Cursos Extraordinarios

Verano 2024

“Inteligencia Artificial y Grandes Modelos de Lenguaje: Funcionamiento, Componentes Clave y Aplicaciones”

Zaragoza, del 3 al 5 de julio

Introducción al Aprendizaje Automático y las Redes Neuronales

Deep Learning

- **Rama de Machine Learning:**
    - Aproximación de propósito general que permite aprender a partir de datos.
    - Basada en el uso de Redes Neuronales Artificiales
- **Asociado actualmente al concepto de Inteligencia Artificial**

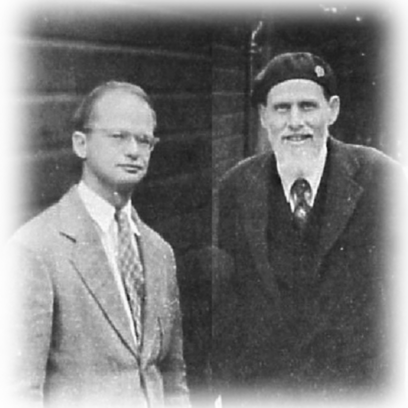
Capacidad de las máquinas para realizar tareas que normalmente requieren destrezas humanas.
Esto incluye:
Aprendizaje, percepción, razonamiento, reconocimiento de voz, comprensión del lenguaje natural, toma de decisiones, ...

Redes Neuronales Artificiales

- **Antecedentes:**

La teoría:

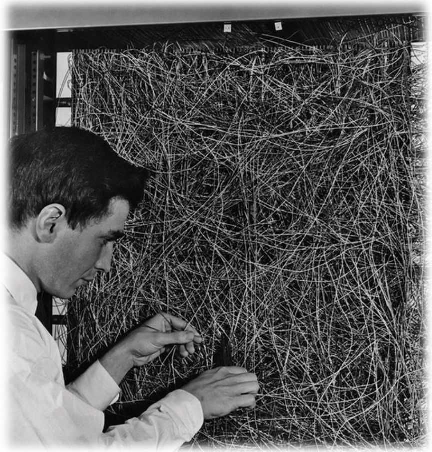
McCulloh & Pitts 1943
Wiener 1948
Alan Turing 1950
Frank Rosenblatt 1957




Máquinas electrónicas:

Mark I, 1944
Widrow & Hoff 1960

1 operación cada 3 segundos



Explosión Tecnológica

Microprocesadores

Intel 8086, 1978
50 mil operaciones por segundo
Intel i5, 2018
25 mil millones de operaciones por segundo

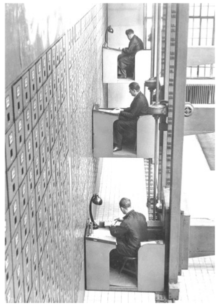
2010s La era de las GPUs

Playstation 4s, 2016
1.8 TFlops (~90 x intel i5)
Playstation 5s, 2020
10.2 TFlops (~411 x intel i5)

Nvidia RTX Titan, 2018
16 TFlops (~640 x intel i5)
Nvidia RTX 3090, 2020
35 TFlops (~1400 x intel i5
Nvidia RTX 4090, 2022
82 TFlops (~3280 x intel i5)

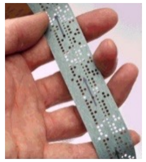
Almacenamiento

Sistema mecánico1937 (República Checa)



Capacidad de almacenamiento

Cinta perforada *1970*	*<1*	*KB*
Disco 3 ½	*1987* *1.4*	*MB*
DVD	*1995* *4.7*	*GB*



Velocidad de almacenamiento

Disco duro *2000* *18GB (48MB/s)*
HD estado sólido *2021* *1TB* (7000 MB/s)

Comunicaciones / Difusión

Proceedings papel 1995 / Revistas papel

Buscadores internet 1998

Google

Software gratuito / Toolkits 2010

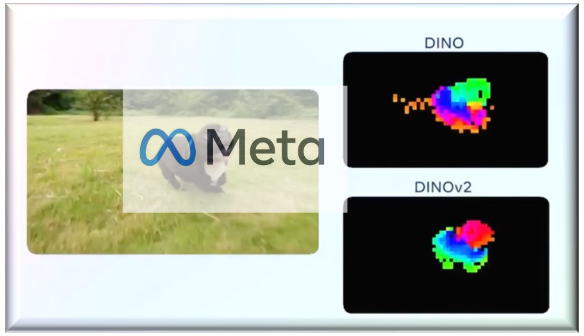
TensorFlow PYTORCH

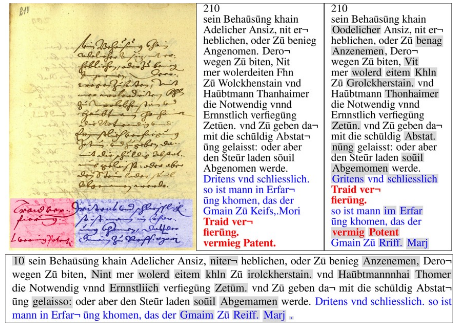
2008 *Redes sociales* /
plataformas de desarrollo colaborativo

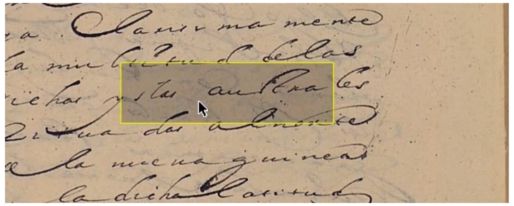
X GitHub

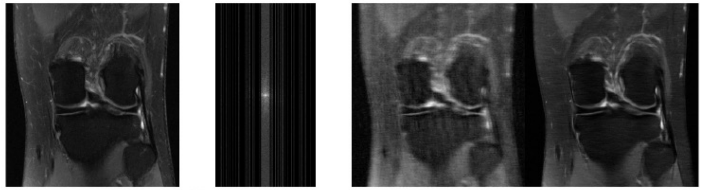
Campos de Aplicación

- Visión por Computador
- Reconocimiento de Escritura
- Ayuda al diagnóstico
- Transcripción de voz
- Traducción
- Algoritmos de Recomendación
- …

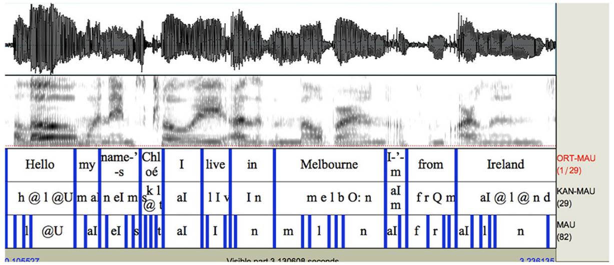








(a) Cropped and vertically flipped reconstruction from fully sampled k-space data (b) Rectangular masked k-space (c) Reconstruction via zero-filled IFFT (d) Deep-learning baseline UNET reconstruction

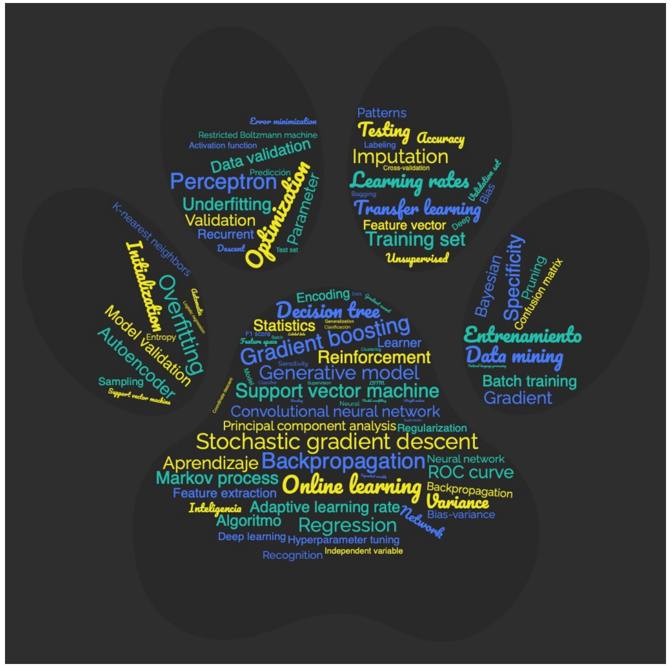


Aprendizaje Automático

- *"Learn from data"*

- Encontrar patrones y tendencias, entendiendo "lo que nos dicen los datos"

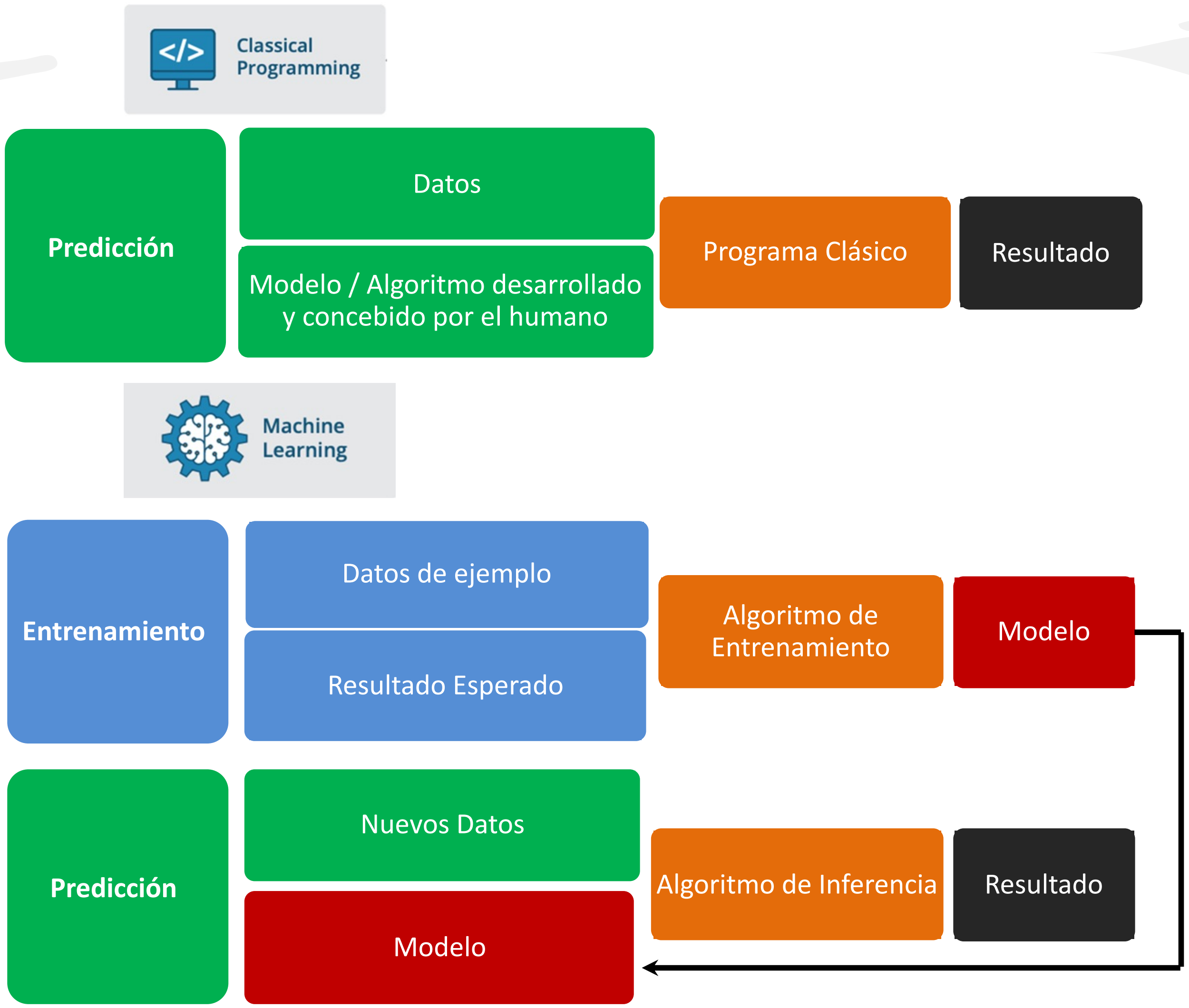
- Sistema automático que aprende de la **Experiencia (E)** para realizar una **Tarea (T)** determinada evaluada a través de una **Métrica (M)** establecida.



Tom Michael Mitchell
American computer scientist; Founders University Professor
Carnegie Mellon University (CMU).

Tom Mitchell [T. Mitchell. Machine Learning. McGraw Hill,1997]

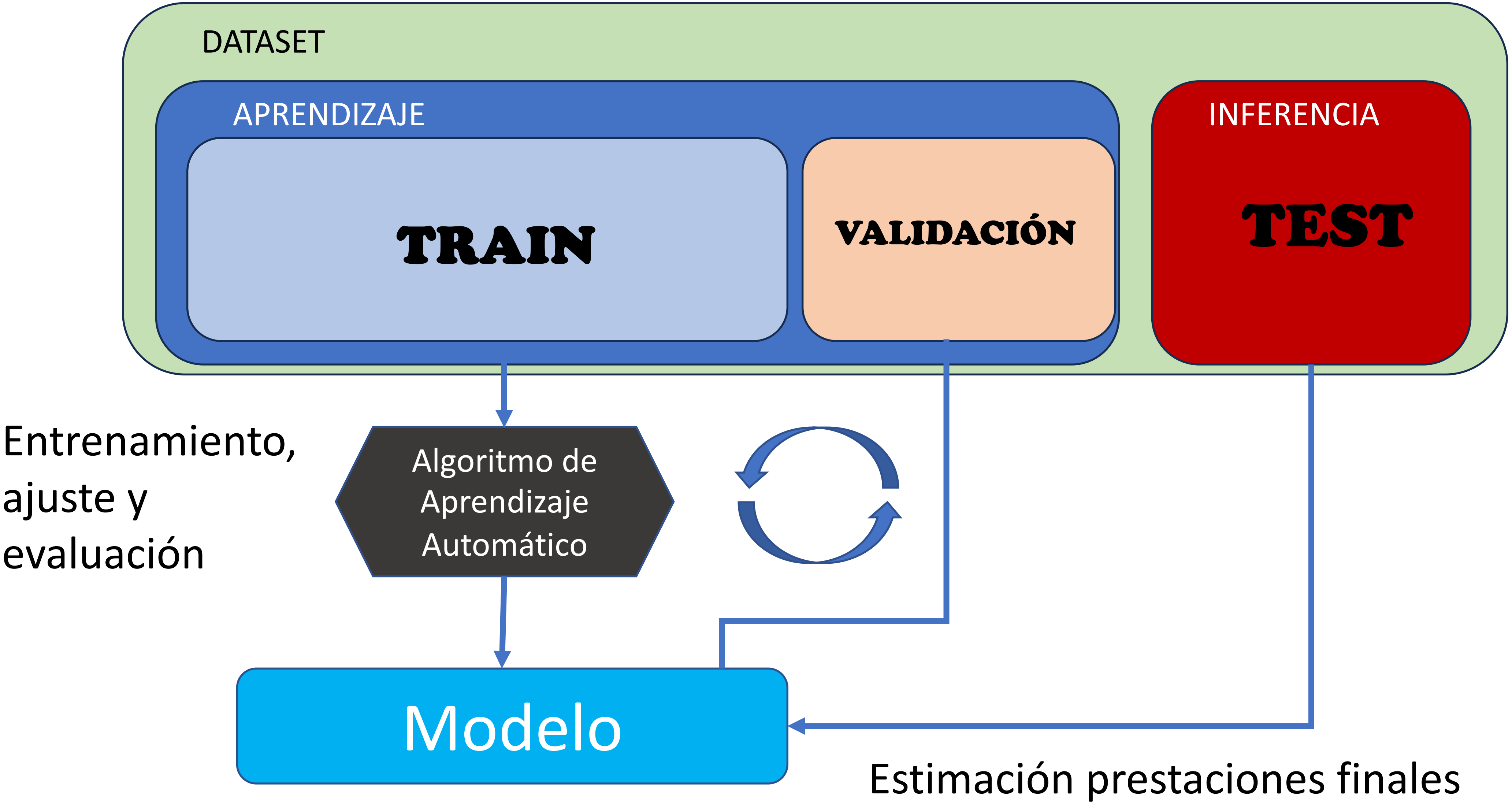
Aprendizaje Automático



Ofrece soluciones a **Problemas** demasiado **complejos** como para ser resueltos por **programas clásicos** concebidos por humanos

Diseño Experimental

- **Uso de los datos:**



Tipos de Tareas

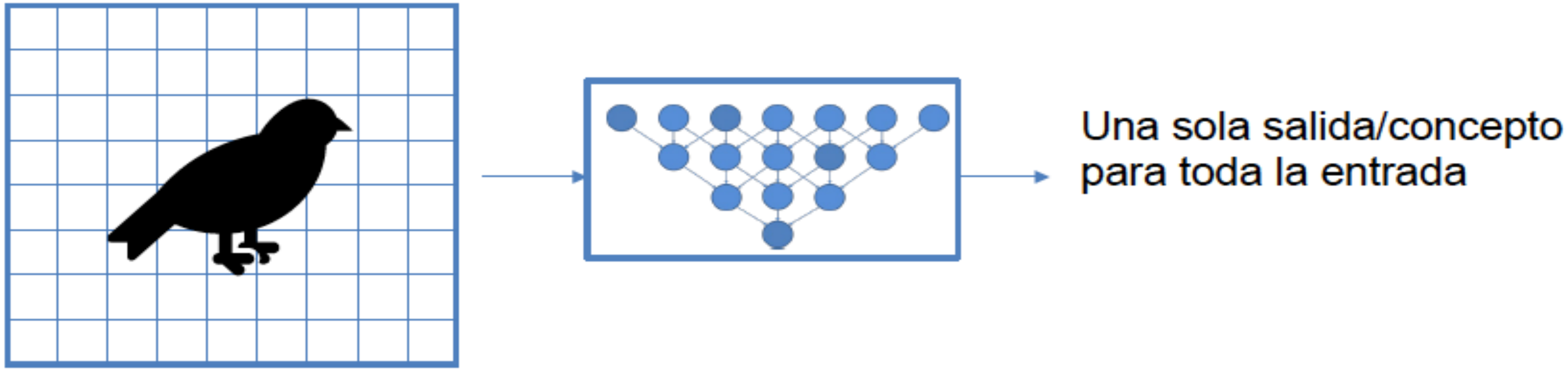
- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

Tipos de Tareas

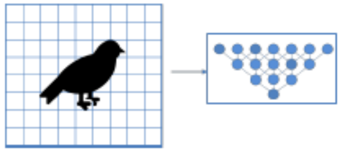
- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

CLASIFICACIÓN SIMPLE

¿Qué CONCEPTO hay tras una imagen/texto/audio ?



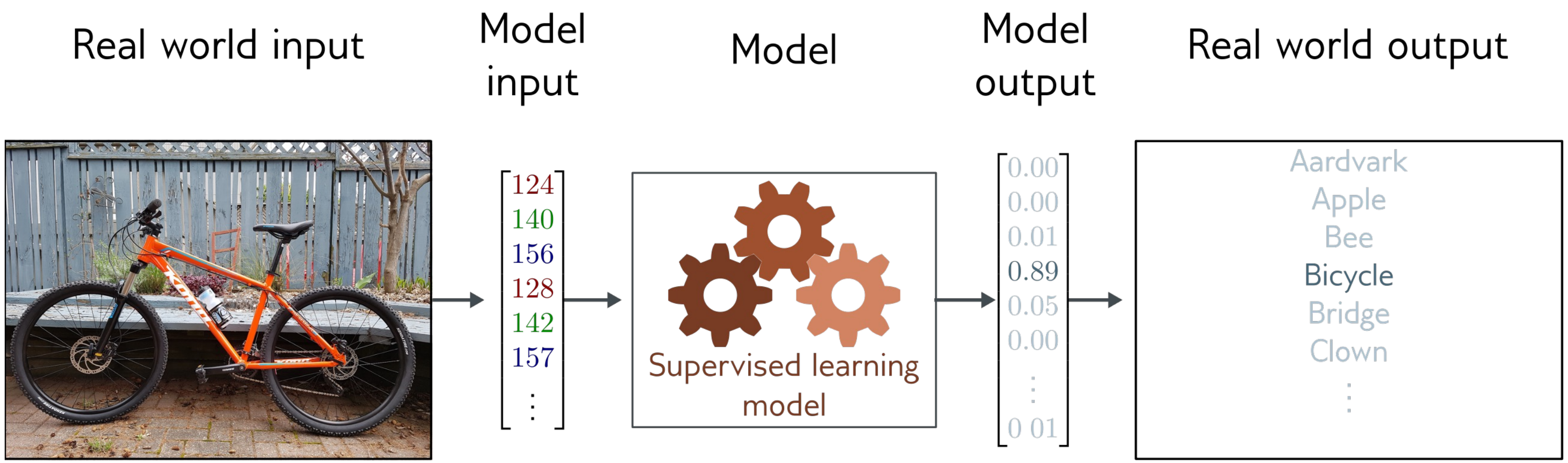
Tipos de Tareas



- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

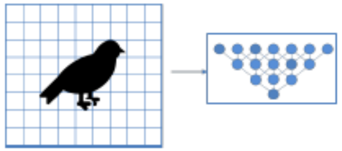
CLASIFICACIÓN SIMPLE

¿Qué CONCEPTO hay tras una imagen/texto/audio ?



A cada entrada, el modelo decide UNA entre C posibles respuestas (Clases)

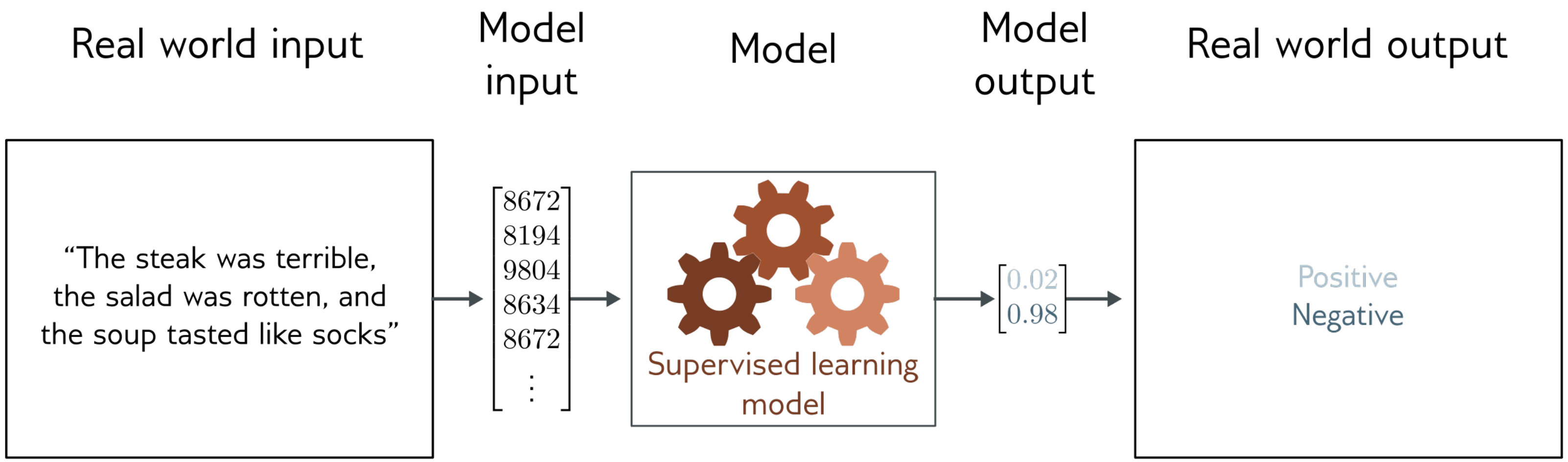
Tipos de Tareas



- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

CLASIFICACIÓN SIMPLE

¿Qué CONCEPTO hay tras una imagen/texto/audio ?



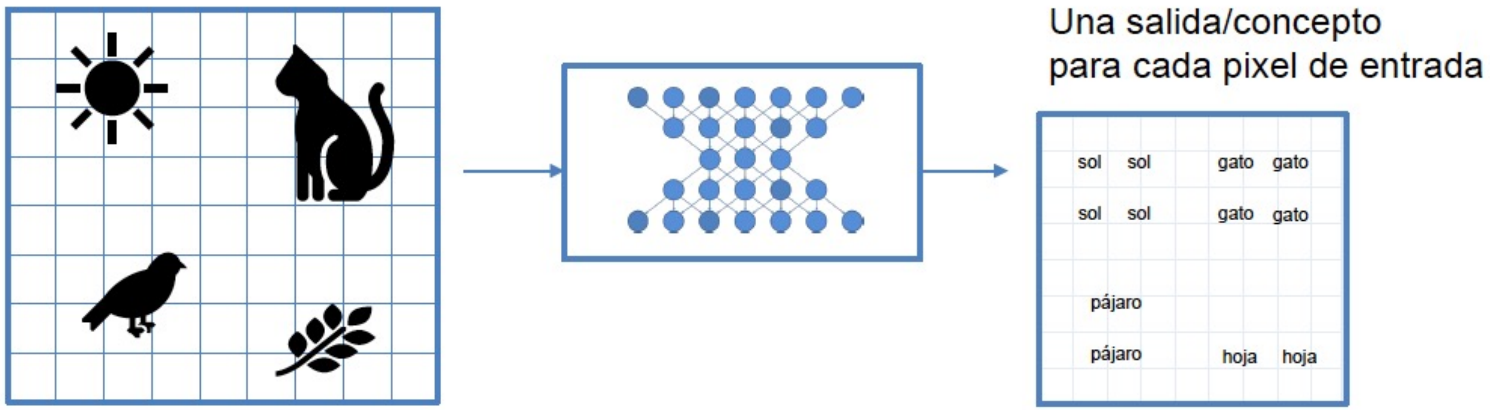
En muchas ocasiones C=2 (Caso binario SI/NO)

Tipos de Tareas

- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

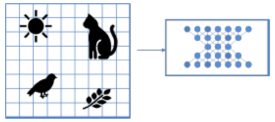
CLASIFICACIÓN MÚLTIPLE

¿Qué CONCEPTO hay en cada zona/pixel/fragmento de una imagen/texto/audio ?



Enunciar VARIAS PROPIEDADES/CONCEPTOS de una zona/pixel/fragmento de una imagen/texto/audio ?

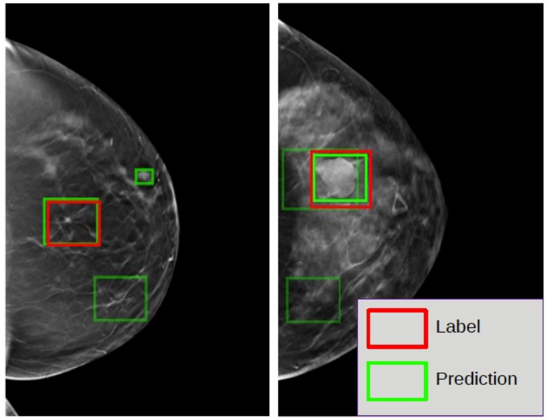
Tipos de Tareas



- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

CLASIFICACIÓN MÚLTIPLE

¿Qué CONCEPTO hay en cada zona/pixel/fragmento de una imagen/texto/audio ?



DBTex Challenge:
SPIE-AAPM-NCI DAIR Digital Breast Tomosynthesis Lesion Detection Challenge

AMERICAN ASSOCIATION of PHYSICISTS IN MEDICINE

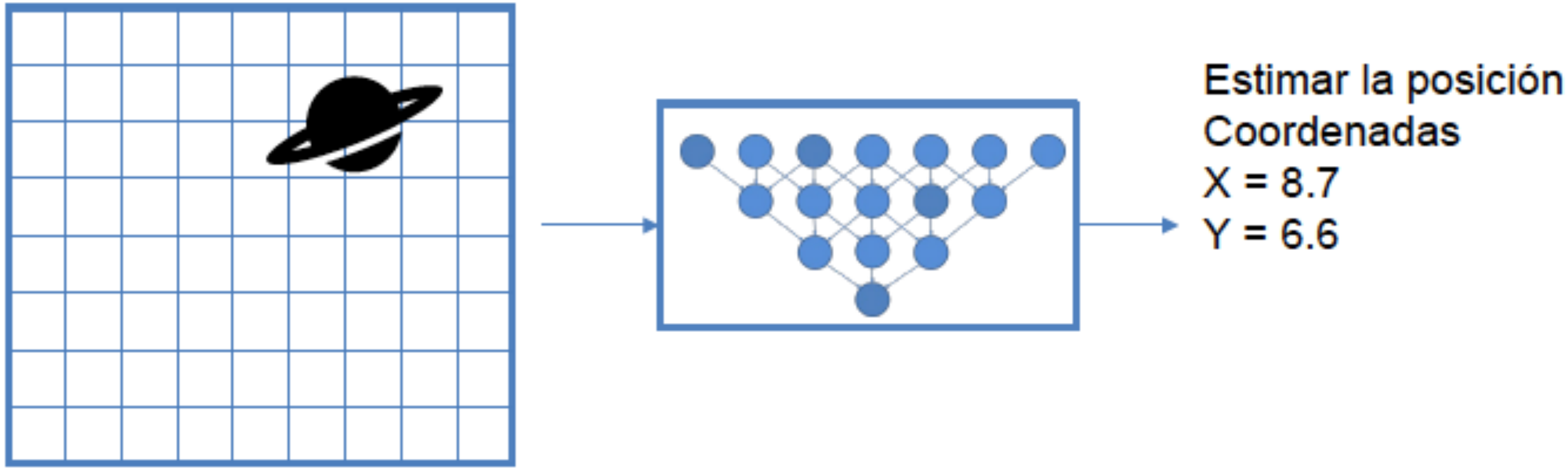
Enunciar VARIAS PROPIEDADES/CONCEPTOS de una zona/pixel/fragmento de una imagen/texto/audio ?

Tipos de Tareas

- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

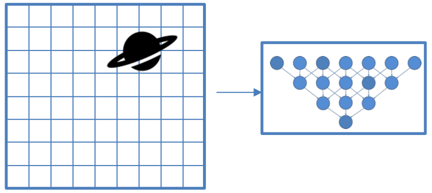
REGRESIÓN SIMPLE

Realizar una PREDICCIÓN NUMÉRICA a partir de una imagen/texto/audio ?



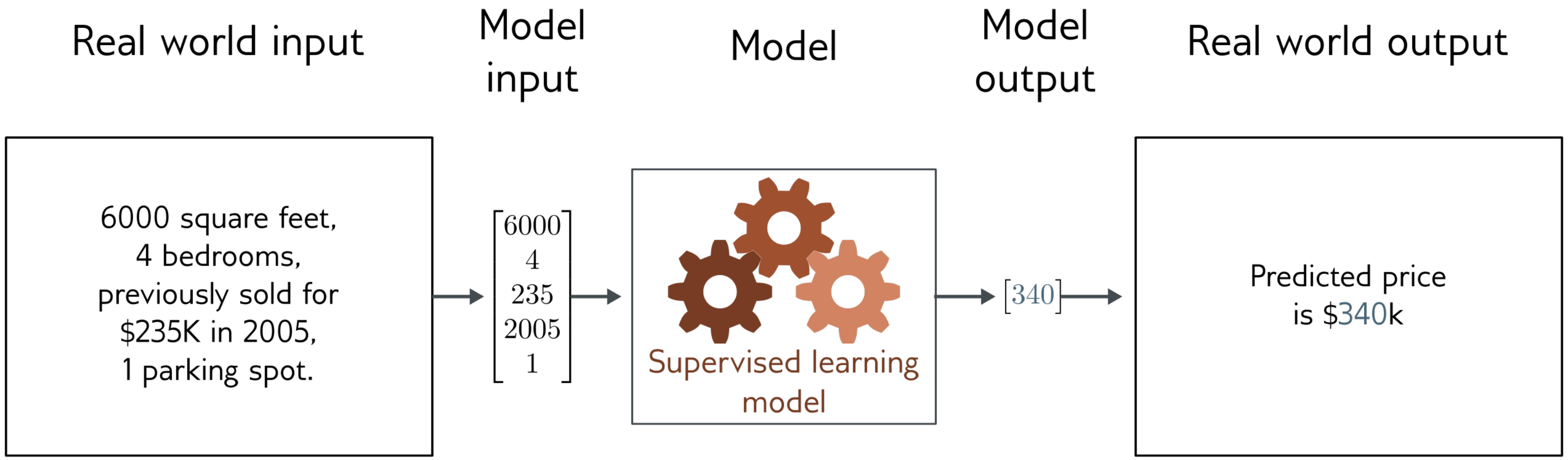
Tipos de Tareas

- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:



REGRESIÓN SIMPLE

Realizar una PREDICCIÓN NUMÉRICA a partir de una imagen/texto/audio ?



Ante cada entrada, el modelo ofrece UNA respuesta

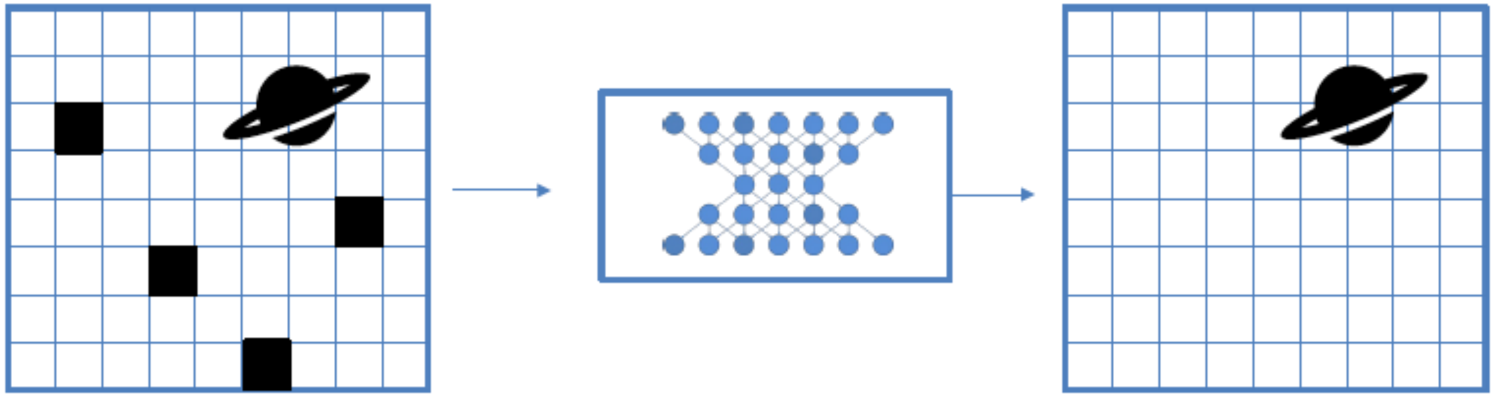
Tipos de Tareas

- **Clasificación:**
  - Simple:
  - Múltiple:

- **Regresión:**
  - Simple:
  - Múltiple:

REGRESIÓN MÚLTIPLE

PREDECIR VARIOS VALORES NUMÉRICOS para cada zona/pixel/fragmento de una entrada



MÉTRICAS DE EVALUACIÓN

- **Clasificación:**
  - Caso Binario

T = P+N		**VALOR REAL**	
		POSITIVO (P = TP + FN)	NEGATIVO (N = FN + TN)
VALOR PREDICHO	POSITIVO	**TRUE POSITIVE (TP)**	**FALSE POSITIVE (FN) TYPE II ERROR FALSA ALARMAFALSA ACEPTACIÓN**
	NEGATIVO	**FALSE NEGATIVE (FN) TYPE I ERROR FALSO RECHAZO MISS**	**TRUE NEGATIVE (TN)**

MÉTRICAS DE EVALUACIÓN

- **Clasificación:**
  - Caso Binario:
    - **Accuracy**: $\mathrm{ACC} = \frac{TP+TN}{P+N}$

T = P+N		VALOR REAL	
		POSITIVO (P = TP + FN)	NEGATIVO (N = FN + TN)
VALOR PREDICHO	POSITIVO	**TRUE POSITIVE (TP)**	**FALSE POSITIVE (FN)** **TYPE II ERROR** **FALSA ALARMAFALSA ACEPTACIÓN**
	NEGATIVO	**FALSE NEGATIVE (FN)** **TYPE I ERROR** **FALSO RECHAZO** **MISS**	**TRUE NEGATIVE (TN)**

POR COLUMNAS

- **Sensitivity / Recall / True Positive Rate**: $\mathrm{TPR} = \frac{TP}{P}$
  - También como complementario a **Miss rate/ False Negative Rate**: $\mathrm{TPR} = 1 - FNR$
- **Especificidad / True Negative Rate**: $\mathrm{TNR} = \frac{TN}{N}$
  - También como complementario a **False Alarm Rate / False Positive Rate:** $\mathrm{TNR} = 1 - \mathrm{FPR}$

POR FILAS

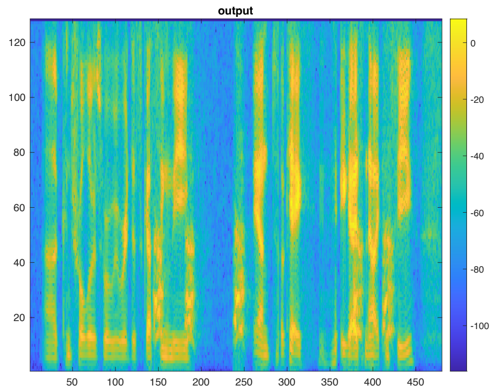
- **Valor Predictivo Positivo / Precisión:** $\mathrm{PPV} = \frac{TP}{TP+FP}$
- **Valor Predictivo Negativo:** $\mathrm{NPV} = \frac{TN}{TN+FN}$

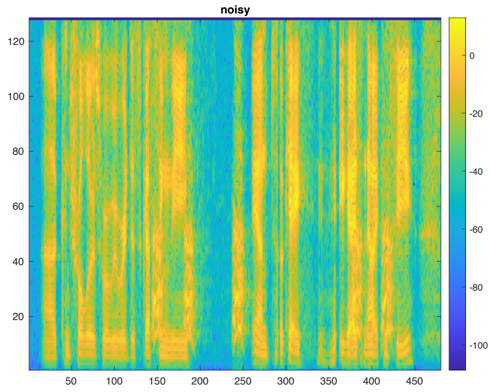
Tipos de Tareas

- **Clasificación:**
  - Simple:
  - Múltiple:
- **Regresión:**
  - Simple:
  - Múltiple:

REGRESIÓN MÚLTIPLE

PREDECIR VARIOS VALORES NUMÉRICOS para cada zona/pixel/fragmento de una entrada

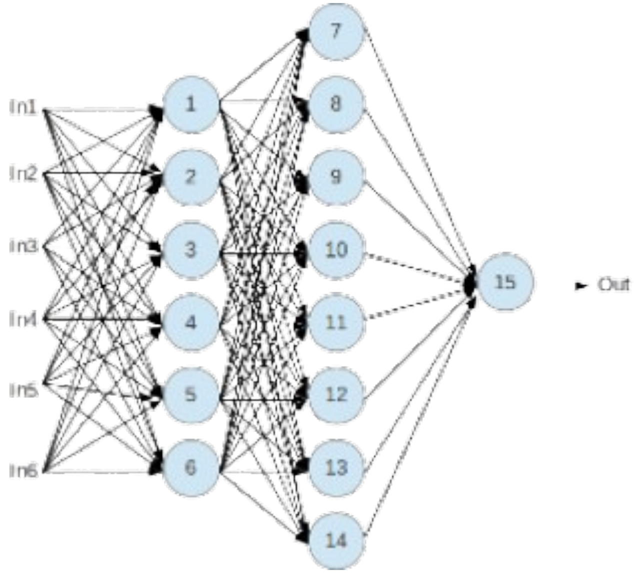




Elimina el ruido de un audio ruidoso

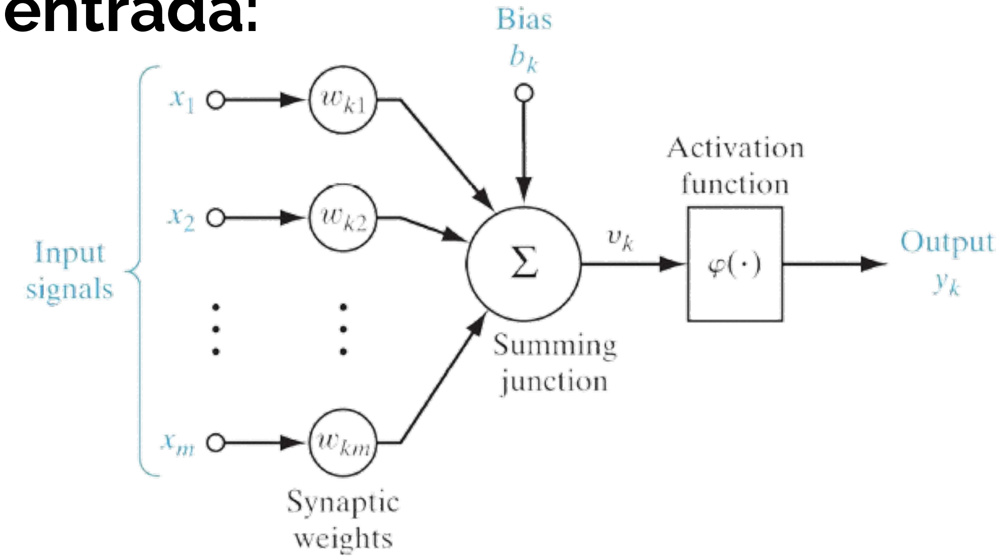
Redes Neuronales Artificiales

- **Conjunto de elementos simples de procesado interrelacionados:**
  - Distribución del procesado de la información
  - Cada unidad recibe información de otras, los agrega y transmite una respuesta a través de una función de activación
- **Cada elemento se conecta con otros**
  - Conexiones Sinápticas:
    - Cada conexión tiene un peso
    - Se ajustan a partir de ejemplos
      - Datos de entrenamiento
      - Algoritmos de aprendizaje



Perceptrón

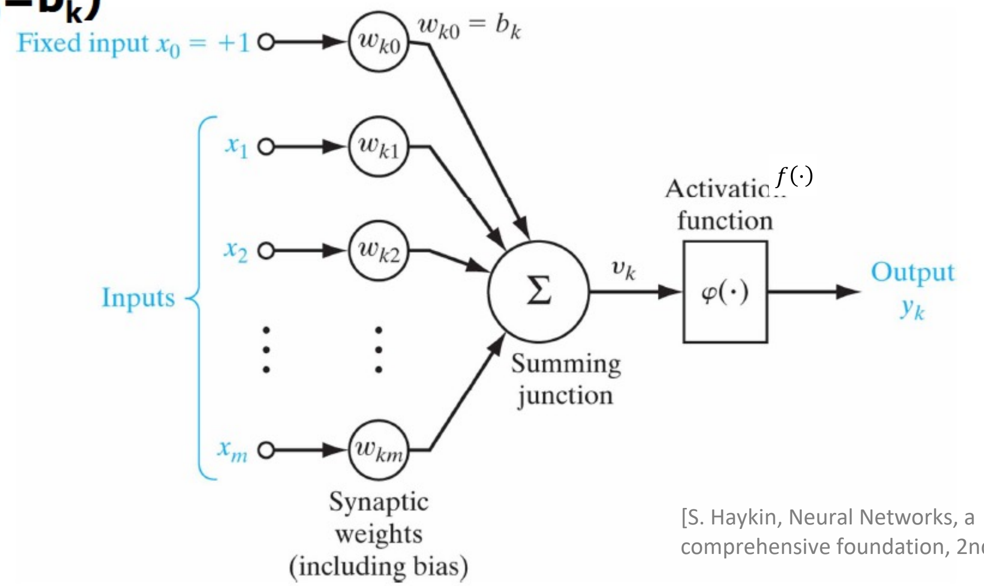
- **Combinación lineal de valores a la entrada:**



Vector de pesos $\underline{w}$ + sesgo $\underline{b}$

Transformación afín

(w_{k0}=b_k)

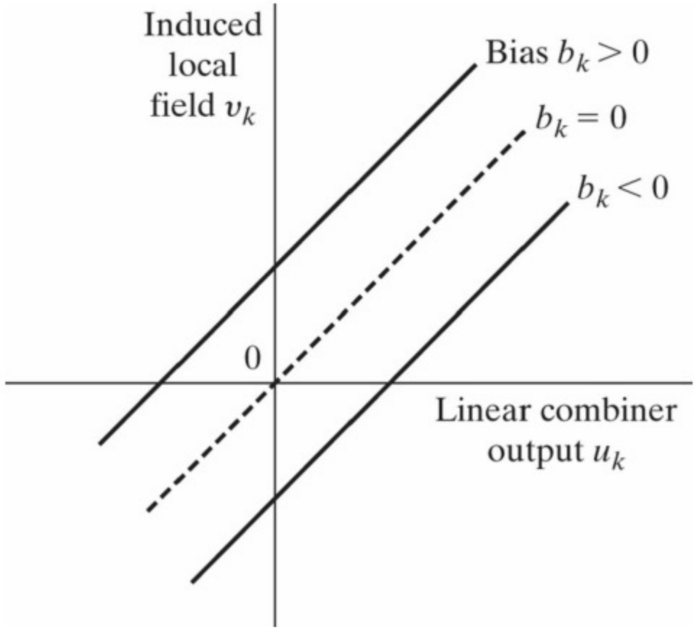


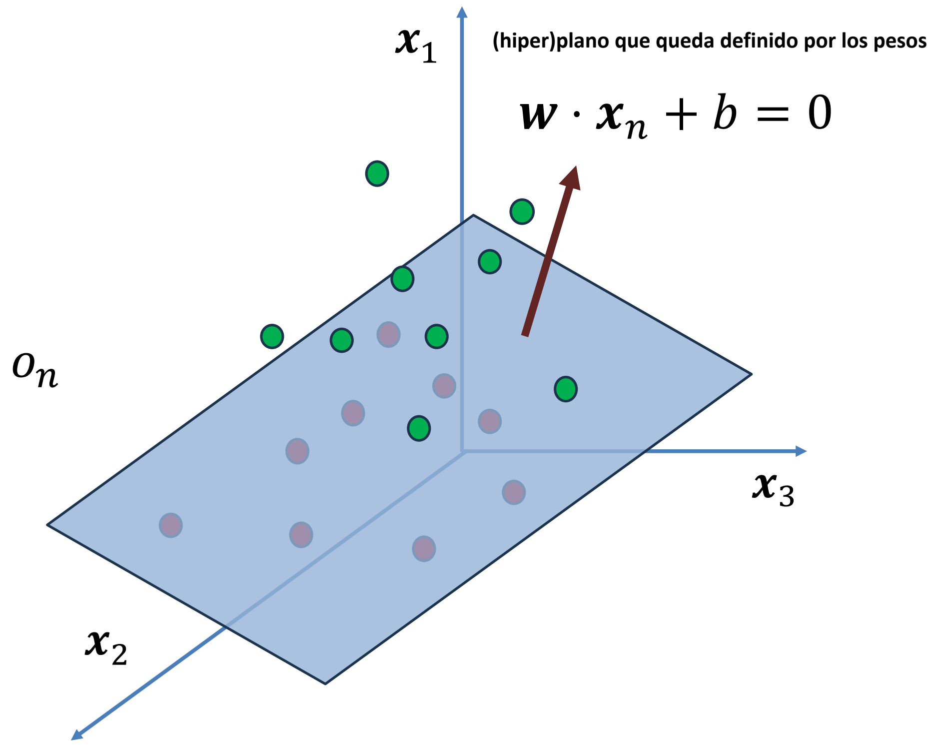
Función de activación, $f(x_n)$, no lineal

$$o_n = f(\boldsymbol{x}_n) = \begin{cases} +1, & \boldsymbol{w} \cdot \boldsymbol{x}_n + b > 0 \\ -1, & \boldsymbol{w} \cdot \boldsymbol{x}_n + b \leq 0 \end{cases}$$

[S. Haykin, Neural Networks, a comprehensive foundation, 2nd Edition]

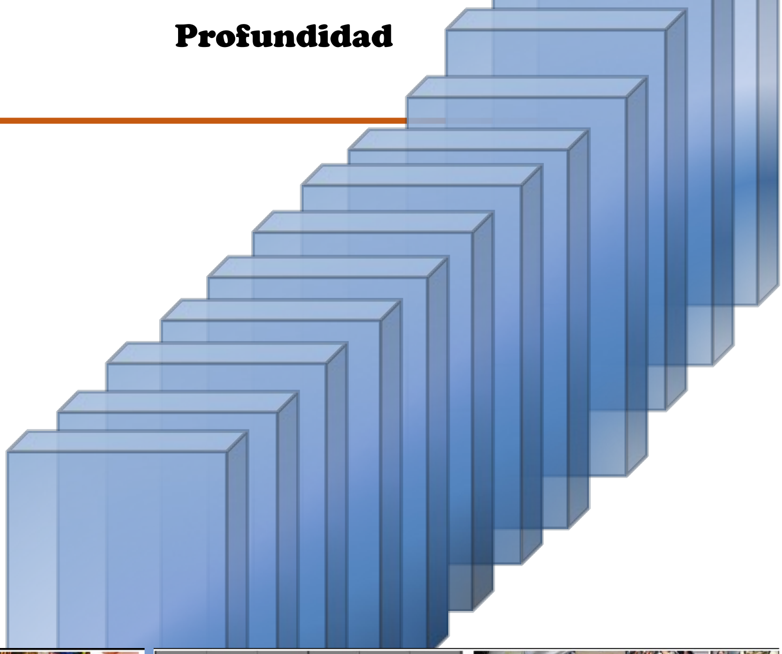
Perceptrón

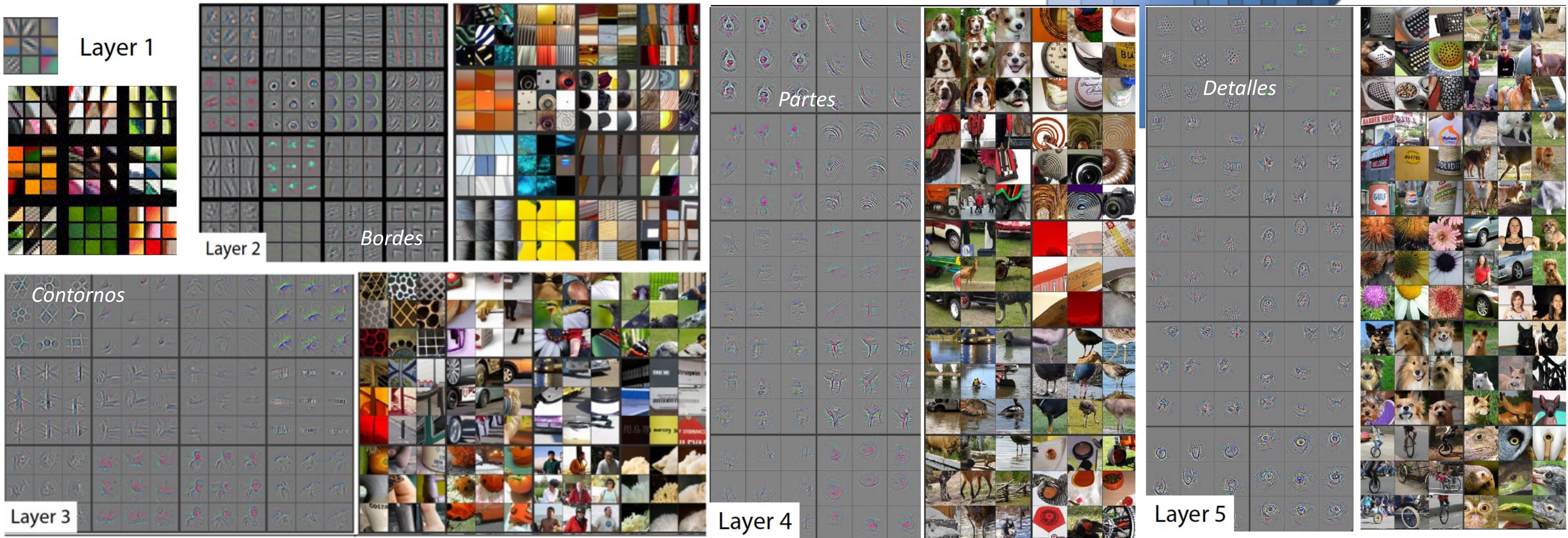

Induced
local
field v_k
Bias $b_k > 0$
$b_k = 0$
$b_k < 0$
0
Linear combiner
output u_k


$\boldsymbol{x}_1$
(hiper)plano que queda definido por los pesos
$\boldsymbol{w} \cdot \boldsymbol{x}_n + b = 0$
o_n
$\boldsymbol{x}_3$
$\boldsymbol{x}_2$

Redes Neuronales Artificiales

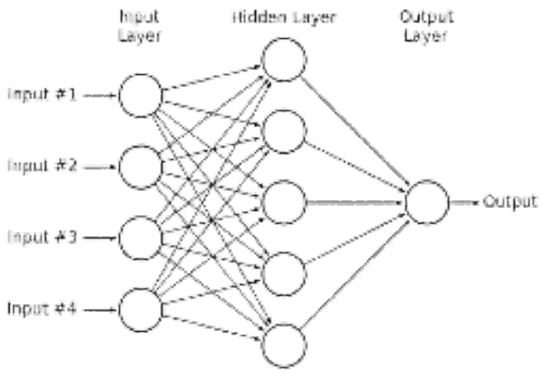
- **Procesado en capas:**
  - Aprenden diferentes niveles de representación de los datos, desde lo más simple, hasta lo más abstracto

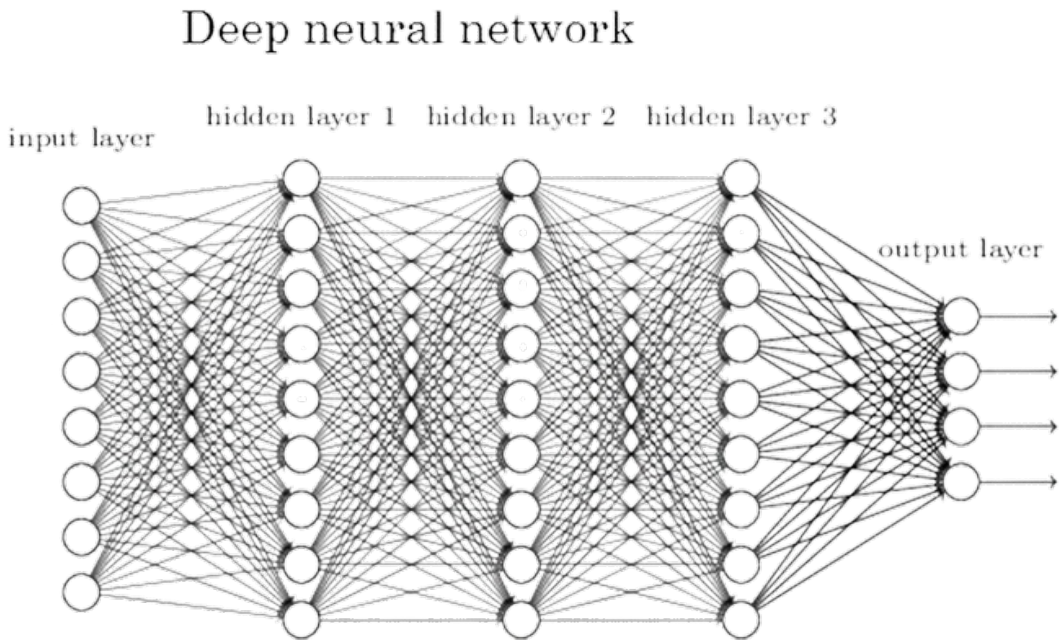




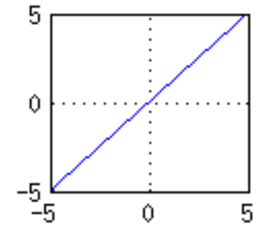
Multilayer Perceptron / Deep Feedforward Network

- **Múltiples Neuronas (perceptrón) básicas organizadas en capas**
    - **Arquitectura de la red**:
        - **Capa de entrada**: Toma los datos de entrada (en forma vector)
            - Conviene acondicionar los datos: Normalización
        - **Capas Ocultas**: Sin contacto ni con la entrada ni con la salida
            - Agrupación de perceptrones
        - **Capas de Salida**:
            - Resultado del proceso



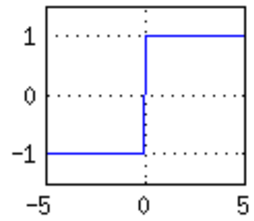


Funciones de Activación



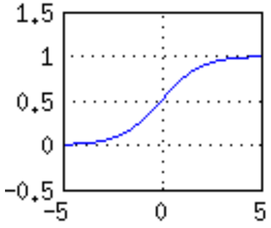
Linear

$$f(x) = x$$



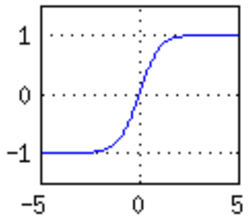
Sign

$$f(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$



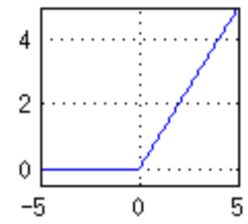
Logistic

$$f(x) = \frac{1}{1 + \exp(-x)} \qquad f'(x) = f(x)(1 - f(x))$$



Tanh

$$f(x) = \tanh(x) \qquad f'(x) = (1 - f(x)^2)$$

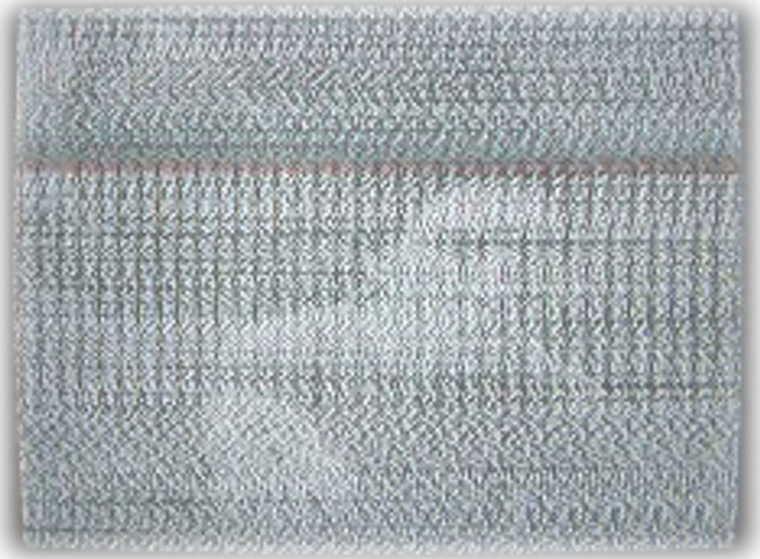


ReLu

$$f(x) = \max(0, x) \qquad f'(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

Entrenamiento / Aprendizaje

- **Ajuste de los parámetros del modelo:**
  - Selección de los datos de entrenamiento
  - Exposición de los datos a la red
  - Observación de la salida
    - Variar los pesos para hacer disminuir el error



Deep Forward Networks

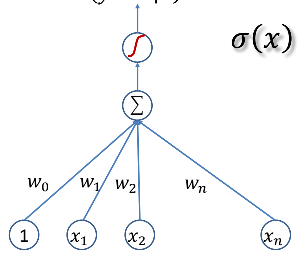
- **Tipos de datos:**
  - **Etiquetas**:
    - Son vectores objetivo: $\boldsymbol{y_n}$
    - Si la red actúa como regresor (prediciendo valores):
      - $\boldsymbol{y_n} \in \mathbb{R}^{\boldsymbol{D}}$
    - Si la red actúa como un clasificador
      - $\boldsymbol{y_n} \in \{\boldsymbol{0}, \boldsymbol{1}\}^{\boldsymbol{C}}$ (C clases) One-hot encoding
      - $\boldsymbol{y_n} \in \{\boldsymbol{0}, \boldsymbol{1}, \dots, \boldsymbol{C} - \boldsymbol{1}\}$ (C clases)
  - **Salidas**:
    - Son el producto de la red: $\boldsymbol{y_n}$
    - Tendrán la misma dimensión y serán del mismo tipo que las etiquetas
  - **Entradas**:
    - Son los datos que alimentan la red: $\boldsymbol{x_n}$

Deep Forward Networks

- **Entrenamiento:**
  - **Pares de datos** $(\boldsymbol{x_n}, \boldsymbol{y_n})$, $\boldsymbol{x_n} \in \mathbb{R}^{\boldsymbol{D}}$; $\boldsymbol{y_n} \in \mathbb{R}^{\boldsymbol{D}}$ ó $\boldsymbol{y_n} \in \{\boldsymbol{0}, \boldsymbol{1}\}^{\boldsymbol{C}}$ …
  - **Función de coste** $\boldsymbol{J(X, Y, \Theta)}$**:**
    - Es una función de los ejemplos $\boldsymbol{X}$, de las etiquetas $\boldsymbol{Y}$ y de los parámetros del modelo $\boldsymbol{\Theta}$
      - **Error cuadrático:**
        - $\boldsymbol{J} = \sum_{\boldsymbol{n}} (\boldsymbol{y_n} - \boldsymbol{o_n})^2$
      - **Errores de Clasificación**
        - $J(\hat{y}, y) = \sum_n L_{CE}(\hat{y}_n, y_n)$

Caso Binario

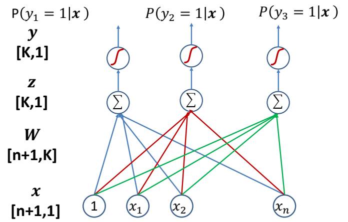
$$L_{CE}(\hat{y}_n, y_n) = -\mathrm{y_m} \log \mathrm{y_m} - \\ -(1 - y_n) \log(1 - \hat{y}_n) = \\ = -\log p(y_n | \boldsymbol{x_n})$$



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

Caso Multiclase

$$L_{CE}(\hat{\boldsymbol{y}}_n, \boldsymbol{y_n}) = -\sum_{k=1}^{K} y_n(k) \log \hat{y}_n(k)$$



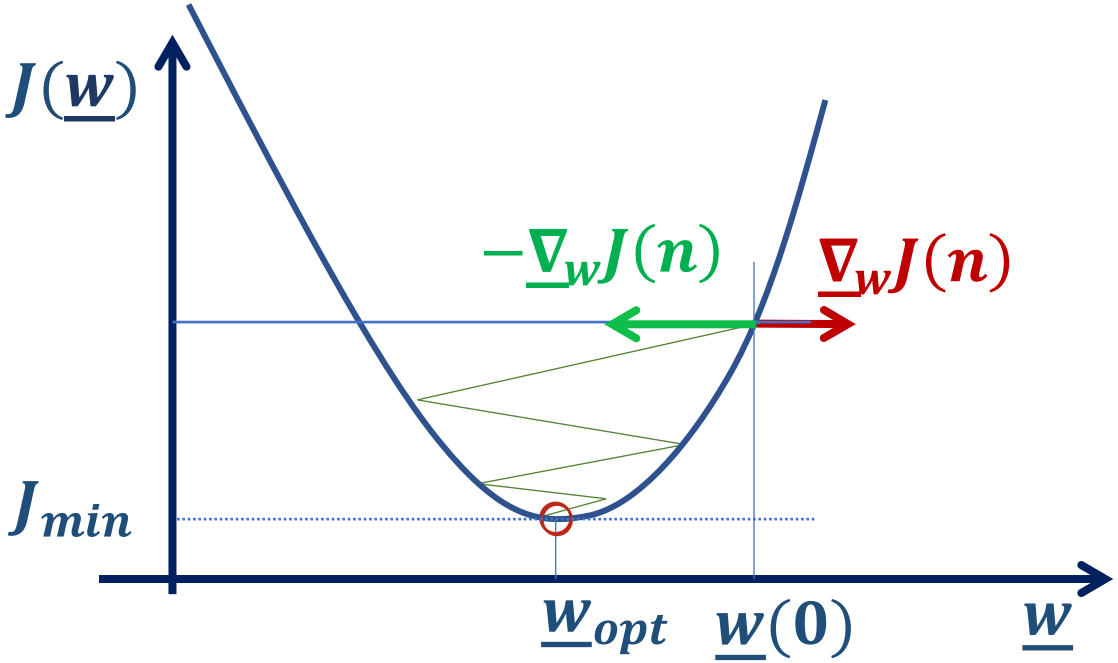
$$\mathrm{softmax}(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{K} exp(z_j)}$$

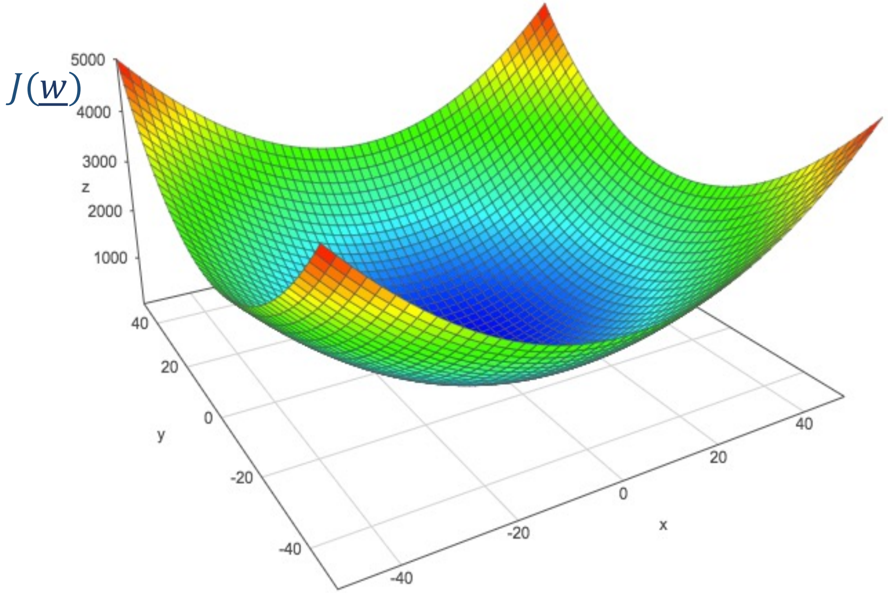
- Objetivo:
  - Encontrar los parámetros del modelo que minimicen la función de coste:

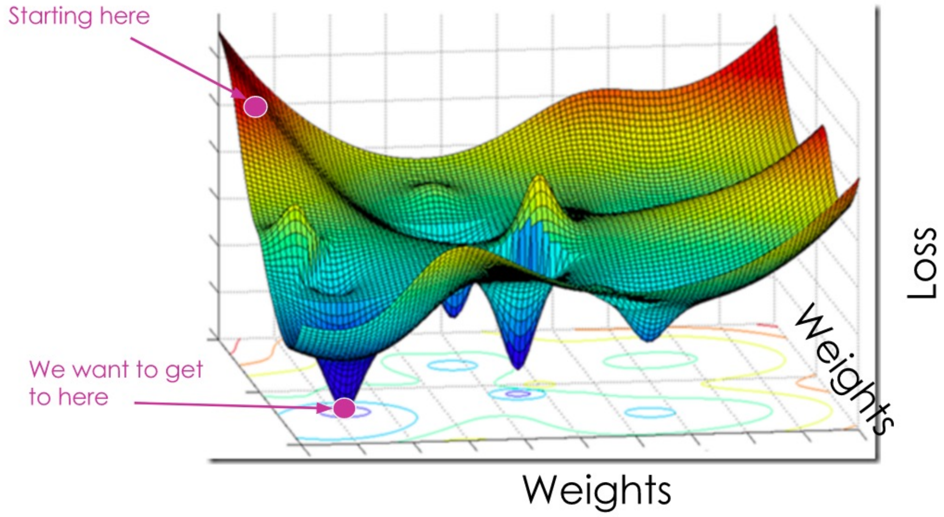
$$\{\hat{\mathbf{w}}, \hat{b}\} = \arg\min_{\boldsymbol{w}\, \boldsymbol{b}} \mathcal{J}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{w}, b)$$

Entrenamiento

- **Gradient Based Optimization:**
  - Superficie de error compleja
  - Múltiples mínimos locales
  - No hay garantía de llegar al mínimo global



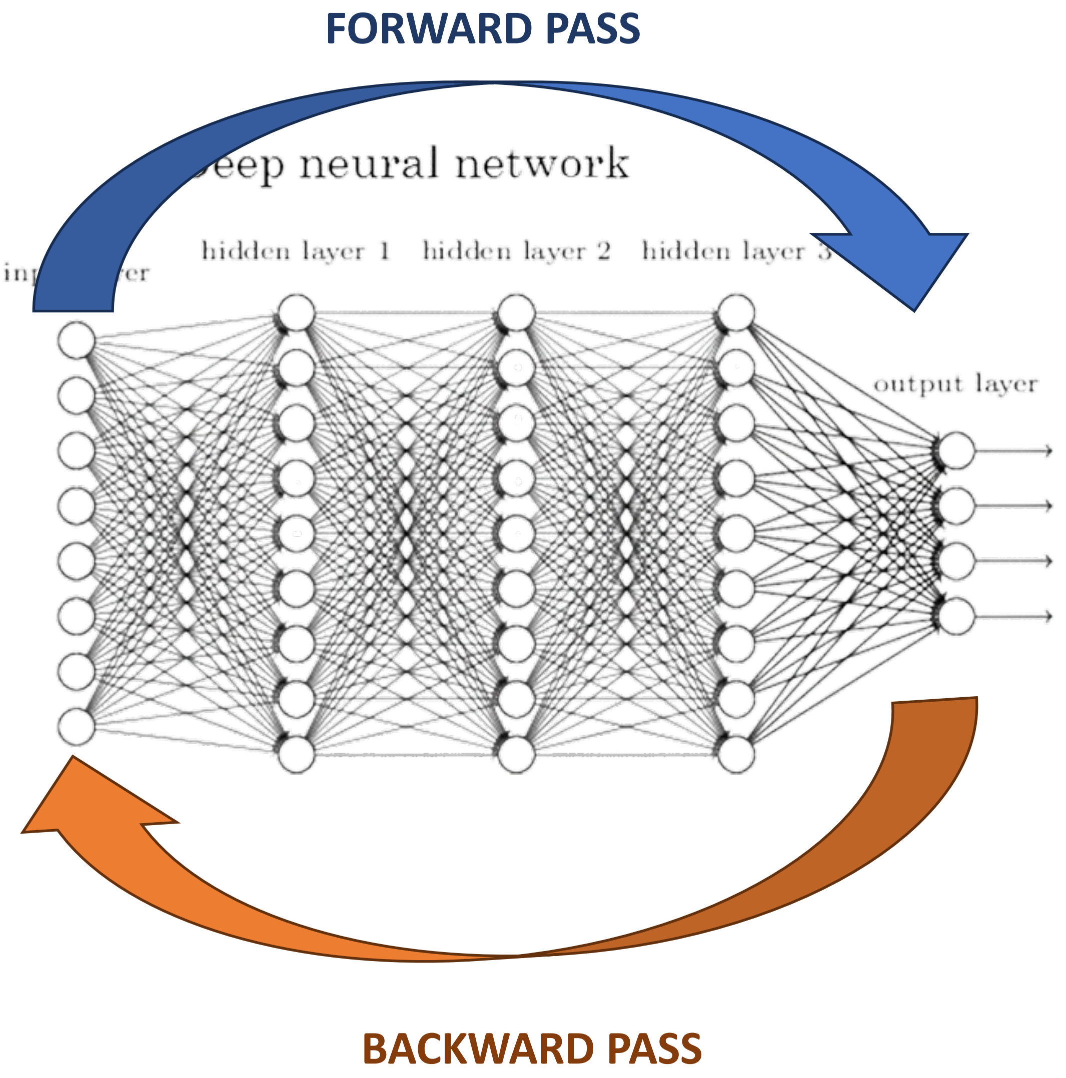




Entrenamiento

- **BACKPROPAGATION:**
  - Método para implementar el aprendizaje basado en gradiente (GRADIENT BASED)
- Red Neuronal: Función (todo lo compleja que podamos imaginar) de los pesos $\boldsymbol{w}$ y de la entrada $\boldsymbol{x}$: $o_n = f(\boldsymbol{w}, \boldsymbol{x})$
- $\boldsymbol{J(X, Y, \Theta)}$: Discrepancia entre la salida obtenida y la esperada (etiquetas) para cada ejemplo.
- **Idea Básica**: Calcular de forma eficiente todas las **derivadas parciales** de la función de coste, $\boldsymbol{J(X, Y, \Theta)}$, respecto de todos los parámetros ajustables de la función, sus pesos $\boldsymbol{w}$, para un valor dado de la entrada $\boldsymbol{x}$

$$\boldsymbol{J} = \sum_{n} J_n(o_n, y_n) \qquad \frac{\partial J}{\partial w_i} = \sum_{n} \frac{\partial J_n}{\partial o_n} \frac{\partial o_n}{\partial w_i}$$



Entrenamiento

- Los pesos se ajustan siguiendo la dirección contraria al gradiente

$$\boldsymbol{w_i^{(j+1)} = w_i^{(j)} - \eta \frac{\partial J}{\partial w_i} = w_i^{(j)} - \eta \sum_n \frac{\partial J_n}{\partial o_n} \frac{\partial o_n}{\partial w_i}}$$

- Este proceso se realza de forma iterativa (como en los algoritmos adaptativos) utilizando los datos de entrenamiento

- $\boldsymbol{\eta}$ es lo que se conoce como tasa de aprendizaje o learning rate :

  - Valores grandes: aprendizaje más rápido y menos preciso.

  - Valores pequeños: aprendizaje más lento, pero más preciso

Entrenamiento

- Es necesario conocer el gradiente para todas y cada una de las capas
- Esta información debe fluir desde la capa de salida hacia la capa de entrada.
- Obtener la expresión analítica del gradiente es posible (sencillo) pero computacionalmente caro
  - Backpropagation hace esto computacionalmente eficiente, aplicando la regla de la cadena (chain rule)

Si f y g son funciones reales de variable real: $y = g(x) \quad z = f(y) = f(g(x))$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

Si $\underline{\boldsymbol{x}} \in \mathbb{R}^m$; $\underline{\boldsymbol{y}} \in \mathbb{R}^n$; $\boldsymbol{g}$ mapea de $\mathbb{R}^m$ a $\mathbb{R}^n$, y f mapea de $\mathbb{R}^n$ a $\mathbb{R}$

$\underline{\boldsymbol{y}} = \boldsymbol{g}(x) \quad z = f\left(\underline{\boldsymbol{y}}\right) = f(\boldsymbol{g}(\underline{\boldsymbol{x}}))$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

Computational Graphs

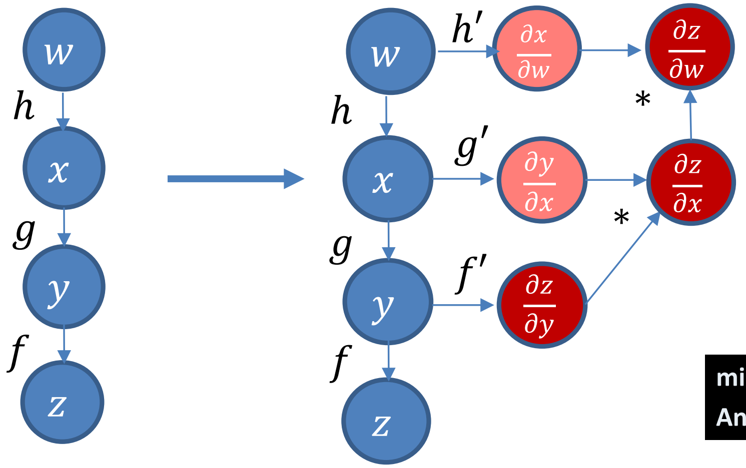
- ¿Como llevar la información del gradiente a todos y cada uno de los pesos de forma eficiente?
  - Se crean grafos de las expresiones involucradas en el cálculo de las salidas
  - Estos grafos indican qué depende de qué y cómo

Forward Pass

$$z = f(y) = f\big(g(x)\big) = \\ = f(g(h(w)))$$

Backward Pass

$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$



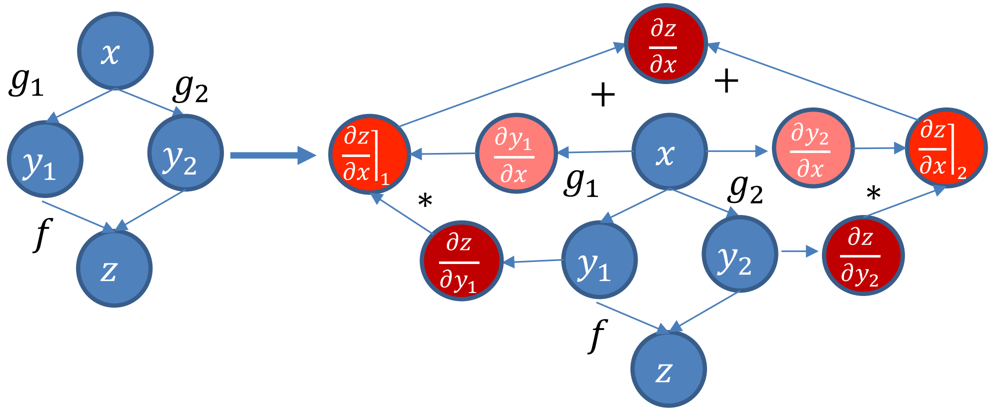
micrograd
Andrej Karpathy

Forward Pass

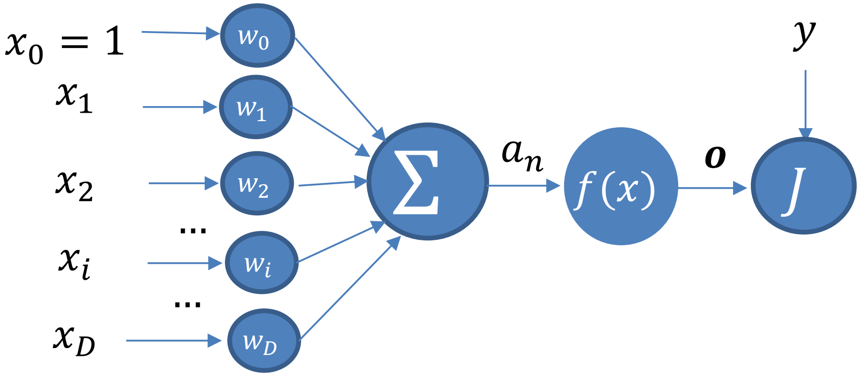
$$z = f(y_1, y_2);$$
$$y_1 = g_1(x)$$
$$y_2 = g_2(x)$$

Backward Pass

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$



Perceptrón con función de coste MSE:



Forward Pass

$$a_n = \sum_{i=0}^{D} w_i x_i = \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_{\boldsymbol{n}}$$

$$o_n = f(a_n)$$

$$e_n = y_n - o_n$$

$$J = \frac{1}{2}(e_n)^2$$

Backward Pass

$$\nabla_{w_i}(J) = \frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial o}\frac{\partial o}{\partial a}\frac{\partial a}{\partial w_i}$$

$$\frac{\partial a}{\partial w_i} = x_i \quad i = 1,2,\ldots,D \qquad \frac{\partial a}{\partial w_0} = x_0 = 1$$

$$\frac{\partial o}{\partial a} = f'(a) = f'(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_{\boldsymbol{n}}) \qquad \frac{\partial J}{\partial o} = -(y - o) = -e$$

$$\frac{\partial J}{\partial w_i} = -e \cdot f'\left(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_{\boldsymbol{n}}\right) \cdot x_i \longrightarrow w_i^{(j+1)} = w_i^{(j)} - \eta \frac{\partial J}{\partial w_i} = w_i^{(j)} + \eta \cdot e \cdot f'\left(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_{\boldsymbol{n}}\right) \cdot x_i$$

Stochastic Gradient Descent - SGD

- El gradiente no se calcula haciendo uso de toda la base de datos de entrenamiento sino por bloques y además en orden aleatorio en cada iteración

- SGD converge a mínimos locales. Como la función de coste ya no es tan simple como en el caso lineal, el mínimo local no tiene por qué ser global

- Gradiente Instantáneo:
  - Actualización tras cada ejemplo de entrenamiento

$$w_i(\mathrm{n}+1) = w_i(n) - \eta \frac{\partial \mathcal{J}\left(\boldsymbol{x}_n, y_n, \boldsymbol{w}^{(n)}\right)}{\partial w_i}$$

- Gradiente por lotes, (Batch Gradient):
  - Actualización tras cada bloque k

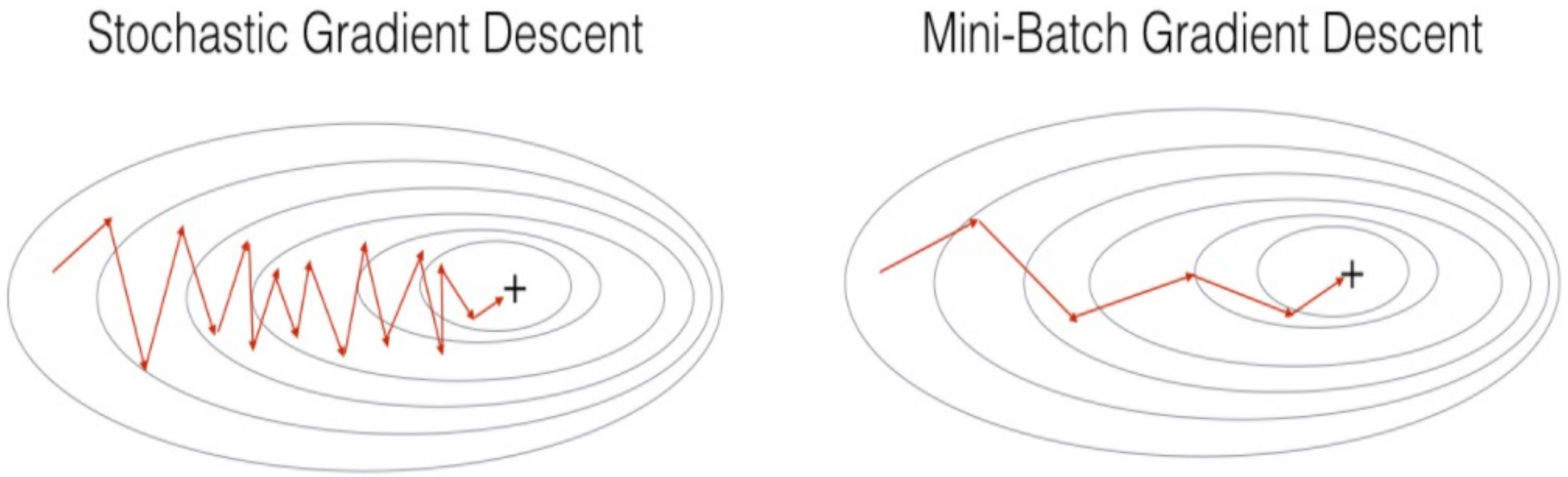
$$w_i(\mathrm{k}+1) = w_i(k) - \eta \frac{\partial \sum_{n \in k} \mathcal{J}\left(\boldsymbol{x}_n, y_n, \boldsymbol{w}^{(k)}\right)}{\partial w_i}$$

- Gradiente Global:
  - Cálculo para toda la base de datos de entrenamiento.
  - Actualización tras cada iteración (epoch) de toda la base de datos.
  - Alto coste computacional, robusto y baja velocidad de convergencia

$$\mathrm{w}_{\mathrm{i}}^{(j+1)} = \mathrm{w}_{\mathrm{i}}^{(j)} - \alpha \frac{\partial \sum_{n} \mathcal{J}\left(\boldsymbol{x}_n, y_n, \boldsymbol{w}^{(j)}\right)}{\partial w_i}$$

Backpropagation: Implicaciones y orientaciones prácticas

- **Mini-Batch processing:**
  - Mejor resultado, uso de “pequeños” lotes: **mini-batches**
  - Se puede optimizar mucho con el uso de GPU
  - Es conveniente aleatorizar los datos de entrenamiento
  - Uso de ejemplos más difíciles



Implicaciones y orientaciones prácticas

- **Normalización:**
  - Acondicionamiento de las entradas (y valores intermedios)
    - Conveniente para un buen funcionamiento
    - Eliminar media y aproximar varianza de todas las dimensiones
    - Decorrelación de las diferentes variables en juego
  - Z-NORM

MIN-MAX

$$\underline{x}' = \frac{\underline{x} - \underline{\mu}_x}{\sigma_x}$$

$$\underline{x}' = \frac{\underline{x} - \underline{x}_{min}}{\underline{x}_{max} - \underline{x}_{min}}$$

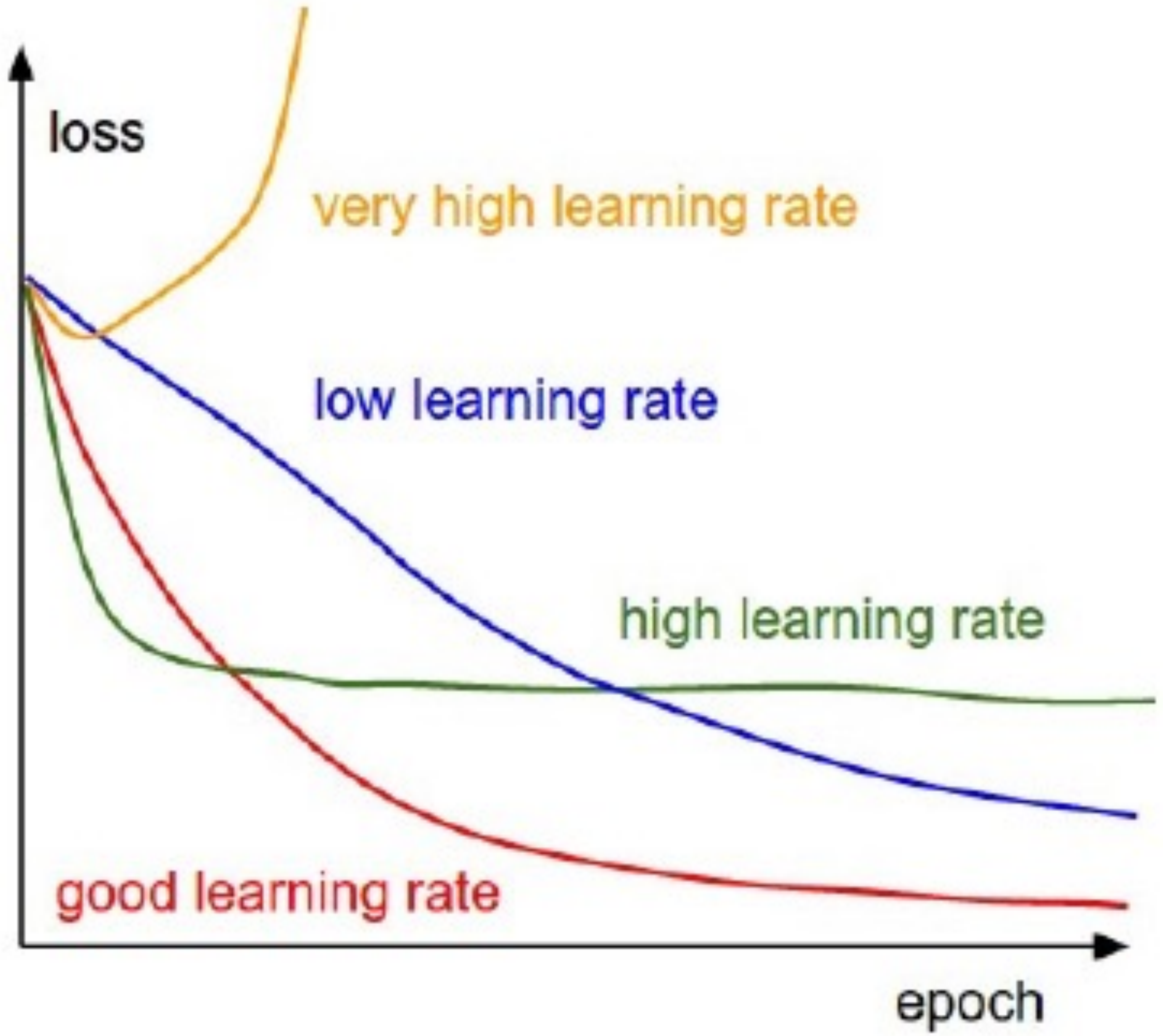
Implicaciones y orientaciones prácticas

- **INICIALIZACIÓN:**
  - Aleatorizar el valor inicial de los pesos
    - Hay que **"romper la simetría"**
    - Distintas unidades conectadas a las mismas entradas inicializadas con el mismo valor tenderán a aprender lo mismo.
    - Distribución uniforme con media 0 y desviación estándar dependiente de las conexiones que llegan al nodo.
      - pequeñas variaciones en los pesos pueden tener alta repercusión si hay muchas conexiones (fan-in alto)
  - Convergencia vs Generalización:
    - Buena convergencia, llegada a un punto estable rápido
    - Buena generalización, buen funcionamiento para datos no vistos durante el entrenamiento y con propiedades ligeramente distintas

    - Algunas inicializaciones son buenas para una rápida convergencia, pero malas para la generalización y otras al revés.

Implicaciones y orientaciones prácticas

- **Learning Rate:**
  - Valor fijo
  - Adaptable
  - Ajustado a cada conexión

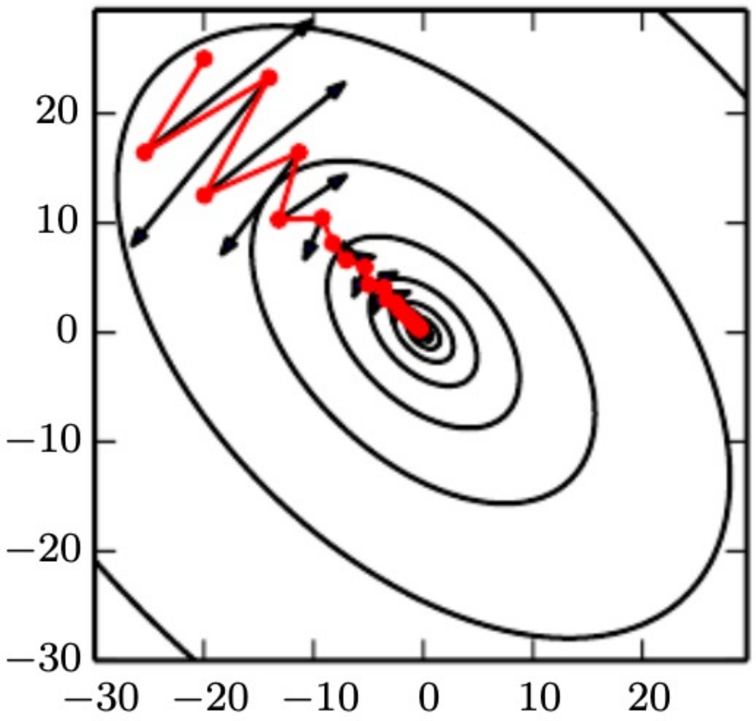


VIVOLAB

IA y Grandes Modelos de Lenguaje

Estrategias de Aceleración de Convergencia

- **Momentos:**
  - El gradiente modifica la “velocidad” de actualización de los pesos
  - Se le aporta “inercia” al proceso
  - Se acumula una media móvil de gradientes pasados para seguir moviéndose en esa dirección



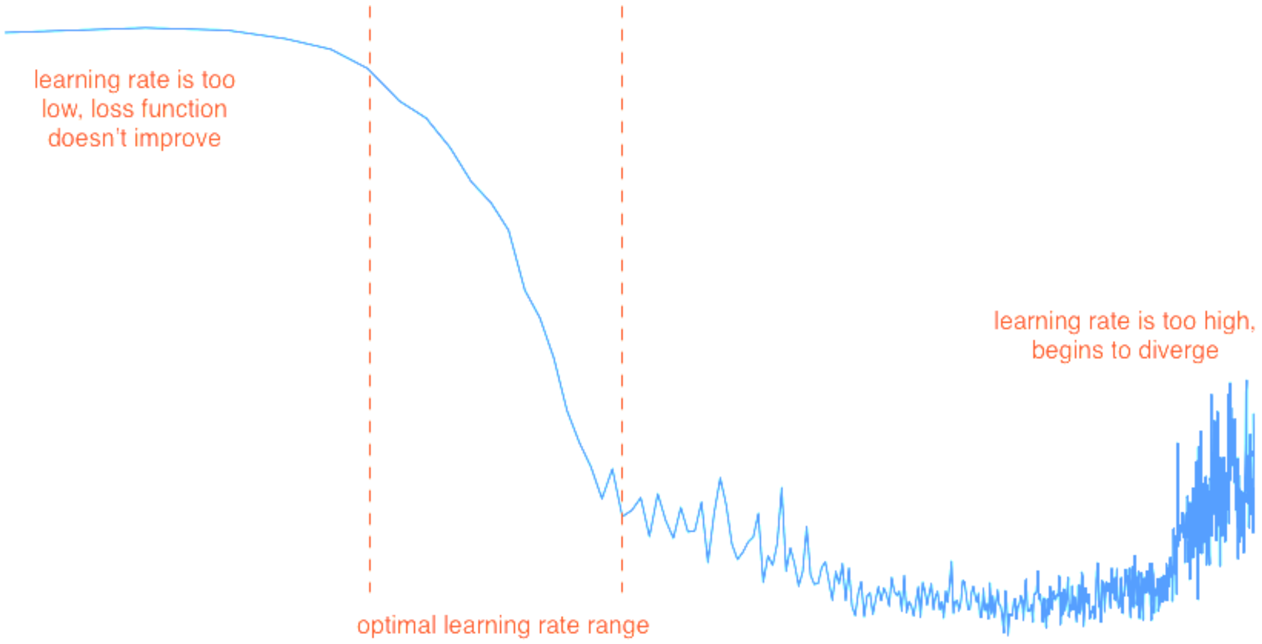
$$v_i(k) = \beta \cdot v_i(k-1) - \eta \frac{\partial \sum_{n \in k} \mathcal{J}(x_n, y_n, w^{(k)})}{\partial w_i}$$

$$w_i(k+1) = w_i(k) + v_i(k)$$

Universidad Zaragoza

Estrategias de Aceleración de Convergencia

- **Adaptive Learning Rate:**
  - Se reduce el learning rate cuando la reducción de la función de coste se ralentiza (plateau)
  - Se suele usar SGD con momentum y decaying learning rate (hasta un valor mínimo del learning rate, con caída exponencial, o dividiéndolo por un factor entre 2 y 10 cuando se detecta la meseta).



Estrategias de Aceleración de Convergencia

- **Adaptive Learning Rate:**
  - **AdaGrad**
    - Se acumulan los gradientes al cuadrado

$$\boldsymbol{r}(k) = \boldsymbol{r}(k-1) + \left(\frac{\boldsymbol{\partial} \sum_{n\in k} \boldsymbol{\mathcal{J}}(\boldsymbol{x_n}, \boldsymbol{y_n}, \boldsymbol{w}^{(k)})}{\boldsymbol{\partial w_i}}\right)^2$$

    - El learning rate se divide por la raíz cuadrada del acumulado de los gradientes al cuadrado

$$\eta(k) = \frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}(\boldsymbol{k})}}$$

Estrategias de Aceleración de Convergencia

- **Adaptive Learning Rate:**

  - **RMSProp**

  - Modificación sobre AdaGrad
  - Cambia la acumulación de gradientes al cuadrado por una media móvil (exponencial)

$$\boldsymbol{r}(k) = \boldsymbol{\rho} \cdot \boldsymbol{r}(k-1) + (1-\rho)\left(\frac{\boldsymbol{\partial} \sum_{\boldsymbol{n} \in \boldsymbol{k}} \boldsymbol{\mathcal{J}}(\boldsymbol{x_n}, \boldsymbol{y_n}, \boldsymbol{w}^{(\boldsymbol{k})})}{\boldsymbol{\partial w_i}}\right)^2$$

  - El learning rate se divide por la raíz cuadrada del acumulado de los gradientes al cuadrado

$$\eta(k) = \frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}(\boldsymbol{k})}}$$

Estrategias de Aceleración de Convergencia

- **Adaptive Learning Rate:**

  - **Adam**

    - Añade momento al RMSProp (Adam: “adaptive moments”)

$$\boldsymbol{s_i(k) = \rho_1 \cdot s_i(k-1) + (1-\rho_1)\frac{\partial \sum_{n\in k}\mathcal{J}(x_n, y_n, w^{(k)})}{\partial w_i}}$$

    - Acumulación de gradientes al cuadrado con una media móvil (exponencial) para adaptar Learning rate

$$\boldsymbol{r}(k) = \rho_2 \cdot \boldsymbol{r}(k-1) + (1-\rho_2)\left(\frac{\partial \sum_{n\in k}\mathcal{J}(x_n, y_n, w^{(k)})}{\partial w_i}\right)^2 \qquad \eta(k) = \frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}(\boldsymbol{k})}}$$

    - Combinados

$$\boldsymbol{w_i(k+1) = w_i(k) - \eta\,(k) s_i(k)}$$

Estrategias de Aceleración de Convergencia

- **Momentos**
- **Adaptive Learning Rate**
  - AdaGrad
  - RMSprop
  - Adam
  - …

- ¿Cuál usar?

No hay consenso claro y depende del problema y de los datos.

Implicaciones y orientaciones prácticas

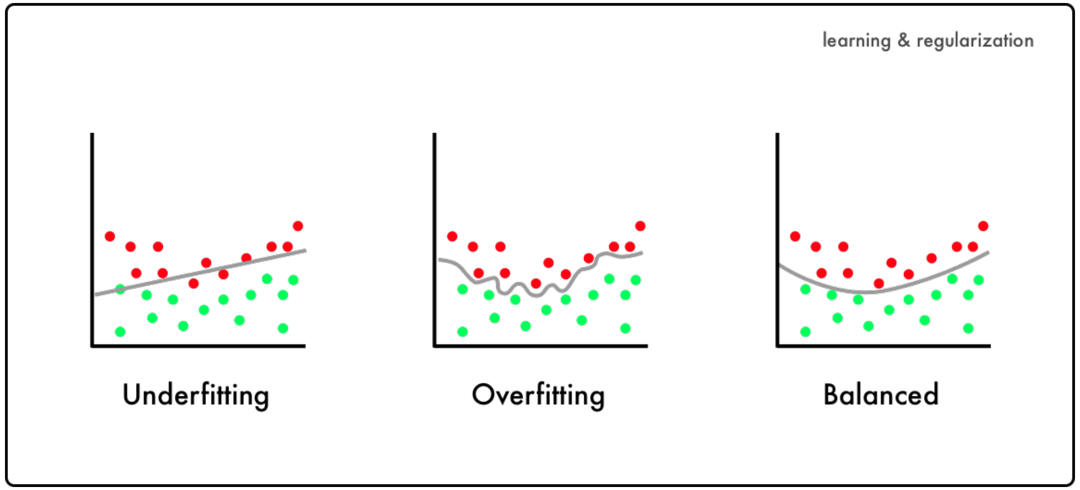
- **Batch Normalization:**
  - Estrategia para reparametrizar redes profundas
  - Los gradientes y actualizaciones en una capa dependen en gran medida del resto de las capas lo que hace difícil el ajuste de hiperparámetros como el learning rate.
  - Si $\boldsymbol{H}$, es un minibatch de activaciones de una capa, estas se normalizan para que tengan media nula y varianza unidad

$$H' = \frac{H - \mu}{\sigma} \qquad \mu = \frac{1}{m}\sum_i H_i \qquad \sigma = \sqrt{\delta + \frac{1}{m}\sum_i (H-\mu)_i^2}$$

  - En test $\boldsymbol{\mu}$ y $\boldsymbol{\sigma}$ pueden tomar valores aprendidos durante el entrenamiento

Implicaciones y orientaciones prácticas

- **Técnicas de Generalización y Regularización:**
  - **Generalización:**
    - Si la red está sobredimensionada (demasiados parámetros) se corre el riesgo de aprender las peculiaridades de los datos de entrenamiento:
      - Sobreajuste u overfitting.
    - Para evitar el sobreajute:
      - Contar con más datos:
        - Aumentar los dataset de train
        - **Data augmentation**
      - Limitar la capacidad de la red para no exceder la complejidad del problema que se aborda:
        - Limitar el número de capas
        - **Weight Sharing**: Varias unidades comparten sus pesos



Implicaciones y orientaciones prácticas

- **Técnicas de Generalización y Regularización** :
  - Estrategias:
    - **Early stopping**:
      - El entrenamiento se detiene antes de que la red llegue a overfitting
    - **Weight decay**:
      - Se intenta que los pesos no alcancen valores muy grandes (norma L2 ó norma L1)
    - **Ruido**:
      - Se añade un pequeño ruido en algunos puntos de la red durante el entrenamiento
    - **Dropout**:
      - Se desactivan algunas unidades durante el entrenamiento para que el resto se adapte en su ausencia.