TP 1 Artificial intelligence and machine learning October 2025 #
Dr. Samir KENOUCHE

This course focuses on studying linear regression in the context of neural networks derived from artificial intelligence. In this context, regression can be understood as a simple form of supervised learning, which often forms the basis for more complex models. It should be noted that adjustment is an optimization operation that involves searching for the theoretical profile that best fits the experimental data (the examples). In many cases, graphical representation is insufficient to understand the functional relationship between the data to be adjusted. In fact, prior knowledge of the theoretical model may be essential. The adjustment process must allow the outputs (y_i) to be interpreted and predicted based on the inputs (x_i). There are many methods for adjusting the parameters of the theoretical model in order to choose the best fit. In this course, we will discuss adjustment using ordinary and weighted least squares. These techniques are widely used after data collection in both physics and chemistry. The algorithms will be written in Python. The theoretical aspect of the least squares method is described at the end of the document.

Problem 1. *Linear fitting consists of finding a linear relationship between the inputs (features) and an output (target). Mathematically, this is written as:*

$$\hat{y} = w^T x + b \tag{1}$$

- *x: Input vector*
- *w: Weight vector*
- *b: Bias*
- *$\hat{y}$: Prediction*

The purpose is to find the optimal values of w and b that minimize the error between $\hat{y}$ (predicted value) and y (true value). In a neural network, each neuron basically performs a linear adjustment, followed (often) by a nonlinear activation function:

$$a = f(w^T x + b) \tag{2}$$

Learning is performed using the gradient descent method. More details about this method are available on my website Click here

(3)

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

```
""" Dr. Samir KENOUCHE 20/10/2025 """

import numpy as np
import matplotlib.pyplot as plt
import random

np.random.seed(1)
```

```
##### EXAMPLE OF DATA #####
nombre_points = 100
x = np.random.rand(nombre_points)
y = 10*x + 1 + np.random.randn(x.size)

##### INITIALIZATION OF LEARNING PARAMETERS ####
weight = 0.0   # WEIGHT
bias = 0.0     # BIAS
neta = 0.1     # LEARNING RATE
ndata = 100    # NUMBER OF LEARNING CYCLES

critere = np.zeros(ndata)

for epoch in range(ndata):

    dweight = 0
    dbias  = 0
    cost = 0

    for i in range(nombre_points):

        yhat  = weight*x[i] + bias
        dweight = dweight + (yhat - y[i]) * x[i]
        dbias = dbias + (yhat - y[i])
        cost = cost + (yhat-y[i])**2

        ##### UPDATE OF W AND B BY GRADIENT DESCENT: LEARNING #####
        weight = weight - neta*(2*dweight/nombre_points)
        bias   = bias   - neta*(2*dbias/nombre_points)
        critere[epoch] = cost

print(weight,bias)

##### DISPLAYING GRAPHS #####
plt.scatter(x, y)
plt.plot(x, weight*x+bias,"r-", label="Fitting", linewidth=1)
plt.title("Random scatter plot")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.show()

plt.plot(critere[:60])
plt.show()
```

A simple neural network without a nonlinear activation function. We can also perform the same task using the Keras library dedicated to neural networks.

```python
""" Dr. Samir KENOUCHE 20/10/2025 """

from keras import *
import numpy as np
import matplotlib.pyplot as plt
import random

##### EXAMPLE OF DATA #####
nombre_points = 100
x = np.random.rand(nombre_points)
y = 10*x + 1 + np.random.randn(x.size)

# MODEL BUILDING
model = Sequential()
model.add(layers.Dense(units=1, input_shape=[1]))
model.compile(loss='mean_squared_error', optimizer='sgd')

# LEARNING
model.fit(x, y, epochs=300, verbose=0)

# RESULTS
weights = model.get_weights()
print("WEIGHTs:", weights[0], "BIAIS :", weights[1])
```

□

1 Theoretical background

Many methods exist for adjusting the parameters of theoretical models in order to choose the best fit. In this section, we will discuss linear regression [1] in the **least squares** sense. From a conceptual point of view, the linear model is characterized by two entities, deterministic and random, according to:

$$\text{Random variable} \leftarrow\!\prec y = \underbrace{f(x;\theta_i)}_{\text{Deterministic}} + \underbrace{\epsilon_i}_{\text{Random}} \tag{4}$$

The dependent variable y has a random character "inherited" from ϵ_i. This regression model is constructed in accordance with the following assumptions:

- The distribution of the residuals (or regression error) ϵ_i is independent of x. This is the assumption of independence.

[1] The word linear here applies to the coefficients and not to the explanatory variable. In this case, $\frac{\partial f(a_i,x_i)}{\partial ai} \neq f(a_i)$. For a nonlinear model, we have $\frac{\partial f(a_i,x_i)}{\partial ai} = f(a_i)$.

- The distribution of regression errors follows a normal distribution with mean zero and constant variance $\mathcal{N}(0, \sigma^2)$. This assumption is also known as the assumption of homoscedasticity:
$$\forall i = 1, 2, ..., n \quad \mathbb{E}(\epsilon_i) = 0, \quad \mathbb{V}(\epsilon_i) = \sigma^2$$
- Cov $[\epsilon_i, \epsilon_j] = 0$ for $i \neq j$

The basic idea behind the least squares method is to minimize the sum of squared differences (also called residuals) between the experimental data and the model we're looking at. The goal is to find the theoretical model, generally written as $f(x_i; \theta_i)$, that best fits the experimental measurements. Here, θ_i are the parameters of the model in question, which have physical significance. In the least squares method, these are determined by minimizing an objective function (also called a cost function or optimization criterion), denoted $S(\hat{\theta_1}, \hat{\theta_2})$ [2]

The optimality criterion is that of minimizing residuals. The estimates, $\hat{\theta_1}$ and $\hat{\theta_2}$, reflect the minimum of the objective function. We will illustrate this point by first considering the model: $\hat{y}_i = \theta_1 \, x_i + \theta_2$

$$S(\hat{\theta_1}, \hat{\theta_2}) = \sum_i^n e_i^2 = \sum_i^n (y_i - f(x_i; \theta_i))^2 \tag{5}$$

$$S(\hat{\theta_1}, \hat{\theta_2}) = \sum_i^n e_i^2 = \sum_i^n (y_i - \theta_1 \times x_i - \theta_2)^2 \tag{6}$$

This expression reflects the sum of all vertical distances, whose unit is that of the x-axis, between the experimental data and the theoretical model under consideration. Mathematically, this minimization is expressed as:

$$\begin{cases} \left.\dfrac{\delta S(\theta_1, \theta_2)}{\delta \theta_1}\right|_{\theta_1 = \hat{\theta}_1} = \displaystyle\sum_{i=1}^n 2\,(y_i - \hat{\theta_1} \times x_i - \hat{\theta_2}) \times (-x_i) = 0 \\ \left.\dfrac{\delta S(\theta_1, \theta_2)}{\delta \theta_2}\right|_{\theta_2 = \hat{\theta}_2} = \displaystyle\sum_{i=1}^n 2\,(y_i - \hat{\theta_1} \times x_i - \hat{\theta_2}) \times (-1) = 0 \end{cases} \tag{7}$$

$$\Rightarrow \begin{cases} 2\left[\displaystyle\sum_{i=1}^n y_i - \hat{\theta_1} \sum_{i=1}^n x_i - \hat{\theta_2}\right] \times (-x_i) = 0 \\ 2\left[\displaystyle\sum_{i=1}^n y_i - \hat{\theta_1} \sum_{i=1}^n x_i - \hat{\theta_2}\right] \times (-1) = 0 \end{cases} \tag{8}$$

[2] It should be noted that the function $S(\hat{\theta_1}, \hat{\theta_2})$ (see equation (6)) is strictly convex. It has a minimum at a single point ($\hat{\theta_1}$ and $\hat{\theta_2}$), which is obtained by setting the partial derivatives of S to zero.

$$\Rightarrow \begin{cases} 2\left[-\sum_{i=1}^{n} y_i\, x_i + \hat{\theta}_1 \sum_{i=1}^{n} x_i^2 + \hat{\theta}_2\, x_i\right] = 0 \\ 2\left[-\sum_{i=1}^{n} y_i + \hat{\theta}_1 \sum_{i=1}^{n} x_i + \hat{\theta}_2\right] = 0 \end{cases} \tag{9}$$

$$\Rightarrow \begin{cases} \hat{\theta}_1 \sum_{i=1}^{n} x_i^2 + \hat{\theta}_2\, x_i = \sum_{i=1}^{n} y_i\, x_i \\ \hat{\theta}_1 \sum_{i=1}^{n} x_i + \hat{\theta}_2 = \sum_{i=1}^{n} y_i \end{cases} \tag{10}$$

This system of equations can be written in matrix form, as follows:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{bmatrix} \times \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{11}$$

The vector of estimated parameters is obtained as follows:

$$\begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{12}$$

Remember that the inverse of a $\mathcal{M}_{2\times 2}$ matrix is simply calculated using the general formula:

$$\mathcal{M}_{2\times 2} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad \Rightarrow \quad \mathcal{M}_{2\times 2}^{-1} = \frac{1}{det(\mathcal{M})} \begin{bmatrix} D & -B \\ -C & A \end{bmatrix} \tag{13}$$
$$\text{Avec} \quad det(\mathcal{M}) = A\,D - B\,C$$

From equation (12), the slope $\hat{\theta}_1$ and the y-intercept $\hat{\theta}_2$ are determined from the formulas:

$$\hat{\theta}_1 = \frac{n \sum_{i=1}^{n} x_i\, y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \quad \text{and} \quad \hat{\theta}_2 = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i\, y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$