

✓ Neurona McCulloch y Pitts

(Ejercicio, 3 puntos posibles)

El modelo de McCulloch y Pitts, concebido por Warren McCulloch, neurocientífico, y Walter Pitts, lógico matemático, en 1943 [1], representa uno de los fundamentos teóricos de las redes neuronales y la inteligencia artificial. Este modelo es una simplificación abstracta de las neuronas biológicas, propuesta para entender cómo podrían las neuronas del cerebro generar patrones complejos de pensamiento a partir de operaciones simples.

 [Open in Colab](#)

[1] McCulloch, W. S., & Pitts, W. (1990). A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biology, 52, 99-115.

Para formalizar el modelo de McCulloch y Pitts, representaremos una neurona con la letra C y definiremos que la entrada de dicha neurona, denominada *input*, está determinada por una tupla (E, I) , donde E es el conjunto de señales excitatorias e I es el conjunto de señales inhibitorias. Asimismo, estableceremos como restricciones que tanto la entrada como la salida sean variables binarias, es decir, $input, output \in \{0, 1\}$. Adicionalmente, definiremos un umbral u que la neurona utiliza para determinar los casos de excitación..

Dado lo anterior la salida de C se calcula usando las siguientes reglas:

1. En caso de que alguna de las entradas inhibitorias esté activa la neurona no se excita, es decir,

$$C(E, I) = 0 \text{ if } \exists l_i = 1; l_i \in I$$

2. La neurona se excita si la integral de sus entradas excitatorias es igual o superior al umbral, es decir,

$$C(E, I) = 1 \text{ if } \sum e_i \geq u; e \in E$$

3. En cualquier otro caso la neurona permanece sin excitación.


```
#importamos paquetes necesarios
import numpy as np
```

```
# TODO: (2 puntos) Implemente la función del modelo M&P, no use funciones predefinidas de numpy
```

```
def neuronaMyP(E,I,u):
    if sum(I) >= 1:
        return 0
    elif sum(E) >= u:
        return 1
    else:
        return 0
```

```
# suponga
E = [1]
I = [0]
u = 1
```

```
# Calcule la salida de la neurona y verifique si es correcto
print(f"Entrada excitatoria E={E}, Entrada inhibitoria I={I}, Umbral u={u} => Salida: {neuronaMyP(E,I,u)}")
```

 Entrada excitatoria E=[1], Entrada inhibitoria I=[0], Umbral u=1 => Salida: 1

```
# TODO: (1 punto) Implemente un programa que reciba vectores arbitrarios de E, I y u y devuelva la salida de la neurona.
```

```
def vectores_arbitrarios():
    casos_prueba = [
        # Compuerta AND
        ([0,0], [], 2),
        ([0,1], [], 2),
        ([1,0], [], 2),
        ([1,1], [], 2),
        # Compuerta OR
        ([0,0], [], 1),
        ([0,1], [], 1),
        ([1,0], [], 1),
        ([1,1], [], 1),
        # Compuerta NOT
        ([], [0], 0),
        ([], [1], 0)
    ]
```

```

J
for E, I, u in casos_prueba:
    salida = neuronaMyP(E, I, u)
    print(f"Entrada excitatoria E={E}, Entrada inhibitoria I={I}, Umbral u={u} => Salida: {salida}")

```

vectores_arbitrarios()

```

↳ Entrada excitatoria E=[0, 0], Entrada inhibitoria I=[], Umbral u=2 => Salida: 0
Entrada excitatoria E=[0, 1], Entrada inhibitoria I=[], Umbral u=2 => Salida: 0
Entrada excitatoria E=[1, 0], Entrada inhibitoria I=[], Umbral u=2 => Salida: 0
Entrada excitatoria E=[1, 1], Entrada inhibitoria I=[], Umbral u=2 => Salida: 1
Entrada excitatoria E=[0, 0], Entrada inhibitoria I=[], Umbral u=1 => Salida: 0
Entrada excitatoria E=[0, 1], Entrada inhibitoria I=[], Umbral u=1 => Salida: 1
Entrada excitatoria E=[1, 0], Entrada inhibitoria I=[], Umbral u=1 => Salida: 1
Entrada excitatoria E=[1, 1], Entrada inhibitoria I=[], Umbral u=1 => Salida: 1
Entrada excitatoria E=[], Entrada inhibitoria I=[0], Umbral u=0 => Salida: 1
Entrada excitatoria E=[], Entrada inhibitoria I=[1], Umbral u=0 => Salida: 0

```