REDES NEURONALES

M. Sc. Jonathan Domínguez Aldana

AGENDA

01 **Modelo biológico**

Anatomía y fisiología neuronal.

02 **La neurona idealizada**

Modelos matemáticos y proceso de aprendizaje.

03 **Redes neuronales**

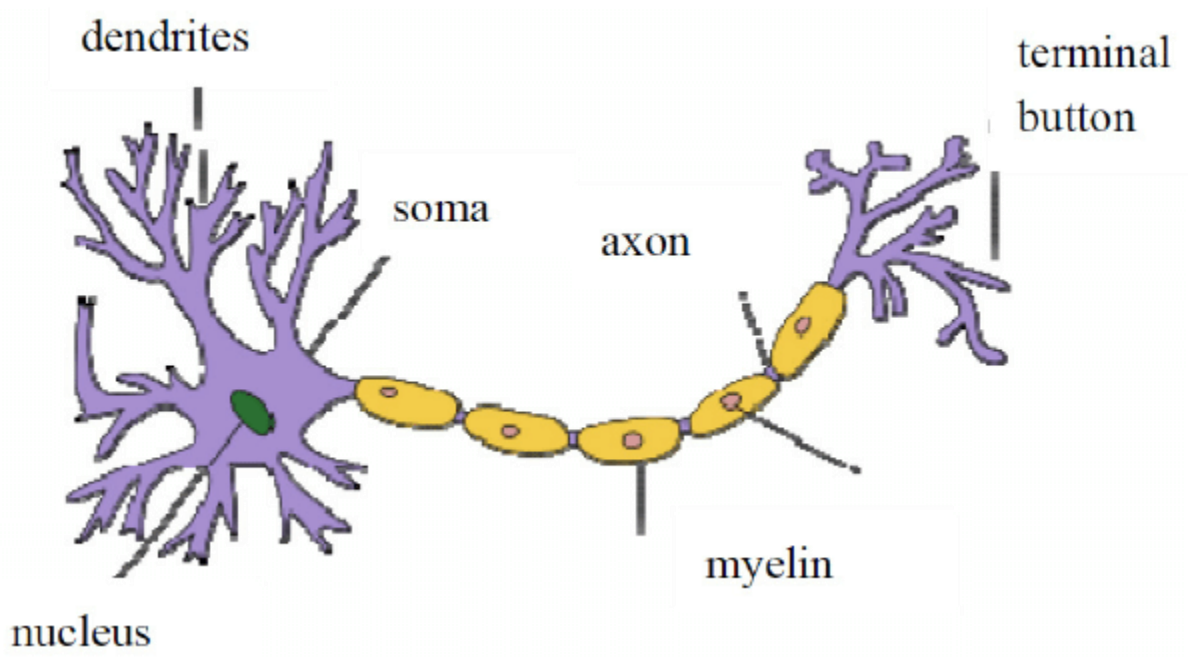
Modelo, Propagación hacia delante, Retropropagación

04 **Redes neuronales convolucionales**

Convolución, ejemplo de una capa, red, motivación.

MODELO BIOLÓGICO

ANATOMÍA
DE LA NEURONA BIOLÓGICA



Dendritas:
Reciben señales electro-químicas de neuronas adyacentes.

Axones:
Hacen **contacto** con las **dendritas** en los espacios **sinápticos** para **envíar impulsos** de **actividad**, ante una **despolarización** de la **membrana**.

Sinápsis:
Transmisores químicos se **difunden** y se **adhieren** a los **receptores** de **membrana** para **cambiar** la **morfología** de la neurona post-sináptica permitiendo el **paso** de **iones**.

Tortora, G., & Derrickson, B. (2011). Principios de Anatomía y Fisiología (13°). México: Editorial Médica Panamericana S. A.

FISIOLOGÍA

DE LA ACTIVIDAD NEURONAL

Entradas:
Todas las **neuronas reciben señales** de **entrada** de otras neuronas.

Ponderaciones:
El **efecto** que tienen **cada señal** de **entrada,** se encuentra **controlado** por un **peso sináptico**.

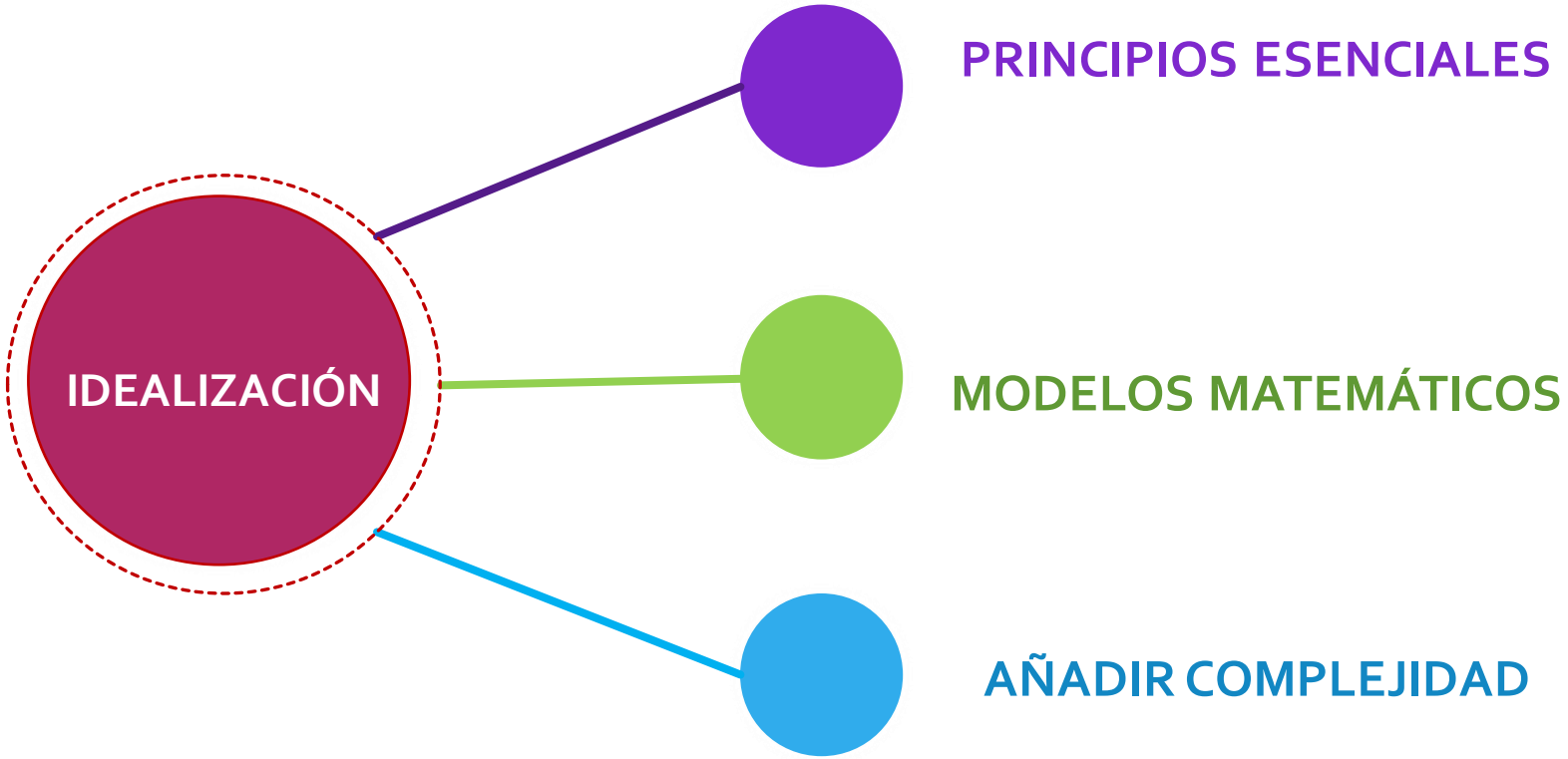
Adaptación:
Todos los **pesos sinápticos** de la red neuronal biológica se **adaptan** para lograr el **proceso** de **aprendizaje**.

Funciones neuronales:
Reconocimiento de **objetos, entendimiento** del **lenguaje,** planeación y **sentimientos**.

Tortora, G., & Derrickson, B. (2011). Principios de Anatomía y Fisiología (13°). México: Editorial Médica Panamericana S. A.

LA NEURONA IDEALIZADA

¿PORQUÉ SE NECESITA UN MODELO IDEALIZADO?



MODELO MATEMÁTICO

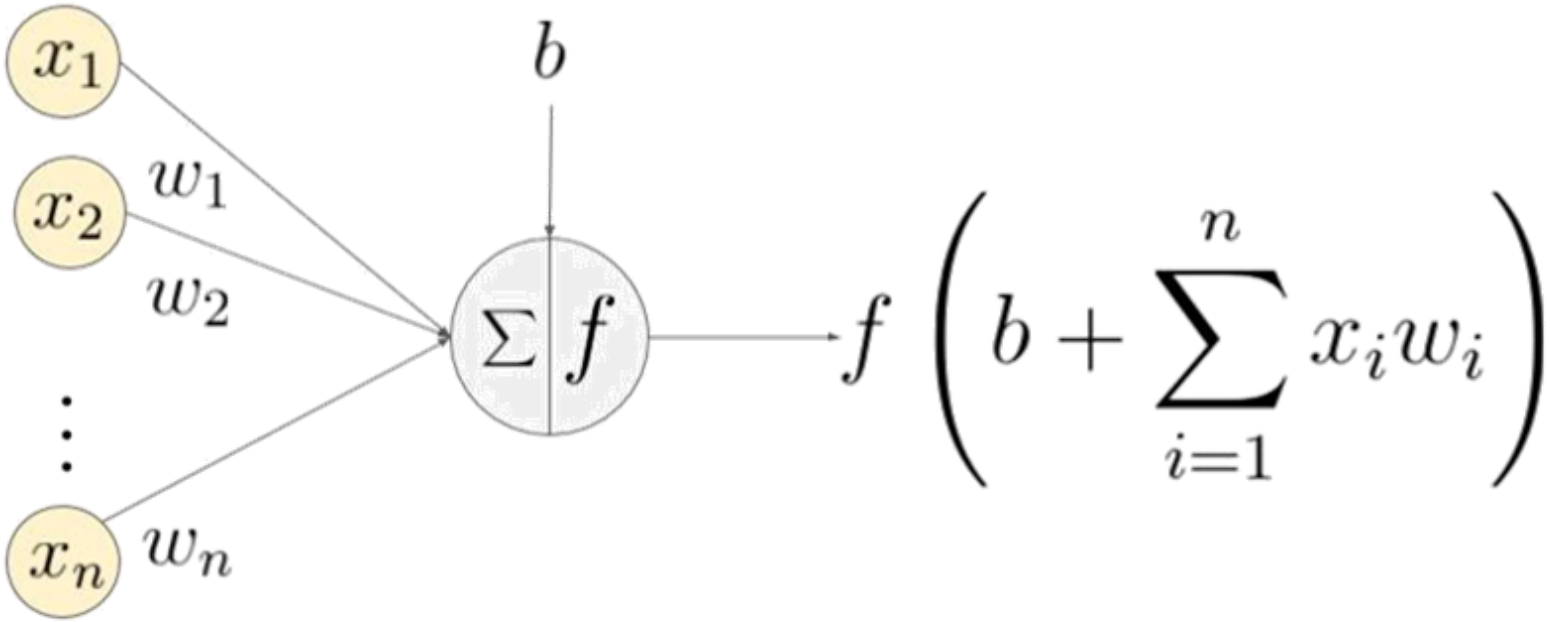
INTRODUCCIÓN A LA NOTACIÓN

Variables	Descripción
$\vec{x} \in \mathbb{R}^n$	Vector de características (variable de entrada)
$y \in \mathbb{R}$	Variable de respuesta (variable de salida)
n	Dimensión del vector de características
m	Número de datos de entrenamiento
$\left(\overrightarrow{x^{(i)}}, y^{(i)}\right)$	Dato de entrenamiento.
$\left(\overrightarrow{x^{(1)}}, y^{(1)}\right), \left(\overrightarrow{x^{(2)}}, y^{(2)}\right), \ldots, \left(\overrightarrow{x^{(m)}}, y^{(m)}\right)$	Conjunto de datos de entrenamiento.
$\overrightarrow{y}$	Vector de variables de respuesta.
$\boldsymbol{X} \in \mathbb{R}^{n\ x\ m}$	Matriz de vectores de entrada

EJEMPLO VENTA DE BIENES INMOBILIARIOS

MODELO MATEMÁTICO

LA NEURONA ARTIFICIAL



$\boldsymbol{x_i} = característica\ del\ vector\ de\ características\ (señal\ de\ entrada)$

$\boldsymbol{w_i} = peso\ aplicado\ a\ esa\ característica\ (peso\ sináptico)$

$\boldsymbol{f} = función\ de\ activación$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

PROPONIENDO LA HIPÓTESIS

Se **establece** la **función** de **activación** f de la **neurona**, como un **modelo probabilístico** h_θ (**hipótesis**) para estimar si la neurona se "**prende**" o se "**apaga**".

No linealidad y no decreciente

Modelar patrones complejos y **NO** incurrir en una **composición** de **transformaciones lineales factorizables**.

Axioma de probabilidad

$f = h_\theta \in [0, 1]$

Infinitamente diferenciable.

Dada una función f $\exists$ $f', f'', \ldots, f^{(k)}$ donde $k \in \mathbb{N}$ y $k \to \infty$

TEOREMA DEL APROXIMADOR UNIVERSAL

Composición de **funciones** para **mapear** de **cualquier espacio** a otro

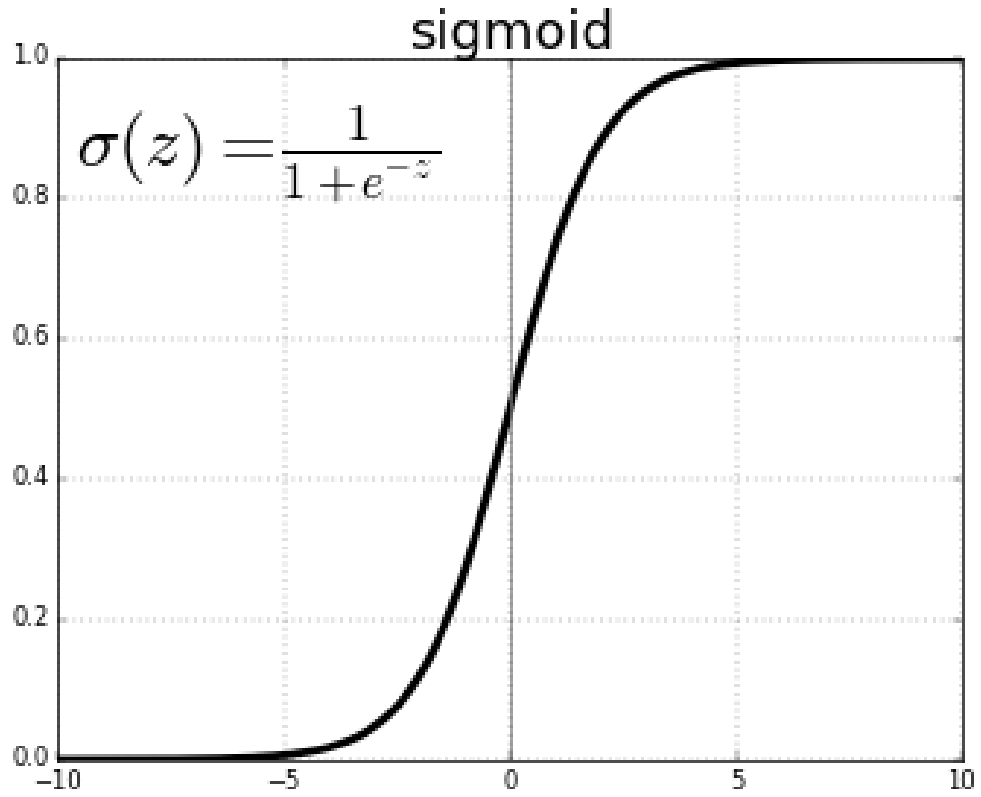
$$f: \mathbb{R}' \to \mathbb{R}: f(x)$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

Kurt Hornik, Maxwell Stinchcombe, & Halbert White. (1989). Multilayer Feedforward Networks are Universal Approximators. Neural Networks, 2, 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

MODELO PROBABILÍSTICO

FUNCIÓN SIGMOIDE (LOGÍSTICA)



Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

MODELANDO PROBABILIDADES

Por lo tanto, se **plantea** la **hipótesis** de **estimación** (**activación** de la **neurona**) como:

$$P(y = 1/\vec{x}; \vec{W}) = \hat{y} = \frac{1}{1+e^{-\vec{W}^T\vec{x}}}$$

Donde

$\hat{y}$**: hipótesis** h_θ para **estimar** la **probabilidad** de **activación** de la **neurona.**

$\vec{x}$**: vector** de **todas** las **señales pertenecientes** a **otras neuronas** (**datos** de **entrenamiento**)

$\vec{W}$: **vector** de **pesos sinápticos** de las **conexiones**.

MODELO PROBABILÍSTICO

MODELANDO PROBABILIDADES

Por el **2° axioma** de **probabilidad** se sabe que la **suma** de **probabilididades** de los **eventos** en el **universo** debe ser **igual** a **1.** Dado que solo **existen dos posibilidades** "prendido" y "apagado", se tiene:

$$P(y = 0/\vec{x}; \overrightarrow{W}) = 1 - \hat{y}$$

Modelando ambas probabilidades en una sola ecuación, se **encuentra** la **distribución** de **Bernoulli** para eventos **binomiales**:

Distribución de Bernoulli

$$P(y/\vec{x}; \overrightarrow{W}) = \hat{y}^{y}(1 - \hat{y})^{1-y}$$

SE INTERPRETA LA ECUACIÓN

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

FUNCIÓN DE COSTO

Se **modifica** la **probabilidad aplicando logaritmos**, lo que permite obtener un **error** de **estimación** con **respecto** a **un solo dato** de **entrenamiento**:

$$\boldsymbol{l(y,\hat{y}) = (y\,log(\hat{y}) + (1-y)log(1-\hat{y}))}$$

Se **añade** un **signo menos** por **conveniencia** (**minimización** en **lugar** de **maximización**):

$$\boldsymbol{J(y,\hat{y}) = -(y\,log(\hat{y}) + (1-y)log(1-\hat{y}))}$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

FUNCIÓN DE COSTO

Interpretando **la función de costo:**

$$J(y, \hat{y}) = -(y\,log(\hat{y}) + (1-y)log(1-\hat{y}))$$

Si $y = 0$	Si $y = 1$
$J(y=0, \hat{y}) = -log(1-\hat{y})$	$J(y=1, \hat{y}) = -log(\hat{y})$
$\lim_{\hat{y}\to 0} log(1-\hat{y}) \to 0$	$\lim_{\hat{y}\to 0} log(\hat{y}) \to \infty$
$\lim_{\hat{y}\to 1} log(1-\hat{y}) \to \infty$	$\lim_{\hat{y}\to 1} log(\hat{y}) \to 0$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

SUPUESTOS DEL MODELO

Para proceder, se **supone** que **todas** las **salidas** (**variables aleatorias**) que **comprenden** el **conjunto** de **datos** de **entrenamiento,** $\boldsymbol{y^{(1)}, y^{(2)}, \ldots, y^{(m)}}$ son **independientes** y están **idénticamente distribuidas** (**IID**)

Idénticamente distribuidas

Todas las **variables** aleatorias muestreadas **pertenecen** a la **misma distribución** de **probabilidad** (en este caso **Bernoulli**).

Independientes

La **ocurrencia** de una **muestra** $\boldsymbol{y_i}$ no **provee información** alguna **sobre** la **ocurrencia** de las **otras variables**.

$$\boldsymbol{P}\left(\bigcap_{\boldsymbol{i=1}}^{\boldsymbol{m}} \boldsymbol{y^{(i)}}\right) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{m}} \boldsymbol{P}(\boldsymbol{y^{(i)}})$$

MODELO PROBABILÍSTICO

FUNCIÓN DE VEROSIMILITUD

Se **define** la **función** de **verosimilitud** para **todas** las **observaciones** $y^{(i)}$dado un **vector** de **señales** de **características** $\vec{x}^{(i)}$. Los **parámetros** del **modelo** se **definen** como $\overrightarrow{W}$.

$$P(y/\vec{x};\overrightarrow{W}) = P(y^{(1)}, y^{(2)}, \dots, y^{(m)} \ / \ \vec{x}^{(1)}, \dots, \vec{x}^{(m)}; \overrightarrow{W})$$

$$P(y/\vec{x};\overrightarrow{W}) = P(\bigcap_{i=1}^{m} y^{(i)} / \vec{x}^{(i)}; \overrightarrow{W})$$

$$P(y/\vec{x};\overrightarrow{W}) = \prod_{i=1}^{m} P(y^{(i)} / \vec{x}^{(i)}; \overrightarrow{W})$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

FUNCIÓN DE COSTO

Se **adapta** la **función** de **verosimilitud** para que sea más fácil de **manipular** (**derivar**):

$$P(y/\vec{x};\vec{W}) = \prod_{i=1}^{m} P(y^{(i)}/\vec{x}^{(i)};\vec{W})$$

$$J(\vec{W}) = -\frac{1}{m}\sum_{i=1}^{m} l(\widehat{y^{(i)}}, y^{(i)})$$

PROMEDIO DE LOS ERRORES

Donde $l(\widehat{y^{(i)}}, y^{(i)})$ es la **pérdida logarítmica**:

$$l(y,\hat{y}) = (y\,log(\hat{y}) + (1-y)log(1-\hat{y}))$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

Queremos **encontrar** los **parámetros** (**pesos sinápticos**) $\vec{W}$ que **maximicen** la **verosimilitud** (**minimicen** el **error** de **estimación**) de realmente tener la **distribución** de **probabilidad** propuesta (en este caso **Bernoulli**) dado el **conjunto** de datos de **entrenamiento** $\vec{x}$ y los **parámetros** $\vec{W}$.

$$\underset{\vec{W}}{\boldsymbol{argmax}}\, \boldsymbol{P}(\boldsymbol{y}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}}) = \underset{\vec{W}}{\mathbf{argmax}} \prod_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{P}(\boldsymbol{y_i}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}})$$

$$\underset{\vec{W}}{\mathbf{argmin}} - \mathbf{log}(\boldsymbol{P}(\boldsymbol{y}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}})) = \underset{\vec{W}}{\boldsymbol{argmin}} - \frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{log}(\boldsymbol{P}(\boldsymbol{y}^{(\boldsymbol{i})}/\vec{\boldsymbol{x}}^{(\boldsymbol{i})};\vec{\boldsymbol{W}})))$$

$$\underset{\vec{W}}{\boldsymbol{argmin}} - \boldsymbol{log}(\boldsymbol{P}(\boldsymbol{y}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}})) = \underset{\vec{W}}{\boldsymbol{argmin}}\, \boldsymbol{J}(\vec{\boldsymbol{W}})$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

Queremos **encontrar** los **parámetros** (**pesos sinápticos**) $\vec{W}$ que **maximicen** la **verosimilitud** (**minimicen** el **error** de **estimación**) de realmente tener la **distribución** de **probabilidad** propuesta (en este caso **Bernoulli**) dado el **conjunto** de datos de **entrenamiento** $\vec{x}$ y los **parámetros** $\vec{W}$.

$$\underset{\vec{W}}{\boldsymbol{argmax}}\, \boldsymbol{P}(\boldsymbol{y}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}}) = \underset{\vec{W}}{\mathbf{argmax}} \prod_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{P}(\boldsymbol{y_i}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}})$$

$$\underset{\vec{W}}{\mathbf{argmin}} - \mathbf{log}(\boldsymbol{P}(\boldsymbol{y}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}})) = \underset{\vec{W}}{\boldsymbol{argmin}} - \frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{log}(\boldsymbol{P}(\boldsymbol{y}^{(\boldsymbol{i})}/\vec{\boldsymbol{x}}^{(\boldsymbol{i})};\vec{\boldsymbol{W}})))$$

$$\underset{\vec{W}}{\boldsymbol{argmin}} - \boldsymbol{log}(\boldsymbol{P}(\boldsymbol{y}/\vec{\boldsymbol{x}};\vec{\boldsymbol{W}})) = \underset{\vec{W}}{\boldsymbol{argmin}}\, \boldsymbol{J}(\vec{\boldsymbol{W}})$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

Antes de **proceder** en la **minimización**, vamos a utilizar una **propiedad** de la **derivada** de la función **sigmoide**:

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\nabla_z g(z) = \frac{e^{-z}}{(1 + e^{-z})^2}$$

$$\nabla_z g(z) = \left[\frac{1}{(1 + e^{-z})}\right] \frac{e^{-z}}{(1 + e^{-z})}$$

$$\nabla_z g(z) = g(z) \frac{e^{-z}}{(1 + e^{-z})}$$

$$\nabla_z g(z) = g(z) \left[\frac{e^{-z}}{(1 + e^{-z})} + (1 - 1)\right]$$

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

$$\nabla_z g(z) = g(z)\left[\frac{e^{-z}}{(1+e^{-z})} + (1-1)\right]$$

$$\nabla_z g(z) = g(z)\left[\frac{e^{-z} + (1+e^{-z}) - (1+e^{-z})}{(1+e^{-z})}\right]$$

$$\nabla_z g(z) = g(z)\left[\frac{(1+e^{-z}) - 1}{(1+e^{-z})}\right]$$

$$\nabla_z g(z) = g(z)\left[1 - \frac{1}{(1+e^{-z})}\right]$$

$$\boxed{\nabla_z g(z) = g(z)[1 - g(z)]}$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

Procedemos con la **minimización** de la **función de costo:**

$$\underset{\vec{W}}{\boldsymbol{argmin}}\, \boldsymbol{J}(\vec{\boldsymbol{W}}) = \underset{\vec{W}}{\boldsymbol{argmin}} - \frac{\boldsymbol{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \boldsymbol{l}(\widehat{\boldsymbol{y}^{(i)}}, \boldsymbol{y}^{(i)})$$

$$\underset{\vec{W}}{\boldsymbol{argmin}}\, \boldsymbol{J}(\vec{\boldsymbol{W}}) = \underset{\vec{W}}{\boldsymbol{argmin}} \frac{\boldsymbol{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} (\boldsymbol{y}^{(i)}\, \boldsymbol{log}\big(\widehat{\boldsymbol{y}^{(i)}}\big) + \big(\boldsymbol{1} - \boldsymbol{y}^{(i)}\big) \boldsymbol{log}(\boldsymbol{1} - \widehat{\boldsymbol{y}^{(i)}}))$$

$$\underset{\vec{W}}{\boldsymbol{argmin}}\, \boldsymbol{J}(\vec{\boldsymbol{W}}) = \underset{\vec{W}}{\boldsymbol{argmin}} \frac{\boldsymbol{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} (\boldsymbol{y}^{(i)}\, \boldsymbol{log}\left(\boldsymbol{g}\big(\boldsymbol{z}^{(i)}\big)\right) + \big(\boldsymbol{1} - \boldsymbol{y}^{(i)}\big) \boldsymbol{log}(\boldsymbol{1} - \boldsymbol{g}\big(\boldsymbol{z}^{(i)}\big)))$$

Donde $\boldsymbol{z}^{(i)} = \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}^{(i)}$

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

$$\underset{\vec{W}}{argmin}\, J(\vec{W}) = \underset{\vec{W}}{argmin} \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log\left(g\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log(1 - g\left(z^{(i)}\right)))$$

$$\underset{\vec{W}}{argmin}\, J(\vec{W}) = \frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \left(\frac{\nabla_w g\left(z^{(i)}\right)}{g(z^{(i)})} \right) - \left(1 - y^{(i)}\right) \left(\frac{\nabla_w g\left(z^{(i)}\right)}{1 - g(z^{(i)})} \right) \right)$$

$$\underset{\vec{W}}{argmin}\, J(\vec{W}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_w g\left(z^{(i)}\right) \left(y^{(i)} \left(\frac{1}{g(z^{(i)})} \right) - \left(1 - y^{(i)}\right) \left(\frac{1}{1 - g(z^{(i)})} \right) \right)$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

Sabemos que $\nabla_w g(z^{(i)}) = \nabla_w g(w^T x^{(i)})$**,** por lo que se aplica **regla de la cadena:**

$$\nabla_w g(w^T x^{(i)}) = \nabla_z g(z^{(i)}) \nabla_w z^{(i)}$$

Utilizando la **definición anterior** $\nabla_z g(z^{(i)}) = g(z^{(i)})[1 - g(z^{(i)})]$ **:**

$$\nabla_w g(w^T x^{(i)}) = g(z^{(i)})[1 - g(z^{(i)})] \nabla_w w^T x^{(i)}$$

Recordando que el gradiente de un producto punto $\nabla_w w^T x^{(i)} = x^{(i)}$

$$\nabla_w g(w^T x^{(i)}) = g(z^{(i)})[1 - g(z^{(i)})] x^{(i)}$$

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

Sustituyendo $\nabla_w g\left(z^{(i)}\right) =$:

$$\underset{\vec{W}}{argmin} J(\vec{W}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_w g\left(z^{(i)}\right)\left(y^{(i)}\left(\frac{1}{g(z^{(i)})}\right) - \left(1 - y^{(i)}\right)\left(\frac{1}{1 - g(z^{(i)})}\right)\right)$$

$$\underset{\vec{W}}{argmin} J(\vec{W}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(\frac{1}{g(z^{(i)})}\right) - \left(1 - y^{(i)}\right)\left(\frac{1}{1 - g(z^{(i)})}\right)\right) g\left(z^{(i)}\right)\left[1 - g\left(z^{(i)}\right)\right] x^{(i)}$$

$$\underset{\vec{W}}{argmin} J(\vec{W}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(1 - g\left(z^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\left(g\left(z^{(i)}\right)\right)\right) x^{(i)}$$

MODELO PROBABILÍSTICO

PRINCIPIO DE MÁXIMA VEROSIMILITUD

$$\underset{\vec{W}}{argmin}\, J(\vec{W}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(1 - g\left(z^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\left(g\left(z^{(i)}\right)\right)\right)x^{(i)}$$

$$\underset{\vec{W}}{argmin}\, J(\vec{W}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - y^{(i)}g\left(z^{(i)}\right) - g\left(z^{(i)}\right) + y^{(i)}g\left(z^{(i)}\right)\right)x^{(i)}$$

$$\underset{\vec{W}}{argmin}\, J(\vec{W}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - g\left(z^{(i)}\right)\right)x^{(i)}$$

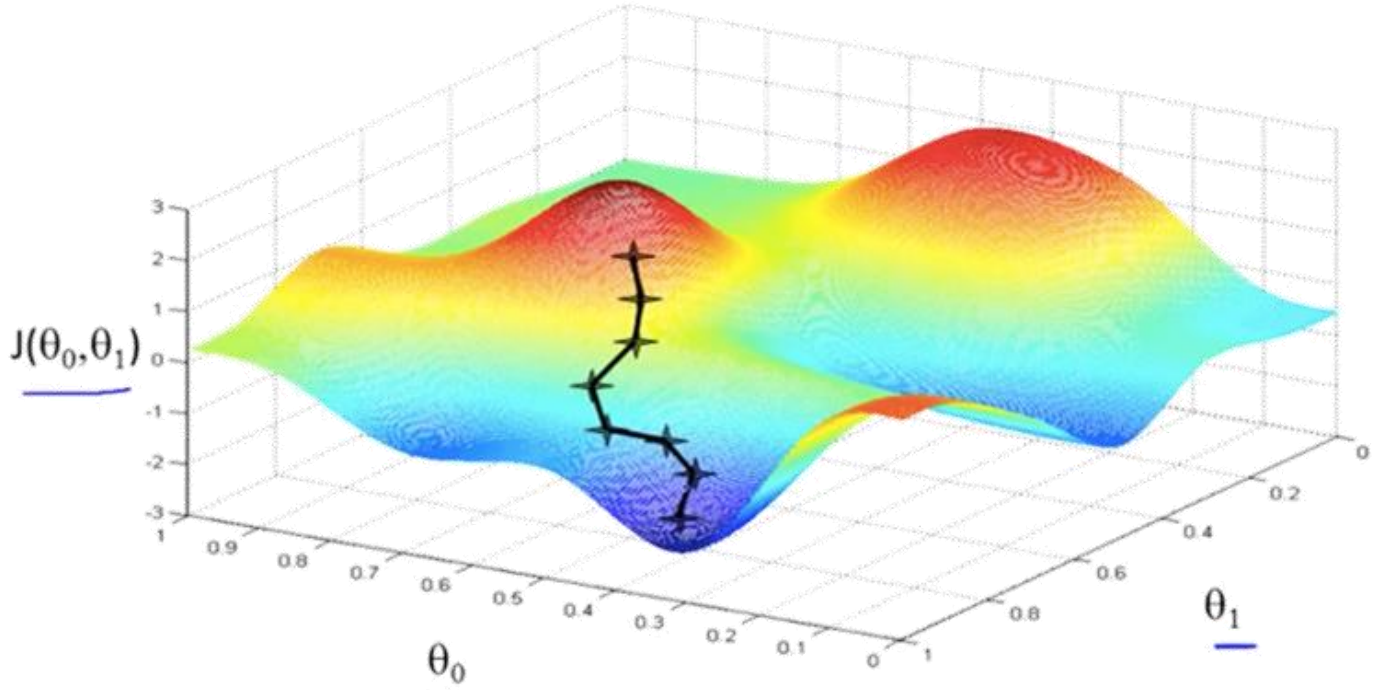
$$\underset{\vec{W}}{argmin}\, J(\vec{W}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \widehat{y^{(i)}}\right)x^{(i)}$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

PROCESO DE APRENDIZAJE

D E S C E N S O P O R G R A D I E N T E

Se plantea como **ejemplo** una **superficie** de **costo** basada solo en **dos pesos sinápticos**.



Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

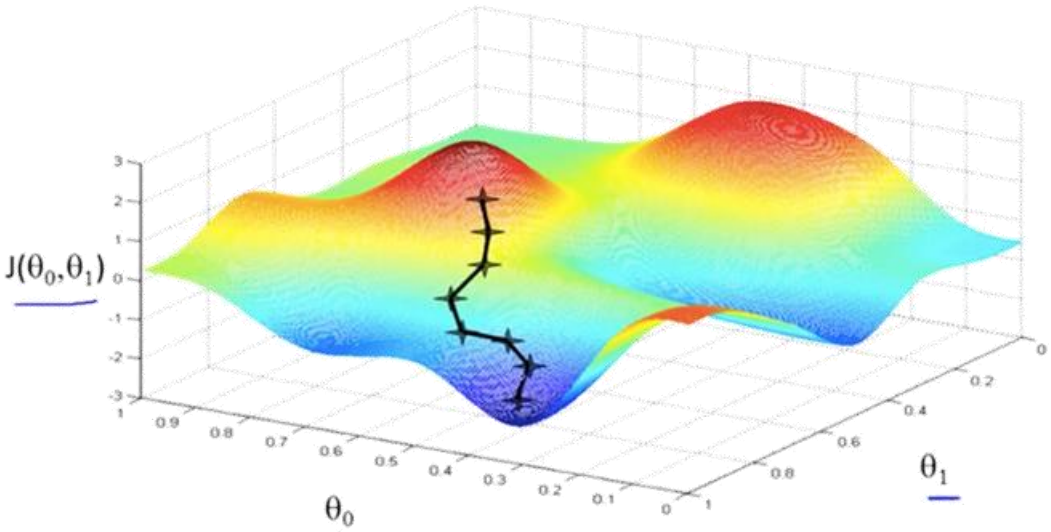
PROCESO DE APRENDIZAJE

DESCENSO POR GRADIENTE

Objetivo: encontrar los **valores** de **pesos sinápticos** que **minimicen** la **función** de **costo**.

Se utiliza el **gradiente** de la **función** $\underset{\overrightarrow{W}}{\boldsymbol{argmin}}\, \boldsymbol{J}(\overrightarrow{\boldsymbol{W}}) = \boldsymbol{\nabla_w J}(\overrightarrow{\boldsymbol{W}})$.

Para **actualizar** los **valores** de los **pesos**, donde cada **iteración** se encuentra **escalada** por **una constante** de **aprendizaje** $\boldsymbol{\alpha}$**.**



$$\boxed{\overrightarrow{\boldsymbol{w}}_{\boldsymbol{it+1}} \coloneqq \overrightarrow{\boldsymbol{w}}_{\boldsymbol{it}} - \boldsymbol{\alpha}\boldsymbol{\nabla_w J}(\overrightarrow{\boldsymbol{W}})}$$

Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

EJEMPLO PRÁCTICO NEURONA IDEAL

CASO DE ESTUDIO

ADMISIÓN A LA UNIVERSIDAD

El problema consiste en predecir si un estudiante de preparatoria será admitido en la universidad o no dadas las calificaciones de dos exámenes.

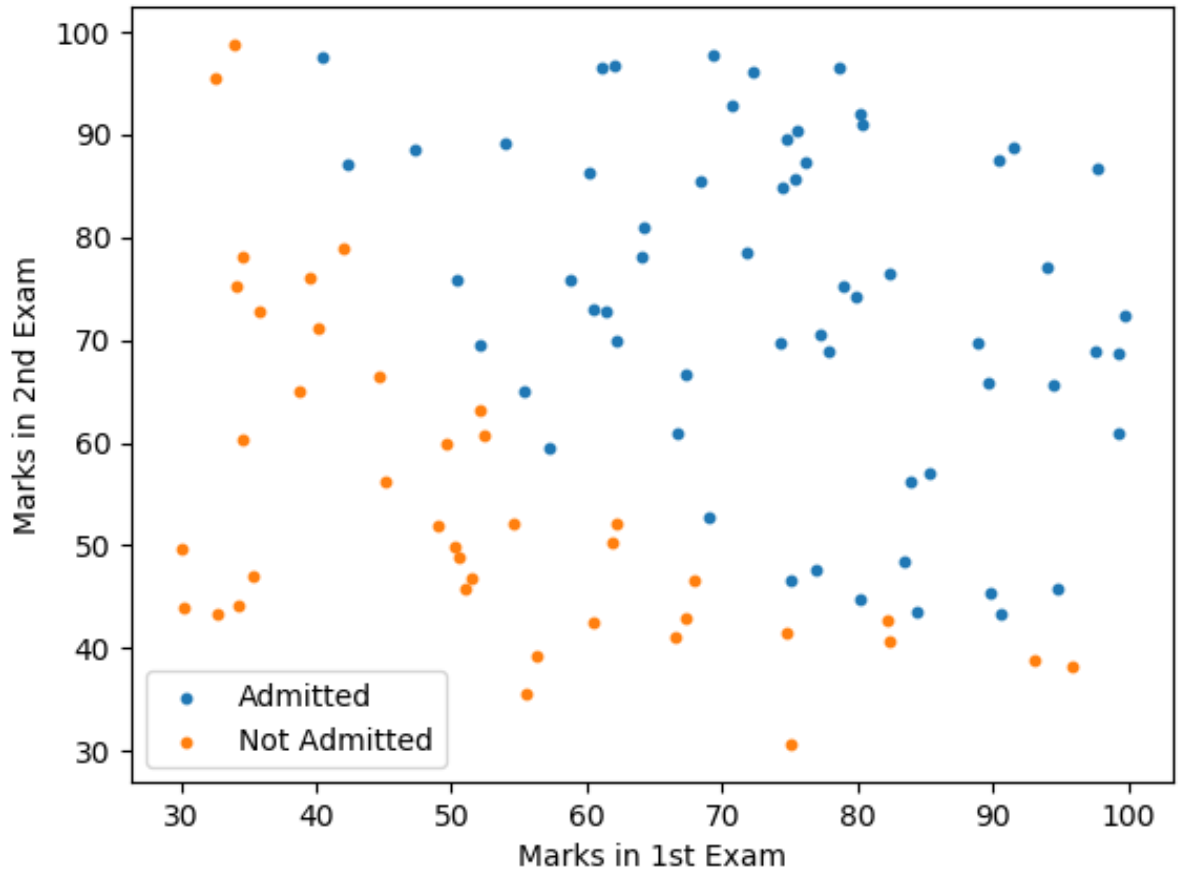
El conjunto de datos consta de 100 aplicantes. Donde $\vec{x}^{(i)} = [x_1, x_2]$ es el vector de entradas a la neurona y está compuesto por las 2 calificaciones x_1 y x_2 del estudiante i.

Además, se tienen las observaciones $y^{(i)} \in \{0, 1\}$ que indican sí el estudiante fue admitido "1" o rechazado "0".

Calificaciones $\vec{x}^{(i)}$	Admisión $y^{(i)}$
[38, 78]	0
…	…
[60, 86]	1

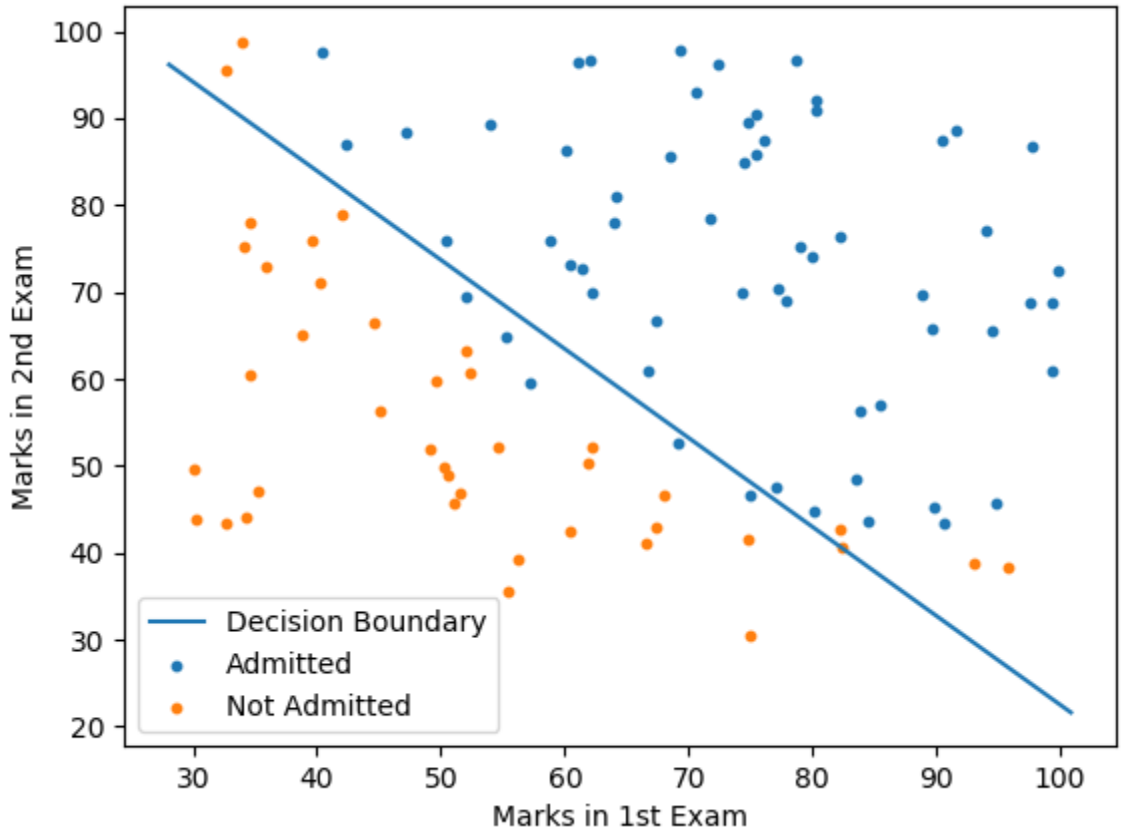
CASO DE ESTUDIO

GRAFICANDO LOS DATOS



CASO DE ESTUDIO

Frontera de Decisión



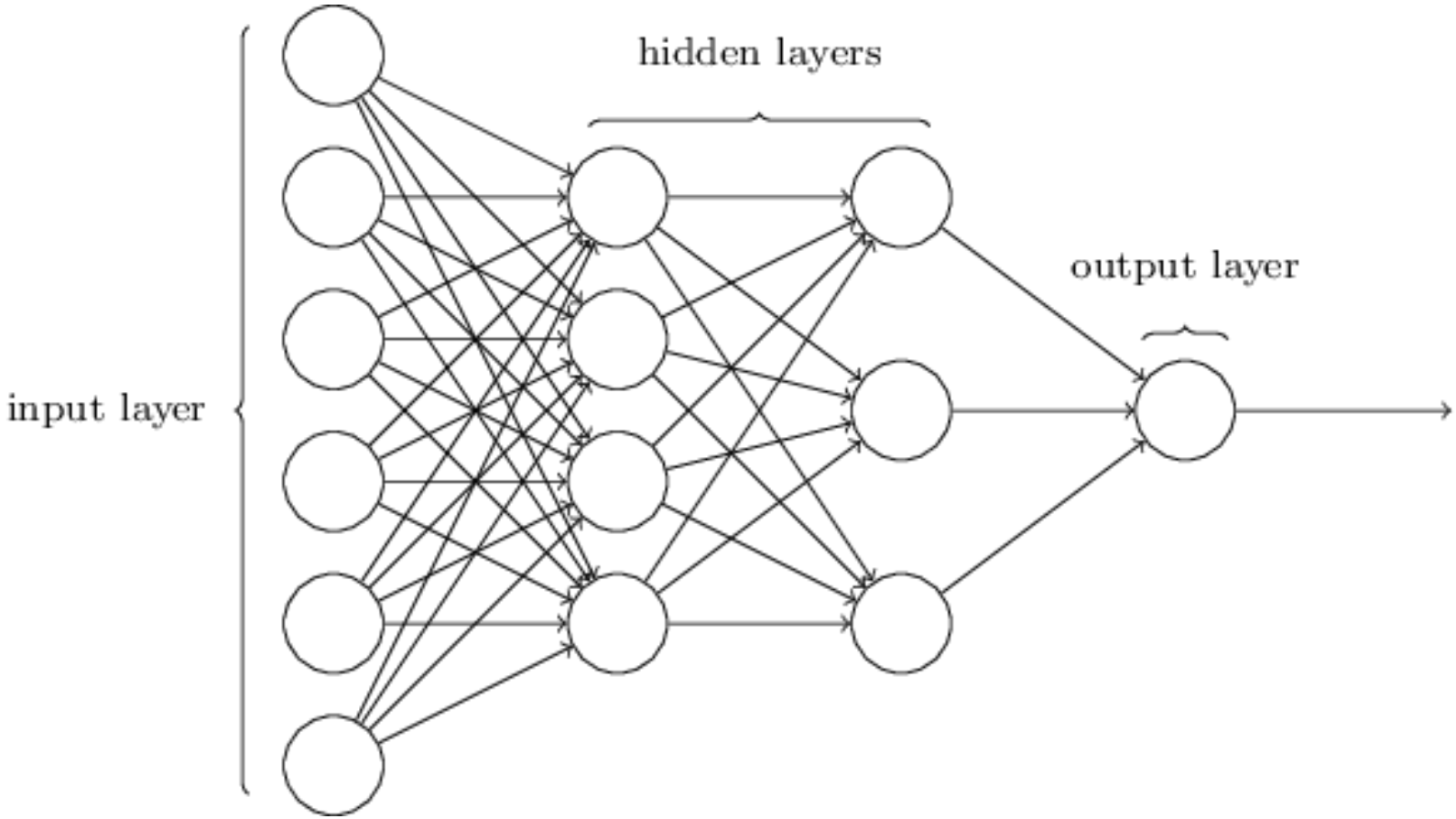
UMBRAL DE 0.5

PRECISIÓN = 89%

REDES NEURONALES

MODELO DE LAS REDES

A R Q U I T E C T U R A



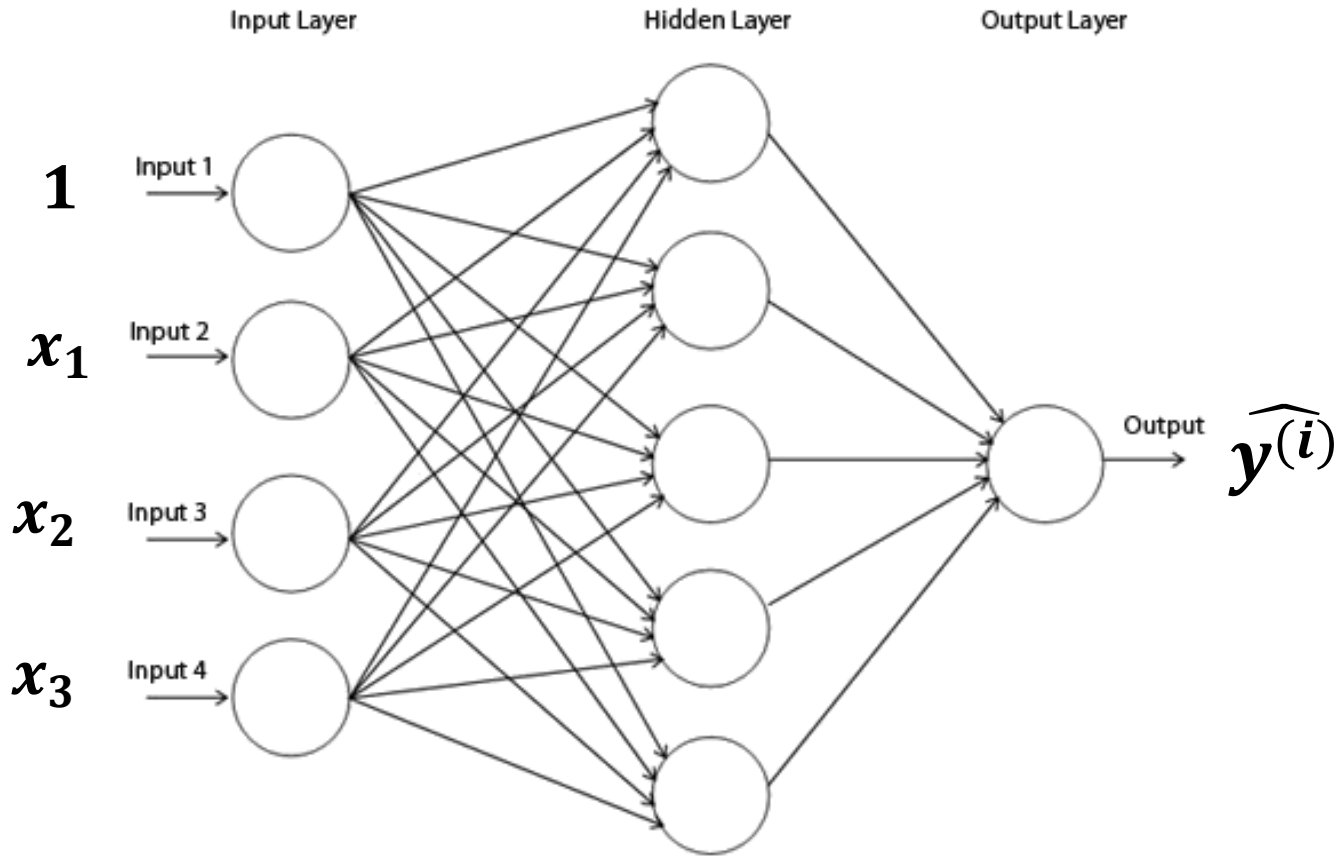
Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO DE LAS REDES

VECTORIZANDO UN SÓLO DATO

Data de entrada i:

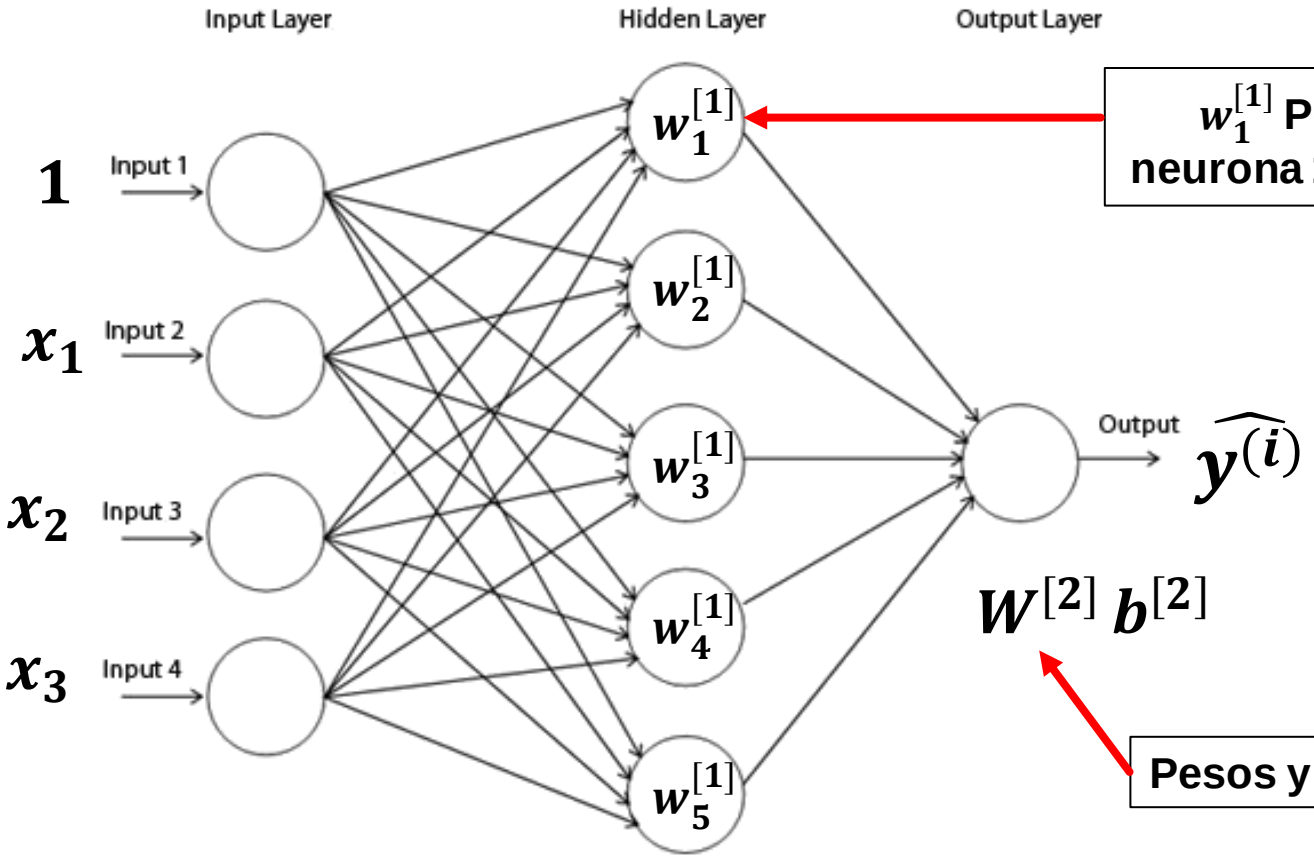
$$x^{(i)} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO DE LAS REDES

VECTORIZANDO UN SÓLO DATO



$w_1^{[1]}$ **Pesos de la neurona 1 de la capa 1**

Pesos de la capa l

$$W^{[l]} = \begin{bmatrix} -w_1^{[l]} - \\ -w_2^{[l]} - \\ \vdots \\ -w_{n_x}^{[l]} - \end{bmatrix}$$

$W^{[2]}\ b^{[2]}$

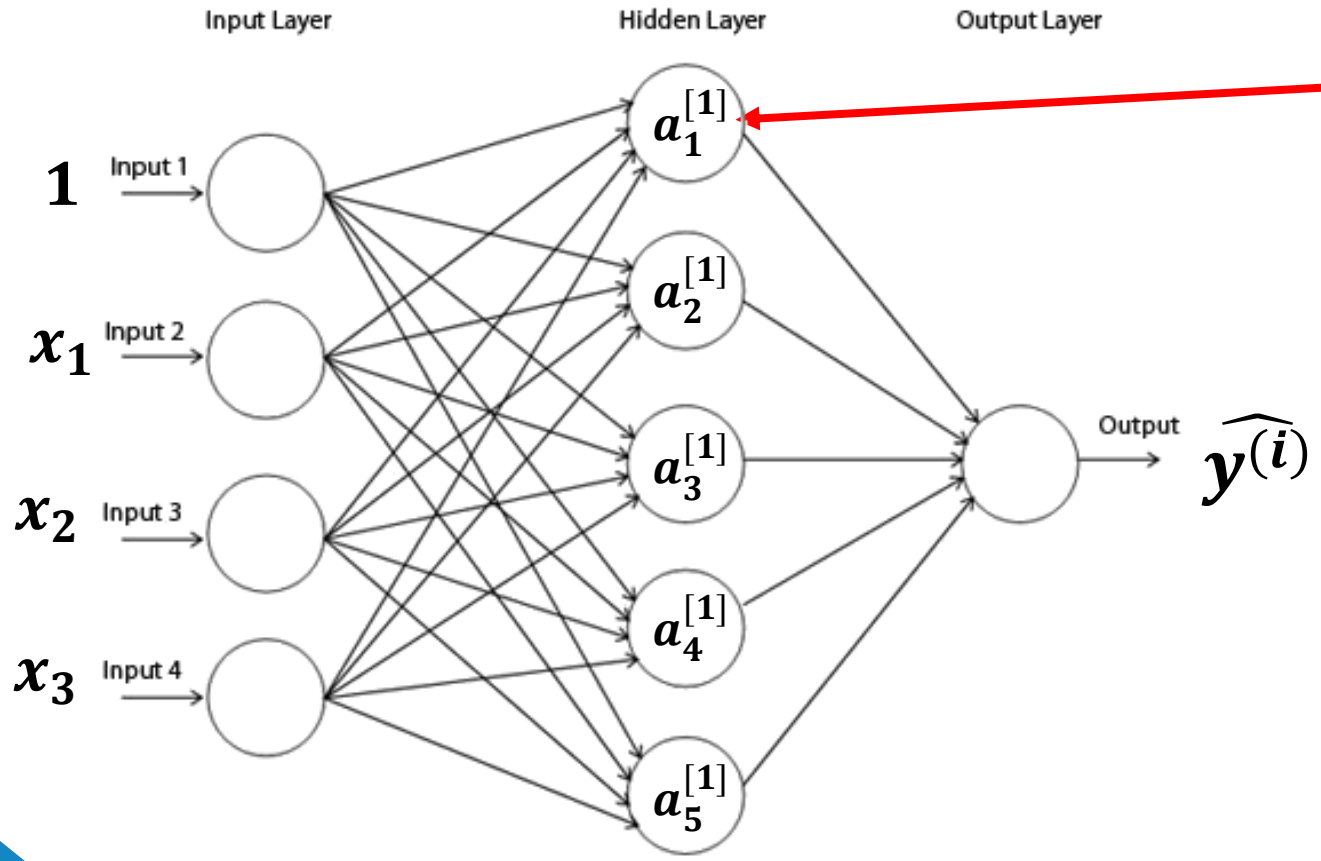
Pesos y sesgos de la capa 2

$W^{[1]}\ b^{[1]}$

Pesos y sesgos de la capa 1

MODELO DE LAS REDES

VECTORIZANDO UN SÓLO DATO



Cálculo una sola neurona

$$z_1^{[1]} = {w_1}^{[1]^T} x^{(i)} + {b_1}^{[1]}$$

$$a_1^{[1]} = g\left(z_1^{[1]}\right)$$

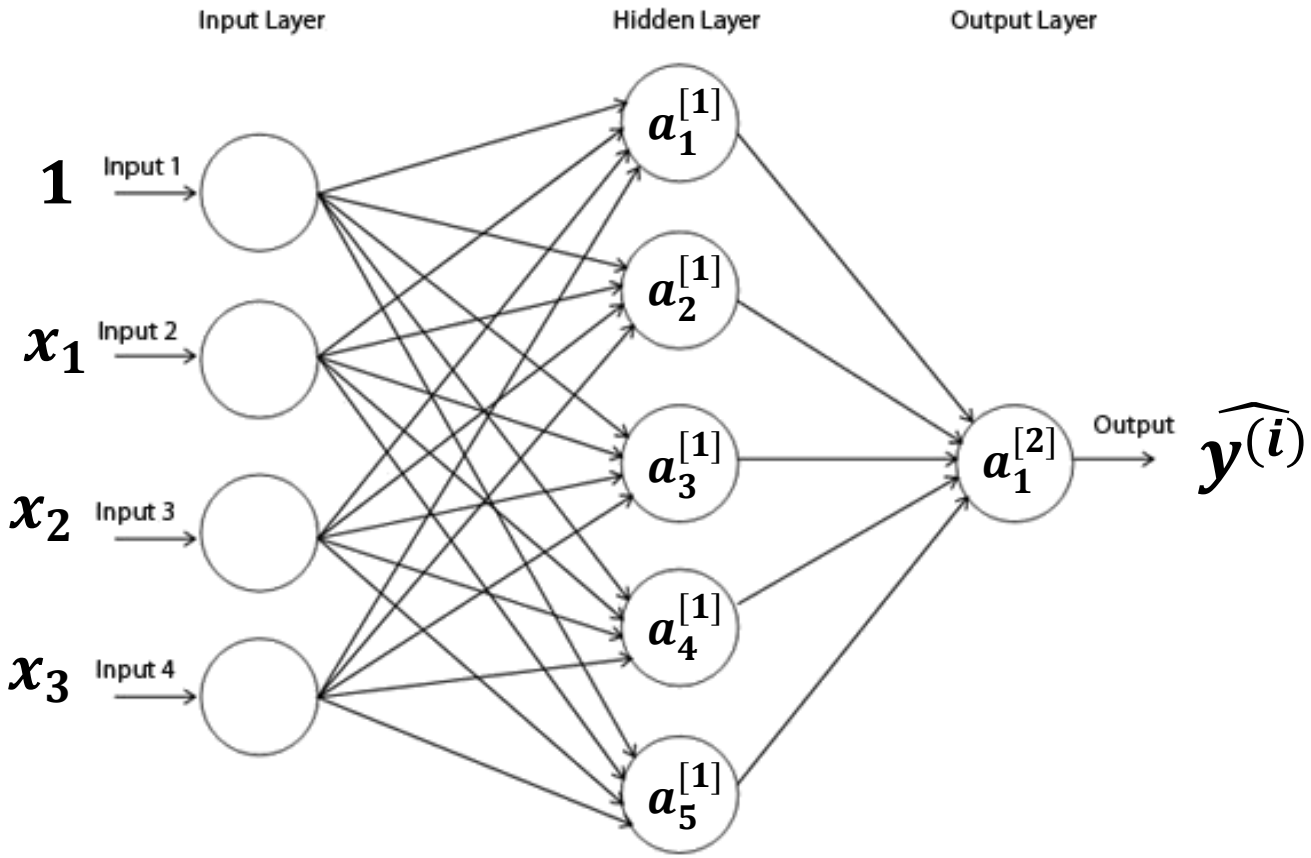
Cálculo de una capa l

$$z^{[l]} = {W^{[l]}}^T x^{(i)} + b^{[1]} = \begin{bmatrix} z_1^{[1]} \\ \vdots \\ z_{n_x}^{[1]} \end{bmatrix}$$

$$a^{[l]} = g\left(z^{[l]}\right) = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_{n_x}^{[1]} \end{bmatrix}$$

MODELO DE LAS REDES

PROPAGACIÓN ADELANTE UN SÓLO DATO



$a^{[1]}$: Vector de activaciones de la capa 1

Propagación hacia delante
(un solo dato)

Capa oculta:

$$z^{[1]} = W^{[1]^T} x^{(i)} + b^{[1]}$$

$$a^{[1]} = g\left(z^{[1]}\right)$$

Capa de salida

$$z^{[2]} = W^{[2]^T} a^{[1]} + b^{[2]}$$

$$a^{[2]} = g\left(z^{[2]}\right)$$

MODELO DE LAS REDES

PROPAGACIÓN ADELANTE UN SÓLO DATO

$$z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ \vdots \\ z_{n_x}^{[1]} \end{bmatrix} = \begin{bmatrix} {w_1}^{[1]T} x^{(i)} + {b_1}^{[1]} \\ \vdots \\ = {w_{n_x}}^{[1]T} x^{(i)} + {b_{n_x}}^{[1]} \end{bmatrix}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_{n_x}^{[1]} \end{bmatrix} = \begin{bmatrix} g\left(z_1^{[1]}\right) \\ \vdots \\ = g\left(z_{n_x}^{[1]}\right) \end{bmatrix}$$

Propagación hacia delante
(un solo dato)

Capa oculta:

$$z^{[1]} = {W^{[1]}}^T x^{(i)} + b^{[1]}$$

$$a^{[1]} = g\left(z^{[1]}\right)$$

Capa de salida

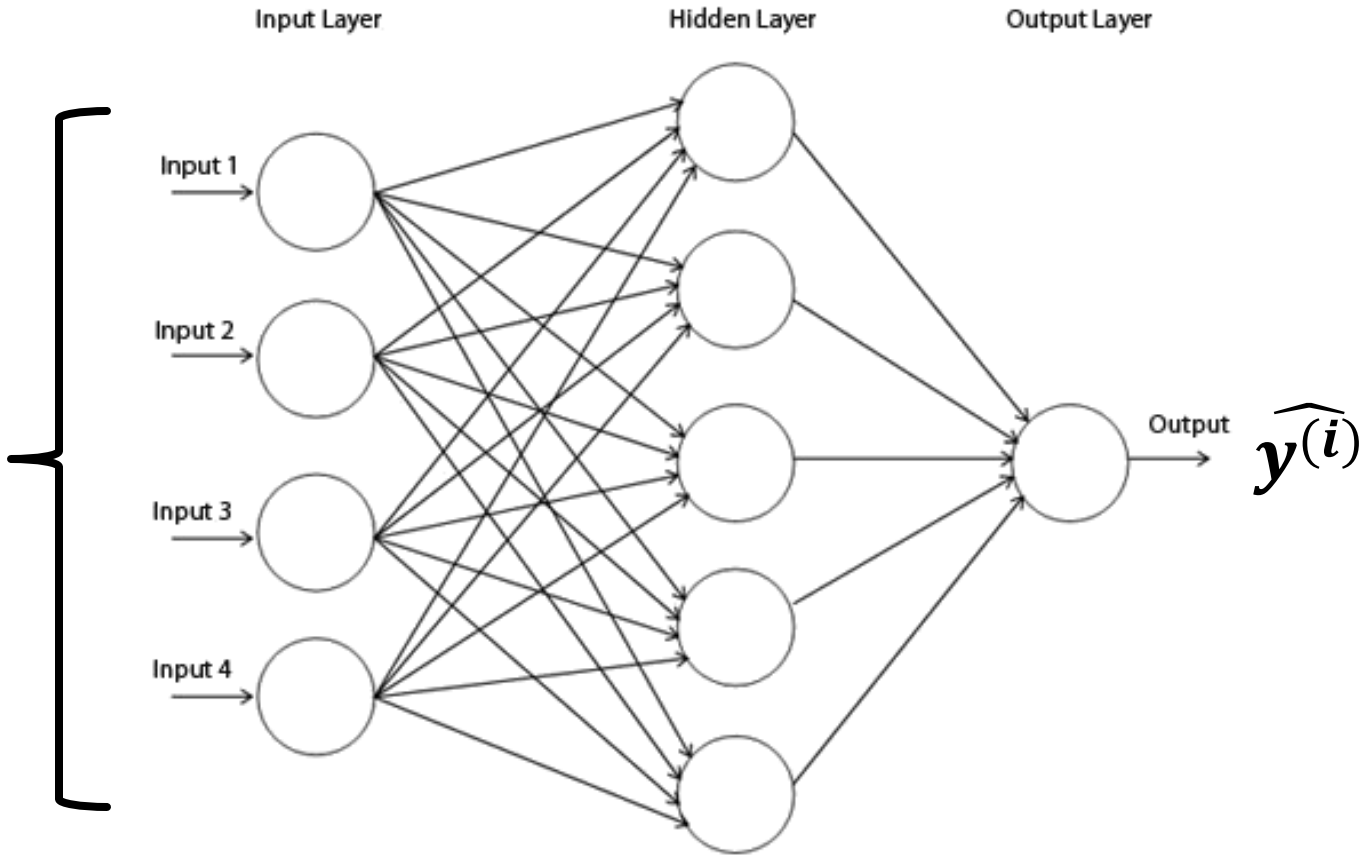
$$z^{[2]} = {W^{[2]}}^T a^{[1]} + b^{[2]}$$

$$a^{[2]} = g\left(z^{[2]}\right)$$

MODELO DE LAS REDES
VECTORIZANDO TODOS LOS DATOS

m datos de entrenamiento:

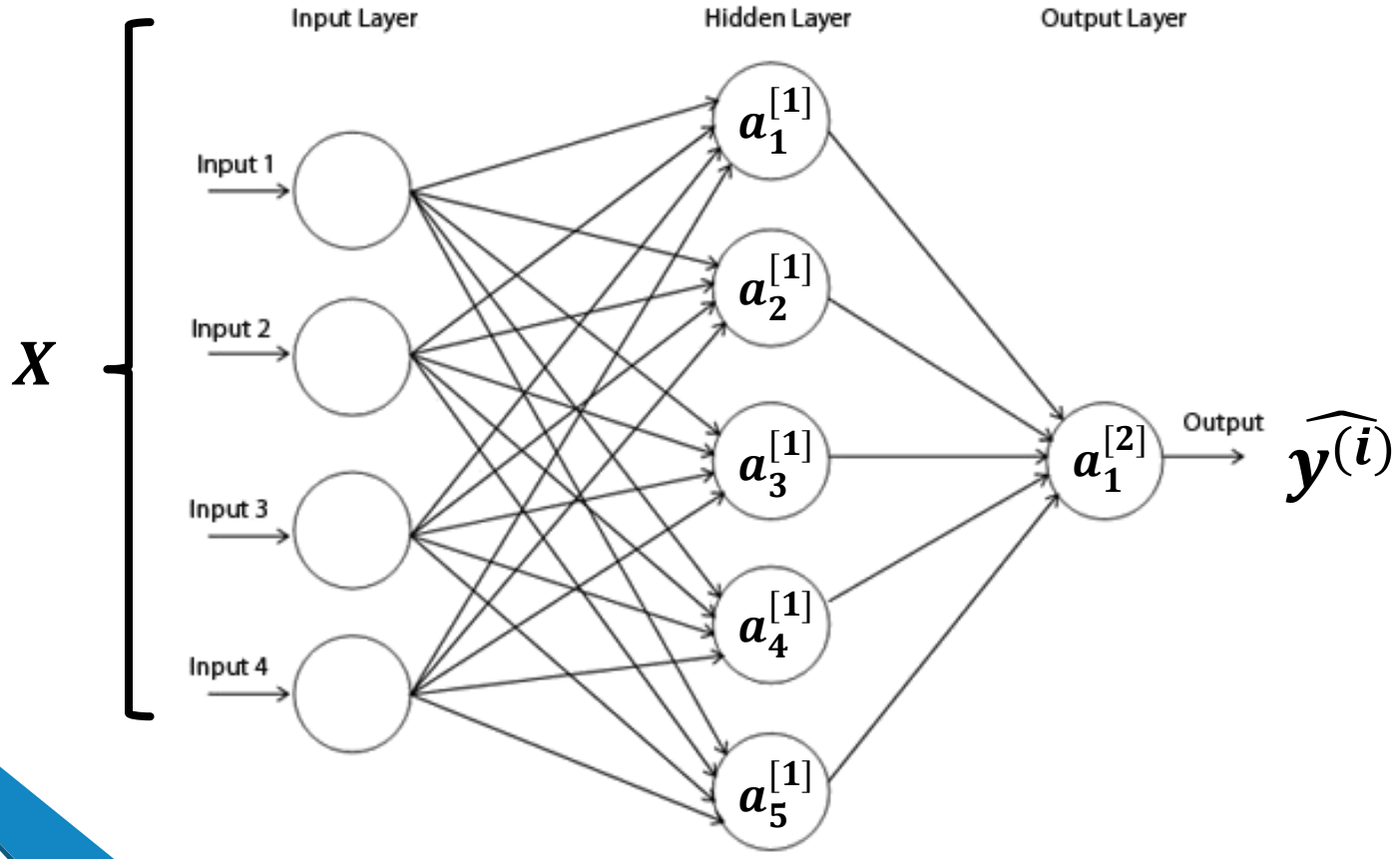
$$X = \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{bmatrix}$$



Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO DE LAS REDES

PROPAGACIÓN ADELANTE TODOS LOS DATOS



Propagación hacia delante
(todos los datos)

Capa oculta:

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = g(Z^{[1]})$$

Capa de salida

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = g(Z^{[2]})$$

MODELO DE LAS REDES

PROPAGACIÓN ADELANTE TODOS LOS DATOS

$$\boldsymbol{Z}^{[l]} = \begin{bmatrix} | & | & & | \\ z^{[l](1)} & z^{[l](2)} & \dots & z^{[l](m)} \\ | & | & & | \end{bmatrix}$$

$$\boldsymbol{A}^{[l]} = \begin{bmatrix} | & | & & | \\ a^{[l](1)} & a^{[l](2)} & \dots & a^{[l](m)} \\ | & | & & | \end{bmatrix}$$

Propagación hacia delante
(todos los datos)

Capa oculta:

$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$

$$\boldsymbol{A}^{[1]} = \boldsymbol{g}(\boldsymbol{Z}^{[1]})$$

Capa de salida

$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{A}^{[1]} + \boldsymbol{b}^{[2]}$$

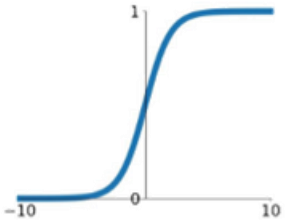
$$\boldsymbol{A}^{[2]} = \boldsymbol{g}(\boldsymbol{Z}^{[2]})$$

MODELO DE LAS REDES

OTRAS FUNCIONES DE ACTIVACIÓN

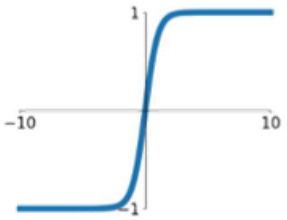
Sigmoid

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



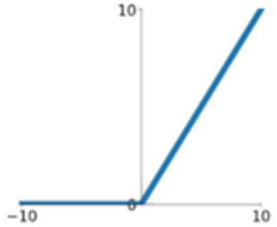
tanh

$$\tanh(x)$$



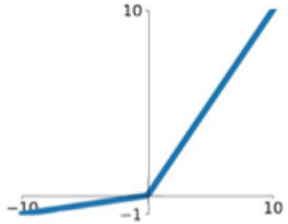
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

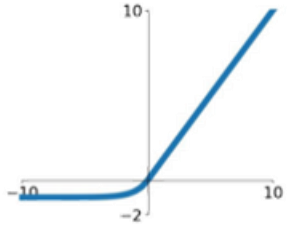


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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



Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO DE LAS REDES

FUNCIONES DE COSTO

Clasificación:

$$J(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\, log\left(g\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - g\left(z^{(i)}\right)\right)\right.$$

Regresión

$$J(y,\hat{y}) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \widehat{y^{(i)}}\right)^2$$

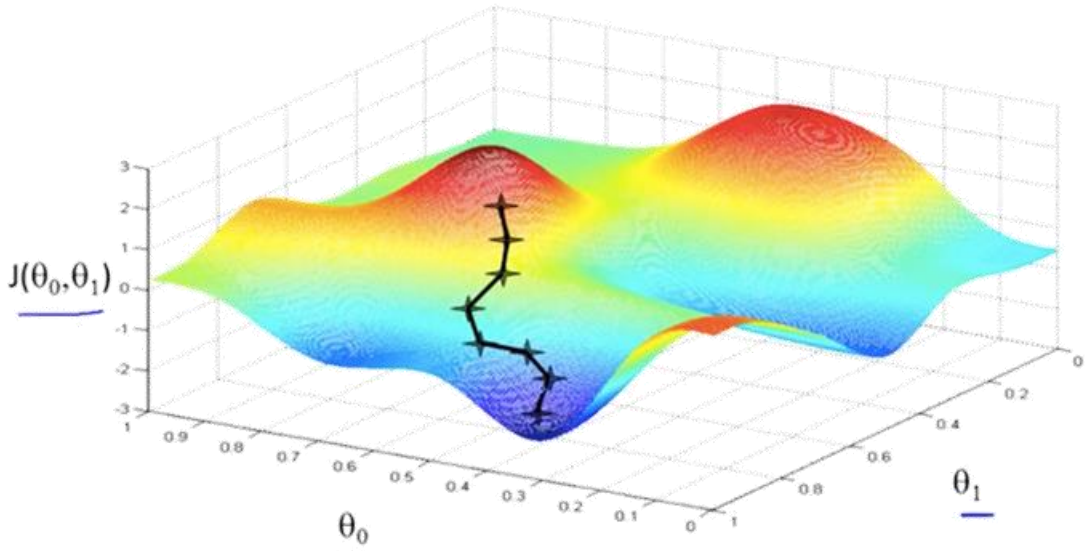
Aggarwal, C. C. (2018). Neural networks and deep learning : a textbook (1st Edition). Springer.

MODELO DE LAS REDES

RETROPROPAGACIÓN UN SOLO DATO

Se **calcula** el **descenso por gradiente** del **error respecto** a todos los **pesos** de la **capa inmediata anterior**.

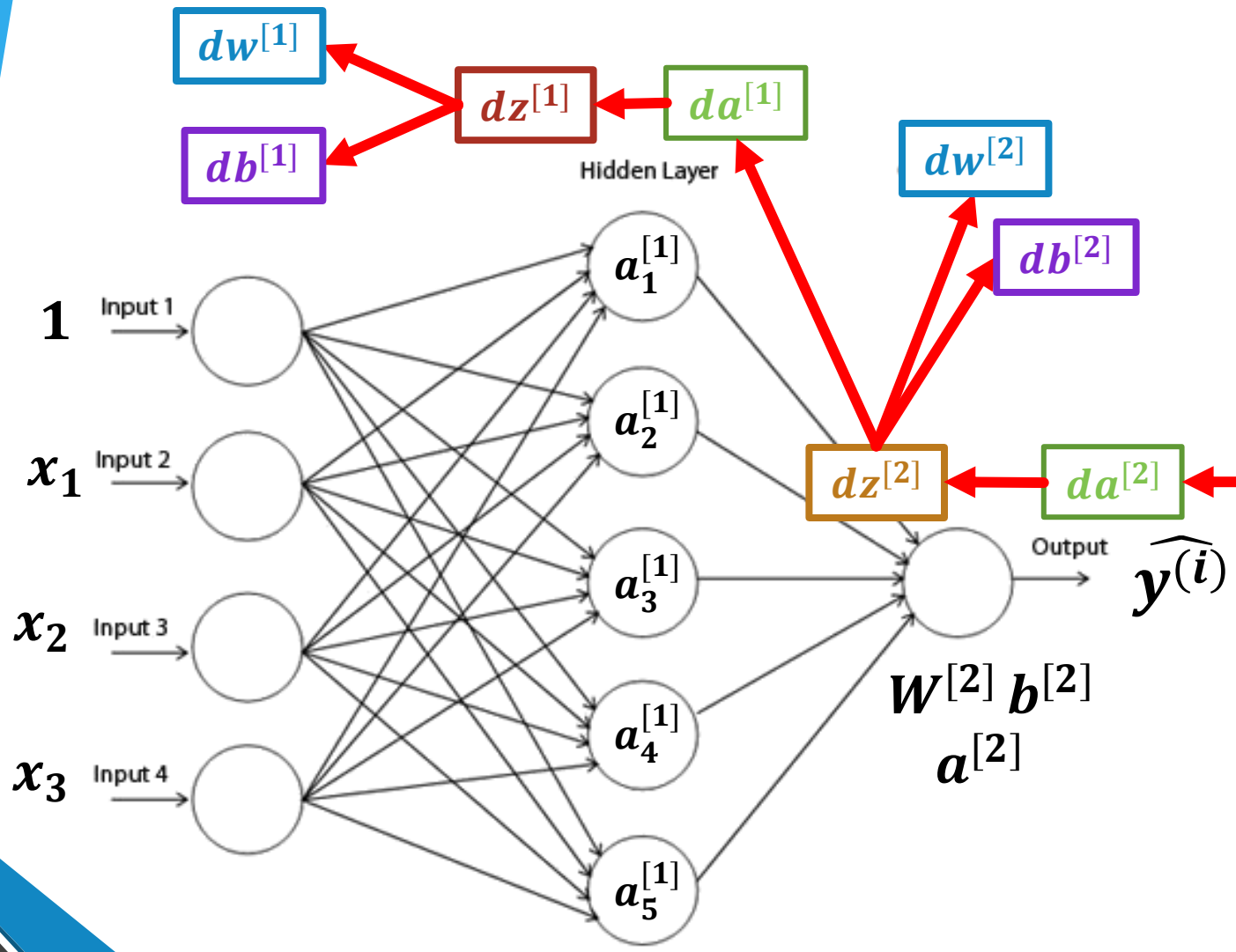
Se usa la **regla** de la **cadena** de **manera sucesiva** para **calcular** el **gradiente** respecto a **capas** más **profundas**.



SE TIENE UNA SUPERFICIE DE COSTO N-DIMENNSIONAL

MODELO DE LAS REDES

RETROPROPAGACIÓN UN SOLO DATO



SEGUNDO PASO

$$da^{[1]} = \nabla_{a^{[1]}} J(y,\hat{y}) \rightarrow dz^{[2]} \frac{\partial z^{[2]}}{\partial a^{[1]}}$$

$$dz^{[1]} = \nabla_{z^{[1]}} J(y,\hat{y}) \rightarrow da^{[1]} \frac{\partial a^{[1]}}{\partial z^{[1]}}$$

$$dw^{[1]} = \nabla_{w^{[2]}} J(y,\hat{y}) \rightarrow dz^{[1]} \frac{\partial z^{[1]}}{\partial w^{[1]}}$$

$$db^{[1]} = \nabla_{b^{[2]}} J(y,\hat{y}) \rightarrow dz^{[1]} \frac{\partial z^{[1]}}{\partial b^{[1]}}$$

PRIMER PASO

$$da^{[2]} = \nabla_{a^{[1]}} J(y,\hat{y}) \rightarrow \frac{\partial J}{\partial a^{[2]}}$$

$$dz^{[2]} = \nabla_{z^{[2]}} J(y,\hat{y}) \rightarrow da^{[2]} \frac{\partial a^{[2]}}{\partial z^{[2]}}$$

$$dw^{[2]} = \nabla_{w^{[2]}} J(y,\hat{y}) \rightarrow dz^{[2]} \frac{\partial z^{[2]}}{\partial w^{[2]}}$$

$$db^{[2]} = \nabla_{b^{[2]}} J(y,\hat{y}) \rightarrow dz^{[2]} \frac{\partial z^{[2]}}{\partial b^{[2]}}$$

MODELO DE LAS REDES

RETROPROPAGACIÓN TODOS LOS DATOS

En **resumen** se **tiene**:

1. **Propagación hacia adelante.**
2. **Calcular función de costo.**
3. **Retropropagación hacia atrás.**
4. **Actualizar los pesos de todas las capas con los gradientes.**
5. **Repetir el proceso hasta que el error de la función de costo tienda a cero.**

REDES NEURONALES CONVOLUCIONALES

REDES CONVOLUCIONALES

CONVOLUCIÓN

La **convolución** se **expresa** como una **operación** entre **dos funciones** f y g que **produce** una **tercera función** que define **como** la **forma** de una **función** es **cambiada** por la **otra**.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

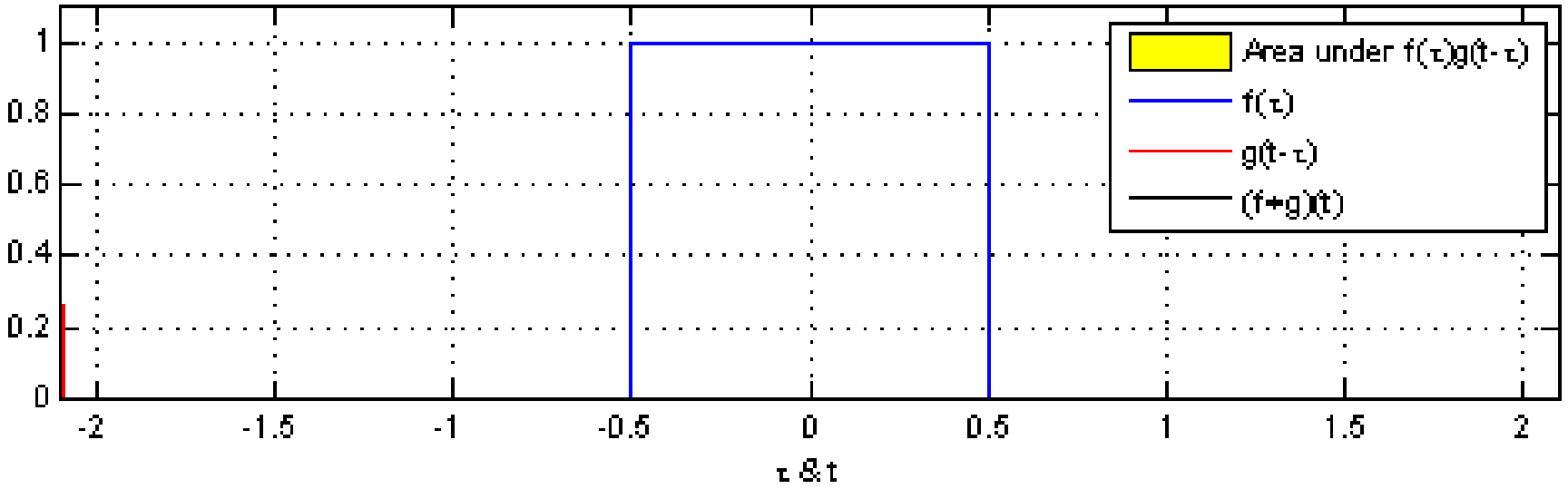
La **convolución** es **similar** a la **correlación cruzada** entre **dos funciones**, la cual **mide** la **similitud entre** dos **funciones** conforme **una** de **ellas** se **desplaza** sobre la **otra**.

$$(f \star g)(\tau) = \int_{-\infty}^{\infty} f(t) g(t + \tau) dt$$

Diferencia: las **funciones** están **reflejadas respecto** al **eje** y**.**

REDES CONVOLUCIONALES

C O N V O L U C I Ó N



REDES CONVOLUCIONALES

NOMENCLATURA

En la **terminología** del **área** de **aprendizaje máquina** f es la **entrada**, g es el **kernel** y el **resultado** se denomina como **mapa** de **características**.

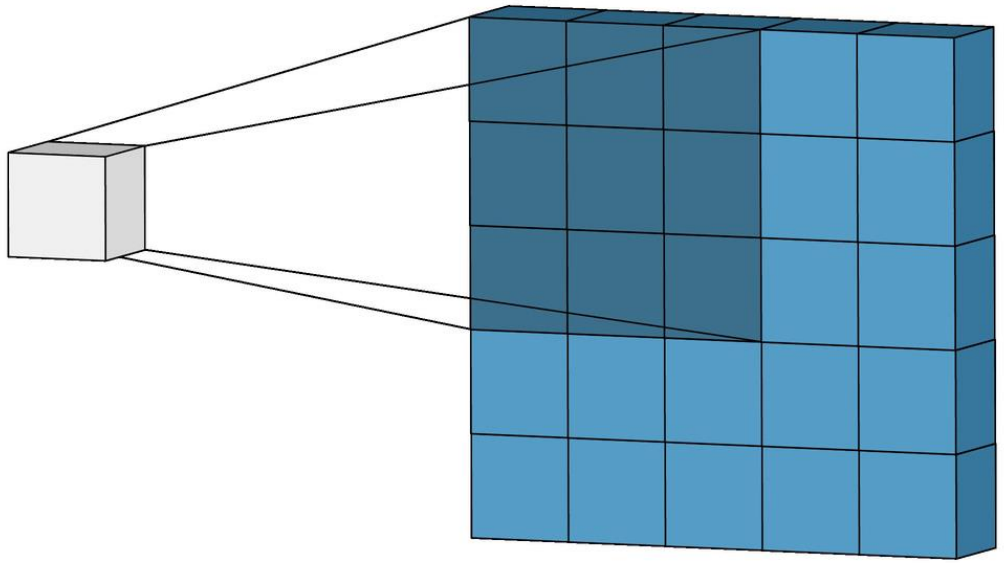
$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

En la **práctica**, se realiza la **convolución discreta** a lo largo de **varias dimensiones**:

$$(f * g)(t_1, \dots, t_N) = \sum_{\tau_1} \dots \sum_{\tau_2} \dots \sum_{\tau_N} f(\tau_1, \dots, \tau_N) g(t_1 - \tau_1, \dots, t_N - \tau_N)$$

REDES CONVOLUCIONALES

CONVOLUCIÓN DISCRETA



REDES CONVOLUCIONALES

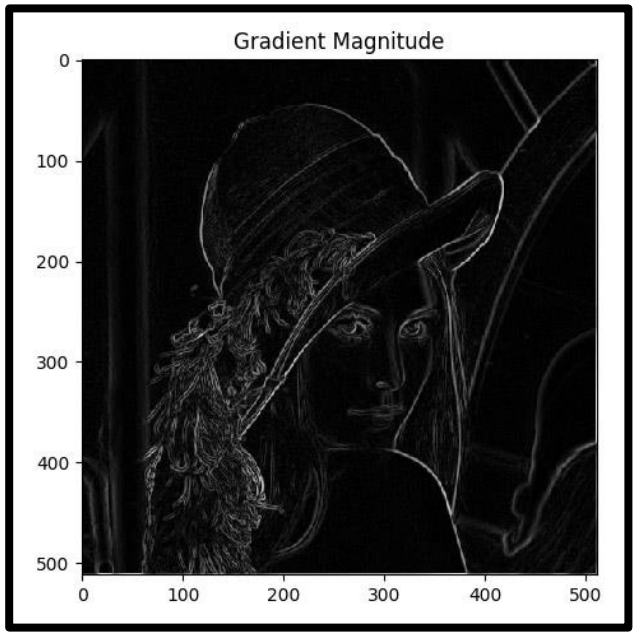
EJEMPLO CONVOLUCIÓN

Se puede observar un **ejemplo famoso** de **convolución**: **detección** de **bordes**.

-1	0	+1		+1	+2	+1
-2	0	+2		0	0	0
-1	0	+1		-1	-2	-1
Gx				Gy		

KERNELS

ENTRADA



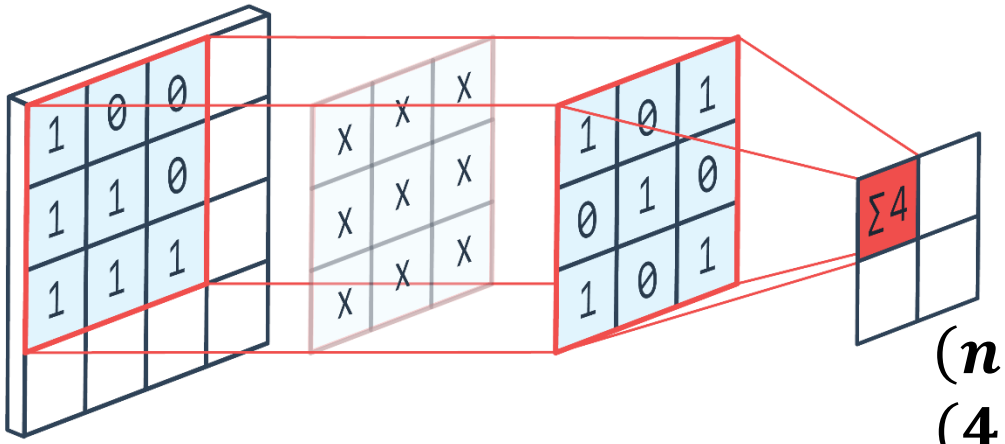
RESULTADO

$$\sqrt{G_X^2 + G_Y^2}$$

REDES CONVOLUCIONALES

CONVOLUCIÓN CON "PADDING"

Un **problema** que existe con la **convolución** es que los **volúmenes** resultantes **reducen** su **tamaño**. Además, los **píxeles** de las **esquinas** solo se **convolucionan una vez.**



$n \; x \; n$
$4 \; x \; 4$

$f \; x \; f$
$3 \; x \; 3$

$(n - f + 1) \; x \; (n - f + 1)$
$(4 - 3 + 1) \; x \; (4 - 3 + 1)$
$2 \; x \; 2$

REDES CONVOLUCIONALES

CONVOLUCIÓN CON "PADDING"

Este **problema** se **soluciona agregando "0"s** (relleno) **alrededor** de los **bordes** del **volumen** de **entrada** para **evitar** la **reducción**.

Input

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

$$(n + 2p)\ x(n + 2p)$$
$$(3 + 2)\ x(3 + 2)$$
$$5\ x\ 5$$

$*$

Kernel

0	1
2	3

$$f\ x\ f$$
$$2\ x\ 2$$

$=$

Output

0	3	8	4
9	19	25	10
21	37	43	16
6	7	8	0

$$(n + 2p - f + 1)\ x\ (n + 2p - f + 1)$$
$$(4 + 2 - 3 + 1)\ x\ (4 + 2 - 3 + 1)$$
$$4\ x\ 4$$

REDES CONVOLUCIONALES

CONVOLUCIÓN CON "PADDING"

Existen **dos tipos** de **convoluciones:**

1. **Convoluciones Válidas: no** hay "**padding**"

$$(n - f + 1)\, x\, (n - f + 1)$$

2. **Convoluciones Iguales:** se **establece** el "**padding**" de tal manera que el **volumen** de **entrada** sea **igual** al **volumen** de **salida**.

$$(n + 2p - f + 1) = n$$

$$p = \frac{f - 1}{2} \longrightarrow$$

Se quiere que el tamaño del kernel sea impar

REDES CONVOLUCIONALES

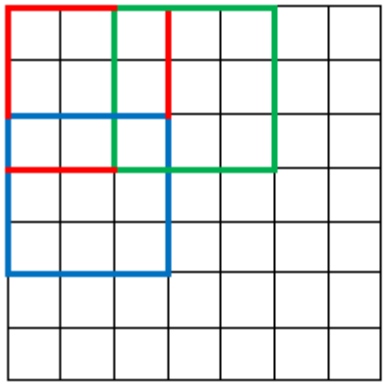
CONVOLUCIONES A SALTOS

En este caso hemos visto solo un **deslizamiento** del **kernel** con **salto** $s = 1$.

$f \; x \; f$

$3 \; x \; 3$

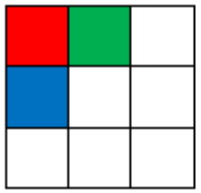
7 x 7 Input Volume



$n \; x \; n$

$7 \; x \; 7$

3 x 3 Output Volume



$$\left(\frac{n + 2p - f}{s} + 1\right) \; x \; \left(\frac{n + 2p - f}{s} + 1\right)$$

$$\left(\frac{7 + 0 - 3}{2} + 1\right) \; x \; \left(\frac{7 + 0 - 3}{2} + 1\right)$$

$3 \; x \; 3$

REDES CONVOLUCIONALES

APLICADO A REDES

En las **redes convolucionales,** se **usa** la **correlación discreta** en **varias dimensiones,** donde **no** se **voltea** el **kernel.**

Aun cuando **no** es **estrictamente válido**, se usará el **término convolución** y **correlación** de **manera intercambiable.**

$$(\boldsymbol{I} * \boldsymbol{K})(\boldsymbol{t_1}, \dots, \boldsymbol{t_N}) = \sum_{\boldsymbol{\tau_1}} \dots \sum_{\boldsymbol{\tau_2}} \dots \sum_{\boldsymbol{\tau_N}} \boldsymbol{I}(\boldsymbol{t_1} + \boldsymbol{\tau_1}, \dots, \boldsymbol{t_N} + \boldsymbol{\tau_N}) \boldsymbol{K}(\boldsymbol{\tau_1}, \dots, \boldsymbol{\tau_N})$$

Donde $\boldsymbol{I}$ es la **entrada** y $\boldsymbol{K}$ es el **kernel**.

REDES CONVOLUCIONALES

EJEMPLO DE UNA CAPA

En las **redes convolucionales** clásicas**,** las **capas** se **diferencian bastante** de las de una **red neuronal clásica.**

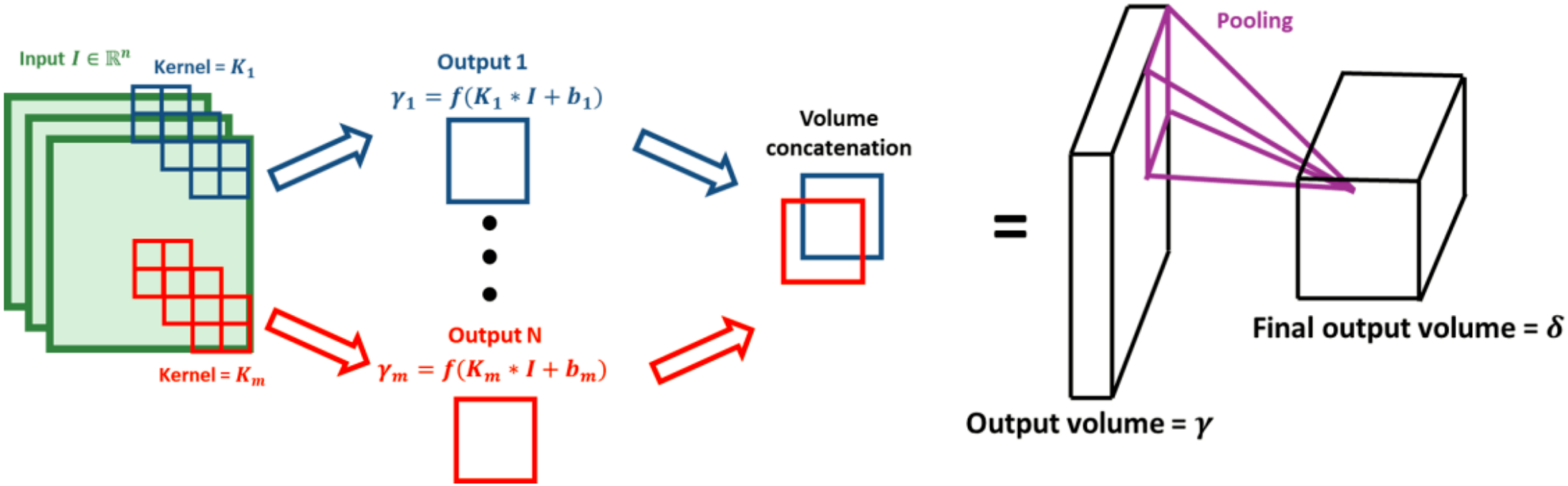
En este caso, se **convolucionan múltiples kernels** con la **entrada**. El **resultado** son **diferentes volúmenes reducidos** que se **concatenan** para generar un **único volumen**.

Se tienen **3 cálculos** en una **sola capa** de una **red convolucional**:

1. **Convolución.**
2. **Detección.**
3. **Agrupamiento.**

REDES CONVOLUCIONALES

EJEMPLO DE UNA CAPA

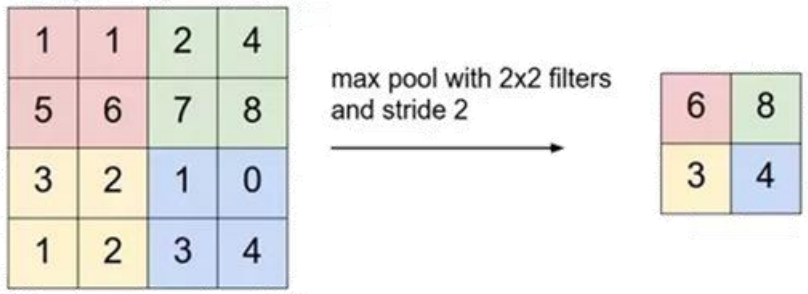


REDES CONVOLUCIONALES

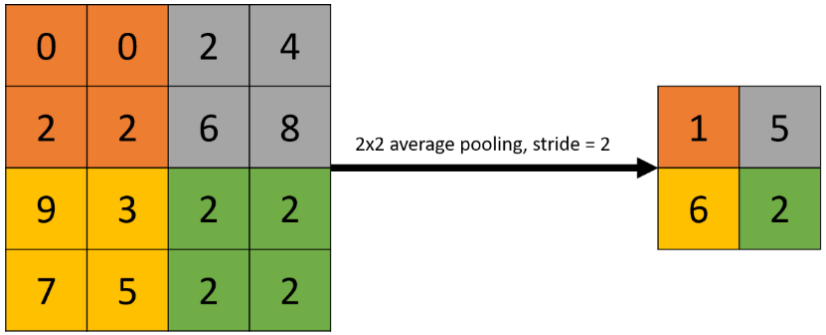
TIPOS DE AGRUPACIONES

El **objetivo** de la **agrupación** es **transformar** la **representación conjunta** de **características** en una **nueva representación**, que **conserve información importante** y **descarte detalles irrelevantes**.

Agrupación Máxima

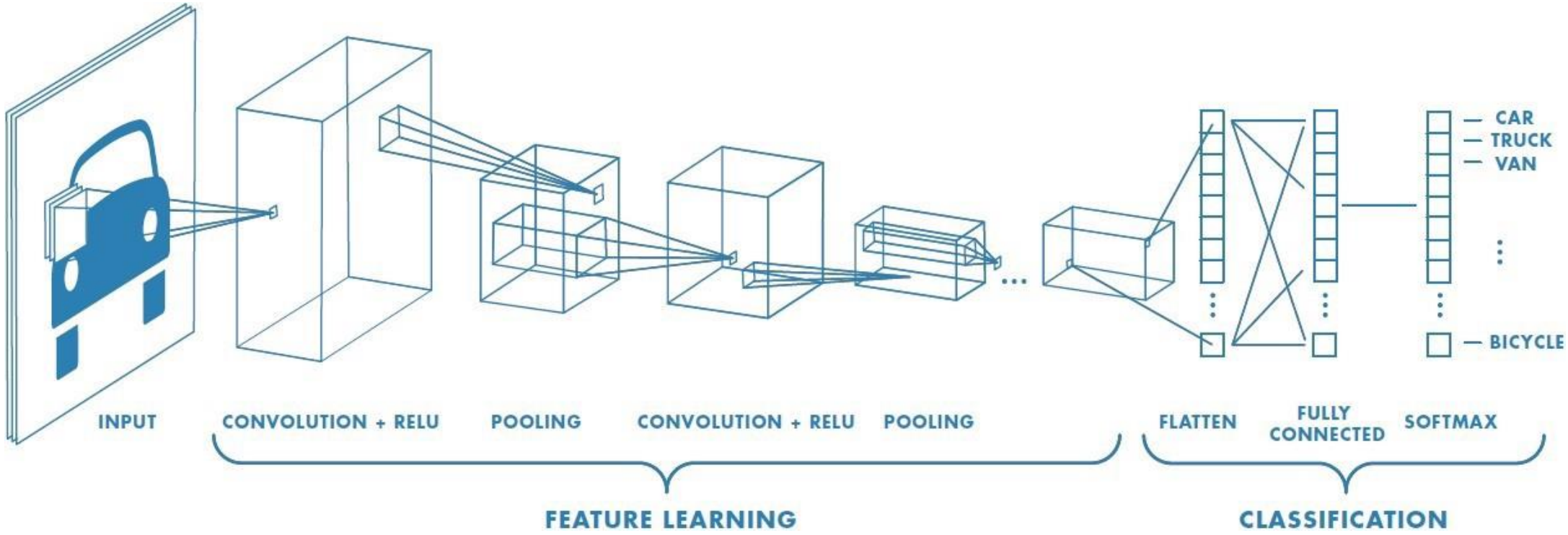


Agrupación Promedio



REDES CONVOLUCIONALES

EJEMPLO DE UNA RED CONVOLUCIONAL



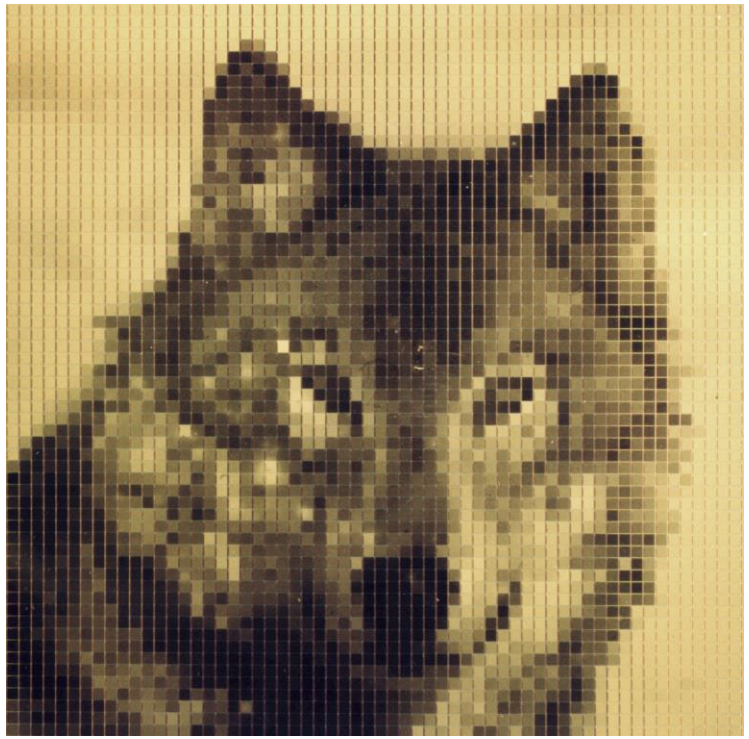
REDES CONVOLUCIONALES

MOTIVACIÓN

¿CUÁL ES EL PROBLEMA CON LAS REDES NEURONALES CLÁSICAS?

REDES CONVOLUCIONALES

MOTIVACIÓN



64 x 64 = 4,096 píxeles

1024 x 1024x3 = 3,145,728 píxeles

REDES CONVOLUCIONALES

M O T I V A C I Ó N

MUCHOS PARÁMETROS

=

SOBRE ENTRENAMIENTO + MUCHOS RECURSOS

REDES CONVOLUCIONALES

MOTIVACIÓN

La **ventaja** de **usar** una **arquitectura convolucional respecto** a una **arquitectura clásica** recae en **tres cuestiones** fundamentales:

Interacciones escasas.

Redundancia de parámetros.

Traslación equivalente.

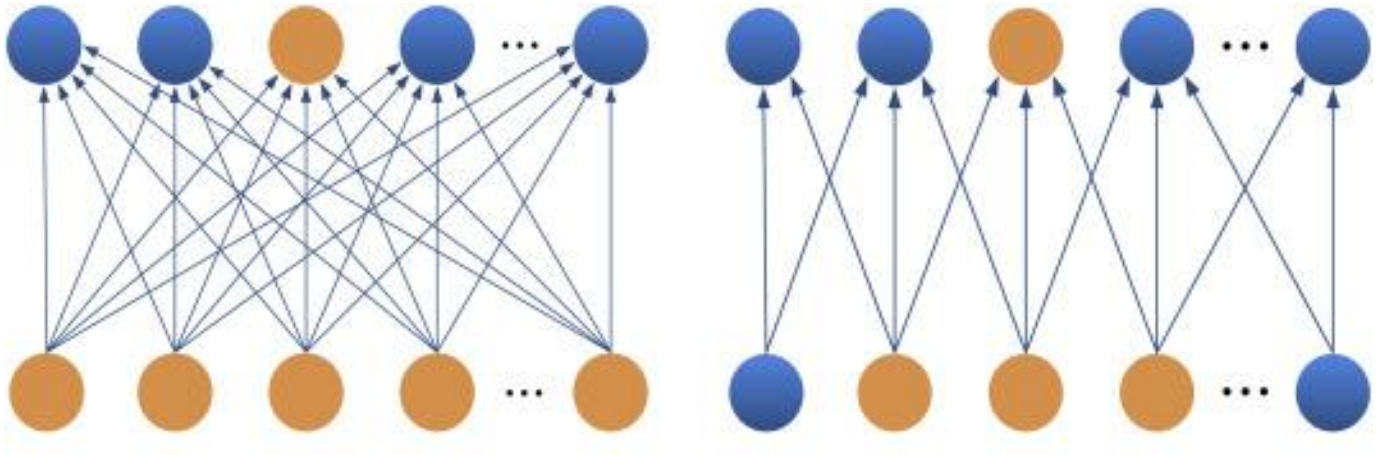
REDES CONVOLUCIONALES

M O T I V A C I Ó N

Interacciones escasas:

Al usar un **kernel** $\boldsymbol{K}$ que tiene una **dimensión mucho menor** respecto a la **entrada** $\boldsymbol{I}$**,** se tiene un **menor número** de **parámetros asociados** a cada **salida**.

En el caso de las **redes clásicas**, el **número** de **parámetros incrementa** considerablemente porque **todas** las **unidades** están **conectadas** entre sí.



(a) A full interaction structure

(b) A sparse interaction structure

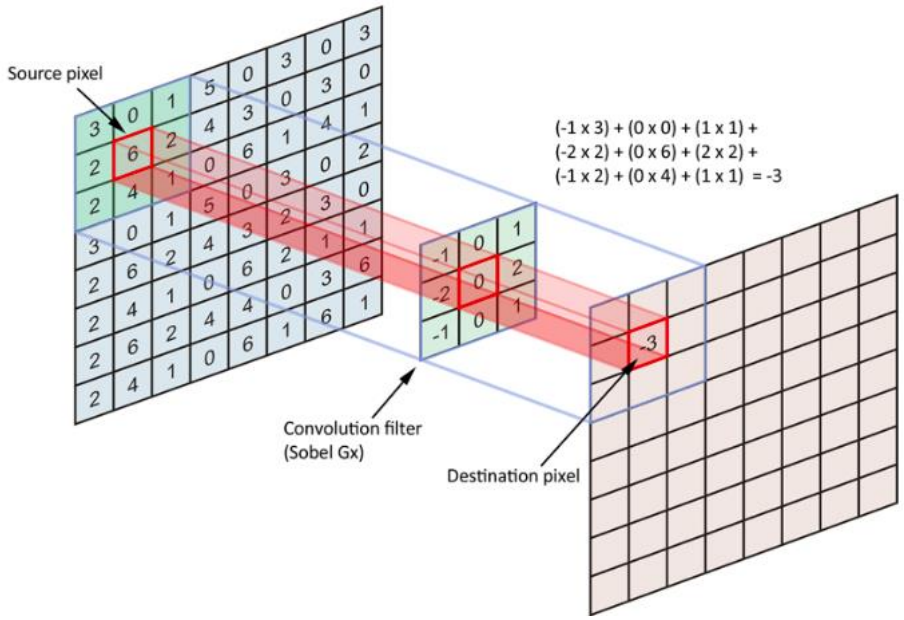
REDES CONVOLUCIONALES

M O T I V A C I Ó N

Redundancia de parámetros:

Un solo parámetro es **usado más** de **una vez** en la **red convolucional**.

Al deslizar el **kernel** a lo **largo** de la **entrada**, se **están aplicando** los **mismos pesos** en **diferentes porciones** de la **entrada**.



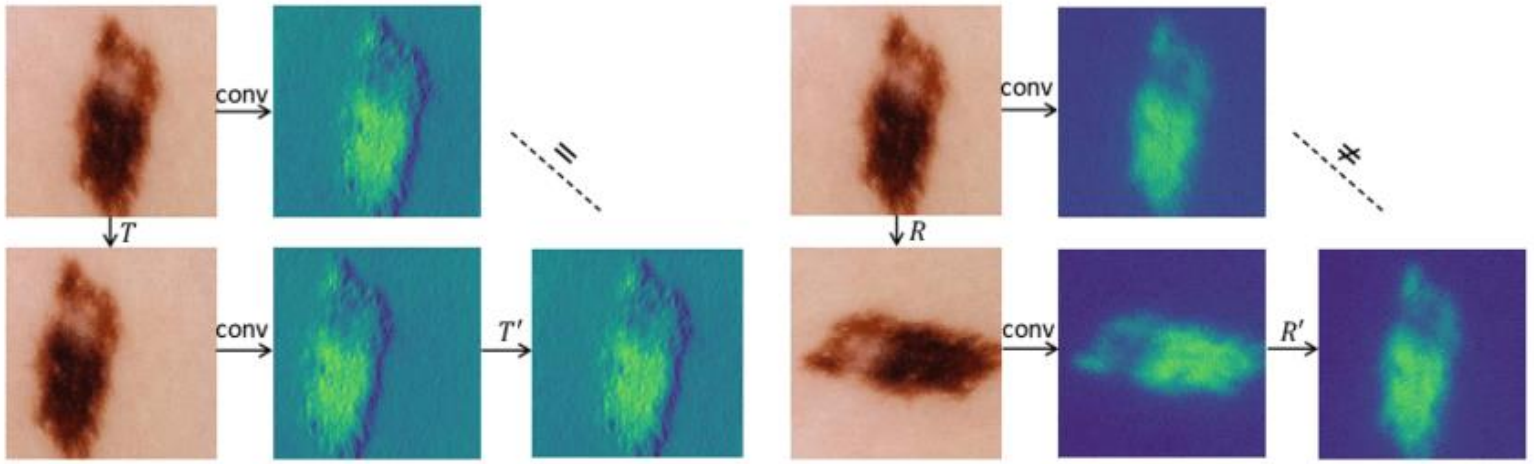
REDES CONVOLUCIONALES

MOTIVACIÓN

Traslación equivalente:

Aún cuando la **entrada** se **traslade,** las **representaciones construidas** por la red convolucional **permanecerán.** Esto se **asocia** a la **naturaleza inherente** del **operador** de **convolución.**

Aún cuando los **píxeles** se **trasladen**, la **red seguirá** siendo **capaz** de **detectar** el **patrón**.



(a) Conv is translation equivariant

(b) Conv is not rotation equivariant

REDES CONVOLUCIONALES

OPTIMIZACIÓN

El **aprendizaje** de la **red convolucional** se realiza **usando** la **misma** técnica de **retropropagación** que **usan** las **redes neuronales clásicas**.

Por lo tanto, las **redes convolucionales** se pueden **usar** para **regresión** y **clasificación**.