años, los investigadores lucharon por **encontrar una manera de entrenar MLP**, sin éxito.

* En **1986**, *David Rumelhart*, *Geoffrey Hinton* y *Ronald Williams* publicaron un documento innovador que presenta el algoritmo de entrenamiento *backpropagation.*

* En resumen, es simplemente **Gradiente descendente** usando una técnica eficiente para calcular los gradientes automáticamente.

MLP and backpropagation III

En solo dos pasos a través de la red **(uno hacia adelante, uno hacia atrás)**, el algoritmo de *backpropagation* puede calcular el gradiente del error de la red con respecto a cada parámetro del modelo.



Es decir, **puede descubrir cómo se debe ajustar cada peso de conexión y cada término de sesgo para reducir el error**.

Una vez que tiene estos gradientes, solo realiza un paso GD regular, y todo el proceso se repite hasta que la red **converja en la solución**.

En resumen...

- Para cada instancia de entrenamiento, el **algoritmo de retropropagación** primero realiza una predicción (*forward pass*), mide el error, luego pasa a través de cada capa en reversa para medir la contribución del error de cada conexión (*backward pass*), y finalmente modifica ligeramente los pesos de conexión para reducir el error (*GD pass*).

`https://thumbs.gfycat.com/FickleHorribleBlackfootedferret-small.gif`

MLP: Funciones de activación I

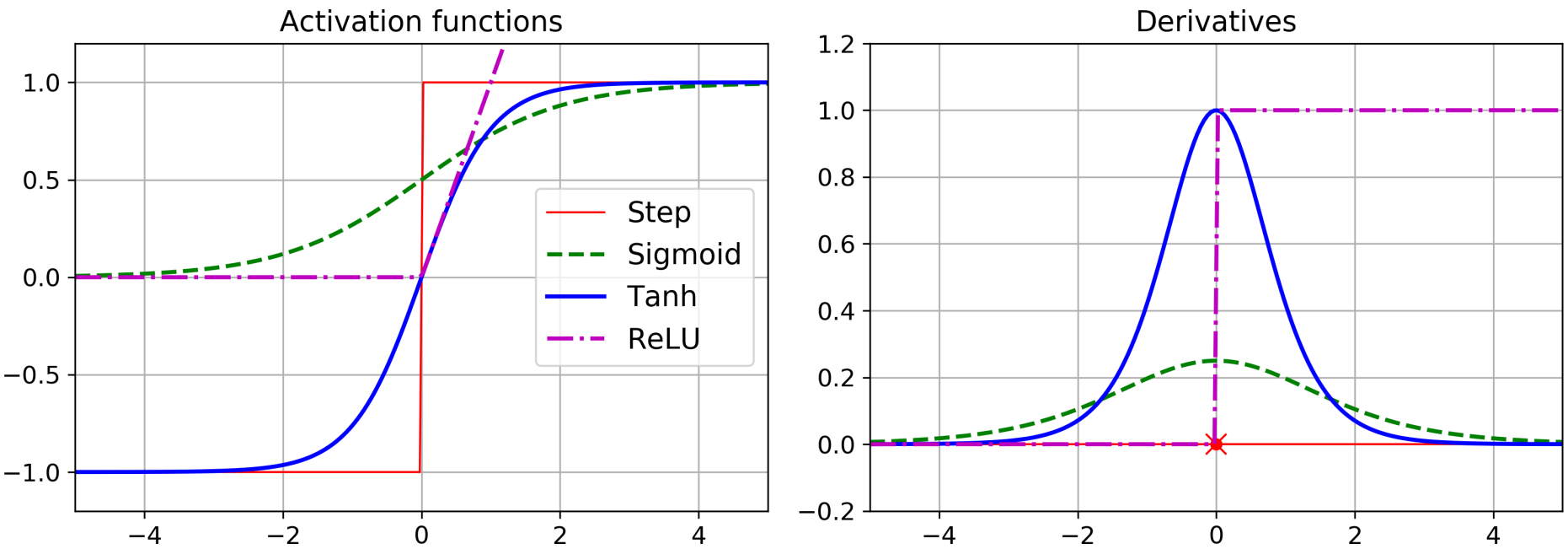
- Los autores realizaron un **cambio clave en la arquitectura de MLP**: reemplazaron la función de paso con la función logística, $\sigma(z) = 1/(1 + \exp(z))$.

- La función de paso contiene solo segmentos planos, por lo que no hay gradiente con el que trabajar.

- La **función logística tiene una derivada no nula**, lo que permite que el GD progresar a cada paso.

- Otras dos funciones de activación populares son:
  - The **hyperbolic tangent function** $\tanh(z) = 2\sigma(2z)1$
  - The **Rectified Linear Unit function**: $ReLU(z) = max(0, z)$

MLP: Funciones de activación II



- Si **encadenamos varias transformaciones lineales**, todo lo que obtiene es una **transformación lineal**.
- Entonces, si no tiene cierta **no-linealidad** entre las capas, incluso una pila profunda de capas es equivalente a una sola capa.
- No puede resolver **problemas muy complejos** con eso:

Contenido

MLP para regresión I

- Si desea predecir un valor único.
  - **Ejemplo:** el precio de una casa dada muchas de sus características, entonces solo necesita una neurona de salida única: su salida es el valor predicho.
- Regresión multivariada (es decir, para predecir múltiples valores a la vez), necesita una neurona de salida por dimensión de salida.
  - **Ejemplo:** para ubicar el centro de un objeto en una imagen, debe predecir las coordenadas 2D, por lo que necesita dos neuronas de salida. Si también desea colocar un cuadro delimitador alrededor del objeto, entonces necesita dos números más: el ancho y la altura del objeto. Entonces terminas con 4 neuronas de salida.

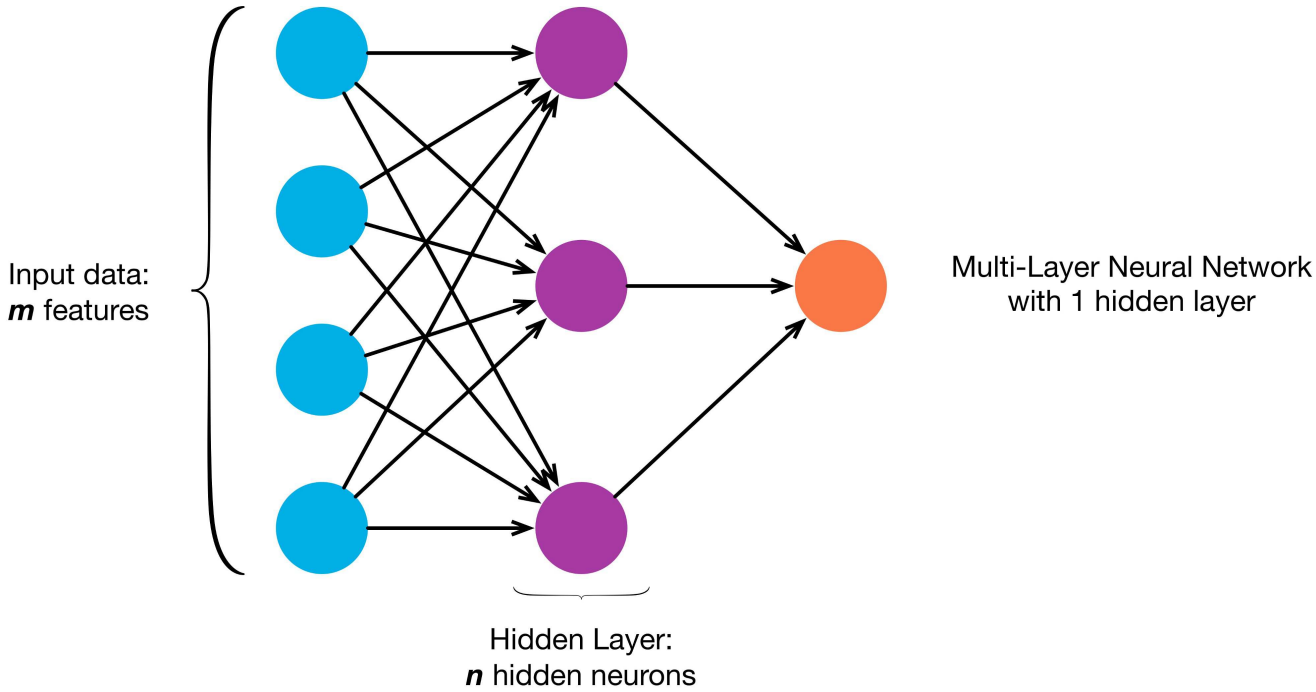
MLP para regresión II

- Al construir un MLP para la regresión, **no se utiliza ninguna función de activación para las neuronas de salida.**
- Sin embargo, si desea garantizar que la salida siempre será positiva, se puede usar la función de activación *ReLU* o la función de activación *softplus* en la capa de salida.
- Si desea garantizar que las predicciones se encuentren dentro de un rango de valores dado, puede usar la función $\sigma()$ o la tanh() y escalar las etiquetas al rango apropiado.

La *loss function* que se usa durante el entrenamiento suele ser el **error cuadrático medio**, pero si tiene muchos valores atípicos, puede preferir usar el **error absoluto medio**.

MLP para clasificación I

Para un problema de **clasificación binaria**, solo se necesita una neurona de salida única que utilice la función de activación logística: la salida será un número entre 0 y 1.

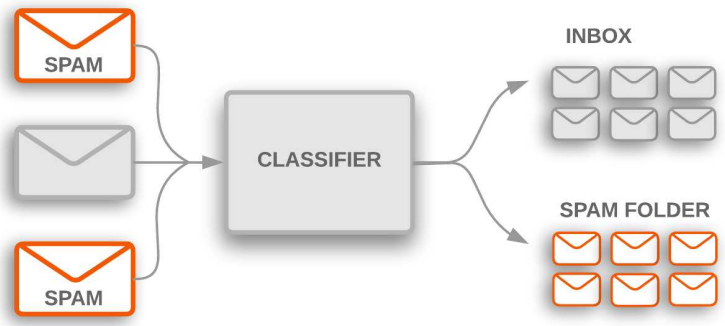


Esto se puede interpretar como la probabilidad estimada de la clase positiva.

MLP para clasificación II

- Los MLP también pueden manejar fácilmente tareas de **clasificación binaria** de múltiples etiquetas.

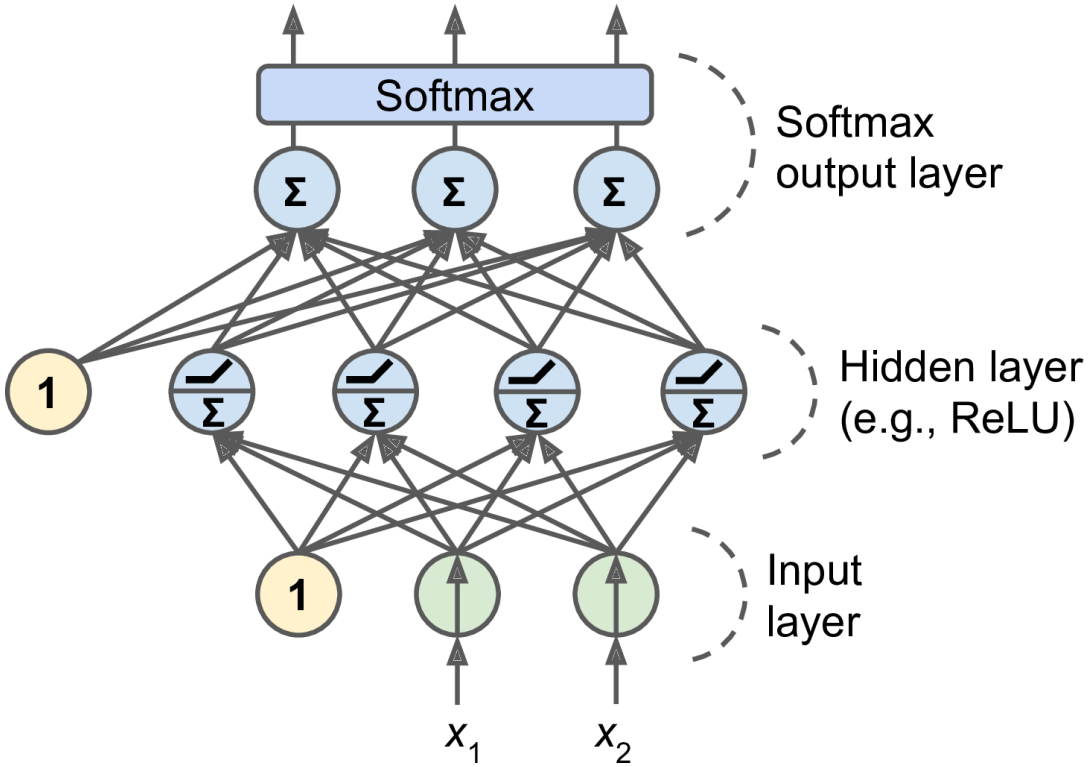
– **Ejemplo:** podría tener un sistema de clasificación de correo electrónico que prediga si cada correo electrónico entrante es no deseado o no deseado, y simultáneamente predice si es un correo electrónico urgente o no urgente.



En este caso, necesitaría dos neuronas de salida, ambas utilizando la función de activación logística.

MLP para clasificación III

Si cada instancia puede pertenecer solo a una sola clase, de 3 o más clases posibles, entonces debe tener una neurona de salida por clase, y debe usar la función de activación *softmax* para toda la capa de salida.



La función *softmax* asegurará que todas las probabilidades estimadas estén entre 0 y 1 y que sumen uno. Esto se llama **clasificación multi-clase**.

Referencias I

Géron, A., (2019).
Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.
O'Reilly Media.