Introducción a la Ciencia de Datos

Maestría en Probabilidad y Estadística

Dr. Marco Antonio Aquino López

Centro de Investigación en Matemáticas

Agosto-Diciembre 2025

Minimización del riesgo esperado

Queremos encontrar una función f que minimice la pérdida esperada:

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[L(f(x), y)\big]$$

En la práctica sólo tenemos una muestra $\{(x_i, y_i)\}_{i=1}^{n}$, por lo que minimizamos el **riesgo empírico**:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)$$

Minimización del riesgo esperado

Queremos encontrar una función f que minimice la pérdida esperada:

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[L(f(x), y)\big]$$

En la práctica sólo tenemos una muestra $\{(x_i, y_i)\}_{i=1}^{n}$, por lo que minimizamos el **riesgo empírico**:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)$$

Idea clave

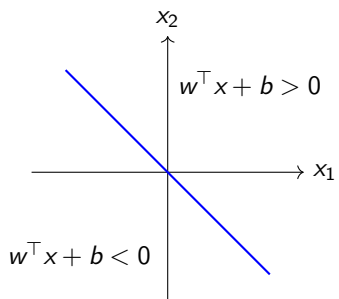
El aprendizaje supervisado busca $f^* = \arg\min_f R(f)$, pero sólo disponemos de $\hat{R}(f)$.

Clasificadores lineales

Un **clasificador lineal** define una frontera:

$$w^\top x + b = 0$$

- Las regiones de decisión son **semiespacios**.
- En $\mathbb{R}^2$, la frontera es una línea recta.
- En $\mathbb{R}^3$, un plano.

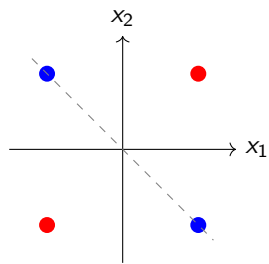


Ejemplo: el problema XOR

Datos:

$$(1,1) \to 0, \quad (-1,-1) \to 0, \quad (1,-1) \to 1, \quad (-1,1) \to 1$$

- Ninguna línea puede separar correctamente las dos clases.
- Las regiones positivas y negativas de un clasificador lineal son **convexas**.
- El patrón XOR requiere regiones **disjuntas y no convexas**.

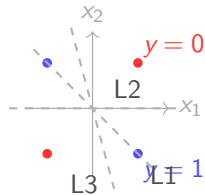


No separabilidad del XOR

Argumento geométrico

Las regiones decidibles por un umbral lineal son convexas. El conjunto de puntos de cada clase en XOR no es convexamente separable.

Clase 0: $(1, 1)$, $(-1, -1)$
Clase 1: $(1, -1)$, $(-1, 1)$



Ninguna de estas rectas separa correctamente las clases.

- Podemos rotar o trasladar infinitas líneas, pero nunca lograremos separar perfectamente.
- $\Rightarrow$ Necesitamos una representación del espacio que permita fronteras **no lineales**.

Animación sugerida: exploración de rectas

Animación

Propósito

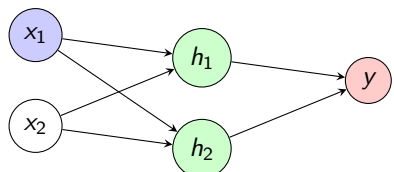
Visualizar que ningún clasificador lineal puede separar las clases.

Más allá de lo lineal

- Introducimos **transformaciones no lineales** de los datos:

$$z = g(Wx + b)$$

- Cada capa aplica una transformación que puede separar clases no lineales.
- Las funciones $g(\cdot)$ son **activaciones**: sigmoide, tanh, ReLU, etc.



Idea: una capa oculta con activaciones permite representar el XOR.

Breve historia de las redes neuronales

- **Décadas de 1940–1950:** McCulloch y Pitts (1943) proponen una neurona formal capaz de computar funciones lógicas. Hebb (1949) introduce la regla de aprendizaje: "neurons that fire together, wire together".
- **1958:** Rosenblatt presenta el **perceptrón**, primer algoritmo entrenable basado en error.
- **1969:** Minsky y Papert demuestran que el perceptrón simple **no puede resolver XOR**.
- **1986:** Rumelhart, Hinton y Williams reintroducen la **retropropagación del error**, permitiendo redes multicapa.
- **2000s–hoy:** Avances en cómputo, regularización y datos masivos impulsan el **aprendizaje profundo**.

Antecedentes históricos: La neurona de McCulloch-Pitts

El primer modelo matemático de neurona (1943)

Warren McCulloch y Walter Pitts propusieron el primer modelo computacional de neurona biológica:

$$y = H\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

donde H es la función escalón (Heaviside).

Antecedentes históricos: La neurona de McCulloch-Pitts

El primer modelo matemático de neurona (1943)

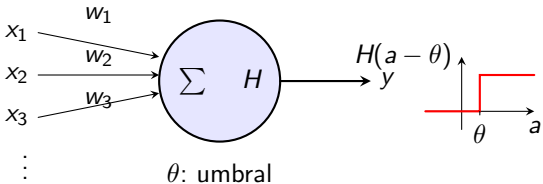
Warren McCulloch y Walter Pitts propusieron el primer modelo computacional de neurona biológica:

$$y = H\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

donde H es la función escalón (Heaviside).

- $x_i \in \{0, 1\}$: entradas binarias (señales de otras neuronas)
- $w_i \in \{-1, 0, 1\}$: pesos sinápticos (inhibitorio, nulo, excitatorio)
- θ: umbral de activación
- $y \in \{0, 1\}$: salida binaria

Antecedentes históricos: La neurona de McCulloch-Pitts



Capacidades y limitaciones de McCulloch-Pitts

Potencia computacional

- Puede implementar funciones booleanas: AND, OR, NOT
- Combinaciones pueden representar cualquier función booleana
- Fundamentos teóricos de las compuertas lógicas digitales

Capacidades y limitaciones de McCulloch-Pitts

Potencia computacional

- Puede implementar funciones booleanas: AND, OR, NOT
- Combinaciones pueden representar cualquier función booleana
- Fundamentos teóricos de las compuertas lógicas digitales

Limitaciones importantes

- **Pesos fijos**: No hay aprendizaje automático
- **Entradas binarias**: No maneja valores continuos
- **No adaptable**: Los parámetros se diseñan manualmente

Capacidades y limitaciones de McCulloch-Pitts

Limitaciones importantes

- **Pesos fijos**: No hay aprendizaje automático
- **Entradas binarias**: No maneja valores continuos
- **No adaptable**: Los parámetros se diseñan manualmente

Capacidades y limitaciones de McCulloch-Pitts

Limitaciones importantes

- **Pesos fijos**: No hay aprendizaje automático
- **Entradas binarias**: No maneja valores continuos
- **No adaptable**: Los parámetros se diseñan manualmente

Ejemplo: Compuerta AND

$$y = H(x_1 + x_2 - 1.5)$$

$$\text{Si } x_1 = 1, x_2 = 1 \Rightarrow 1 + 1 - 1.5 = 0.5 > 0 \Rightarrow y = 1$$

$$\text{Si } x_1 = 0, x_2 = 1 \Rightarrow 0 + 1 - 1.5 = -0.5 < 0 \Rightarrow y = 0$$

Del modelo estático al aprendizaje: El perceptrón

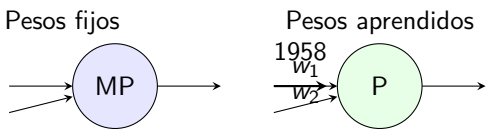
La innovación de Frank Rosenblatt (1958)

- **Extiende** McCulloch-Pitts con capacidad de aprendizaje
- **Pesos continuos**: $w_i \in \mathbb{R}$ (no solo $\{-1, 0, 1\}$)
- **Entradas continuas**: $x_i \in \mathbb{R}$ (no solo $\{0, 1\}$)
- **Regla de aprendizaje**: Actualización automática de pesos

Del modelo estático al aprendizaje: El perceptrón

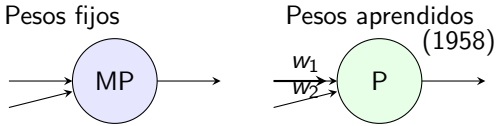
La innovación de Frank Rosenblatt (1958)

- **Extiende** McCulloch-Pitts con capacidad de aprendizaje
- **Pesos continuos**: $w_i \in \mathbb{R}$ (no solo $\{-1, 0, 1\}$)
- **Entradas continuas**: $x_i \in \mathbb{R}$ (no solo $\{0, 1\}$)
- **Regla de aprendizaje**: Actualización automática de pesos



Regla de aprendizaje:
$w \leftarrow w + \eta(z - y)x$

Del modelo estático al aprendizaje: El perceptrón



Regla de aprendizaje: $w \leftarrow w + \eta(z - y)x$

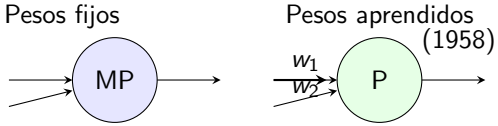
Componentes:

- w: pesos actuales
- η: tasa de aprendizaje
- z: valor verdadero
- y: predicción
- x: entrada

Ejemplo numérico:

$$w = [2, -1]$$
$$\eta = 0.1$$
$$x = [3, 2]$$
$$z = 1, y = 0$$
$$\text{Nuevo } w = [2.3, -0.8]$$

Del modelo estático al aprendizaje: El perceptrón



Regla de aprendizaje: $w \leftarrow w + \eta(z - y)x$

Componentes:

- w: pesos actuales
- η: tasa de aprendizaje
- z: valor verdadero
- y: predicción
- x: entrada

Ejemplo numérico:

$$w = [2, -1]$$
$$\eta = 0.1$$
$$x = [3, 2]$$
$$z = 1, y = 0$$
$$\text{Nuevo } w = [2.3, -0.8]$$

Transición fundamental

De **modelos diseñados manualmente** a **sistemas que aprenden de datos**

Motivación para el estudio actual

De los límites lineales al aprendizaje no lineal

El perceptrón simple, aunque inspirado en la biología, sólo puede representar fronteras lineales:

$$f(x) = \mathbf{1}\{ w^\top x + b > 0 \}$$

Por tanto, no puede aprender funciones como XOR.

Motivación para el estudio actual

De los límites lineales al aprendizaje no lineal

El perceptrón simple, aunque inspirado en la biología, sólo puede representar fronteras lineales:

$$f(x) = \mathbf{1}\{\, w^\top x + b > 0 \,\}$$

Por tanto, no puede aprender funciones como XOR.

La solución

Agregar una capa intermedia con activaciones no lineales:

$$h = g(W_1 x + b_1), \quad y = \sigma(W_2 h + b_2)$$

Ésta es la idea central de las redes neuronales modernas.

El modelo del perceptrón

Definición

$$y(x) = H(w^\top x + b), \qquad H(a) = \begin{cases} 1 & a > 0, \\ 0 & a \leq 0. \end{cases}$$

El modelo del perceptrón

Definición

$$y(x) = H(w^\top x + b), \qquad H(a) = \begin{cases} 1 & a > 0, \\ 0 & a \leq 0. \end{cases}$$

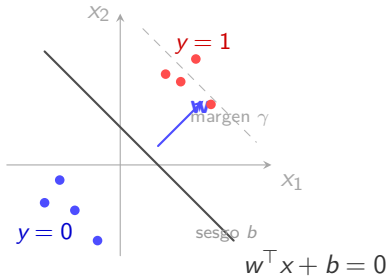
- $w \in \mathbb{R}^d$: vector normal al hiperplano de decisión.
- b: sesgo o desplazamiento.
- La ecuación $w^\top x + b = 0$ define la frontera lineal entre clases.

El modelo del perceptrón

Definición

$$y(x) = H(w^\top x + b), \qquad H(a) = \begin{cases} 1 & a > 0, \\ 0 & a \leq 0. \end{cases}$$

- $w \in \mathbb{R}^d$: vector normal al hiperplano de decisión.
- b: sesgo o desplazamiento.
- La ecuación $w^\top x + b = 0$ define la frontera lineal entre clases.



Intuición geométrica

- Cada observación x se proyecta sobre el vector normal w.
- Si $w^\top x + b > 0$, la muestra cae en el semiespacio positivo.
- El aprendizaje busca un w y b que separen las clases.

Regla de aprendizaje del perceptrón

Dado un ejemplo $(x^{(k)}, z^{(k)})$, se calcula la predicción $y^{(k)} = H(w^\top x^{(k)} + b)$.

Regla de aprendizaje del perceptrón

Dado un ejemplo $(x^{(k)}, z^{(k)})$, se calcula la predicción $y^{(k)} = H(w^\top x^{(k)} + b)$.

Actualización

$$w \leftarrow w + \eta(z^{(k)} - y^{(k)})x^{(k)}, \qquad b \leftarrow b + \eta(z^{(k)} - y^{(k)}),$$

donde $\eta > 0$ es la tasa de aprendizaje.

Regla de aprendizaje del perceptrón

Dado un ejemplo $(x^{(k)}, z^{(k)})$, se calcula la predicción $y^{(k)} = H(w^\top x^{(k)} + b)$.

Actualización

$$w \leftarrow w + \eta(z^{(k)} - y^{(k)})x^{(k)}, \qquad b \leftarrow b + \eta(z^{(k)} - y^{(k)}),$$

donde $\eta > 0$ es la tasa de aprendizaje.

Interpretación geométrica

Si el ejemplo está mal clasificado:

- Si $z^{(k)} = 1$ y $y^{(k)} = 0$: movemos w hacia $x^{(k)}$.
- Si $z^{(k)} = 0$ y $y^{(k)} = 1$: movemos w en dirección opuesta a $x^{(k)}$.

Teorema de convergencia

Enunciado clásico

Si los datos son **linealmente separables**, el algoritmo del perceptrón converge en un número finito de pasos.

Teorema de convergencia

Enunciado clásico

Si los datos son **linealmente separables**, el algoritmo del perceptrón converge en un número finito de pasos.

Idea de la demostración

1. Sea w^* el vector que separa con margen $\gamma > 0$ y $|w^*| = 1$.
2. Para cualquier error: $z_i(w^* \cdot x_i) \geq \gamma$.
3. Mostrar: $\langle w_t, w^* \rangle \geq t\eta\gamma, \qquad |w_t| \leq \sqrt{t}\,\eta R, \quad R = \max_i |x_i|$.
4. Aplicar desigualdad de Cauchy–Schwarz:

$$t\eta\gamma \leq |w_t| \leq \sqrt{t}\,\eta R$$

5. Concluir:

$$t \leq \left(\frac{R}{\gamma}\right)^2.$$

Cota por margen

Resultado

El número de actualizaciones está acotado por:

$$t_{\text{máx}} = O\left(\frac{1}{\gamma^2}\right).$$

Cota por margen

Resultado

El número de actualizaciones está acotado por:

$$t_{\text{máx}} = O\left(\frac{1}{\gamma^2}\right).$$

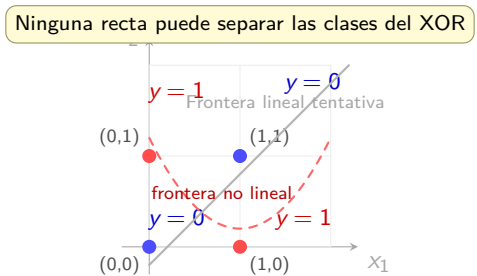
- Cuanto mayor el **margen** γ, más rápido converge el algoritmo.
- Márgenes pequeños implican aprendizaje más lento o inestable.

Limitaciones del perceptrón

- El perceptrón sólo puede aprender fronteras **lineales**.
- No puede resolver el clásico problema **XOR**.
- Esto motiva la extensión a redes con múltiples capas y activaciones no lineales.

Limitaciones del perceptrón

- El perceptrón sólo puede aprender fronteras **lineales**.
- No puede resolver el clásico problema **XOR**.
- Esto motiva la extensión a redes con múltiples capas y activaciones no lineales.

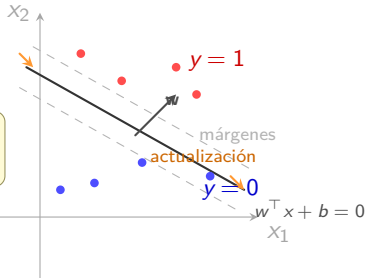


Demostración práctica

- Conjunto de datos 2D separable linealmente.
- Visualizar:
  - Frontera de decisión durante el aprendizaje.
  - Curva de errores por época.
- Resultado esperado:
  - Convergencia rápida si el margen γ es grande.
  - Lentitud o fallo si el conjunto no es separable.

Frontera de decisión del perceptrón

Ajuste iterativo hasta convergencia

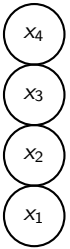


Redes multicapa y retropropagación

Pausa

De una capa a múltiples capas

$z^{(0)} = x$



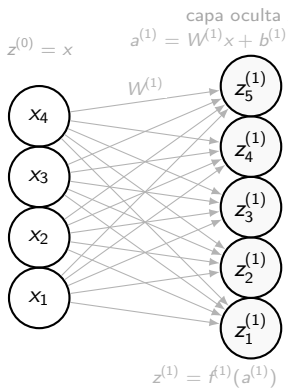
Idea clave

Cada capa aplica una transformación:

$$a^{(l)} = W^{(l)} z^{(l-1)} + b^{(l)}, \quad z^{(l)} = f^{(l)}(a^{(l)}).$$

La red completa es una composición $f^{(L)} \circ \cdots \circ f^{(1)}$.

De una capa a múltiples capas



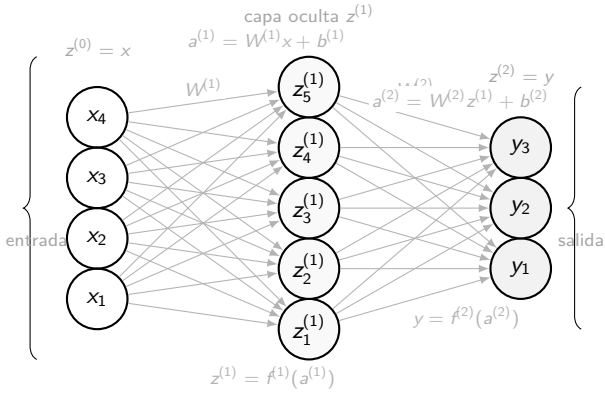
Idea clave

Cada capa aplica una transformación:

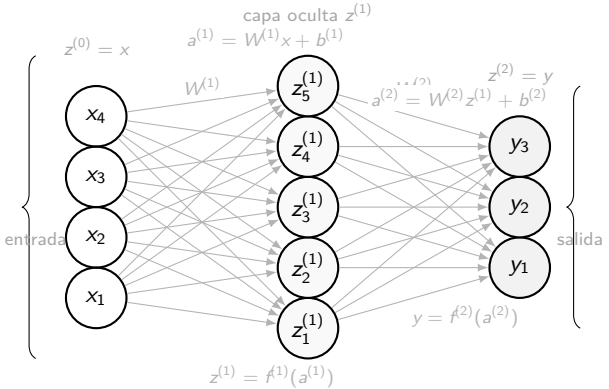
$$a^{(l)} = W^{(l)} z^{(l-1)} + b^{(l)}, \quad z^{(l)} = f^{(l)}(a^{(l)}).$$

La red completa es una composición $f^{(L)} \circ \cdots \circ f^{(1)}$.

De una capa a múltiples capas



De una capa a múltiples capas



Idea clave

Cada capa aplica una transformación:

$$a^{(l)} = W^{(l)}z^{(l-1)} + b^{(l)}, \quad z^{(l)} = f^{(l)}(a^{(l)}).$$

La red completa es una composición $f^{(L)} \circ \cdots \circ f^{(1)}$.

Propagación hacia adelante

$$a^{(1)} = W^{(1)}x + b^{(1)}, \qquad z^{(1)} = f^{(1)}(a^{(1)}),$$
$$a^{(2)} = W^{(2)}z^{(1)} + b^{(2)}, \qquad y = f^{(2)}(a^{(2)}).$$

Propagación hacia adelante

$$a^{(1)} = W^{(1)}x + b^{(1)}, \qquad z^{(1)} = f^{(1)}(a^{(1)}),$$
$$a^{(2)} = W^{(2)}z^{(1)} + b^{(2)}, \qquad y = f^{(2)}(a^{(2)}).$$

Símbolo	Dimensión	Significado
x	$d \times 1$	Entrada
$z^{(1)}$	$h \times 1$	Capa oculta
y	$k \times 1$	Salida
$W^{(l)}$	$n_l \times n_{l-1}$	Pesos capa l
$b^{(l)}$	$n_l \times 1$	Bias capa l

Función de pérdida cuadrática

$$E = \frac{1}{2} \sum_n \|y_n - t_n\|^2.$$

Función de pérdida cuadrática

$$E = \frac{1}{2} \sum_n \|y_n - t_n\|^2.$$

El objetivo del entrenamiento es minimizar $E(W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)})$.

Gradiente en la capa de salida

$$\delta^{(2)} = (y - t) \odot f'^{(2)}(a^{(2)}),$$
$$\frac{\partial E}{\partial W^{(2)}} = \delta^{(2)}(z^{(1)})^{\top}.$$

Gradiente en la capa de salida

$$\delta^{(2)} = (y - t) \odot f'^{(2)}(a^{(2)}),$$
$$\frac{\partial E}{\partial W^{(2)}} = \delta^{(2)} (z^{(1)})^{\top}.$$

Intuición: el error se propaga multiplicando por la derivada de la activación.

Propagación del error hacia la capa oculta

$$\delta^{(1)} = (W^{(2)})^\top \delta^{(2)} \odot f'^{(1)}(a^{(1)}),$$

$$\frac{\partial E}{\partial W^{(1)}} = \delta^{(1)} x^\top.$$

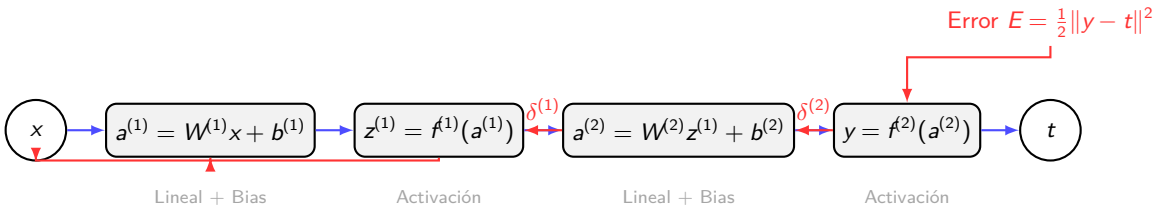
Propagación del error hacia la capa oculta

$$\delta^{(1)} = (W^{(2)})^\top \delta^{(2)} \odot f'^{(1)}(a^{(1)}),$$

$$\frac{\partial E}{\partial W^{(1)}} = \delta^{(1)} x^\top.$$

Estructura recurrente: $\delta^{(l)}$ depende de $\delta^{(l+1)}$.

Backprop como aplicación de la regla de la cadena



Cada nodo: aplica la regla de la cadena localmente.
La retropropagación sigue el orden inverso del *forward pass*.

Forma general compacta

$$\delta^{(L)} = \nabla_{a^{(L)}} E = (y - t) \odot f'^{(L)}(a^{(L)}),$$
$$\delta^{(l)} = (W^{(l+1)})^\top \delta^{(l+1)} \odot f'^{(l)}(a^{(l)}),$$
$$\frac{\partial E}{\partial W^{(l)}} = \delta^{(l)} (z^{(l-1)})^\top.$$

Forma general compacta

$$\delta^{(L)} = \nabla_{a^{(L)}} E = (y - t) \odot f'^{(L)}(a^{(L)}),$$
$$\delta^{(l)} = (W^{(l+1)})^\top \delta^{(l+1)} \odot f'^{(l)}(a^{(l)}),$$
$$\frac{\partial E}{\partial W^{(l)}} = \delta^{(l)} (z^{(l-1)})^\top.$$

Backprop = *regla de la cadena implementada en paralelo.*

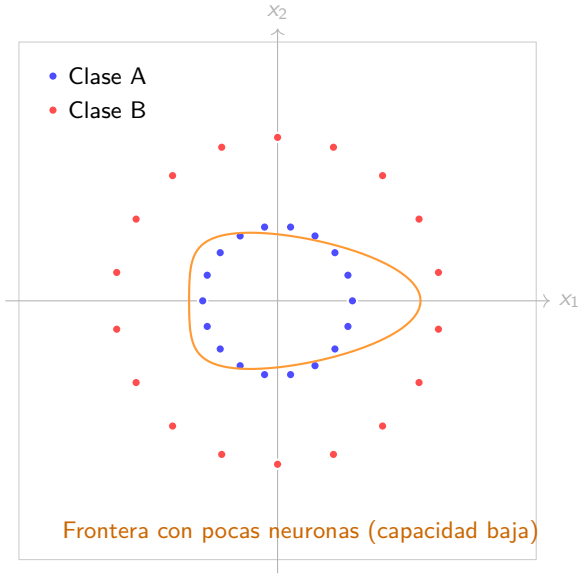
Optimización de redes

- **SGD / Mini-batch:** actualización incremental

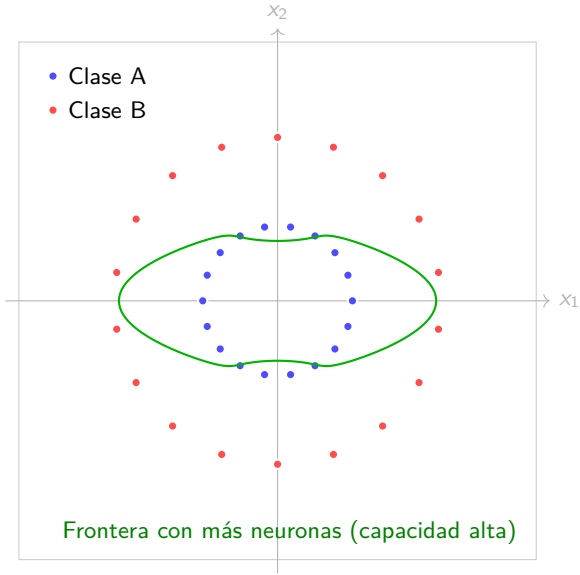
$$W \leftarrow W - \eta \frac{\partial E}{\partial W}.$$

- **Tasa de aprendizaje** η: controla la magnitud del paso.
- **Inicialización:** He, Xavier, etc. (evitar saturación).
- **No convexidad:** múltiples mínimos locales; importancia del *early stopping*.

Demostración: capacidad del MLP



Demostración: capacidad del MLP



Pausa

Funciones de activación

Motivación

Las funciones de activación introducen **no linealidad** en las redes neuronales. Sin ellas, una red profunda sería equivalente a una transformación lineal.

Funciones de activación

Motivación

Las funciones de activación introducen **no linealidad** en las redes neuronales. Sin ellas, una red profunda sería equivalente a una transformación lineal.

- Controlan cómo las neuronas transforman la suma ponderada de sus entradas.
- Afectan el **entrenamiento, la velocidad de convergencia y la estabilidad del gradiente**.

Funciones de activación

Motivación

Las funciones de activación introducen **no linealidad** en las redes neuronales. Sin ellas, una red profunda sería equivalente a una transformación lineal.

- Controlan cómo las neuronas transforman la suma ponderada de sus entradas.
- Afectan el **entrenamiento, la velocidad de convergencia y la estabilidad del gradiente**.

Ejemplos clásicos

Sigmoide, tanh, ReLU, Softmax.

Funciones de activación clásicas

Función	Expresión	Rango	Comentario
Lineal	$f(a) = a$	$\mathbb{R}$	Sin no linealidad (modelo lineal).
Sigmoide	$f(a) = \frac{1}{1+e^{-a}}$	$(0,1)$	Suave, pero satura en extremos.
Tanh	$f(a) = \tanh(a)$	$(-1,1)$	Centrada, más simétrica que la sigmoide.
ReLU	$f(a) = \max(0,a)$	$[0,\infty)$	No lineal a trozos, evita saturación positiva.
Leaky ReLU	$f(a) = \max(0.01a, a)$	$\mathbb{R}$	Evita "neuronas muertas".
Softmax	$f(a_i) = \frac{e^{a_i}}{\sum_j e^{a_j}}$	$(0,1)$	Capa de salida multiclase (probabilidades).

Funciones de activación clásicas

Función	Expresión	Rango	Comentario
Lineal	$f(a) = a$	$\mathbb{R}$	Sin no linealidad (modelo lineal).
Sigmoide	$f(a) = \dfrac{1}{1+e^{-a}}$	$(0,1)$	Suave, pero satura en extremos.
Tanh	$f(a) = \tanh(a)$	$(-1,1)$	Centrada, más simétrica que la sigmoide.
ReLU	$f(a) = \max(0,a)$	$[0,\infty)$	No lineal a trozos, evita saturación positiva.
Leaky ReLU	$f(a) = \max(0.01a, a)$	$\mathbb{R}$	Evita "neuronas muertas".
Softmax	$f(a_i) = \dfrac{e^{a_i}}{\sum_j e^{a_j}}$	$(0,1)$	Capa de salida multiclase (probabilidades).

Idea general

Cada activación transforma la entrada lineal $a = Wx + b$ en una salida **no lineal**, lo que permite que la red represente funciones más complejas.

¿De verdad necesitamos activaciones?

Preguntas para arrancar

- Si una red profunda *no* tuviera activaciones, ¿qué clase de funciones podría aproximar?
- ¿Podemos "aprender no linealidad" solo con muchas capas lineales?
- Si **Softmax** ya da probabilidades, ¿por qué no usar algo parecido en ocultas?

¿De verdad necesitamos activaciones?

Preguntas para arrancar

- Si una red profunda *no* tuviera activaciones, ¿qué clase de funciones podría aproximar?
- ¿Podemos "aprender no linealidad" solo con muchas capas lineales?
- Si **Softmax** ya da probabilidades, ¿por qué no usar algo parecido en ocultas?

Hipótesis a discutir

"Una red sin activaciones profundas es tan potente como una sola capa."

Hecho clave: Composición de transformaciones lineales

Composición lineal

Sea $x \in \mathbb{R}^d$ y una red de L capas *sin* activación:

$$z^{(1)} = W^{(1)}x + b^{(1)}, \quad z^{(2)} = W^{(2)}z^{(1)} + b^{(2)}, \ \ldots, \ z^{(L)} = W^{(L)}z^{(L-1)} + b^{(L)}.$$

Hecho clave: Composición de transformaciones lineales

Composición lineal

Sea $x \in \mathbb{R}^d$ y una red de L capas *sin* activación:

$$z^{(1)} = W^{(1)}x + b^{(1)}, \quad z^{(2)} = W^{(2)}z^{(1)} + b^{(2)}, \ \ldots, \ z^{(L)} = W^{(L)}z^{(L-1)} + b^{(L)}.$$

Por expansión,

$$z^{(L)} = \underbrace{\left(W^{(L)} \cdots W^{(1)}\right)}_{=:W'} x + \underbrace{\sum_{\ell=1}^{L} \left(W^{(L)} \cdots W^{(\ell+1)}\right) b^{(\ell)}}_{=:b'}.$$

Hecho clave: Composición de transformaciones lineales

Composición lineal

Sea $x \in \mathbb{R}^d$ y una red de L capas *sin* activación:

$$z^{(1)} = W^{(1)}x + b^{(1)}, \quad z^{(2)} = W^{(2)}z^{(1)} + b^{(2)}, \ \ldots, \ z^{(L)} = W^{(L)}z^{(L-1)} + b^{(L)}.$$

Por expansión,

$$z^{(L)} = \underbrace{\left(W^{(L)} \cdots W^{(1)}\right)}_{=:W'} x + \underbrace{\sum_{\ell=1}^{L} \left(W^{(L)} \cdots W^{(\ell+1)}\right) b^{(\ell)}}_{=:b'}.$$

Conclusión

La red completa equivale a una **única** transformación afín:

$$f(x) = W'x + b'.$$

Sin activaciones, la profundidad *no* añade capacidad de no linealidad.

Hecho clave (con activaciones): Composición no lineal

Red con activación por capa

Sea $x \in \mathbb{R}^d$ y una red de L capas con activaciones $f^{(l)}$:

$$
\begin{aligned}
a^{(1)} &= W^{(1)}x + b^{(1)}, & z^{(1)} &= f^{(1)}(a^{(1)}),\\
a^{(2)} &= W^{(2)}z^{(1)} + b^{(2)}, & z^{(2)} &= f^{(2)}(a^{(2)}),\\
&\vdots & &\vdots\\
a^{(L)} &= W^{(L)}z^{(L-1)} + b^{(L)}, & z^{(L)} &= f^{(L)}(a^{(L)}).
\end{aligned}
$$

Hecho clave (con activaciones): Composición no lineal

Red con activación por capa

Sea $x \in \mathbb{R}^d$ y una red de L capas con activaciones $f^{(l)}$:

$$\begin{aligned} a^{(1)} &= W^{(1)}x + b^{(1)}, & z^{(1)} &= f^{(1)}(a^{(1)}), \\ a^{(2)} &= W^{(2)}z^{(1)} + b^{(2)}, & z^{(2)} &= f^{(2)}(a^{(2)}), \\ &\vdots & &\vdots \\ a^{(L)} &= W^{(L)}z^{(L-1)} + b^{(L)}, & z^{(L)} &= f^{(L)}(a^{(L)}). \end{aligned}$$

Composición

La salida se escribe como composición anidada:

$$f(x) \;=\; \big(f^{(L)} \circ \mathcal{A}^{(L)}\big) \circ \cdots \circ \big(f^{(1)} \circ \mathcal{A}^{(1)}\big)(x), \quad \mathcal{A}^{(l)}(u) = W^{(l)}u + b^{(l)}.$$

En general no colapsa a una sola transformación afín, salvo que todas las $f^{(l)}$ sean lineales.

Caso ReLU: afín a trozos vía matrices de compuerta

Idea

Para ReLU, $f(u) = \max(0, u)$, podemos escribir

$$z^{(l)} \;=\; \mathrm{ReLU}\big(a^{(l)}\big) \;=\; D^{(l)}(x)\, a^{(l)} \;=\; D^{(l)}(x)\, \big(W^{(l)} z^{(l-1)} + b^{(l)}\big),$$

donde $D^{(l)}(x) = \mathrm{diag}\big(\mathbf{1}[a_i^{(l)}(x) > 0]\big)$ depende de x (qué neuronas "activas" hay).

Caso ReLU: afín a trozos vía matrices de compuerta

Idea

Para ReLU, $f(u) = \max(0, u)$, podemos escribir

$$z^{(l)} \;=\; \mathrm{ReLU}\big(a^{(l)}\big) \;=\; D^{(l)}(x)\, a^{(l)} \;=\; D^{(l)}(x)\,\big(W^{(l)} z^{(l-1)} + b^{(l)}\big),$$

donde $D^{(l)}(x) = \mathrm{diag}\big(\mathbf{1}[a_i^{(l)}(x) > 0]\big)$ depende de x (qué neuronas "activas" hay).

Desenrollando (2 capas para intuición)

$$z^{(1)} = D^{(1)}(x)\big(W^{(1)}x + b^{(1)}\big),$$

$$z^{(2)} = D^{(2)}(x)\Big(W^{(2)}\, z^{(1)} + b^{(2)}\Big) = D^{(2)}(x)\Big(W^{(2)} D^{(1)}(x)\, W^{(1)}x \;+\; W^{(2)} D^{(1)}(x) b^{(1)} \;+\; b^{(2)}\Big).$$

Para un *patrón de activación fijo* (i.e., $D^{(1)}, \ldots, D^{(L)}$ constantes), esto es $f(x) \;=\; A\,x + c$ (afín).

Caso ReLU: afín a trozos vía matrices de compuerta

Idea

Para ReLU, $f(u) = \max(0, u)$, podemos escribir

$$z^{(l)} \;=\; \mathrm{ReLU}\big(a^{(l)}\big) \;=\; D^{(l)}(x)\, a^{(l)} \;=\; D^{(l)}(x)\,\big(W^{(l)} z^{(l-1)} + b^{(l)}\big),$$

donde $D^{(l)}(x) = \mathrm{diag}\big(\mathbf{1}[a_i^{(l)}(x) > 0]\big)$ depende de x (qué neuronas "activas" hay).

Desenrollando (2 capas para intuición)

$$z^{(1)} = D^{(1)}(x)\big(W^{(1)}x + b^{(1)}\big),$$

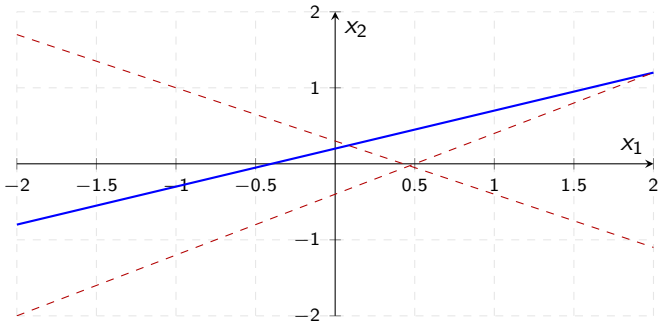
$$z^{(2)} = D^{(2)}(x)\Big(W^{(2)}\, z^{(1)} + b^{(2)}\Big) = D^{(2)}(x)\Big(W^{(2)} D^{(1)}(x)\, W^{(1)}x \;+\; W^{(2)} D^{(1)}(x) b^{(1)} \;+\; b^{(2)}\Big).$$

Para un *patrón de activación fijo* (i.e., $D^{(1)}, \ldots, D^{(L)}$ constantes), esto es $f(x) \;=\; A\,x + c$ (afín).

Conclusión

La red ReLU define una función **afín a trozos**: el espacio de entrada se particiona en regiones (según los signos de $a^{(l)}$), y en cada región $f(x)$ es afín con matrices efectivas *dependientes del patrón* $D^{(1)}, \ldots, D^{(L)}$.

Intuición visual: lineal vs. lineal-a-trozos



(A) sin activación: una frontera recta. (B) con ReLU: combinación lineal a trozos con múltiples regiones.

Activaciones suaves (sigmoide, tanh): no linealidad composicional

Forma composicional (2 capas como ejemplo)

$$f(x) = f^{(2)}\Big(W^{(2)} f^{(1)}(W^{(1)}x + b^{(1)}) + b^{(2)}\Big),$$

que no se puede reducir a $W'x + b'$ salvo $f^{(1)}, f^{(2)}$ lineales.

Activaciones suaves (sigmoide, tanh): no linealidad composicional

Forma composicional (2 capas como ejemplo)

$$f(x) \;=\; f^{(2)}\Big(W^{(2)}\, f^{(1)}\big(W^{(1)}x + b^{(1)}\big) + b^{(2)}\Big),$$

que no se puede reducir a $W'x + b'$ salvo $f^{(1)}, f^{(2)}$ lineales.

Derivadas y backprop (intuición local)

El Jacobiano se compone en cadena:

$$J_f(x) \;=\; \underbrace{D_f^{(L)}(a^{(L)})\, W^{(L)}}_{\text{capa } L} \cdots \underbrace{D_f^{(1)}(a^{(1)})\, W^{(1)}}_{\text{capa } 1},$$

donde $D_f^{(l)}(a^{(l)}) = \mathrm{diag}\big((f^{(l)})'(a^{(l)})\big)$. La **no linealidad** entra tanto por la forma de $f^{(l)}$ como por su **derivada** en cada punto.

Activaciones suaves (sigmoide, tanh): no linealidad composicional

Forma composicional (2 capas como ejemplo)

$$f(x) \;=\; f^{(2)}\Big(W^{(2)}\, f^{(1)}\big(W^{(1)}x + b^{(1)}\big) + b^{(2)}\Big),$$

que no se puede reducir a $W'x + b'$ salvo $f^{(1)}, f^{(2)}$ lineales.

Derivadas y backprop (intuición local)

El Jacobiano se compone en cadena:

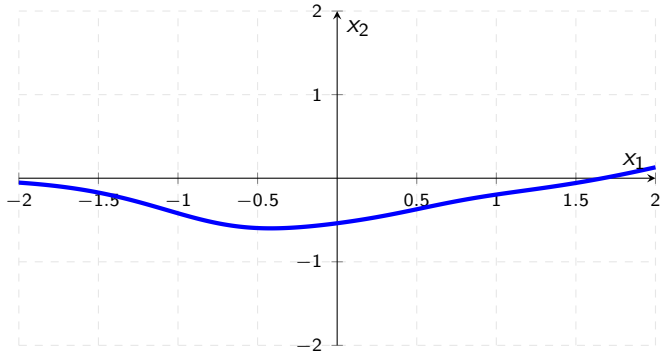
$$J_f(x) \;=\; \underbrace{D_f^{(L)}(a^{(L)})\, W^{(L)}}_{\text{capa } L} \cdots \underbrace{D_f^{(1)}(a^{(1)})\, W^{(1)}}_{\text{capa } 1},$$

donde $D_f^{(l)}(a^{(l)}) = \mathrm{diag}\big((f^{(l)})'(a^{(l)})\big)$. La **no linealidad** entra tanto por la forma de $f^{(l)}$ como por su **derivada** en cada punto.

Mensaje

Con sigmoide/tanh, la red es **suave y no lineal** por composición; con ReLU es **afín a trozos**. En ambos casos, la profundidad aporta capacidad que *no* puede colapsarse a una sola capa afín.

Intuición visual: transición suave entre clasificadores lineales



La activación **suave** "mezcla" las fronteras lineales locales (rojas), produciendo una frontera **curva continua** (azul) sin quiebres abruptos.

Implicaciones prácticas

¿Qué perdemos sin activación?

- Capacidad de **aproximar funciones no lineales**.
- **Interacciones** complejas entre variables.
- **Decisiones no lineales** (fronteras de clasificación curvas).

Ejemplos inviables

- XOR (frontera no lineal).
- Sistemas con umbrales/saturaciones.
- Dinámicas multimodales o con "mesetas".

Implicaciones prácticas

¿Qué perdemos sin activación?

- Capacidad de **aproximar funciones no lineales**.
- **Interacciones** complejas entre variables.
- **Decisiones no lineales** (fronteras de clasificación curvas).

Ejemplos inviables

- XOR (frontera no lineal).
- Sistemas con umbrales/saturaciones.
- Dinámicas multimodales o con "mesetas".

Mensaje

Las activaciones introducen **no linealidad controlada**, habilitando aproximación universal y representaciones jerárquicas.

Softmax: ¿por qué salida y no ocultas?

Rol de Softmax

- En salida multiclase: produce una **distribución de probabilidad** sobre clases.
- Su derivada empareja bien con la **entropía cruzada** en entrenamiento.

Softmax: ¿por qué salida y no ocultas?

Rol de Softmax

- En salida multiclase: produce una **distribución de probabilidad** sobre clases.
- Su derivada empareja bien con la **entropía cruzada** en entrenamiento.

Por qué no en ocultas

- Forzar normalización por capa (sumar 1) restringe la **expresividad** intermedia.
- Puede inducir **competencia prematura** entre neuronas ocultas.
- ReLU/tanh permiten representaciones **más flexibles** antes de la decisión final.

Cierre: la pregunta resuelta

Respuesta breve

Sí: necesitamos activaciones para escapar del mundo lineal.

Cierre: la pregunta resuelta

Respuesta breve

Sí: necesitamos activaciones para escapar del mundo lineal.

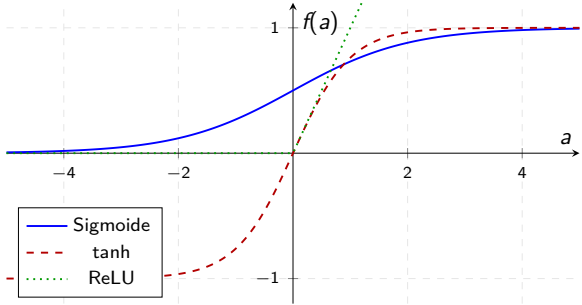
Claves para lo que sigue

- Activaciones $\Rightarrow$ no linealidad & composicionalidad.
- Elección de activación afecta **optimización** (gradientes) y **generalización**.
- En salidas: **sigmoide** (binaria) / **softmax** (multiclase).

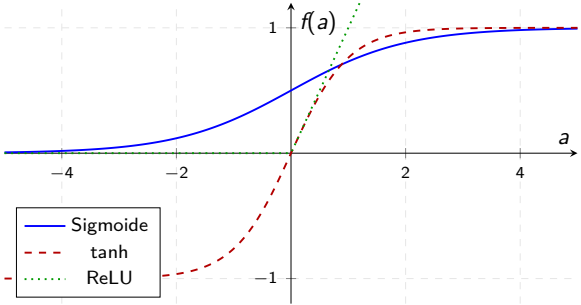
Comparación de funciones de activación

Función	**Expresión** $f(a)$	**Derivada** $f'(a)$	**Rango**
Sigmoide	$\dfrac{1}{1+e^{-a}}$	$f(a)\,[1-f(a)]$	$(0,1)$
tanh	$\dfrac{e^{a}-e^{-a}}{e^{a}+e^{-a}}$	$1-f^{2}(a)$	$(-1,1)$
ReLU	$\max(0,a)$	$\mathbf{1}[a>0]$	$[0,\infty)$
Leaky ReLU	$\max(0.01a,a)$	$\begin{cases}1, & a>0\\ 0.01, & a<0\end{cases}$	$\mathbb{R}$
Softmax	$\dfrac{e^{a_i}}{\sum_j e^{a_j}}$	—	$(0,1)$

Curvas de activación



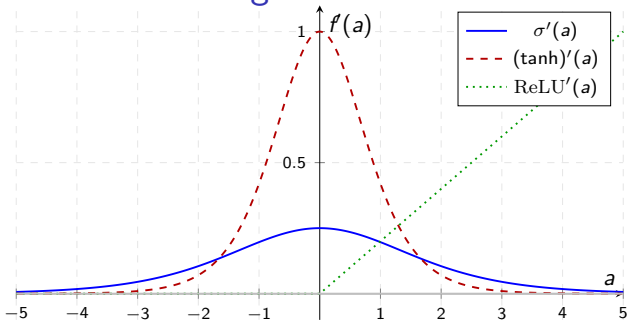
Curvas de activación



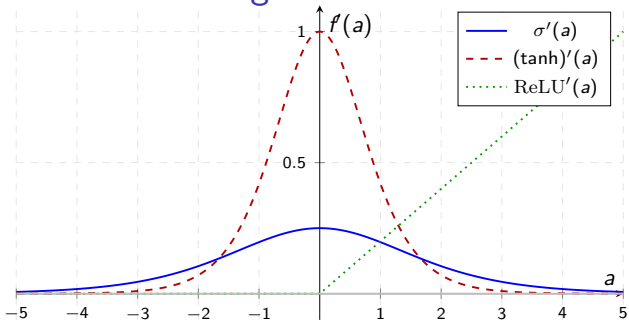
Observaciones

- Sigmoide y tanh son suaves pero saturan en los extremos.
- ReLU es más simple y evita saturación positiva.
- Variantes (Leaky ReLU, ELU, Swish) suavizan el gradiente negativo.

Derivadas y desvanecimiento del gradiente



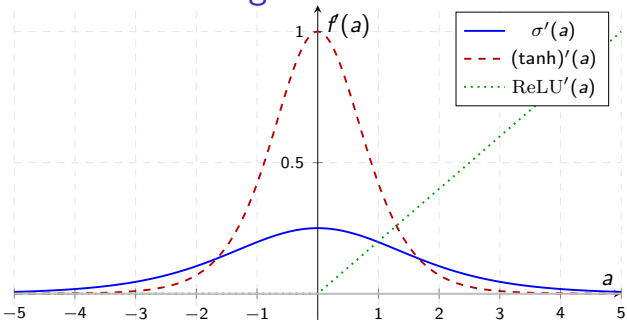
Derivadas y desvanecimiento del gradiente



Problema

En capas profundas, las funciones saturadas (sigmoide, tanh) provocan **gradientes cercanos a cero**, dificultando el aprendizaje.

Derivadas y desvanecimiento del gradiente



Problema

En capas profundas, las funciones saturadas (sigmoide, tanh) provocan **gradientes cercanos a cero**, dificultando el aprendizaje.

Soluciones prácticas

- Usar ReLU o variantes (Leaky ReLU, ELU, Swish).
- Normalizar/estandarizar entradas (evita caer en zonas saturadas)

Consejos prácticos

- **Ocultas:** ReLU o Leaky ReLU suelen funcionar mejor.
- **Salida binaria:** Sigmoide.
- **Salida multiclase:** Softmax.

Consejos prácticos

- **Ocultas:** ReLU o Leaky ReLU suelen funcionar mejor.
- **Salida binaria:** Sigmoide.
- **Salida multiclase:** Softmax.

En resumen

$$\text{activación} = \text{no linealidad} \Rightarrow \text{capacidad de aproximar funciones complejas.}$$

Pausa

Pausa

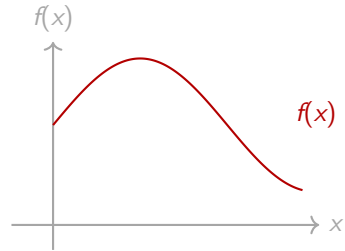
¿Qué tan potentes son las redes neuronales?

- Hasta ahora hemos visto cómo una red combina transformaciones lineales y funciones de activación.
- Pero surge una pregunta natural:

Pregunta central

¿Puede una red neuronal aproximar cualquier función?

- Si la respuesta es sí, ¿qué condiciones necesita?
- ¿Cuántas neuronas o capas son suficientes?

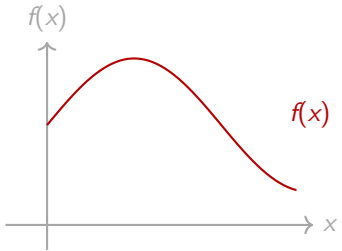


¿Qué tan potentes son las redes neuronales?

- Hasta ahora hemos visto cómo una red combina transformaciones lineales y funciones de activación.
- Pero surge una pregunta natural:

Pregunta central

¿Puede una red neuronal aproximar cualquier función?



- Si la respuesta es sí, ¿qué condiciones necesita?
- ¿Cuántas neuronas o capas son suficientes?

Motivación

El teorema de **Cybenko (1989)** responde afirmativamente: una red con una sola capa oculta y activación adecuada puede aproximar cualquier función continua.

Teorema de aproximación universal

Enunciado informal

Sea $f\colon \mathbb{R}^n \to \mathbb{R}$ una función continua en un conjunto compacto $K \subset \mathbb{R}^n$. Entonces, para toda $\varepsilon > 0$, existe una red neuronal con:

$$z(x) = \sum_{j=1}^{m} \alpha_j \, \sigma(w_j^\top x + b_j)$$

tal que

$$|z(x) - f(x)| < \varepsilon, \quad \forall x \in K,$$

donde σ es una función de activación continua, sigmoidal y no polinómica.

Teorema de aproximación universal

Enunciado informal

Sea $f: \mathbb{R}^n \to \mathbb{R}$ una función continua en un conjunto compacto $K \subset \mathbb{R}^n$. Entonces, para toda $\varepsilon > 0$, existe una red neuronal con:

$$z(x) = \sum_{j=1}^{m} \alpha_j \, \sigma(w_j^\top x + b_j)$$

tal que

$$|z(x) - f(x)| < \varepsilon, \quad \forall x \in K,$$

donde σ es una función de activación continua, sigmoidal y no polinómica.

- Este resultado se debe a **Cybenko (1989)** y fue extendido por **Hornik (1991)**.
- La red necesita sólo una capa oculta.
- No establece cuántas neuronas m son necesarias.

Teorema de Cybenko (1989) — versión formal

Teorema

Sea $\sigma : \mathbb{R} \to \mathbb{R}$ una función sigmoidal continua. Entonces el conjunto de funciones

$$\mathcal{H} = \left\{ x \mapsto \sum_{j=1}^{m} \alpha_j \, \sigma(w_j^\top x + b_j) \;:\; m \in \mathbb{N},\ \alpha_j \in \mathbb{R},\ w_j \in \mathbb{R}^n,\ b_j \in \mathbb{R} \right\}$$

es *denso en* $C([0,1]^n)$ con la norma del supremo.

Teorema de Cybenko (1989) — versión formal

Teorema

Sea $\sigma : \mathbb{R} \to \mathbb{R}$ una función sigmoidal continua. Entonces el conjunto de funciones

$$\mathcal{H} = \left\{ x \mapsto \sum_{j=1}^{m} \alpha_j \, \sigma(w_j^\top x + b_j) \;:\; m \in \mathbb{N},\ \alpha_j \in \mathbb{R},\ w_j \in \mathbb{R}^n,\ b_j \in \mathbb{R} \right\}$$

es *denso en* $C([0,1]^n)$ con la norma del supremo.

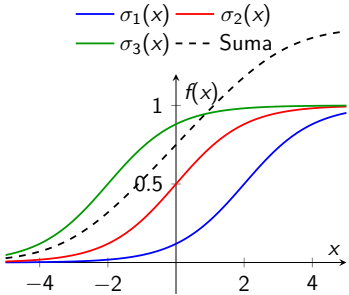
- Densidad $\Rightarrow$ toda función continua puede aproximarse arbitrariamente bien.
- La demostración usa resultados de análisis funcional y el teorema de Hahn–Banach.

Intuición geométrica: suma de bases no lineales

- Cada neurona define una “curva sigmoidal”.
- Combinando muchas, se puede aproximar formas arbitrarias.
- La idea es similar a la aproximación por series de Fourier o polinomios.

Intuición geométrica: suma de bases no lineales

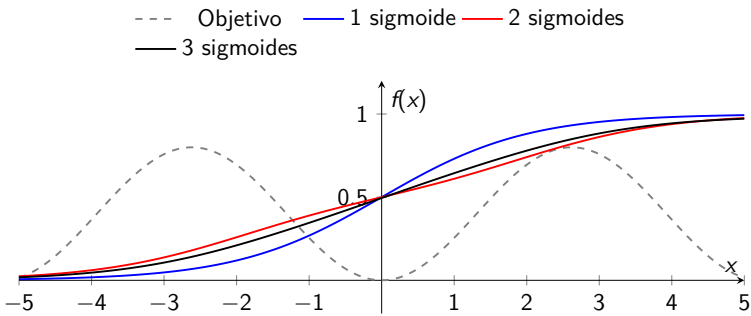
- Cada neurona define una “curva sigmoidal”.
- Combinando muchas, se puede aproximar formas arbitrarias.
- La idea es similar a la aproximación por series de Fourier o polinomios.



Clave conceptual

El poder de la red proviene de la no linealidad y la combinación lineal de las salidas ocultas.

Construyendo una aproximación



Alcance y limitaciones

Importante distinguir

Existencia: garantiza que tal red existe.

Entrenabilidad: no asegura que el algoritmo de entrenamiento la encuentre.

Generalización: no dice nada sobre el comportamiento fuera del conjunto de entrenamiento.

Alcance y limitaciones

Importante distinguir

Existencia: garantiza que tal red existe.

Entrenabilidad: no asegura que el algoritmo de entrenamiento la encuentre.

Generalización: no dice nada sobre el comportamiento fuera del conjunto de entrenamiento.

- No ofrece un límite superior sobre el número mínimo de neuronas.
- Profundidad puede mejorar *eficiencia representacional* (menor número de nodos).
- Para activaciones como ReLU, se han probado resultados equivalentes (Lu et al., 2017).

Bibliografía y lecturas complementarias I

G. Cybenko (1989).
Approximation by superpositions of a sigmoidal function.
Mathematics of Control, Signals, and Systems, 2(4):303–314.

K. Hornik, M. Stinchcombe, H. White (1989).
Multilayer feedforward networks are universal approximators.
Neural Networks, 2(5):359–366.

K. Hornik (1991).
Approximation capabilities of multilayer feedforward networks.
Neural Networks, 4(2):251–257.

A. R. Barron (1993).
Universal approximation bounds for superpositions of a sigmoidal function.
IEEE Transactions on Information Theory, 39(3):930–945.

Bibliografía y lecturas complementarias II

A. Pinkus (1999).
Approximation theory of the MLP model in neural networks.
Acta Numerica, 8:143–195.

G. Montúfar, R. Pascanu, K. Cho, Y. Bengio (2014).
On the number of linear regions of deep neural networks.
Advances in Neural Information Processing Systems (NeurIPS).

M. Telgarsky (2016).
Benefits of depth in neural networks.
Proceedings of the 29th Annual Conference on Learning Theory (COLT).

D. Yarotsky (2017).
Error bounds for approximations with deep ReLU networks.
Neural Networks, 94:103–114.

Bibliografía y lecturas complementarias III

Z. Lu, H. Pu, F. Wang, Z. Hu, L. Wang (2017).
The expressive power of neural networks: A view from the width.
Advances in Neural Information Processing Systems (NeurIPS 30).

H. N. Mhaskar, T. Poggio (2016).
Deep vs. shallow networks: An approximation theory perspective.
Analysis and Applications, 14(6):829–848.

I. Goodfellow, Y. Bengio, A. Courville (2016).
Deep Learning.
MIT Press.

C. M. Bishop (2006).
Pattern Recognition and Machine Learning.
Springer.

Bibliografía y lecturas complementarias IV


T. Hastie, R. Tibshirani, J. Friedman (2009).
The Elements of Statistical Learning (2nd ed.).
Springer.


K. P. Murphy (2022).
Probabilistic Machine Learning: An Introduction.
MIT Press.


S. Shalev-Shwartz, S. Ben-David (2014).
Understanding Machine Learning: From Theory to Algorithms.
Cambridge University Press.

Resumen práctico I: Preparación de datos

Técnica	Cuándo usarla / propósito
Limpieza de datos	Cuando existen valores faltantes, inconsistencias o errores en la captura.
Imputación	Cuando no se pueden eliminar observaciones; usar media, regresión o modelos bayesianos.
Codificación (One-Hot, Label)	Para transformar variables categóricas en formato numérico.
Escalamiento (z-score, min–max)	Antes de aplicar métodos sensibles a magnitud (p. ej. k-NN, PCA, redes neuronales).
Visualización exploratoria	Para detectar outliers, patrones o relaciones entre variables.

Resumen práctico I: Preparación de datos

Técnica	Cuándo usarla / propósito
Limpieza de datos	Cuando existen valores faltantes, inconsistencias o errores en la captura.
Imputación	Cuando no se pueden eliminar observaciones; usar media, regresión o modelos bayesianos.
Codificación (One-Hot, Label)	Para transformar variables categóricas en formato numérico.
Escalamiento (z-score, min–max)	Antes de aplicar métodos sensibles a magnitud (p. ej. k-NN, PCA, redes neuronales).
Visualización exploratoria	Para detectar outliers, patrones o relaciones entre variables.

Idea clave

Una buena preparación de los datos suele mejorar más el desempeño que cambiar de modelo.

Resumen práctico II: Aprendizaje supervisado

Técnica	Cuándo usarla / propósito
Regresión lineal / logística	Relaciones simples o fácilmente interpretables entre variables.
LDA / QDA	Cuando las clases son aproximadamente normales y las fronteras de decisión son elípticas.
k-vecinos más cercanos (k-NN)	Cuando se desea un modelo no paramétrico simple y los datos están bien escalados.
Árboles de decisión	Buena interpretación, manejo de interacciones y variables categóricas.
Bosques aleatorios	Para obtener mayor precisión y estabilidad; robustos al ruido.

Resumen práctico II: Aprendizaje supervisado

Técnica	Cuándo usarla / propósito
Regresión lineal / logística	Relaciones simples o fácilmente interpretables entre variables.
LDA / QDA	Cuando las clases son aproximadamente normales y las fronteras de decisión son elípticas.
k-vecinos más cercanos (k-NN)	Cuando se desea un modelo no paramétrico simple y los datos están bien escalados.
Árboles de decisión	Buena interpretación, manejo de interacciones y variables categóricas.
Bosques aleatorios	Para obtener mayor precisión y estabilidad; robustos al ruido.

Consejo

Verifica siempre la métrica adecuada: RMSE, accuracy, AUC, F1, según el tipo de problema.

Resumen práctico III: Aprendizaje no supervisado

Técnica	Cuándo usarla / propósito
k-medias	Cuando se sospecha que existen grupos esféricos bien separados.
Clustering jerárquico	Cuando se busca una jerarquía de similitudes o dendrogramas.
Clustering espectral	Cuando la estructura de los datos no es lineal o los grupos no son convexos.
EM y mezclas gaussianas	Cuando los datos provienen de subpoblaciones con distribución probabilística conocida.
PCA, SVD, NMF	Para reducir dimensionalidad, eliminar ruido o visualizar datos en 2–3D.

Resumen práctico III: Aprendizaje no supervisado

Técnica	Cuándo usarla / propósito
k-medias	Cuando se sospecha que existen grupos esféricos bien separados.
Clustering jerárquico	Cuando se busca una jerarquía de similitudes o dendrogramas.
Clustering espectral	Cuando la estructura de los datos no es lineal o los grupos no son convexos.
EM y mezclas gaussianas	Cuando los datos provienen de subpoblaciones con distribución probabilística conocida.
PCA, SVD, NMF	Para reducir dimensionalidad, eliminar ruido o visualizar datos en 2–3D.

Idea clave

El aprendizaje no supervisado revela estructura, pero siempre requiere interpretación humana.

Resumen práctico IV: Redes neuronales y cierre

Técnica	Cuándo usarla / propósito
Perceptrón y MLP	Cuando las relaciones son altamente no lineales o hay gran cantidad de datos.
Funciones de activación	Permiten modelar fronteras suaves y mejorar la capacidad expresiva.
Entrenamiento (SGD, backprop)	Optimización iterativa mediante gradientes; requiere normalización y buen diseño del modelo.

Resumen práctico IV: Redes neuronales y cierre

Técnica	Cuándo usarla / propósito
Perceptrón y MLP	Cuando las relaciones son altamente no lineales o hay gran cantidad de datos.
Funciones de activación	Permiten modelar fronteras suaves y mejorar la capacidad expresiva.
Entrenamiento (SGD, backprop)	Optimización iterativa mediante gradientes; requiere normalización y buen diseño del modelo.

Reflexión final

La Ciencia de Datos no se trata solo de modelos, sino de combinar pensamiento estadístico, curiosidad y comunicación efectiva.

¡Gracias!

por su atención y participación durante el curso

Presentaciones Finales

Martes y Jueves 25 y 27 de Nov

Introducción a la Ciencia de Datos
Marco Antonio Aquino-López
CIMAT, 2025