

# Deep Learning I

## *Introducción al machine learning y a las redes neuronales*

---

MIGUEL ÁNGEL MARTÍNEZ DEL AMOR

DEPARTAMENTO CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

UNIVERSIDAD DE SEVILLA

A solid green horizontal bar at the bottom of the slide.

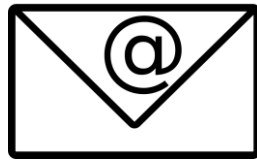
# About me

---

- **Miguel Ángel Martínez del Amor**
- Profesor Ayudante Doctor del Departamento de Ciencias de la Computación e Inteligencia Artificial



[www.cs.us.es/~mdelamor](http://www.cs.us.es/~mdelamor)



[mdelamor@us.es](mailto:mdelamor@us.es)



[@miguelamda](https://twitter.com/miguelamda)



[miguelamda](https://github.com/miguelamda)



[Research Group on Natural Computing](#)



[DeepKnowledge](#)



DEEP  
LEARNING  
INSTITUTE

[NVIDIA Deep Learning Institute](#)

# Índice

---

1. Motivación
2. Introducción al Machine Learning
3. Redes neuronales multicapa
4. Optimización de redes neuronales
5. Entornos software para Deep Learning
6. Nuestra primera red con Keras

# Índice

---

1. Motivación
2. Introducción al Machine Learning
3. Redes neuronales multicapa
4. Optimización de redes neuronales
5. Entornos software para Deep Learning
6. Nuestra primera red con Keras

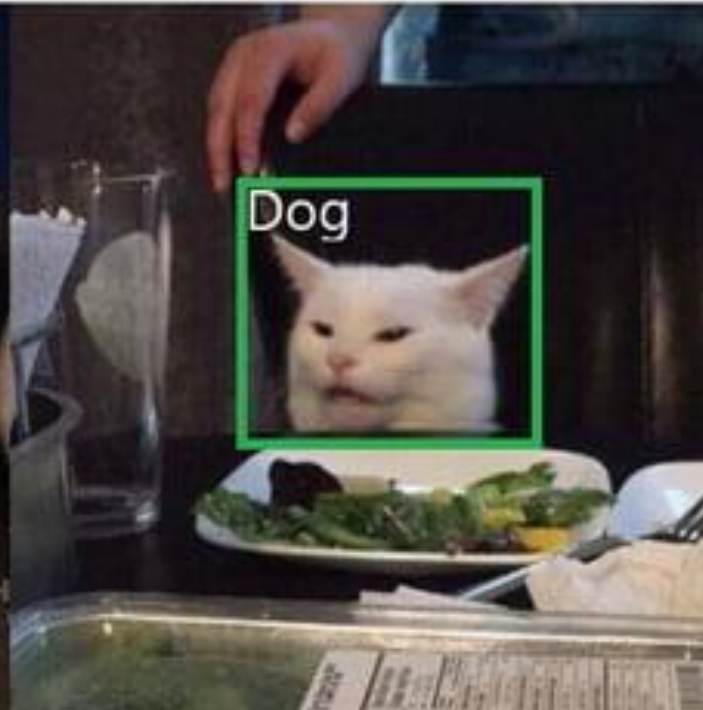
# ¿Inteligencia Artificial?

---

People with no idea  
about AI, telling me my  
AI will destroy the world

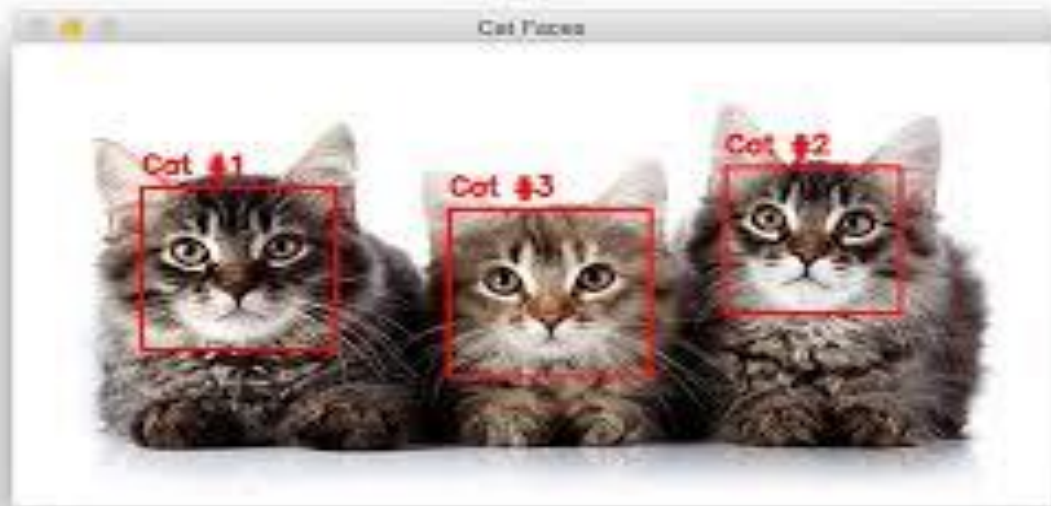


Me wondering why my  
neural network is  
classifying a cat as a dog..



# Motivación

Clasificación de objetos (y gatos...)



First Name

L	O	R	I										
---	---	---	---	--	--	--	--	--	--	--	--	--	--

Last Name

W	A	L	T	E	R	S							
---	---	---	---	---	---	---	--	--	--	--	--	--	--

# Motivación

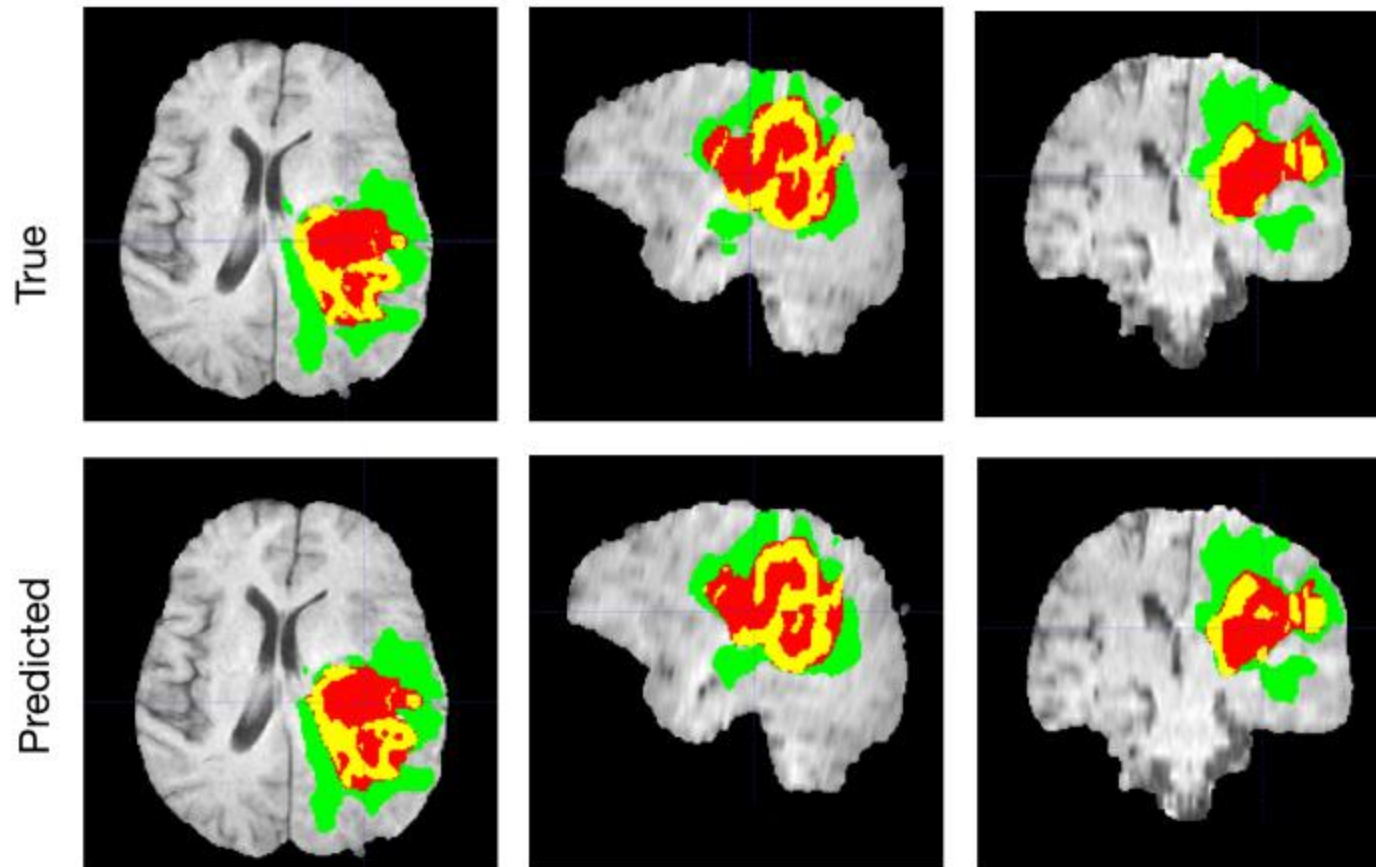
---

## Localización de objetos en imágenes



# Motivación

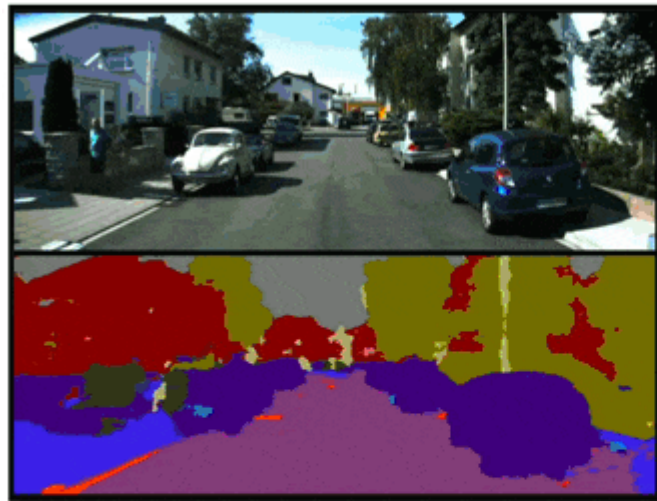
## Segmentación de tumores



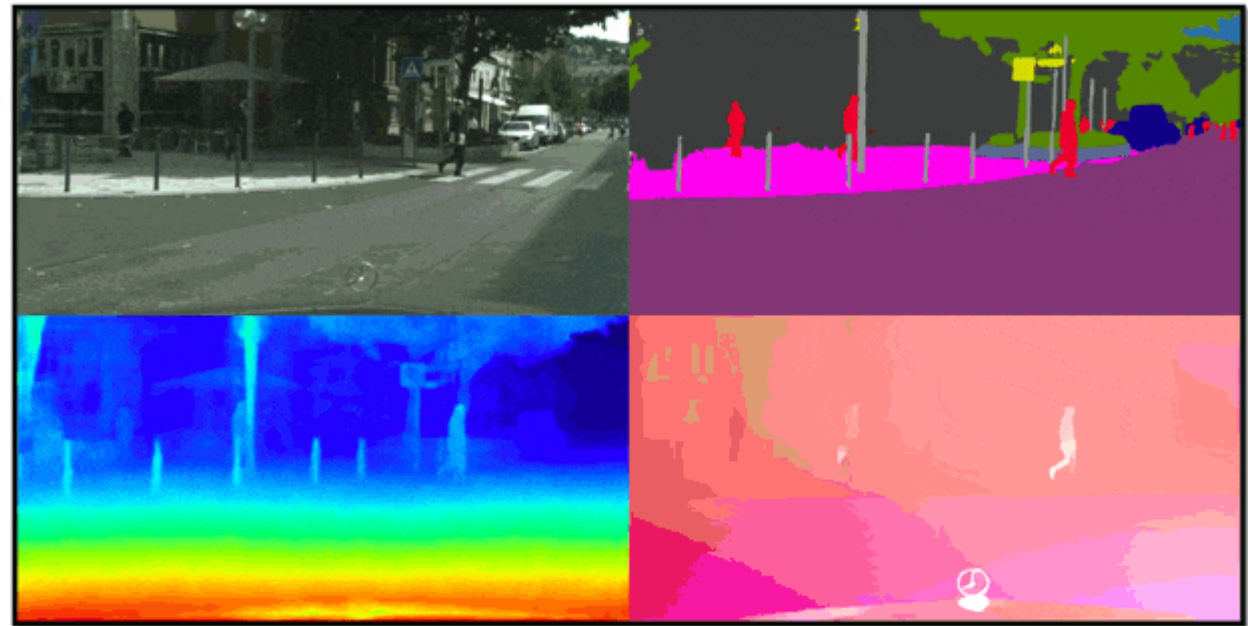


# Motivación

Conducción autónoma (segmentación, localización obstáculos)



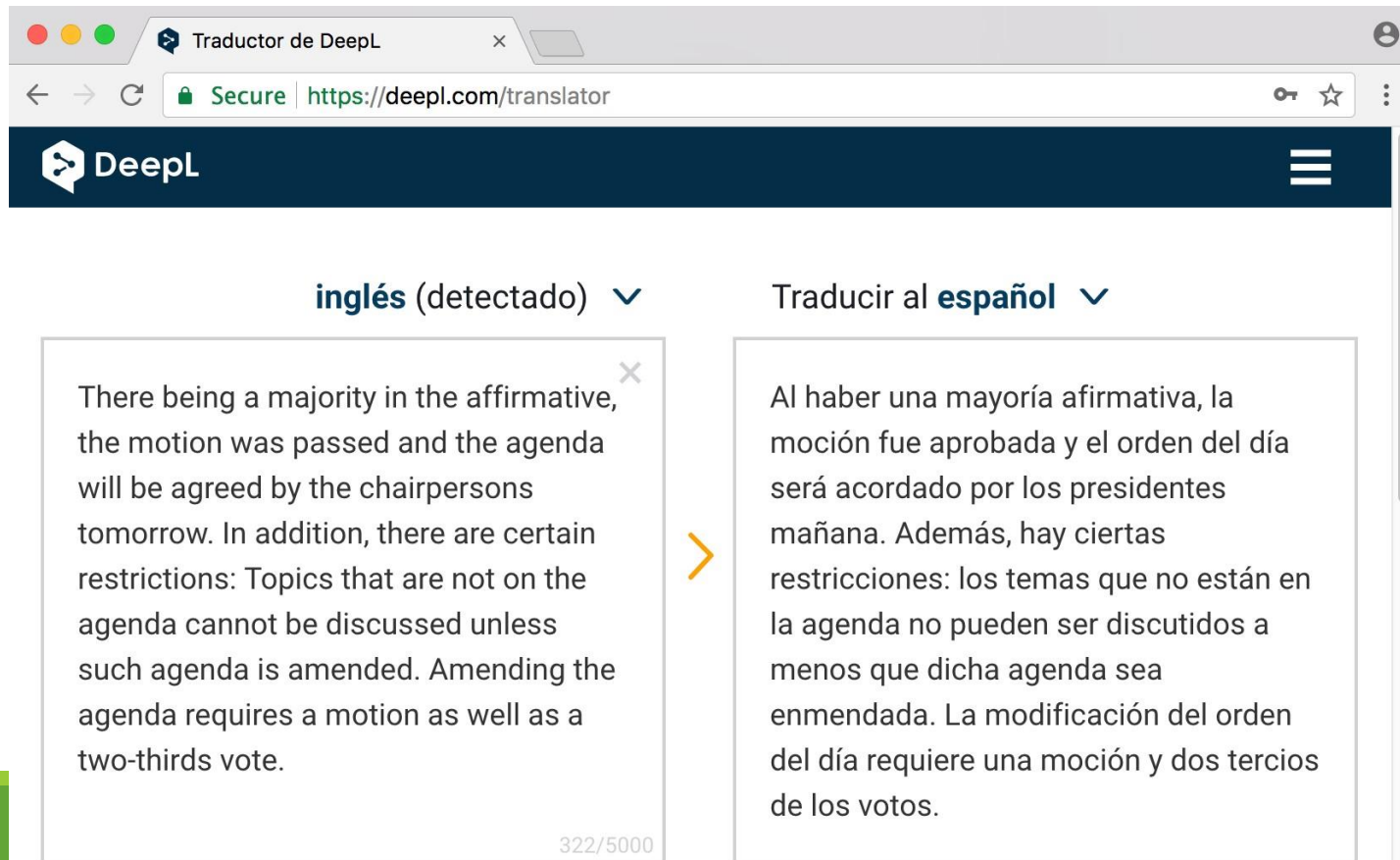
**Progression of  
computer vision from  
2015**



**... to 2018**

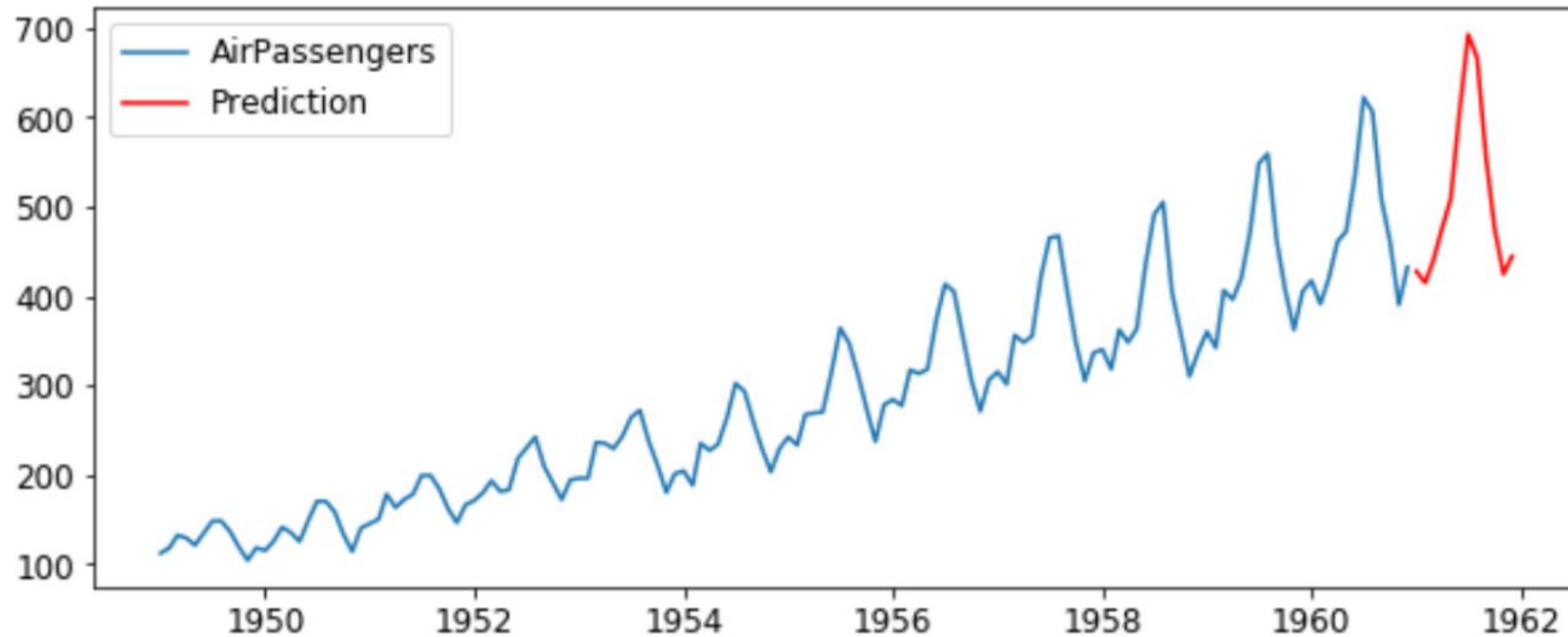
# Motivación

## Traductores automáticos



# Motivación

## Series temporales



# Motivación

---

Generando caras artificiales





# Motivación

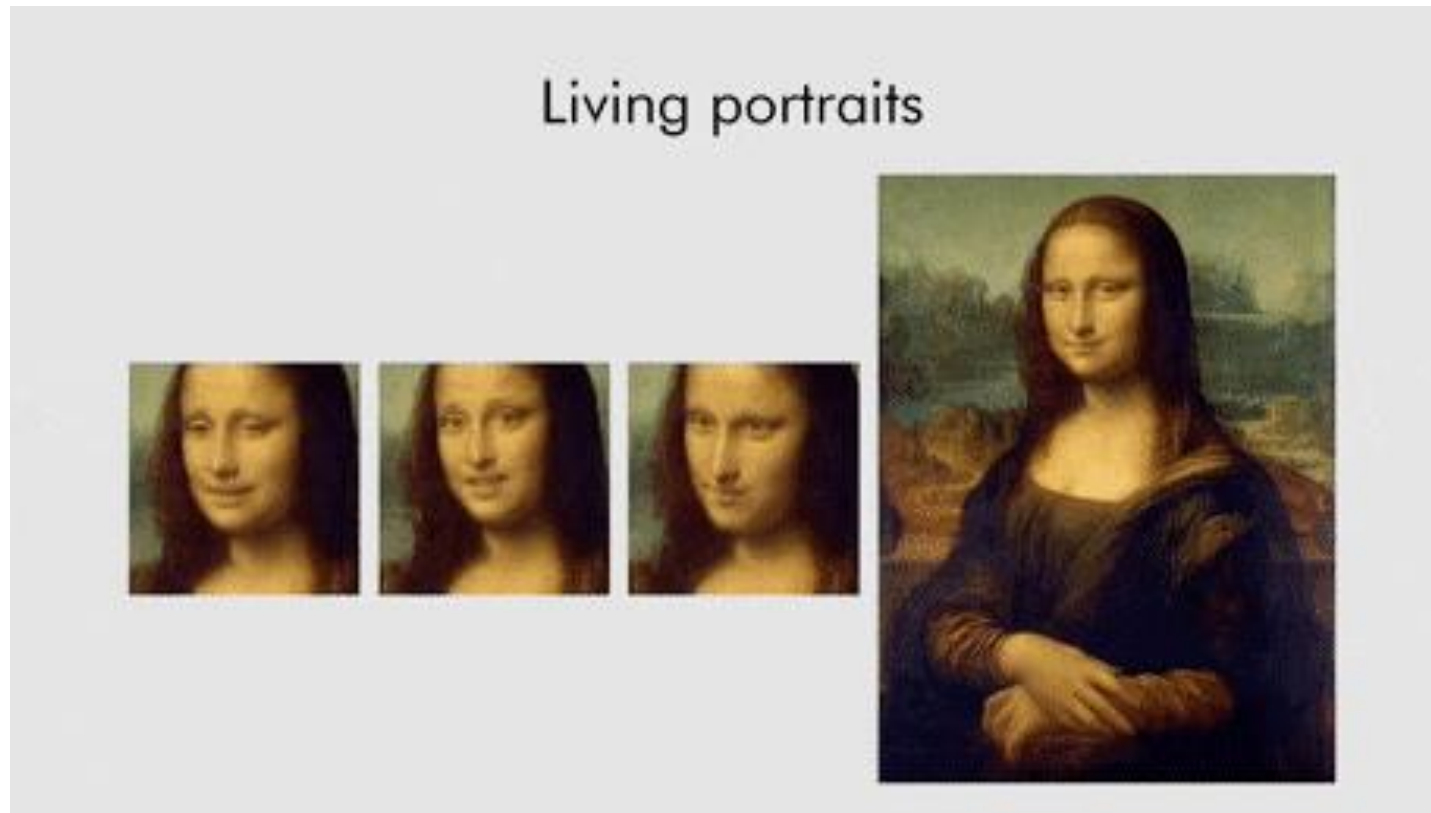
Paisajes fotorealísticos desde un dibujo



# Motivación

---

Haciendo hablar a la Mona Lisa:

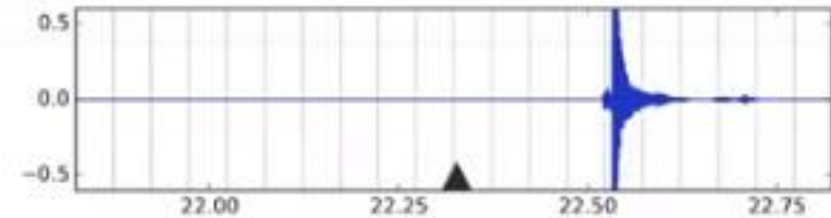


# Motivación

Añadir audio a vídeos sin sonido



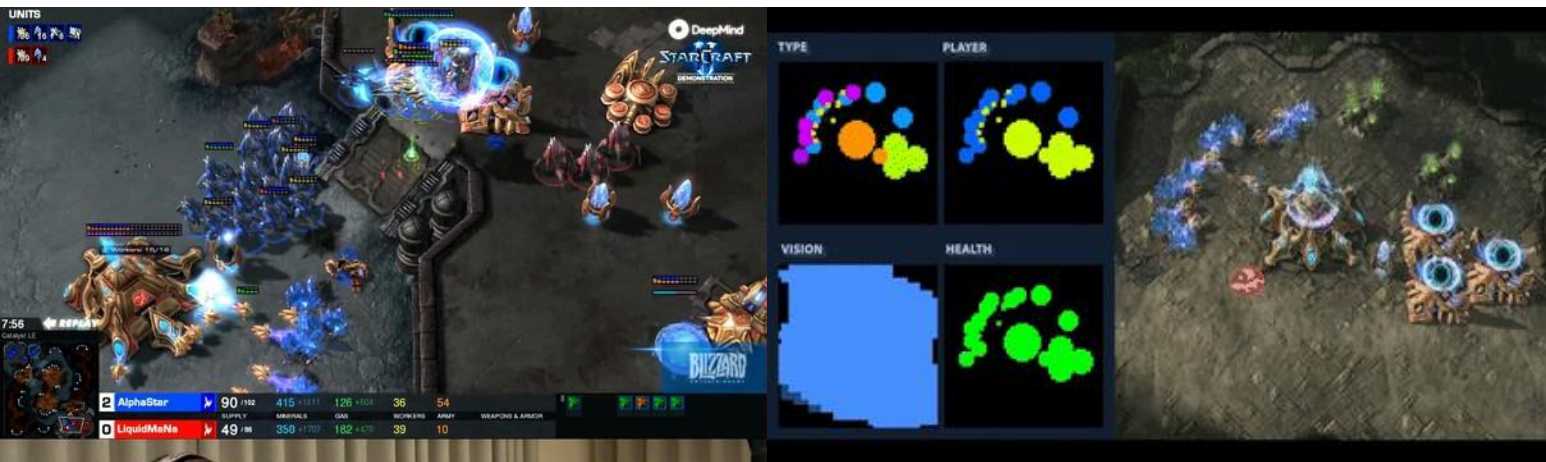
Silent video



Predicted soundtrack

# Motivación

Ganar a StarCraft II (**AlphaStar**) y al GO (**AlphaGo**)



© 2019 DeepMind Technologies LTD  
www.deepmind.com



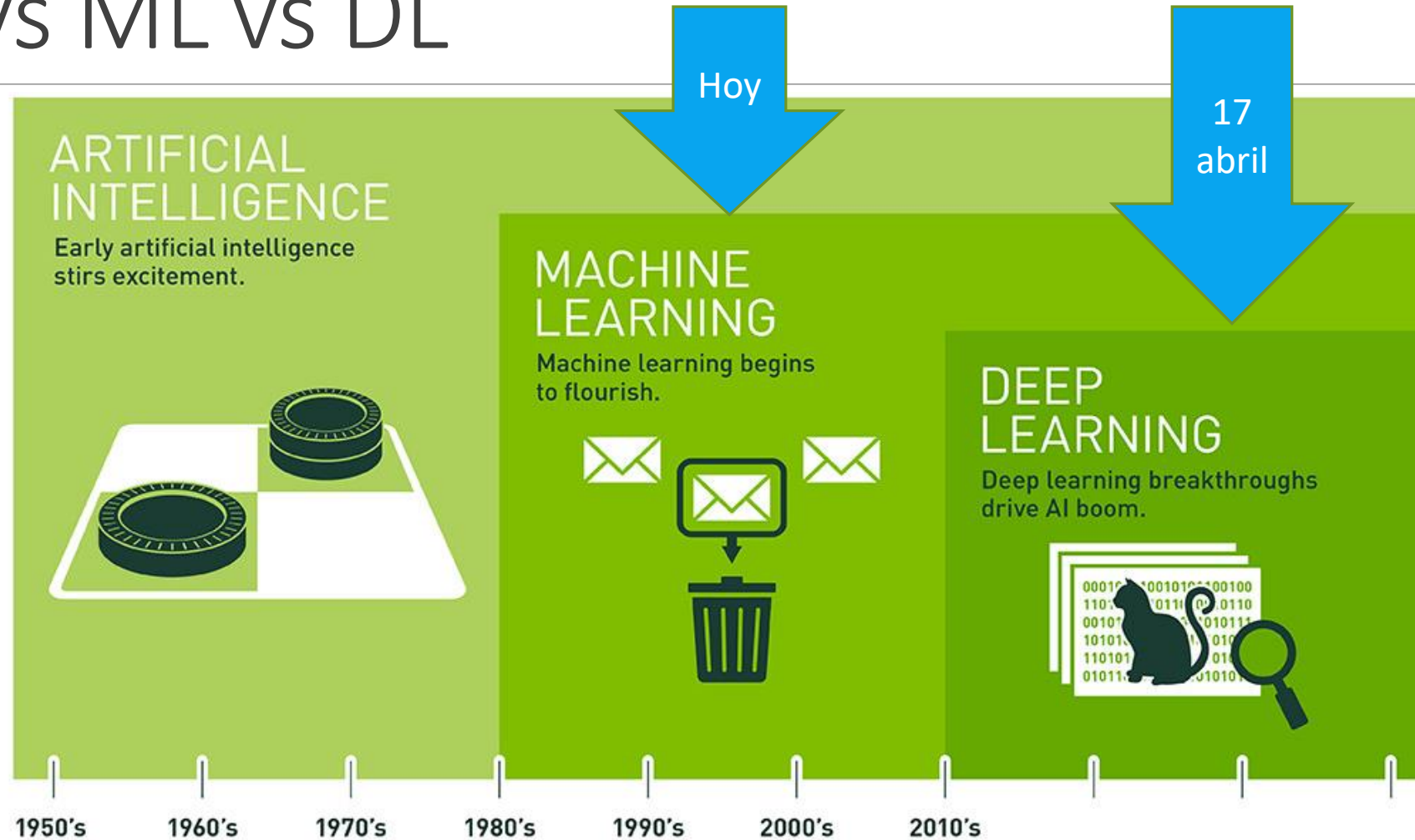


# Índice

---

1. Motivación
2. Introducción al Machine Learning
3. Redes neuronales multicapa
4. Optimización de redes neuronales
5. Entornos software para Deep Learning
6. Nuestra primera red con Keras

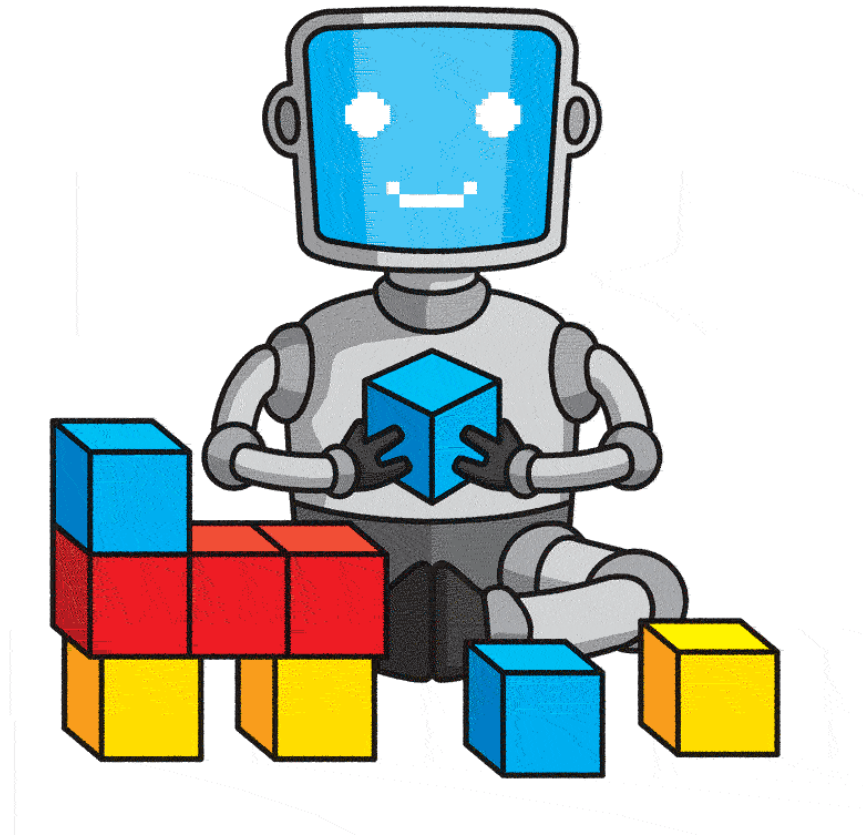
# IA vs ML vs DL



<http://www.cs.us.es/~fsancho/?p=deep-learning>

# ¿Qué es Machine Learning?

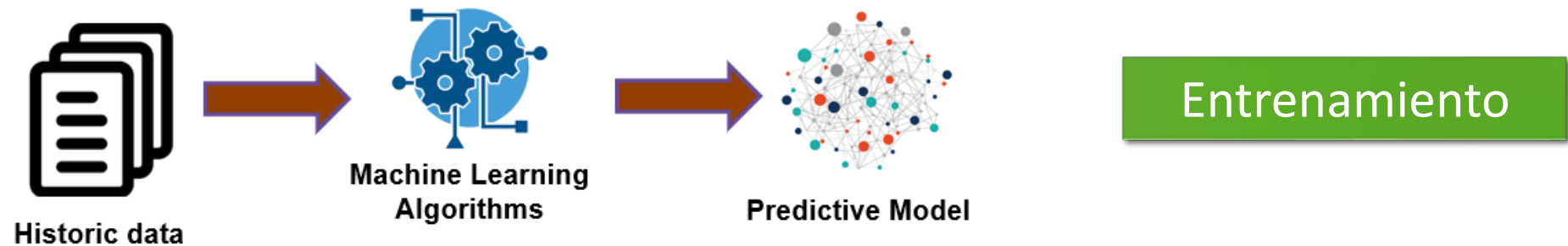
---



# ¿Qué es Machine Learning?

---

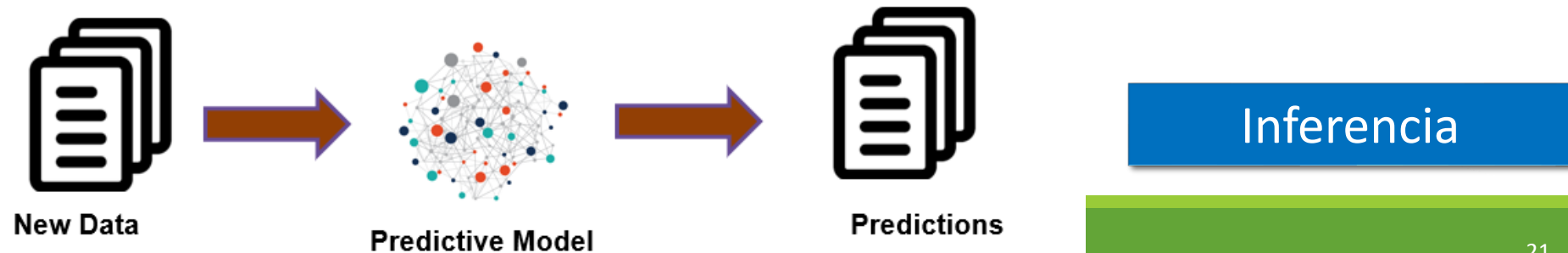
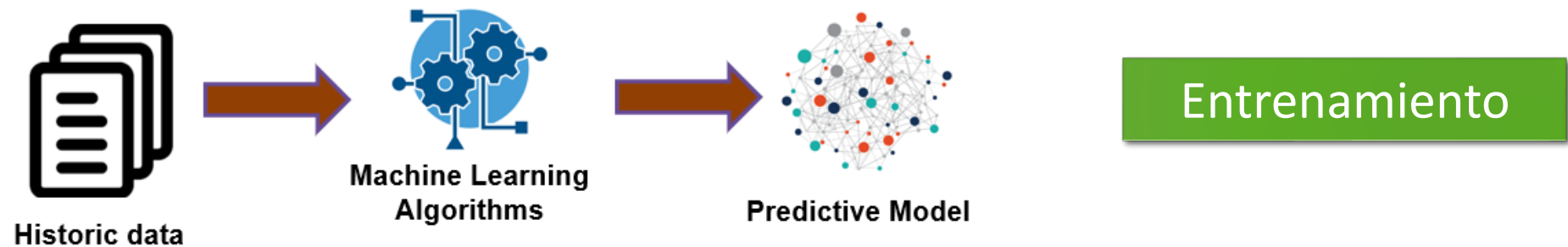
Rama de la **Inteligencia Artificial** cuyo objetivo es conseguir que las computadoras “aprendan” a base de ejemplos (**Learn by example**)



<http://singaporebusinessintelligence.blogspot.com/2018/10/what-is-automated-machine-learning.html>

# ¿Qué es Machine Learning?

Rama de la **Inteligencia Artificial** cuyo objetivo es conseguir que las computadoras “aprendan” a base de ejemplos (**Learn by example**)



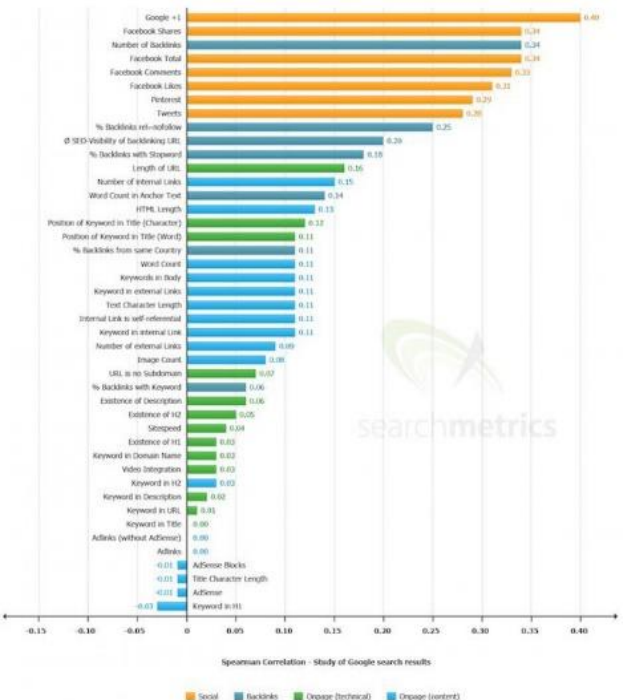
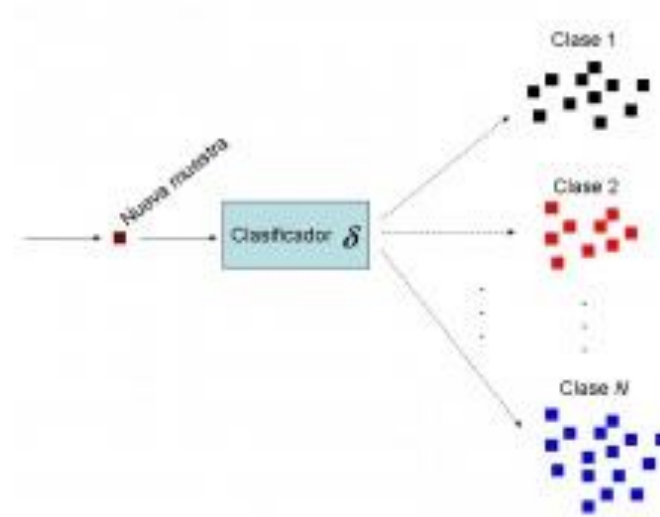
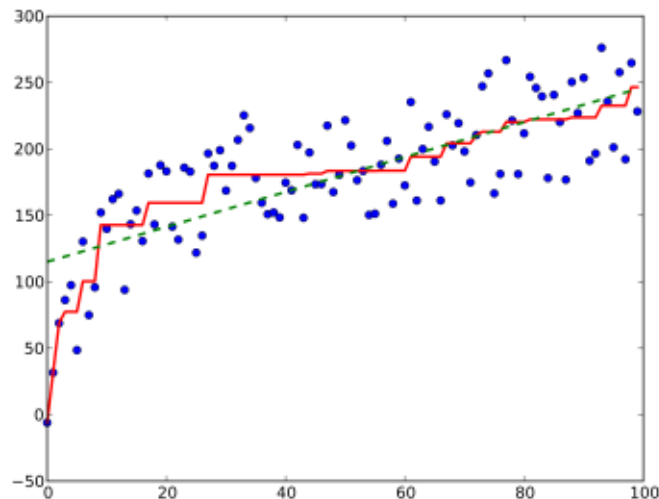
# Tipos de Machine Learning

Según el objetivo a predecir

Regression

Classification

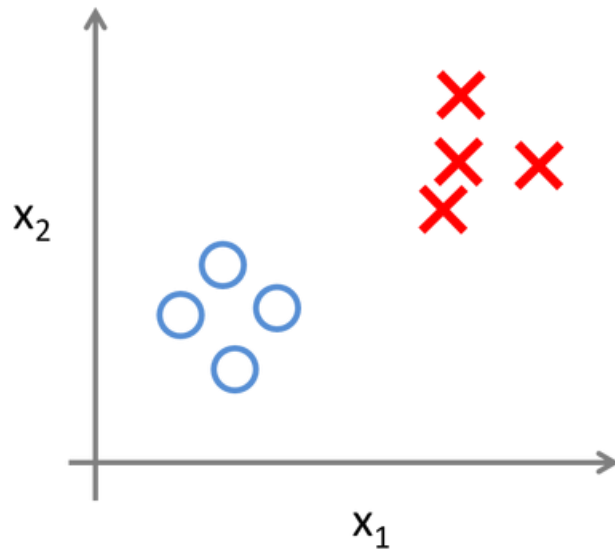
Ranking



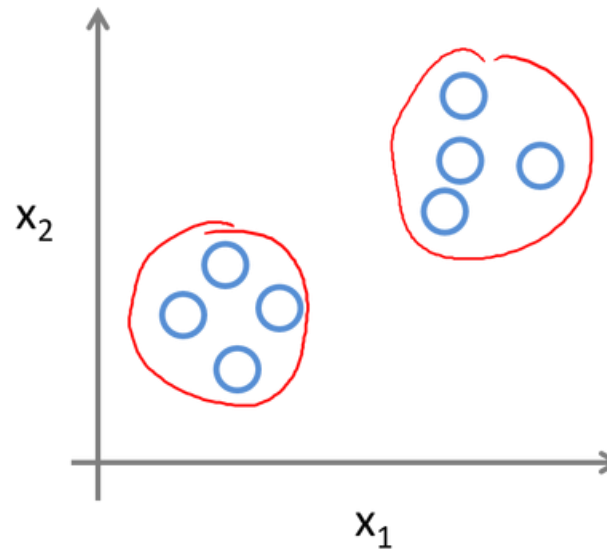
# Tipos de Machine Learning

Según se usan los ejemplos

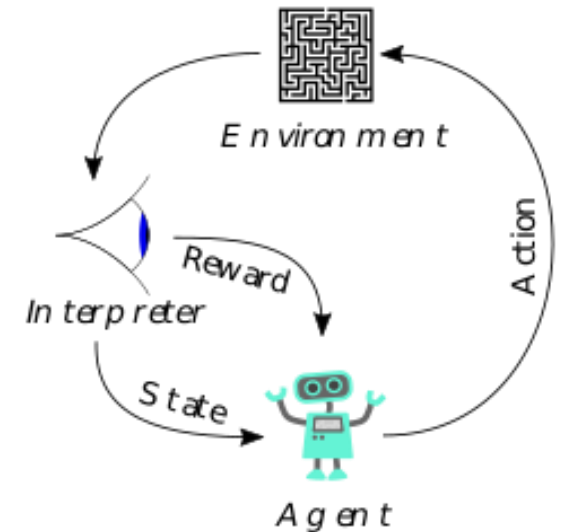
Supervised Learning



Unsupervised Learning



Reinforcement Learning

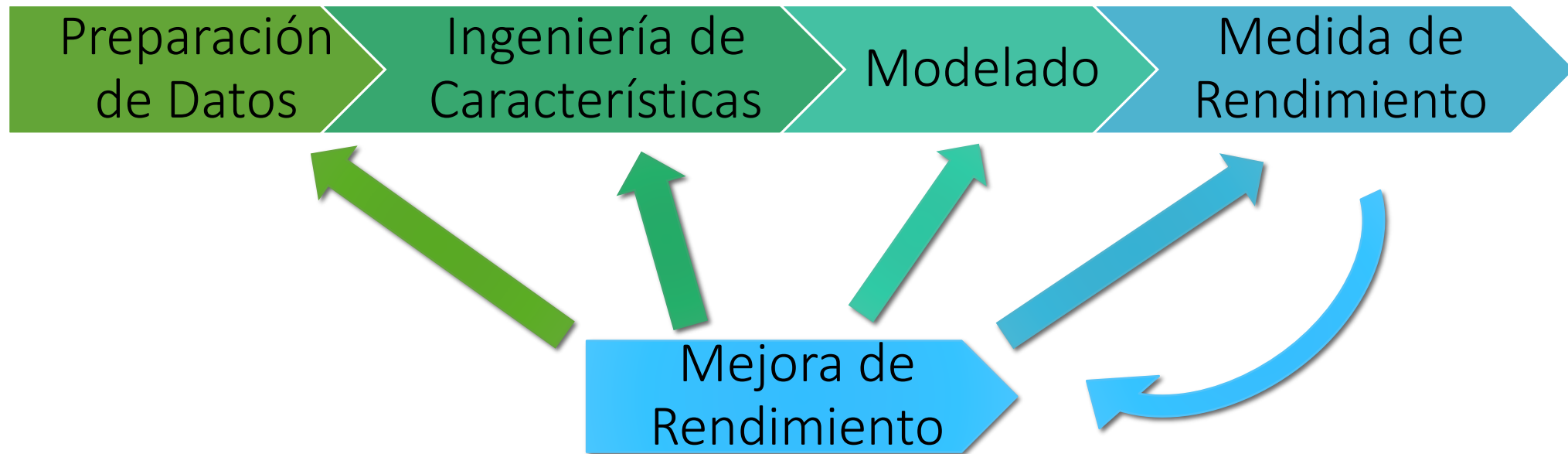


<https://lakshaysuri.wordpress.com/2017/03/19/machine-learning-supervised-vs-unsupervised-learning/>

# Metodología por pasos

---

Hay 5 pasos básicos para construir un modelo ML:

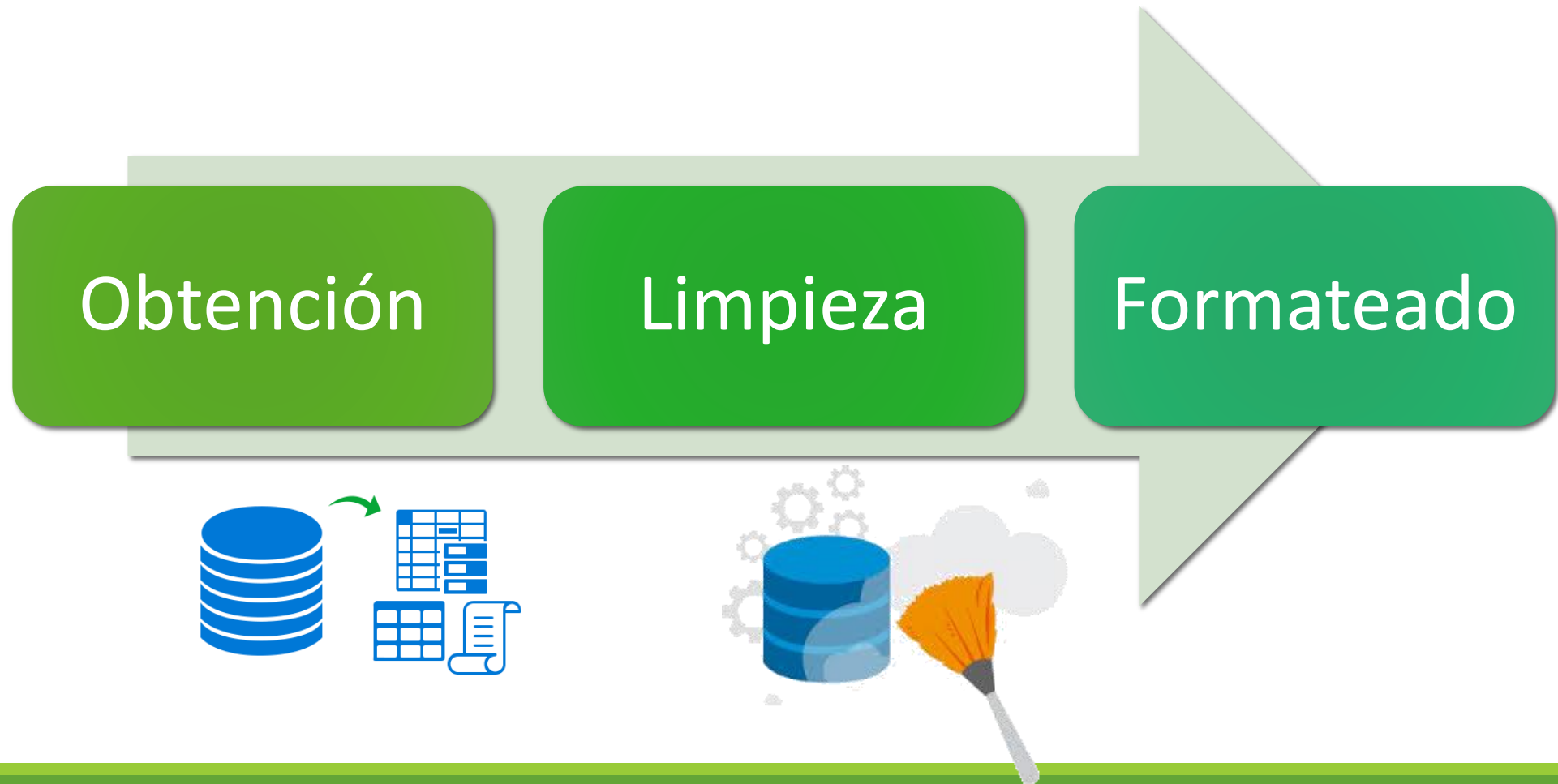


Aunque es un proceso altamente iterativo, que debe repetirse hasta encontrar resultados satisfactorios...



# Paso 1. Preparación de Datos

---



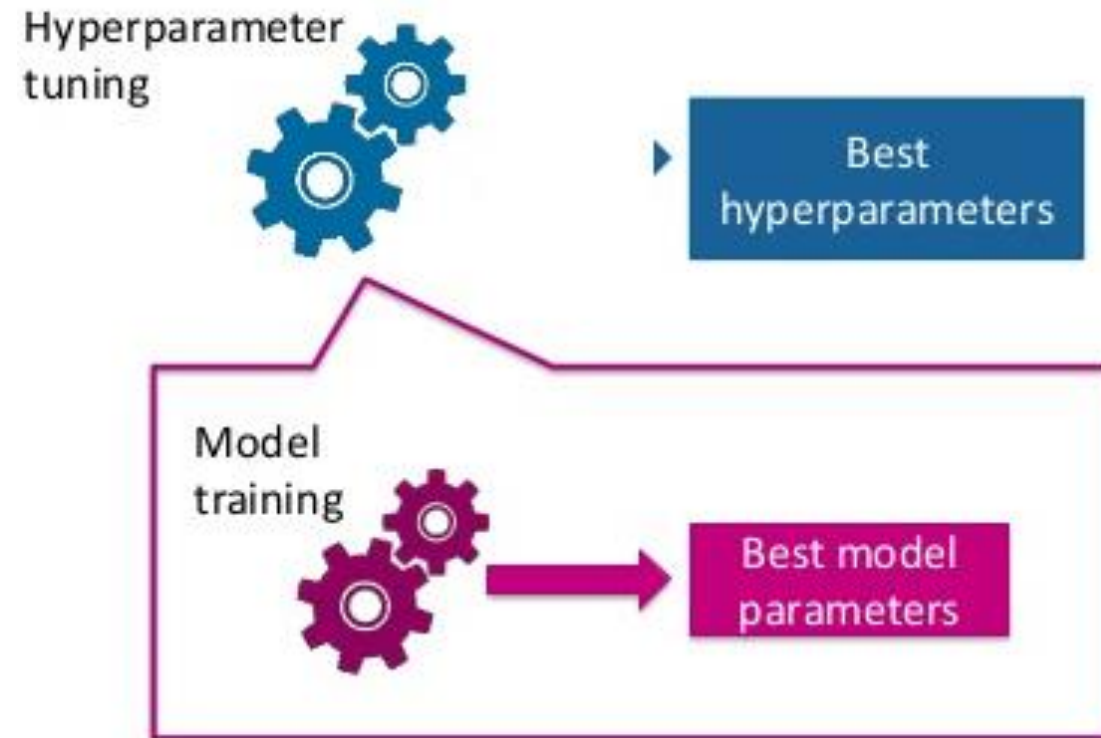
# Paso 2. Ingeniería de Características

- Una **característica (feature)** es una propiedad individual medible del fenómeno/problema que está siendo analizado, y que será usado para formar predicciones.
  - imágenes: píxeles
  - coches autónomos: datos cámaras, sensores, GPS...
- El número de características se llama **dimensión**.

	iris	sepal length	sepal width	petal length	petal width
111	Iris-virginica	6.500	3.200	5.100	2.000
117	Iris-virginica	6.500	3.000	5.500	1.800
148	Iris-virginica	6.500	3.000	5.200	2.000
59	Iris-versicolor	6.600	2.900	4.600	1.300
76	Iris-versicolor	6.600	3.000	4.400	1.400
66	Iris-versicolor	6.700	3.100	4.400	1.400
78	Iris-versicolor	6.700	3.000	5.000	1.700
87	Iris-versicolor	6.700	3.100	4.700	1.500
109	Iris-virginica	6.700	2.500	5.800	1.800
125	Iris-virginica	6.700	3.300	5.700	2.100
141	Iris-virginica	6.700	3.100	5.600	2.400
145	Iris-virginica	6.700	3.300	5.700	2.500
146	Iris-virginica	6.700	3.000	5.200	2.300
77	Iris-versicolor	6.800	2.800	4.800	1.400
113	Iris-virginica	6.800	3.000	5.500	2.100
144	Iris-virginica	6.800	3.200	5.900	2.300
53	Iris-versicolor	6.900	3.100	4.900	1.500
121	Iris-virginica	6.900	3.200	5.700	2.300

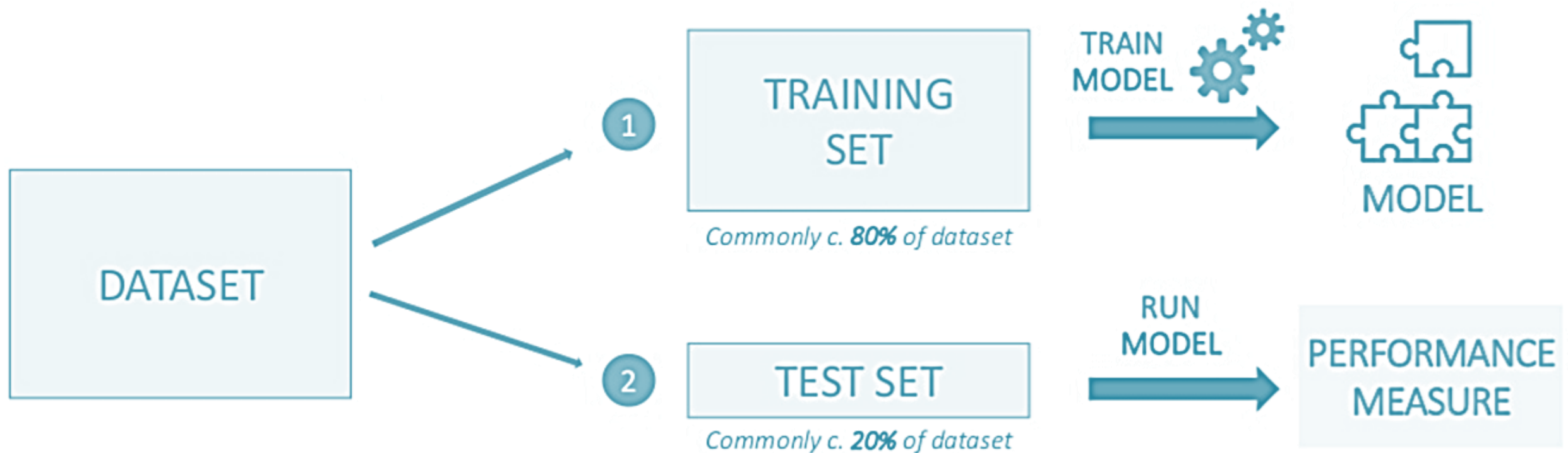
# Paso 3. Modelado

- Hay que elegir un tipo de modelo:
  - **Paramétrico**: El modelo resume los datos con un conjunto de parámetros (p.ej. Regresión lineal, **redes neuronales**, ...)
  - **No paramétrico**: El modelo representa los datos sin parámetros, basado directamente en información de los ejemplos (p.ej. Árbol de decisión, KNN...)
- No confundir un parámetro del modelo con un **hiperparámetro**: parámetro que se emplea para ajustar el entrenamiento del modelo



<https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568>

# Paso 4. Medida del Rendimiento



# Paso 4. Medida del Rendimiento

Por ejemplo, en clasificación binaria:

Predicción:  
(¿es gato?)



Imagen:



**True  
Positive  
(TP)**

**True  
Negative  
(TN)**

**False  
Negative  
(FN)**

**False  
Positive  
(FP)**

Images from the STL-10 dataset

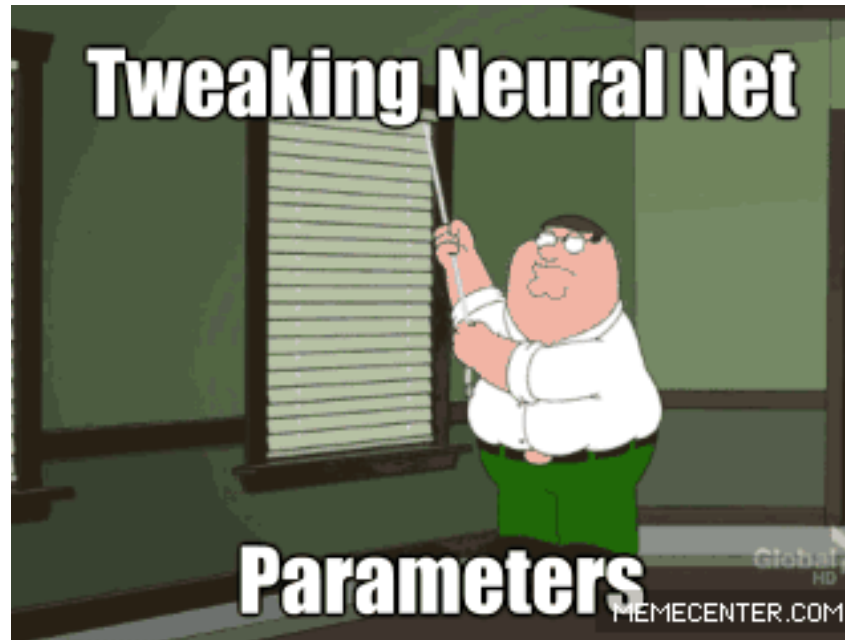
$$\text{Accuracy} = \frac{TP + TN}{TOTAL}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

# Paso 5. Mejora de Rendimiento

---

Al final, todo se reduce a un proceso de mejora continuado



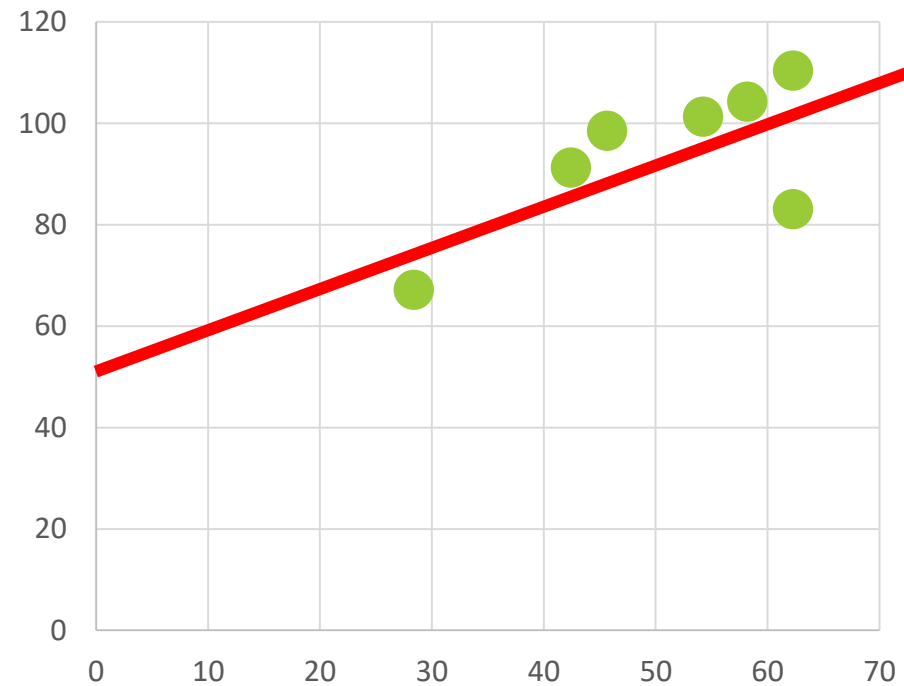
# Índice

---

1. Motivación
2. Introducción al Machine Learning
3. Redes neuronales multicapa
4. Optimización de redes neuronales
5. Entornos software para Deep Learning
6. Nuestra primera red con Keras

# Regresión lineal

Tamaño (m²)	Precio (€)
42,45	91241
54,25	101251
32,5	83051
62,3	110341
28,4	67124
45,69	98525
58,2	104251



X

Y

$$y \approx f(x) = Wx + b = 0,8233x + 52,096$$



# Regresión lineal

$x_1$	$x_2$	$x_3$	$x_4$	$y$
Tamaño (m <sup>2</sup> )	Número habitaciones	Número plantas	Años construido	Precio (€)
42,45	2	1	10	91241
54,25	3	2	23	101251
32,5	2	1	5	83051
62,3	4	3	41	110341
28,4	1	1	24	67124

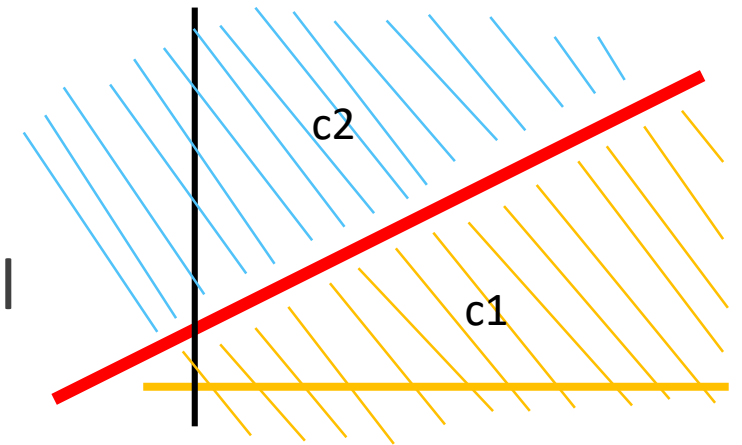
$$y \approx f(x) = Wx^T = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

$$f = 80 + 0,9x_1 + 0,5x_2 + 3x_3 - 2x_4$$

# Perceptrón simple / Regresión logística

- *Frank Rosenblatt, ~1957*
- Clasificación **binaria** (dos clases, 0 y 1).

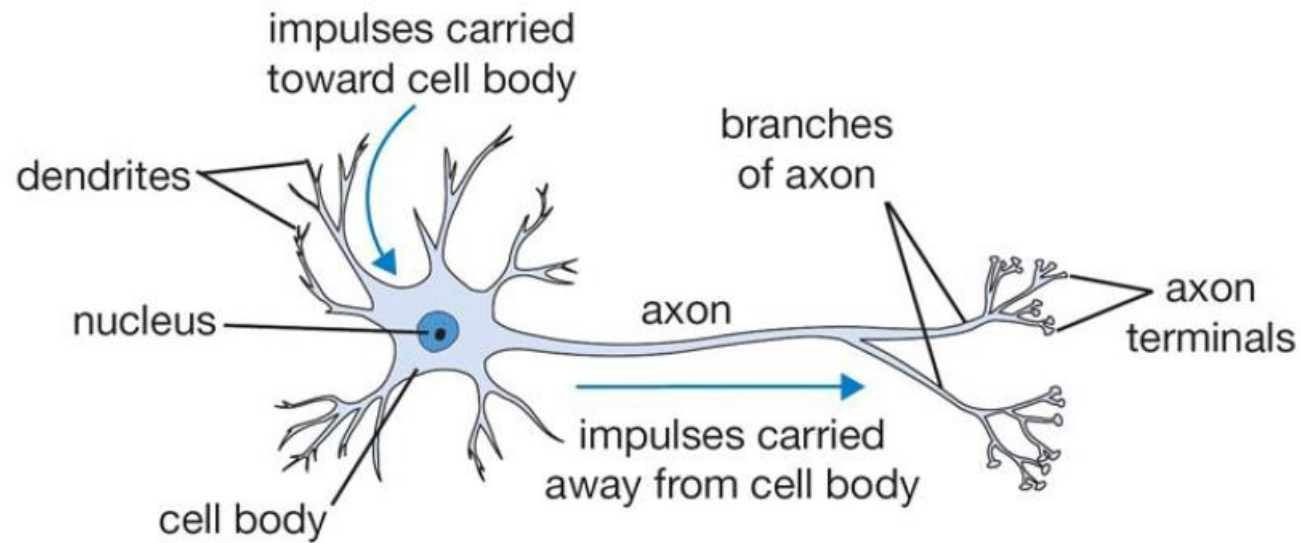
$$y \approx f(Wx^T + b) = \begin{cases} 1, & \text{si } Wx^T + b > \text{valor\_umbral} \\ 0, & \text{en otro caso} \end{cases}$$



- Otras funciones de activación: sigmoide (regresión logística), signo, ...
- Si pensamos en 2 dimensiones, sería partir el plano mediante una recta.

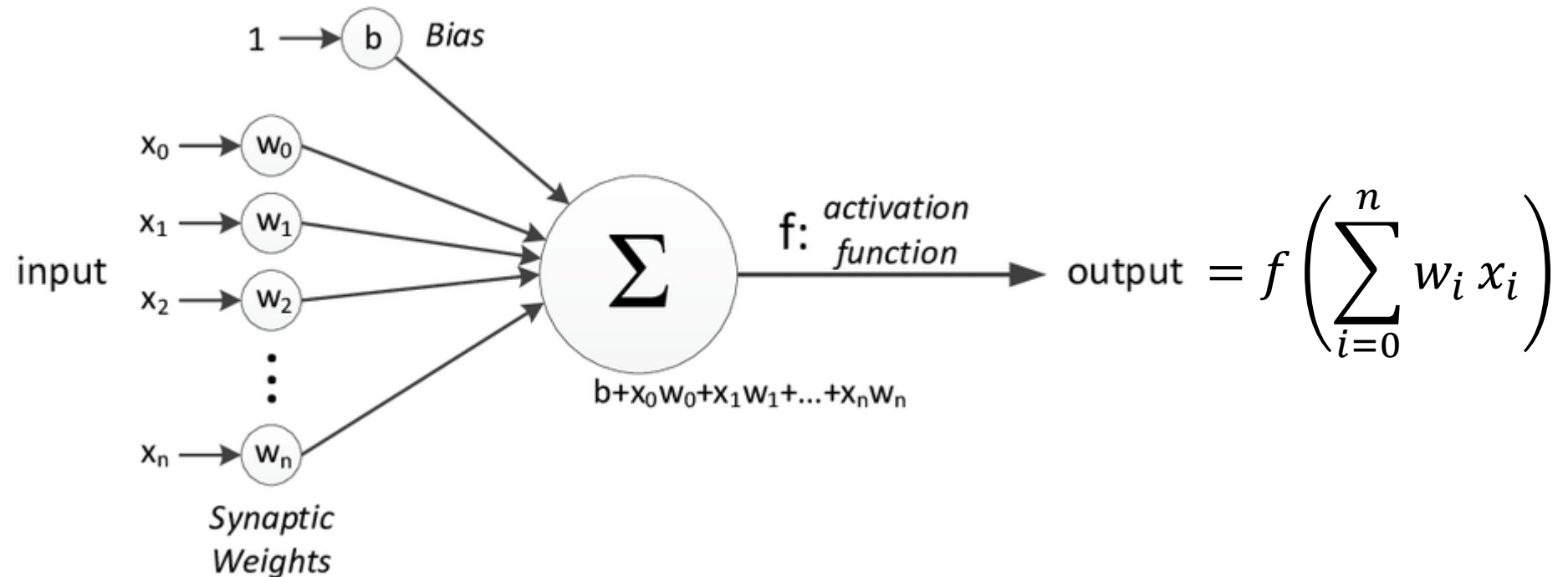
# La neurona artificial

- **Neurona artificial**, 1943, McCulloch y Pitts



# La neurona artificial

- **Neurona artificial**, 1943, McCulloch y Pitts



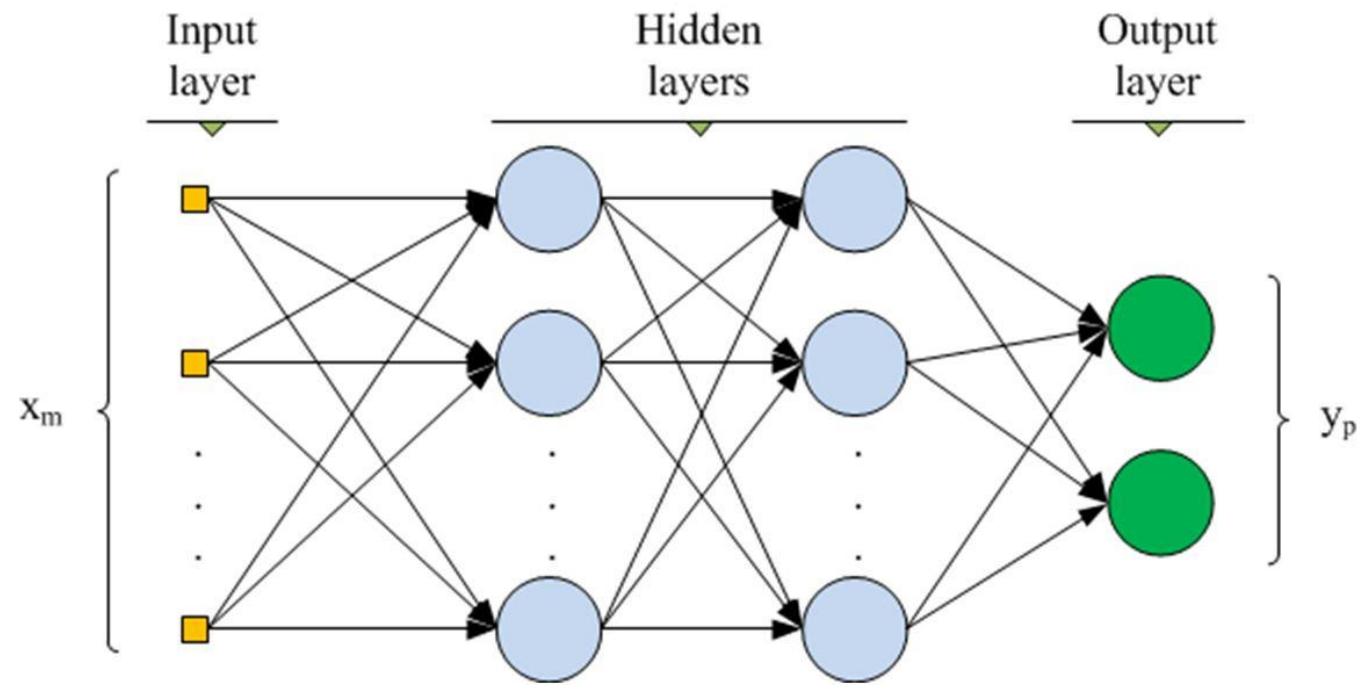
# Redes multicapa

- Organizando **perceptrones simples en múltiples capas (MLP)**:

- Capa de **neuronas** de entrada
- Capa/s de **neuronas** ocultas
- Capa de **neuronas** de salida

- Cada neurona de una capa conectada con todas de la capa anterior (**fully connected**)

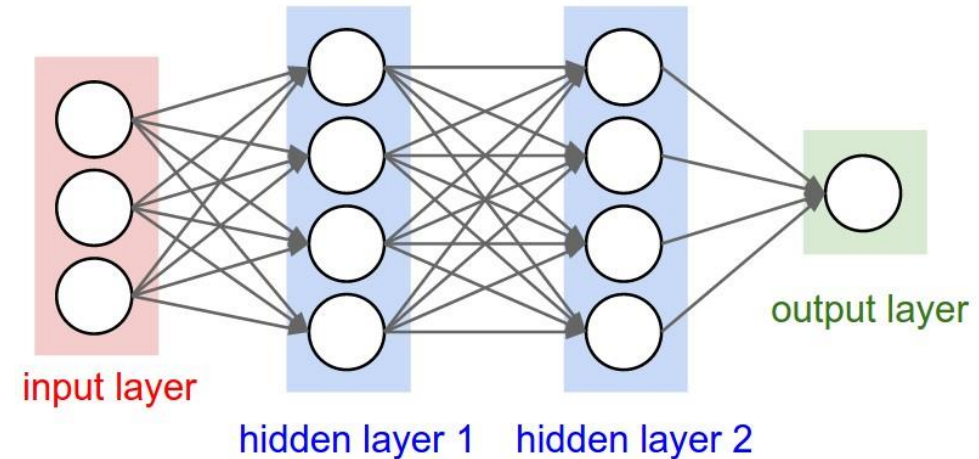
- Capa de **entrada** sin pesos



# Redes multicapa: capa de salida

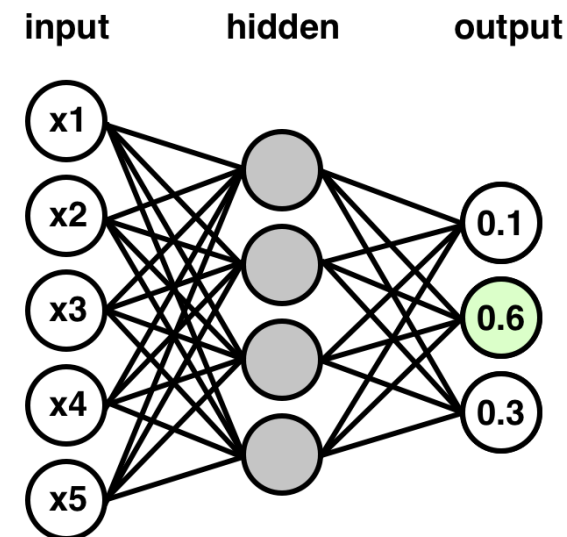
- Clasificación **binaria**:

- $K=2$  clases
- Variable de salida es  $y=0$  o  $1$
- *1 unidad de salida*



- Clasificación **multiclase**:

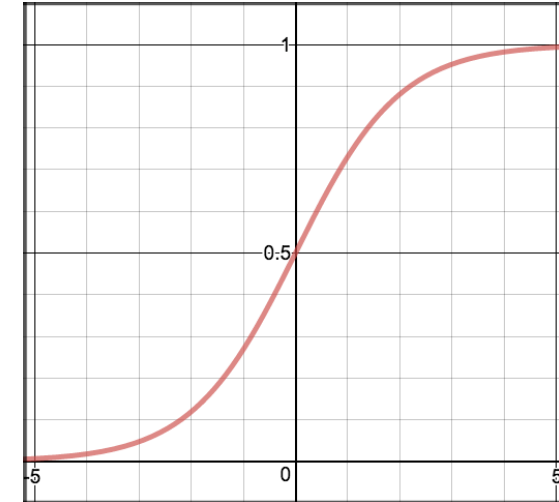
- $K \geq 3$  clases
- $K$  variables de salida,  $y_1 \dots y_k$
- *$K$  unidades de salida*
- Se considera la más alta



# Funciones de activación: capa de salida

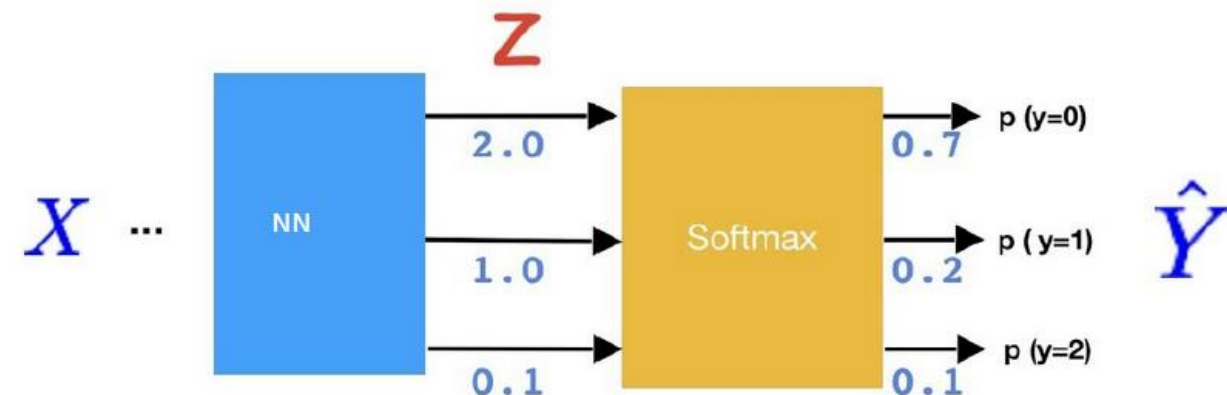
- **Clasificación binaria:**

- La función **sigmoide** o **logística**:  $\sigma(x) = \frac{1}{1+e^{-x}}$

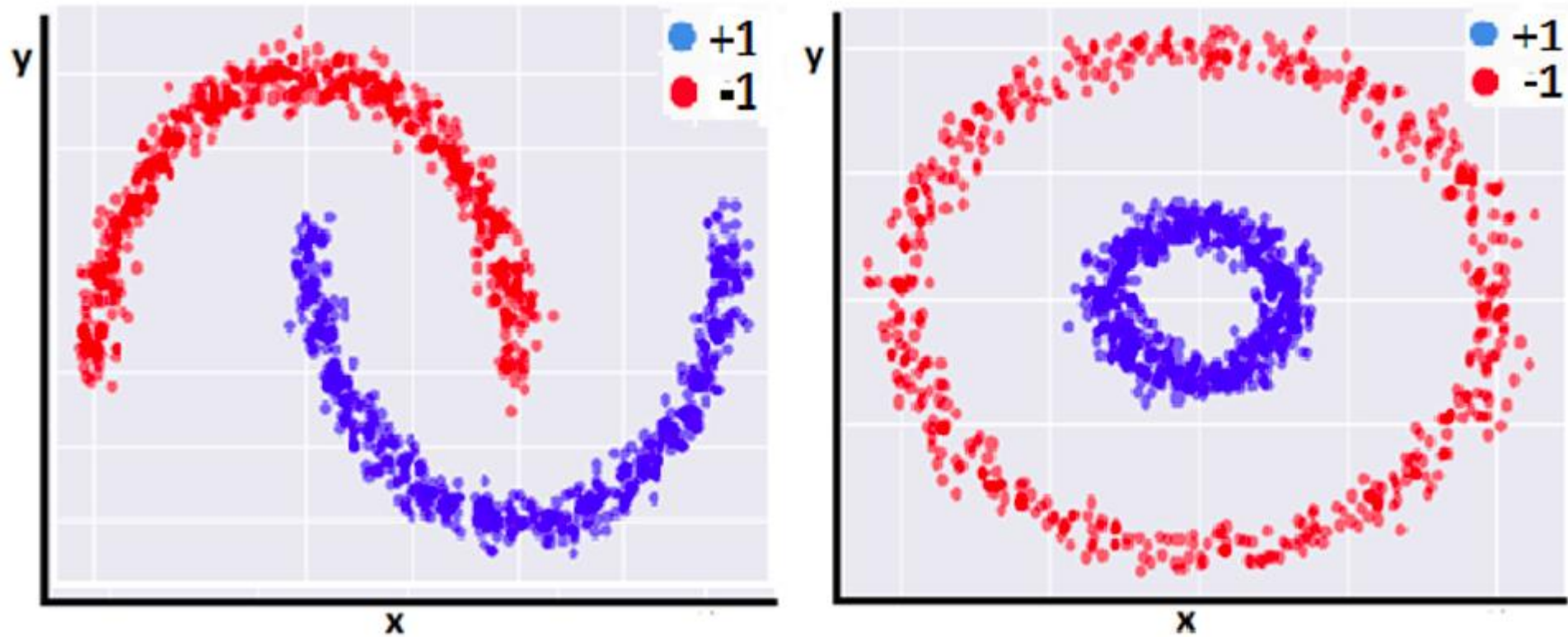


- **Clasificación multiclase:**

- La función **softmax**:  $\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^N e^{z_k}}$



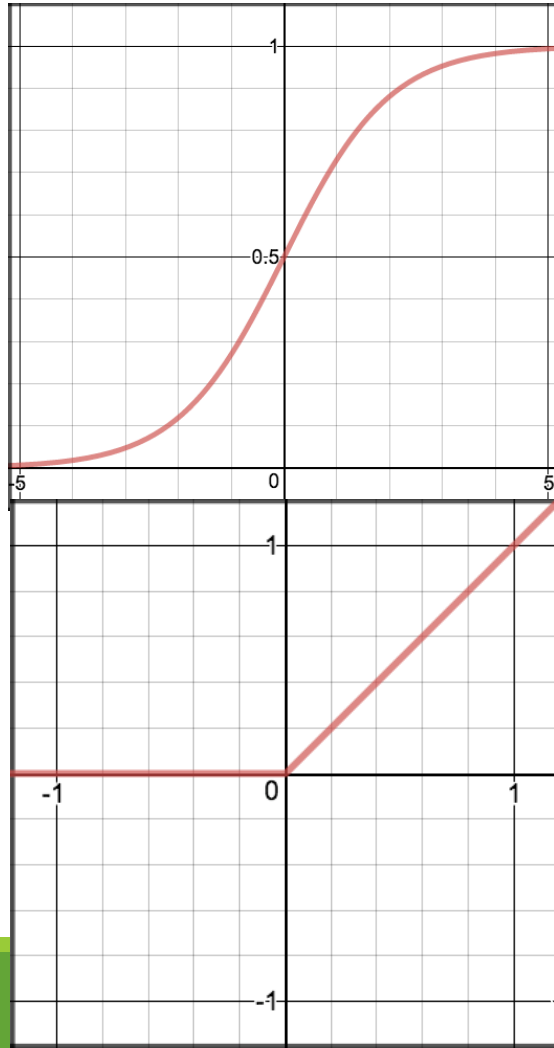
# Funciones de activación: No linealidad





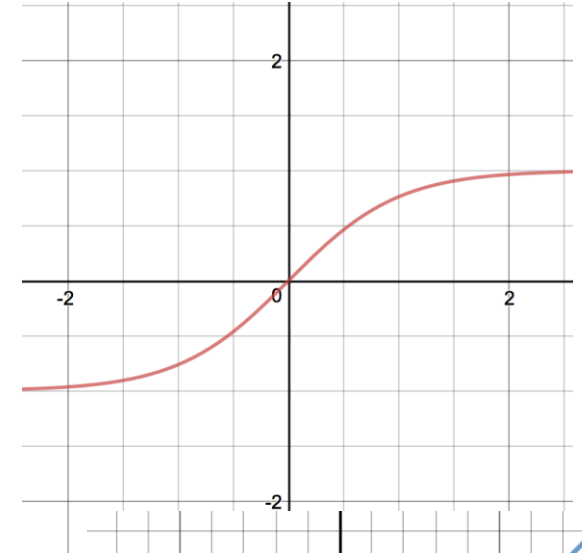
# Funciones de activación: capas ocultas

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

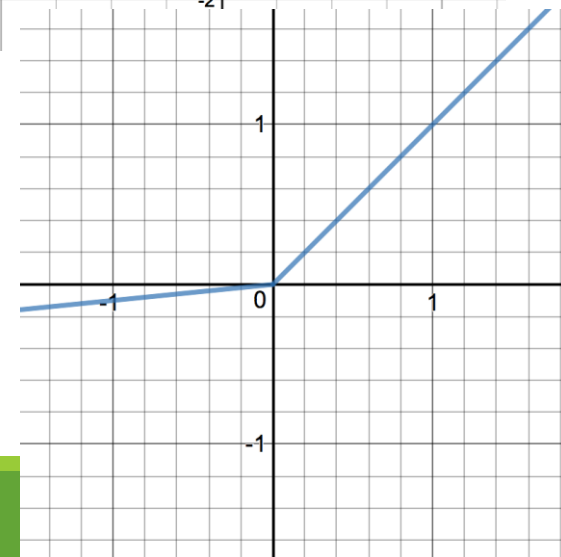


$ReLU(x) = \max(0, x)$

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



$LeakyReLU(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$



# Índice

---

1. Motivación
2. Introducción al Machine Learning
3. Redes neuronales multicapa
4. Optimización de redes neuronales
5. Entornos software para Deep Learning
6. Nuestra primera red con Keras

# Función de coste

---

- Necesitaremos ajustar los parámetros del modelo (pesos  $W$ ) para que se comporte mejor con los datos.
- Por tanto, necesitamos **cuantificar** cuánto de “*buena*” es nuestra red para un ejemplo.
- Definiremos:
  - La **función de pérdida (loss)**: para un ejemplo
  - La **función de coste (cost)**: para un conjunto de ejemplos (dataset, batch)
  - La **función objetivo** a minimizar. La función de coste es una función objetivo.
- El nombre de estas funciones se suelen confundir

# Función de coste

---

- En **regresión** lineal es **MSE** (error cuadrático medio):

$$J(W) = \frac{1}{m} \sum_{i=1}^m (f_W(x^i) - y^i)^2$$

- En regresión logística para **clasificación binaria (binary cross entropy)**:

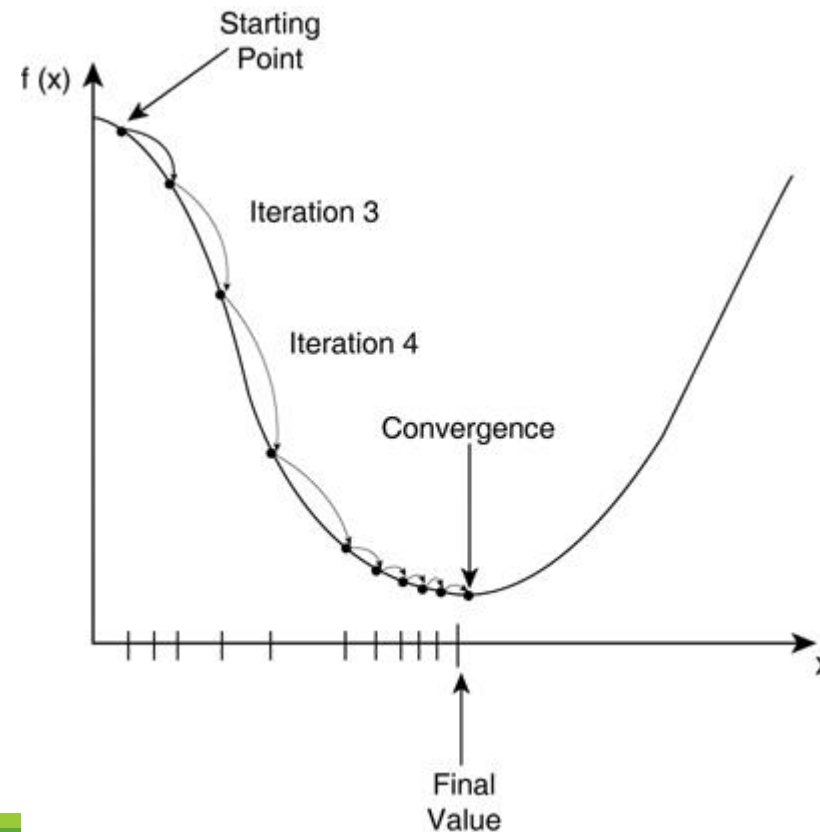
$$J(W) = \frac{1}{m} \left( \sum_{i=1}^m y^i \log(f_W(x^i)) + (1 - y^i) \log(1 - f_W(x^i)) \right)$$

- En regresión logística para **clasificación multiclase (cross entropy)**:

$$J(W) = \frac{1}{m} \left( \sum_{i=1}^m \sum_{k=1}^K y_k^i \log(f_W^k(x^i)) + (1 - y_k^i) \log(1 - f_W^k(x^i)) \right)$$

# Descenso por gradiente

Buscar el mínimo posible de la función de coste, es decir  $\min_W J(W)$



# Descenso por gradiente

---

Buscar el mínimo posible de la función de coste, es decir  $\min_W J(W)$

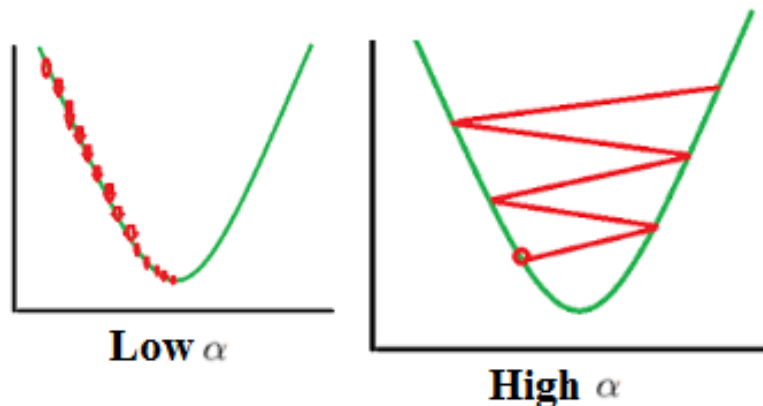


# Descenso por gradiente

- Seguir el **gradiente** en negativo (la mayor pendiente)
- Es decir: calculamos el coste, su derivada, y actualizamos cada parámetro  $w_j$ :

$$w_j = w_j - \alpha \frac{d}{dw_j} J(W) = w_j - \alpha \sum_{i=1}^m (f_W(x^i) - y^i) f'_W(x^i) x_j^i$$

- $\alpha$  es el factor de aprendizaje (un *hiperparámetro*). Hay que ajustarlo bien:

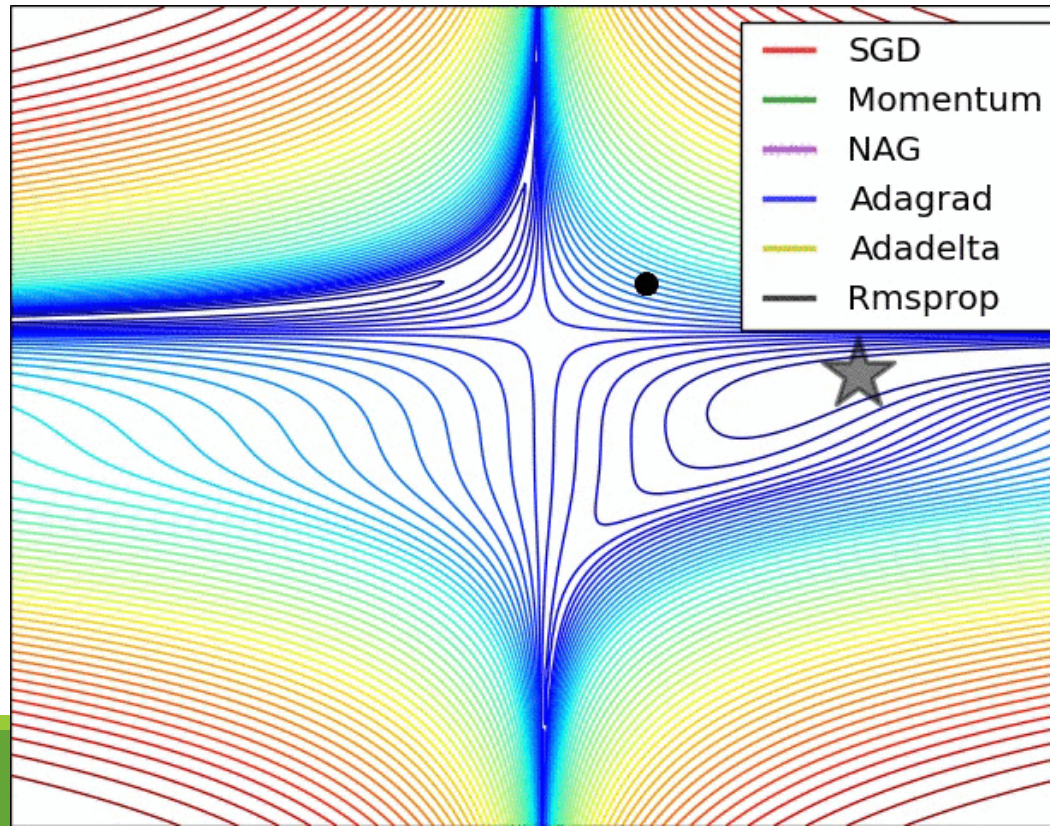




# Descenso por gradiente: métodos

<http://cs231n.github.io/neural-networks-3/>

- Métodos varían según actualización de parámetros
- Uso de mini-batch (**batch**)
- **Adam**

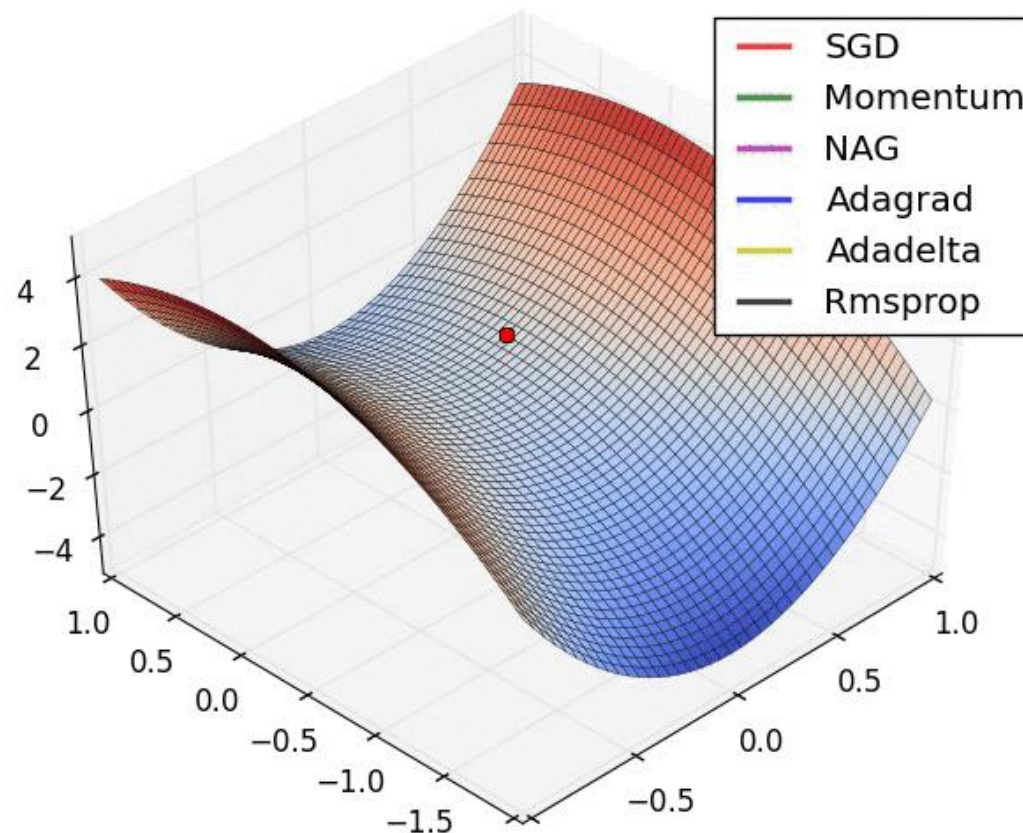




# Descenso por gradiente: métodos

<http://cs231n.github.io/neural-networks-3/>

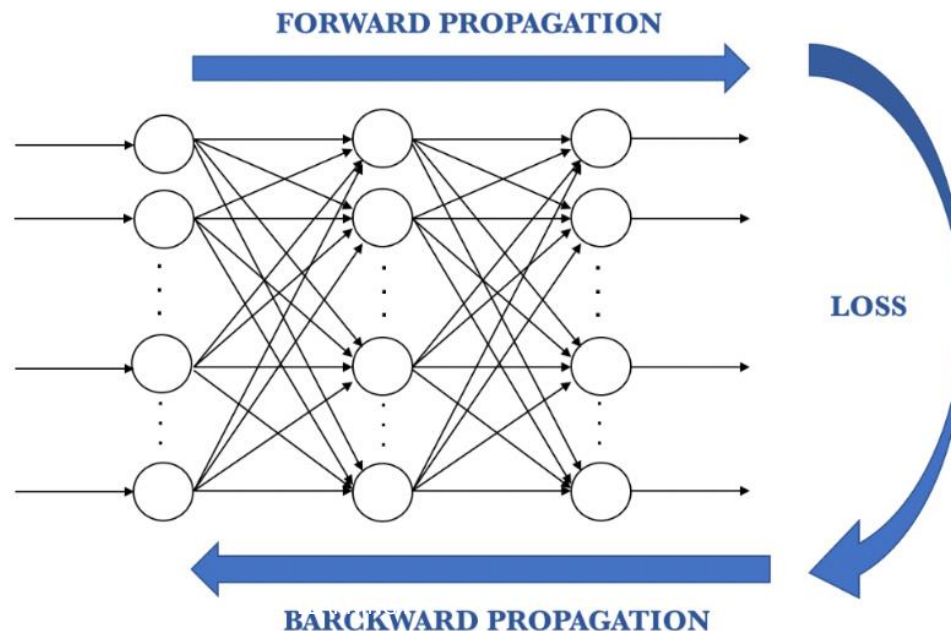
- Métodos varían según actualización de parámetros
- Uso de mini-batch (**batch**)
- **Adam**



# Retropropagación

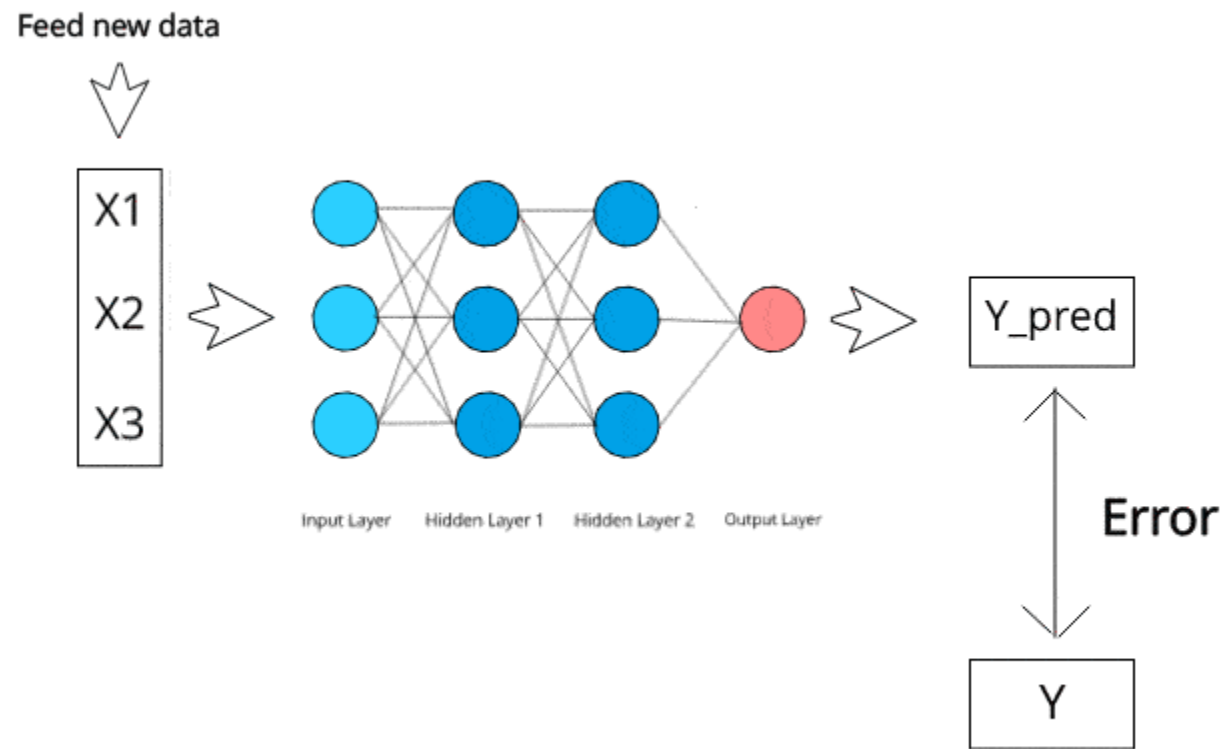
<https://www.kdnuggets.com/2019/10/introduction-artificial-neural-networks.html>

- Para entrenar una red, hacemos una iteración (**época**) sobre el dataset:
  1. Pasarle una serie de ejemplos y calcular sus salidas
  2. Calcular el valor de la función de coste
  3. Calcular los errores y los gradientes en la capa de salida
  4. Propagar los errores y gradientes hacia atrás (la capa de entrada)
  5. Actualizar los pesos de la red



# Retropropagación

<https://www.kdnuggets.com/2019/10/introduction-artificial-neural-networks.html>



# Demo

---

Comprobemos la potencia representacional de una red con <https://playground.tensorflow.org>

# Índice

---

1. Motivación
2. Introducción al Machine Learning
3. Redes neuronales multicapa
4. Optimización de redes neuronales
5. Entornos software para Deep Learning
6. Nuestra primera red con Keras

# Niveles de programación

- **Programación a nivel 0:**

- Podemos elegir el lenguaje de nuestra elección, una buena tarde e implementar los conceptos.
- ¿Qué pasa si quiero cambiar la arquitectura de la red? Si no lo he hecho bien, tendría que re-programarla desde cero, sobre todo para ajustar la propagación del gradiente.

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import expit

def sigmoid(eval):
    return expit(eval)

def Neural_Training(Y01, Labels01, eta, Epochs):

    d, samplenumb = Y01.shape

    # Random [-1,1] init from Haykin
    WIH = 2*np.mat(np.random.rand(2*d,d)) - 1.0
    WHO = 2*np.mat(np.random.rand(1,2*d)) - 1.0
    diff1 = Labels01.astype(np.float64)

    for i in xrange(1, Epochs):

        #Get the input to the output layer
        y_j_temp = sigmoid(WIH*Y01)
        netk = WHO*y_j_temp
        zk = sigmoid(netk)

        # Creating Delta Wk
        diff1 = diff1 - zk
        tDeltaWk = eta*np.multiply(diff1, np.multiply(sigmoid(netk), 1.0-sigmoid(netk)))
        tDeltaWk = np.tile(tDeltaWk, (2*d,1))
        DeltaWk = np.multiply(y_j_temp, tDeltaWk)
        DeltaWk = np.transpose(np.sum(DeltaWk, 1))

        # New Weights
        WHO = WHO + DeltaWk

        #Creating Delta Wj
        dnetj = np.multiply(y_j_temp, 1.0-y_j_temp)
        tprodsumk = np.multiply(np.transpose(DeltaWk), np.transpose(WHO))
        tprodsumk = np.tile(tprodsumk, (1, samplenumb))
        tprodsumk = eta*np.multiply(tprodsumk, dnetj)
        DeltaWj = tprodsumk * np.transpose(Y01)

        # New Weights
        WIH = WIH + DeltaWj

    return WIH, WHO

# Number of samples
N= 60000

#Number of Epochs
Epochs = 20

#Learning Rate
eta = 0.001

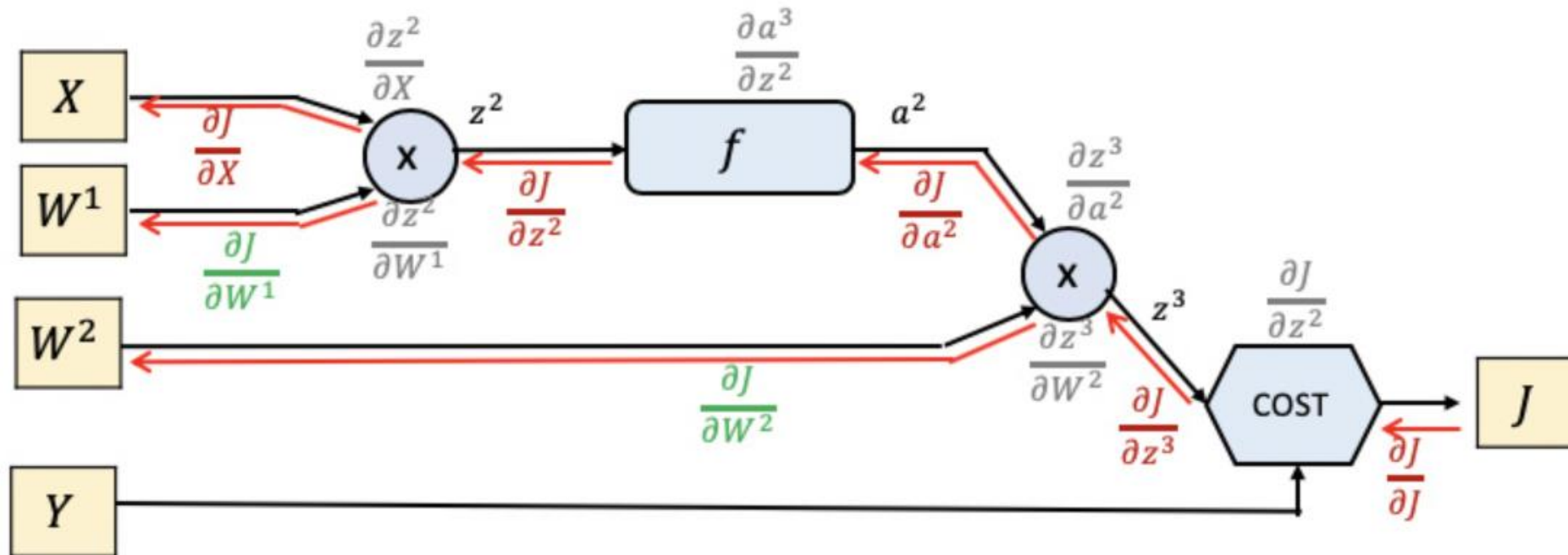
# opening images for [r]eading as [b]inary
in_file = open("train-images.idx3-ubyte", "rb")
in_file.read(16)
Data = in_file.read()
in_file.close()

# Transform the data stream
X = np.fromstring(Data, dtype=np.uint8)
X = X.astype(np.float64)
X = np.mat(X)
```

# Niveles de programación

- **Programación a nivel 1:**

- APIs con bloques reutilizables y **diferenciación automática**.
- **TensorFlow y PyTorch**



# Niveles de programación

---

- **Programación a nivel 2:**

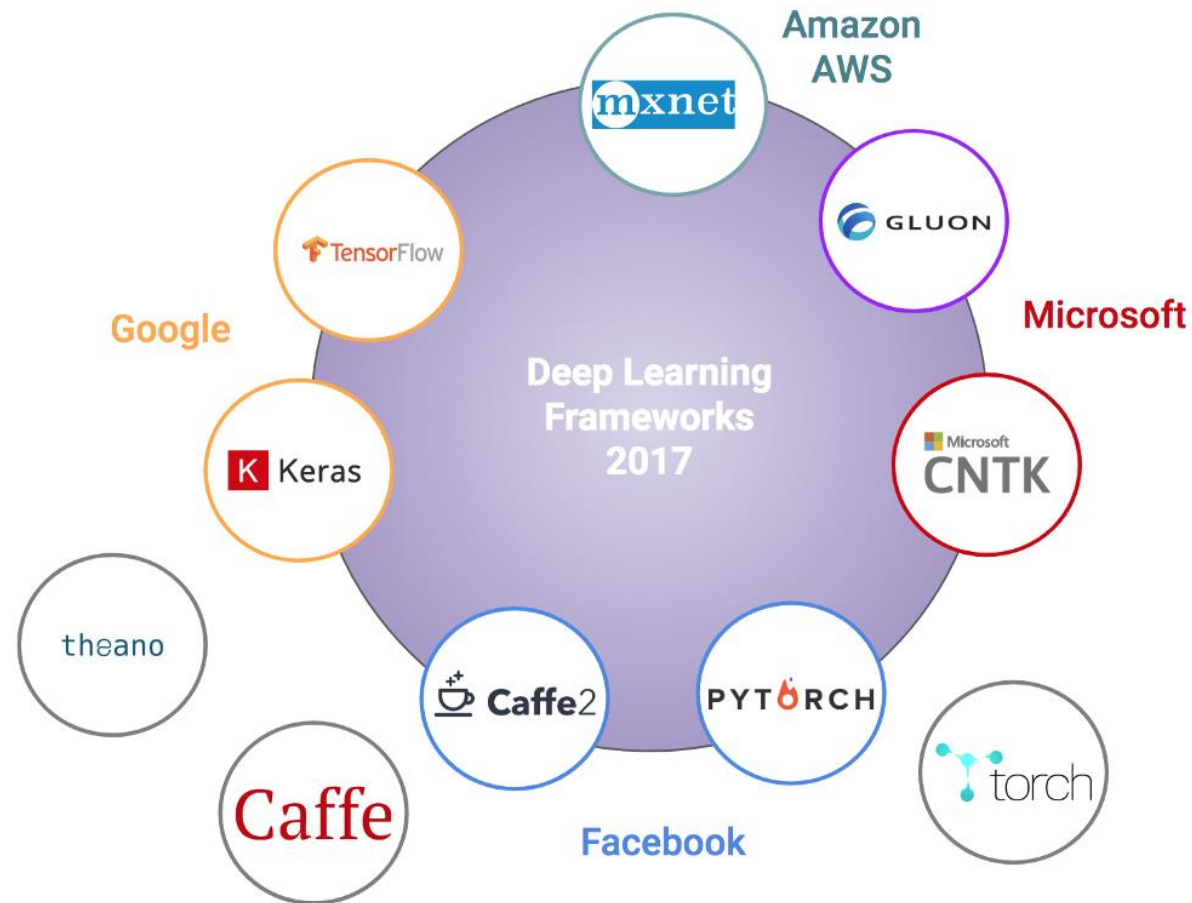
- APIs que proveen una capa de abstracción sobre los detalles de modelo.
- **Encajamos bloques** que representan partes de la arquitectura que queremos montar.
- **Keras, Caffe, Fast.ai**





# Ecosistema actual

---



# Índice

---

1. Motivación
2. Introducción al Machine Learning
3. Redes neuronales multicapa
4. Optimización de redes neuronales
5. Entornos software para Deep Learning
6. Nuestra primera red con Keras