

*Departamento de Física Médica - Centro atómico Bariloche - IB*

# Introducción a las Redes Neuronales

Ariel Hernán Curiale

[ariel.curiale@cab.cnea.gov.ar](mailto:ariel.curiale@cab.cnea.gov.ar)



**UNCUYO**  
UNIVERSIDAD  
NACIONAL DE CUYO



# Inteligencia Artificial

Introducción / historia

## Machine Learning

Conceptos  
Básicos

## Redes Neuronales

Deep Learning

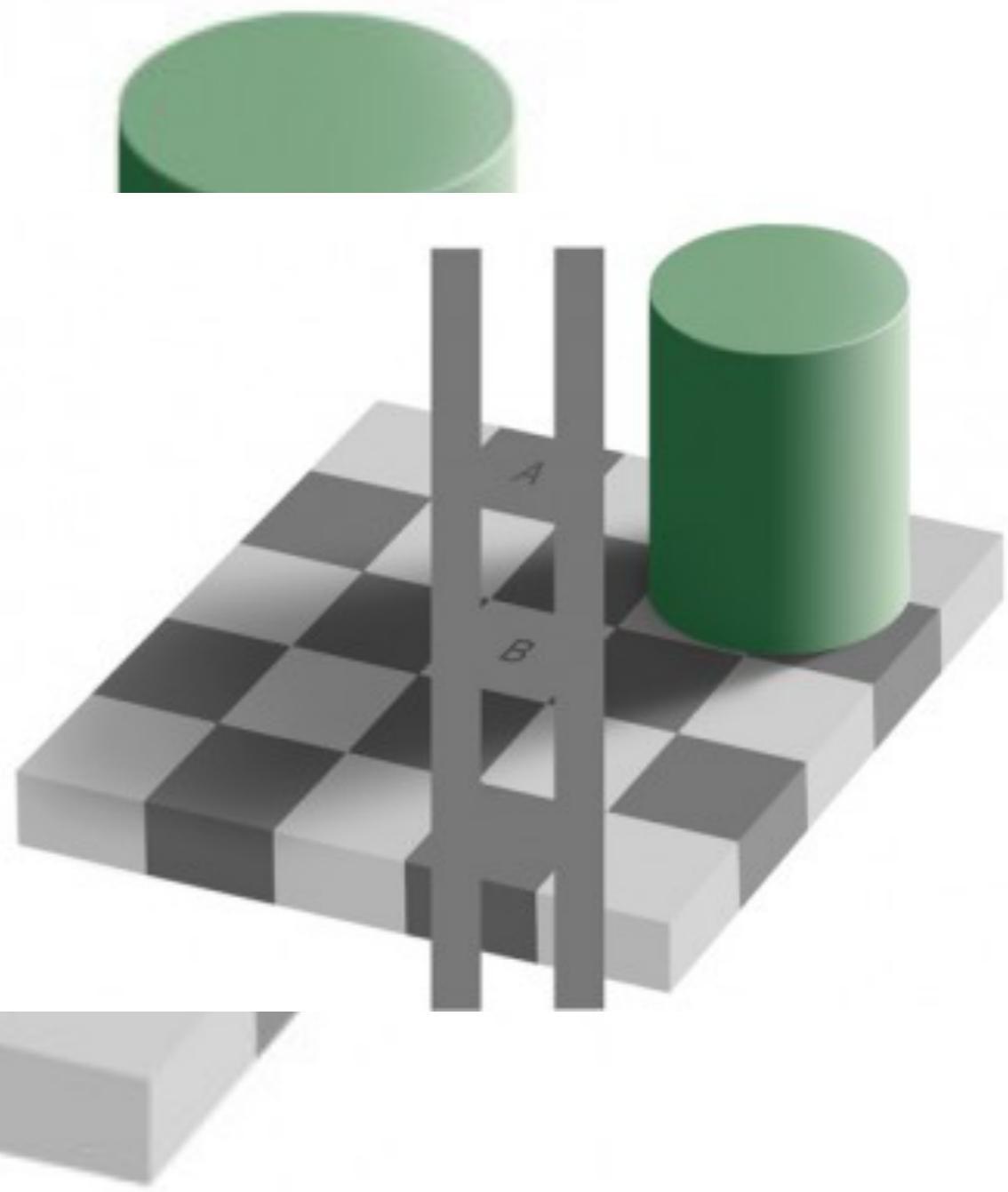
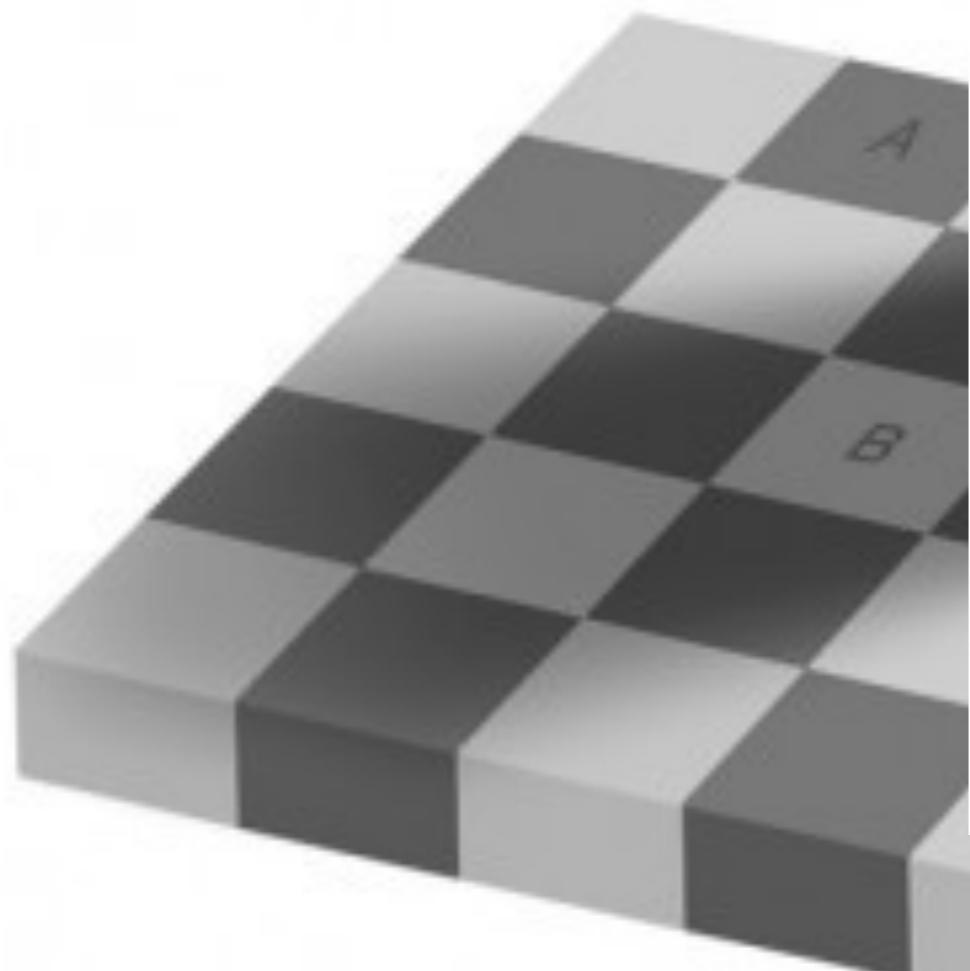
# ¿Inteligencia Artificial?

Dotar a las maquinas de cierta inteligencia para que a partir de la experiencia puedan tomar una acción respecto a una tarea en particular

# Humanos



# Maquinas

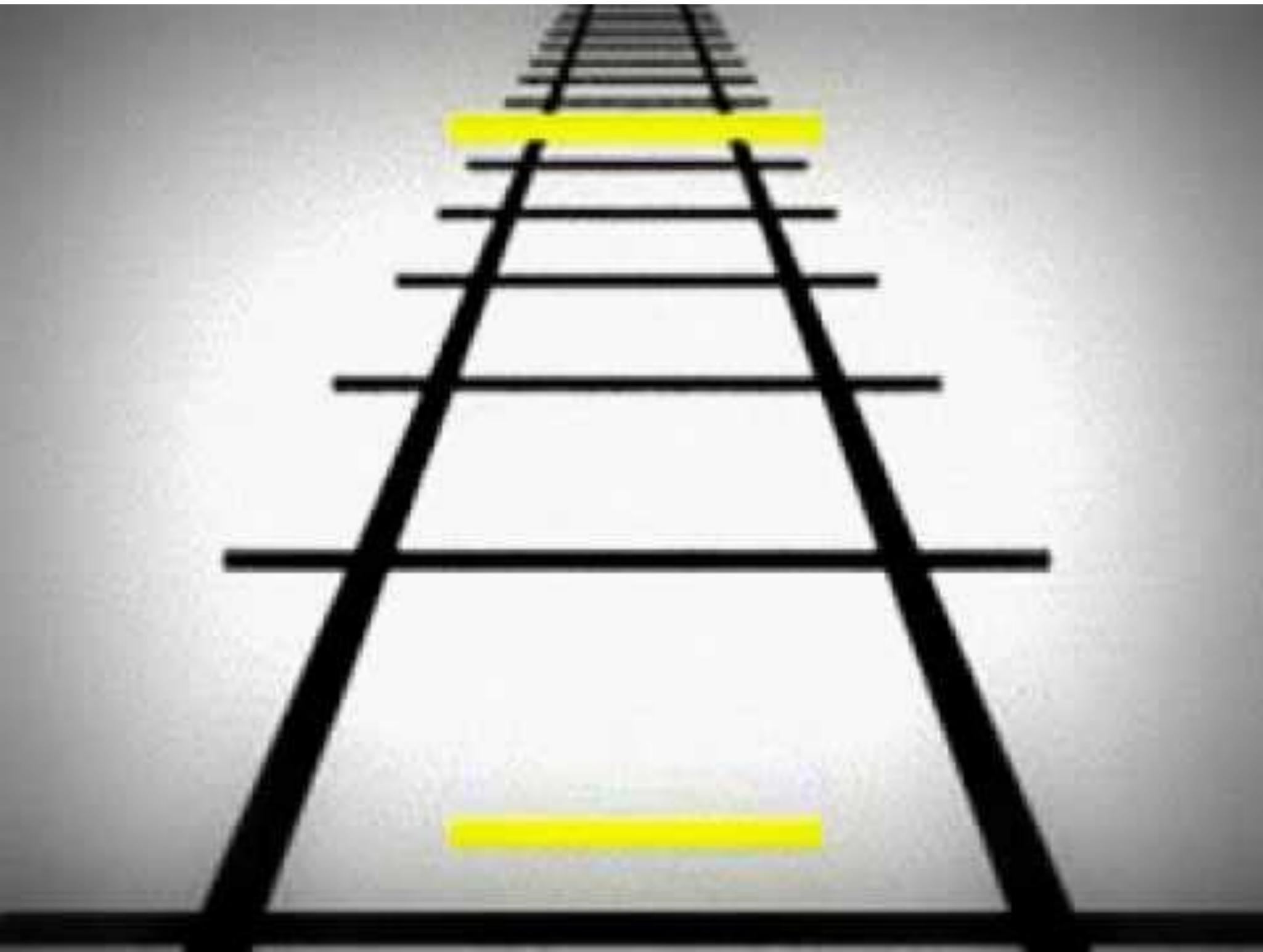


Edward H. Adelson

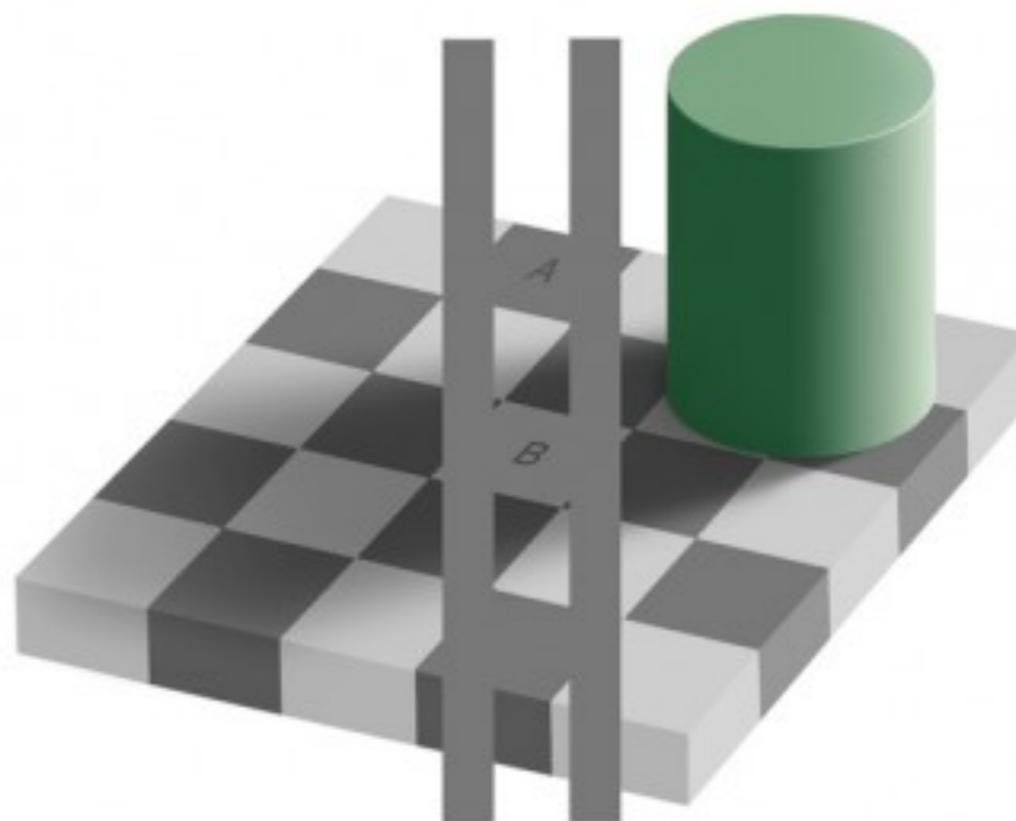
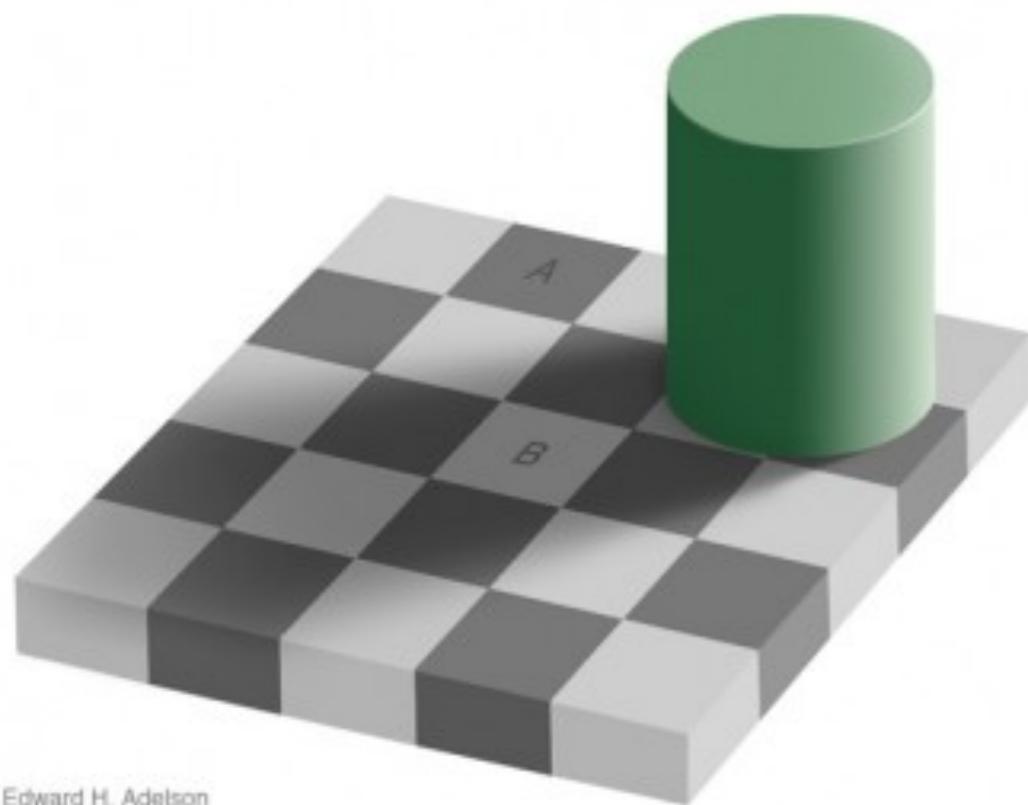
---

# Maquinas

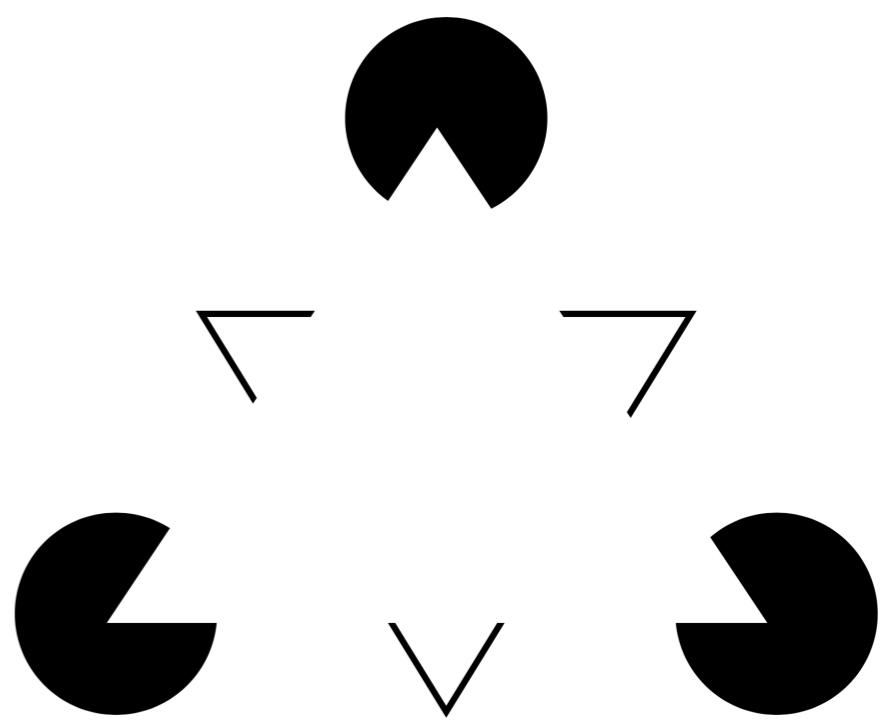
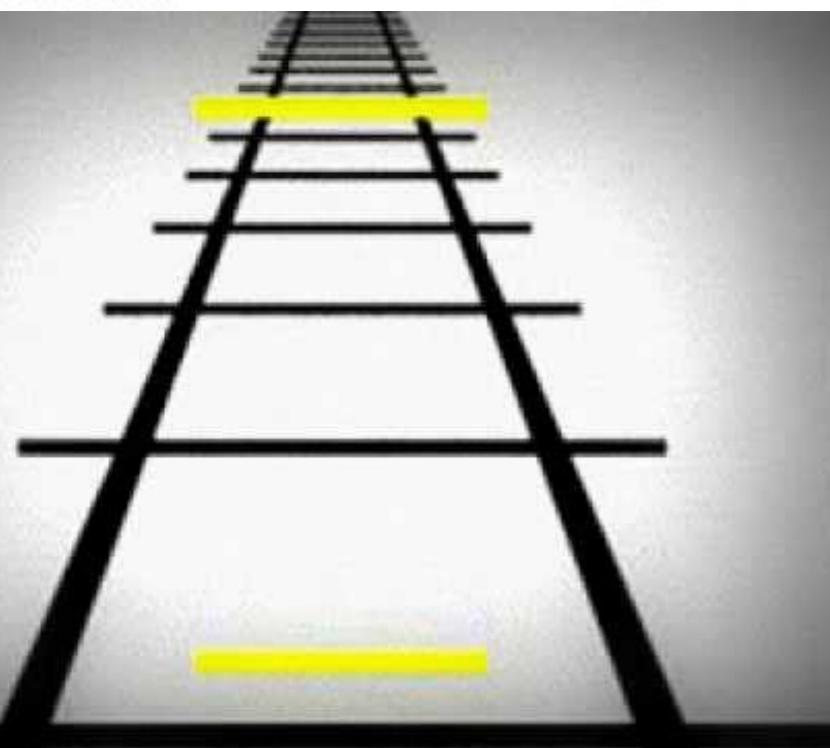
---



# Maquinas

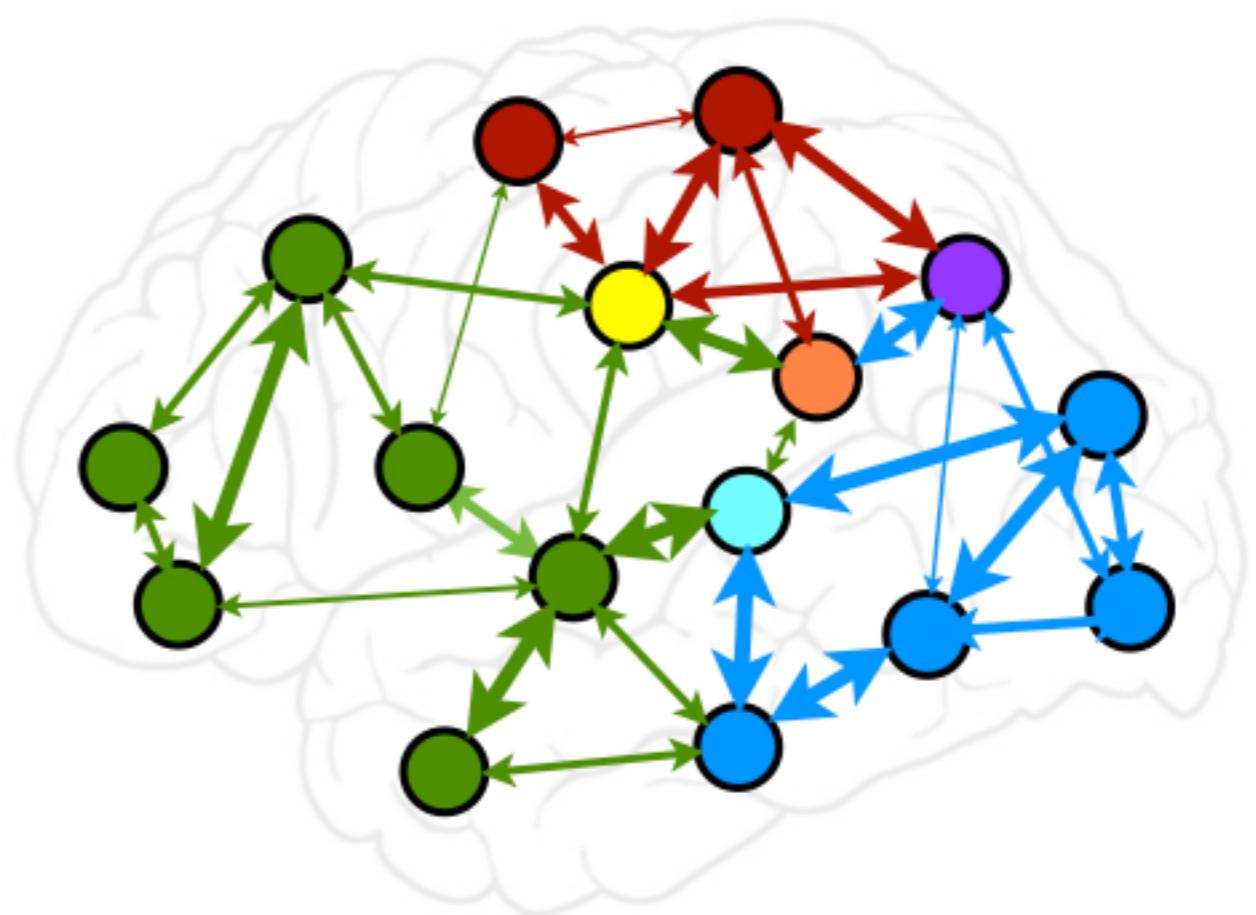


Edward H. Adelson

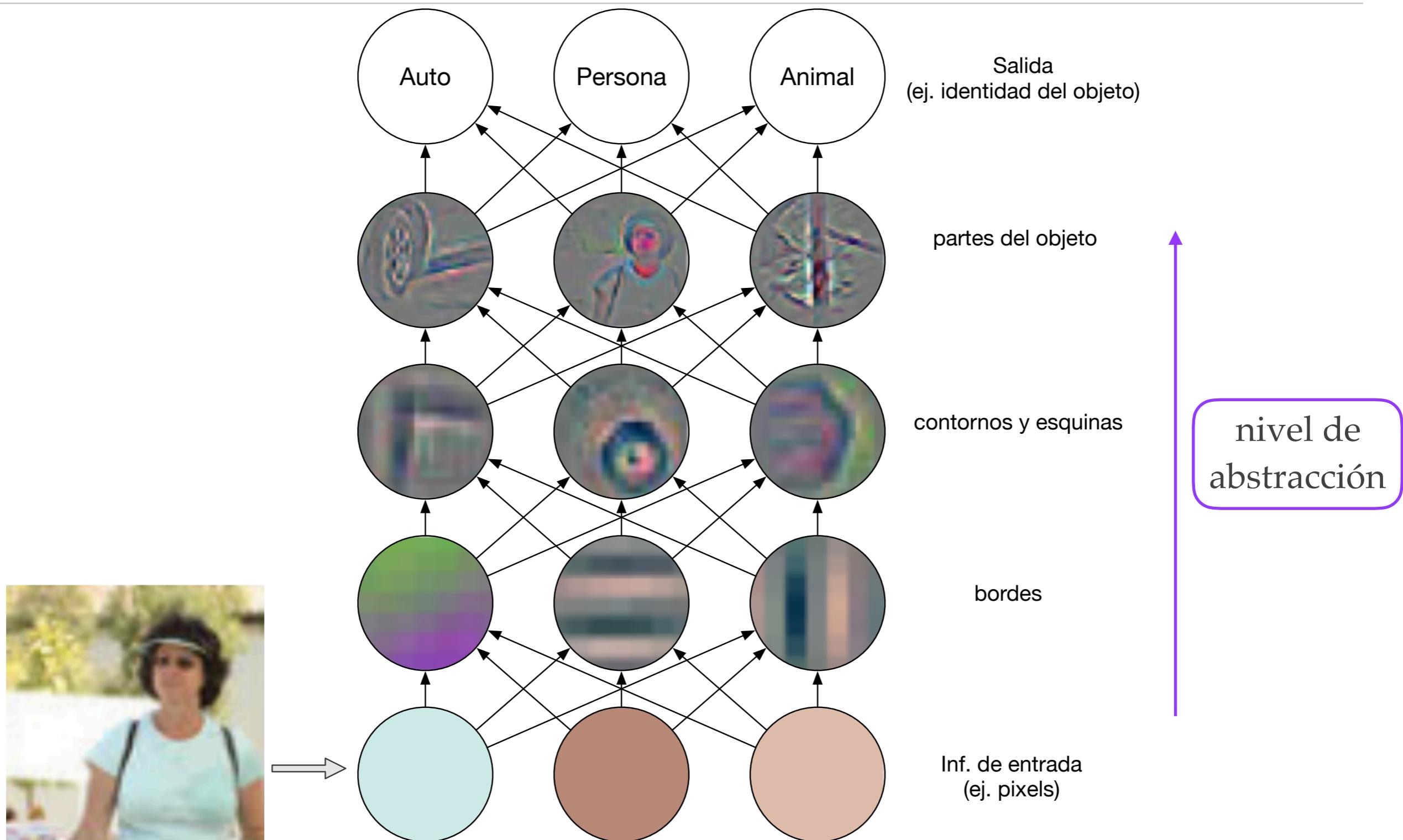


# Introducción

- ❖ Una estrategia simple es representar conceptos complicados a partir de otros más simples.

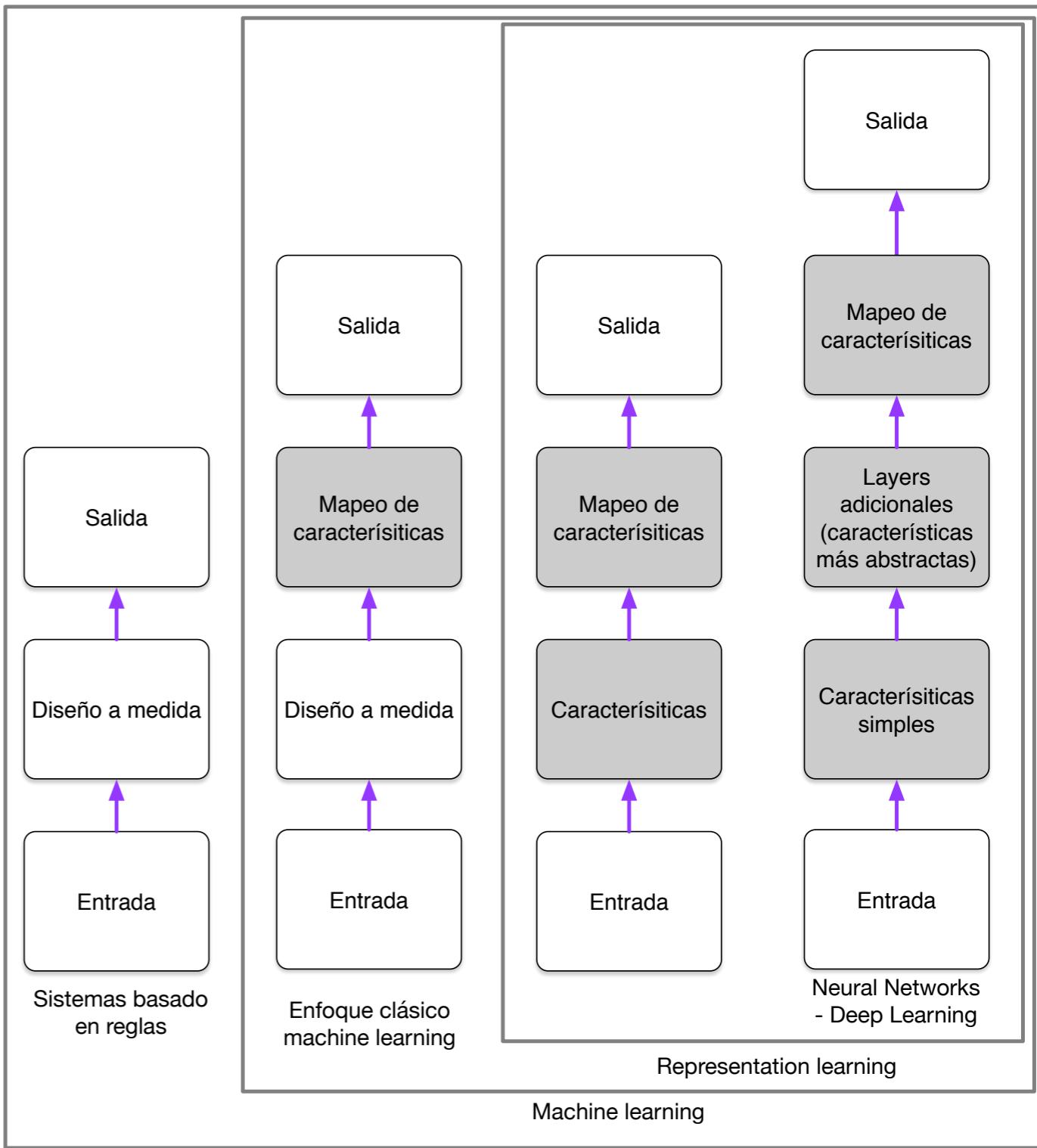
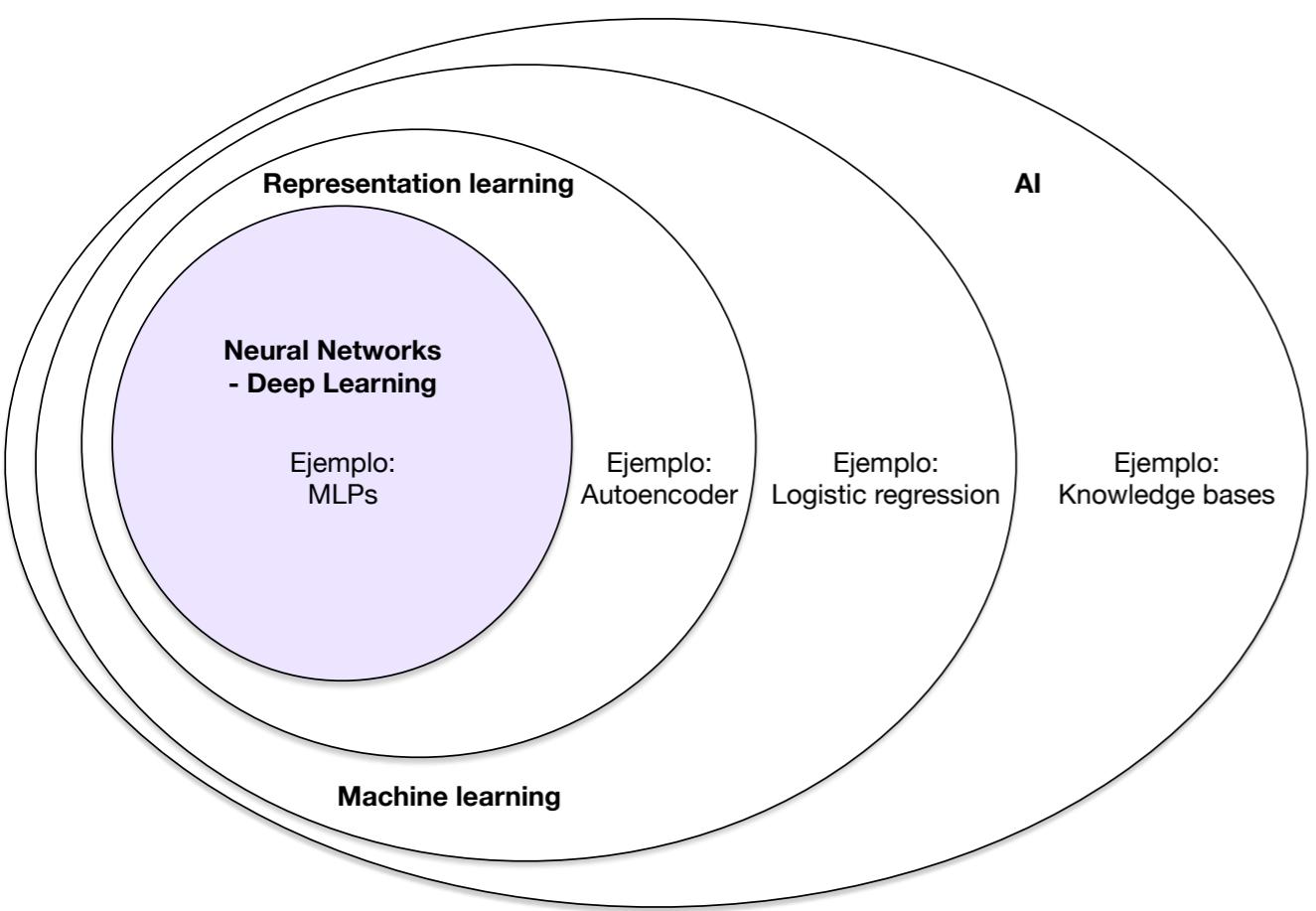


# Introducción



# Introducción

- ❖ Relación entre distintas técnicas de AI

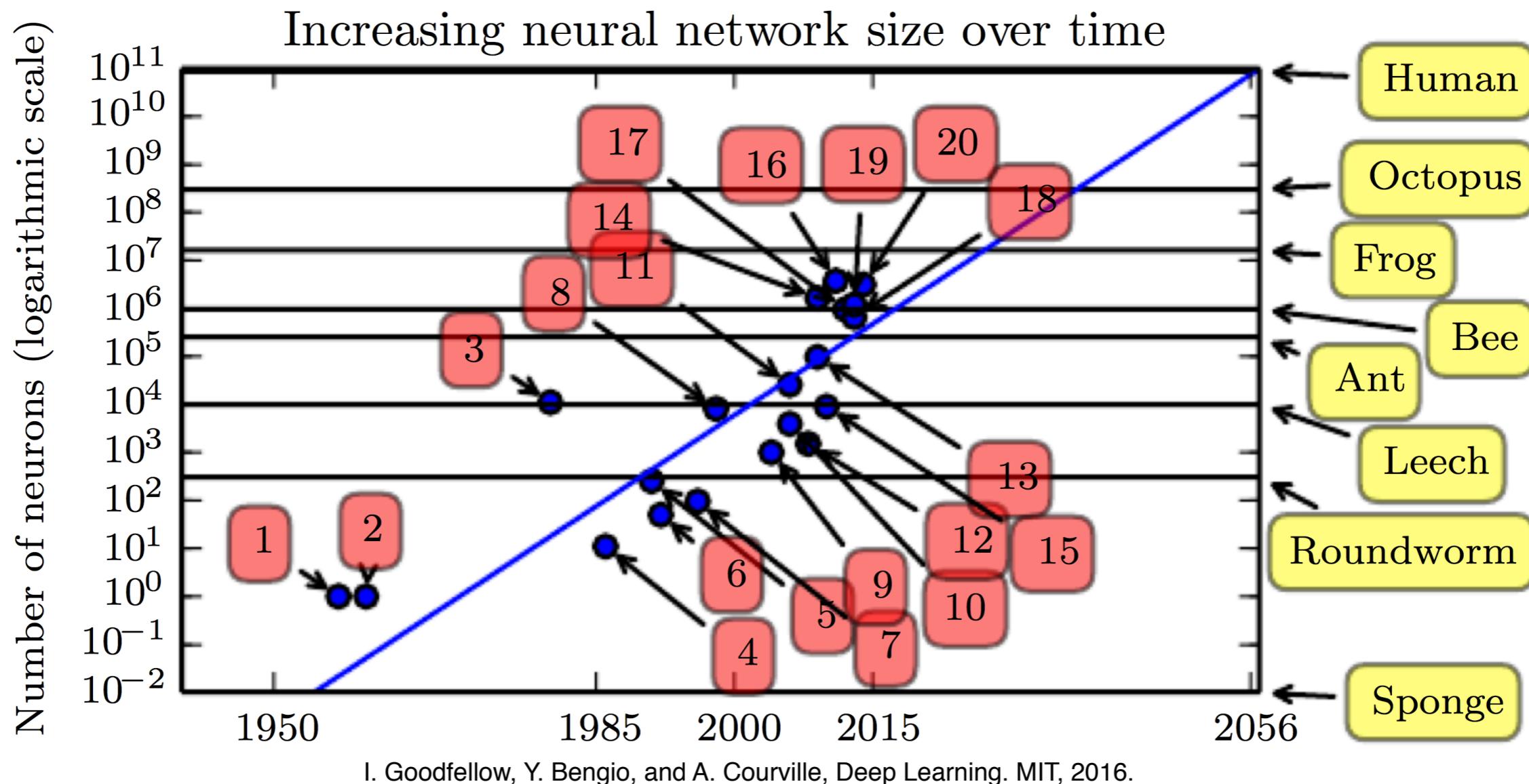


# Historia de las Redes Neuronales

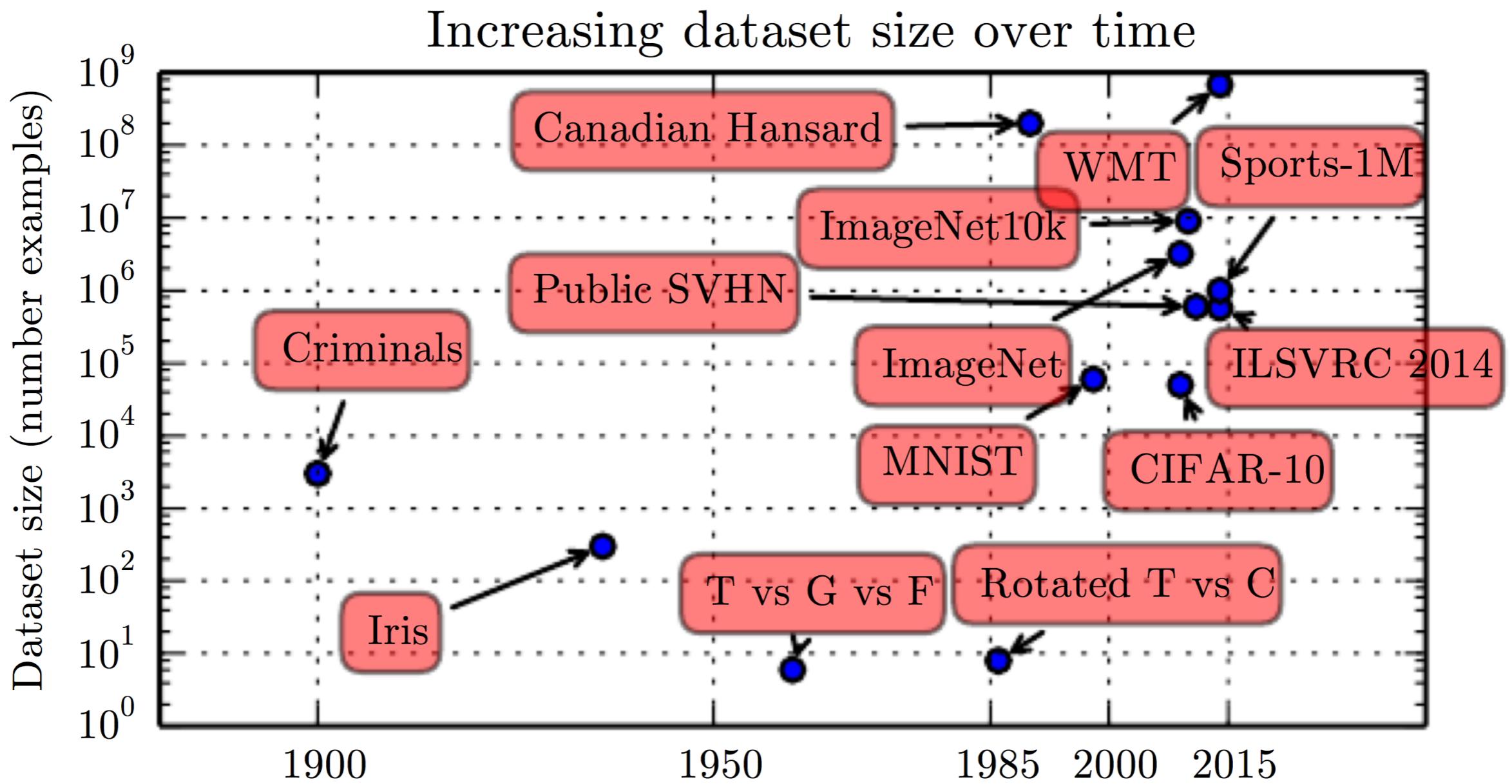
---

- ❖ *Cybernetics* [1940s - 1960s]
  - ❖ McCulloch-Pitts Neuron (1943)
  - ❖ Perceptrón (Rosenblatt, 1958, 1962)
- ❖ *Connectionism o parallel distributed processing* [1980s - 1990s]
  - ❖ Perceptrón multi-capas (Tesis de P. Werbos, 1974 )
  - ❖ Back-propagation (D.E. Rumelhart, G.E. Hinton et al., 1986; LeCun, 1985)
  - ❖ Convolutional neural network (LeCun et al., 1989)
- ❖ *Deep learning* [2000s - hoy].
  - ❖ GPU

# Historia



# Historia



# Machine learning

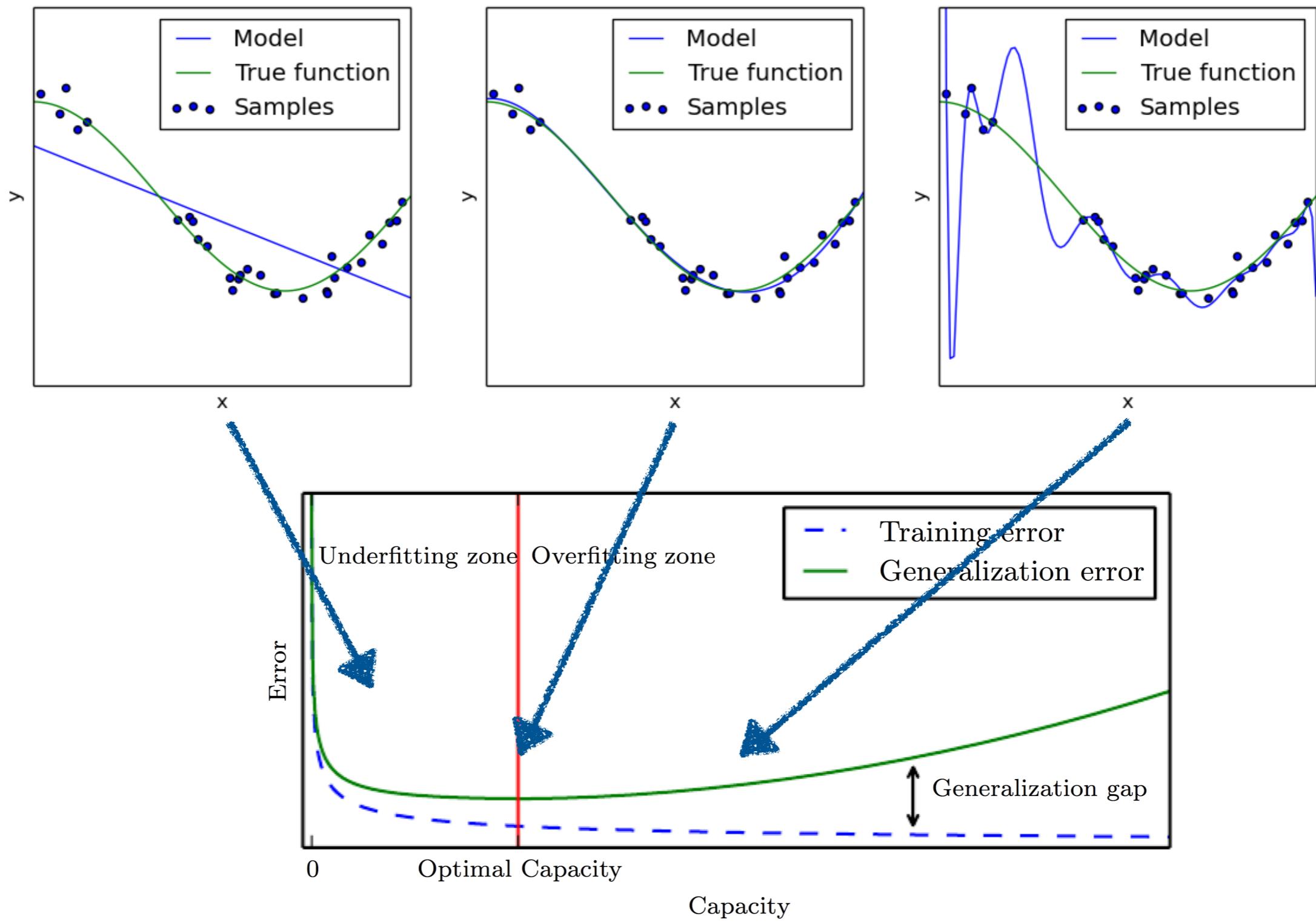
---

- ❖ Machine learning: métodos o algoritmos que **aprenden** una tarea a partir de experiencia (datos)
- ❖ Aplicaciones:
  - ❖ Regresión (aproximación de funciones) : se intenta predecir que valor le corresponde a una entrada.
  - ❖ Clasificación: con que categoría identificamos la entrada.
    - ❖ Ej. transcripción de texto o voz, detección de tejido, etc.
    - ❖ Clasificación frente a datos incompletos.
      - ❖ Reconstrucción de imágenes.
  - ❖ Agrupamiento (clustering)/categorización: Se conoce como clasificación si supervisión (K-means).
  - ❖ Eliminación de ruido (denoising).
  - ❖ ...

# Machine learning

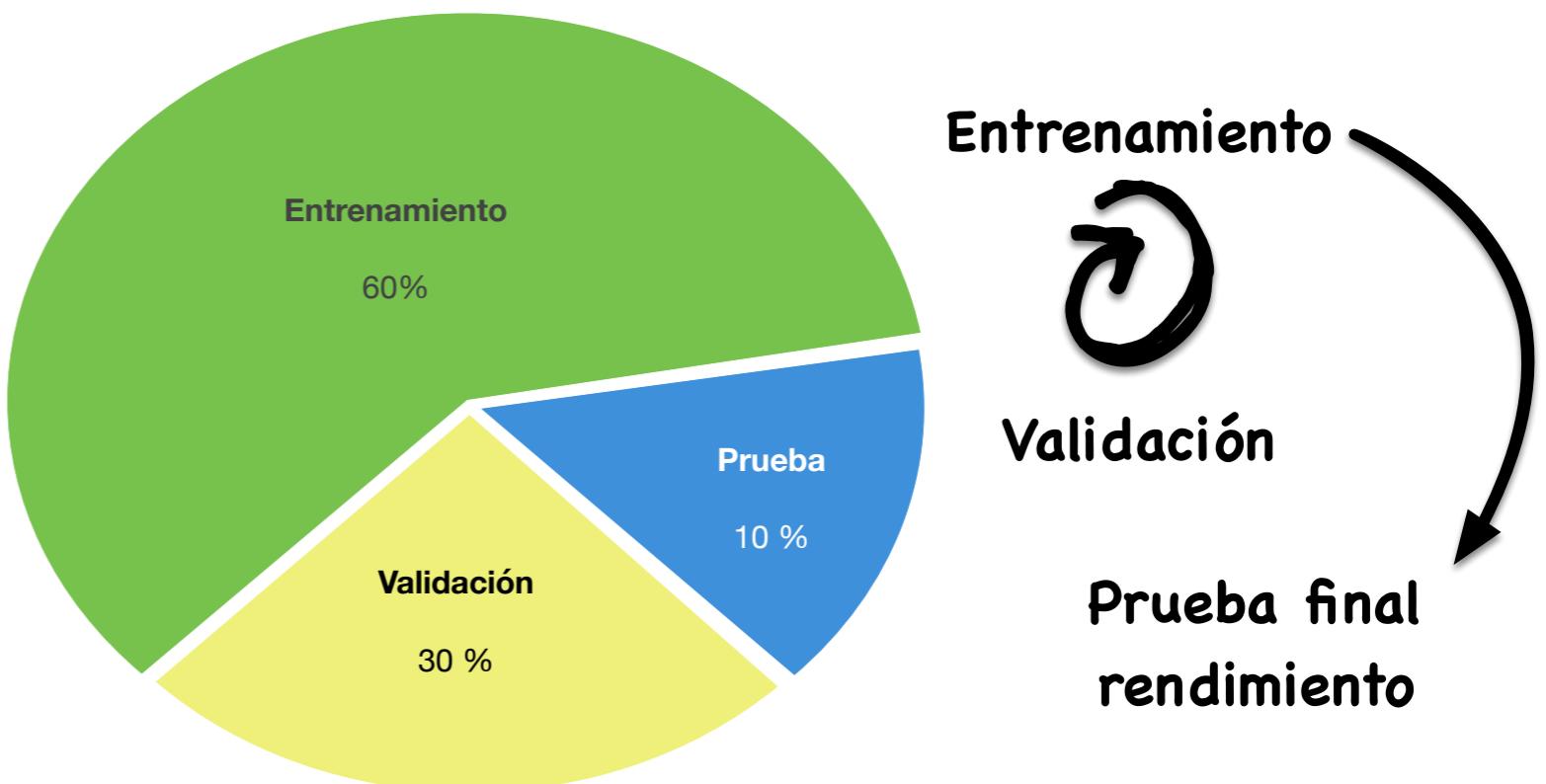


# Machine learning



# Machine learning

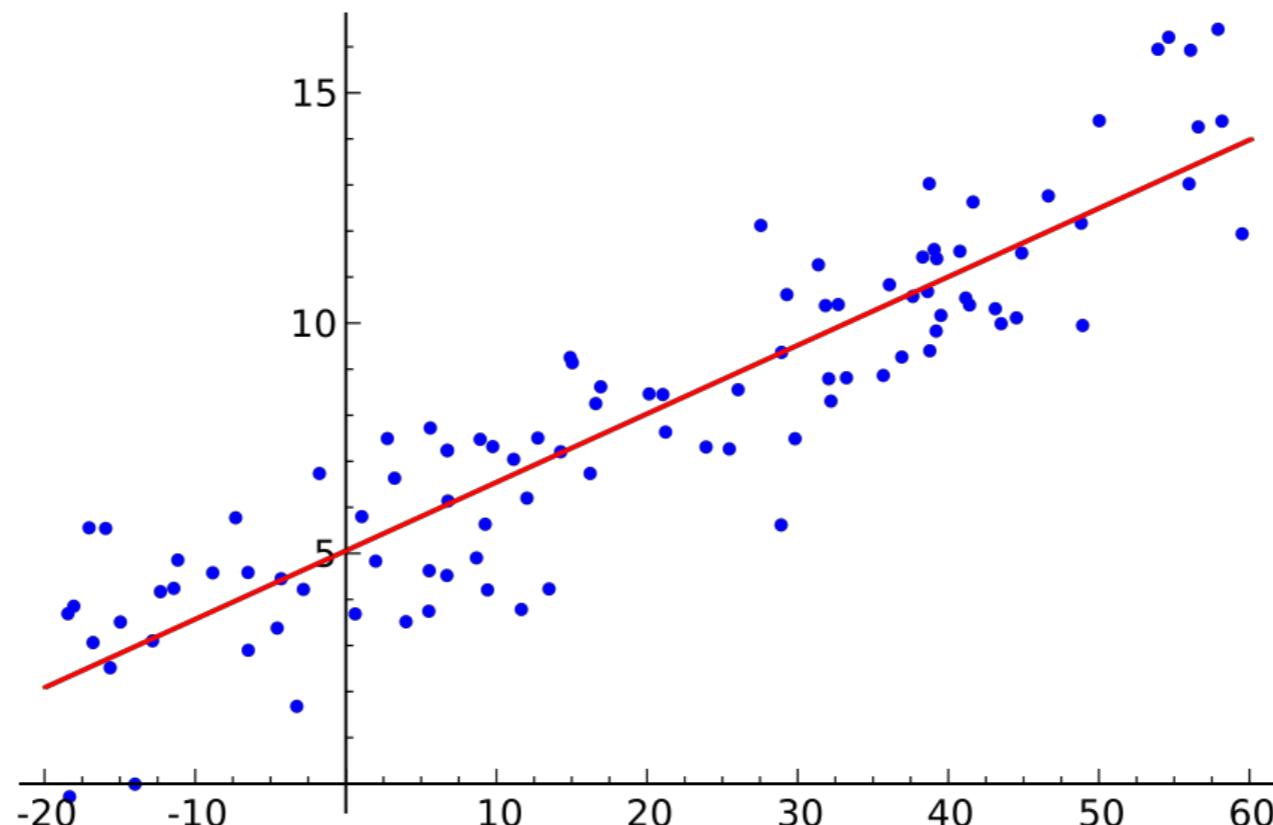
- ❖ Validación.
  - ❖ Los datos de prueba no se usan en el entrenamiento, de ninguna forma, ni siquiera para determinar los **hiperparámetros**.
  - ❖ Para evaluar los hiperparámetro usamos un subconjunto de los datos de entrenamiento que llamamos datos de validación.
  - ❖ Validación cruzada.



# Regresión Lineal (supervisado)

Objetivo: predecir el valor  $y \in \mathbb{R}$  dado un conjunto de características  $\mathbf{x} \in \mathbb{R}^n$  mediante una relación lineal:

$$\hat{y} = \mathbf{x}^T \mathbf{w}$$



# Regresión Lineal

Objetivo: predecir el valor  $y \in \mathbb{R}$  dado un conjunto de características  $\mathbf{x} \in \mathbb{R}^n$  mediante una relación lineal:

$$\hat{y} = \mathbf{x}^T \mathbf{w}$$

- ❖ Evaluamos el método con el error cuadrático medio.

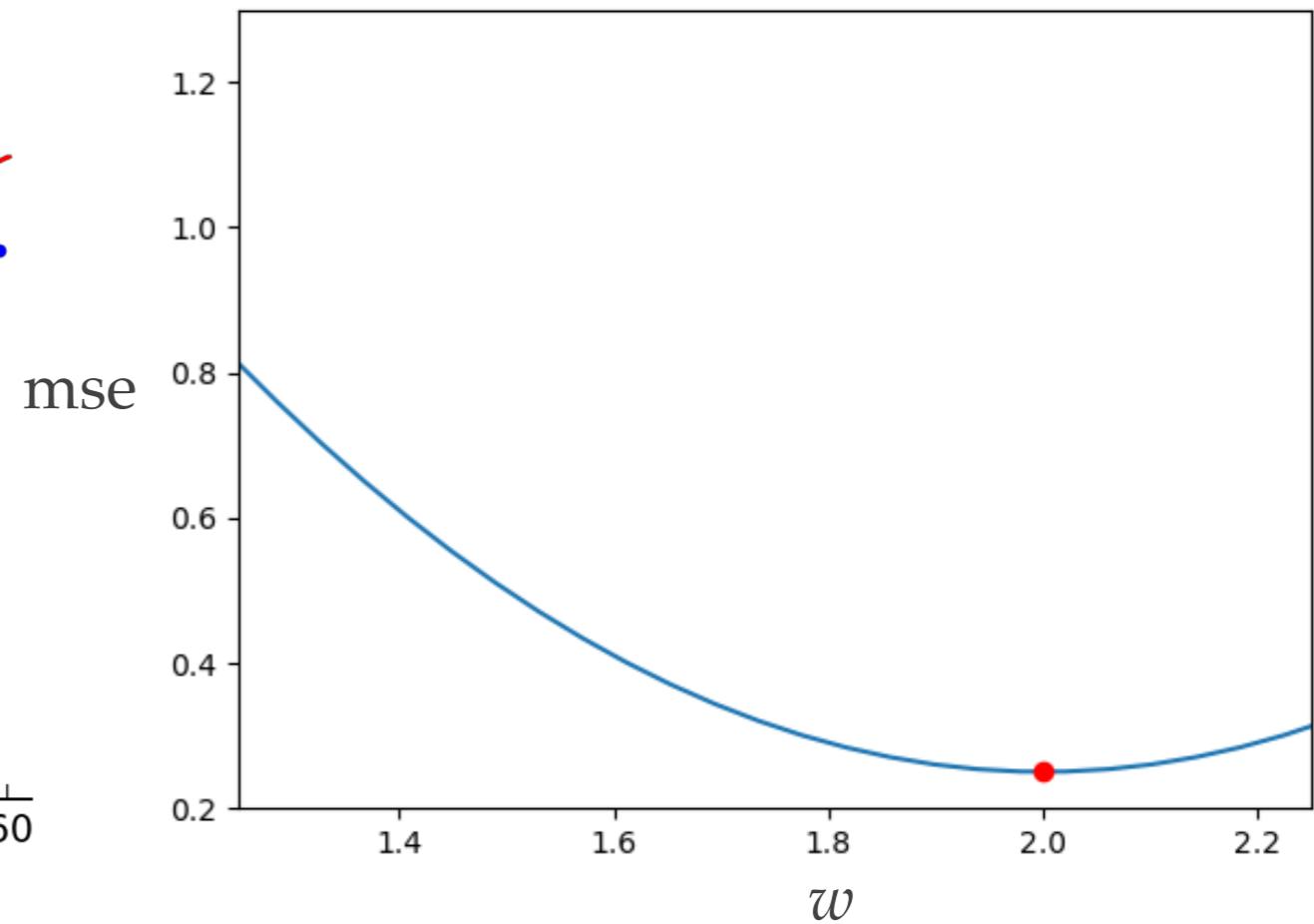
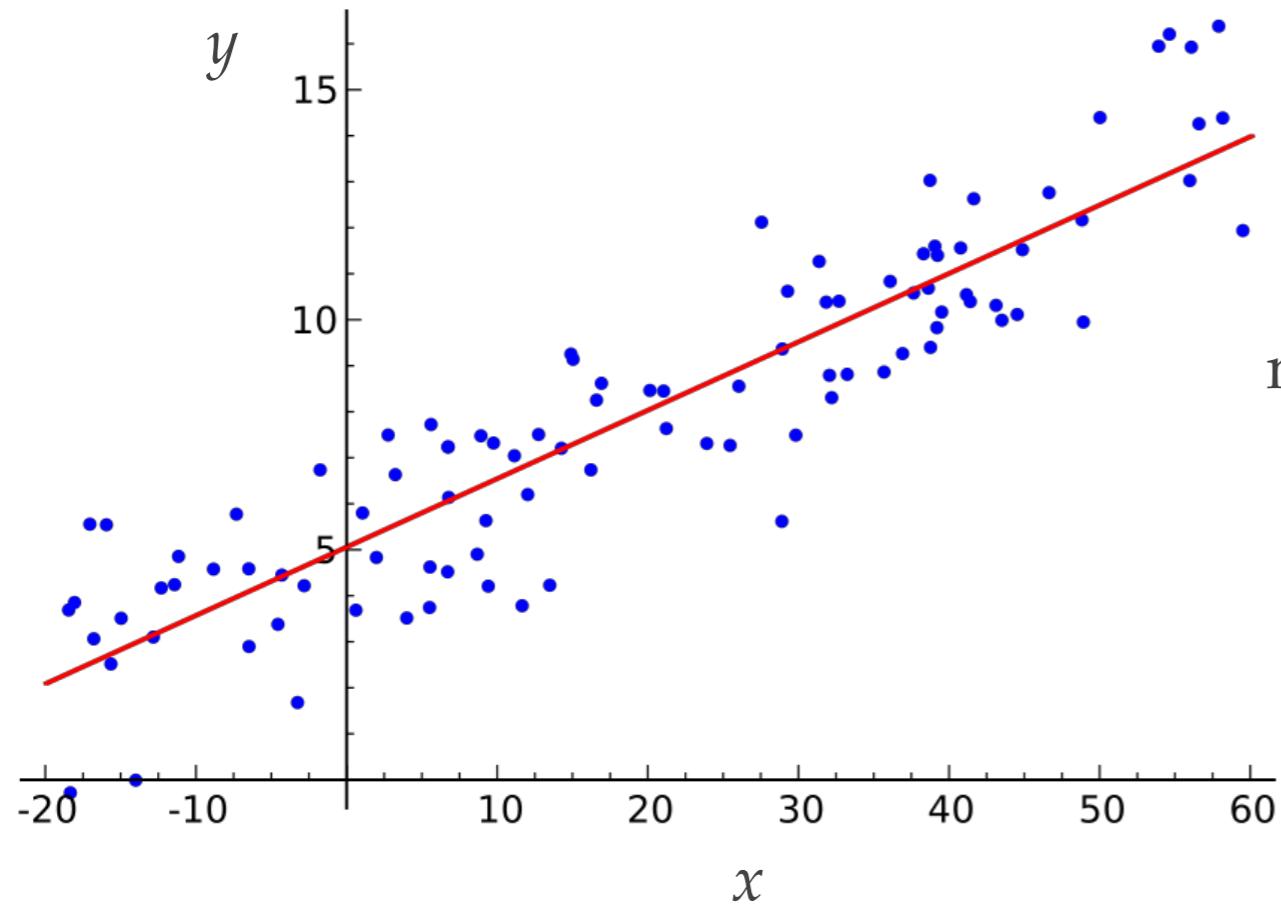
$$E = \frac{1}{m} \sum_i^m (\hat{y}_i - y_i)^2 = \frac{1}{m} \|\hat{\mathbf{y}} - \mathbf{y}\|^2$$

- ❖ Buscamos encontrar los pesos que minimicen dicho error:  $\nabla E = 0$

$$\begin{aligned} \nabla E &= \frac{1}{m} \nabla_w (\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2) = \frac{1}{m} \nabla_w ((\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y})) = 0 \\ &\approx 2\mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{X}^T \mathbf{y} = 0 \end{aligned}$$

$$\boxed{\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}} \quad \xrightarrow{\text{bias}} \quad \hat{y} = \mathbf{w}^T \mathbf{x} + b$$

# Regresión Lineal

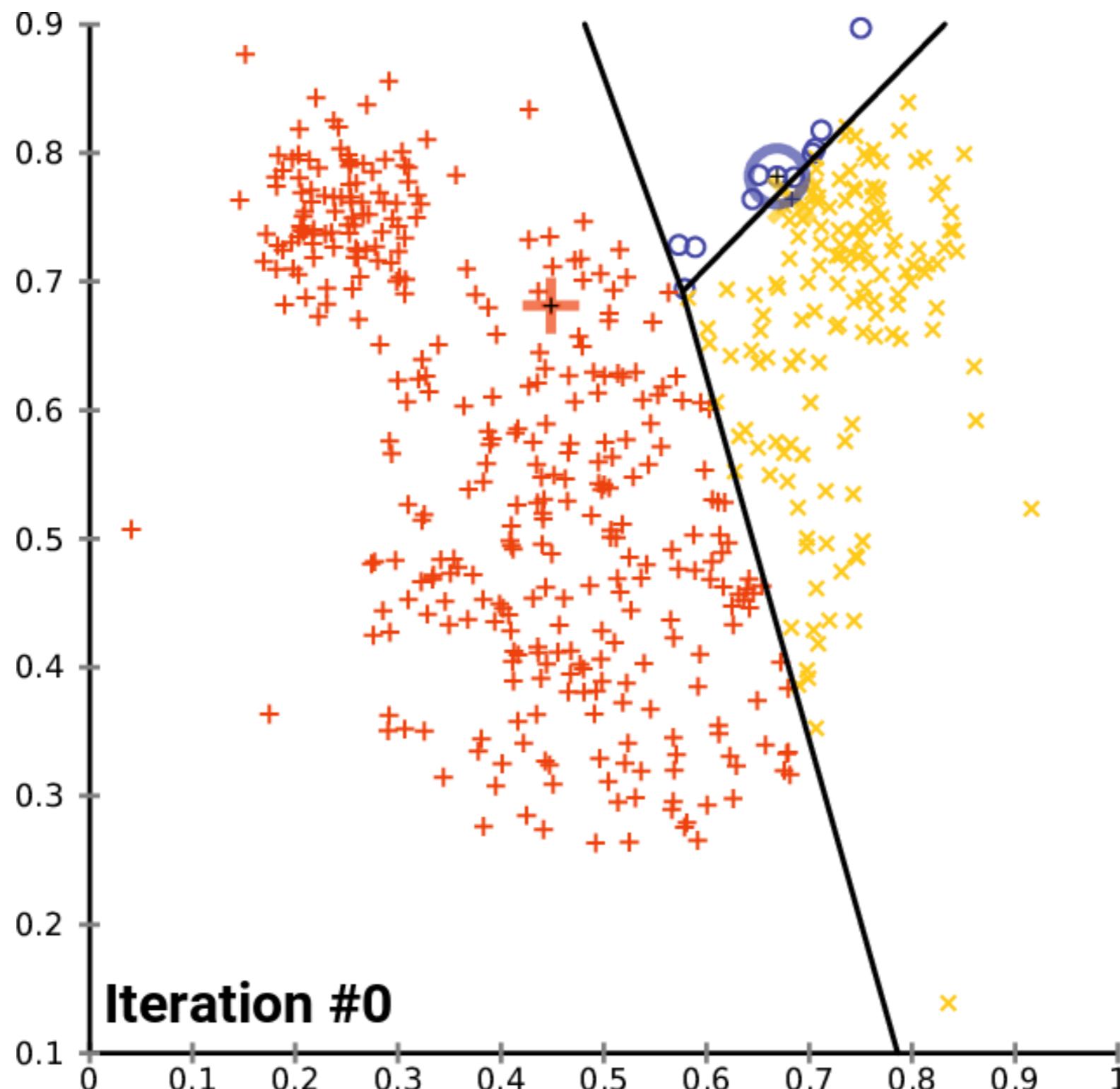


# K-means (no supervisado)

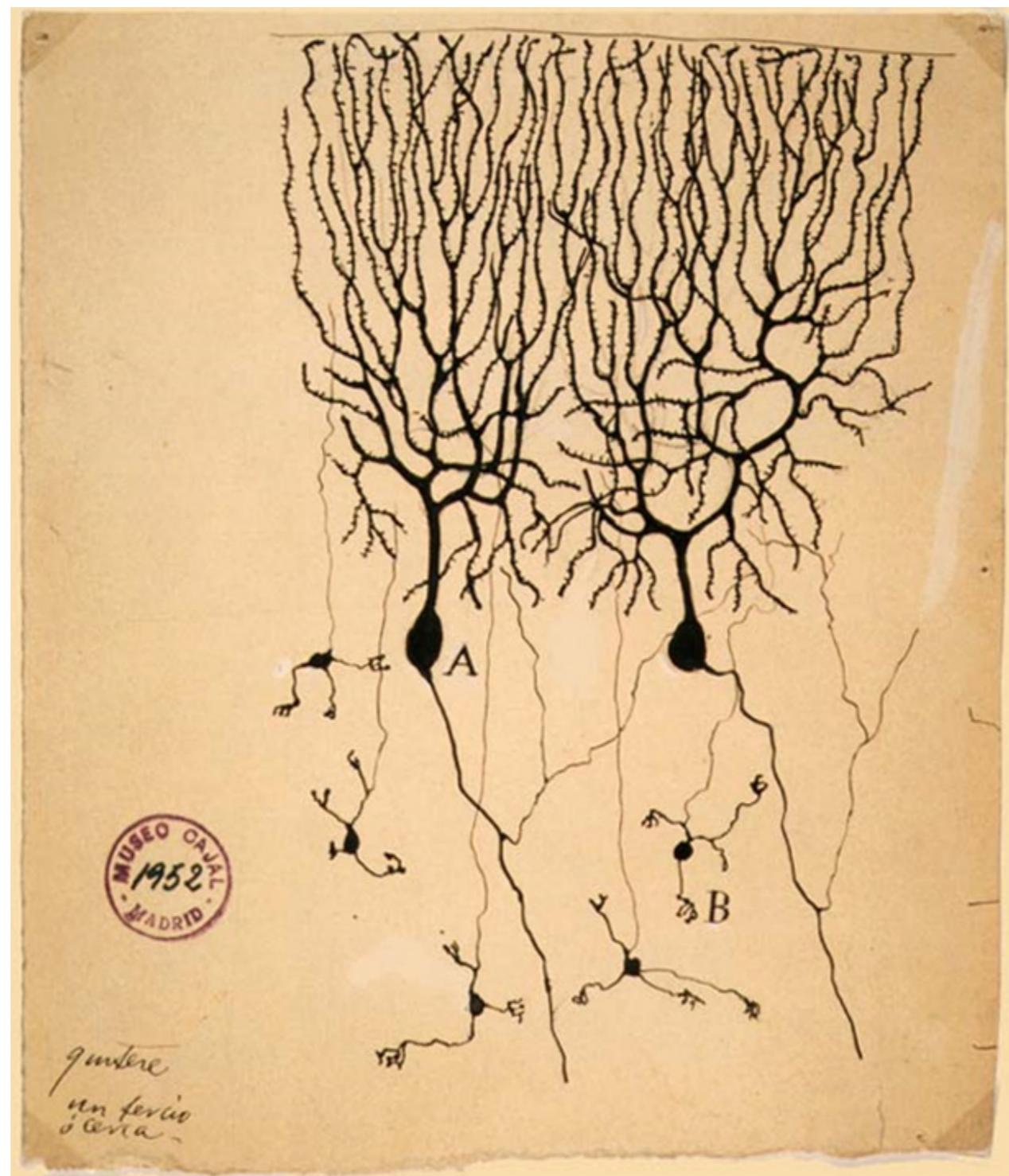
- ❖ Agrupamos  $n$  muestras en  $k$  clases o agrupaciones (clusters).
  - ❖ Para ello minimizamos la suma de distancias intra-clase al cuadrado:
$$\min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \mu_i\|^2$$
- ❖ Método:
  - ❖ Partimos de un conjunto de medias  $\mu_1, \dots, \mu_k$ .
  - ❖ Asignamos cada muestra a una agrupación (cluster) que tiene la media más cercana.
  - ❖ Se vuelve a calcular la media para cada agrupación (cluster).
  - ❖ Iteramos.
- ❖ Problema: determinar el hiperparámetro  $K$ .

# Machine learning

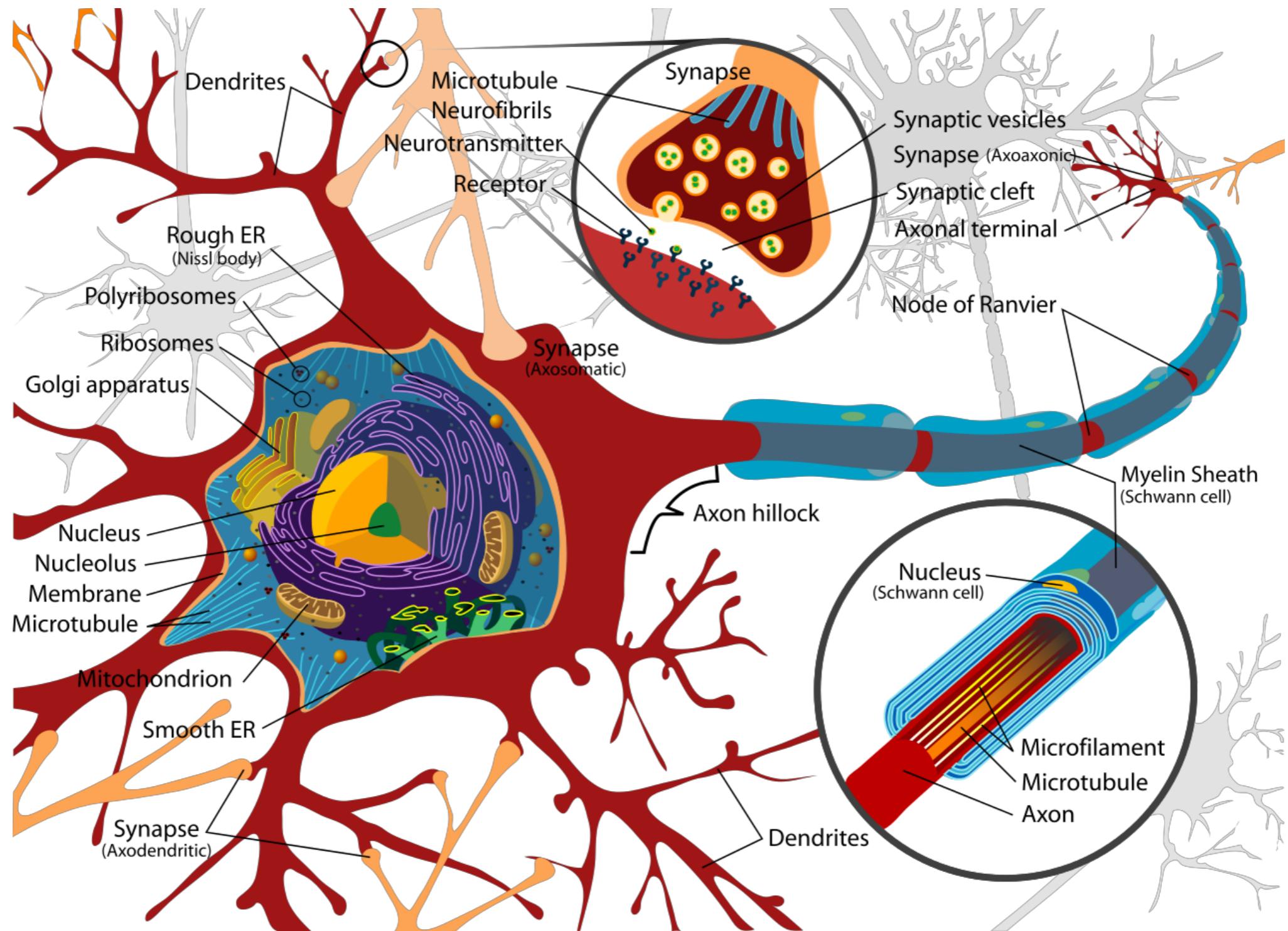
## ❖ K-means



# Redes Neuronales

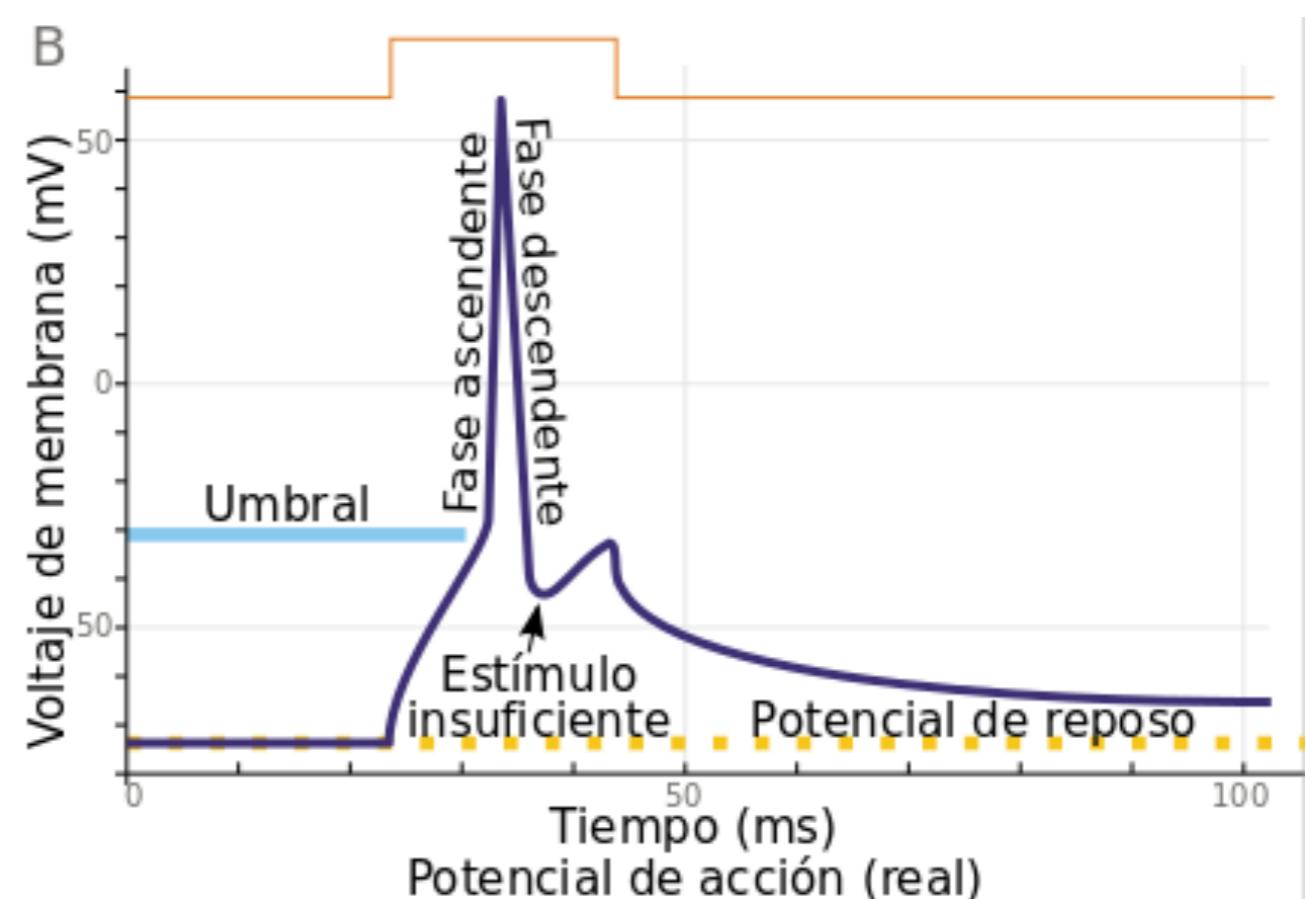
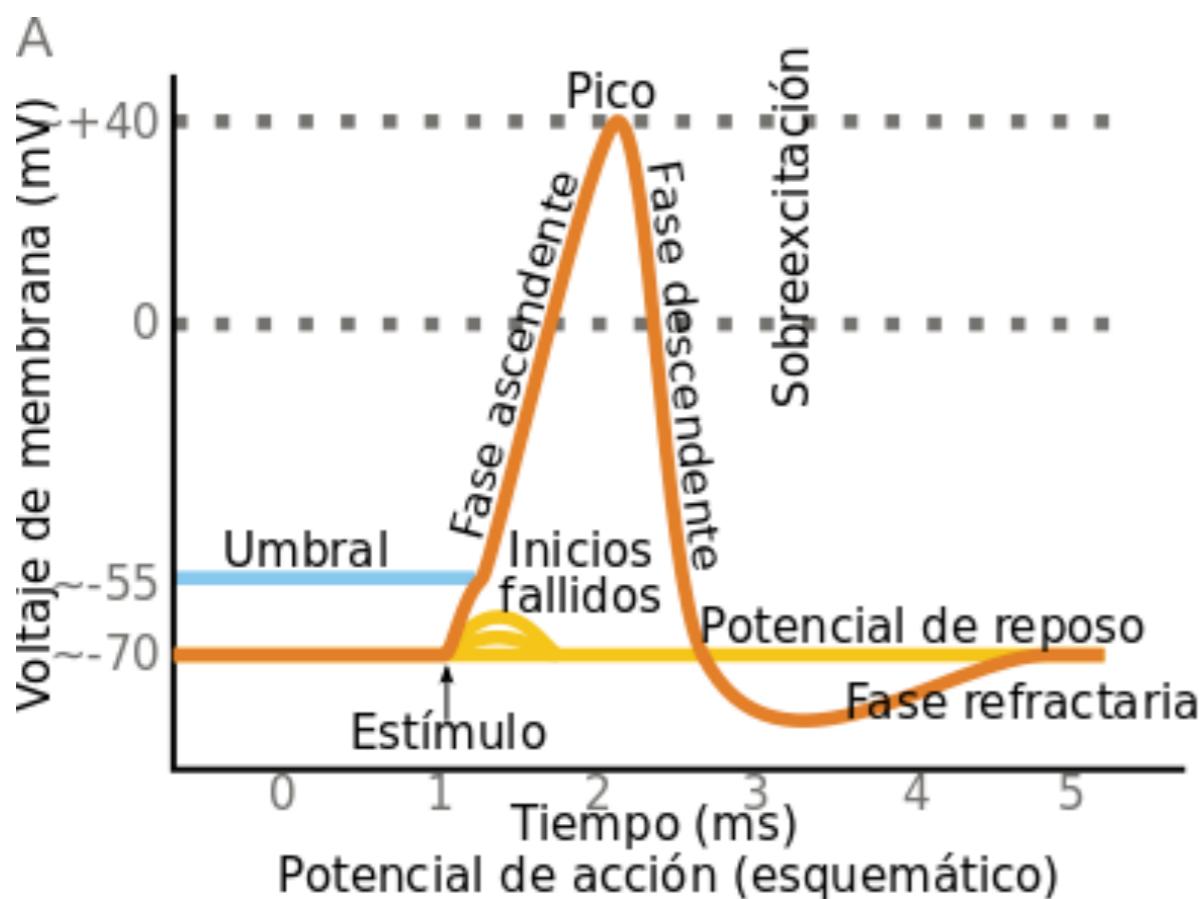


# Redes Neuronales



# Redes Neuronales

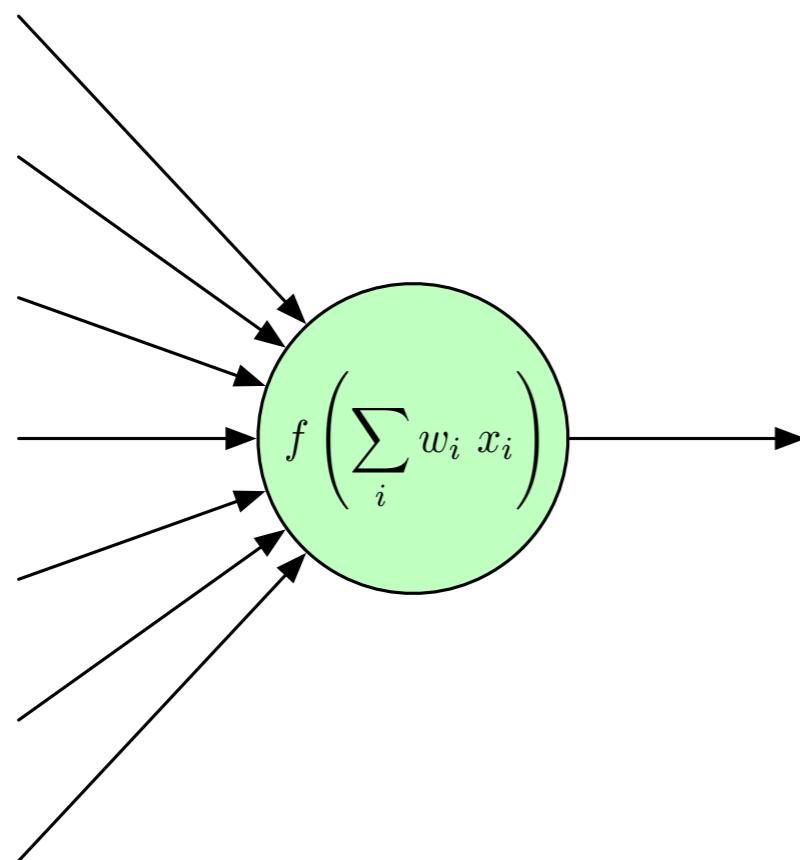
Si el estímulo es mayor que cierto umbral se produce un fenómeno altamente no lineal que se propaga por el axón



Una vez que la perturbación llega a las sinapsis se produce una liberación de neurotransmisores

# Redes Neuronales

- ❖ McCulloch-Pitts Neuron (1943)

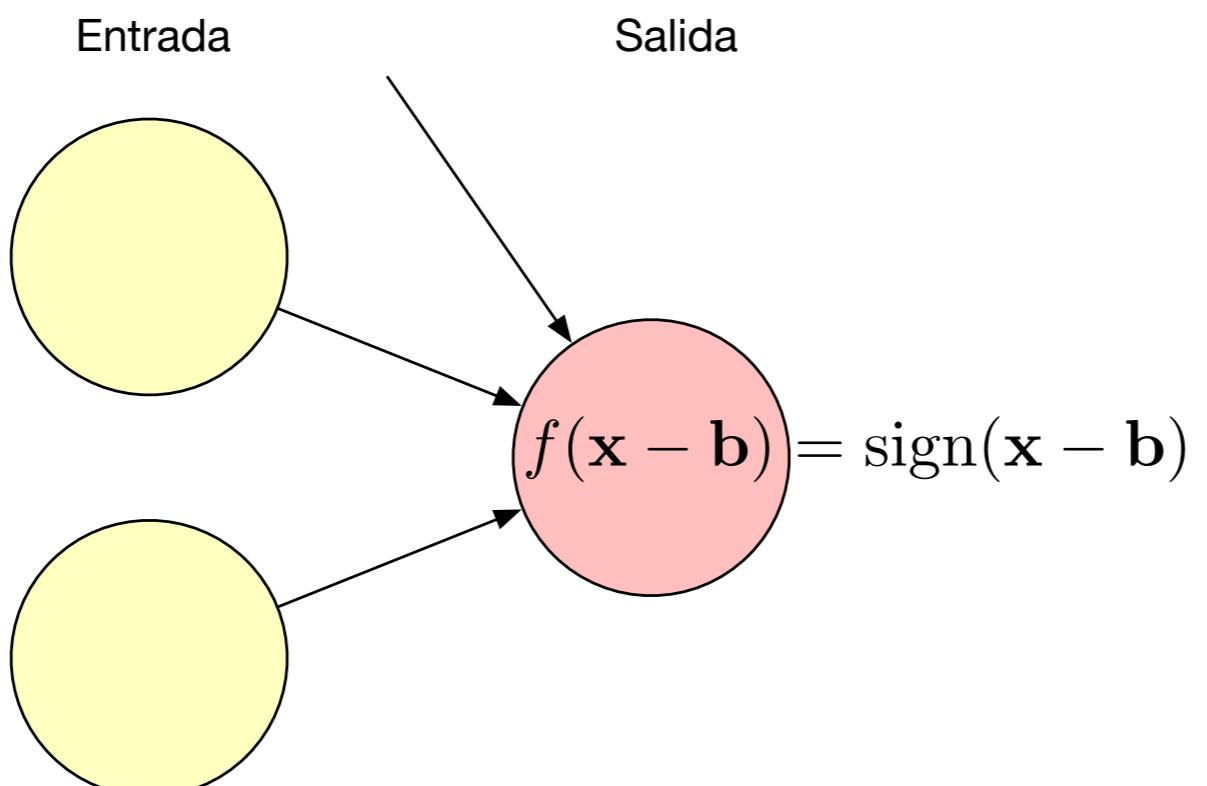


$$f(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} > \text{umbral} \\ 0 & \mathbf{x} \leq \text{umbral} \end{cases}$$

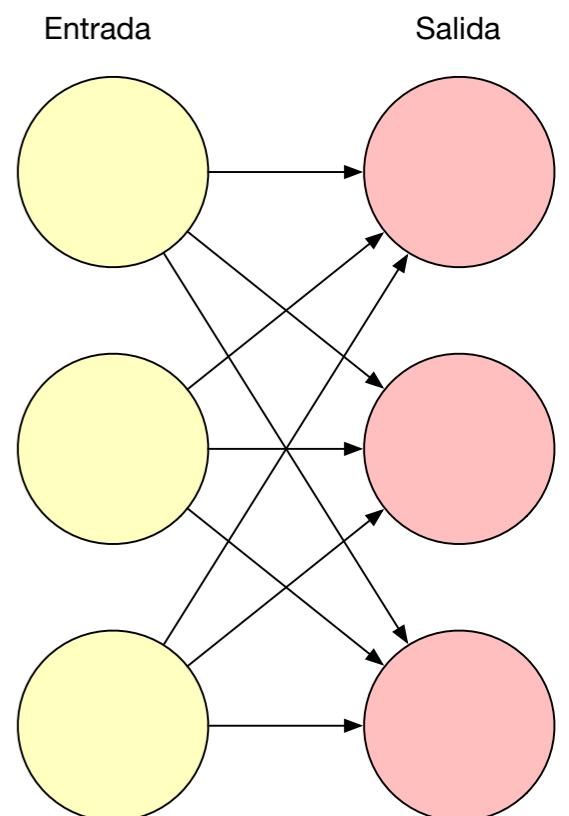
$$f(\mathbf{x} - \mathbf{b}) = \text{sign}(\mathbf{x} - \mathbf{b})$$

# Redes Neuronales

- ❖ Perceptrón (Rosenblatt, 1958, 1962)

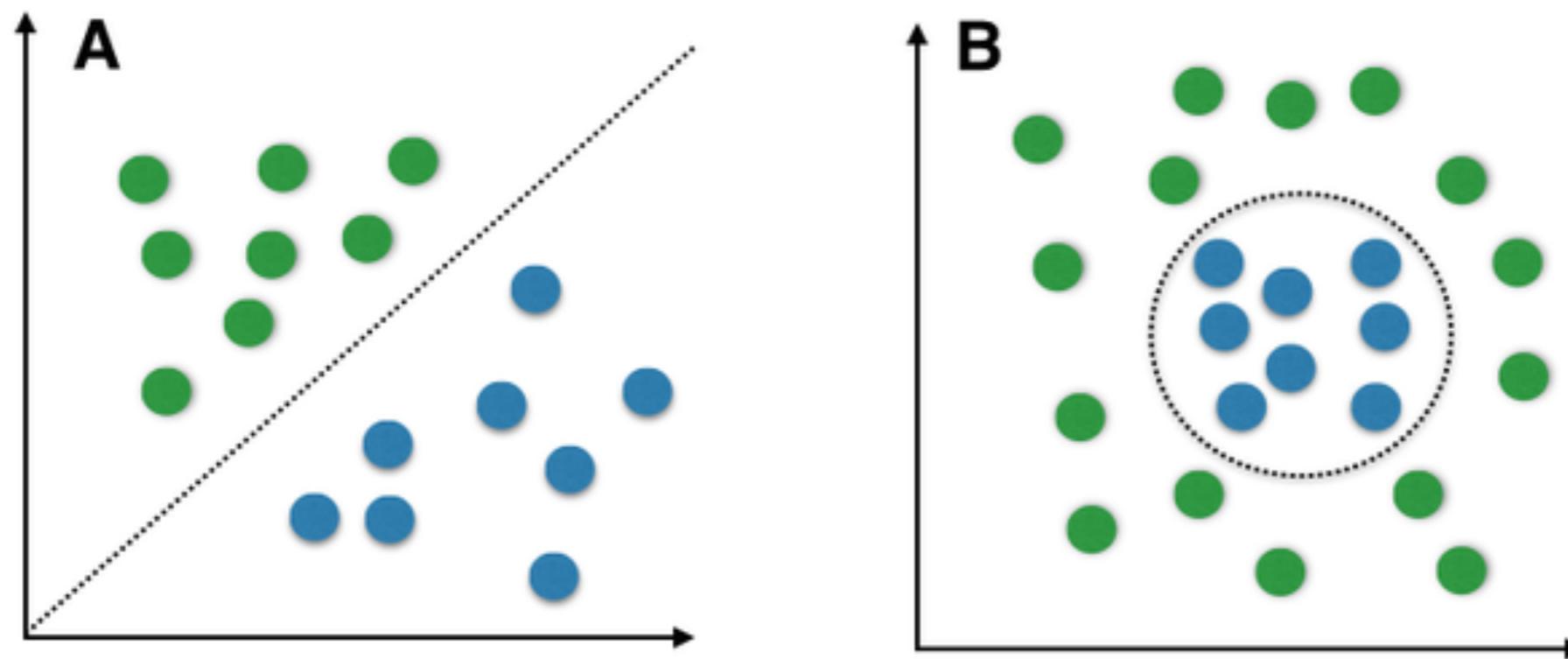


Surgen las primeras Redes Neuronales



# Redes Neuronales

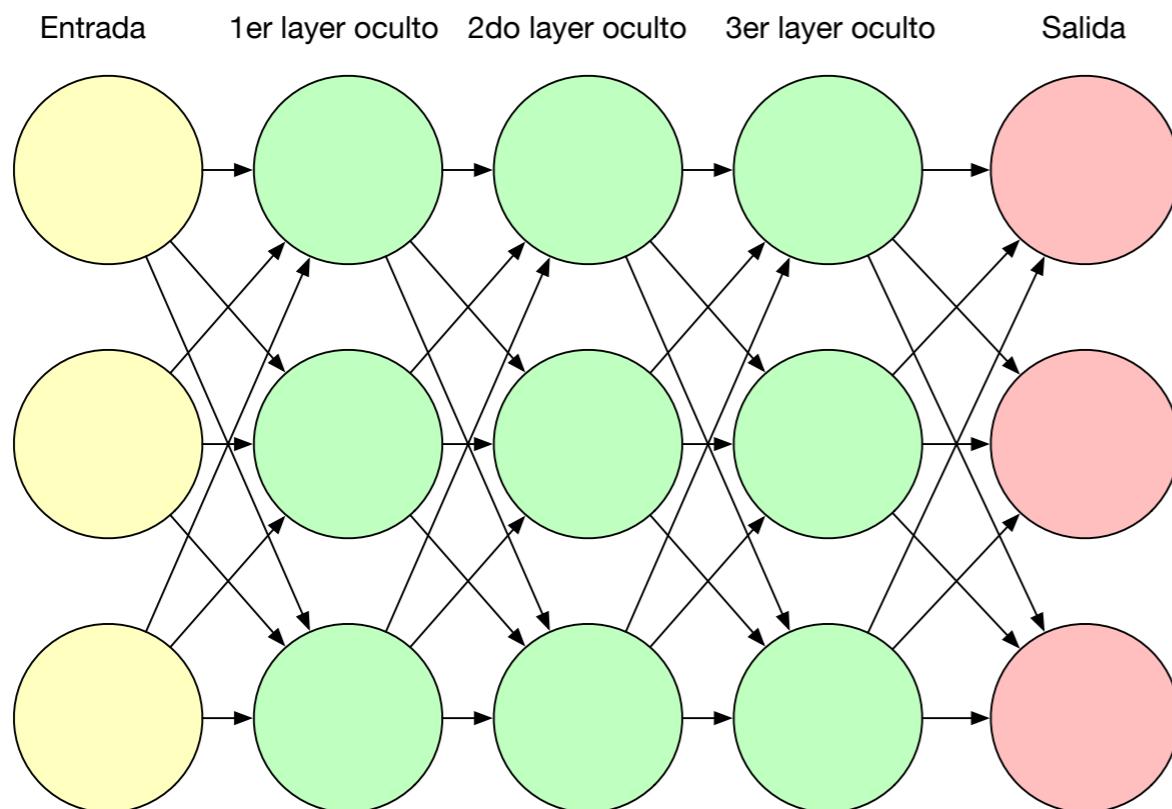
- ❖ Perceptrón (Rosenblatt, 1958, 1962): solo puede resolver problemas linealmente separables



Si el problema es linealmente separable, el algoritmo del perceptrón para encontrar los pesos encuentra la solución en tiempo finito

# Redes Neuronales

- ❖ Perceptron multi-capas (Tesis de P. Werbos, 1974 )



- ❖ Feed-forward: Nos vamos a centrar en este tipo de redes, pero también están las recurrentes.
- ❖ Una red con más de una capa puede resolver un problema arbitrario si tiene suficientes neuronas.
- ❖ No hay algoritmo con un teorema de convergencia

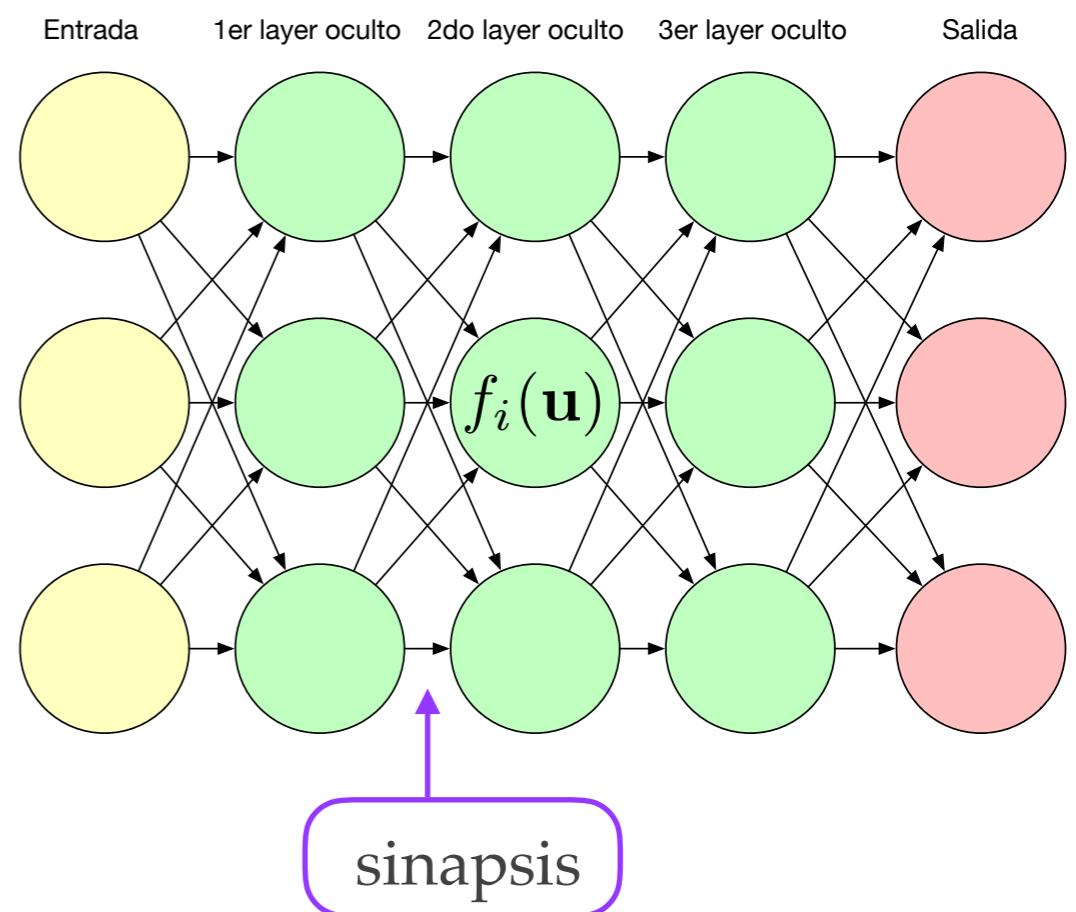
# Deep learning

## ❖ Neural Networks (NN).

$$\mathbf{y} = f(\mathbf{x})$$

neuronas/unidades  
 $z_i = f_i(\mathbf{u})$

Las redes neuronales modernas **no buscan** modelar el cerebro humano.



Mejor verlas como aproximaciones de algún tipo de generalización, ocasionalmente inspiradas en el funcionamiento cerebral, más que modelar el funcionamiento cerebro.

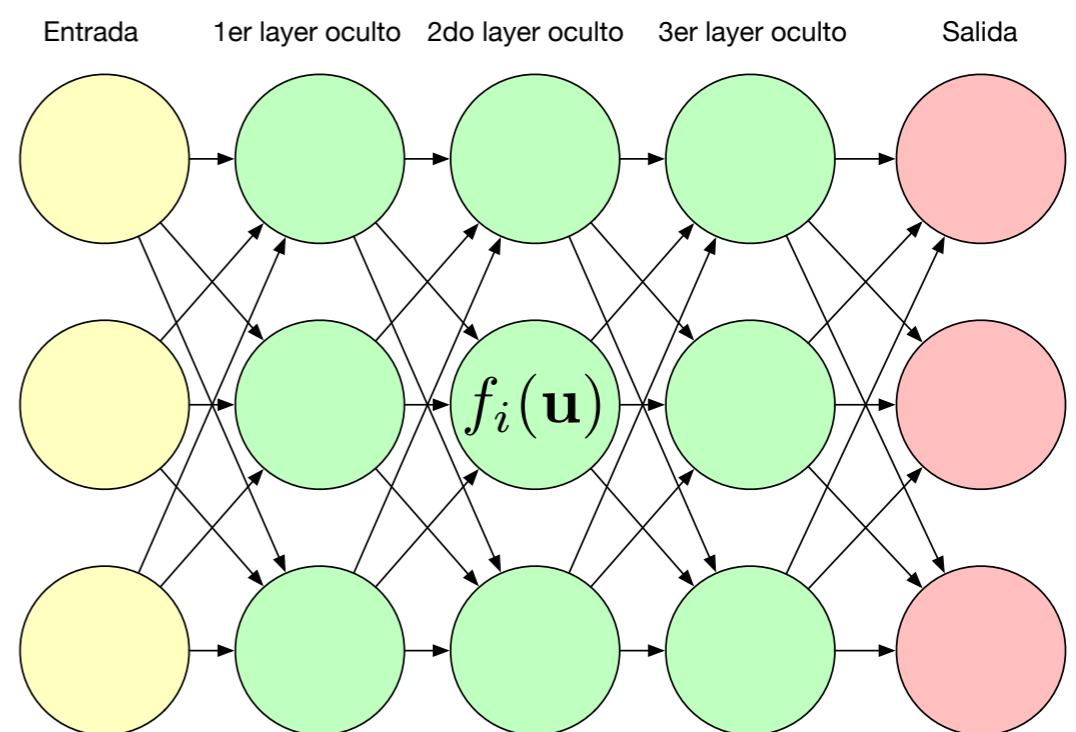
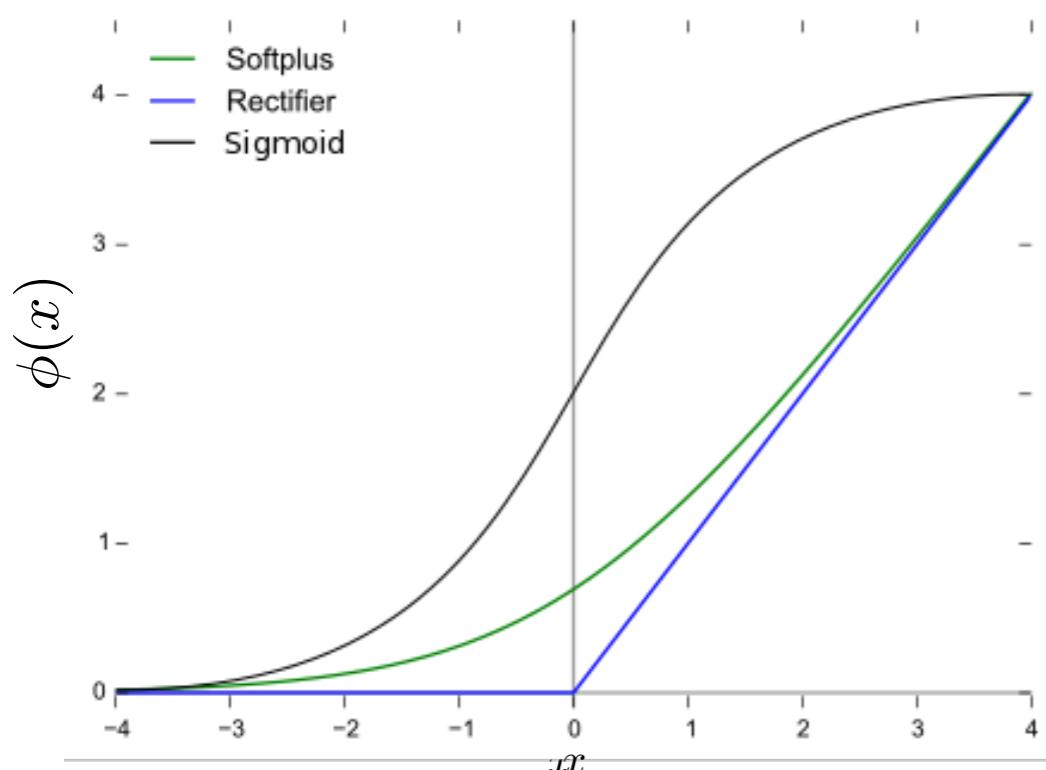
# Deep learning

- ❖ Neural Networks (NN).

- ❖ neuronas o *hidden unit*

$$f_i(\mathbf{u}) = \phi \left( \sum_i w_i \mathbf{u}_i + b \right)$$

- ❖ función de activación

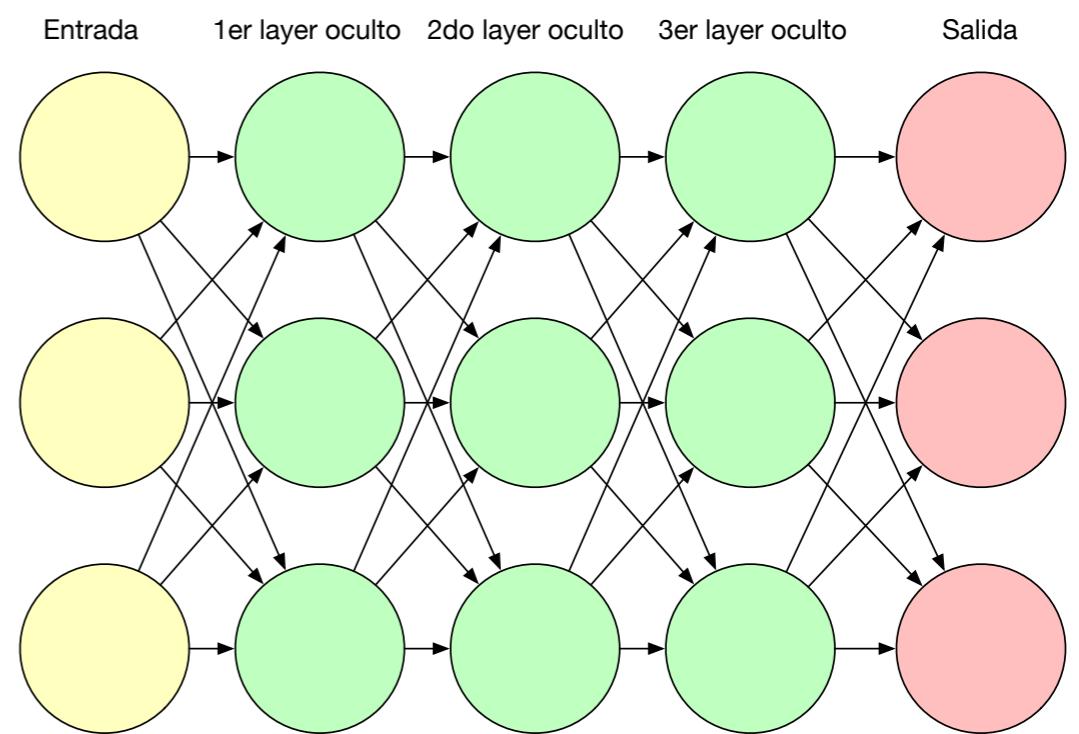


- ❖ Arquitectura:
  - ❖ cuantas neuronas
  - ❖ como se conectan las neuronas
  - ❖ que funciones de activación utilizar

# Deep learning

## ❖ Salida de la NN.

- ❖ La forma de representar la salida por lo general va a determinar la función de costo a utilizar.
  - ❖ Linear unit: la más simple donde no hay función no lineal.
  - ❖ Sigmoid: predecir el valor de una clasificación binaria.
  - ❖ Softmax: predecir el valor de una clasificación con  $n$  valores.



Las neuronas de salida no son diferentes al resto, sacando las de entrada.

# Redes Neuronales

---

- ❖ Las arquitecturas feed-forward implementan mapeos entrada-salida. Pensar como una función compuesta por una gran cantidad de composición de funciones no lineales.
  - ❖ Clasificación (salidas es discreta).
  - ❖ Aproximación de funciones (salidas continuas).

¿ Cómo elegimos los pesos ?

---

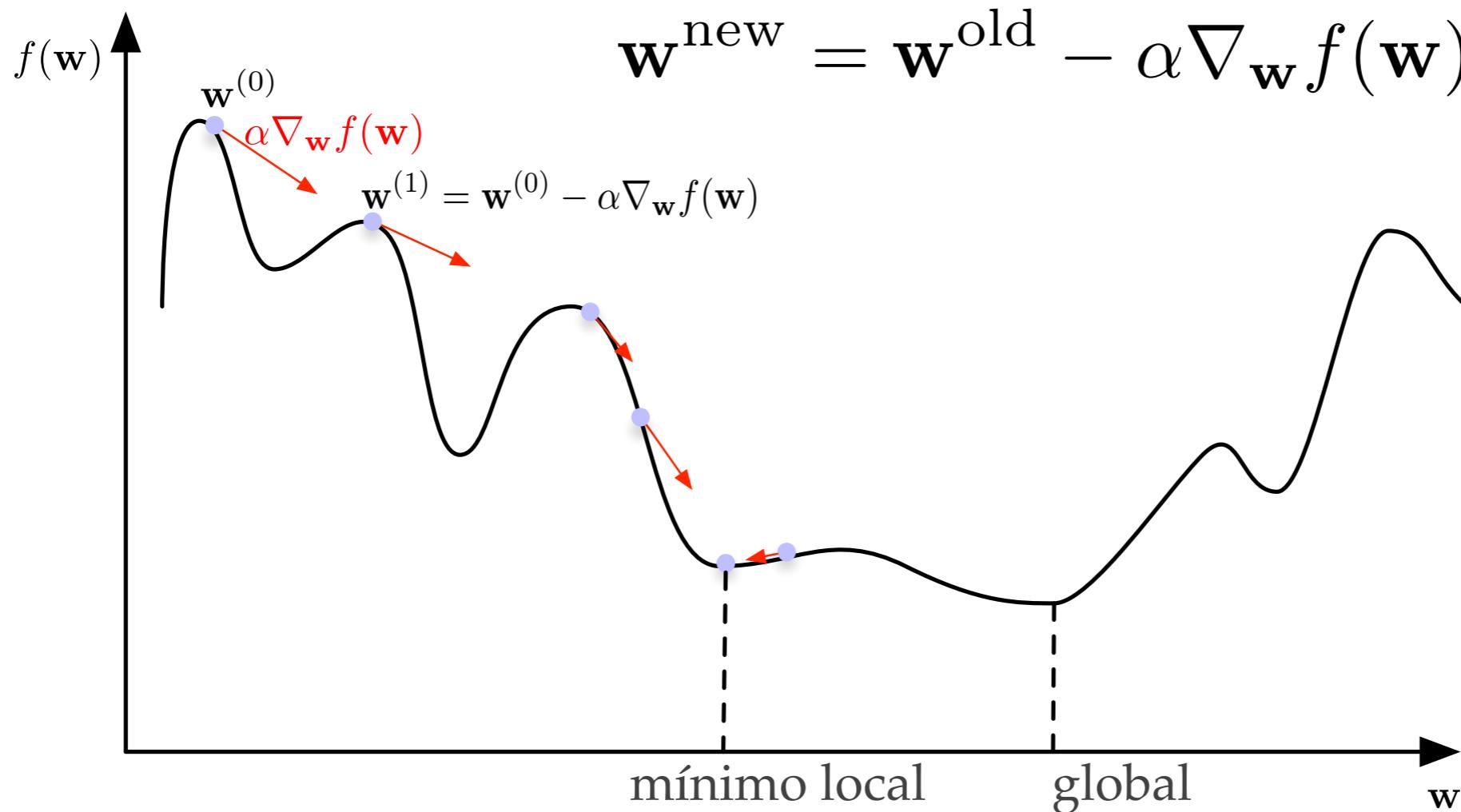
# Redes Neuronales

---

- ❖ Algoritmo de aprendizaje
  1. Se presenta una entrada en particular.
  2. Se calcula la salida.
  3. Se calcula el error, por ejemplo la diferencia cuadrática entre lo que da y lo que debe dar.
  4. Se suma sobre todas las entradas.
  5. Se minimiza este error respecto a los parámetros de la red, es decir, los pesos.

# Redes Neuronales

- ❖ La minimización se realiza mediante algún método de optimización. Por ejemplo, gradiente descendente.
- ❖ Necesitamos el gradiente del error respecto de los pesos.



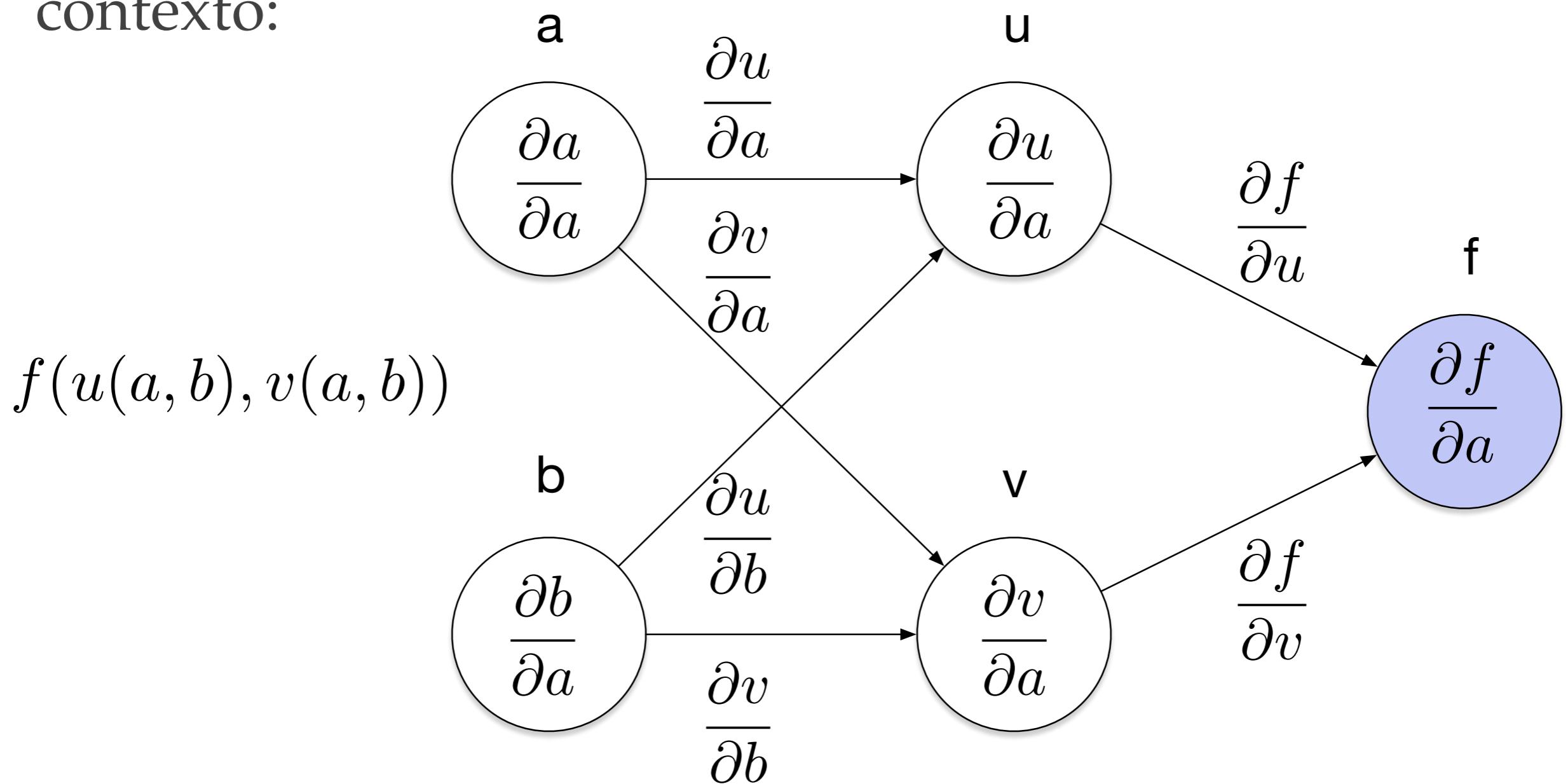
# Redes Neuronales

---

- ❖ Back-propagation (D.E. Rumelhart, G.E. Hinton et al., 1986; LeCun, 1985)
  - ❖ En sí es una forma **inteligente** de aplicar la **regla de la cadena** para calcular derivadas.
  - ❖ Aunque se asocie siempre a redes neuronales, el método de back-propagation **NO** es solo para redes neuronales.

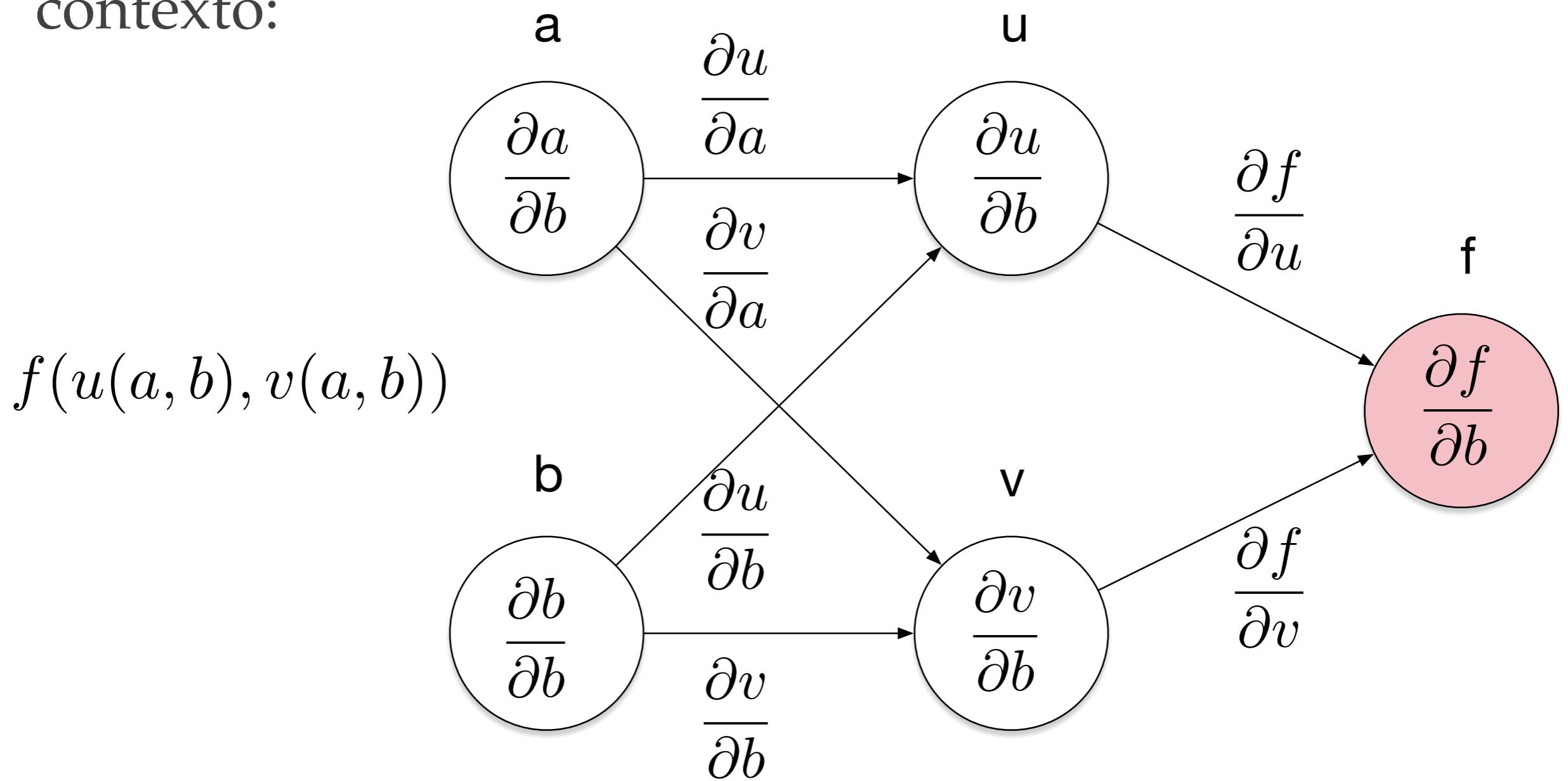
# Redes Neuronales

- ❖ Veamos como se aplica la regla de la cadena en este contexto:



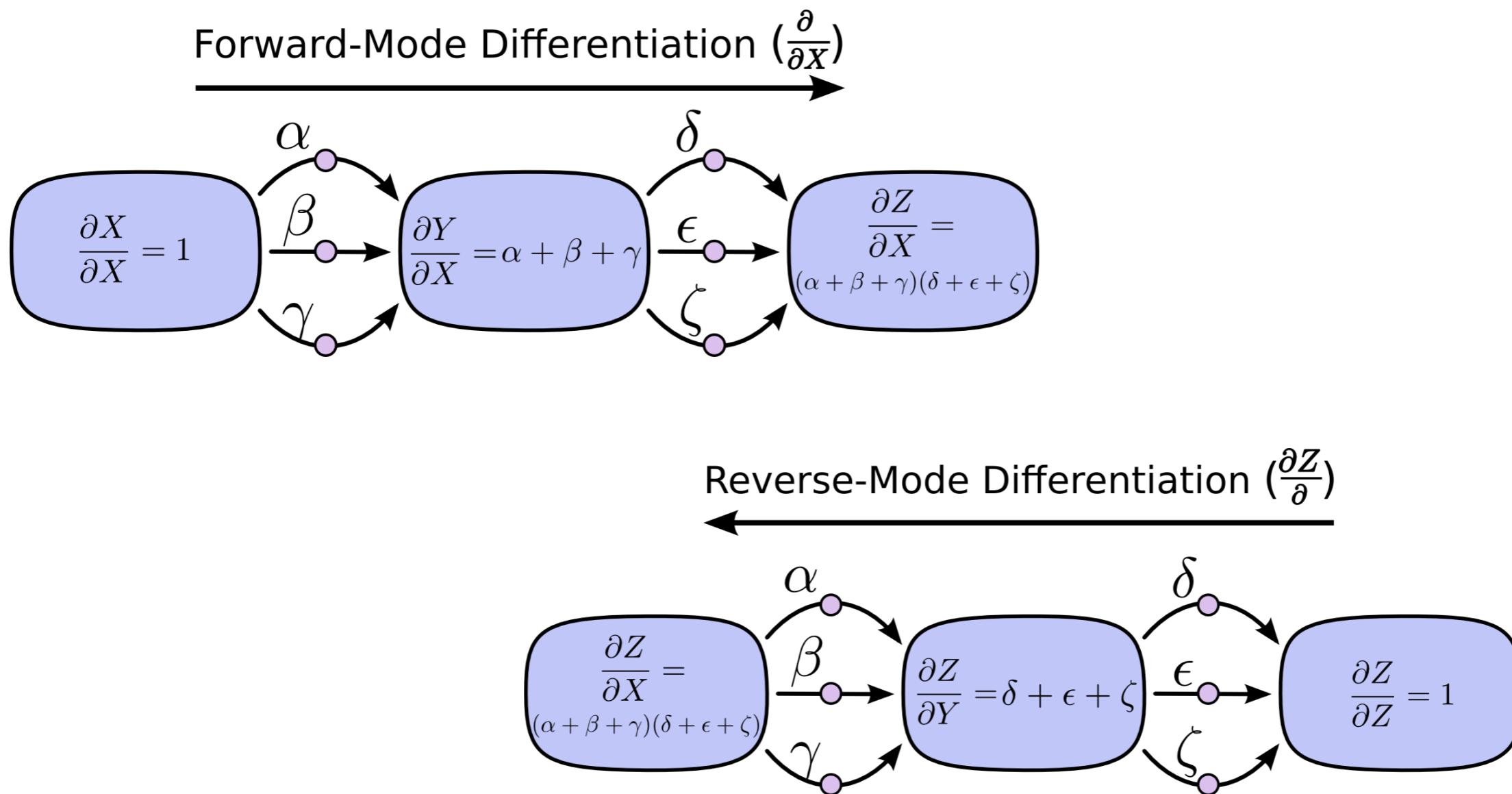
# Redes Neuronales

- ❖ Veamos como se aplica la regla de la cadena en este contexto:



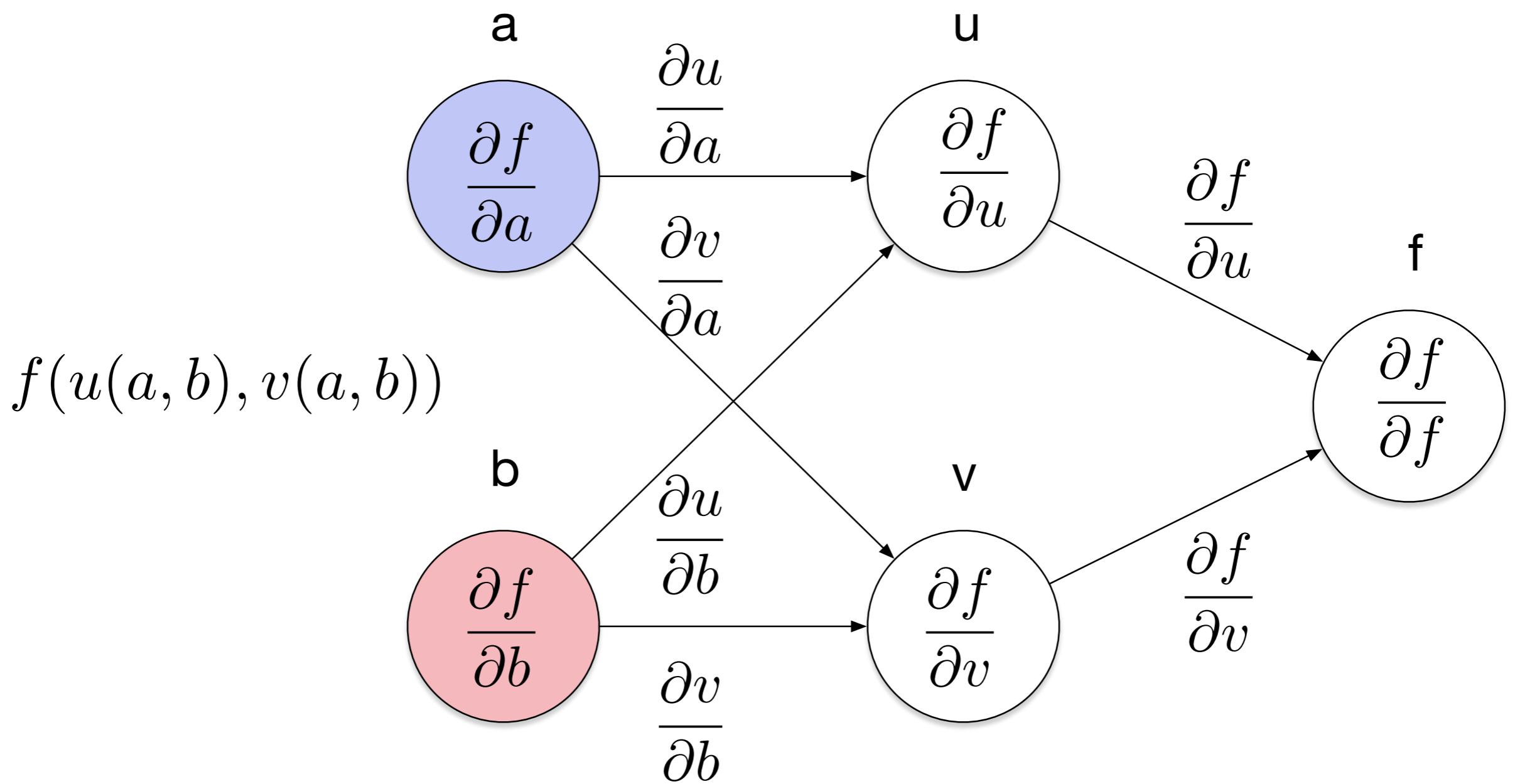
# Redes Neuronales

- ❖ Back-propagation (reverse mode differentiation).



# Redes Neuronales

- ❖ Back-propagation (reverse mode differentiation).



# Redes Neuronales

- ❖ Algoritmo de aprendizaje con Back-propagation.

- ❖ Conjunto de datos de entrada-salida:

$$X = (\mathbf{x}_i, \mathbf{y}_i) \quad i = 1 \dots P$$

- ❖ Conjunto de parámetros de la red (pesos y umbrales):  $\Theta_k$

- ❖ Función error:

$$E(\Theta) = \frac{1}{2P} \sum_i^P |\mathbf{y} - \hat{\mathbf{y}}(\mathbf{x}_i, \Theta)|^2$$

donde  $\hat{\mathbf{y}}(\mathbf{x}_i, \Theta)$  es la salida **observada** cuando la entrada es  $\mathbf{x}_i$  e  $\mathbf{y}_i$  es la salida deseada.

El error se puede minimizar mediante  $\Delta\Theta_k = -\eta \frac{\partial E}{\partial \Theta_k}$

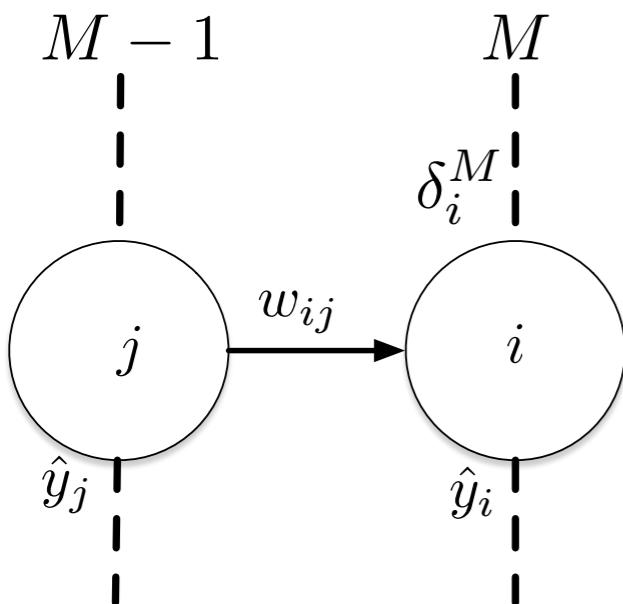
# Redes Neuronales

- ❖ Algoritmo de aprendizaje con Back-propagation (continuación).

La salida de una neurona esta dada por:

$$\hat{y}_i = g(h_i) = g\left(\sum_j w_{ij} \hat{y}_j\right)$$

- ❖ El gradiente de los parámetros (pesos) de las neuronas de la capa de salida para una entrada dada es el siguiente:



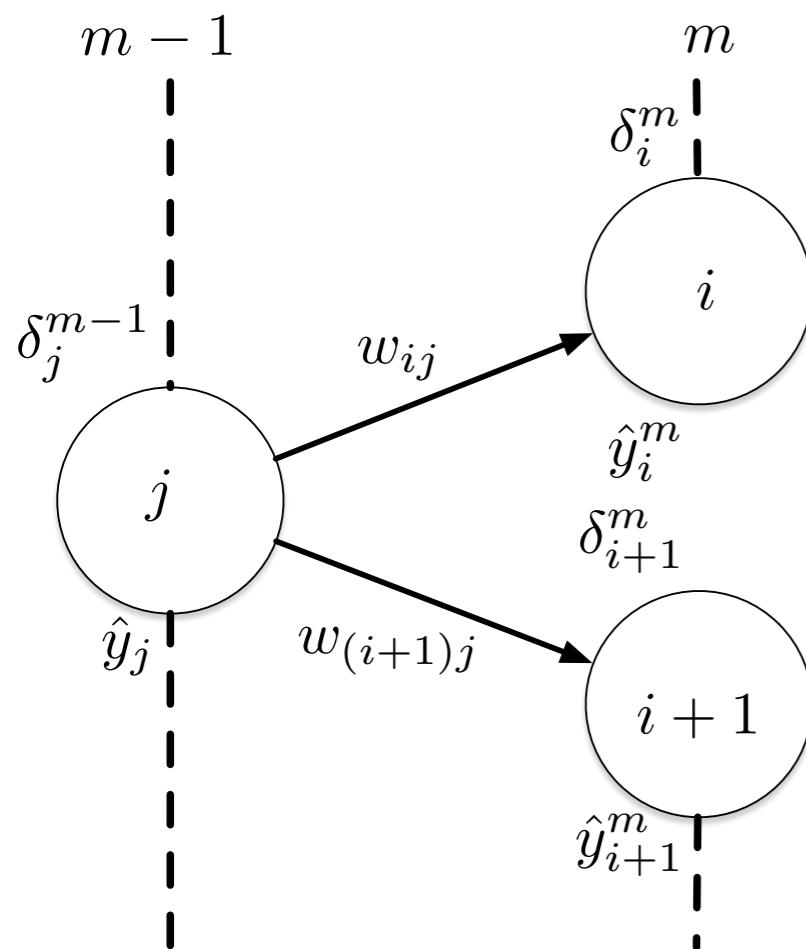
$$\frac{\partial E}{\partial w_{ij}} \approx -(y_i - \hat{y}_i^M)g'(h_i^M)\hat{y}_{j-1}^M = \delta_i^M \hat{y}_{j-1}^M$$

$\delta_i^M$ : se corresponde con la variación del error E respecto del peso  $w_{ij}$

# Redes Neuronales

- ❖ Algoritmo de aprendizaje con Back-propagation (continuación).

Se puede demostrar mediante la regla de la cadena (reverse mode differentiation), que el gradiente de los parámetro (pesos) que no son los de las neuronas de salidas tiene la forma:



$$\delta_j^{m-1} = g'(h_j^{m-1}) \sum_i w_{ij}^m \delta_i^m$$

Repetimos el cálculo de delta para todas las capas desde M-1 hasta la 2

# Redes Neuronales

- ❖ Algoritmo de aprendizaje con Back-propagation (continuación).

Una vez estimado todos los delta calculamos el gradiente y actualizamos los parámetros de la red utilizando gradiente descendente de la siguiente forma:

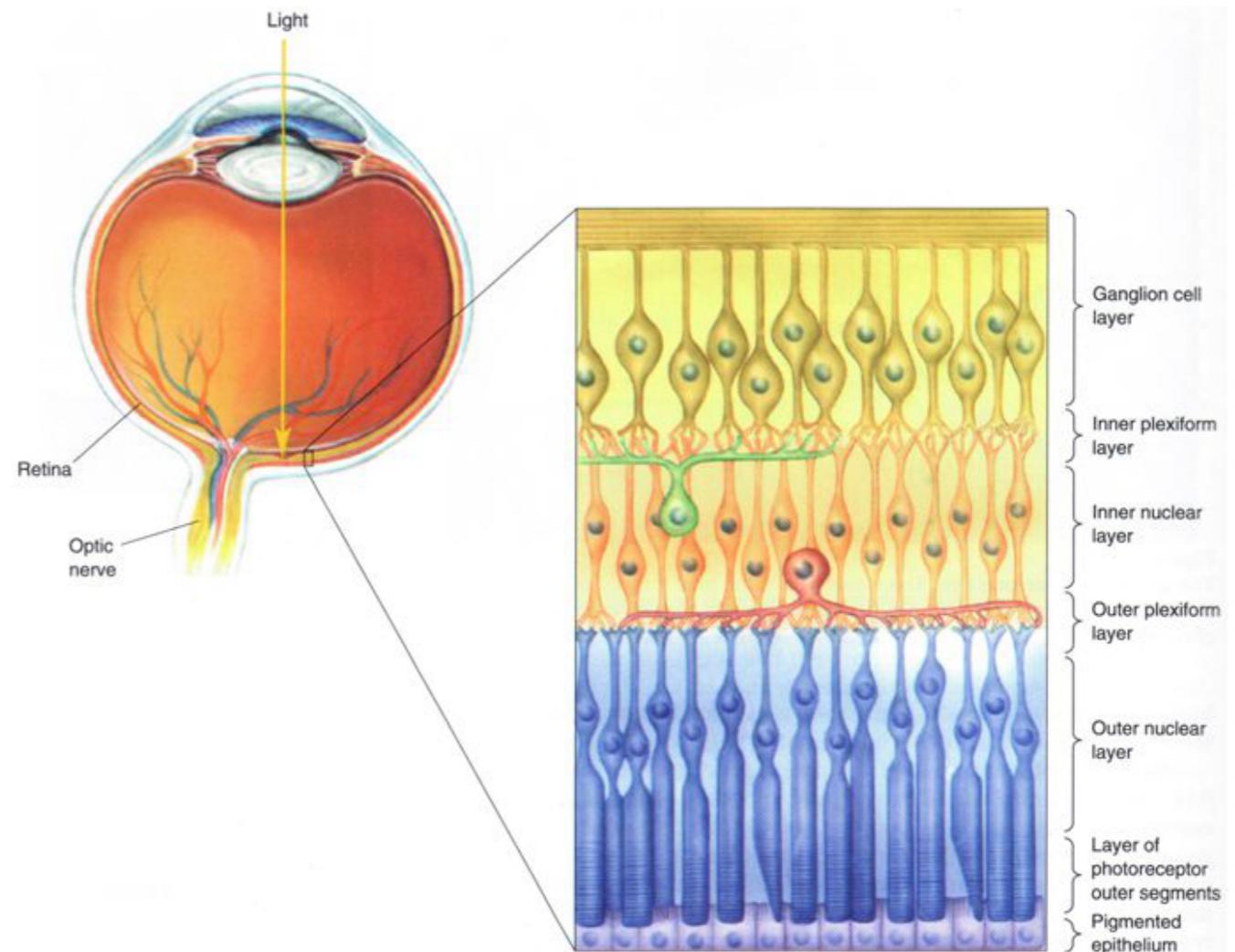
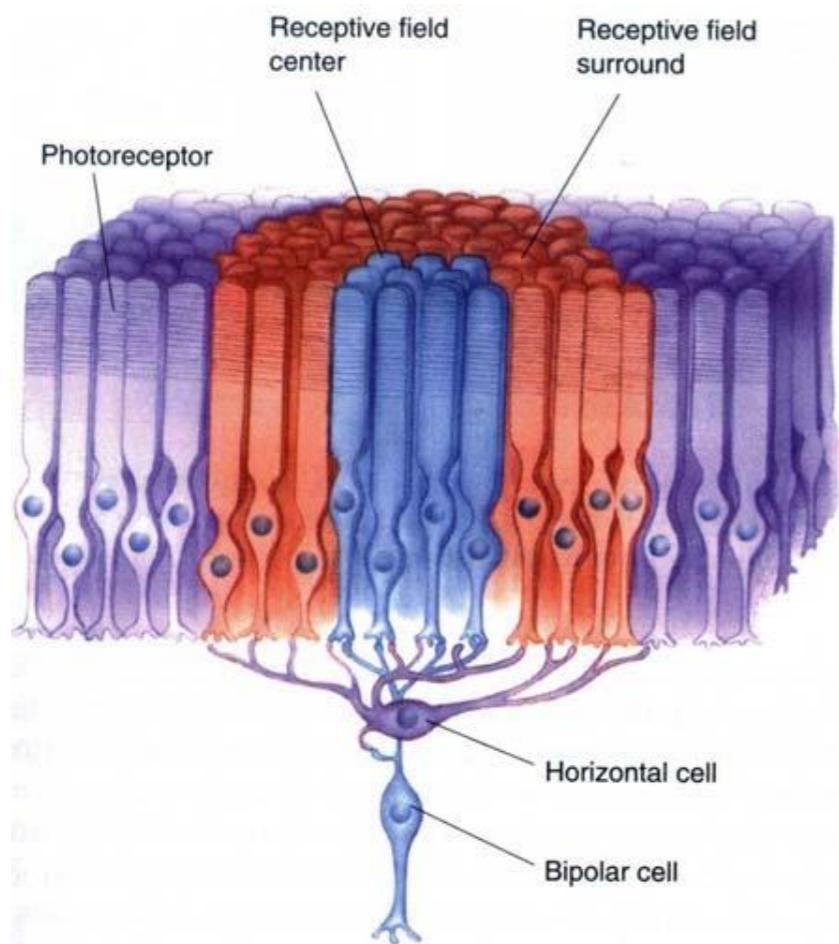
$$\Delta w_{ij}^m = -\eta \delta_i^m \hat{y}_j^{m-1}$$

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \Delta w_{ij}^m$$

Repetimos este procedimiento de calcular los delta y ajustar los pesos para todos los ejemplos de entrenamiento

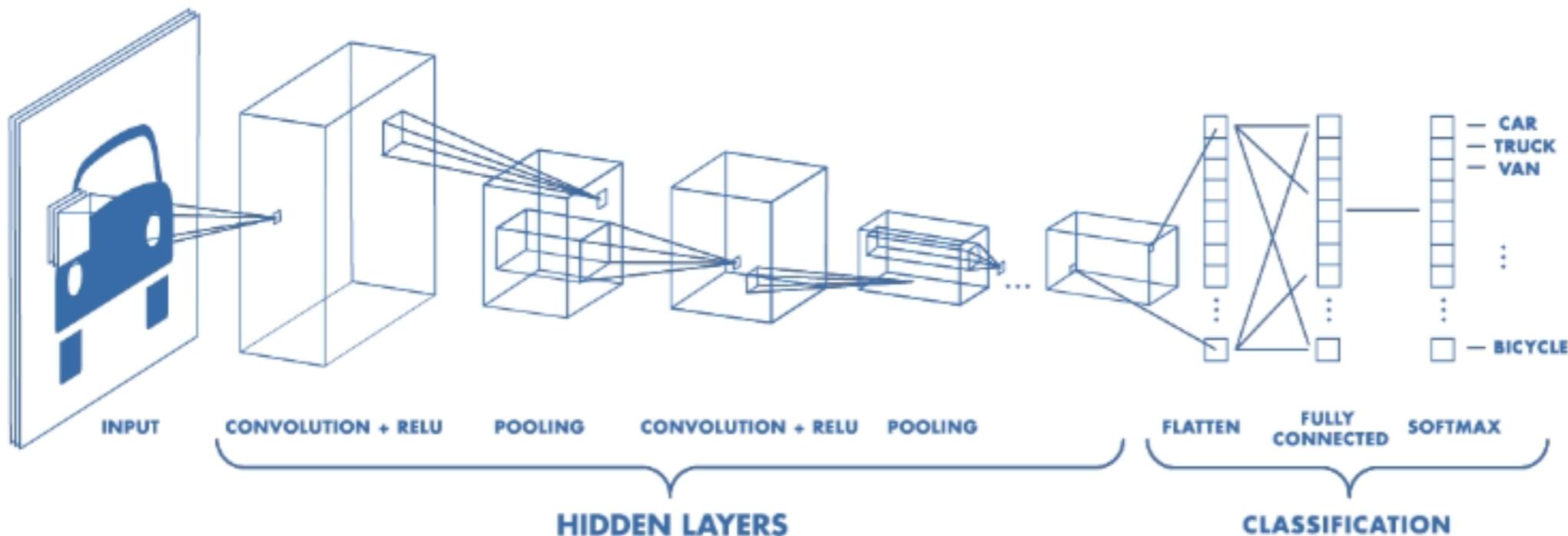
# Redes Convolucionales

- ❖ Convolutional Neural Networks (LeCun, 1989): Este tipo de redes neuronales esta inspirada en la idea de campo receptivo de los sistemas sensoriales.
  - ❖ Por ejemplo, las células ganglionares de la retina pueden ser excitadas o inhibidas dependiendo del lugar del campo receptivo donde pasa el estimulo visual.



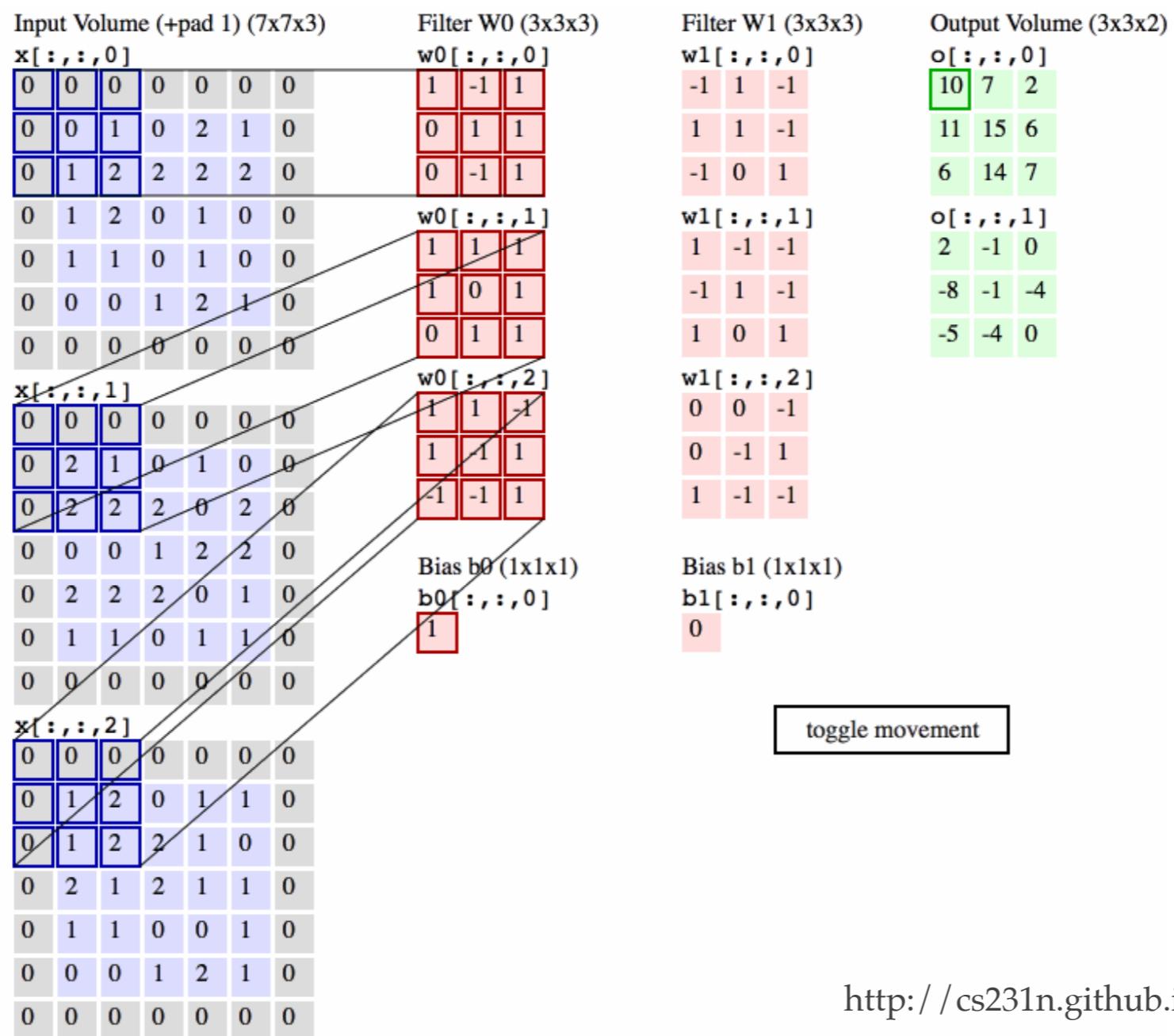
# Redes Convolucionales

- ❖ Convolutional Neural Networks (LeCun, 1989).
  - ❖ Cada neurona busca identificar en la capa anterior rasgos específicos.
  - ❖ Todas las neuronas de una capa buscan el mismo rasgo pero en una posición diferente.
  - ❖ Representation learning: intentamos aprender la representación de los datos y además la relación entre esta representación y la salida (primeras clases).
  - ❖ Estas dos etapas se ven bien marcadas en CNN's



# Deep learning

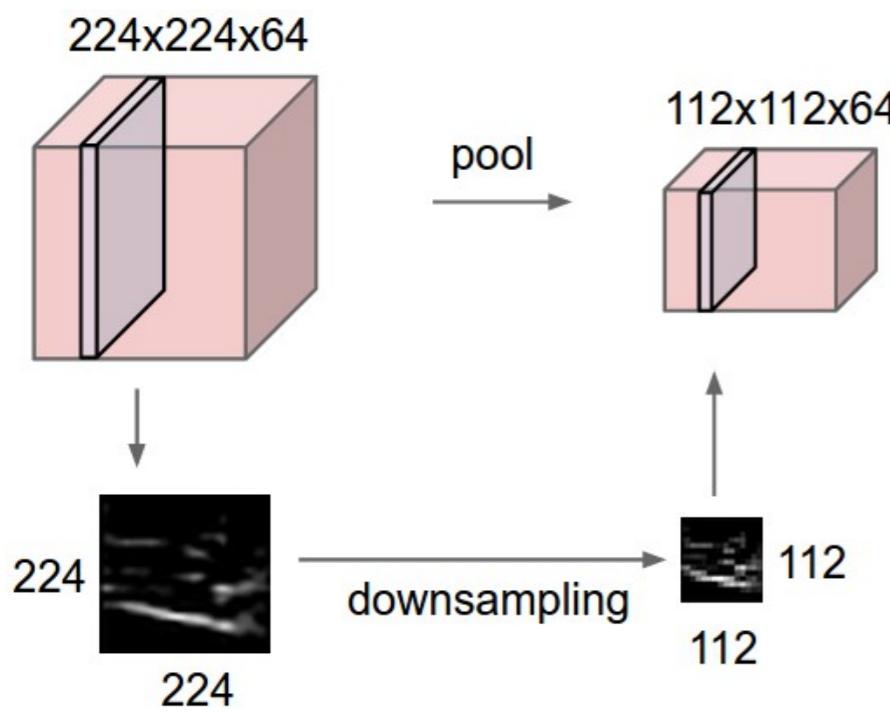
## ❖ Layer de Convolución



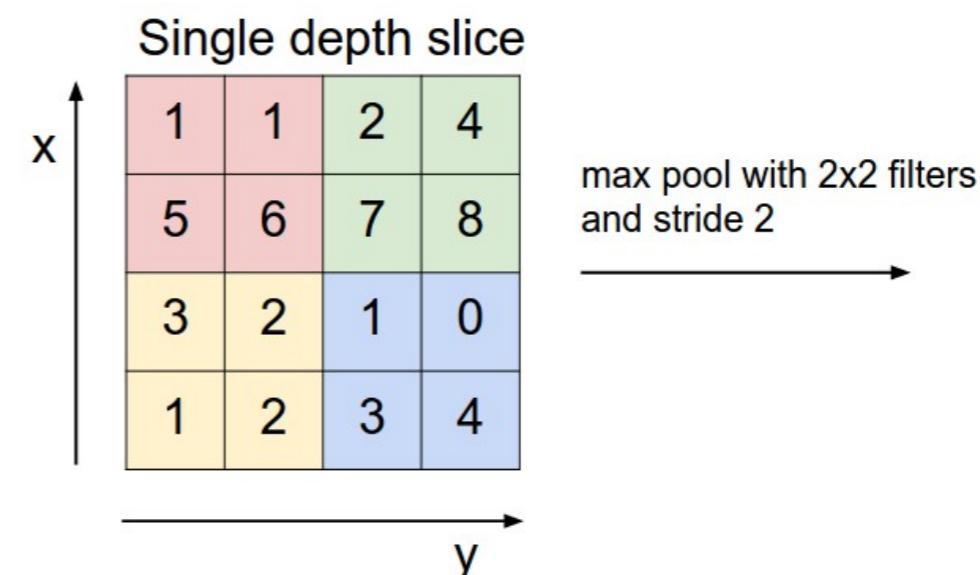
# Deep learning

## ❖ Submuestreo

### ❖ Pool



### ❖ Max pooling



---

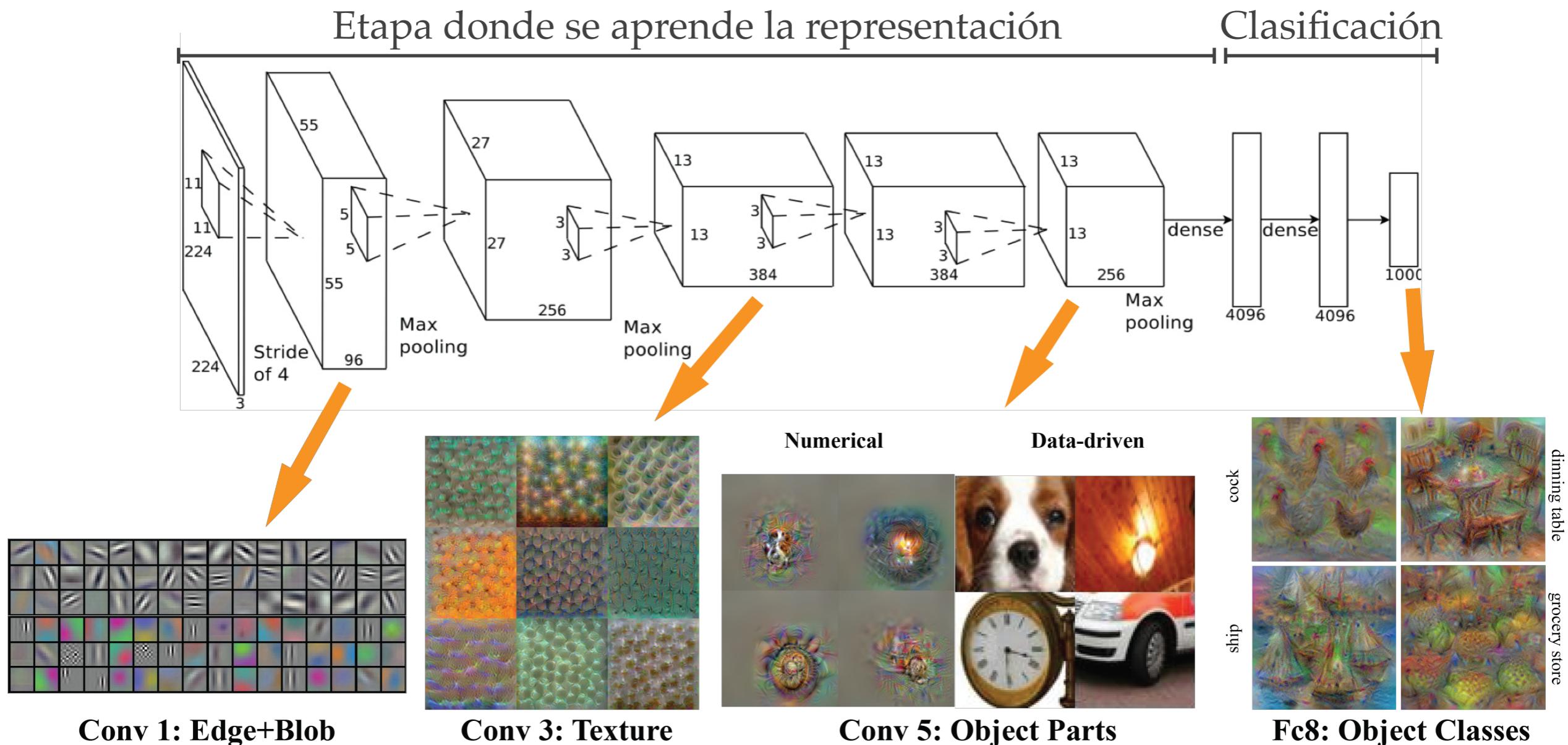
# Redes Convolucionales

---

- ❖ Convolutional Neural Networks (LeCun, 1989).
  - ❖ Se logra reducir notablemente la cantidad de parámetros a entrenar, pero aún no se cuenta con el poder de cómputo adecuado para ser aplicarlo a problemas interesantes.
- ❖ *Deep learning* [2006s - hoy].

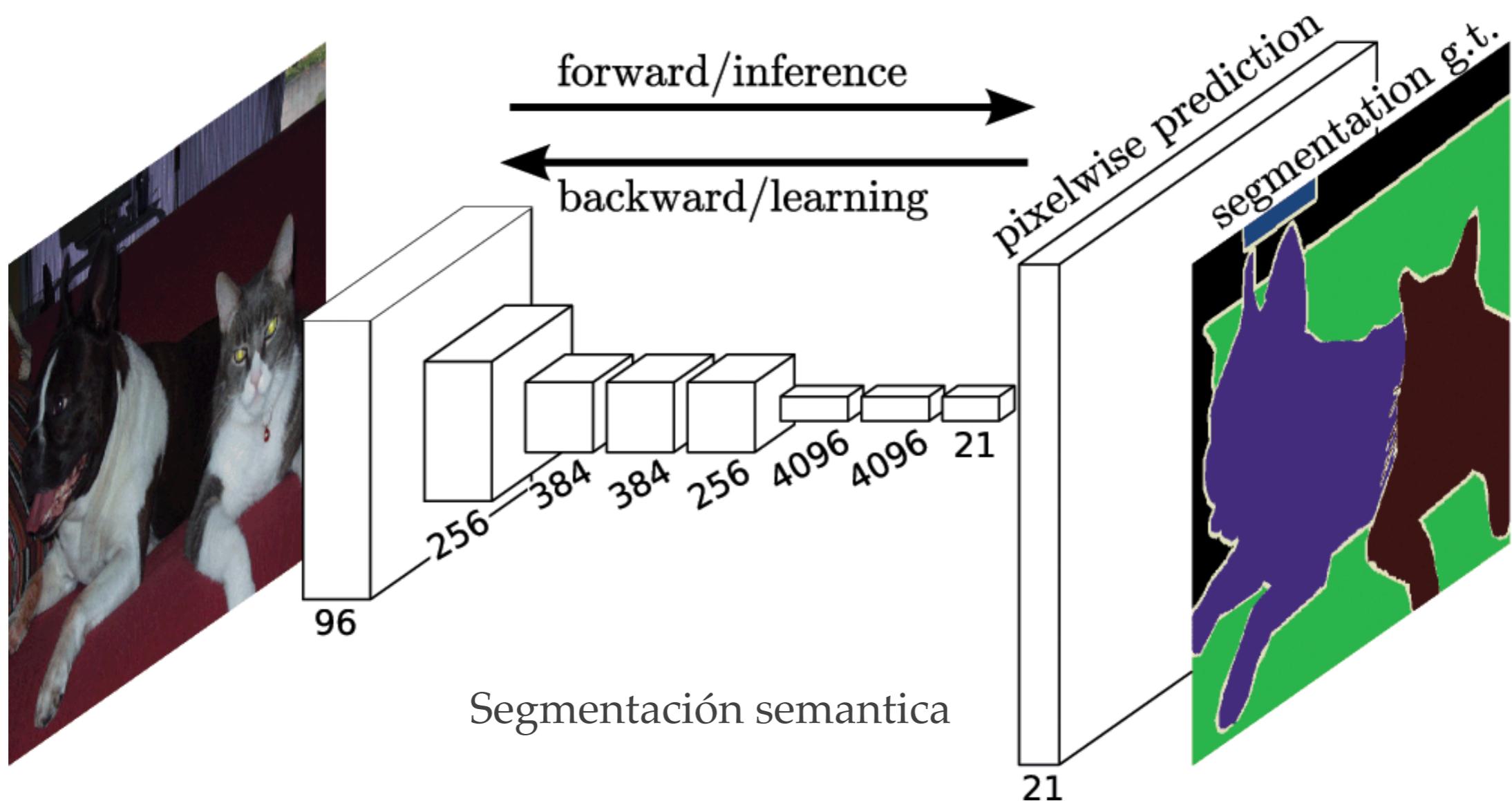
# Deep learning

## ❖ AlexNet.



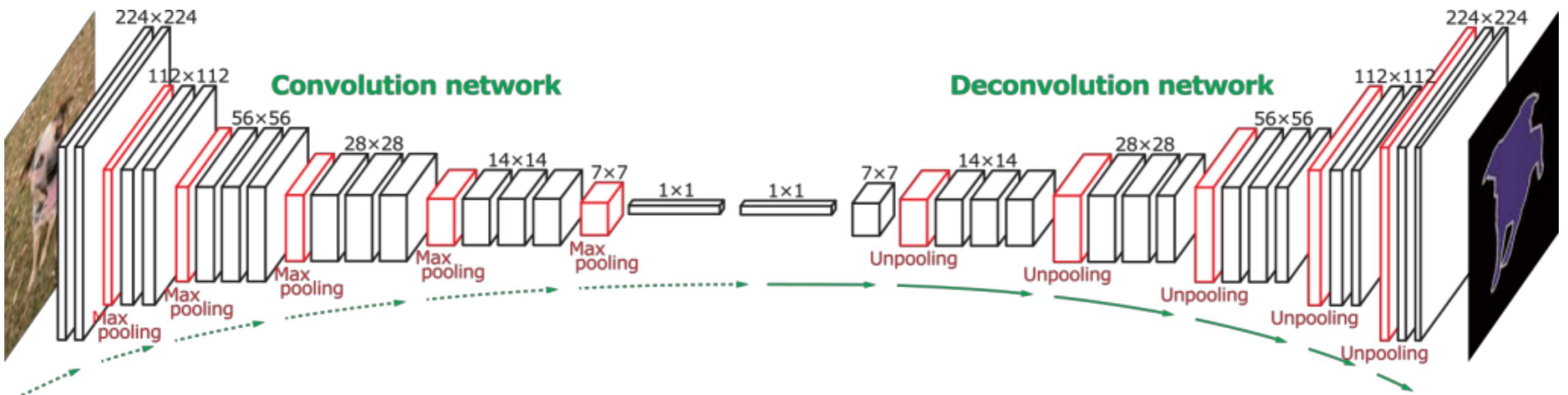
# Deep learning

- ❖ Fully convolutional neural networks (predicciones densas).



# Deep learning

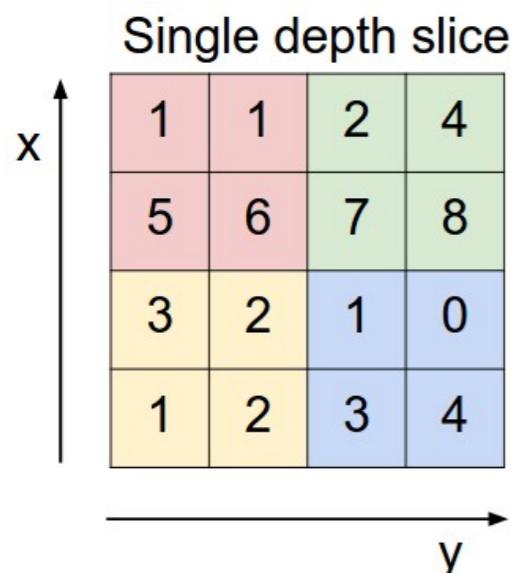
- ❖ Fully convolutional neural networks (autoencoder).



# Deep learning

- ❖ Unpooling

  - ❖ Max pool





  - ❖ Unpooling

