



Aprendizaje Automático

Introducción a las Redes Neuronales Artificiales

Viviana Cotic
1er cuatrimestre 2019



Redes neuronales artificiales (RNA -RN, ANN, NN-)

Aprendizaje supervisado

- Perceptrón simple
- Redes feedforward multicapa
- ...

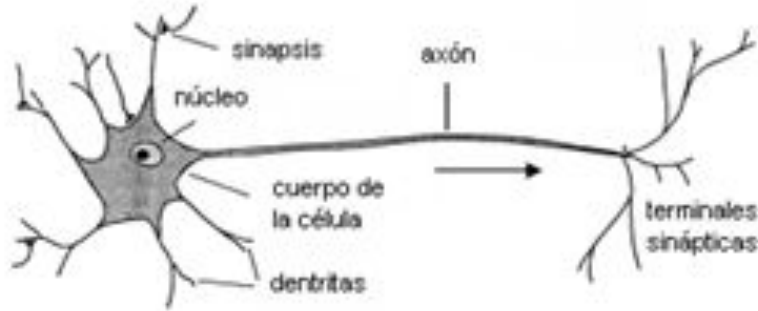
• **Aprendizaje no supervisado**

- Aprendizaje Hebbiano
- Aprendizaje Competitivo
- Mapas Auto-Organizados

Redes neuronales

- inspiradas en un modelo biológico
- conexión de neuronas, unidades de procesamiento sencillas
- opera en paralelo, es robusto ante fallas

La Neurona

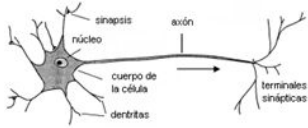


Esquema de una neurona

procesador de información muy simple:

- Canal de entrada: dendritas.
- Procesador: soma.
- Canal de salida: axón.

Modelo matemático de una neurona



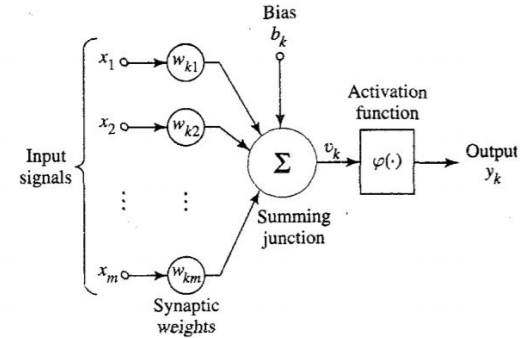
Esquema de una neurona

Introduction to the theory of Neural Computation.
Hertz et al.

Modelo matemático de una neurona:

neurona: **unidad básica de procesamiento de información** de una red neuronal. Un modelo de una neurona está dado por:

- **cjto. de conexiones** (sinapsis) con pesos asociados.
- Una **señal x_j** conectada a la **neurona k** se multiplica por **peso w_{kj}**
- **sumador de señales** de entrada pesadas por la sinapsis, arroja combinación lineal de las entradas



Esquema del modelo matemático de una neurona -
Neural Networks, Haykin.

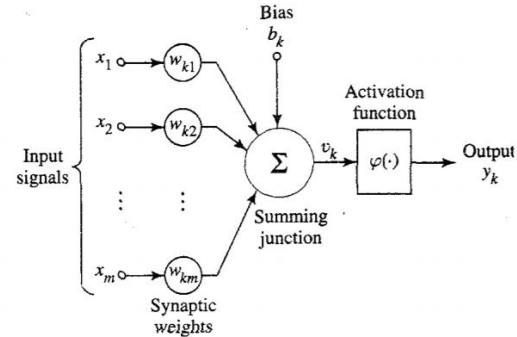
- una **función de activación** que limita la amplitud de salida de una neurona
- un **umbral b_k** , para variar la actividad de la neurona

La neurona

$$O_k = g\left(\left(\sum_{i=1}^m w_{ki} \cdot \xi_i\right) - b_k\right),$$

Ej:

$$g = \text{sgn}\left(\sum_{i=1}^m w_{ki} \cdot \xi_i - b_k\right)$$

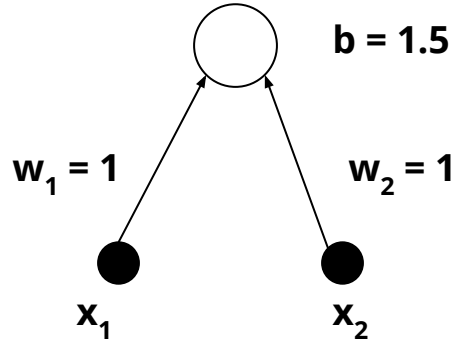


El perceptrón simple resuelve problemas linealmente separables.

Ej. para funciones booleanas: AND, OR, NOT. No funciona para XOR.

Ejemplo: AND

Perceptrón que implementa AND



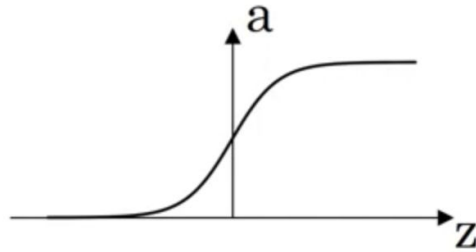
Función de activación: Signo.

En pizarrón:

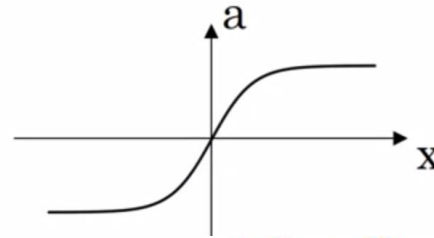
- Tabla de verdad
- Gráfico de AND

Funciones de activación

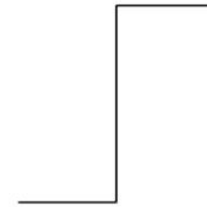
Algunas funciones de activación



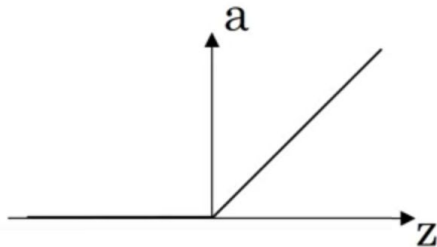
sigmoid: $a = \frac{1}{1 + e^{-z}}$
(logística)



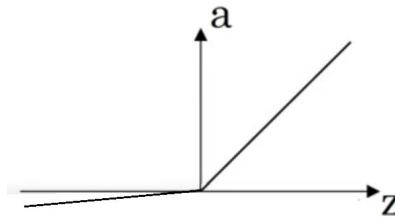
tanh: $a = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$



Escalón
 $s(x) = \text{sgn}(x)$



ReLU: $a = \max(0, z)$



leaky ReLU: $a = \max(0.01 z, z)$

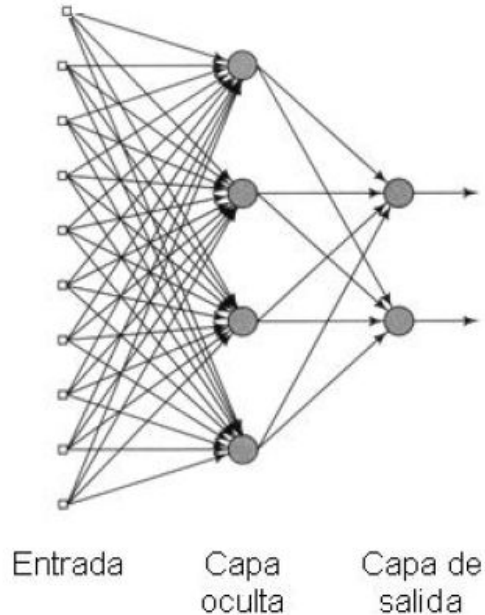
Lineal

$s(x) = x$



Redes neuronales

Arquitectura feed forward

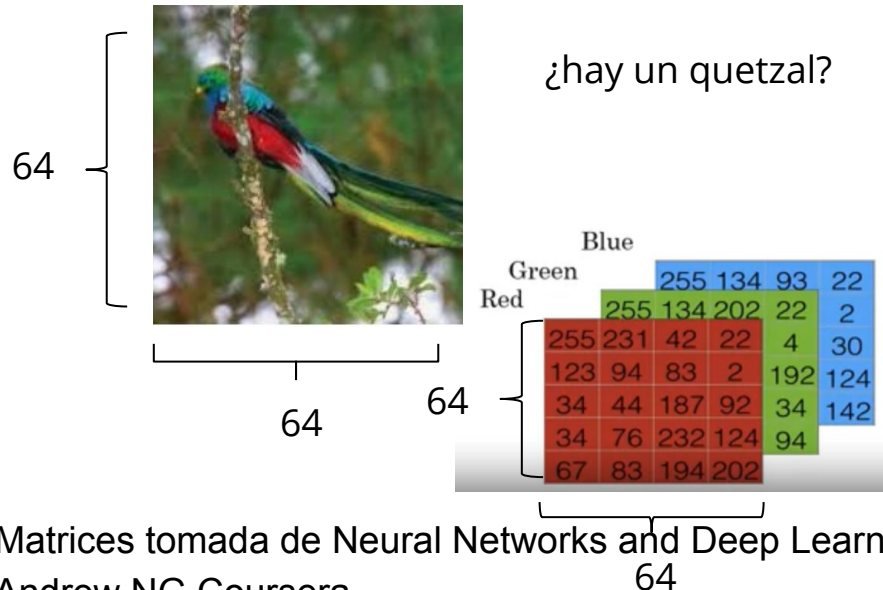


Distintas arquitecturas.
Aquí

- perceptrón multicapa
- conexión total

Regresión logística como red neuronal

- algoritmo de clasificación binaria
- representación de imagen en computadora



Matrices tomada de Neural Networks and Deep Learning,
Andrew NG Coursera

Entradas

$x = 64 * 64 * 3 = 12288$ (vector de entrada) = n_x

Notación (x,y) . $x \in \mathbb{R}^{n_x}$

$y \in \{0,1\}$

m = cant. conjuntos de entrenamiento

Regresión logística, funciones de pérdida y de costo

Dado $X \in \mathbb{R}^{n \times}$, **quiero encontrar** $y^\wedge = P(y = 1 \mid x)$

$$y^\wedge = \sigma(w^T x + b) \quad \sigma(z) = 1/(1 + e^{-z})$$

Parámetros a aprender: $W \in \mathbb{R}^{n \times}$, $b \in \mathbb{R}$

Función de pérdida:

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)} \cdot \log \hat{y}^{(i)} - (1 - y^{(i)}) \cdot \log(1 - \hat{y}^{(i)})$$

Función de costo:

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Algoritmo de descenso por gradiente

Teníamos

$$\hat{y} = \sigma(w^T x + b), \quad \sigma(z) = \frac{1}{1+e^{-z}}$$

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Quiero encontrar w, b que minimicen $J(w, b)$

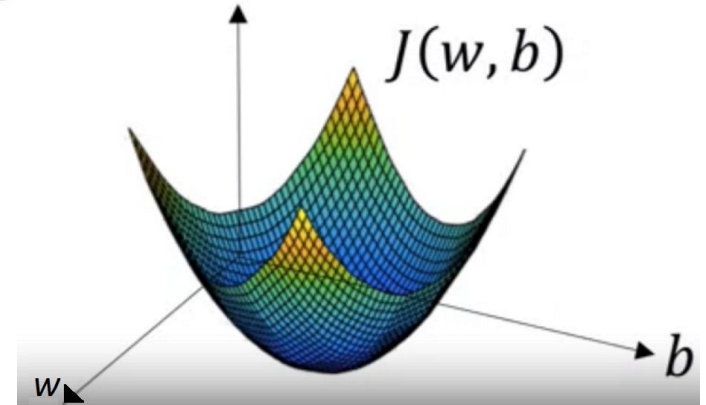
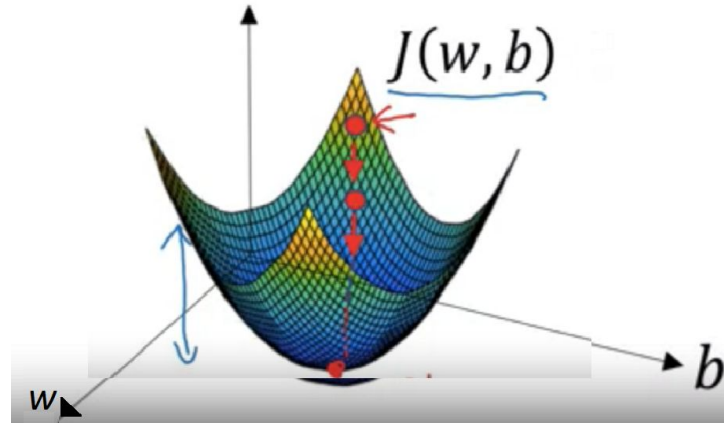


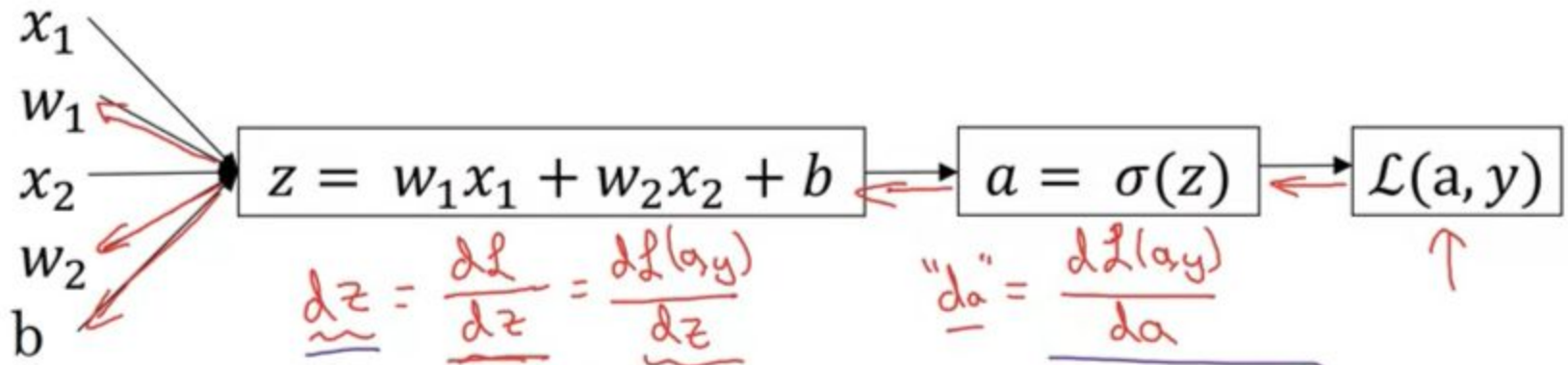
Gráfico tomado de Neural Networks and Deep Learning,
Andrew NG Coursera

Algoritmo de descenso por gradiente

$$w \leftarrow w - \eta \frac{\partial J(w,b)}{\partial w}$$

$$b \leftarrow b - \eta \frac{\partial J(w,b)}{\partial b}$$





$$\underline{\underline{dz}} = \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}(a, y)}{dz}$$

$$\text{"da"} = \frac{d\mathcal{L}(a, y)}{da}$$

$$= \frac{d\mathcal{L}}{da} \cdot \frac{da}{dz} = (a - y) \cdot a(1-a) = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \text{"dw}_1\text{"} = x_1 \cdot dz$$

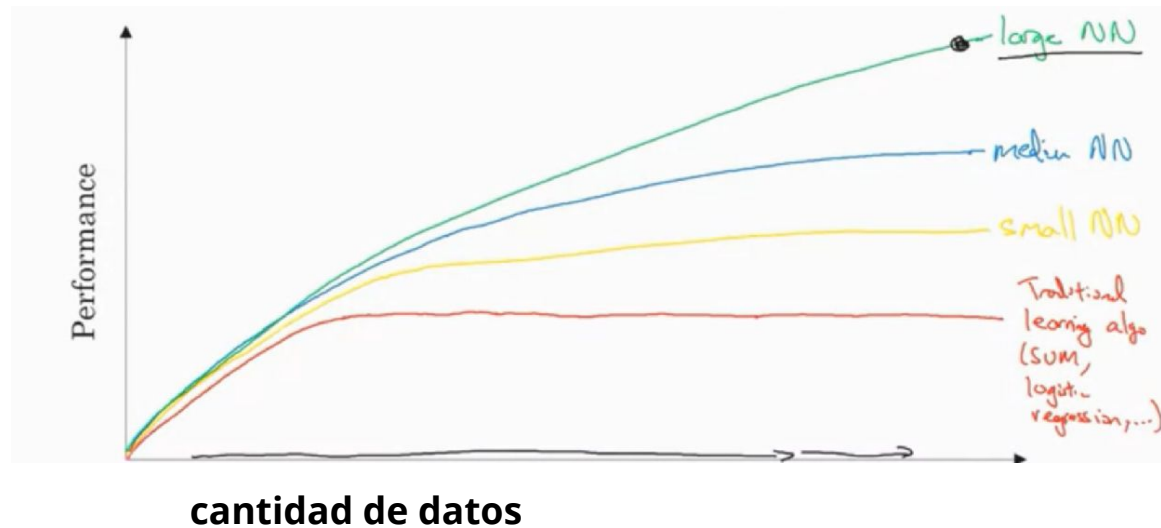
$$dw_2 = x_2 \cdot dz \quad db = dz$$

$$\begin{aligned} w_1 &:= w_1 - \alpha dw_1 \\ w_2 &:= w_2 - \alpha dw_2 \\ b &:= b - \alpha db \end{aligned}$$

Arquitecturas de redes neuronales

- Estándares
- Profundas
 - Convolutional Neural Networks (CNN): Imágenes
 - Recurrent Neural Networks (RNN): Texto, Habla
 - Híbridas

Resurgimiento de las RN



- Más **datos** disponibles
- Más poder de **cómputo GPU**
- Cambios en **algoritmos ReLU**

Tomado de Neural Networks and Deep Learning, Andrew NG Coursera

Parámetros de una red neuronal

- función de activación
- inicialización de pesos
- cantidad de capas ocultas
- conexión entre capas
- tasa de aprendizaje (learning rate)

Hay que evaluarlas en cada caso.

Resumen

Introducción a las redes neuronales artificiales (RN):

- Inspiración biológica
- Modelo matemático. Funciones de activación
- Arquitectura feed-forward. Perceptrón simple y perceptrón multicapa
- Función de pérdida, función de costo
- Método del descenso por gradiente
- Existen variantes de descenso por gradiente para acelerar la convergencia
- Forward y backward pass
- Distintas arquitecturas de RN y resurgimiento de RN

Bibliografía

Capítulos de libros:

Mitchell, cap. 4

Libros enteros:

[Neural Networks. A comprehensive foundation.](#) Haykin

[Introduction To The Theory Of Neural Computation.](#) Hertz, Krogh, Palmer

[Deep Learning.](#) Goodfellow, Bengio, Courville

Curso online:

[Neural Networks and Deep Learning.](#) Coursera.

[Machine Learning.](#) Coursera. Semanas 4 y 5.

[Tensor Flow Playground.](#)