Tema 2.4
Configuración de una red neuronal

Deep Learning

Máster Oficial en Ingeniería Informática

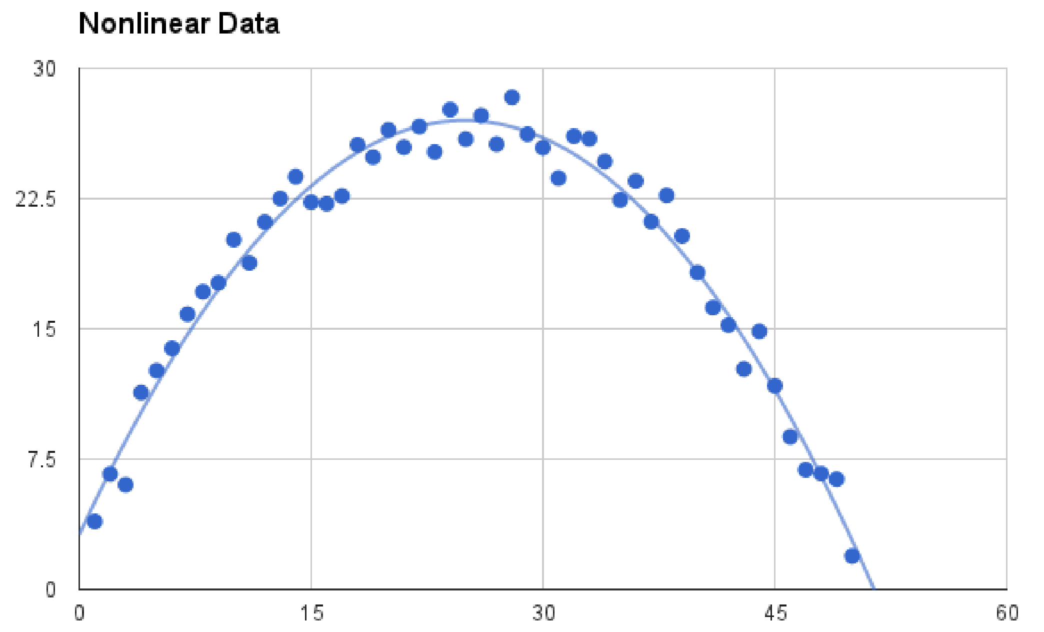
Universidad de Sevilla

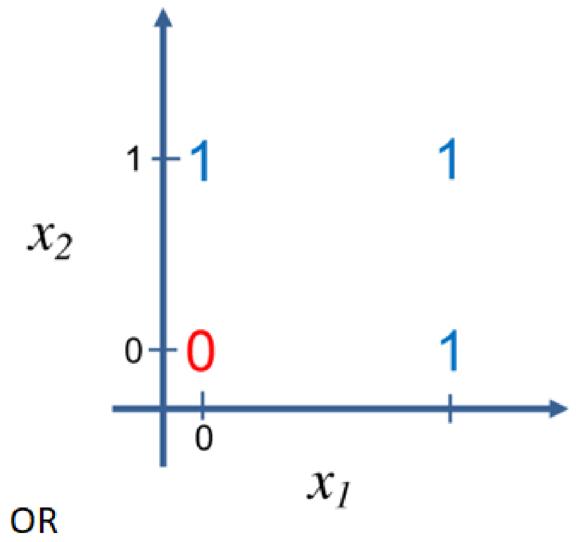
Contenido

- No linealidad
- Funciones de activación:
  - Capa de salida
  - Capas ocultas
  - Pautas prácticas
- Inicialización de pesos
- Demo

No linealidad

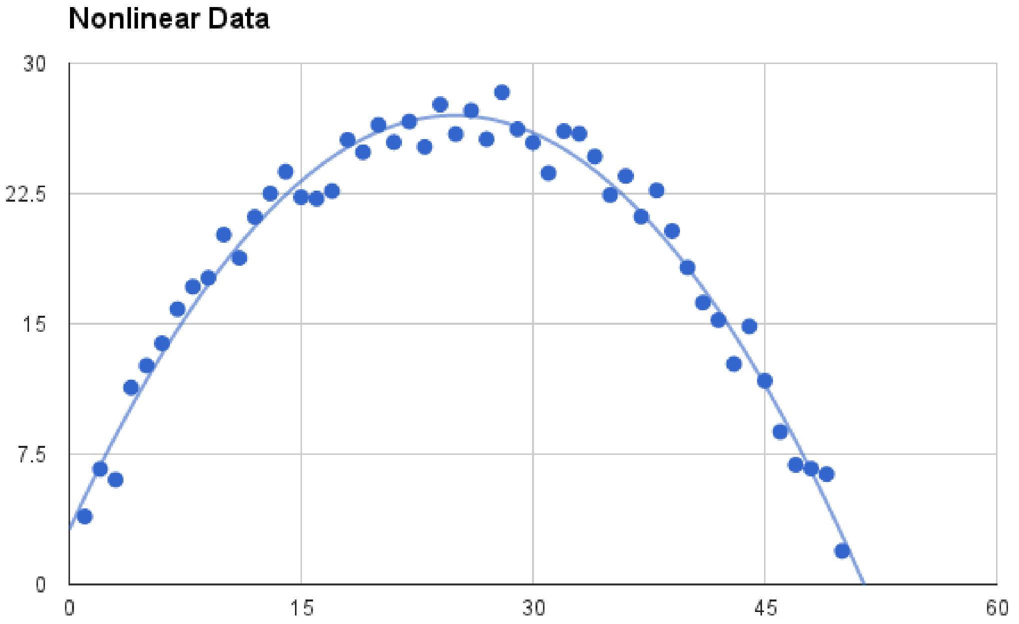
- Los modelos lineales presentan limitaciones por su falta de capacidad.
- Por ejemplo: un modelo lineal no puede aprender la función XOR

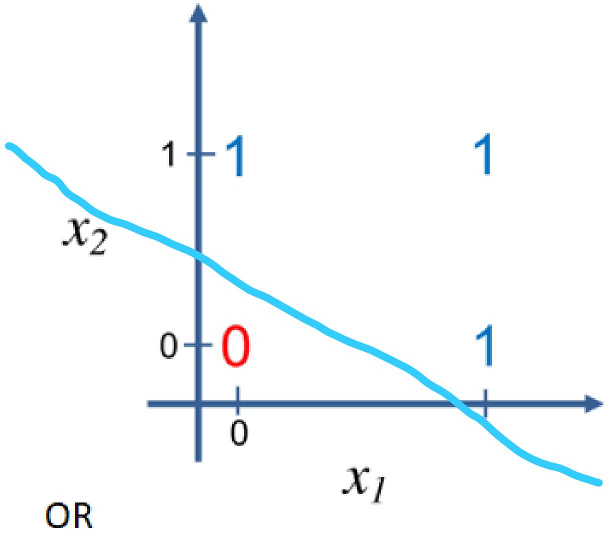




No linealidad

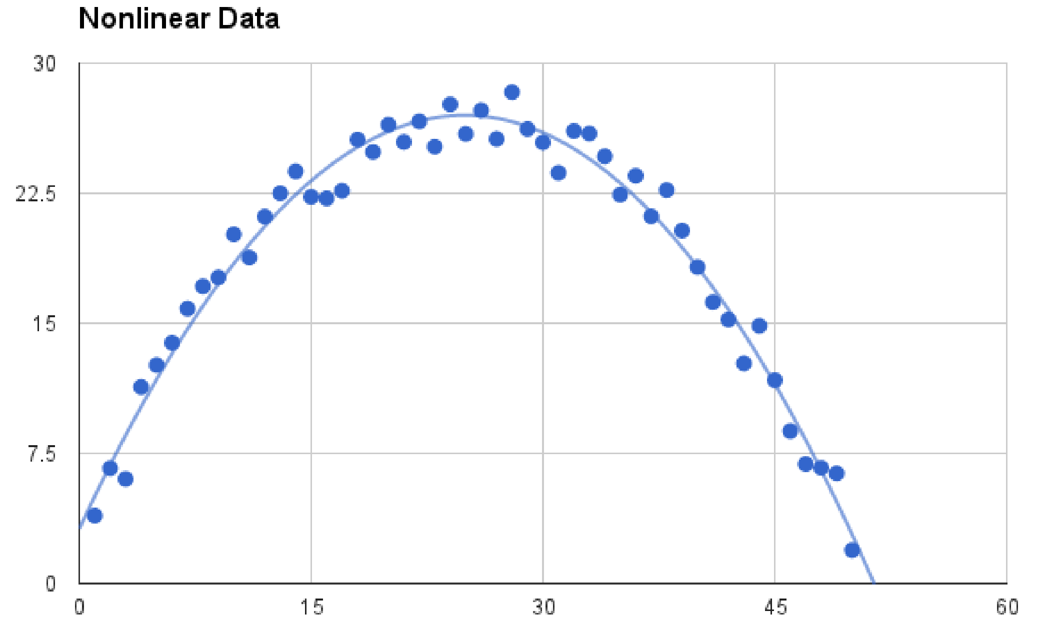
- Los modelos lineales presentan limitaciones por su falta de capacidad.
- Por ejemplo: un modelo lineal no puede aprender la función XOR

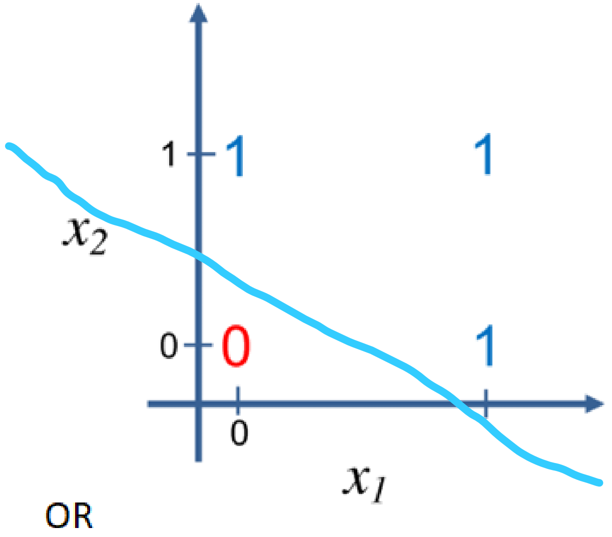


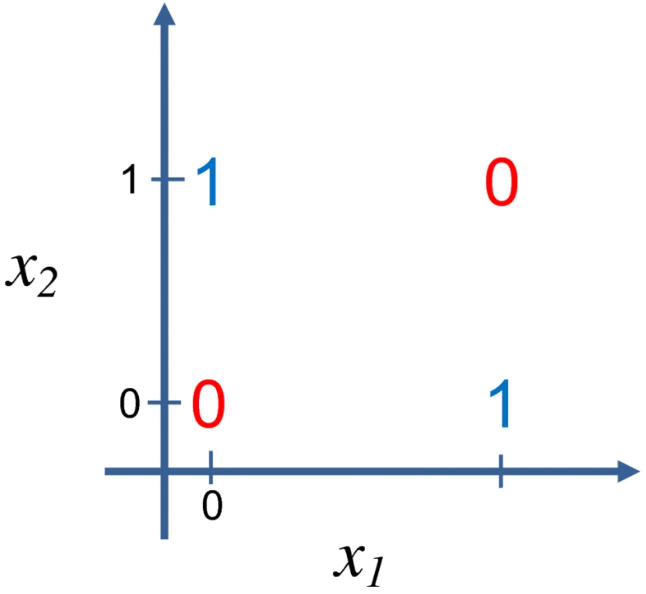


No linealidad

- Los modelos lineales presentan limitaciones por su falta de capacidad.
- Por ejemplo: un modelo lineal no puede aprender la función XOR

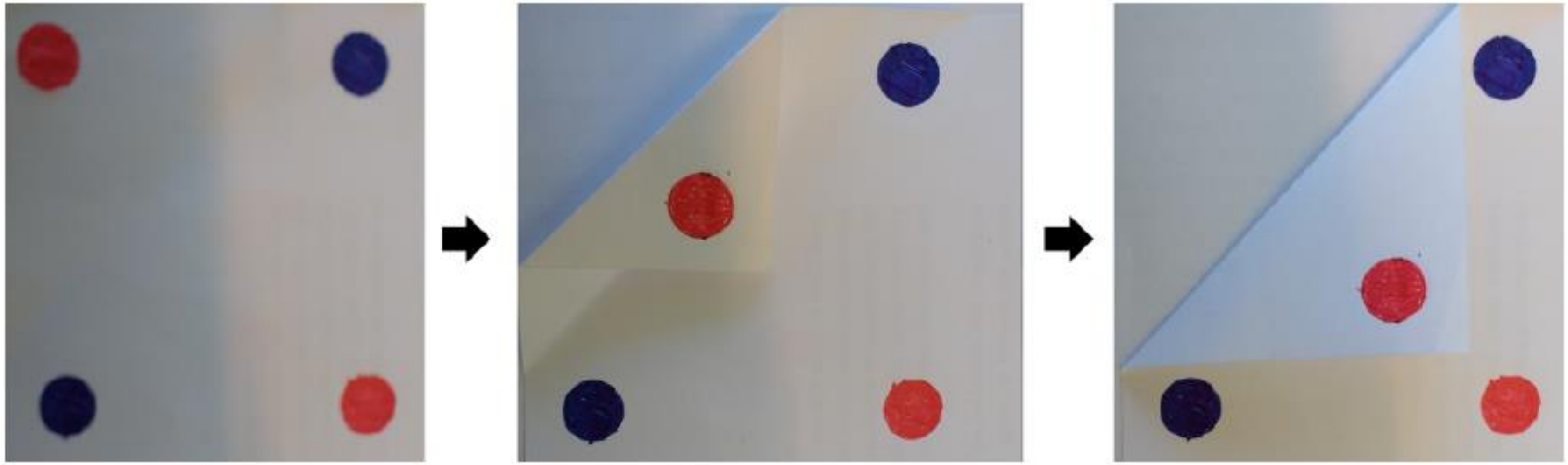






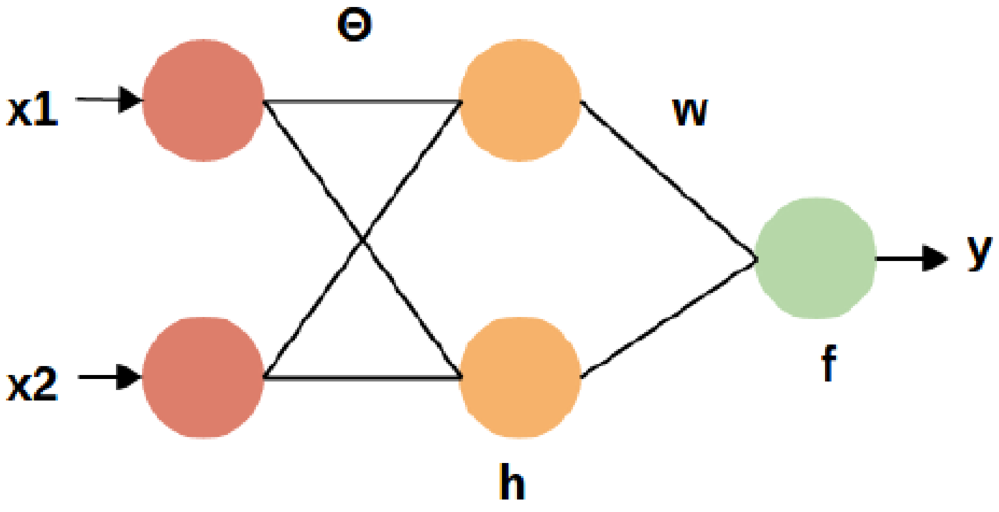
No linealidad

- Una solución: que el modelo aprenda una representación diferente de los datos de entrada, para que un modelo lineal sí que pueda separarlos.



No linealidad

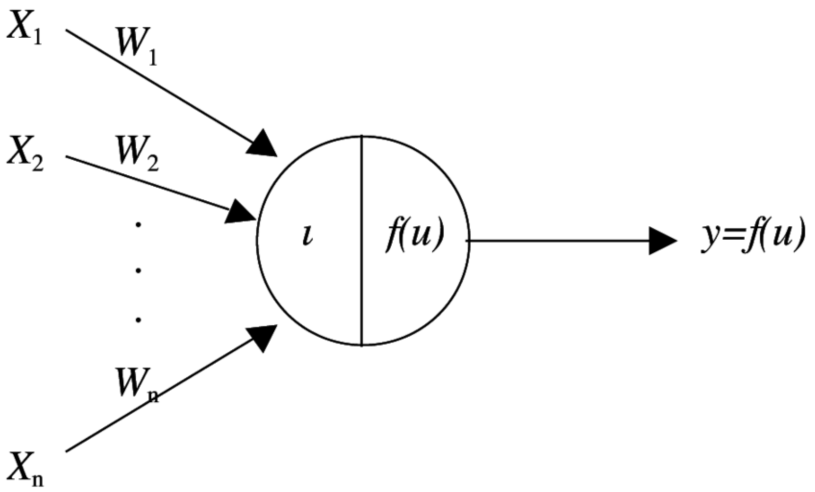
- Usar una red multicapa para transformar la representación
- h: capa oculta de 2 neuronas
  - Computa la función $f^1(x, \theta, c)$
  - θ es una matriz de pesos
  - Transforma las entradas x
- f: capa de salida de una neurona
  - Computa la función $f^2(h, w, b)$
  - Regresión lineal sobre h



- El modelo final es: $f(x, \theta, c, w, b) = f^2\left(f^1(x)\right) = (f^2 \cdot f^1)\,(x)$

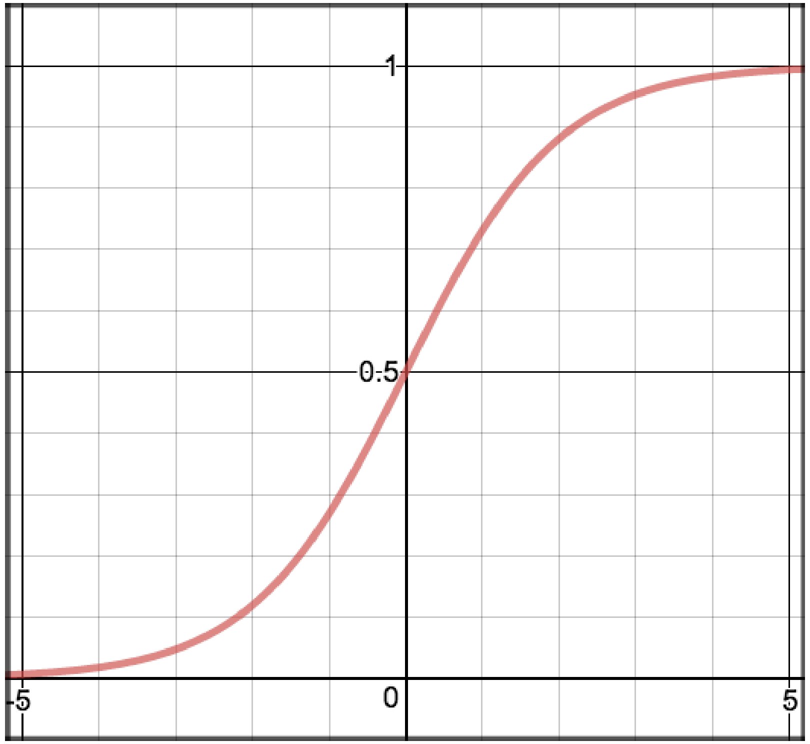
No linealidad

- Si f^1 y f^2 son lineales, f también lo es!
  - Si $f^1(x) = \theta^T x$ y que $f^2(h) = h^T w$
  - Entonces $f(x) = x^T w'$, donde $w' = \theta w$
- Hay usar una función no lineal, mediante la función de activación:
  - $h = g(\theta^T x + c)$
  - g es la función de activación



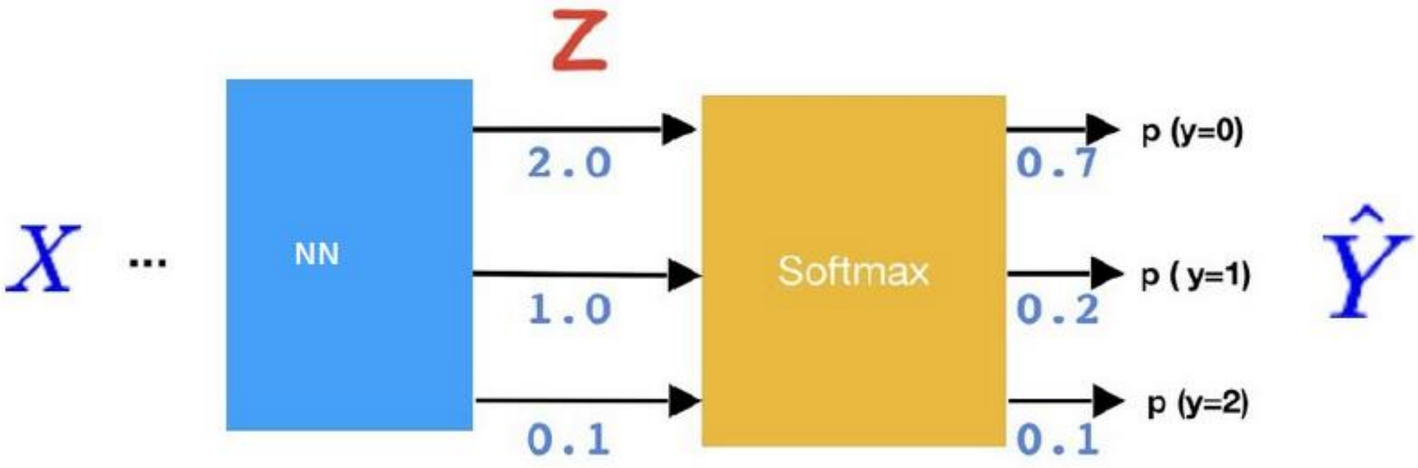
Funciones de activación: capa de salida

- La función **sigmoide o logística**: $\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}}$
- Se suele usar en la capa de salida para **clasificación binaria**, y **multiclase** *no mutuamente excluyente*.
- Función de pérdida: *binary cross entropy*
  - −(ylog(p)+(1−y)log(1−p))
- La salida se puede interpretar como una probabilidad (entre 0 y 1).
- Atenúa valores extremos sin llegar a eliminarlos.



Funciones de activación: capa de salida

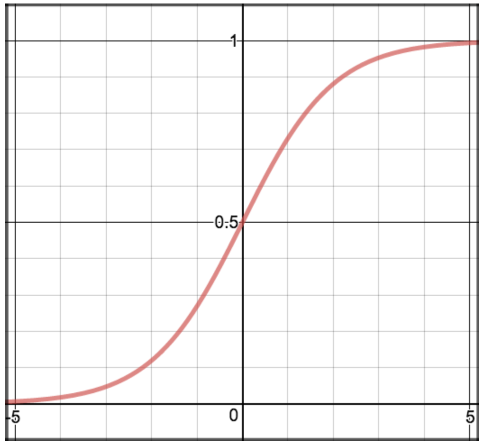
- La función **softmax**: $\boldsymbol{\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}}}$



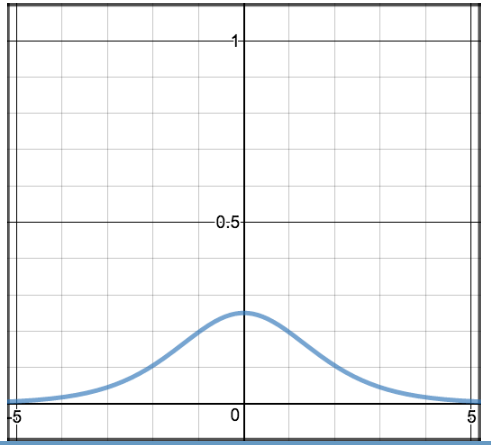
- Se suele usar en la capa de salida para **clasificación multiclase** (*mutuamente excluyente*).
- Son probabilidades por clase.
- Función de pérdida: *cross entropy*
  - $L(p,q) = -\sum_{x=1}^{N} q(x) \log p(x)$, *siendo p la función estimada, q la verdadera.*

Funciones de activación: capas ocultas

Función de activación



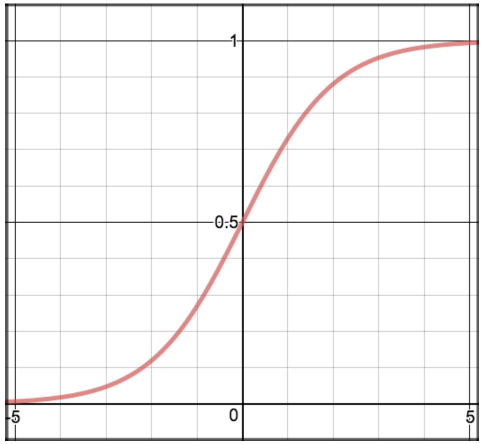
Derivada de la función



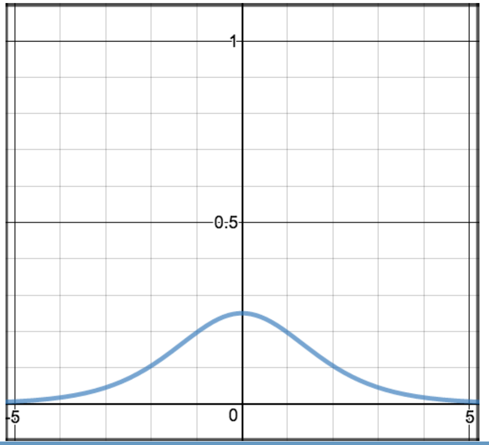
Funciones de activación: capas ocultas

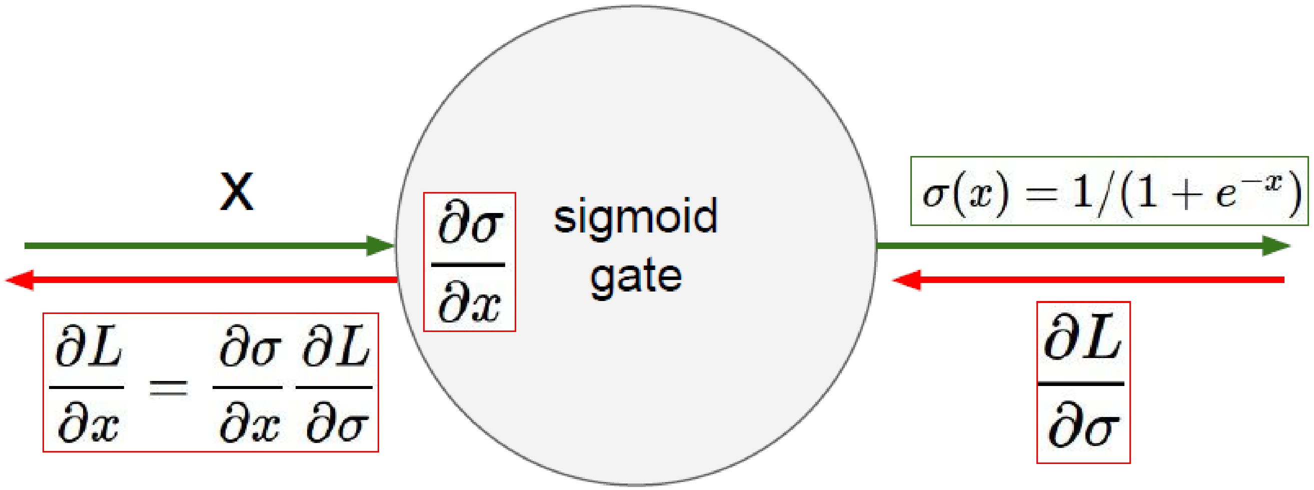
- La función **sigmoide** o logística: $\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}}$
- Desventajas:
  - **Vanishing/exploiding gradient problem**: Las neuronas saturadas "matan" el gradiente

Función de activación



Derivada de la función



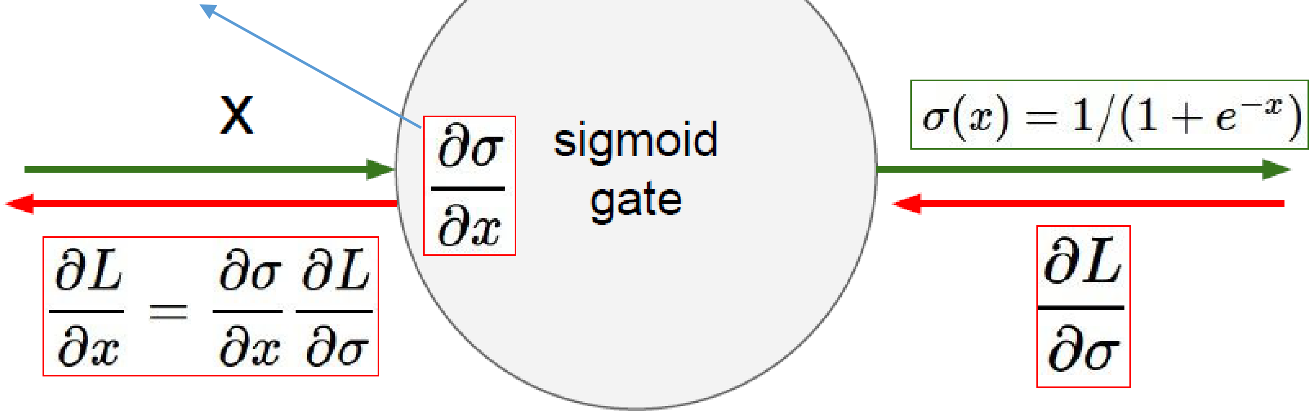


¿Qué ocurre si x=-10, x=0, x=10?

Funciones de activación: capas ocultas

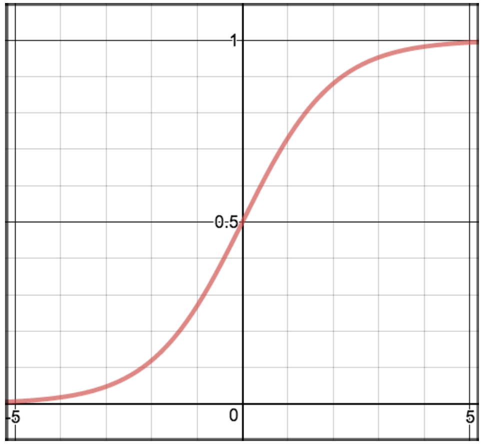
- La función **sigmoide** o logística: $\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}}$
- Desventajas:
  - **Vanishing/exploiding gradient problem**: Las neuronas saturadas “matan” el gradiente

$$\frac{\partial\sigma(x)}{\partial x} = (1 - \sigma(x))\sigma(x)$$

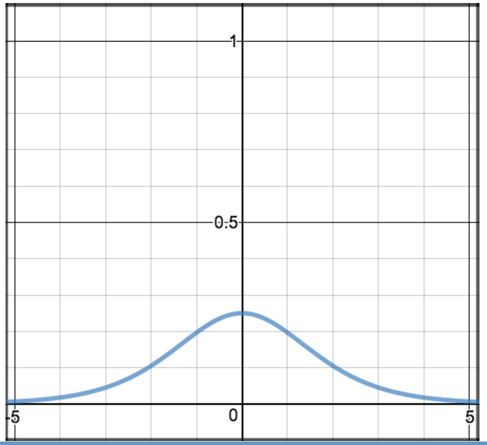


¿Qué ocurre si x=-10, x=0, x=10?

Función de activación

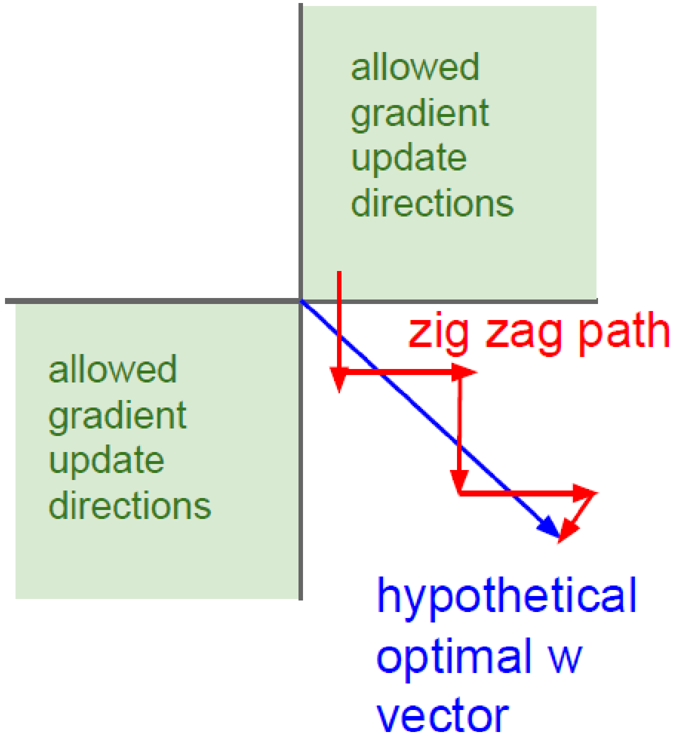


Derivada de la función



Funciones de activación: capas ocultas

- La función **sigmoide** o logística: $\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}}$
- Desventajas:
  - **Vanishing/exploiding gradient problem**: Las neuronas saturadas “matan” el gradiente
  - La salida de la sigmoide **no** está **centrada en cero**.
    - Implicación en descenso por gradiente.
    - Las salidas son siempre positivas (y por tanto, las entradas de los siguientes nodos de la red), por tanto:
    - **Los gradientes para los pesos son siempre negativos o positivos**!
    - La actualización de pesos dará lugar a un zig-zag
    - Se ve compensando cuando usamos batch.



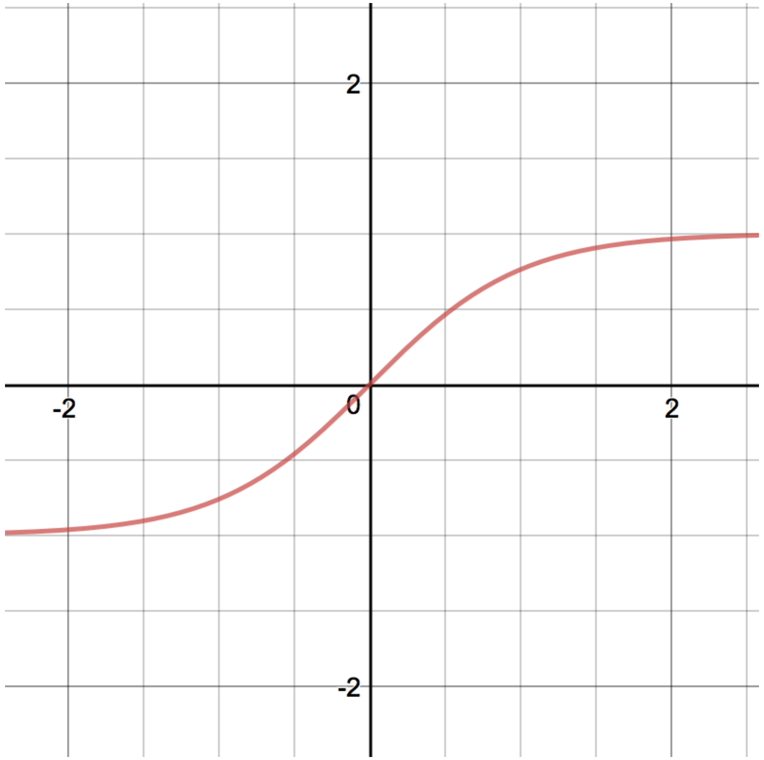
Funciones de activación: capas ocultas

- La función **sigmoide** o logística: $\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}}$
- Desventajas:
  - **Vanishing/exploiding gradient problem**: Las neuronas saturadas “matan” el gradiente
  - La salida de la sigmoide **no** está **centrada en cero**.
  - La computación de la exponencial ($\boldsymbol{e^{-x}}$) es costoso.

Funciones de activación: capas ocultas

- La función **tanh**: $\boldsymbol{tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1}$

[LeCun et al. 1991]

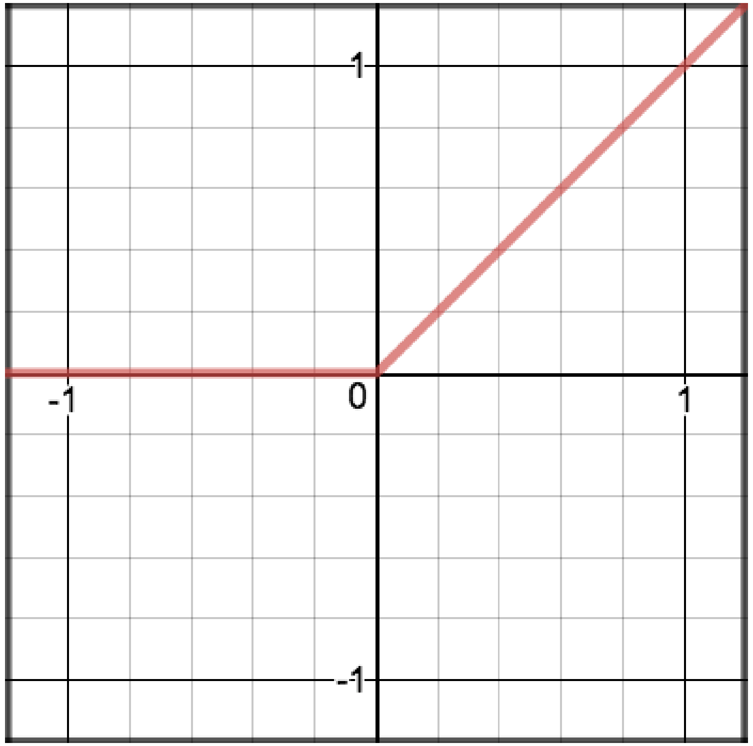


- Limita valores entre -1 y 1
- Ventaja:
  - Salida centrada en cero.
- Desventajas:
  - Todavía mata los gradientes cuando se satura.

Funciones de activación: capas ocultas

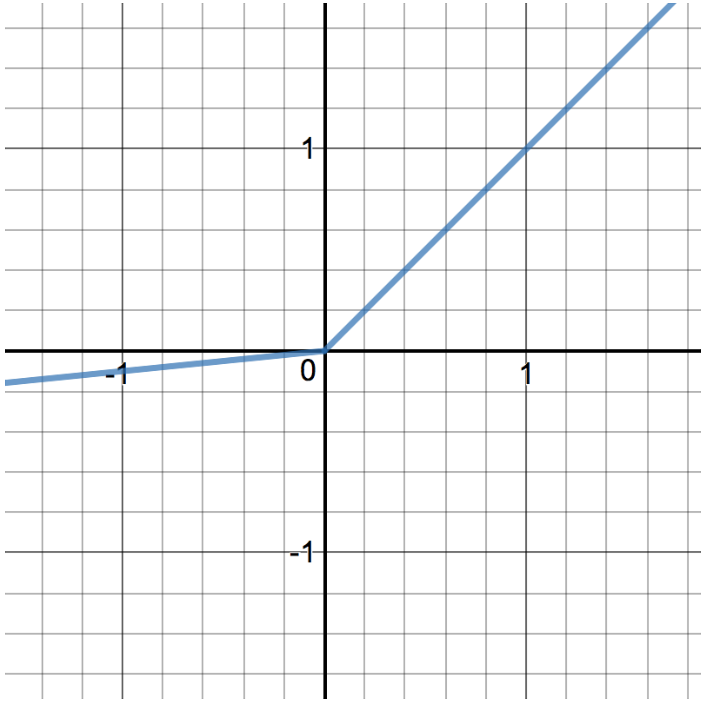
- La función **ReLU**: $\boldsymbol{ReLU(x) = max(0, x)}$
- Ventaja:
  - No se satura (en la región positiva).
  - Muy eficiente computacionalmente.
  - Converge mucho más rápido en la práctica (6x).
- Desventajas:
  - Salida no centrada en cero, y sin límite superior.
  - Problema **dying ReLUs**: Pueden morir
    - Nunca activarse, nunca actualizarse.
    - Debido a veces por alto learning rate.
    - Se suele usar un bias positivo bajo (p.ej. 0,01).

[Krizhevsky et al. 2012]



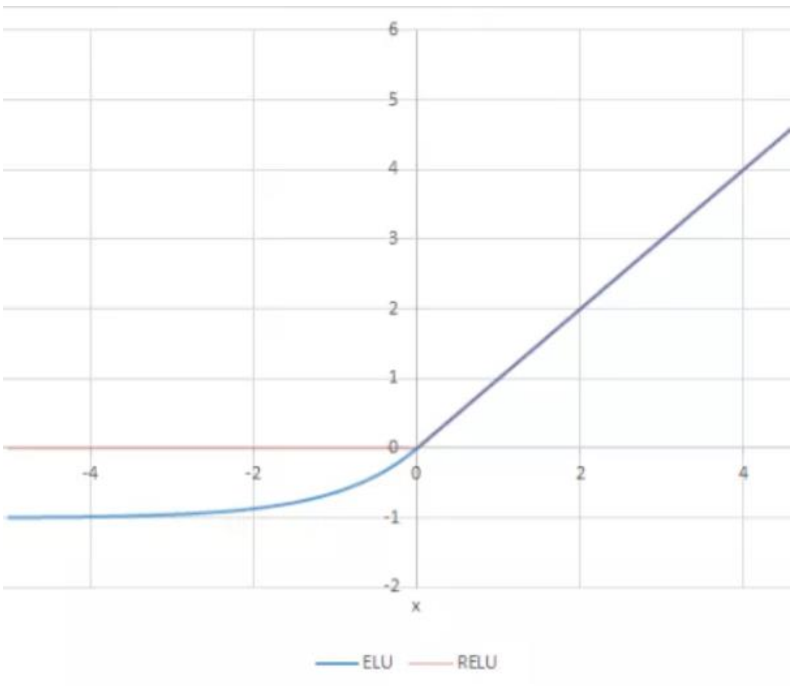
Funciones de activación: capas ocultas

- La función **Leaky ReLU**: $\boldsymbol{LReLU(x)} = \begin{cases} \boldsymbol{x, x > 0} \\ \boldsymbol{\alpha x, x \leq 0} \end{cases}$ [Maas et al. 2013]
- Parametrizada.
- Ventaja:
  - No se satura (en la región positiva).
  - Muy eficiente computacionalmente.
  - Converge mucho más rápido en la práctica (6x).
  - Evita dying ReLUs, con gradiente positivo cuando la unidad no está activada.
- Desventaja:
  - Incluye algo de linealidad (no para casos complejos).



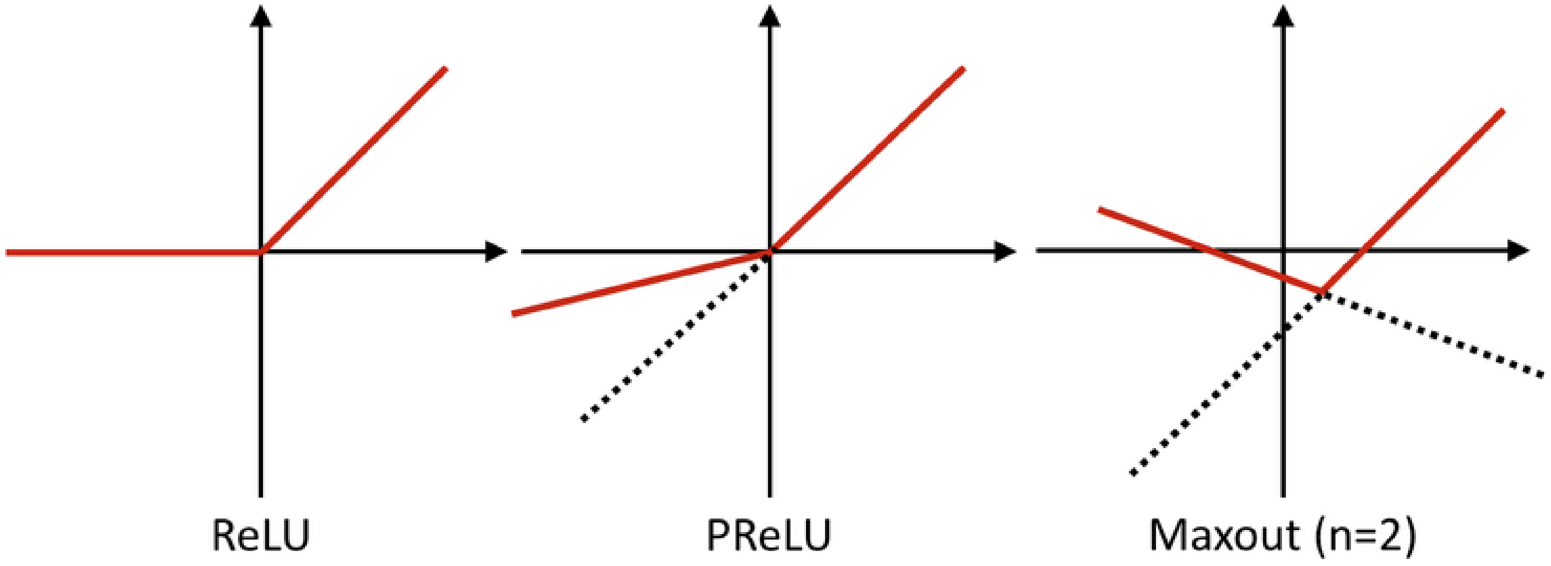
Funciones de activación: capas ocultas

- La función **Exponential ReLU**: $ELU(x) = \begin{cases} x, x > 0 \\ \alpha(e^z - 1), x \leq 0 \end{cases}$ [Clevert et al. 2015]
- Suaviza la respuesta en valores cercanos a cero:
  - ayuda a acelerar el entrenamiento.
- Ventaja:
  - Todos los de ReLU.
  - No muere.
  - Salidas cercanas con media cero.
- Desventaja:
  - Requiere usar la función exponencial.



Funciones de activación: capas ocultas

- La función **Maxout**: $\boldsymbol{maxout(x) = max(w_1^T x + b_1, w_2^T x + b_2)}$ [Goodfellow et al. 2013]
- Generalización de ReLU y Leaky ReLU.
- Es no lineal (no es tan solo el producto de matrices).
- Ventaja:
  - No se satura.
  - No muere.
- Desventaja:
  - Duplica los parámetros de la neurona!



Funciones de activación

- Pautas prácticas:
  - **No usar sigmoide** en capas ocultas.
  - Usar **ReLU**, llevando cuidado con el learning rate, y si es posible, monitorizar el porcentaje de unidades muertas.
  - Si esto nos afecta mucho, probar Leaky ReLU, ELU o Maxout.
  - Probar también tanh, pero podemos esperar que funcione peor que ReLU/Maxout.

Inicialización de pesos

- Inicialización con valores **aleatorios (distribución normal).**
- Inicialización **Xavier** (Glorot): [Glorot et al. 2010]
  - Los pesos w_i de cada unidad *i* se inicializan con un aleatorio normal con media 0 y desviación estándar $\frac{1}{\sqrt{fan_{in}^{i}}}$, es decir: $\mathrm{N}(0, \frac{1}{\sqrt{fan_{in}^{i}}})$,
  - fan_{in}^{i} es el número de entradas que tiene la unidad i.
  - Generalmente usado con tanh, pero con ReLU no funciona bien.
- Inicialización **He Normal** (He-etal): [He et al. 2015]
  - Corrige la inicialización Xavier para unidades ReLU.
  - Inicializa los pesos de cada unidad *i*: $\mathrm{N}(0, \sqrt{\frac{2}{fan_{in}^{i}}})$

Recapitulación

- Necesitamos unidades **no lineales** para poder tratar con datos complejos.
- Usaremos **funciones de activación** no lineales:
  - Para capa de salida: sigmoide si clasificación **binaria**, softmax para **multiclase**.
  - Para capas ocultas: usar **ReLU**, o Leaky ReLU, ELU, Maxout.
- **Inicialización** de pesos:
  - Usar valores **aleatorios**.
  - Con ReLU se recomienda **He-normal**, Xavier para tanh.
- Hemos vistos algunas arquitecturas de red, y una demo.

Demo

- Comprobemos la potencia representacional de una red con https://playground.tensorflow.org