¿Por qué aprenden las redes neuronales?

Teorema de la aproximación universal

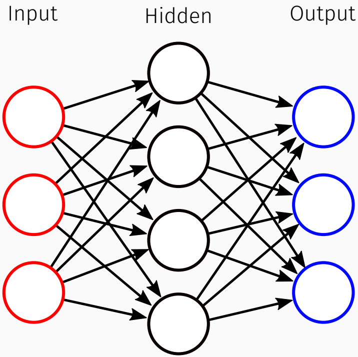
Manuel Carlevaro

Grupo de Materiales Granulares - Dpto. Ing. Mecánica, UTN FRLP.

Taller de aprendizaje GMG - ML
6 de octubre, 2020

¿Por qué aprenden las redes neuronales?

Repaso: ¿qué hacen las NN?



- Colección de neuronas que intentan $x \mapsto y$
- Entrenamiento - *backpropagation* $f : y = f(x, \boldsymbol{\omega}, \boldsymbol{b})$
- Función de pérdida: distancia entre valor predicho y actual

Teorema de aproximación universal

Una red *feedforward* con una capa *output* lineal y al menos una capa oculta con cualquier función de activación acotada puede aproximar cualquier función Borel-medible[1], desde un espacio dimensional finito a otro con error distinto de cero de magnitud arbitraria , siempre que la red tenga suficientes neuronas ocultas.

Teorema de aproximación universal - versión corta

Una red neuronal con una capa oculta que contenga un número suficiente de neuronas puede aproximar cualquier función continua con precisión arbitraria, bajo ciertas condiciones para las funciones de activación.

[1] Cualquier función continua definida en un subconjunto cerrado y acotado de $\mathbb{R}^n$ es Borel-medible

Teorema de aproximación universal

Una red *feedforward* con una capa *output* lineal y al menos una capa oculta con cualquier función de activación acotada puede aproximar cualquier función Borel-medible[1], desde un espacio dimensional finito a otro con error distinto de cero de magnitud arbitraria , siempre que la red tenga suficientes neuronas ocultas.

Teorema de aproximación universal - versión corta

Una red neuronal con una capa oculta que contenga un número suficiente de neuronas puede aproximar cualquier función continua con precisión arbitraria, bajo ciertas condiciones para las funciones de activación.

[1] Cualquier función continua definida en un subconjunto cerrado y acotado de $\mathbb{R}^n$ es Borel-medible

Notas

- Probado por Cybenko en 1989[2] solo para funciones sigmoideas.
- En 1991, Hornik[3] probó que aplica a todas las funciones de activación (es la arquitectura de la red, no la elección de la función, lo que determina el desempeño de la red).
- Generalizado por Leshno y colaboradores[4] para funciones de activación continuas por tramos, localmente acotadas, siempre que no sea un polinomio.

[2] George Cybenko. "Approximation by superpositions of a sigmoidal function". En: *Mathematics of control, signals and systems* 2.4 (1989), págs. 303-314.
[3] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". En: *Neural Networks* 4.2 (1991), págs. 251-257.
[4] Moshe Leshno y col. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". En: *Neural Networks* 6.6 (1993), págs. 861-867.

Notas

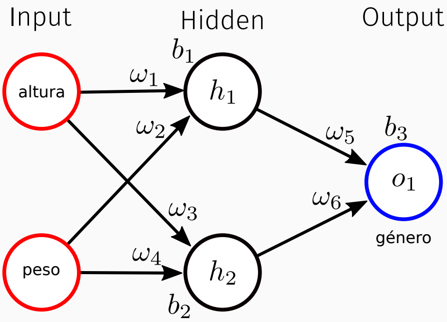
- Probado por Cybenko en 1989[2] solo para funciones sigmoideas.
- En 1991, Hornik[3] probó que aplica a todas las funciones de activación (es la arquitectura de la red, no la elección de la función, lo que determina el desempeño de la red).
- Generalizado por Leshno y colaboradores[4] para funciones de activación continuas por tramos, localmente acotadas, siempre que no sea un polinomio.

[2] George Cybenko. "Approximation by superpositions of a sigmoidal function". En: *Mathematics of control, signals and systems* 2.4 (1989), págs. 303-314.
[3] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". En: *Neural Networks* 4.2 (1991), págs. 251-257.
[4] Moshe Leshno y col. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". En: *Neural Networks* 6.6 (1993), págs. 861-867.

Notas

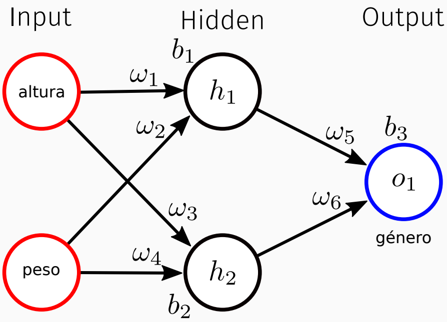
- Probado por Cybenko en 1989[2] solo para funciones sigmoideas.
- En 1991, Hornik[3] probó que aplica a todas las funciones de activación (es la arquitectura de la red, no la elección de la función, lo que determina el desempeño de la red).
- Generalizado por Leshno y colaboradores[4] para funciones de activación continuas por tramos, localmente acotadas, siempre que no sea un polinomio.

[2] George Cybenko. "Approximation by superpositions of a sigmoidal function". En: *Mathematics of control, signals and systems* 2.4 (1989), págs. 303-314.
[3] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". En: *Neural Networks* 4.2 (1991), págs. 251-257.
[4] Moshe Leshno y col. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". En: *Neural Networks* 6.6 (1993), págs. 861-867.



$$h_1 = \omega_1[\text{altura}] + \omega_2[\text{peso}] + b_1$$
$$h_2 = \omega_3[\text{altura}] + \omega_4[\text{peso}] + b_2$$
$$o_1 = \omega_5 h_1 + \omega_6 h_2 + b_3$$

Regresión lineal



$$h_1 = \omega_1[\text{altura}] + \omega_2[\text{peso}] + b_1$$
$$h_2 = \omega_3[\text{altura}] + \omega_4[\text{peso}] + b_2$$
$$o_1 = \omega_5 h_1 + \omega_6 h_2 + b_3$$

Regresión lineal

¿Funciona el TAU?: no.

Cualquier función definida sobre un conjunto compacto (acotado y cerrado) puede ser aproximada por una función definida por tramos:

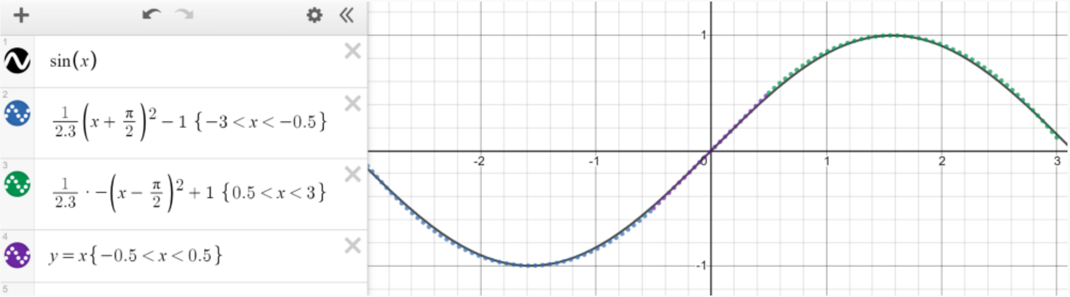


Imagen de Ye[4]

Aproximación por funciones a tramos constantes

Cybenko especifica que la función definida a tramos puede ser **constante**:

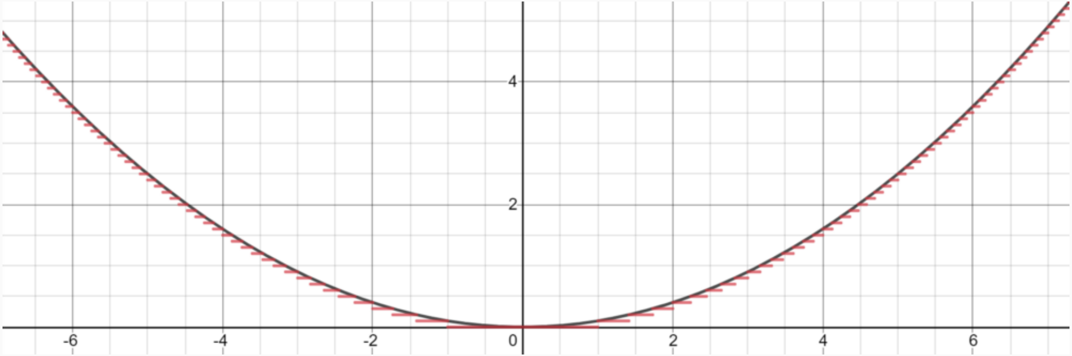
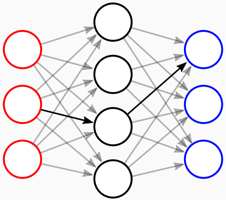


Imagen de Ye[4]

¿Cómo funciona?

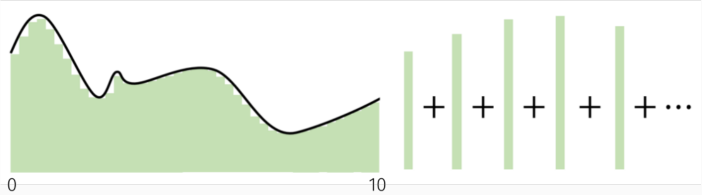


- Para *inputs* que caen cerca de la neurona responsable:

$$\omega_i \gg 0,\ \sigma \to 1$$

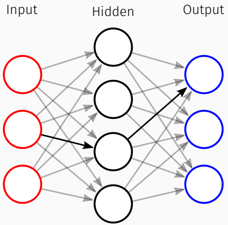
- El resto:

$$\omega_i \ll 0,\ \sigma \to 0$$



Adaptado de Ye[4]

¿Cómo funciona?

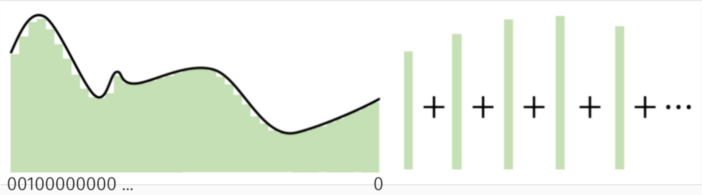


- Para *inputs* que caen cerca de la neurona responsable:

$$\omega_i \gg 0,\ \sigma \to 1$$

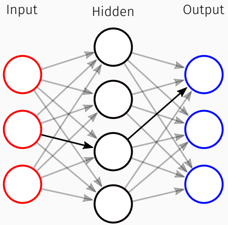
- El resto:

$$\omega_i \ll 0,\ \sigma \to 0$$



Adaptado de Ye[4]

¿Cómo funciona?

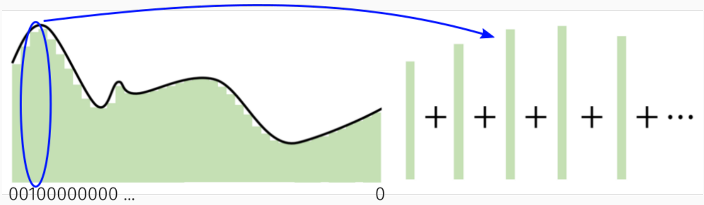


- Para *inputs* que caen cerca de la neurona responsable:

$$\omega_i \gg 0,\ \sigma \to 1$$

- El resto:

$$\omega_i \ll 0,\ \sigma \to 0$$



Adaptado de Ye[4]

Comentarios

- El punto clave del TAU es que en vez de crear relaciones matemáticas complejas entre la entrada y la salida, se utilizan manipulaciones lineales simples para dividir la función compleja en muchas piezas simples menos complejas, cada una de las cuales es manipulada por una neurona.
- Desde la prueba inicial del TAU por Cybenko, se han realizado muchos desarrollos para validar el TAU para diferentes funciones de activación (por ejemplo para ReLU).
- Cada neurona “vigila” un patrón o cierta área del espacio de características (*features*), cuyo tamaño está determinado por el número de neuronas en la red. Si hay pocas neuronas, mayor es el espacio que está a cargo de cada una, y por lo tanto la capacidad de aproximar decae. Con más neuronas, independientemente de la función de activación, cualquier función puede aproximarse con pequeños fragmentos.

Comentarios

- El punto clave del TAU es que en vez de crear relaciones matemáticas complejas entre la entrada y la salida, se utilizan manipulaciones lineales simples para dividir la función compleja en muchas piezas simples menos complejas, cada una de las cuales es manipulada por una neurona.
- Desde la prueba inicial del TAU por Cybenko, se han realizado muchos desarrollos para validar el TAU para diferentes funciones de activación (por ejemplo para ReLU).
- Cada neurona “vigila” un patrón o cierta área del espacio de características (*features*), cuyo tamaño está determinado por el número de neuronas en la red. Si hay pocas neuronas, mayor es el espacio que está a cargo de cada una, y por lo tanto la capacidad de aproximar decae. Con más neuronas, independientemente de la función de activación, cualquier función puede aproximarse con pequeños fragmentos.

Comentarios

- El punto clave del TAU es que en vez de crear relaciones matemáticas complejas entre la entrada y la salida, se utilizan manipulaciones lineales simples para dividir la función compleja en muchas piezas simples menos complejas, cada una de las cuales es manipulada por una neurona.
- Desde la prueba inicial del TAU por Cybenko, se han realizado muchos desarrollos para validar el TAU para diferentes funciones de activación (por ejemplo para ReLU).
- Cada neurona “vigila” un patrón o cierta área del espacio de características (*features*), cuyo tamaño está determinado por el número de neuronas en la red. Si hay pocas neuronas, mayor es el espacio que está a cargo de cada una, y por lo tanto la capacidad de aproximar decae. Con más neuronas, independientemente de la función de activación, cualquier función puede aproximarse con pequeños fragmentos.

$$\boxed{\text{Propósito de NN} \mapsto \textbf{generalizar}}$$

Por lo general fallan en **extrapolar:**

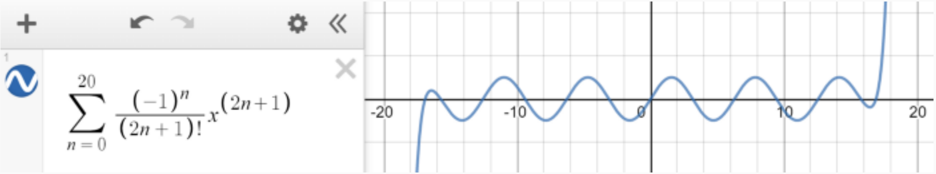


Imagen de Ye[4]

Comentarios finales

- El TAU establece que independientemente de la función que tratamos de aprender, un MLP suficientemente grande será capaz de representar dicha función. Sin embargo, no está garantizado que el algoritmo de entrenamiento sea capaz de aprender tal función.

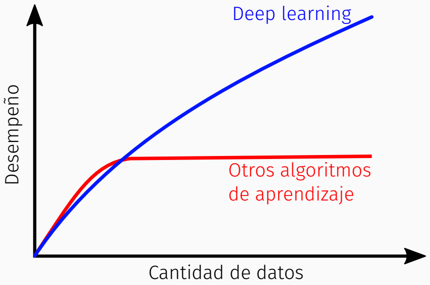
Comentarios finales

- El TAU establece que independientemente de la función que tratamos de aprender, un MLP suficientemente grande será capaz de representar dicha función. Sin embargo, no está garantizado que el algoritmo de entrenamiento sea capaz de aprender tal función.
- El TAU dice que hay una red suficientemente grande para alcanzar cualquier grado de precisión, pero no dice cuán grande debe ser.

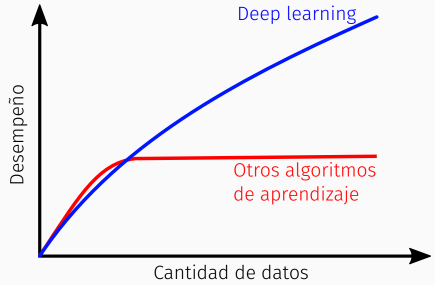
Comentarios finales

- El TAU establece que independientemente de la función que tratamos de aprender, un MLP suficientemente grande será capaz de representar dicha función. Sin embargo, no está garantizado que el algoritmo de entrenamiento sea capaz de aprender tal función.
- El TAU dice que hay una red suficientemente grande para alcanzar cualquier grado de precisión, pero no dice cuán grande debe ser.
- El TAU asume que hay solo una capa oculta. A medida que se agregan más capas ocultas, la complejidad y por la tanto la aproximación universal crecen exponencialmente. Neuronas de la segunda capa buscan **patrones dentro de patrones**.

La supremacía cuántica



La supremacía cuántica



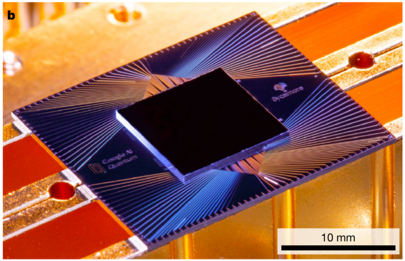


Fig. 1 | The Sycamore processor. a, Layout of processor, showing a rectangular array of 54 qubits (grey), each connected to its four nearest neighbours with couplers (blue). The inoperable qubit is outlined. **b**, Photograph of the Sycamore chip.

"*Quantum supremacy using a programmable superconducting processor*". Google, 23 de octubre 2019 (Nature): 10.000 años de cálculo en una supercomputadora $\mapsto$ 200 segundos.

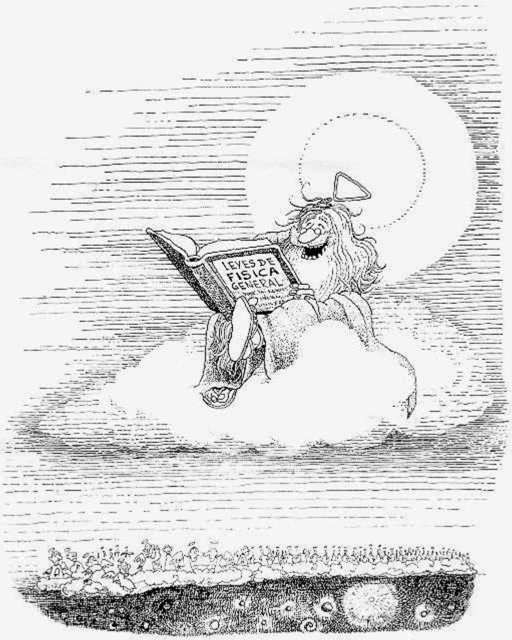
[1] George Cybenko. "Approximation by superpositions of a sigmoidal function". En: *Mathematics of control, signals and systems* 2.4 (1989), págs. 303-314.

[2] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". En: *Neural Networks* 4.2 (1991), págs. 251-257.

[3] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus y Shimon Schocken. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". En: *Neural Networks* 6.6 (1993), págs. 861-867.

[4] Andre Ye. *You Don't Understand Neural Networks Until You Understand the Universal Approximation Theorem*. 2020. URL: https://medium.com/analytics-vidhya/you-dont-understand-neural-networks-until-you-understand-the-universal-approximation-theorem-85b3e7677126 (visitado 05-10-2020).

[5] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.


LEYES DE
FISICA
GENERAL

En el próximo episodio:

¿...?

manuel.carlevaro@gmail.com

• LuaLaTeX