Introducción a Redes Neuronales Artificiales

D. A. Pérez, M.eng.
dieaperezros@unal.edu.co

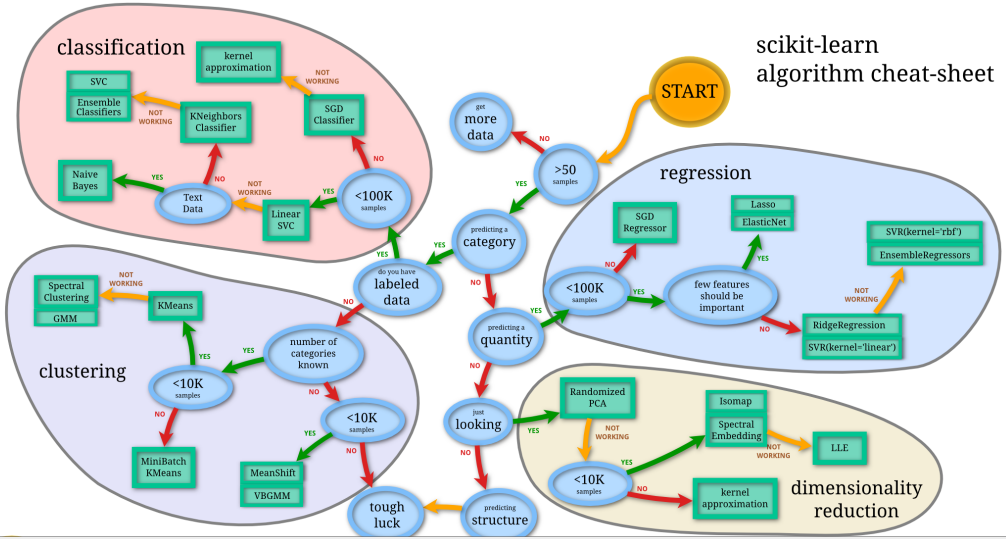
Universidad Nacional de Colombia
Signal Processing and Recognition Group - SPRG

October 17, 2024

Contenido

Contenido

Métodos clásicos de aprendizaje de máquina - Scikit-learn



Aprendizaje automática actual

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.

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.

Concepto de deep learning

- El concepto de *Deep Learning* surgió en los años 1980, cuando investigadores como John Hopfield y Yann LeCun comenzaron a trabajar en redes neuronales artificiales.
- En 2006, Geoffrey Hinton y sus colaboradores introdujeron el término *Deep Learning*, basando sus ideas en redes neuronales profundas, con múltiples capas.
- Este enfoque revolucionó áreas como la visión por computadora y el reconocimiento de voz, permitiendo que las máquinas aprendieran patrones complejos directamente de los datos sin intervención manual.

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.

Premio Turing con IA

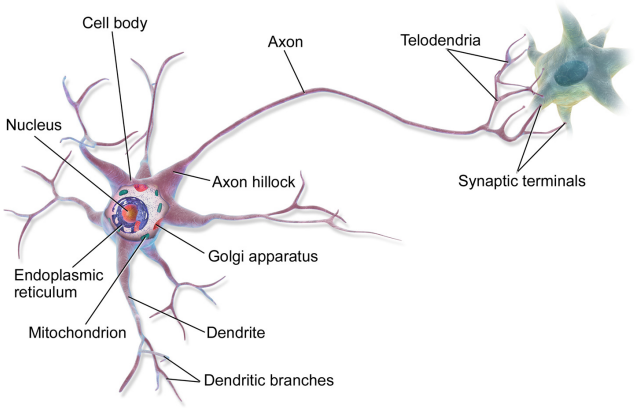
- En 2018, Geoffrey Hinton, Yann LeCun y Yoshua Bengio fueron galardonados con el Premio Turing por sus contribuciones al campo de la inteligencia artificial, especialmente en el desarrollo del aprendizaje profundo (*Deep Learning*).
- Estos investigadores son conocidos como los "Padrinos del Deep Learning".

Premio nobel de física 2024 con IA


- John Hopfield y Geoffrey Hinton ganaron el Premio Nobel de Física 2024 por sus descubrimientos sobre el aprendizaje automático con redes neuronales artificiales.
- Hopfield creó una memoria asociativa en 1982, mientras que Hinton desarrolló métodos clave para que las máquinas aprendan de los datos.

Contenido

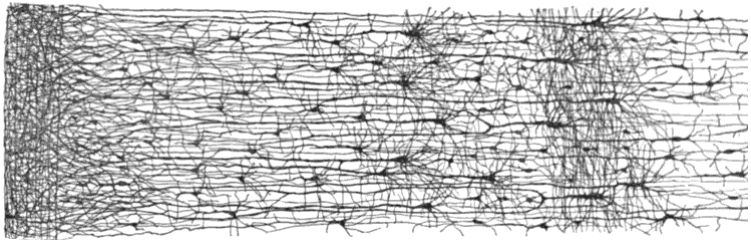
Neurona I



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

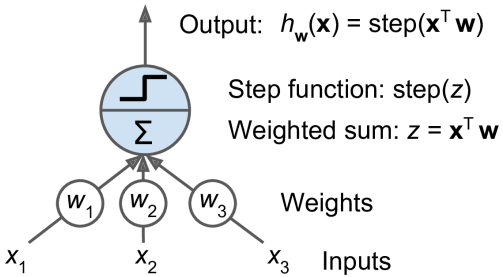
Neurona II

- 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

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.

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

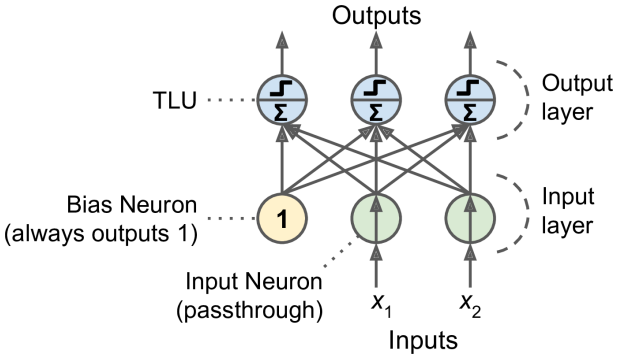
Se calcula una combinación lineal de las entradas y, si el resultado excede un umbral, genera la clase positiva o la clase negativa

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 VII: Cómo se entrena?

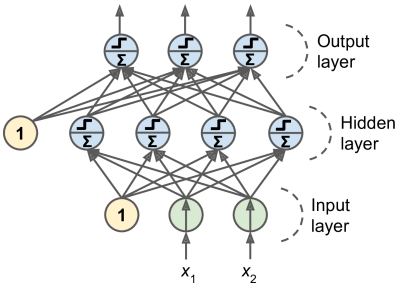
Regla de aprendizaje de perceptrón **(actualización de peso)**:

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

- 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.

Contenido

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.

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

MLP and backpropagation II

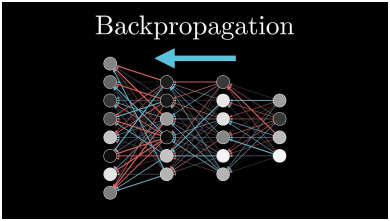
* Durante muchos años, los investigadores lucharon por **encontrar una manera de entrenar MLP**, sin éxito.

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



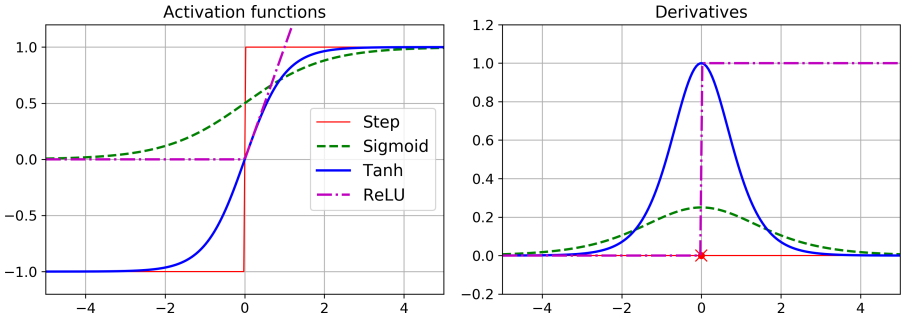
Es decir, **puede descubrir cómo se debe ajustar cada peso de conexión y cada término de sesgo para reducir el error**.

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**.

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.

Muchas Gracias!

Contacto:

Diego Armando Pérez Rosero

dieaaperezros@unal.edu.co