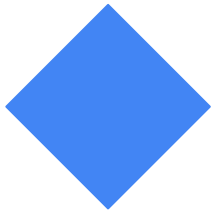
Developer Student Clubs

Universidad Politécnica de Valencia



Developer Student Clubs

Universidad Politécnica de Valencia

INTRODUCCIÓN A MACHINE LEARNING

Ponentes:
Vladyslav Mazurkevych
Eric Marti Haynes

- Datos de la historia.
- Breve explicación de algunos algoritmos.
- Aplicación del Machine Learning.

Developer Student Clubs
Universidad Politécnica de Valencia

Cómo definirías la Inteligencia Artificial?

Developer Student Clubs
Universidad Politécnica de Valencia

Pues... depende!

- **Sistemas que piensan como humanos:** automatizan actividades como la toma de decisiones, aprendizaje (*redes neuronales*).
- **Sistemas que actúan como humanos:** realizan tareas de forma similar a como lo hacen las personas (*robots*).
- **Sistemas que piensan racionalmente:** intentan emular el pensamiento lógico racional de los humanos, razonar.. (*sistemas expertos*).
- **Sistemas que actúan racionalmente:** trata de imitar de manera racional el comportamiento humano (*agentes inteligentes*).

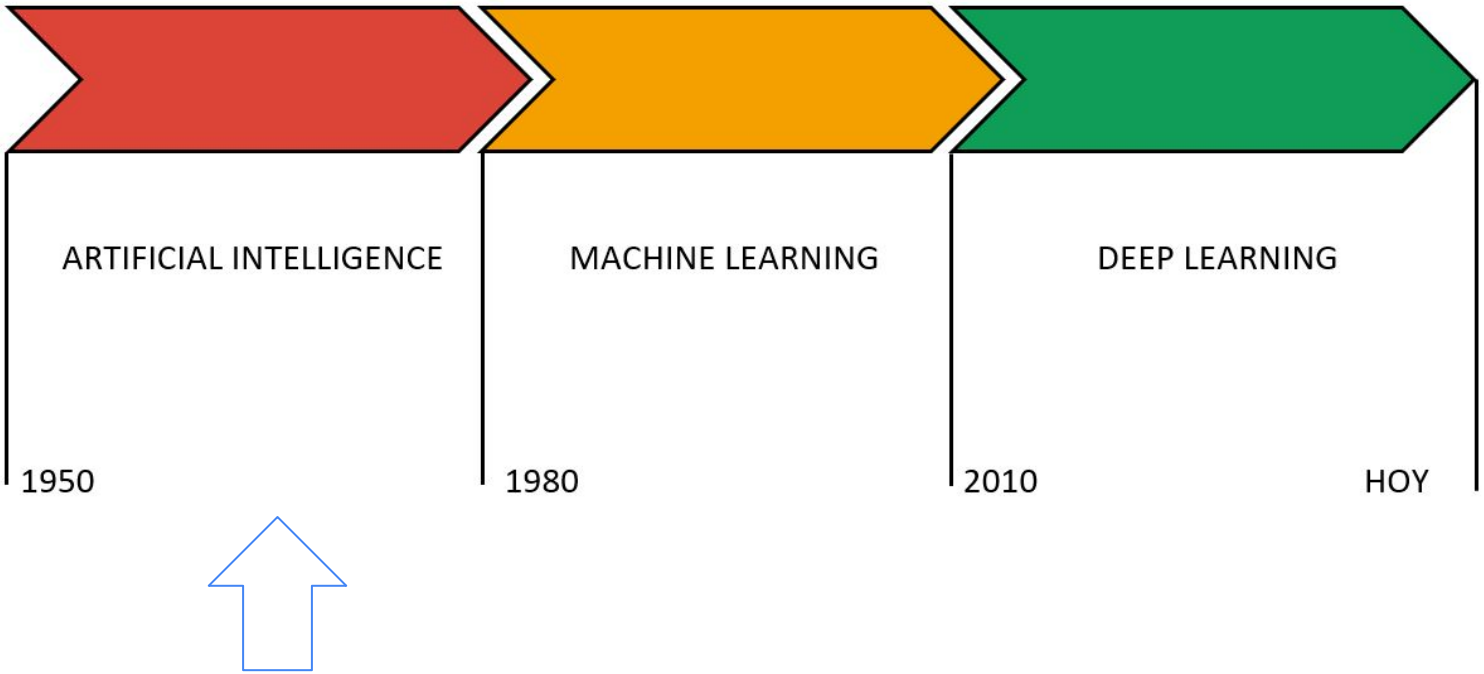
Developer Student Clubs
Universidad Politécnica de Valencia

Una posible definición general.

- La inteligencia artificial es una disciplina que intenta emular mediante máquinas ciertas funciones cognitivas humanas

Developer Student Clubs

Universidad Politécnica de Valencia



Developer Student Clubs
Universidad Politécnica de Valencia

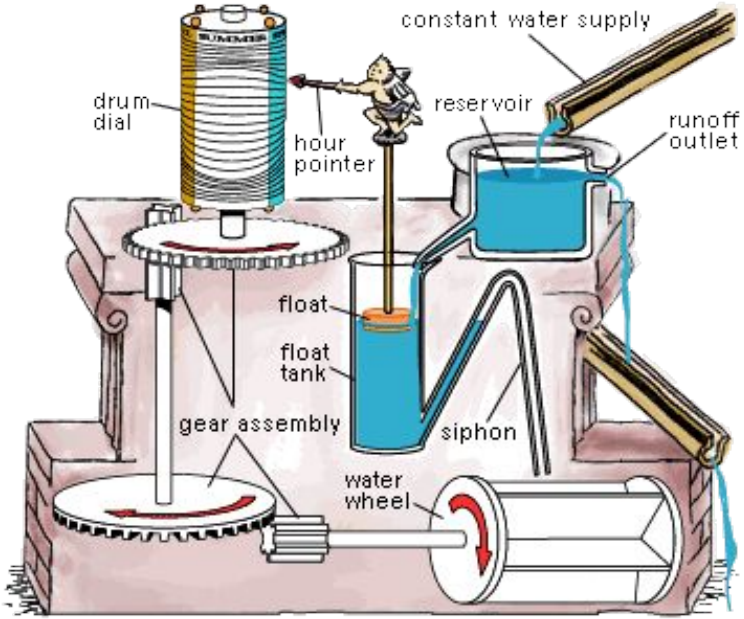
Comienzos de la Inteligencia Artificial

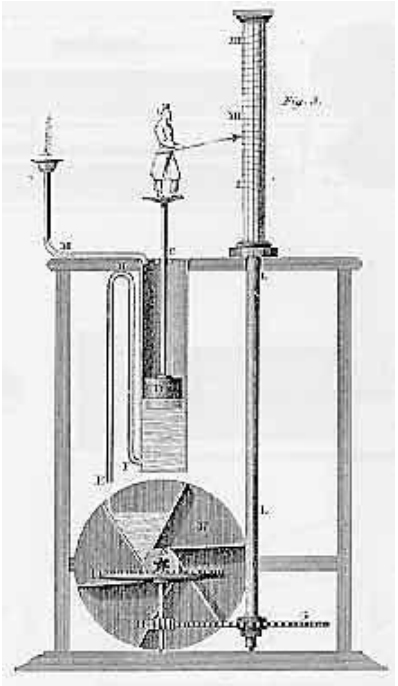
- **Aristóteles** (384-322 a.C): formula los *silogismos*, los cuales intentan establecer la relación entre un sujeto y un predicado.
- **Ctesibio de Alejandría** (250 a.C): crea un reloj hidráulico autocontrolado.
- **Alan Turing** (1936): introduce el concepto de la *Máquina de Turing*.
- **Alan Turing** (1950): propone el *Test de Turing*.

Developer Student Clubs
Universidad Politécnica de Valencia

Comienzos de la Inteligencia Artificial

- Reloj hidráulico autocontrolado de Ctesibio:

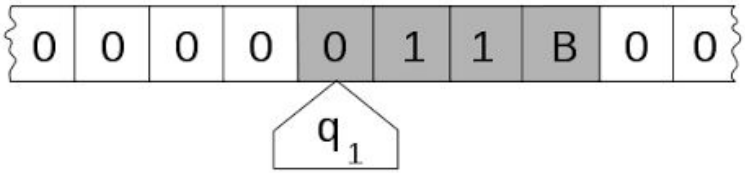




Developer Student Clubs
Universidad Politécnica de Valencia

Comienzos de la Inteligencia Artificial

- **Máquina de Turing:** contiene un conjunto de instrucciones detalladas y precisas que lleva a un fin concreto.

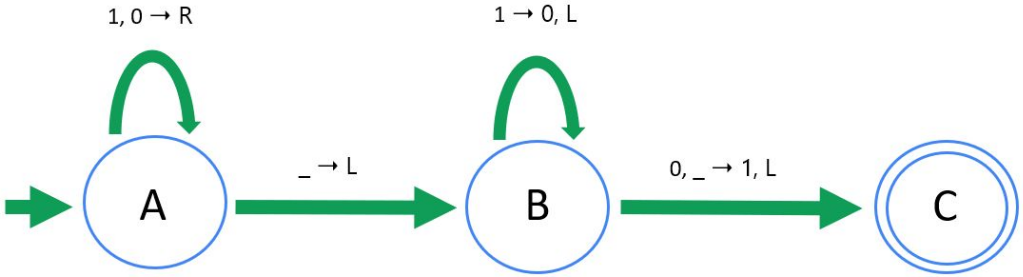


1

0	1	R	2
1	1	L	1

2

0	1	R	2
1	0	R	2



Developer Student Clubs
Universidad Politécnica de Valencia

Comienzos de la Inteligencia Artificial

- Máquina de Turing:

```
iteration = 0
state = 'A'
position = 0
data = ['0','1','1','1','1','1','']
#output of first data
print(f"Input data: ", data)

#algorithm
print(f"⋙State: ",state, f"⋙Position: ", position, f"⋙Data state: ", data)

def stateA():
    global position
    global state
    if data[position] == '1' or data[position] == '0':
        # same state
        position += 1                       # move right

    elif data[position] == '':
        state = 'B'                         # change state to B
        position -= 1                       # move left
    print(f"⋙State: ",state, f"⋙Position: ", position, f"⋙Data state: ", data)
```

```
def stateB():
    global position
    global state
    if data[position] == '1':
        # same state
        data[position] = '0'                # change data to 0
        position -= 1                       # move left
    elif data[position] == '0' or data[position] == '':
        state = 'C'                         # change state to C
        data[position] = '1'                # change data to 1
        position -= 1                       # move
    print(f"⋙State: ",state, f"⋙Position: ", position, f"⋙Data state: ", data)

def stateC():
    print(f"Done! ⋙Output: " , data)


while True:
    if state == 'A':
        stateA()
    elif state == 'B':
        stateB()
    else:
        stateC()
        break
```

Developer Student Clubs
Universidad Politécnica de Valencia

Comienzos de la Inteligencia Artificial

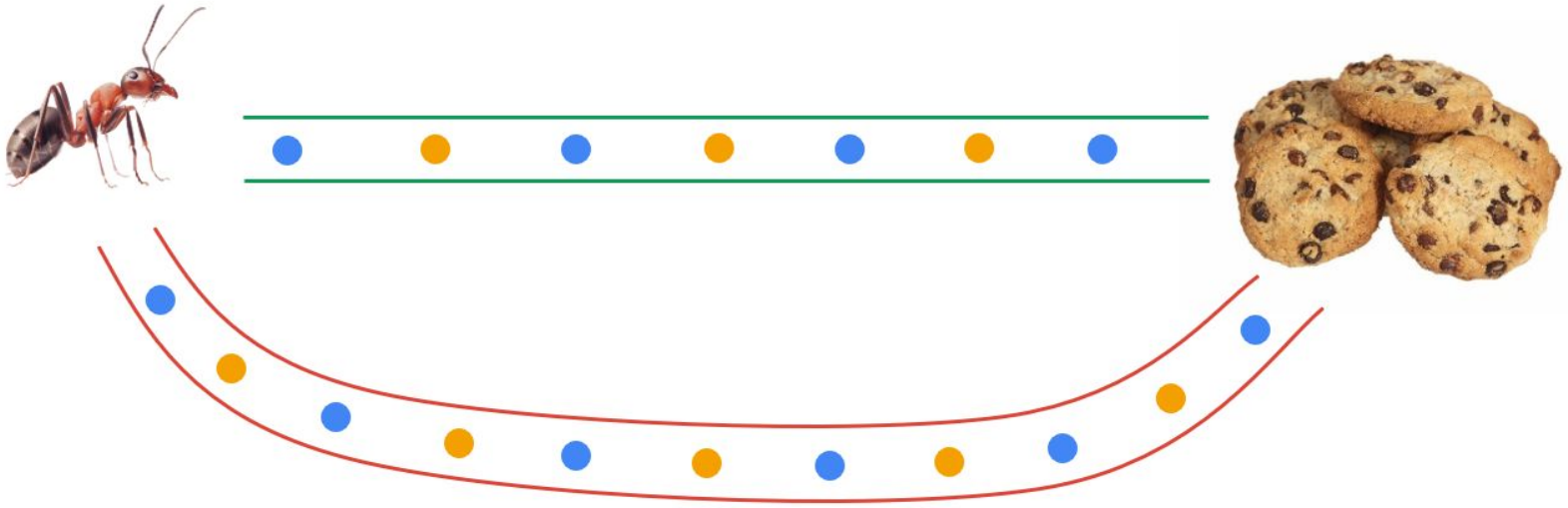
- Máquina de Turing:

```
Input data:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  A ⋙Position:  0 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  A ⋙Position:  1 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  A ⋙Position:  2 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  A ⋙Position:  3 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  A ⋙Position:  4 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  A ⋙Position:  5 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  A ⋙Position:  6 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  B ⋙Position:  5 ⋙Data state:  ['0', '1', '1', '1', '1', '1', '']
⋙State:  B ⋙Position:  4 ⋙Data state:  ['0', '1', '1', '1', '1', '0', '']
⋙State:  B ⋙Position:  3 ⋙Data state:  ['0', '1', '1', '1', '0', '0', '']
⋙State:  B ⋙Position:  2 ⋙Data state:  ['0', '1', '1', '0', '0', '0', '']
⋙State:  B ⋙Position:  1 ⋙Data state:  ['0', '1', '0', '0', '0', '0', '']
⋙State:  B ⋙Position:  0 ⋙Data state:  ['0', '0', '0', '0', '0', '0', '']
⋙State:  C ⋙Position:  -1 ⋙Data state:  ['1', '0', '0', '0', '0', '0', '']
Done! ⋙Output:  ['1', '0', '0', '0', '0', '0', '']
[Finished in 0.1s]
```

Developer Student Clubs
Universidad Politécnica de Valencia

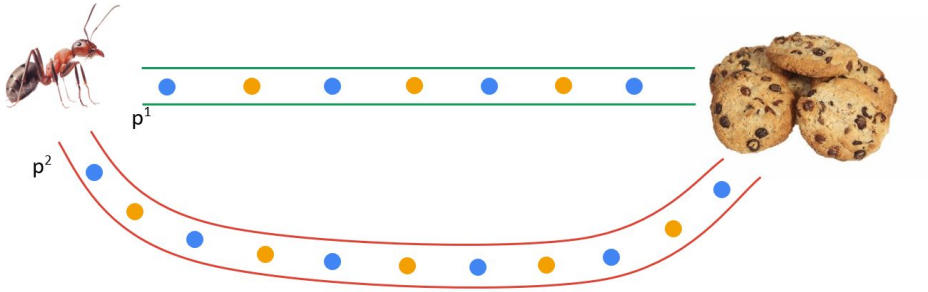
Algoritmo de la colonia de hormigas

- Usado también en clasificación dando mejores resultados que el perceptrón



Developer Student Clubs
Universidad Politécnica de Valencia

Algoritmo de la colonia de hormigas



- Selección del camino:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}$$

- Cantidad de feromonas depositadas para un estado *xy* : τ_{xy}
- Conveniencia del estado (1/distancia) : η_{xy}
- Controla la influencia de τ_{xy} : 0 ≤ α
- Controla la influencia de η_{xy} : 1 ≤ β

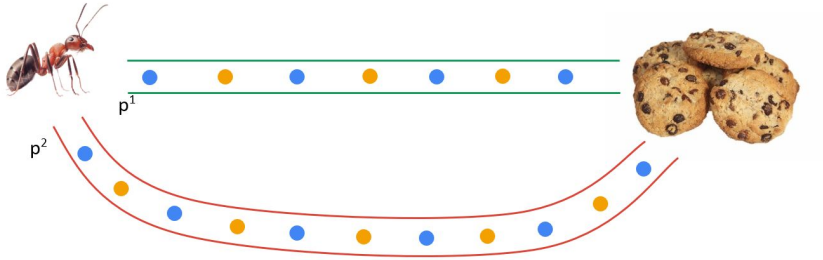
Developer Student Clubs
Universidad Politécnica de Valencia

Algoritmo de la colonia de hormigas

- Actualización de feromonas:

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_{k} \Delta\tau_{xy}^{k}$$

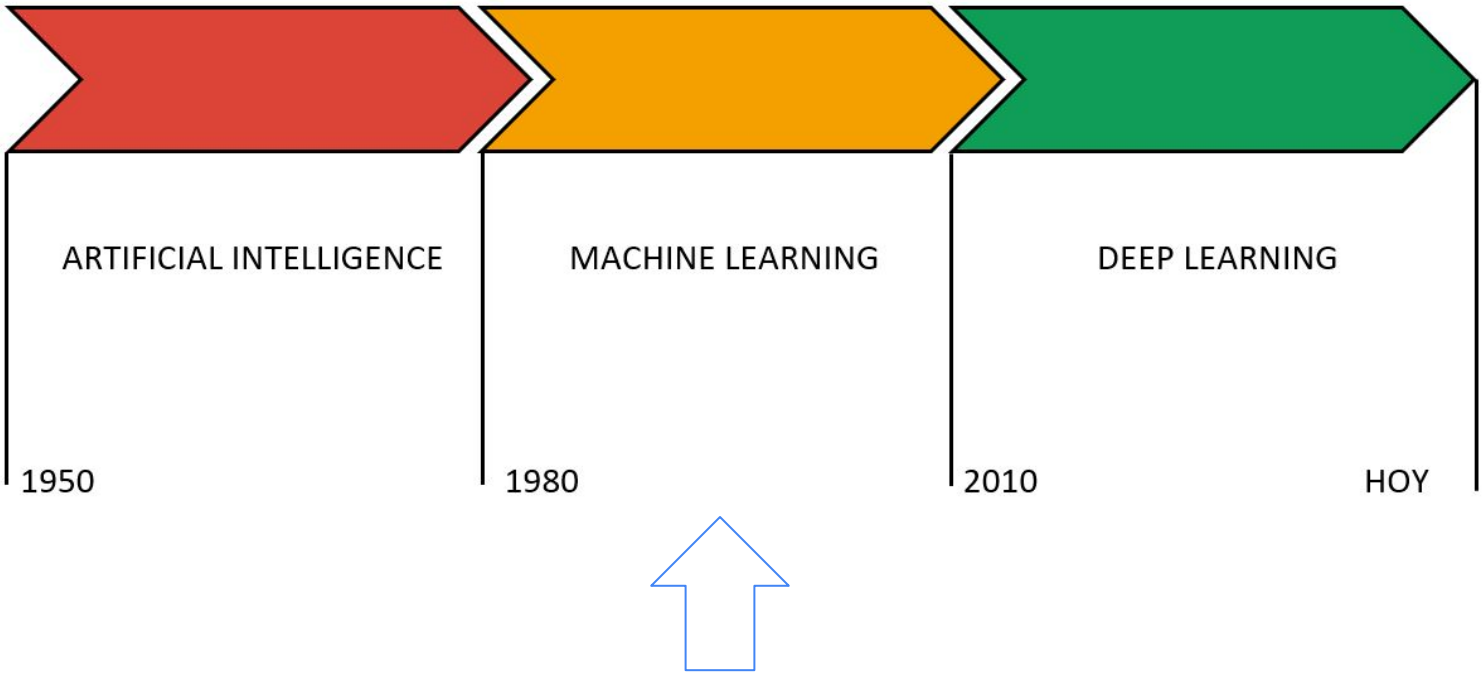
$$\Delta\tau_{xy}^{k} = \begin{cases} Q/L_k & \textbf{Si la hormiga usa el camino} \\ 0 & \textbf{En otro caso} \end{cases}$$



- Cantidad de feromonas depositadas para un estado *xy* : τ_{xy}
- Q = constante, L_k = coste de la ruta (distancia de normal)
- Coeficiente de evaporación de las feromonas: ρ

Developer Student Clubs

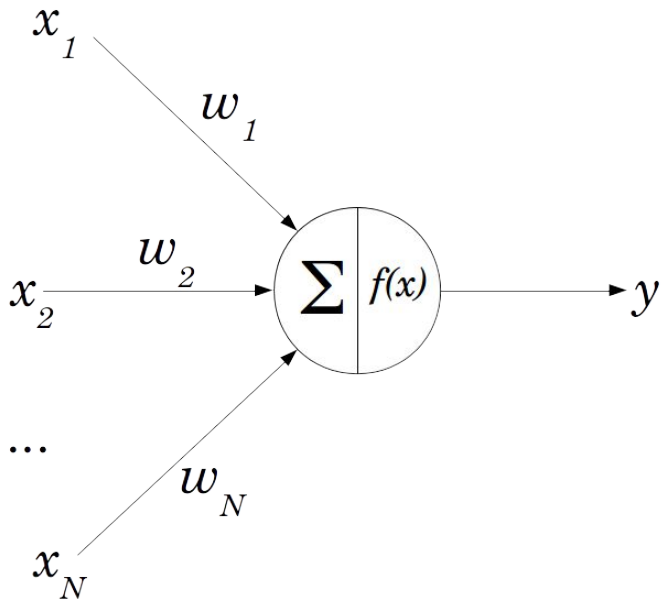
Universidad Politécnica de Valencia



Developer Student Clubs
Universidad Politécnica de Valencia

El perceptrón (Rosenblatt 1957)

- Puede ser considerado como una neurona artificial, unidad de cálculo que intenta modelar el comportamiento de una neurona natural, como las de nuestro cerebro.



Aprendizaje del perceptrón

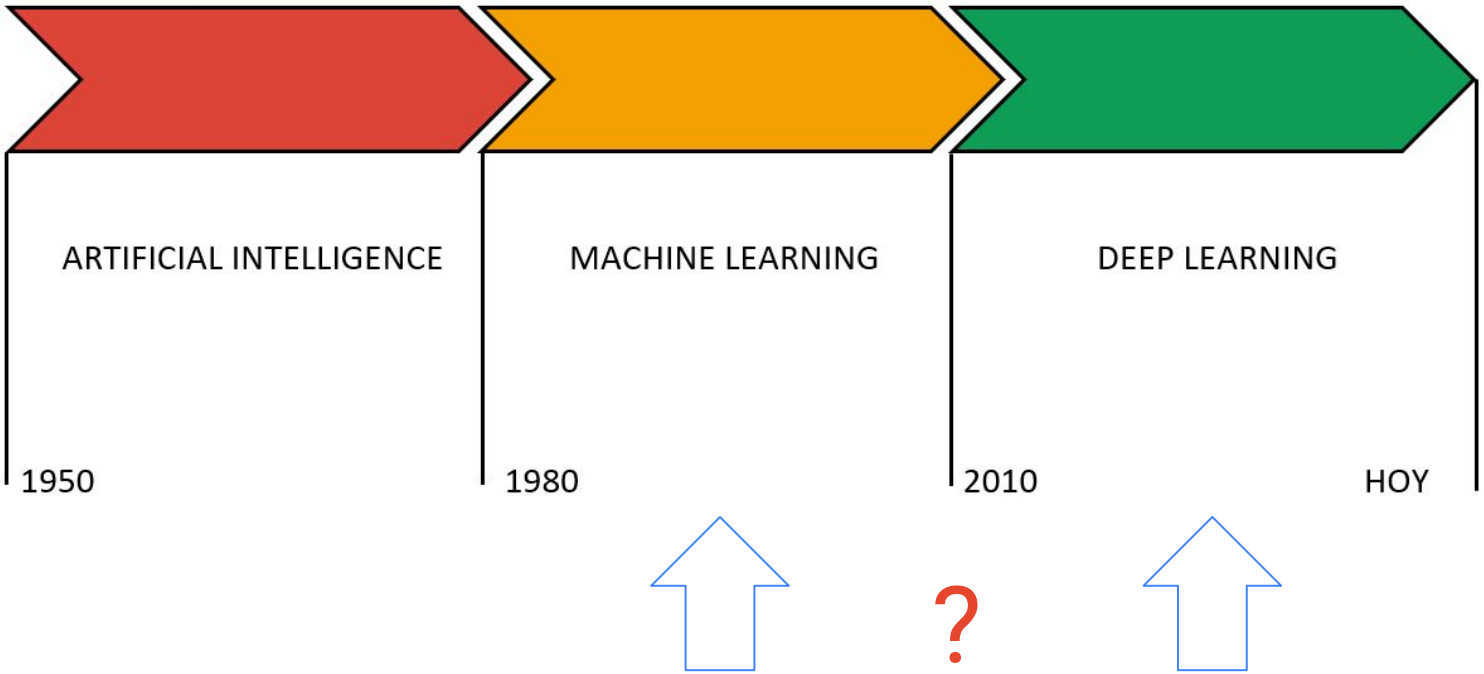
- **Aprendizaje supervisado:**
  - **Aprendizaje por refuerzo.**
  - **Aprendizaje por corrección del error** (ADALINE, Backpropagation**).**

- **Aprendizaje no supervisado:** clustering.
  - **Aprendizaje competitivo y comparativo** (ART, ART2**).**
  - **Aprendizaje Hebbiano.**

- **Aprendizaje OFFLINE y ONLINE.**

Developer Student Clubs

Universidad Politécnica de Valencia



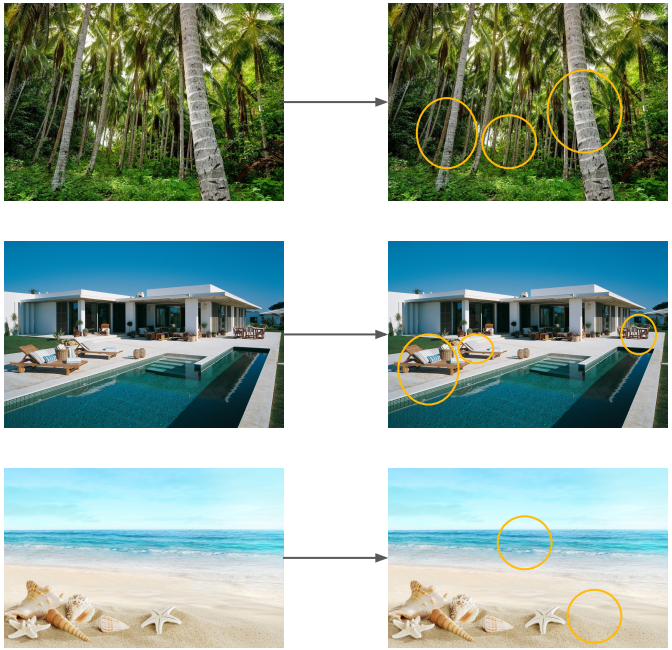
Developer Student Clubs
Universidad Politécnica de Valencia

Machine Learning

Developer Student Clubs

Universidad Politécnica de Valencia

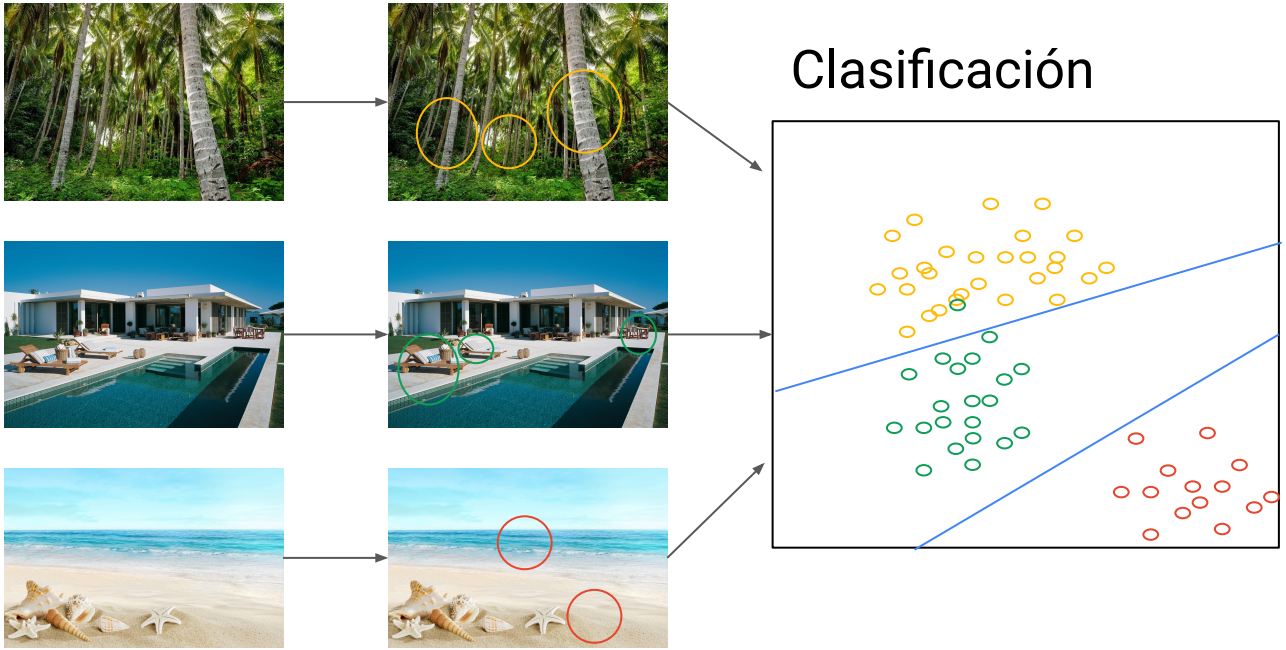
Machine Learning



Developer Student Clubs

Universidad Politécnica de Valencia

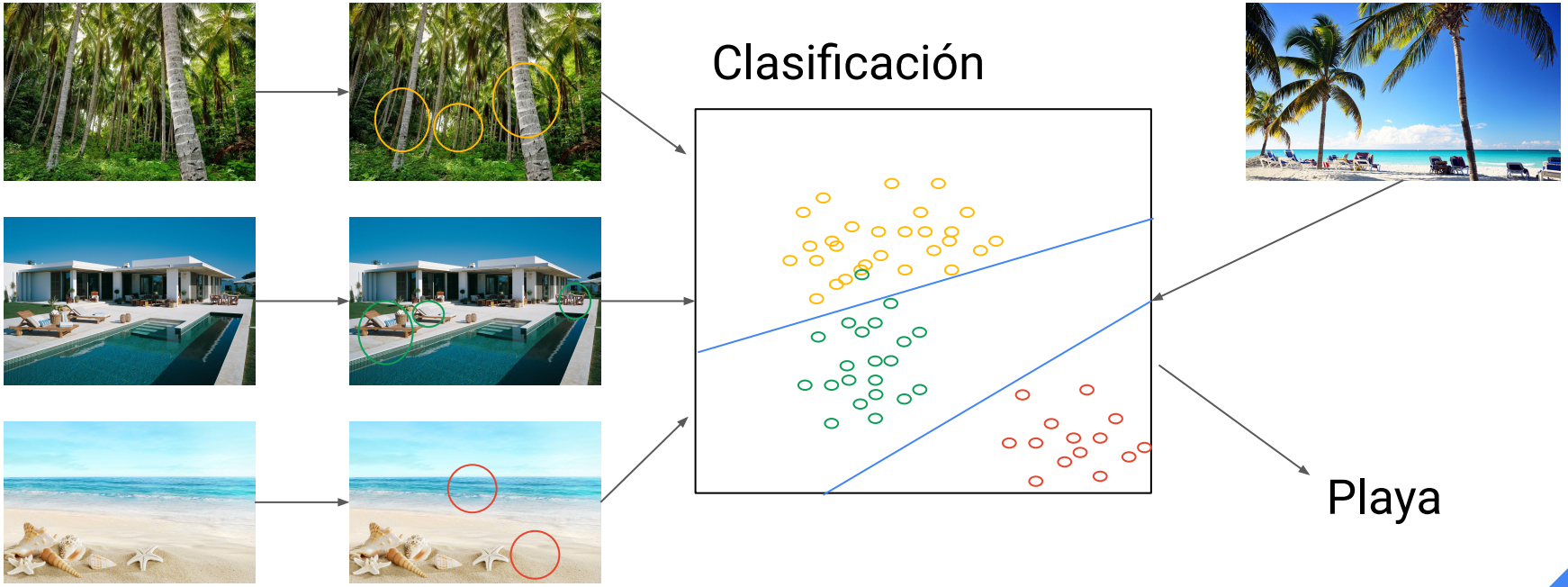
Machine Learning



Developer Student Clubs

Universidad Politécnica de Valencia

Machine Learning



Developer Student Clubs
Universidad Politécnica de Valencia

Deep Learning

Convolutional Neural Network (CNN)

Aprendizaje automático de características + Clasificación

Playa

Developer Student Clubs
Universidad Politécnica de Valencia

Modelo Hopfield

- **Es OFFLINE:** primero hay que entrenarlo antes de usar.
- **Usa memoria asociativa:** almacena y recupera la información por asociación con otras informaciones.
- **Fórmula:** $W_x = T^t \cdot T - I \quad \rightarrow W = \sum(W_x)$
- **Función:** $f(x) = \begin{cases} -1 \text{ if } x < 0 \\ 1 \text{ if } x \geq 0 \end{cases}$
- **Solución**: $f(W \cdot E)$, si no hay solución, iterar.

Developer Student Clubs

Universidad Politécnica de Valencia

Modelo Hopfield

```python
#Hopfield network:
#OFFLine
#Remember that:
#W = T^t * T - I
#Wi = W0 + W1 + ... + Wi
#Also we need some function
#f(x) -> if x < 0 so = -1 ,
#     -> if x >= 0 so = 1
import numpy as np
#Defining identity matrix (I) using numpy
def createI(x):
    return (np.identity(x))

#Defining W matrix formula
def solveW(T):
    #Let's create our matrices
    T = np.matrix(T)
    #T transpose Matrix
    T1 = np.matrix(T.transpose())
    #Identity matrix
    I = np.matrix(createI(len(T)))
    #Let`s use our formula to find W
    W = T1 * T - I
    #And return the solution
    return W
```

```python
#defining the searching algorythm
def solve(E):
    #E * W must be equals to some of our learned items,
    #in case of difference we'll iterate again.
    #In this particular case it will be perfect solution
    #with 1 iteration only
    S = E * W
    #Matrix to Array
    Sa = np.squeeze(np.asarray(S))
    #Now we apply the function f(x)
    for x in range(0, len(Sa)):
        if Sa[x] < 0:
            Sa[x] = -1.0
        else:
            Sa[x] = 1.0
    #Turning the array into the matrix again
    S = np.matrix(Sa)
    #Our basic prediction
    if (S == E0).all():
        print("|   The solution is E0 |")
    elif (S == E1).all():
        print("|   The solution is E1 |")
    else:
        print("|   More loops needed... |")

```

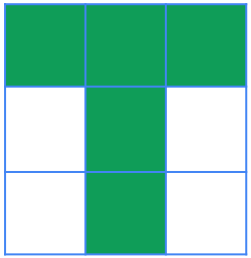
Developer Student Clubs

Universidad Politécnica de Valencia

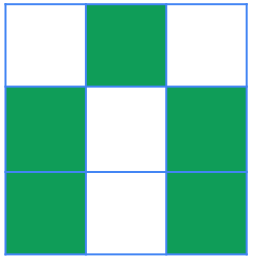
Modelo Hopfield

```
#Test:we suppose that we have 3X3 matrix with two pictures to learn like:
#  111   010
#  010   101
#  010   101
#   w0    w1
#
#1) We'll create two matrix vectors of figures represented before
#1.1)
E0 = np.matrix([1,1,1,-1,1,-1,-1,1,-1])
print(f"E0=",E0)
#1.2)
E1 = np.matrix([-1,1,-1,1,-1,1,1,-1,1])
print(f"E1=",E1)

#2)We solve the W0 for E0
W0 = solveW(E0)
print(f"W0=",W0)

#3)We solve the W1 for E1
W1 = solveW(E1)
print(f"W1=",W1)

#4)Now we must sum W0 and W1 to obtain the final W.
#That's our final trained Hopfield model, just ready for use to proove inputs
W = W0 + W1
print(f"W=", W)



#6)Enjoy and play
print("////////////////////////////////  LET'S PLAY!  \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\")
print("================================================================================")
print("|   >>This solution must be E1
solve(np.matrix([-1,-1,-1,-1,1,1,1,-1,1]))

print("================================================================================")
print("|   >>This solution must be E0
solve(np.matrix([1,-1,1,-1,-1,-1,1,-1]))

print("================================================================================")
print("|   >>This solution must be E1
solve(np.matrix([1,-1,-1,1,-1,1,-1,-1,1]))

print("================================================================================")
print("|   >>This solution must be E1
solve(np.matrix([-1,1,-1,-1,1,1,1,-1,-1]))

print("================================================================================")
print("\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\")
```

- Fase 1: entrenamiento.



E_0



E_1

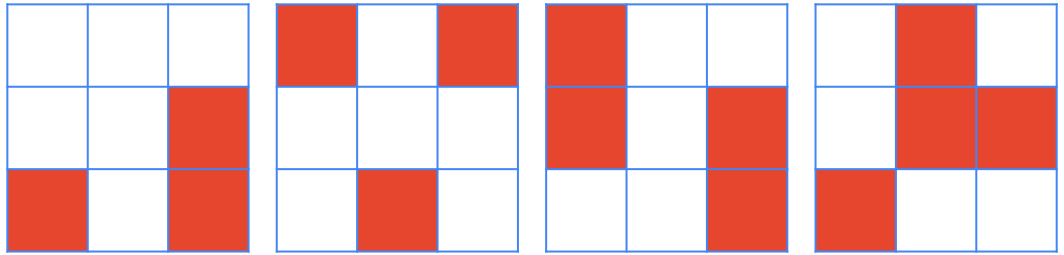
Developer Student Clubs

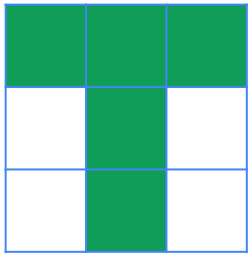
Universidad Politécnica de Valencia

Modelo Hopfield

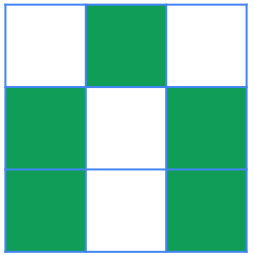
```
#Test:we suppose that we have 3X3 matrix with two pictures to learn like:
#  111   010
#  010   101
#  010   101
#   W0    W1
#
#1) We'll create two matrix vectors of figures represented before
#1.1)
E0 = np.matrix([1,1,1,-1,1,-1,-1,1,-1])
print(f"E0=",E0)
#1.2)
E1 = np.matrix([-1,1,-1,1,-1,1,1,1,-1])
print(f"E1=",E1)

#2)We solve the W0 for E0
W0 = solveW(E0)
print(f"W0=",W0)

#3)We solve the W1 for E1
W1 = solveW(E1)
print(f"W1=",W1)

#4)Now we must sum W0 and W1 to obtain the final W.
#That's our final trained Hopfield model, just ready for use to proove inputs
W = W0 + W1
print(f"W=", W)



#6)Enjoy and play
print("////////////////////////////////  LET'S PLAY!  \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\")
print("================================================================================")
print("|   >>This solution must be E1
solve(np.matrix([-1,-1,-1,-1,-1,1,1,-1,1]))

print("================================================================================")
print("|   >>This solution must be E0
solve(np.matrix([1,-1,1,-1,-1,-1,1,1,-1]))

print("================================================================================")
print("|   >>This solution must be E1
solve(np.matrix([1,-1,-1,1,-1,1,-1,-1,1]))

print("================================================================================")
print("|   >>This solution must be E1
solve(np.matrix([-1,1,-1,-1,1,1,1,-1,-1]))

print("================================================================================")
print("\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\")
```

- Fase 2: uso de la red.





E_0



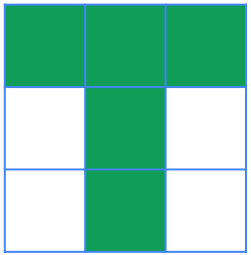
E_1

Developer Student Clubs

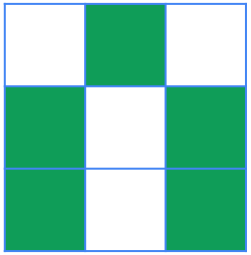
Universidad Politécnica de Valencia

Modelo Hopfield

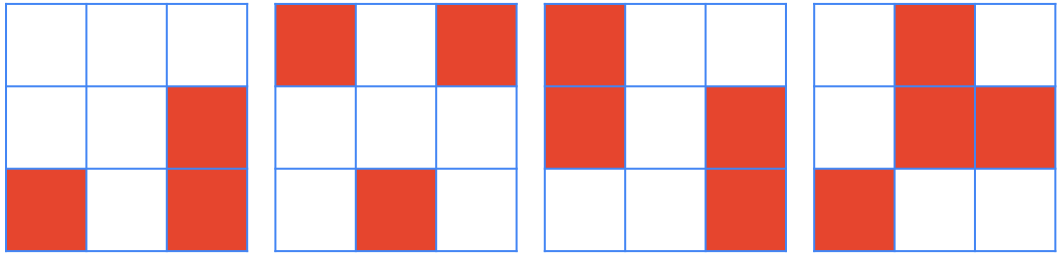
- Fase 2: uso de la red.



E_0



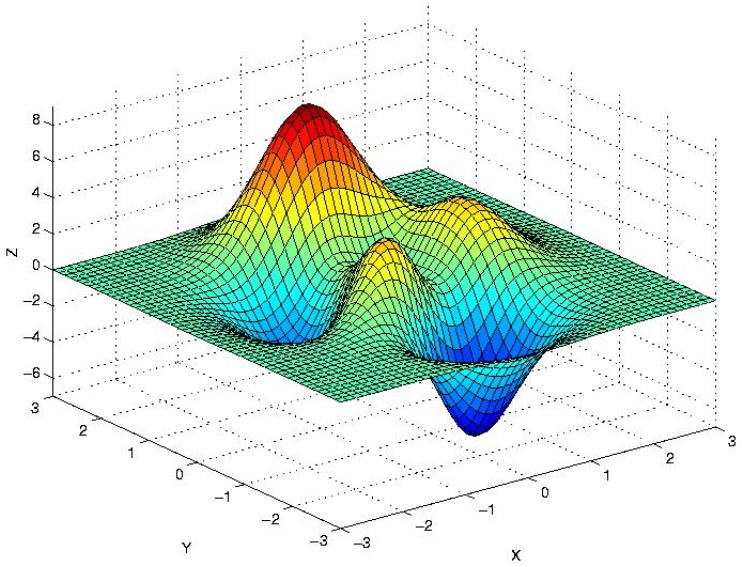
E_1



```
//////////////////////////////  LET'S PLAY!  \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
================================================================================
|   >>This solution must be E1                                                 |
|   The solution is E1                                                         |
================================================================================
|   >>This solution must be E0                                                 |
|   The solution is E0                                                         |
================================================================================
|   >>This solution must be E1                                                 |
|   The solution is E1                                                         |
================================================================================
|   >>This solution must be E1                                                 |
|   The solution is E1                                                         |
================================================================================
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\//////////////////////////////////////////
[Finished in 0.3s]
```

Developer Student Clubs
Universidad Politécnica de Valencia

Gradient descent

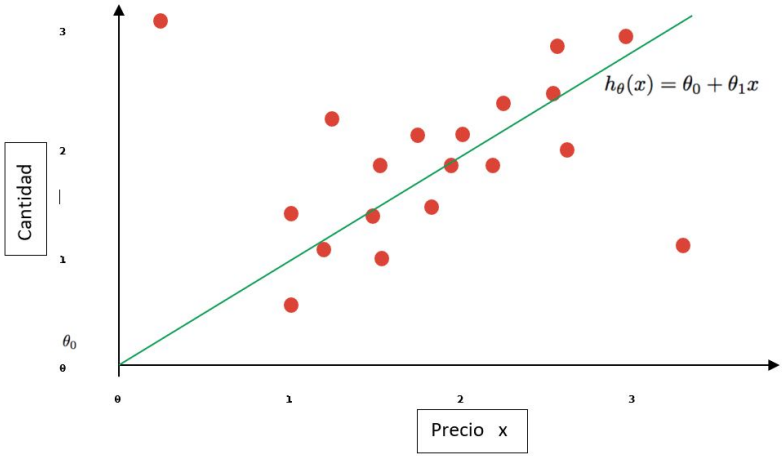


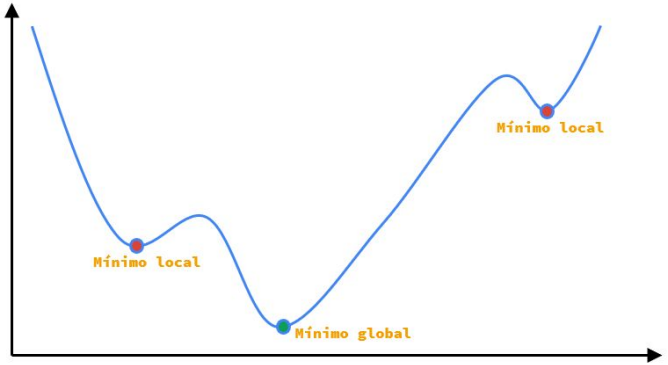
- Algoritmo de optimización que permite converger hacia el valor mínimo de una función mediante un proceso iterativo.

Developer Student Clubs
Universidad Politécnica de Valencia

Gradient descent

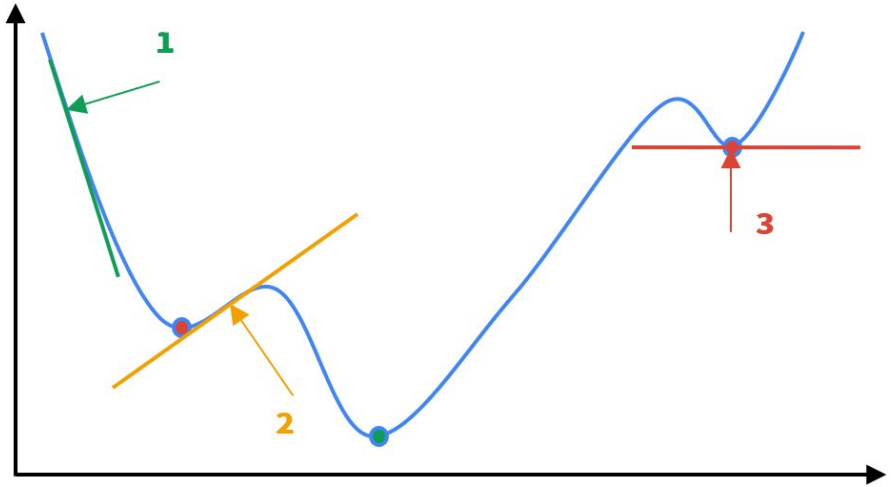
- Un enfoque simplificado: Theta (Θ_0) = 0.
- Aplicado a la regresión lineal.





Developer Student Clubs
Universidad Politécnica de Valencia

Gradient descent

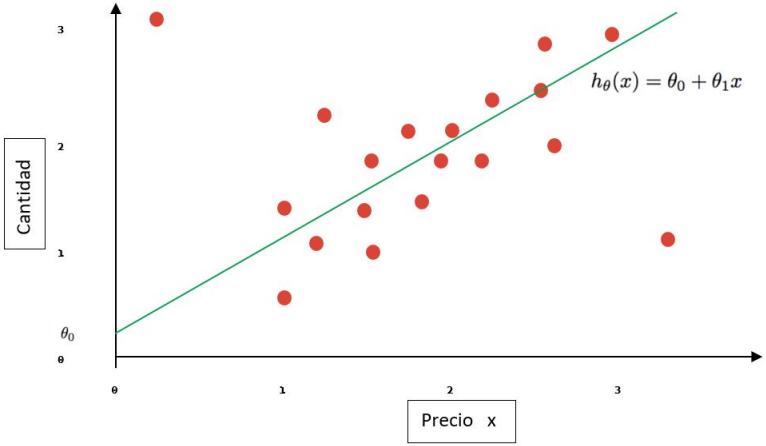


$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1)$$

- Pendiente negativa:
  $\Theta_j := \Theta_j - \alpha \cdot (\text{negativo})$
- Pendiente positiva:
  $\Theta_j := \Theta_j - \alpha \cdot (\text{positivo})$
- Pendiente nula:
  $\Theta_j := \Theta_j - \alpha \cdot 0$

Developer Student Clubs
Universidad Politécnica de Valencia

Gradient descent



- Hipótesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- Función de coste:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Developer Student Clubs
Universidad Politécnica de Valencia

Ecuación normal para regresión lineal

X_0	Cantidad (X_1)	Estado [1-10] (X_2)	Edad [meses] (X_3)	Precio € (X_4)
1	2	5	11	500
1	1	10	2	1250
1	5	1	12	400

$$X = \begin{bmatrix} 1 & 2 & 5 & 11 \\ 1 & 1 & 10 & 2 \\ 1 & 5 & 1 & 12 \end{bmatrix} \quad y = \begin{bmatrix} 500 \\ 1250 \\ 400 \end{bmatrix}$$

- Hipótesis (< 10.000 elem.):

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

- Ecuación normal:

$$\theta = (X^T X)^{-1} X^T y$$

Developer Student Clubs
Universidad Politécnica de Valencia

Ecuación normal para regresión lineal

```
>> y = [500;
        1250;
        400;];
>> X = [1, 2, 5, 11;
        1, 1, 10, 2;
        1, 5, 1, 12;];
>> theta = pinv(X'*X)*X'*y
theta =

    21.636
   132.841
   116.254
   -33.508
```

X_0	Cantidad (X_1)	Estado [1-10] (X_2)	Edad [meses] (X_3)	Precio € (X_4)
1	2	5	11	500
1	1	10	2	1250
1	5	1	12	400

- Ecuación normal:

$$\theta = (X^T X)^{-1} X^T y$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

Developer Student Clubs
Universidad Politécnica de Valencia

Ecuación normal para regresión lineal

X_0	Cantidad (X_1)	Estado [1-10] (X_2)	Edad [meses] (X_3)	Precio € (X_4)
1	2	5	11	500
1	1	10	2	1250
1	5	1	12	400

```
>> x1 = 1; x2 = 10; x3 = 2;
>> h = 21.636 + 132.841*x1 + 116.254*x2 + -33.508*x3;
>> h
h =  1250.0
>> x1 = 1; x2 = 8; x3 = 5;
>> h = 21.636 + 132.841*x1 + 116.254*x2 + -33.508*x3;
>> h
h =  916.97
```

Developer Student Clubs

Universidad Politécnica de Valencia

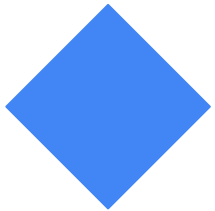
Código

Máquina de Turing:

Hopfield:

Developer Student Clubs

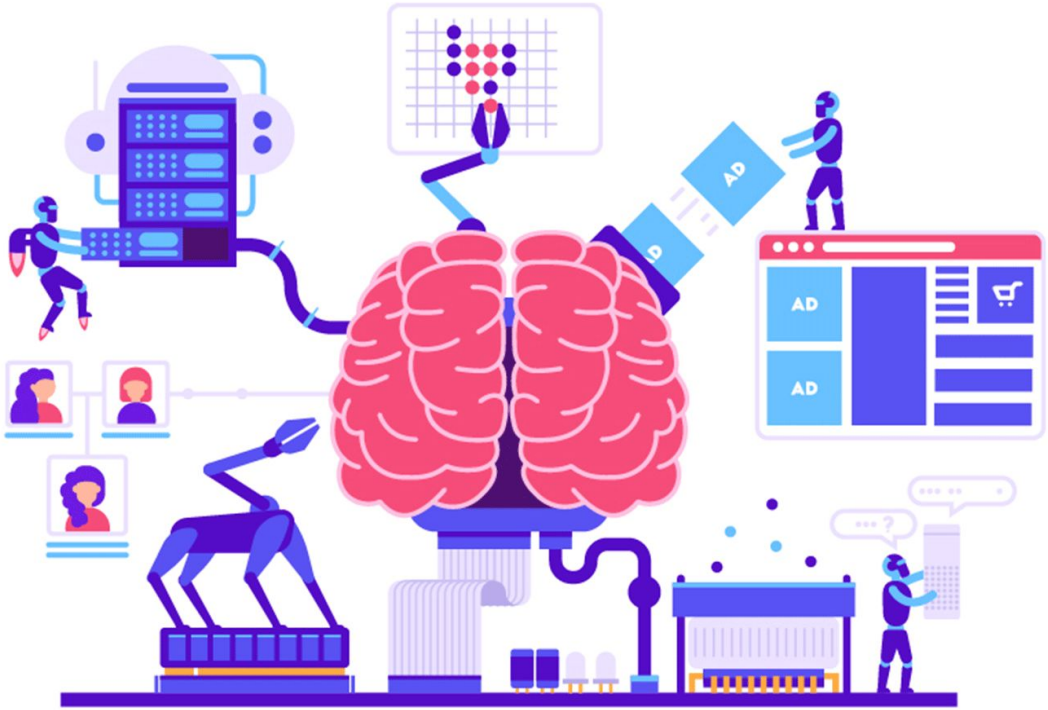
Universidad Politécnica de Valencia



Developer Student Clubs

Universidad Politécnica de Valencia

Aplicaciones de Machine Learning



Developer Student Clubs

Universidad Politécnica de Valencia

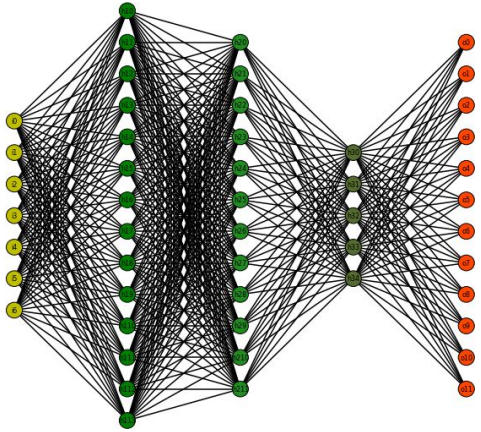
quickdraw.withgoogle.com

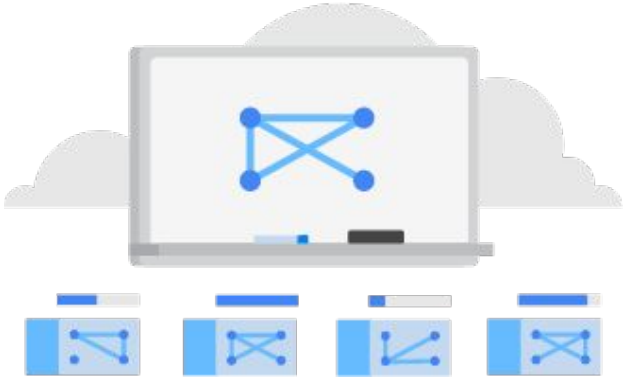
Developer Student Clubs

Universidad Politécnica de Valencia

Escala (de Investigador a Desarrollador):

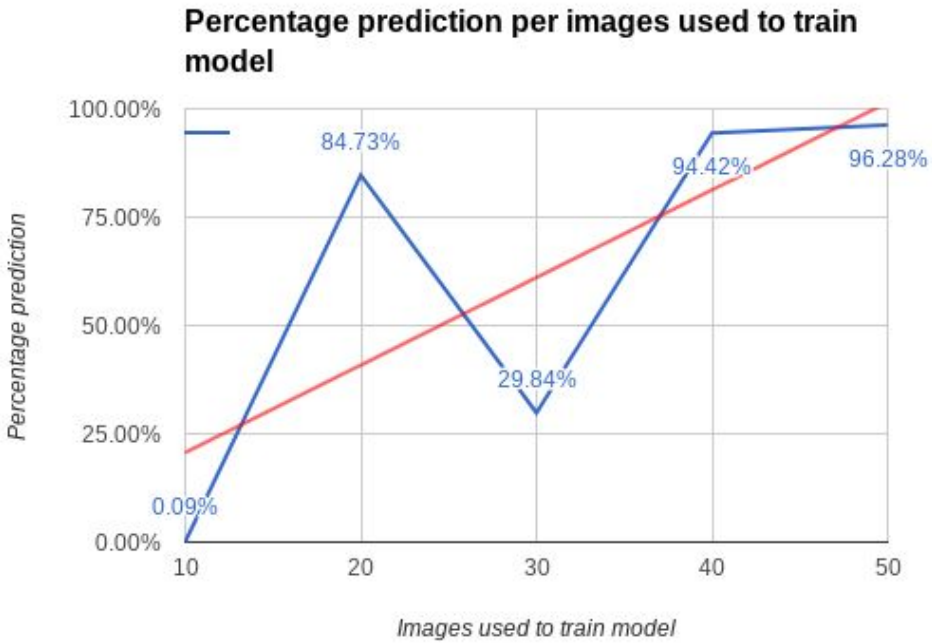
- Redes Neuronales
- AutoML
- Modelos Ready-to-use





Developer Student Clubs

Universidad Politécnica de Valencia



NVIDIA DIGITS

Caffe

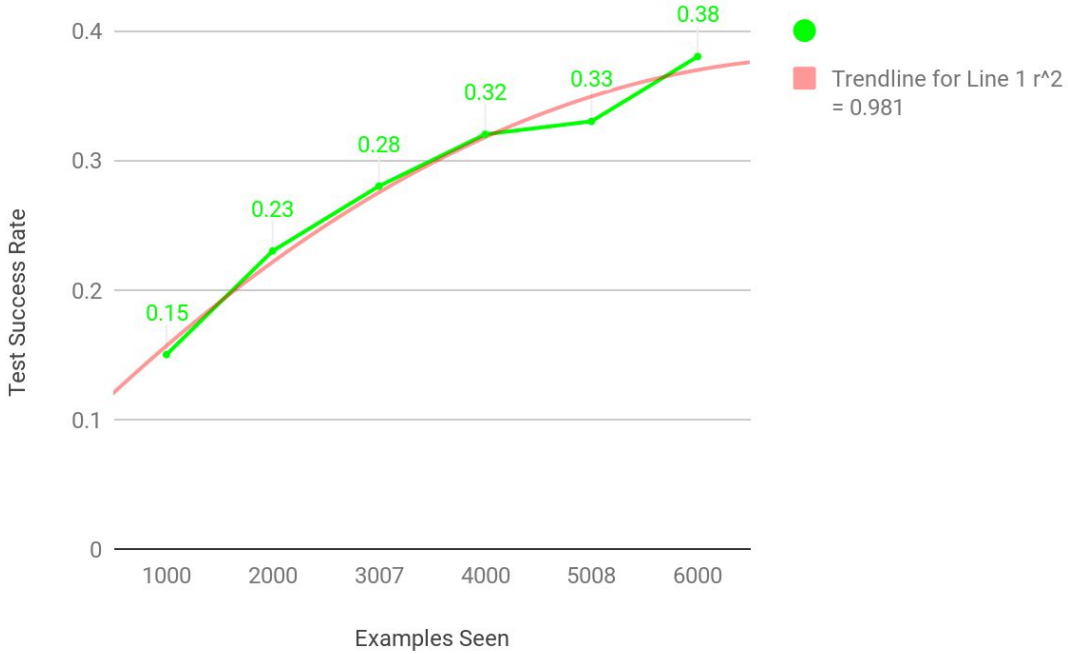
CUDA

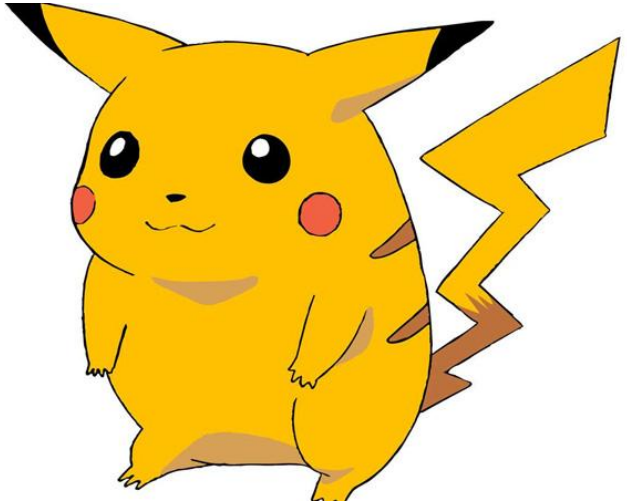
Developer Student Clubs

Universidad Politécnica de Valencia

Developer Student Clubs

Universidad Politécnica de Valencia





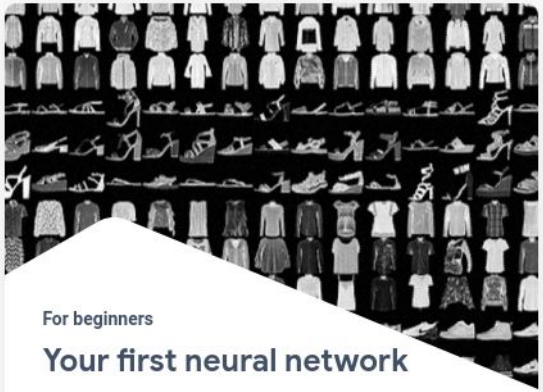
TensorFlow

Developer Student Clubs

Universidad Politécnica de Valencia

Solutions to common problems

Explore step-by-step tutorials to help you with your projects.



For beginners

Your first neural network

Train a neural network to classify images of clothing, like sneakers and shirts, in this fast-paced overview of a complete TensorFlow program.

For experts

Generative adversarial networks

Train a generative adversarial network to generate images of handwritten digits, using the Keras Subclassing API.

For experts

Neural machine translation with attention

Train a sequence-to-sequence model for Spanish to English translation using the Keras Subclassing API.

Developer Student Clubs
Universidad Politécnica de Valencia

tensorflow.org/resources/learn-ml

TensorFlow + Keras

Developer Student Clubs
Universidad Politécnica de Valencia

¿Quién usa TensorFlow?









Developer Student Clubs

Universidad Politécnica de Valencia

Google Vision API

Detecta:

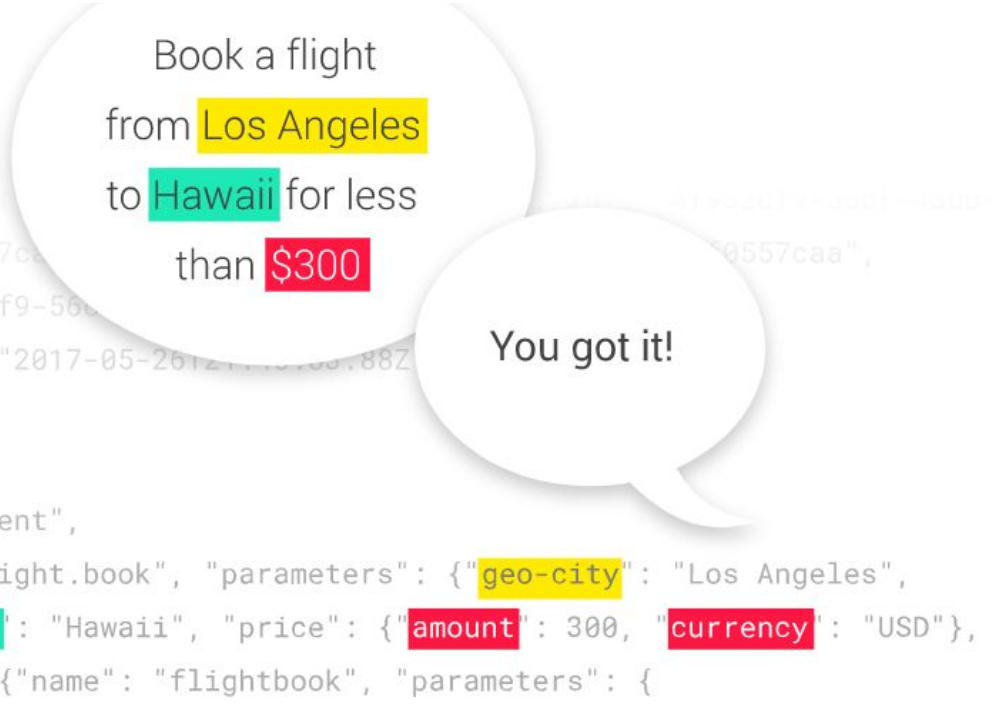
- Objetos
- Etiquetas
- Puntos de referencia
- Caras
- Expresiones faciales
- Texto y texto escrito

Developer Student Clubs
Universidad Politécnica de Valencia

Google Natural Language API

Puede:

- Classificar texto
- Detectar dependencias
- Detectar etiquetas
- Detectar estructuras



Developer Student Clubs
Universidad Politécnica de Valencia

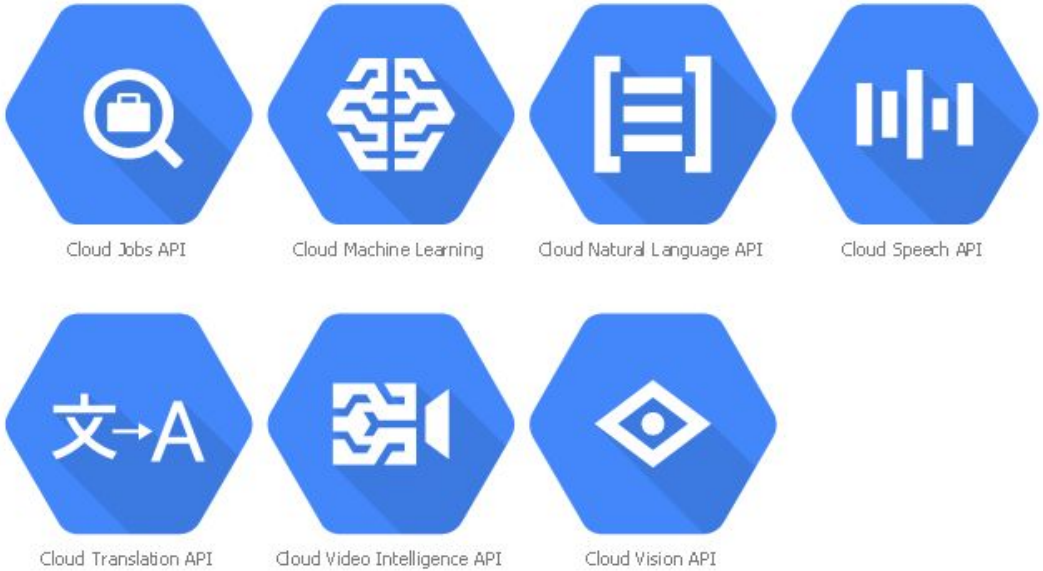
Google Translation API

Puede:

- Traducir
- Converti habla en texto
- Convertir texto en habla

Developer Student Clubs
Universidad Politécnica de Valencia

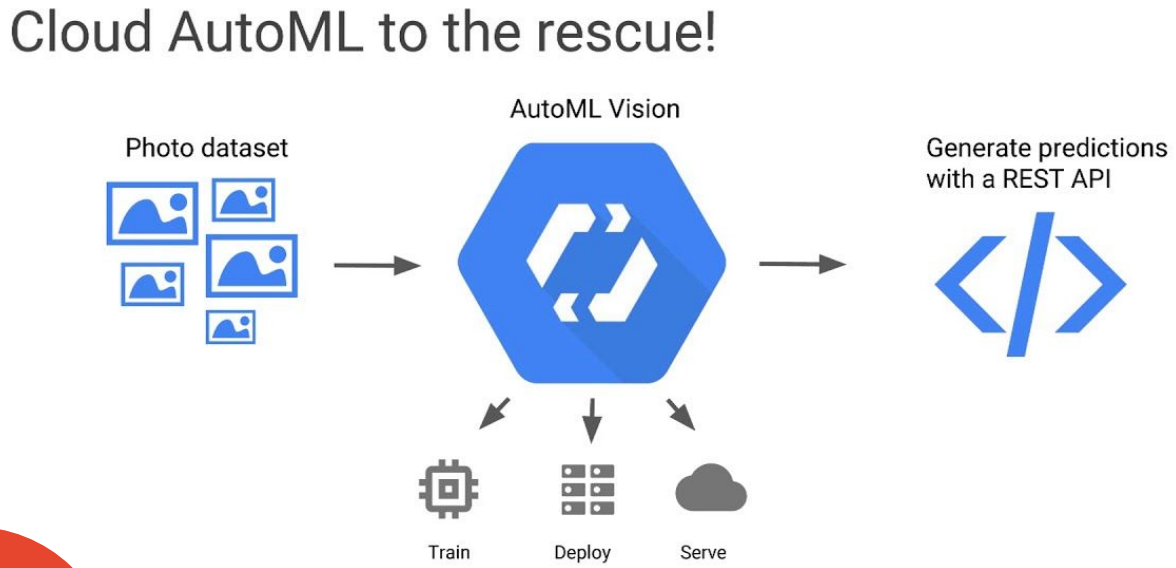
Modelos ya entrenados accesibles a través de las APIs de Google

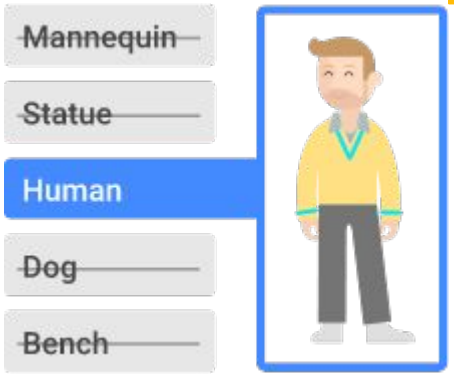


Developer Student Clubs

Universidad Politécnica de Valencia

Google AutoML





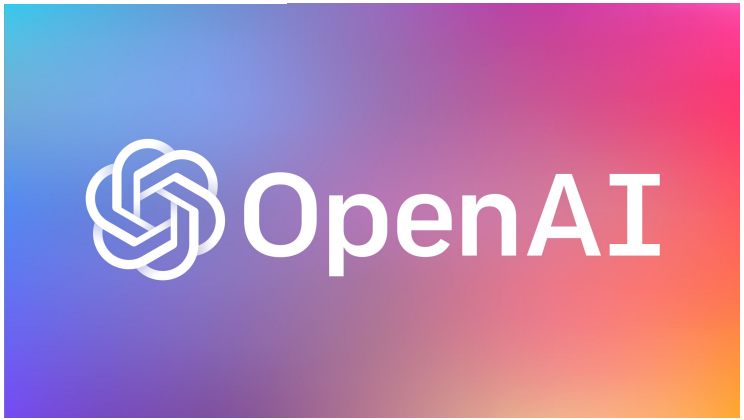
Developer Student Clubs
Universidad Politécnica de Valencia

GPT-2

Developer Student Clubs
Universidad Politécnica de Valencia

OpenAI dice haber creado un generador de texto tan potente que "liberarlo sería peligroso"

Elon Musk's OpenAI builds artificial intelligence so powerful it must be kept locked up for the good of humanity

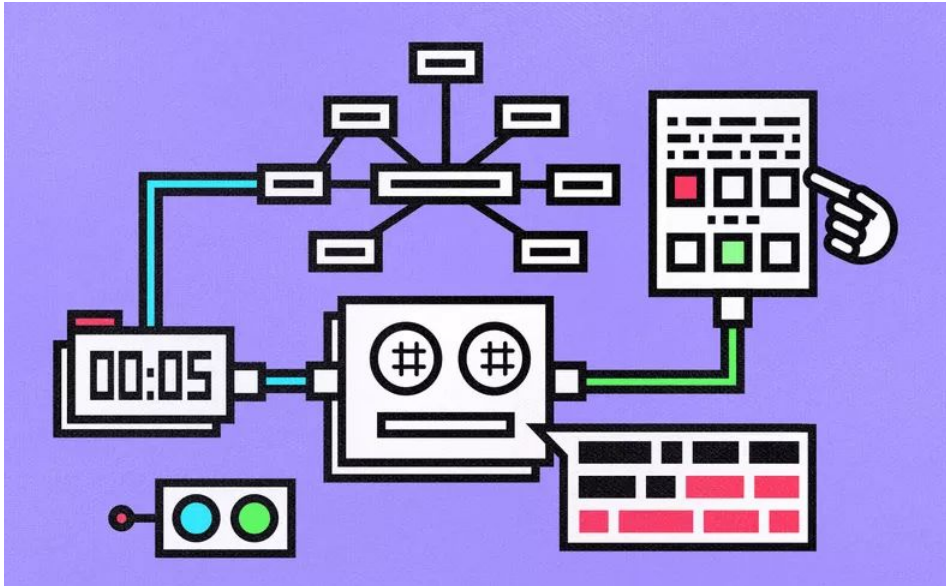


Developer Student Clubs

Universidad Politécnica de Valencia

OpenAI has published the text-generating AI it said was too dangerous to share

The lab says it's seen 'no strong evidence of misuse so far'



Developer Student Clubs

Universidad Politécnica de Valencia

talktotransformer.com

“Words are pale shadows of forgotten names. As names have power, words have power. Words can light fires in the minds of men. Words can wring tears from the hardest hearts.”

— Patrick Rothfuss, The Name of the Wind

Words are pale shadows of forgotten names. As names have power, words have power. Words can light fires in the minds of men. Words can wring tears from the hardest hearts. Words can set the pace of a lifetime.

I want my words to be as potent as they are vivid. I want the words I speak to be remembered forever. And to this end, my words will stand on the shoulders of my forefathers.

The words of my ancestors shall guide and guide me as I write these words. They shall be my legacy.

Las palabras son sombras pálidas de nombres olvidados. Como los nombres tienen poder, las palabras tienen poder. Las palabras pueden encender fuegos en la mente de los hombres. Las palabras pueden estrujar las lágrimas de los corazones más duros. Las palabras pueden marcar el ritmo de tu vida.

Quiero que mis palabras sean tan potentes como vívidas. Quiero que las palabras que digo sean recordadas para siempre. Y para este fin, mis palabras estarán sobre los hombros de mis antepasados.

Las palabras de mis antepasados me guiarán y guiarán mientras escribo estas palabras. Serán mi legado.

talktotransformer.com