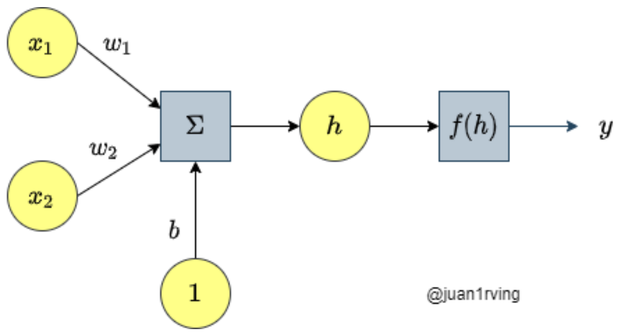
Ejercicio Realizado por ADAIR HERNANDEZ VALDIVIA

Perceptrón simple

(Ejercicio, 3 puntos posibles)

En este notebook programaremos un perceptron simple utilizando numpy. El objetivo es que comprendamos el funcionamiento del perceptrón y que practiquemos la programación en Python. En la siguiente figura se encuentra una representación del perceptrón.



Open in Colab

@juan1rving

In [23]:
```
# Cargamos paquetes
import numpy as np
```

Calcular producto punto

El primer paso es calcular el valor intermedio, *h*, a partir del producto punto. La fórmula explícita es la siguiente:

$$h = WX + b$$

In [24]:
```
# TODO: (1 punto) Implementar la función h sin utilizar funciones predefinidas de numpy como numpy.dot()
#     * Asegúrate de que W y X tengan la misma forma.
#     * Imprime h durante el debug para ver el valor intermedio.

def combinacion_lineal(W, X, b):
    suma = 0
    for i in range(len(W)):
        suma += W[i] * X[i]
    suma += b
    return suma
```

Función de activación

Para este ejemplo utilizaremos la función escalón como función de activación.

$$f(h) = \begin{cases} 0 & \text{if } h < a \\ 1 & \text{if } h \geq a \end{cases}$$

In [25]:
```
# TODO: (1 punto) Completar el código
# Tips:
# - La función escalón debe devolver 0 cuando h < a y 1 cuando h >= a.
# - Puedes elegir un umbral 'a' (por ejemplo 0) o utilizar uno definido fuera de la función.
# - h puede ser un escalar o un array de numpy: usa operaciones vectorizadas para que funcione en ambos casos.
# - Una forma común es usar la comparación (h >= a) y convertir el array booleano a enteros (0/1).

def escalon(h):
    a = 0  # Umbral
    if h >= a:
        return 1
    else:
        return 0
```

Definir perceptrón

Perceptrón como una función

In [26]:
```
# Perceptrón simple
def perceptron(W, X, b, activacion):
    h = combinacion_lineal(W, X, b)
    return activacion(h)
```

Probar inferencia

Ahora definamos unos pesos y veamos el resultado de una pasada frontal (fordward pass).

In [27]:
```
# Definamos unos pesos y sesgo
inputs = np.array([0.7, -0.3])
weights = np.array([0.1, 0.8])
bias = -0.1

# Pasada frontal
activacion = escalon #definir la función a usar
output = perceptron(weights, inputs, bias, activacion)

print('Output:')
print(output)
```

```
Output:
0
```

In [28]:
```
# TODO (1 punto): Realizar el pase frontal y encuentra por prueba y error los pesos que concuerdan con la función OR

# Definir los datos de entrada para la función OR
inputs_or = np.array([[0, 0],
                      [0, 1],
                      [1, 0],
                      [1, 1]])
# Definir los pesos y el sesgo para la función OR
weights_or = np.array([0.3, 0.5])  # Pesos
bias_or = -0.1  # Bias

# Realizar el pase frontal para cada combinación de entradas
for x in inputs_or:
    output_or = perceptron(weights_or, x, bias_or, escalon)
    print(f'Input: {x}, Output: {output_or}')
```

```
Input: [0 0], Output: 0
Input: [0 1], Output: 1
Input: [1 0], Output: 1
Input: [1 1], Output: 1
```