TP #0: Artificial intelligence and machine learning September 2025 #
Dr. Samir KENOUCHE

Problem 1. *The perceptron is the simplest kind of artificial neural network. It is a linear classifier used to separate data into two classes (e.g., 0 and 1, true or false). It is a supervised learning model, meaning that it learns from labeled examples (inputs and expected output). The structure of a perceptron consists of:*

- *Inputs* $x^T = [x_1, x_2, \cdots, x_n]$
- *Weights* $w^T = [w_1, w_2, \cdots, w_n]$
- *Biais* b
- *Activation function (usually a threshold or sigmoid)*

The neuron calculates:

$$z = \sum_{i=1}^{n} w_1 \cdot x_i + b \tag{1}$$

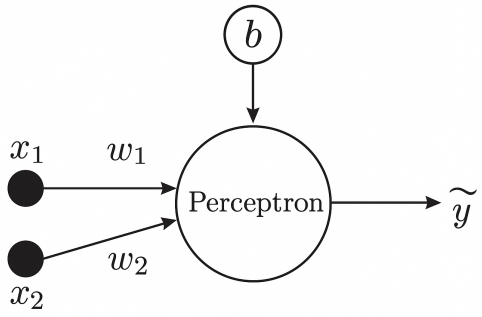
Then applies an activation function:

$$\sigma(z) = \begin{cases} 1 & \text{if } z > 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

During training, weights are adjusted according to the error:

$$w_i \leftarrow w_i + \eta\,(y_i - \hat{y}_i)\,x_i$$
$$b \leftarrow b + \eta\,(y_i - \hat{y}_i)$$

Where $\hat{y}_i$ *is the perceptron output and* η *is the learning rate, which is a hyperparameter taking values from 0 to 1. The structure of this perceptron can be summarized simply using the following diagram:*



Solution. An example of an application is provided as a Phyton program:

```
import numpy as np

""" Dr. SAMIR KENOUCHE - simple example of supervised learning
(case of a perceptron) 25/09/2025 """

# Activation function
def activation_class(z):
    return 1 if z > 0 else 0

# Inputs and expected values
x_inputs = np.array([
    [1, 0],
    [1, 1],
    [0, 1],
    [1, 1]
])
y_target = np.array([1, 0, 0, 1]) # Targets
# Weight an biais initialisation
w = np.random.rand(2)
b = np.random.rand(1)
eta = 0.3 # Learning rate

# Learning
for epoch in range(20):
    for i in range(len(x_inputs)):
        z = np.dot(x_inputs[i], w) + b
        yhat = activation_class(z)
        error = y_target[i] - yhat
        # Update of w and b
        w = w + eta * error * x_inputs[i]
        b = b + eta * error

print("Weight :", w, "Biais :", b )

# Learning Test
X_test = np.array([
    [1, 0],
    [0, 1],
    [1, 0],
    [0, 1]
])

for i in range(len(X_test)):
    z = np.dot(X_test[i], w) + b
    yhat = activation_class(z)
    print(f"Input test : {X_test[i]}, Predicted value : {yhat}")
```

□