Introduccion a ML – Sesion 2

Cesar Garcia

2025

Objetivos

- Comprender neuronas artificiales.
- Conocer pesos, bias (sesgo), activaciones.
- Entender flujo de información (forward propagation).
- Familiarizarse con funciones de activación comunes.

¿Qué es una Red Neuronal?

Una red neuronal es una función compuesta por varias transformaciones simples:

1. Capas

- **Capa de entrada:** recibe las características del problema.
- **Capas ocultas:** combinan pesos, sesgos y activaciones para aprender patrones.
- **Capa de salida:** produce la predicción final.

2. Neuronas

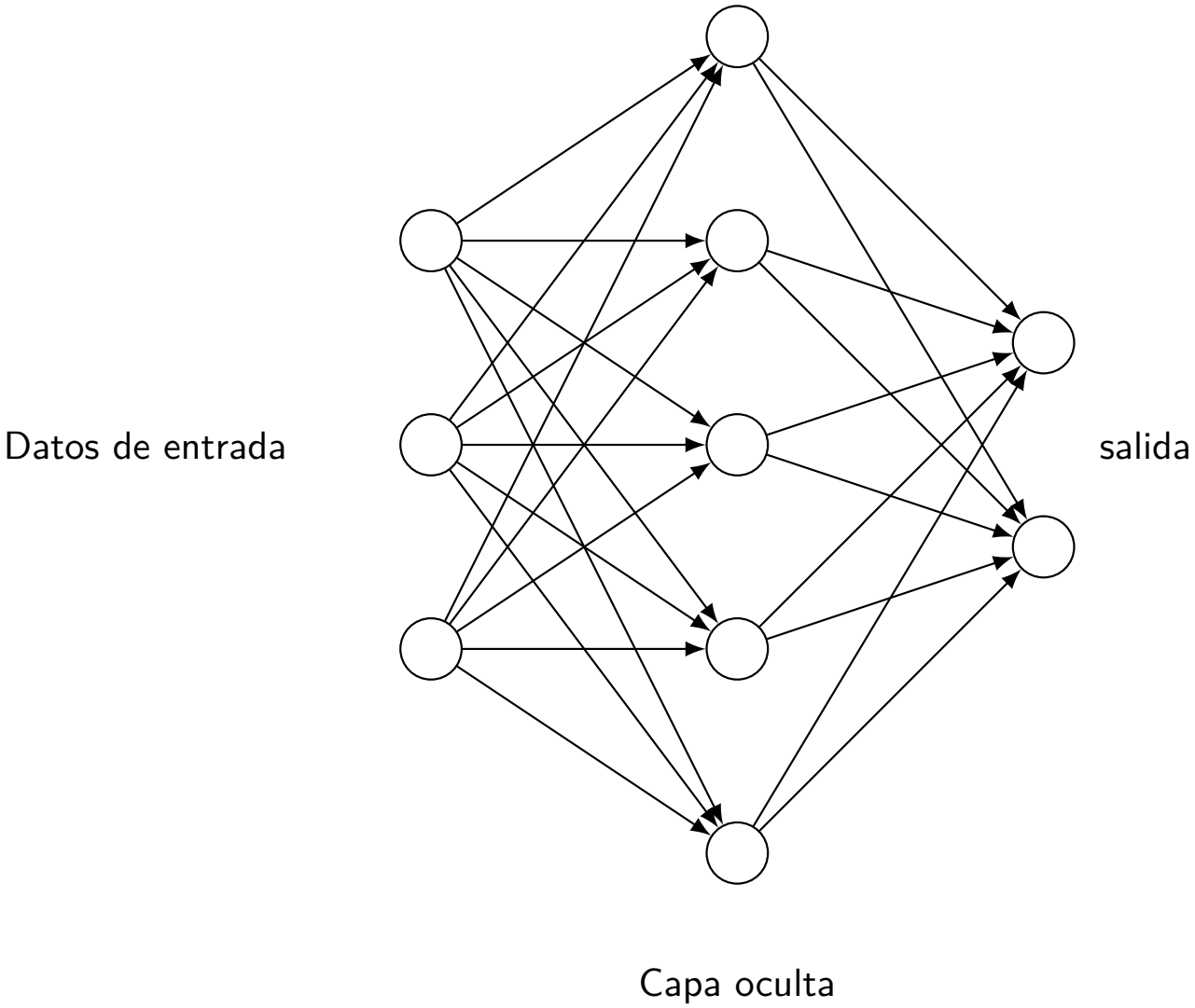
Cada neurona realiza:

$$z = Wx + b \qquad \Rightarrow \qquad a = f(z)$$

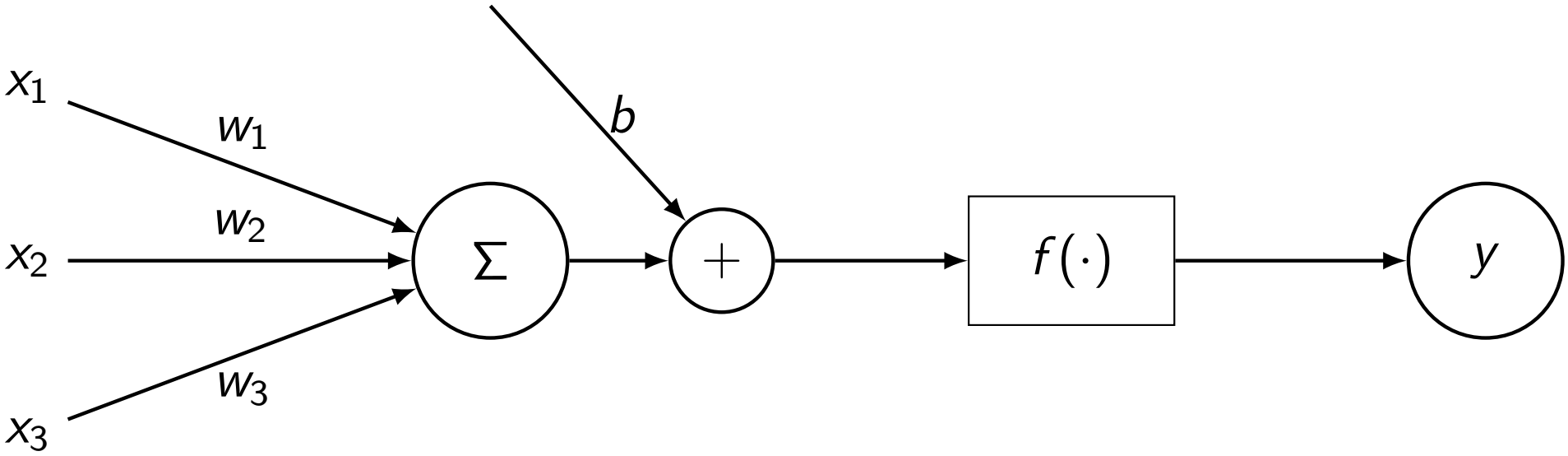
3. Conexiones y aprendizaje

- Pesos que conectan una capa con la siguiente.
- La red ajusta estos pesos durante el entrenamiento para minimizar el error.

Red Neural



Perceptron



$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Todo es Algebra Lineal

$$z = Wx + b$$

$$W = \begin{bmatrix} w*11 & w*12 & w*13 \\ w*21 & w*22 & w*23 \\ w*31 & w*32 & w*33 \\ w*41 & w*42 & w*43 \\ w*51 & w*52 & w*53 \end{bmatrix} * 5 \times 3 \qquad x = \begin{bmatrix} x*1 \\ x_2 \\ x_3 \end{bmatrix} * 3 \times 1$$

Forward Propagation

Propagación hacia adelante (Forward Propagation)

- Tomamos las entradas (x) del modelo.
- En cada capa calculamos una combinación lineal: ($z = Wx + b$).
- Aplicamos una función de activación: ($a = f(z)$).
- Repetimos capa por capa hasta obtener la salida ($\hat{y}$).

Ejemplo: red con 1 capa oculta

$$z^{(1)} = W^{(1)}x + b^{(1)}$$

$$a^{(1)} = f\left(z^{(1)}\right)$$

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}$$

$$\hat{y} = g\left(z^{(2)}\right)$$

Funciones de activación

¿Qué es una función de activación?

- Es una transformación no lineal aplicada después de la combinación lineal $z = Wx + b$.
- Permite que la red neuronal aprenda funciones complejas, no sólo líneas rectas.
- Sin activaciones no lineales, toda la red sería equivalente a una sola transformación lineal.
- Diferentes activaciones producen comportamientos distintos en el aprendizaje.

¿Por qué necesitamos no linealidad?

Idea clave

- Cada capa lineal realiza una transformación del tipo $z = Wx + b$.
- Si encadenamos varias capas solo lineales (sin función de activación), la composición sigue siendo **otra transformación lineal**.
- Eso significa que una "red profunda" sin activaciones no lineales sería equivalente a **una sola capa lineal**.
- No podríamos aproximar funciones complejas ni separar datos con fronteras no lineales.

¿Por qué necesitamos no linealidad?

Viendolo del punto de vista Algebraico

$$
\begin{aligned}
z^{(1)} &= W^{(1)}x + b^{(1)} \\
z^{(2)} &= W^{(2)}z^{(1)} + b^{(2)} \\
&= W^{(2)}\left(W^{(1)}x + b^{(1)}\right) + b^{(2)} \\
&= \left(W^{(2)}W^{(1)}\right)x + \left(W^{(2)}b^{(1)} + b^{(2)}\right)
\end{aligned}
$$

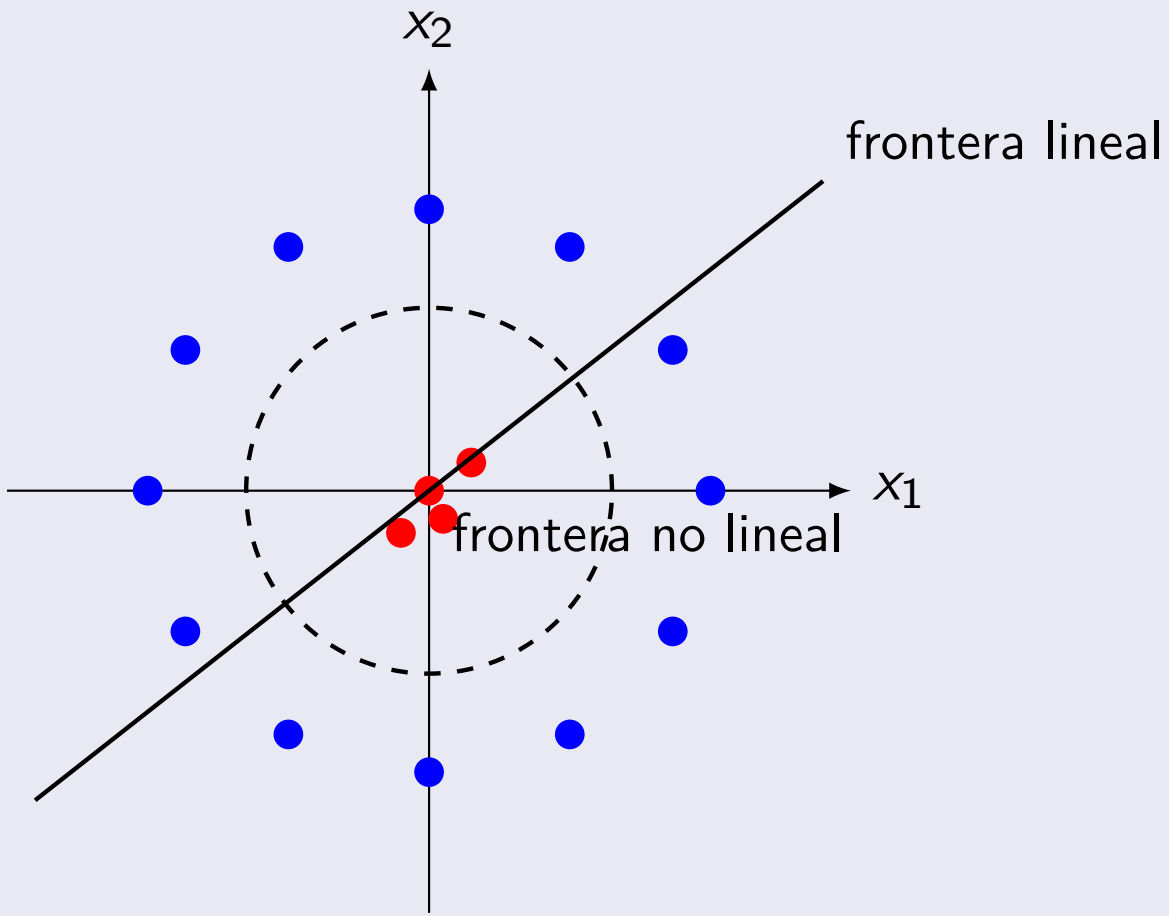
Sigue siendo de la forma:

$$z^{(2)} = W^{*}x + b^{*}$$

es decir, **lineal**.

¿Por qué necesitamos no linealidad?

Visualización



Una recta no puede separar las clases,
pero una frontera curva (no lineal) sí.

Teorema de Aproximación Universal

Un perceptrón multicapa (MLP) con **una sola capa oculta**, pero con suficientes neuronas, puede aproximar **cualquier función continua** en un conjunto compacto. Esto requiere una función de activación **no lineal** (ReLU, Sigmoid, Tanh, etc.).

Enunciado formal

*Para toda función continua $f : \mathbb{R}^n \to \mathbb{R}$; y para todo $\varepsilon > 0$, existe una red neuronal con **una capa oculta** y una activación no lineal $f(\cdot)$ tal que la red aproxima a f con un error menor que ε. en todo el dominio compacto.*

- En teoría, **no necesitamos muchas capas** para representar funciones complejas.
- En la práctica, las redes profundas aprenden **mejor, más rápido y con menos neuronas** que una red extremadamente ancha de una sola capa.

Funciones de Activación

Funciones de activación comunes

- ReLU
- Sigmoid
- Tanh
- GELU

Otras populares (includidas para referencia)

- Leaky ReLU
- ELU
- Swish (SiLU)

Popular para salidas

- Softmax

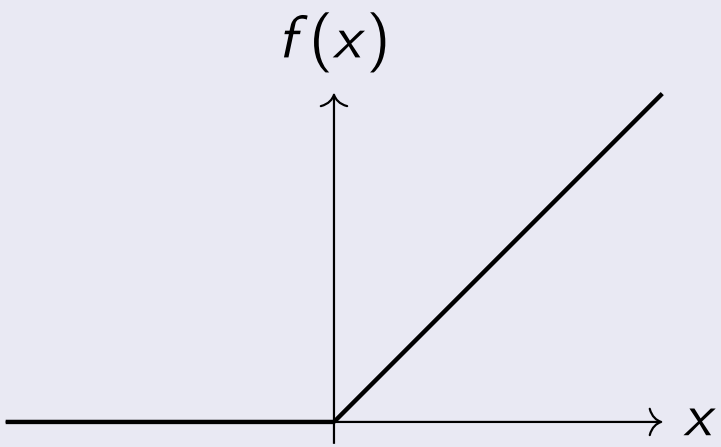
Función de activación: ReLU

Rectified Linear Unit (ReLU)

- Deja pasar solo valores positivos.
- Muy usada en redes profundas por su simplicidad y buen comportamiento en la práctica.
- Problema: neuronas pueden "morir" si la entrada es siempre negativa.

$$\text{ReLU}(x) = \max(0, x)$$

Visualización



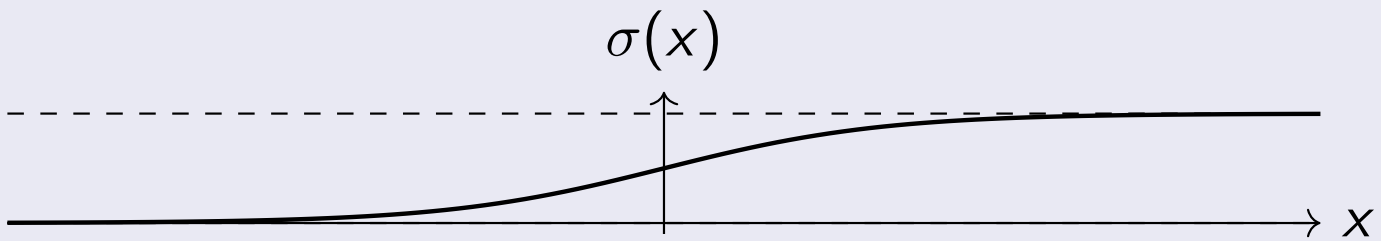
Función de activación: Sigmoide

Sigmoid / logística

- Comprime cualquier valor real al intervalo $(0, 1)$.
- Interpretación probabilística: salida como "probabilidad".
- Problema: gradientes muy pequeños para valores grandes en magnitud (saturación).

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Visualización



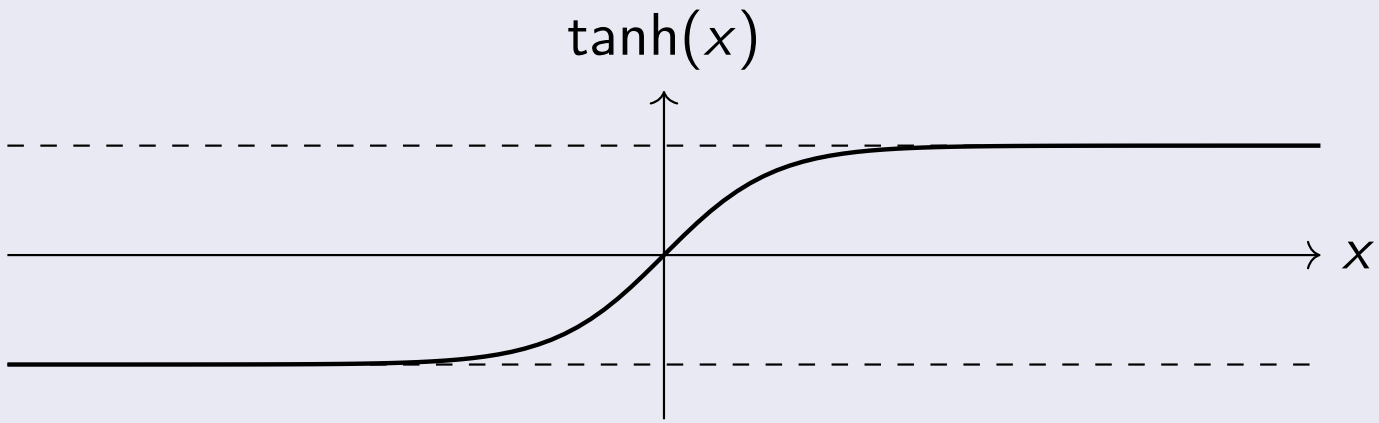
Función de activación: Tanh

Tangente hiperbólica

- Similar a la sigmoide pero centrada en 0.
- Rango en $(-1, 1)$.
- Útil cuando queremos activaciones con media cercana a cero.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Visualización



Gaussian Error Linear Unit (GELU)

- Suaviza la ReLU usando una distribución normal.
- Muy popular en modelos modernos (por ejemplo, Transformers).
- Activa más fuertemente valores positivos grandes, pero de forma suave.

Definición conceptual:

$$\text{GELU}(x) = x\,\Phi(x)$$

donde

$$\Phi(x)$$

es la CDF de la normal estándar.

Aproximación práctica usada en la mayoría de implementaciones:

$$\text{GELU}(x) \approx 0.5\,x\Big(1 + \tanh\big(\sqrt{2/\pi}\,(x + 0.044715\,x^3)\big)\Big)$$

Visualización

Función de activación: Softmax

Softmax

- Convierte un vector de scores en un vector de probabilidades.
- Muy usada en la capa de salida para clasificación multiclase.
- La suma de las salidas es 1.

Para una salida de dimensión K:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, \ldots, K$$

Ejemplo numérico

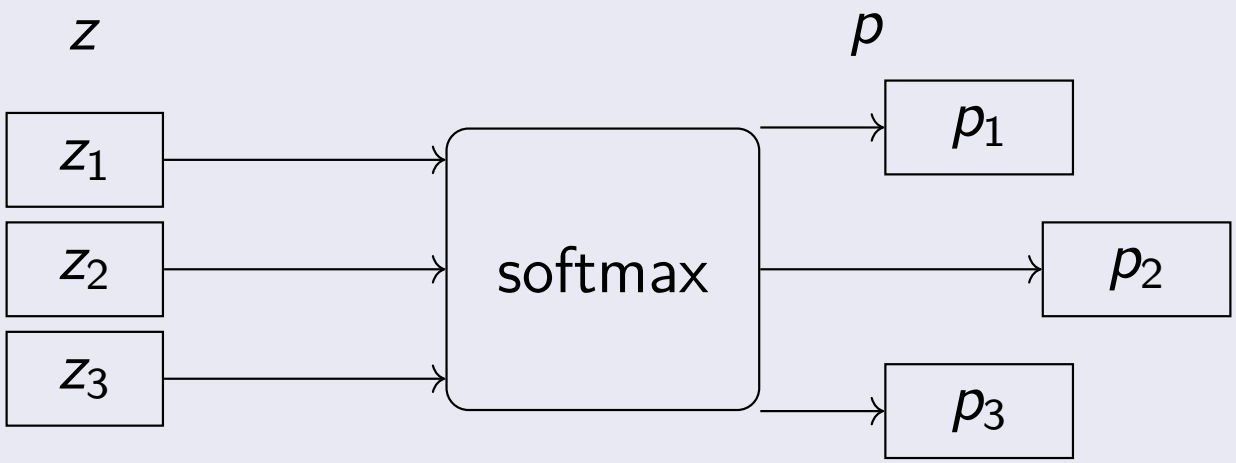
Entrada $z = [2.0, 1.0, 0.1]$

$e^z \approx [7.389, 2.718, 1.105]$

Suma $= 11.212$

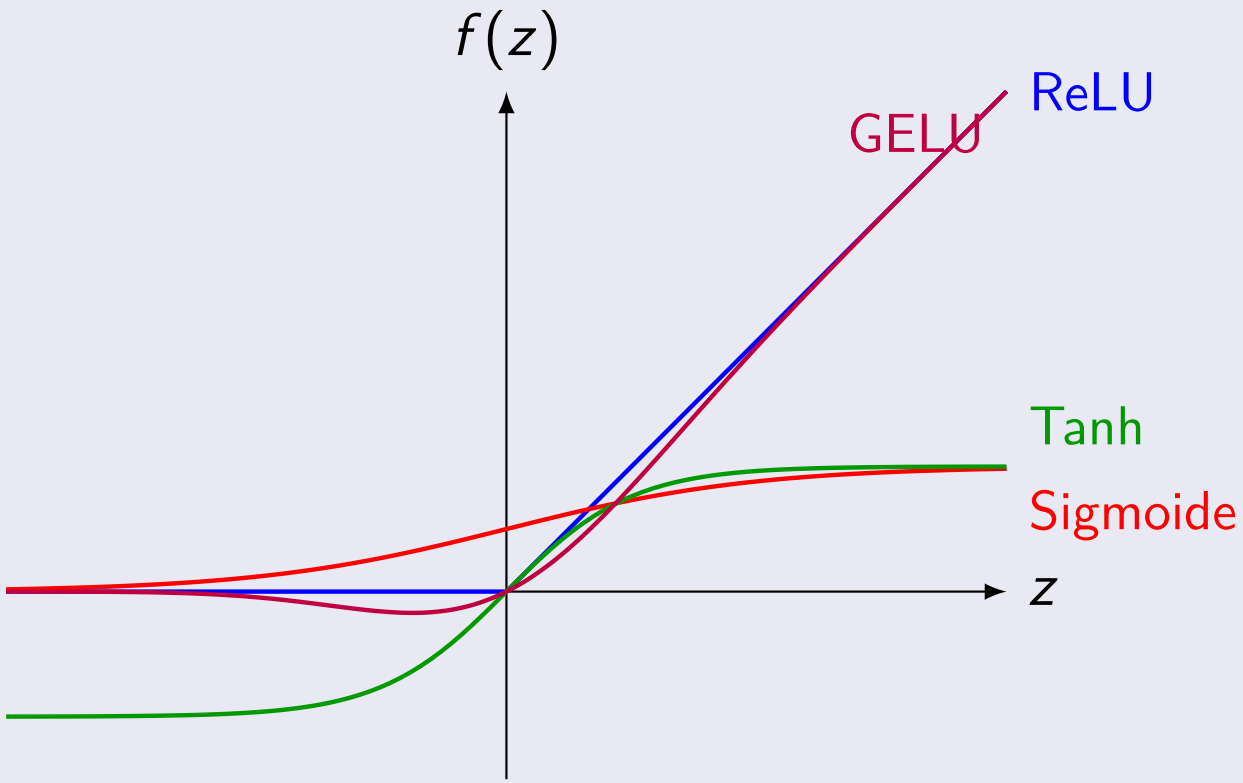
$\text{softmax}(z) \approx [0.66, 0.24, 0.10]$

Visualización



Funciones de Activación – Comparación

Visualización



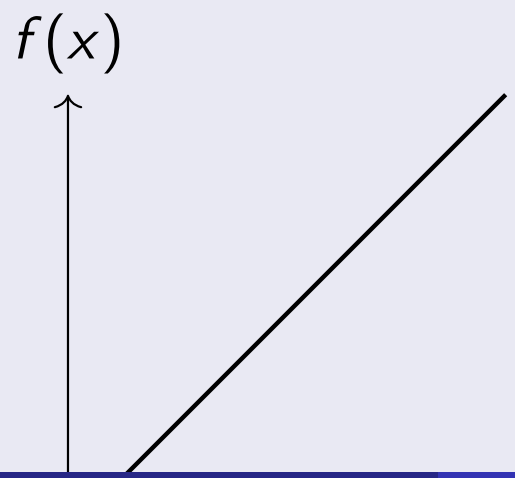
Función de activación: Leaky ReLU

Leaky Rectified Linear Unit

- Variante de ReLU que permite un pequeño gradiente cuando $x < 0$.
- Evita el problema de "neuronas muertas".
- Parámetro α controla la pendiente en la parte negativa (típicamente $\alpha = 0.01$).

$$\text{LeakyReLU}(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$$

Visualización



Función de activación: ELU

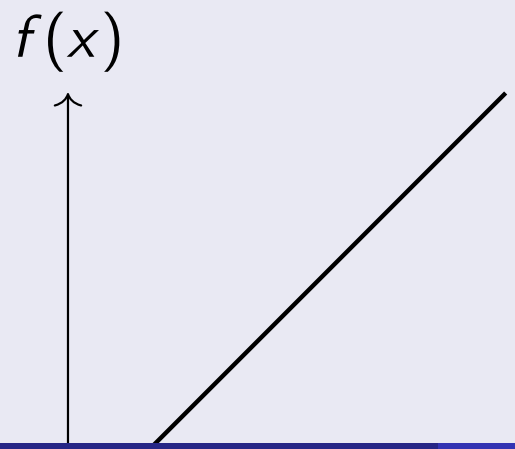
Exponential Linear Unit (ELU)

- Similar a ReLU para $x \geq 0$, pero suave en la parte negativa.
- Produce activaciones negativas que ayudan a centrar la salida alrededor de 0.

$$\text{ELU}(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$$

Típicamente $\alpha = 1$.

Visualización



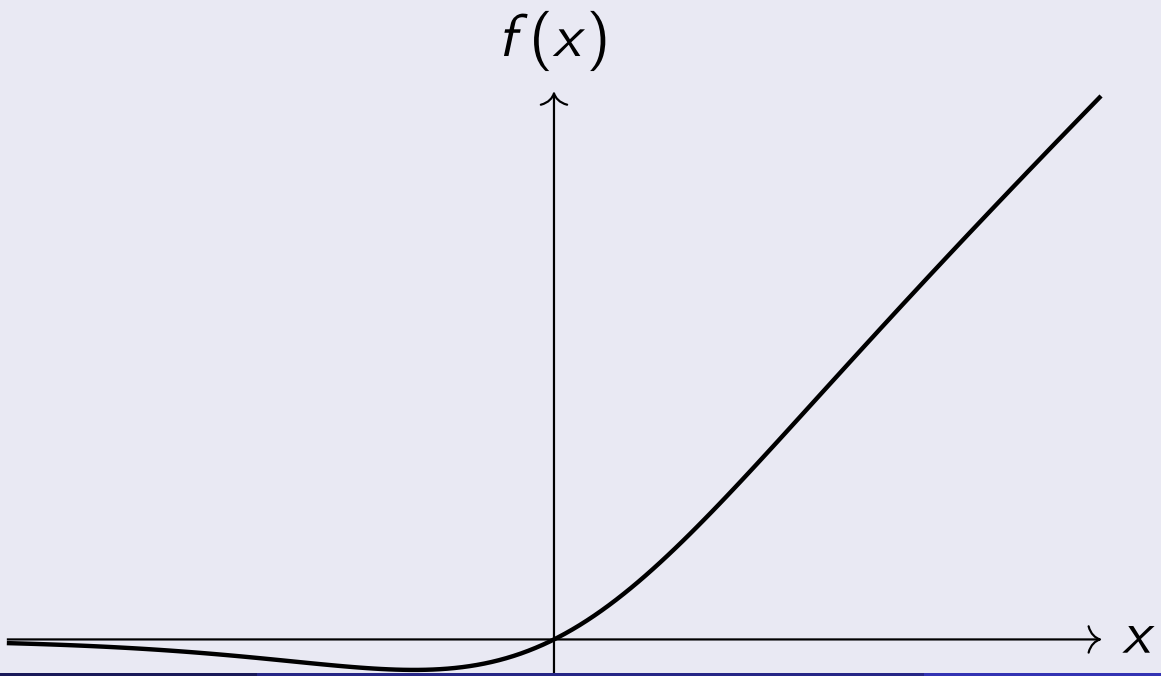
Función de activación: Swish (SiLU)

Swish / SiLU - Sigmoid Linear Unit

- Activación moderna usada en redes eficientes (EfficientNet, etc.).
- Similar a GELU en comportamiento.

$$\text{Swish}(x) = x\,\sigma(x) = \frac{x}{1 + e^{-x}}$$

Visualización



Tensorflow Playground