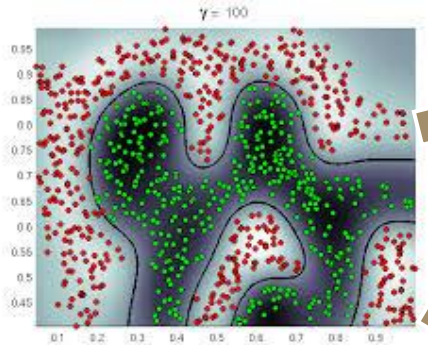
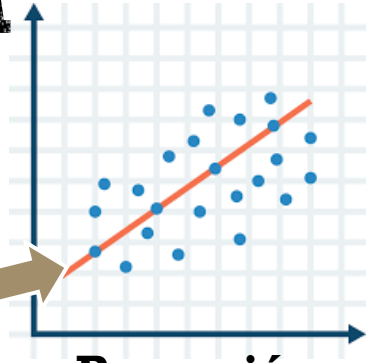
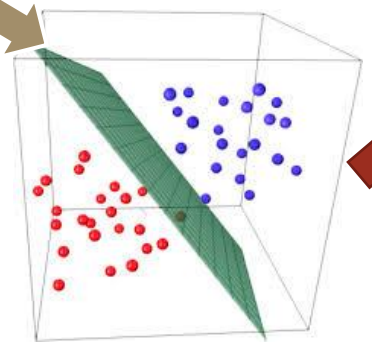
AGENDA



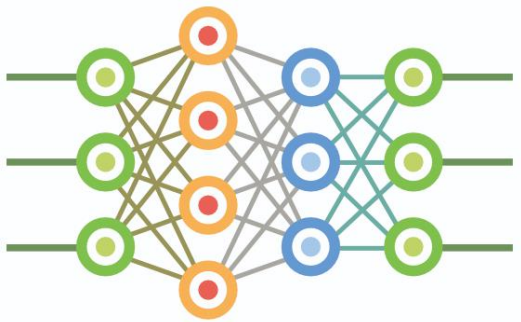
Aprendizaje supervisado



Regresión (OLS)



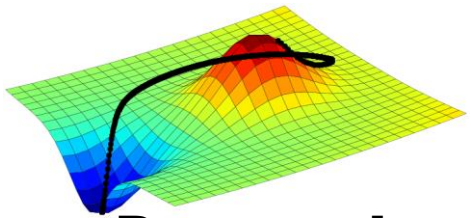
Clasificación (Reg. Logística)



Redes Neuronales Artificial (ANN)

$$\frac{\partial L}{\partial W^{[2]}} = \frac{\partial L}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial W^{[2]}}$$

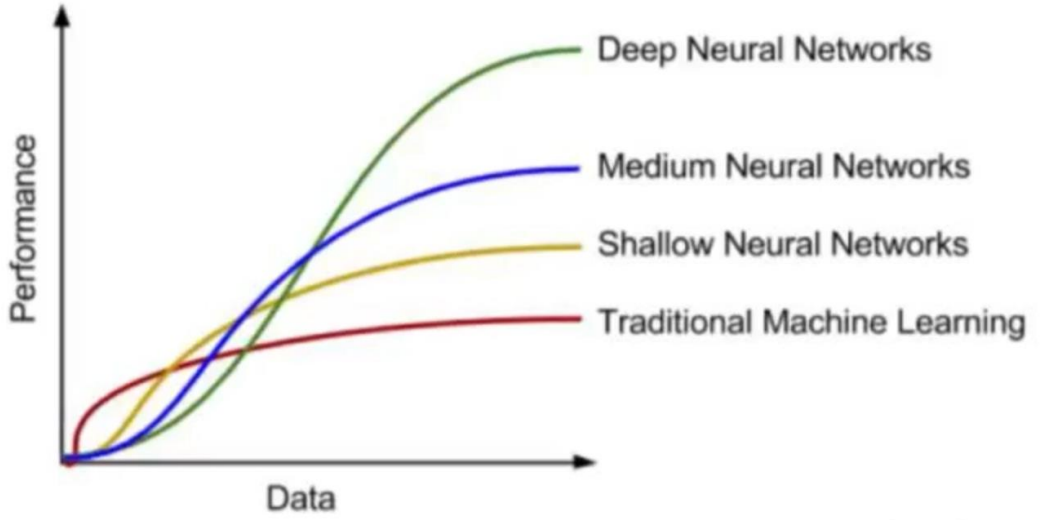
Back-Propagation



Descenso de gradiente

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

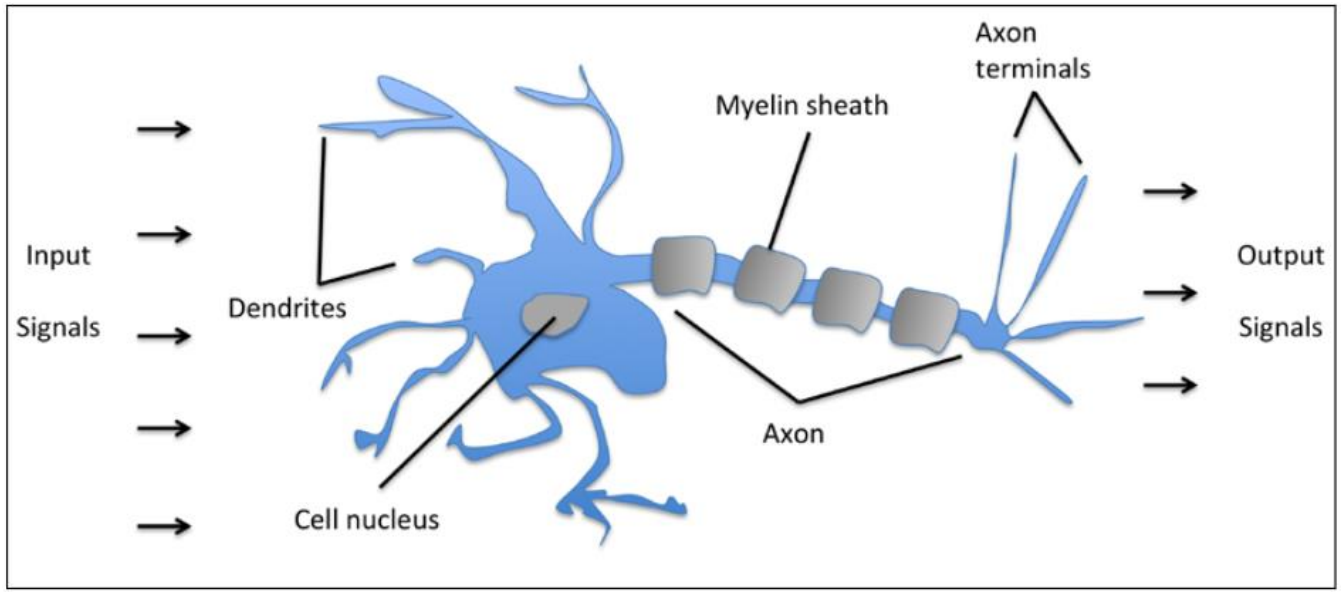
REDES NEURONALES



Alejandro 2016

REDES NEURONALES

- Modelos de aprendizaje automático **bio-inspirados**: tratan de modelar cómo funciona el cerebro → 1943 (McCulloch & Pitts), transmisión de señales eléctricas y químicas
- Simplificación. Una neurona:
  - recibe **múltiples** señales de entrada,
  - que se **acumulan** en el cuerpo de la neurona
  - emiten una señal binaria que evalúa el sobrepaso de un **umbral**
  - Se conectan a otras neuronas a partir de sinapsis entre axones y dendritas



Python Machine Learning, 2015

¿cómo se parece esto a una regresión?

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

REDES NEURONALES

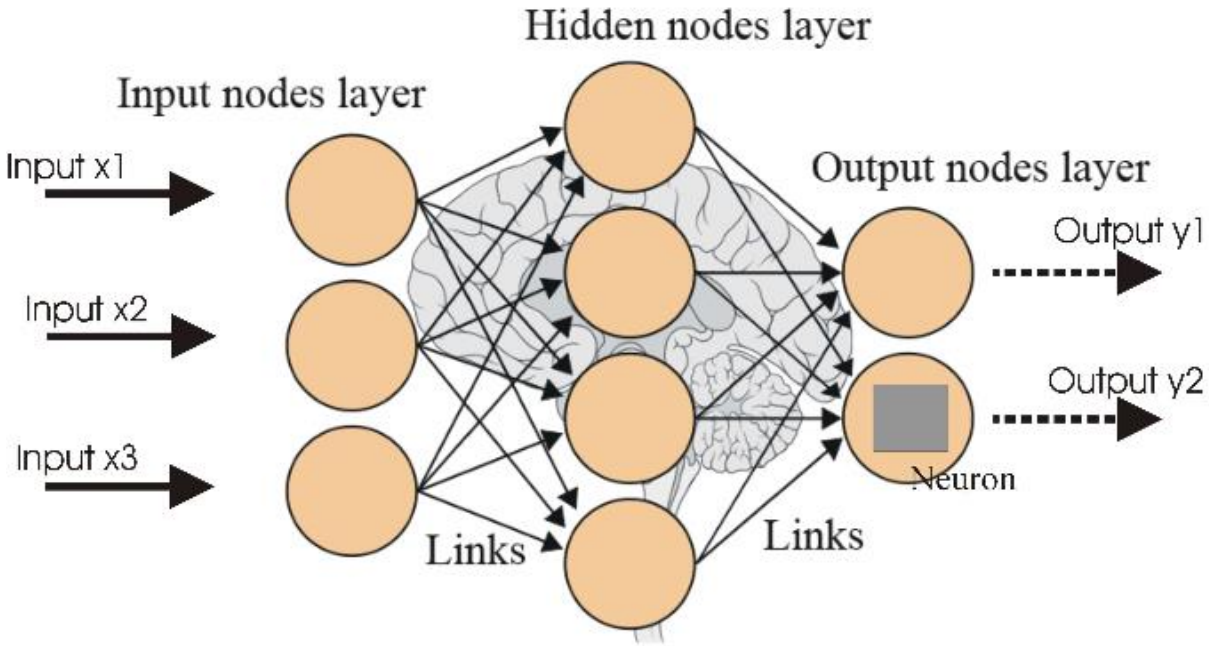
Una red neuronal se distingue por:

- La **topología de arquitectura de red**: describe el número de neuronas en cada una de las capas y la manera como se conectan entre ellas
- La **función de activación**: transforma la combinación de los inputs en una sola señal de salida a ser comunicada a las siguientes neuronas
- El **algoritmo de entrenamiento**: especifica como los pesos de las conexiones se establecen de tal manera que se cohíba o incite la activación de las neuronas en proporción de las señales de entrada. Este algoritmo se llama **back-propagation**, y esta basado en el **descenso de gradiente**.

REDES NEURONALES

Topología: determina la complejidad de las tareas que se pueden aprender

- Número de capas y como se conectan entre ellas. Deep Learning se refiere a la profundidad de las capas.
- **Número de neuronas en cada capa**: salvo por la capa de entrada y salida, depende de la complejidad del problema y calidad de los datos. Cuidado con **overfitting.**
- Dirección del envío de la información
  - Feedforward: hacia adelante
  - Recurrent: se permite retorno (short term memory)



www.analyticsvidhya.com/wp-content/uploads/2016/08/Artificial-Intelligence-Neural-Network-Nodes.jpg

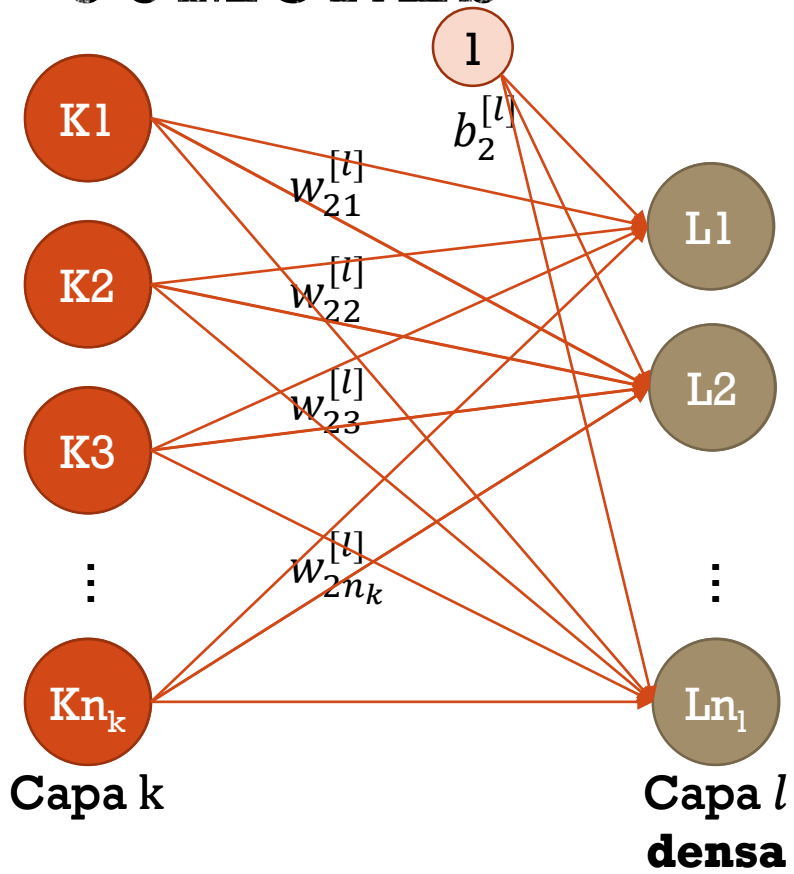
Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

CONVENCIONES

- **Notación**:
  - Los parámetros de los modelos se pueden denotar por θ_i o w_i, y el número total de parámetros es n, teniendo en cuenta n variables independientes. Puede haber un parámetro de sesgo no asociado a una entrada, denotado por θ_0, w_0 o b.
  - El número de instancias de aprendizaje van de 1 a m
  - La i-ésima variable independiente (de un total de n), para la j-ésima instancia de aprendizaje se denota entonces así: $x_i^{(j)}$
  - En capas densas, el subíndice de un parámetro de pesos se compone de la neurona de la capa en cuestión con la neurona de la capa anterior.
  - K denota el número total de clases en un problema de clasificación.
  - El número de instancias de una categoría k se denota entonces m_k

REDES NEURONALES – ARQUITECTURAS DE CAPAS COMUNES

- En una capa **densa** o **fully connected**, cada una de sus neuronas están conectadas con todas las neuronas de la capa anterior (son las que se usan en las redes neuronales tradicionales).
- Cada neurona i de la capa l tiene un peso $w_{ij}^{[l]}$ que la relaciona con cada j-ésima neurona de la capa anterior
- Cada neurona i de la capa l tiene un sesgo $b_i^{[l]}$



Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

REDES NEURONALES – ARQUITECTURAS DE CAPAS COMUNES

- En una capa **densa** o **fully connected**, cada una de sus neuronas están conectadas con todas las neuronas de la capa anterior (son las que se usan en las redes neuronales tradicionales).
- Una capa **convolucional** captura relaciones espaciales de la capa anterior (se utiliza mucho con entradas imágenes, también se puede usar en series de tiempo).
- Una capa puede ser **recurrente**, al considerar como inputs en un siguiente paso de cálculo sus propias salidas de pasos anteriores (se utiliza en señales de audio, secuencias, lenguaje natural, etc.).

REDES NEURONALES

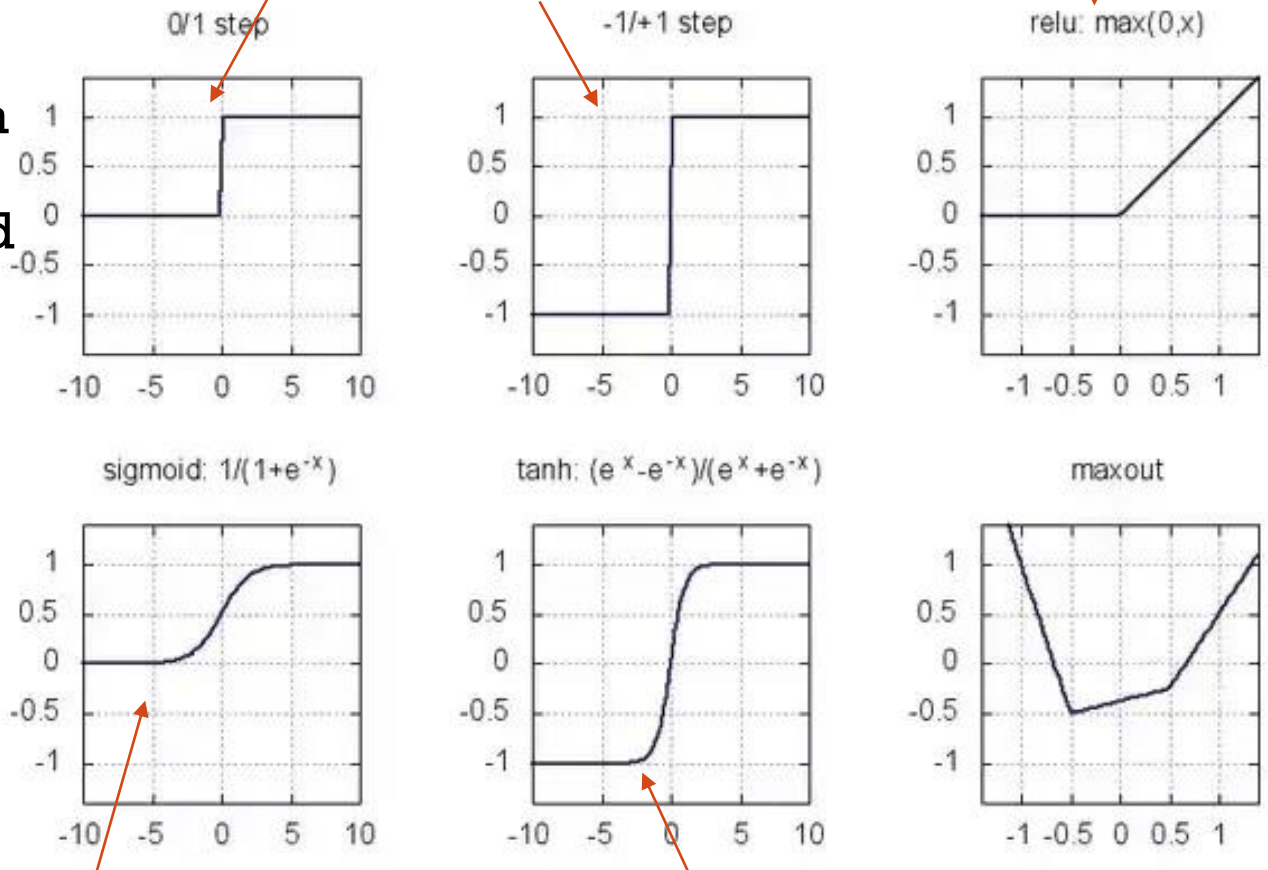
Funciones de activación: mecanismo de procesamiento de la información entrante que permite la propagación de la señal en la red buscando una **no linealidad**.

Se prefieren las que tengan buenas propiedades matemáticas:

- Derivable, sin que se desvanezca el gradiente
- Salida centrada en cero
- Costo computacional bajo

Las más parecidas a la realidad biológica

Muy usada en Deep Learning. Capas intermedias, rapidez



Regresión logística, capa de salida binaria

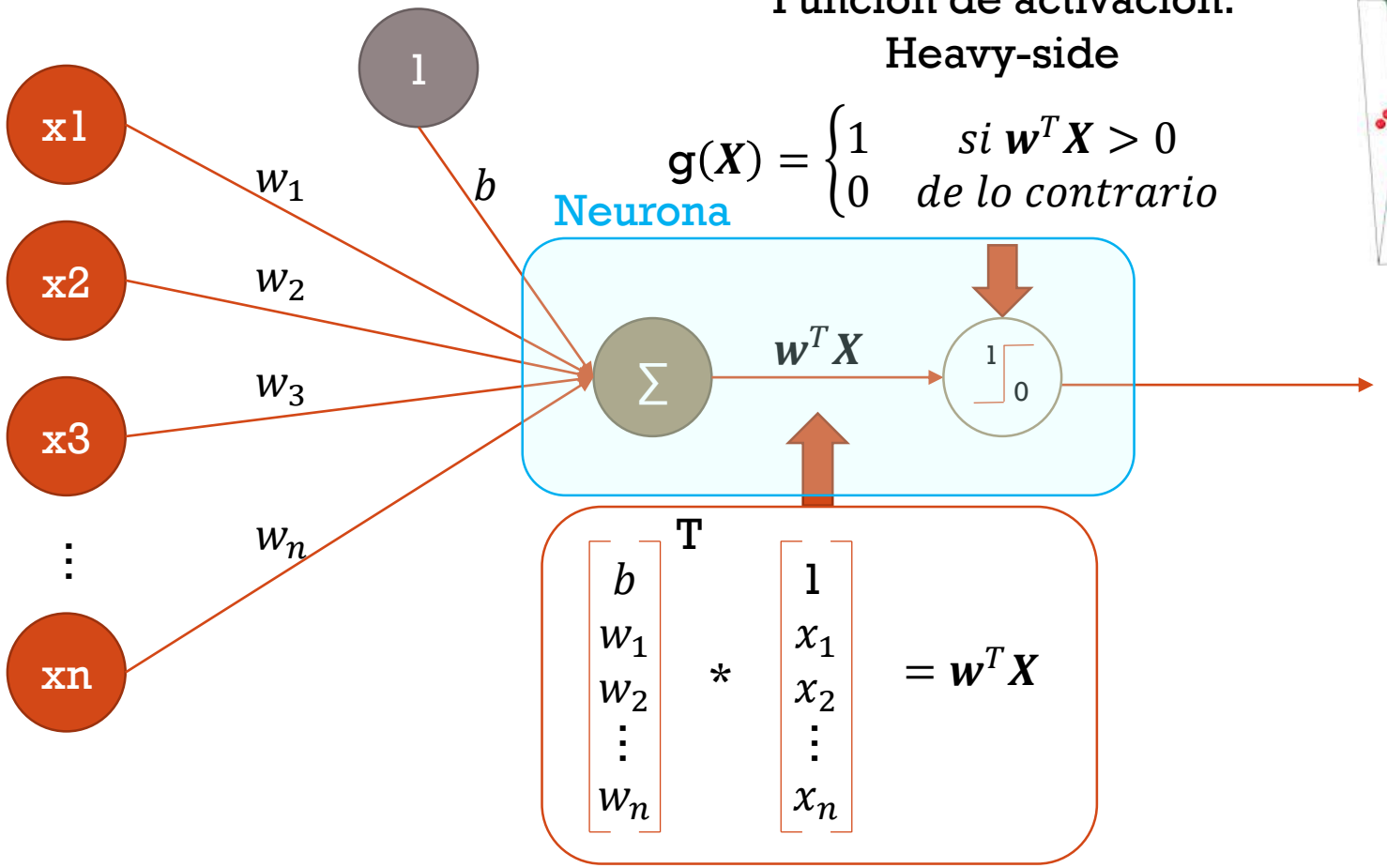
Muy usada, funciona mejor que la sigmoide en las capas escondidas

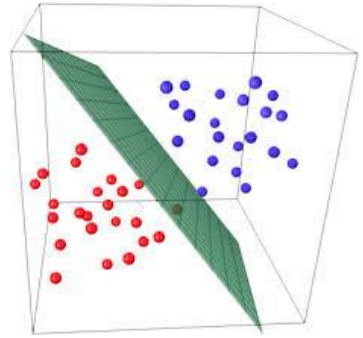
Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

PERCEPTRÓN, 1957 (ROSENBLATT)

Función de activación:
Heavy-side

$$g(X) = \begin{cases} 1 & si\ w^T X > 0 \\ 0 & de\ lo\ contrario \end{cases}$$





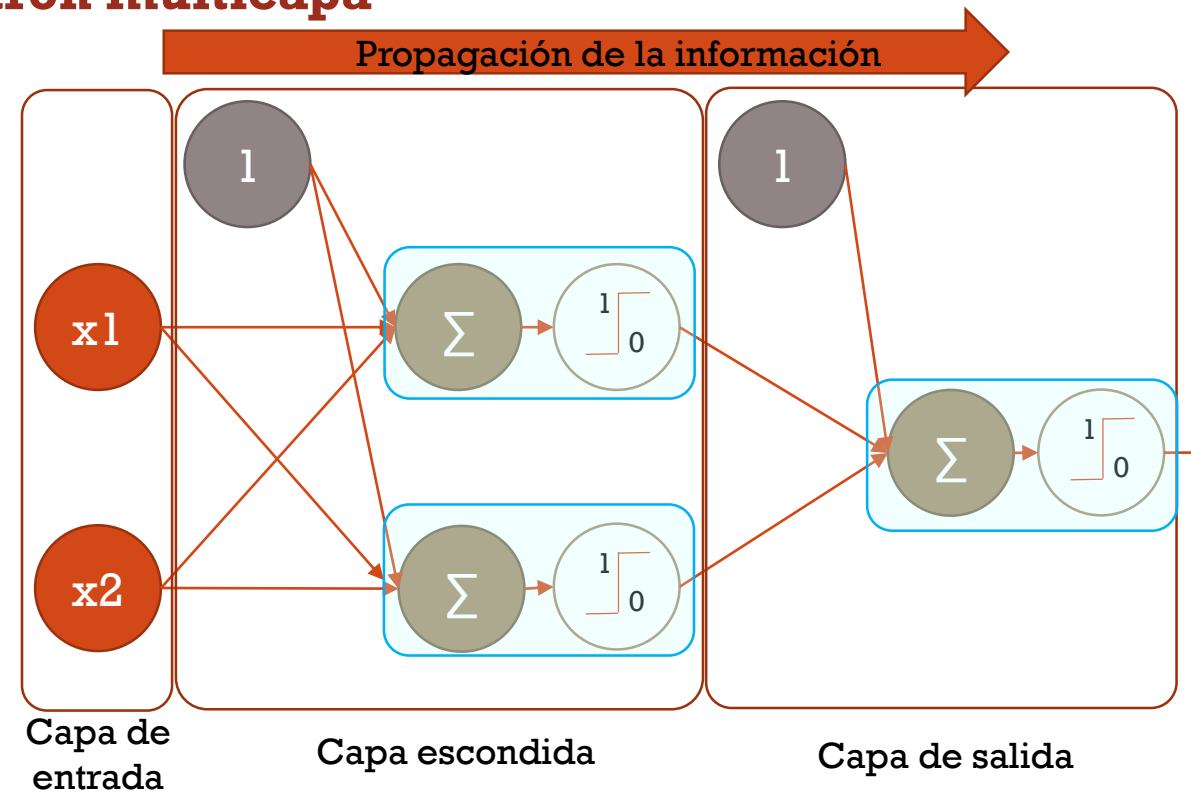
Señal de salida

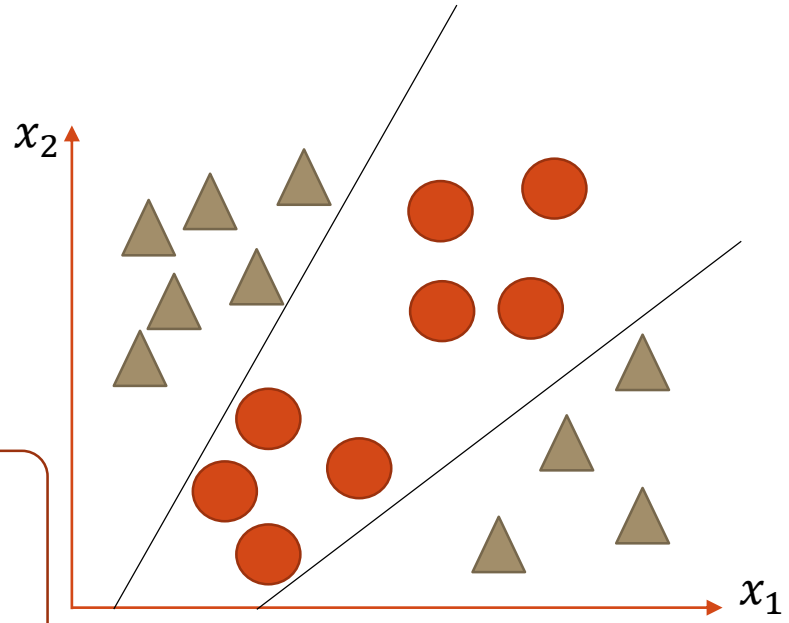
Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

PERCEPTRÓN, 1957

- Imposibilidad de tratar casos no linealmente separables, e.g. XOR, Minsky et Papert, 1969

→ **Perceptrón multicapa**

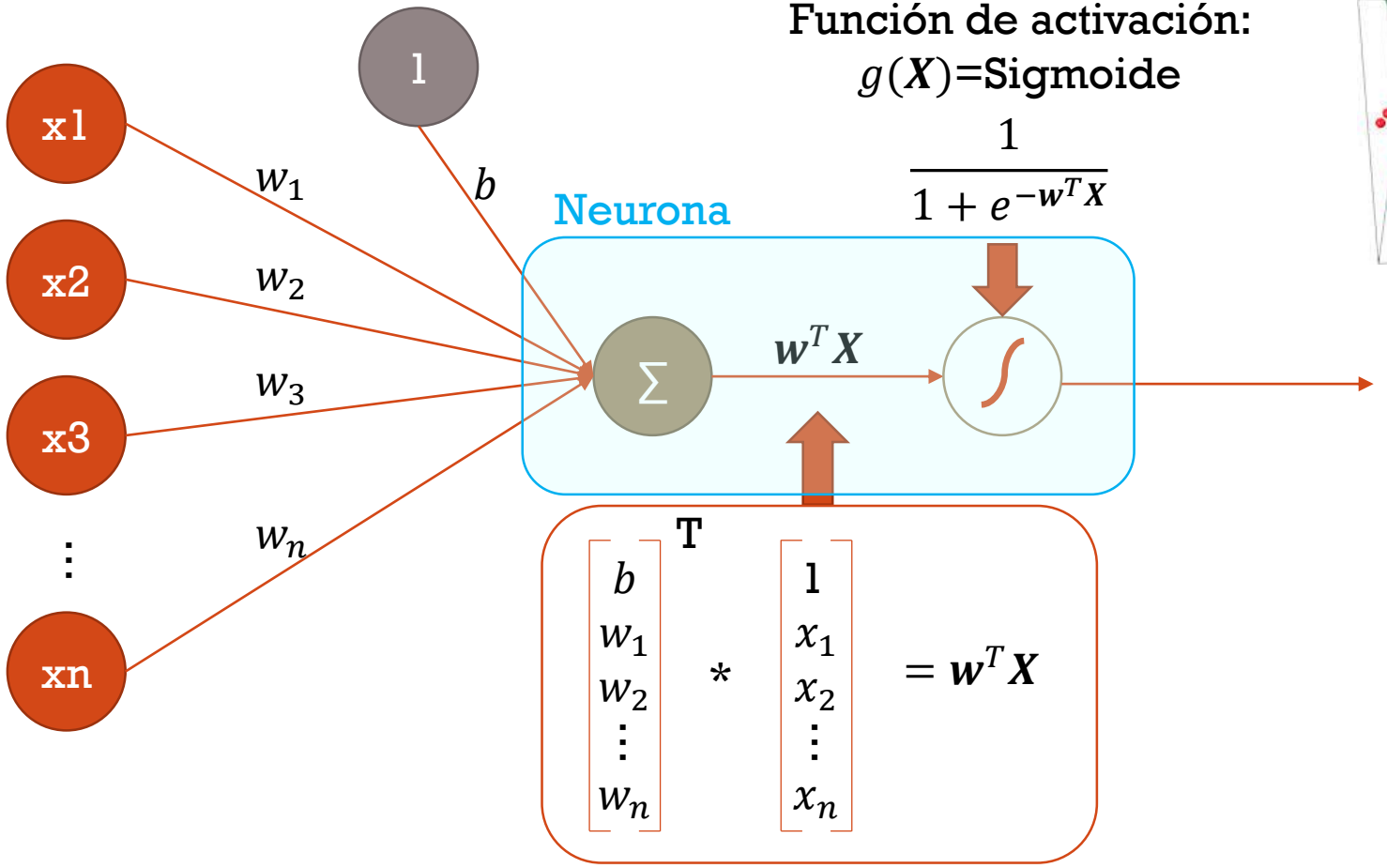


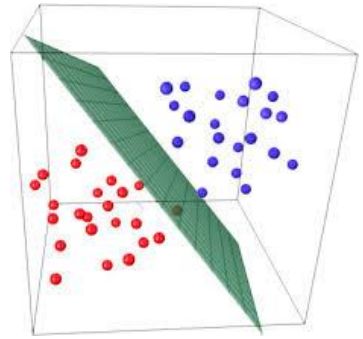


La señal de salida es el resultado de 2 fases:
La agregación lineal y la aplicación de la función de activación

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

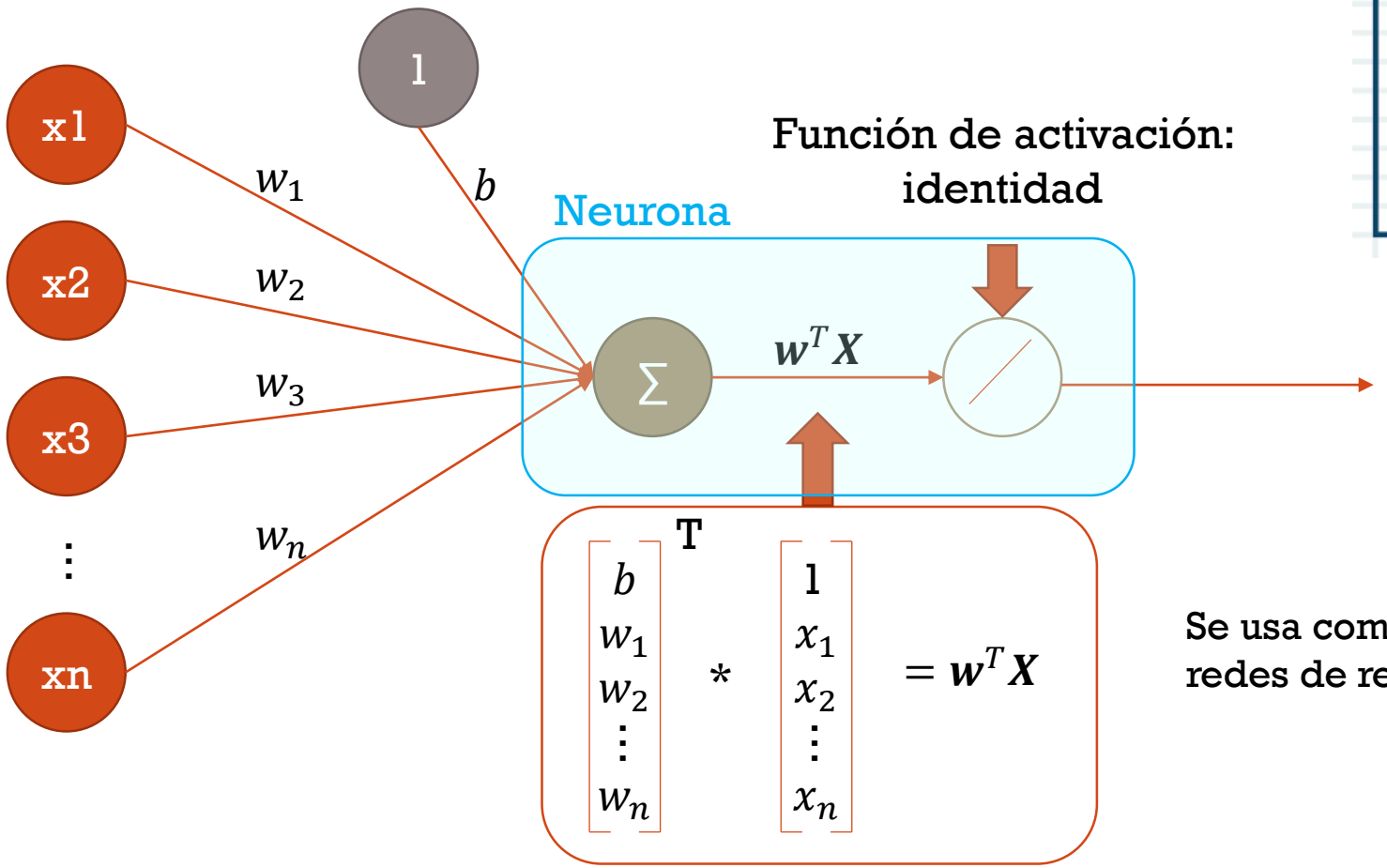
REGRESIÓN LOGÍSTICA

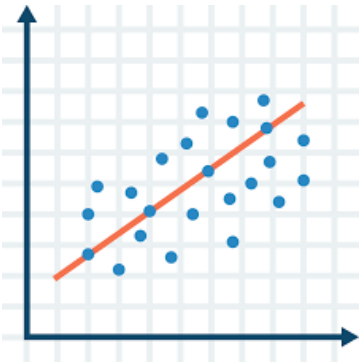




Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

REGRESIÓN LINEAL





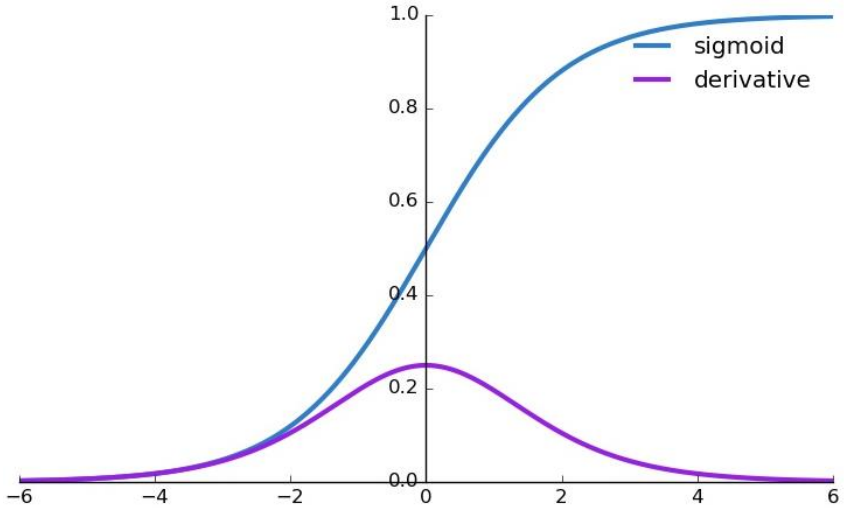
Se usa como neurona de salida en las redes de regresión.

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

FUNCIONES DE ACTIVACIÓN

Funciones de activación: sigmoïde

- Salida en [0; 1]
- Históricamente se usó mucho por su interpretación de sobrepaso de umbral por saturación, similar a la realidad biológica
- Pero:
  - Las neuronas saturadas desvanecen el gradiente
  - Para ayudar a la convergencia, se prefieren salidas centradas en 0
  - La función exponencial es un poco costosa computacionalmente
- Hoy en día:
  - Se prefiere la tanh
  - Se usa solo para una capa de salida binaria



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

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

FUNCIONES DE ACTIVACIÓN

Funciones de activación: softmax

- Clasificación multi categoría (usar solo en la última capa)
- Equivalente a la sigmoide si hay solo dos clases
- Generalización de la sigmoide para mas de dos clases
- Se le conoce también como: MaxEnt (maximum entropy classifier), multinomial logistic regression,
- Salida en [0; 1] para cada salida, suma=1

$$Pr\left(Y^{(i)} = k\right) = \frac{e^{w_k \cdot x^{(i)}}}{\sum_c e^{w_c \cdot x^{(i)}}}$$

En el caso binario:

$$Pr\left(Y^{(i)} = 0\right) = \frac{e^{w_0 \cdot x^{(i)}}}{e^{w_0 \cdot x^{(i)}} + e^{w_1 \cdot x^{(i)}}} = \frac{e^{(w_0 - w_1) \cdot x^{(i)}}}{e^{(w_0 - w_1) \cdot x^{(i)}} + 1} = \frac{e^{-w \cdot x^{(i)}}}{1 + e^{-w \cdot x^{(i)}}}$$

$$Pr\left(Y^{(i)} = 1\right) = \frac{e^{w_1 \cdot x^{(i)}}}{e^{w_0 \cdot x^{(i)}} + e^{w_1 \cdot x^{(i)}}} = \frac{1}{e^{(w_0 - w_1) \cdot x^{(i)}} + 1} = \frac{1}{1 + e^{-w \cdot x^{(i)}}}$$

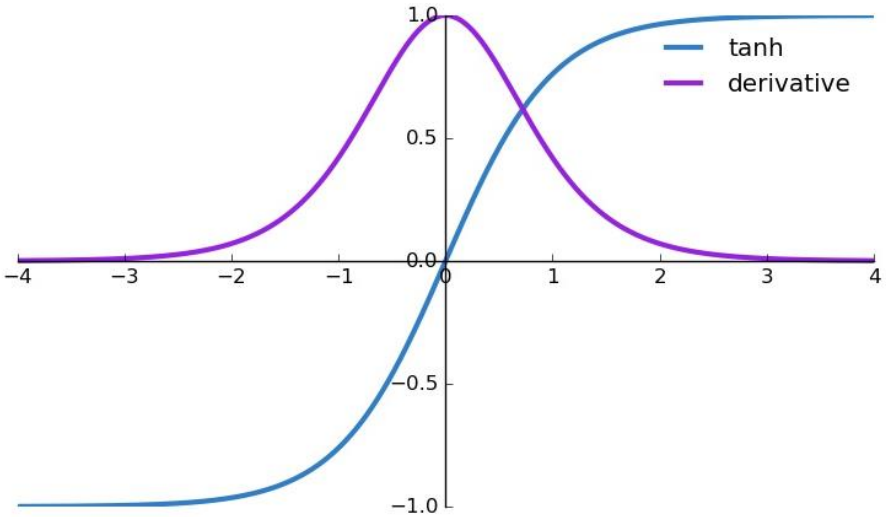
con $w = -(w_0 - w_1)$

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

FUNCIONES DE ACTIVACIÓN

Funciones de activación: tanh (tangente hiperbólica)

- Salida en [-1; 1] → centrada en 0
- Pero (los otros problemas de la sigmoïde):
  - Las neuronas saturadas desvanecen el gradiente
  - La función exponencial es un poco costosa computacionalmente
- Hoy en día:
  - Se prefiere a la sigmoïde
  - Se usa mucho en redes recurrentes LSTM



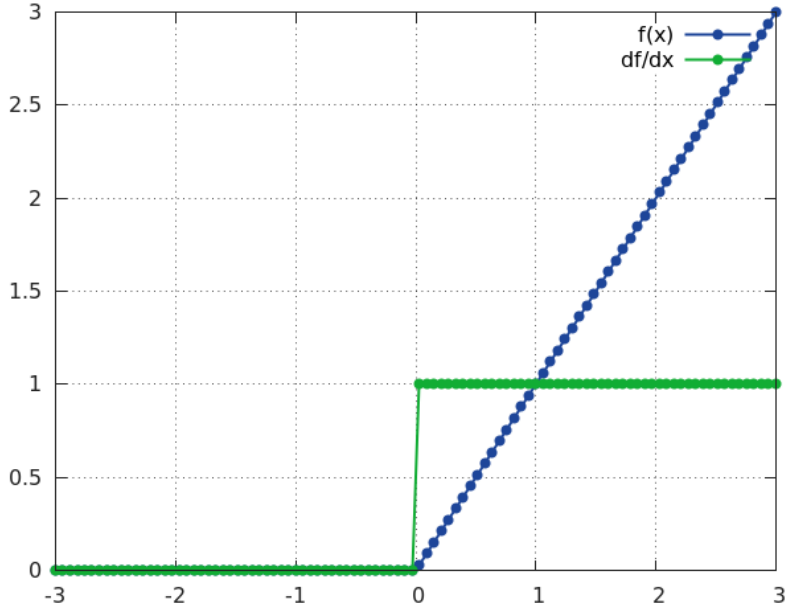
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

FUNCIONES DE ACTIVACIÓN

Funciones de activación: ReLU (Rectified Linear Unit)

- Salida en [0; +∞], funciona como una compuerta
- Converge mucho más rápido (x6) que la sigmoide y tanh, eficiente computacionalmente
- Pero (los otros problemas de la sigmoïde):
  - Se desvanece el gradiente en los valores negativos
  - Salidas no están centradas en 0
  - Gradiente de x=0?
  - Neuronas ReLUs pueden "morir" y nunca reactualizarse
- Hoy en día:
  - Es la más utilizada



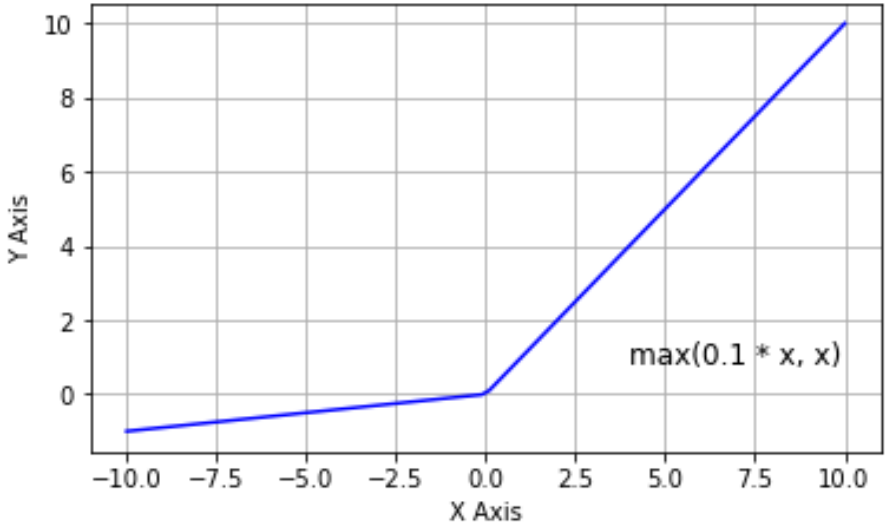
$$\mathrm{ReLU}(x) = max(0, x)$$

Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

FUNCIONES DE ACTIVACIÓN

Funciones de activación: Leaky ReLU

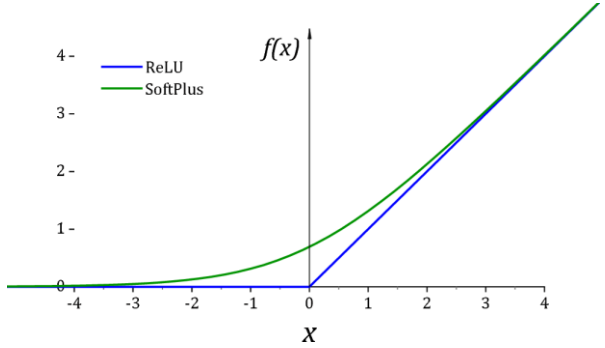
- Salida en [- ∞; +∞], modificación de la
- No se desvanece el gradiente
- El valor de α puede ser una constante, o puede ser otro parámetro a optimizar (Parametric ReLU – PReLU)
- Pero (los otros problemas de la sigmoïde):
  - Salidas no están centradas en 0, pero están mejor que las neuronas ReLU
  - Gradiente de x=0?
- Hoy en día:
  - Alternativa a la ReLU



$$\text{Leaky ReLU}(x) = max(\alpha * x, x)$$

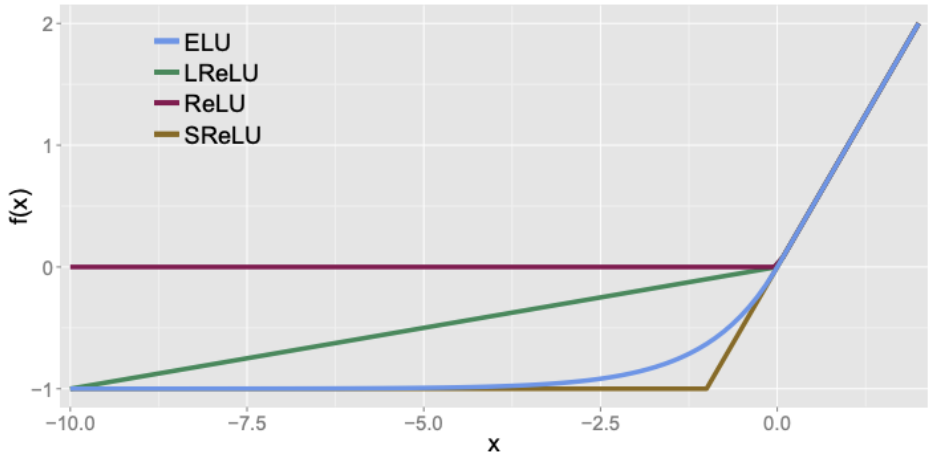
Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

FUNCIONES DE ACTIVACIÓN

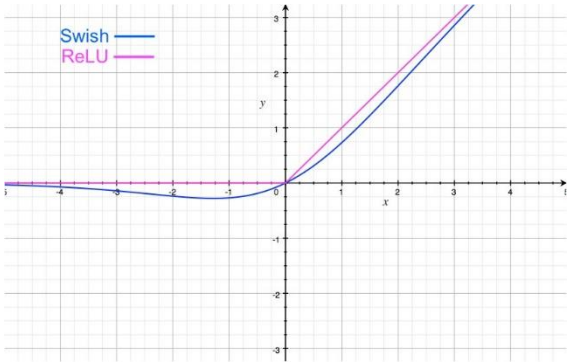


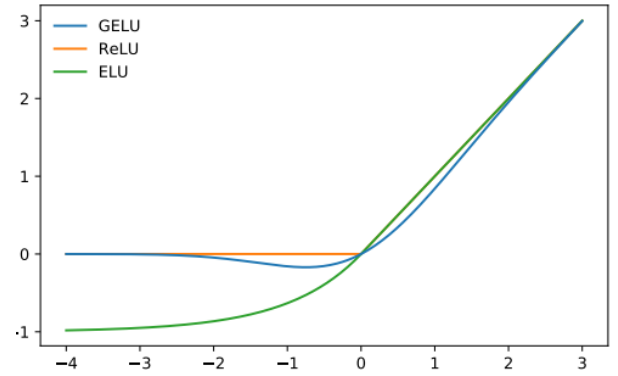
Otras funciones de activación :

- pReLU: Parametric Rectified Linear Unit
- Softplus: $\mathrm{f}(z) = \ln(1 + e^z)$
- ELU: Exponential Linear Unit $\mathrm{f}(z) = \begin{cases} z & , \ si\ z > 0 \\ \alpha \cdot (e^z - 1), & si\ z < 0 \end{cases}$
- SReLU: Shifted ReLU
- SELU: Scaled Exponential Linear Unit
- SWISH: $\mathrm{f}(z) = z \cdot sigmoid(z)$
- GELU: Gaussian Error Linear Unit



https://www.tensorflow.org/api_docs/python/tf/keras/activations



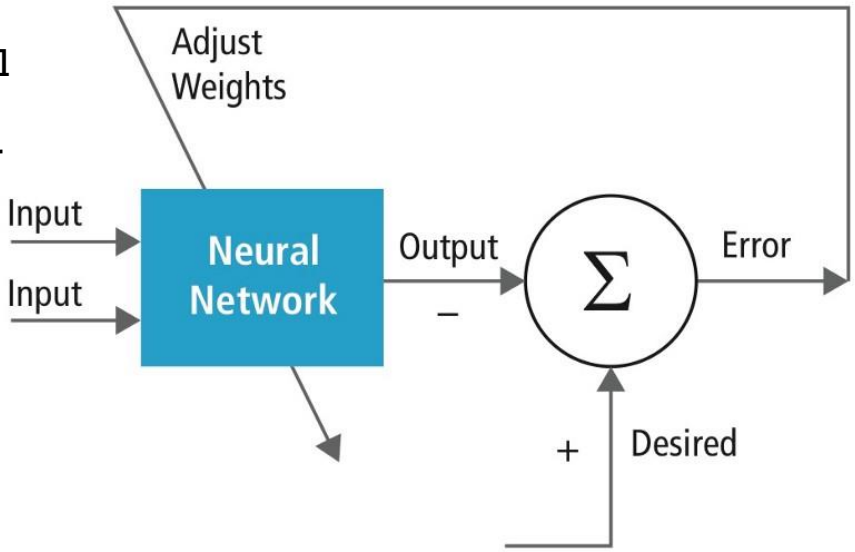


Javier Diaz Cely, PhD
UNIVERSIDAD ICESI

BACK-PROPAGATION

Back-propagation: el algoritmo más común para entrenar una red neuronal feed-forward de múltiples capas, 1986 (Hinton).

- 2 fases
  - **Forward**: para cada ejemplo propagar la información hasta llegar al final, calcular el error de predicción.
  - **Backward**: modificar los pesos de la capa inmediatamente anterior de tal manera que se reduzcan los errores. Continuar el proceso con las capas anteriores.
- Basado en la propagación del error y el **descenso de gradiente** (derivadas parciales de las funciones de activación que van en la dirección de la reducción del error). Necesidad de que las funciones de activación sean derivables.
- Computacionalmente intensivo
- Influencia de la inicialización aleatoria de las neuronas
- Tasa de aprendizaje a establecer



towardsdatascience.com

INTUICIÓN

https://playground.tensorflow.org/

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

TALLER: REDES NEURONALES (SCIKIT-LEARN)

- Desarrollar el taller de redes neuronales para la detección del cancer

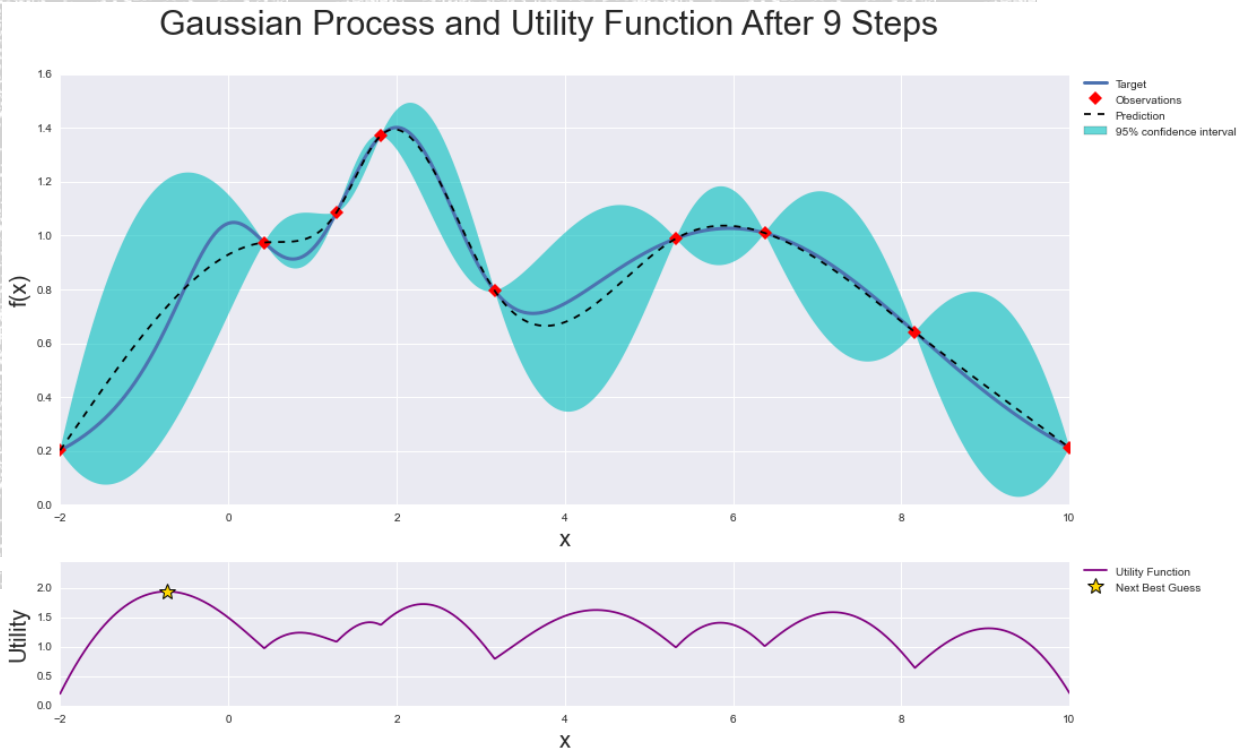
TALLER: REDES NEURONALES (SCIKIT-LEARN)

- Desarrollar el taller de redes neuronales para la predicción de la fuerza del concreto

REFERENCIAS

- *Python Data Science Handbook,* Jake VanderPlas, 2017, O'Reilly
- *Python Machine Learning (2nd ed.),* Sebastian Raschka, 2017, Packt

BAYESIAN OPTIMISATION



https://github.com/fmfn/BayesianOptimization

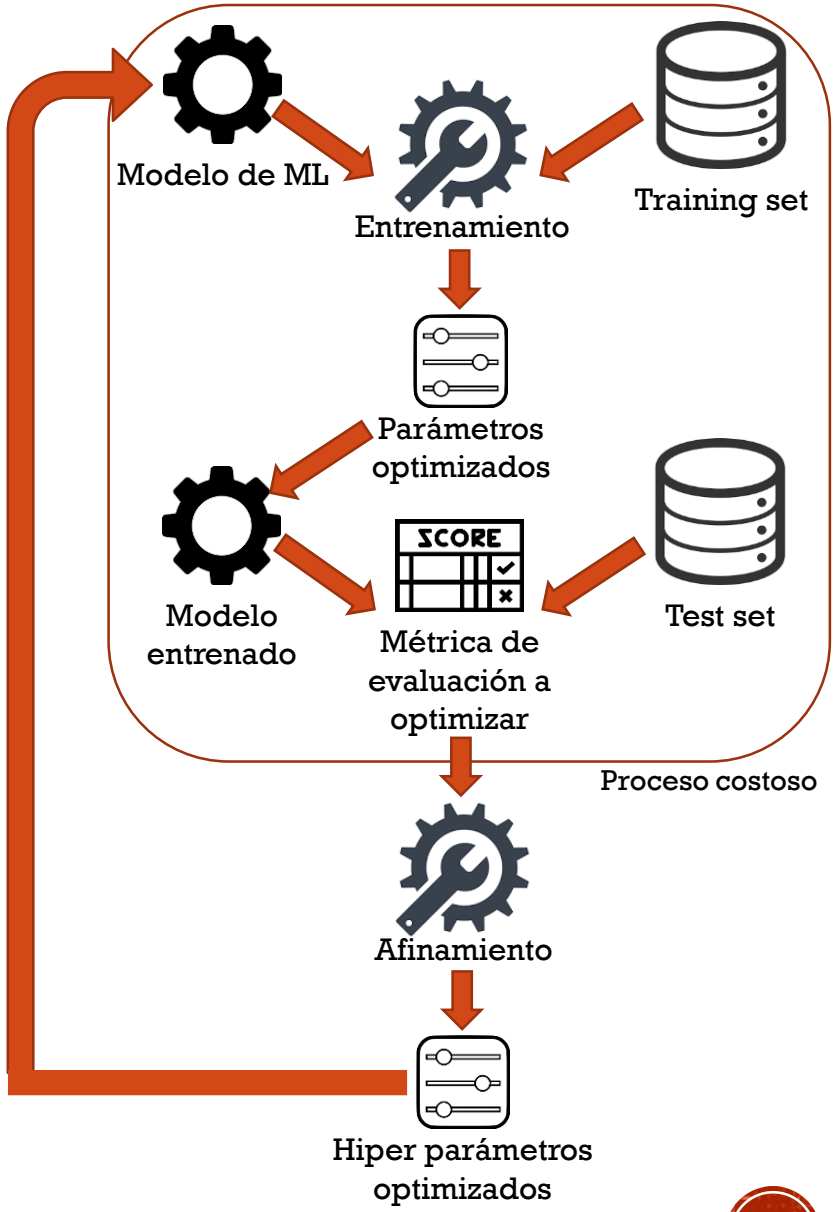
UNIVERSIDAD ICESI

OPTIMIZACIÓN DE MODELOS

Optimización de modelos

- **Entrenamiento de modelos:** búsqueda de los mejores valores de los parámetros, lo realiza el proceso de entrenamiento del modelo
- **Afinamiento de modelos:** búsqueda de los mejores valores de los hiper parámetros. Se realiza a partir de:
  - **Naïve Grid search**: se definen posibles valores de los hiperparámetros a combinar por fuerza bruta
  - **Random search**: búsqueda aleatoria de combinaciones de hiperparámetros dentro de rangos definidos
  - **Optimización bayesiana**: utilizar procesos gausianos para estimar los mejores valores de hiper parámetros teniendo buscando un balance entre exploración y explotación.

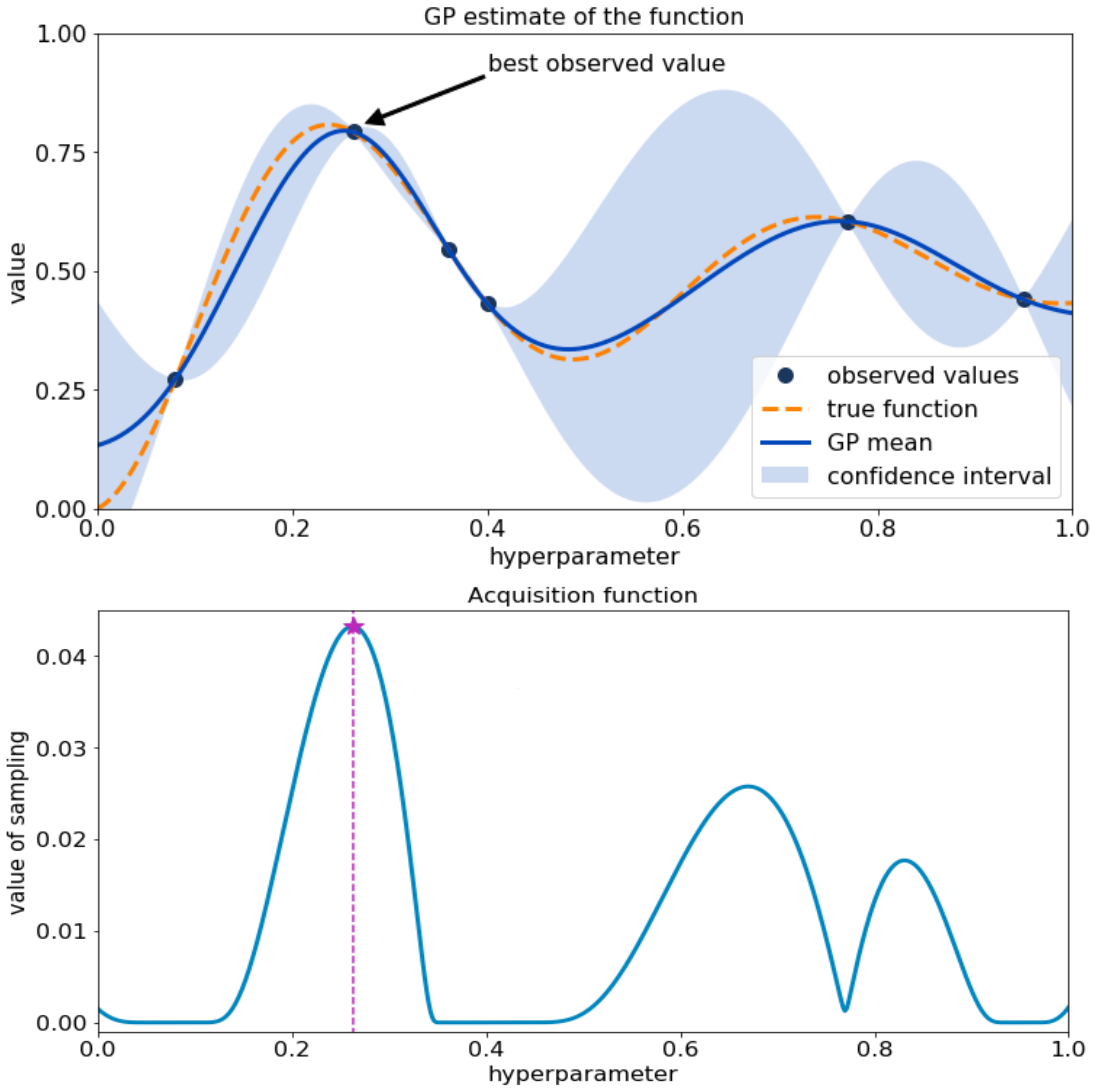
P(función de evaluación | hiper parámetros)



UNIVERSIDAD ICESI

OPTIMIZACIÓN BAYESIANA

- **Proceso Gaussiano** secuencial (no paralelizable)
  1. Se parte de la evaluación de unas cuantas configuraciones aleatorias
  2. Se hace un fit de un proceso gaussiano con las configuraciones conocidas. Se obtiene una distribución a posteriori de la función objetivo (función promedio, función de covarianza)
  3. Se sustituye el problema de optimización por el de maximizar una función de **función de adquisición** conocida y sencilla para escoger la siguiente configuración de parámetros
  4. Se evalúa la función objetivo sobre la configuración escogida
  5. Se repite desde el paso 2, hasta cumplir con un cierto criterio (tiempo, # iteraciones)
  6. Se escoge la mejor combinación de parámetros evaluados



UNIVERSIDAD ICESI

OPTIMIZACIÓN BAYESIANA

- La función de adquisición
  - Tiene en cuenta lo que ya se sabe de la función objetivo
  - Trata de minimizar el número de pasos
  - Considera una **función de adquisición** sustituta a maximizar que balancea entre:
    - **Exploración**: buscar alrededor de configuraciones óptimas ya evaluadas
    - **Explotación**: buscar regiones del espacio de hiperparámetros de gran incertidumbre (varianza)
  - Varias alternativas: expected improvement, upper confidence bound, probability of improvement
  - Mayor costo de escoger los siguientes parámetros a evaluar, pero menor número de evaluaciones a realizar
  - https://github.com/fmfn/BayesianOptimization/blob/master/examples/bayesian_optimization.gif

REFERENCIAS

- Lecture 16.3 – Bayesian optimization of hyper parameters. Deep Learning, University of Toronto, Geoffrey Hinton: https://www.youtube.com/watch?v=i0cKa0di_lo
- Bayesian Hyperparameter Optimization for Keras – adf T81-558: Applications of Deep neural networks, Washington University, Jeff Heaton: https://www.youtube.com/watch?v=sXdxyUCCm8s
- Bayesian Optimization, Fernando Nogueira: https://github.com/fmfn/BayesianOptimization