Introduction à Python

Jean-Christophe LOISEAU

Arts & Métiers Institute of Technology, 2021-2022

NumPy : bibliothèque (ou package) pour **Python** destinée à manipuler des **matrices** ou **tableaux multi-dimensionels** ainsi que des fonctions mathématiques opérant sur ces tableaux.

Comment calculer $\boldsymbol{x} \cdot \boldsymbol{y}$ avec $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$?

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$$

```python
def produit_scalaire(x, y):

    # Initialise la variable.
    z = 0

    # Calcul le produit scalaire.
    for i in range(len(x)):
        z = z + x[i] * y[i]

    return z
```

```
In [1]: x, y = npr.rand(100_000), npr.rand(100_000)

In [2]: %%timeit
        produit_scalaire(x, y)
```

26.9 ms ± 110 µs (mean ± std. dev. of 7 runs, 100 loops each)

```python
def produit_scalaire(x, y):
    # Calcul le produit scalaire.
    return np.sum(x * y)
```

```
In [3]: x, y = npr.rand(100_000), npr.rand(100_000)

In [4]: %%timeit
        produit_scalaire(x, y)
```

100 µs ± 1.85 µs (mean ± std. dev. of 7 runs, 10000 loops each)

```
In [5]: x, y = npr.rand(100_000), npr.rand(100_000)

In [6]: %%timeit
        np.vdot(x, y)    # Produit scalaire en NumPy.
```

25 µs ± 8.95 µs (mean ± std. dev. of 7 runs, 10000 loops each)

Comment calculer $\|\boldsymbol{A}-\boldsymbol{B}\|_2$ avec $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$?

$$\|\boldsymbol{A}-\boldsymbol{B}\|_2=\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij}-B_{ij})^2}$$

```python
def distance(A, B):
    # Taille des tableaux.
    m, n = A.shape

    # Initialise la variable.
    d = 0

    # Calcul la distance
    for i in range(m):
        for j in range(n):
            d += (A[i, j] - B[i, j])**2

    return np.sqrt(d)
```

```
In [7]: A, B = npr.rand(1000, 1000), npr.rand(1000, 1000)

In [8]: %%timeit
        distance(A, B)
```

615 ms ± 22.3 ms (mean ± std. dev. of 7 runs, 1 loop each)

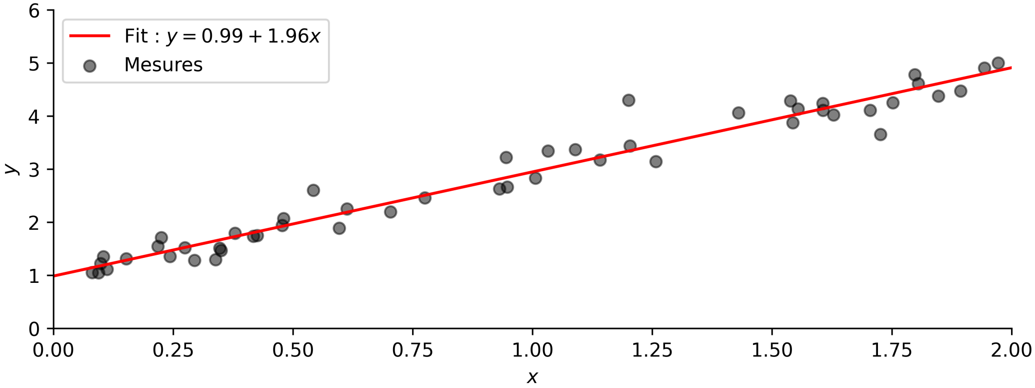
```
def distance(A, B):
    d = np.sum( (A-B)**2 )
    return np.sqrt(d)

```

```
In [9]: A, B = npr.rand(1000, 1000), npr.rand(1000, 1000)

In [10]: %%timeit
         distance(A, B)

4.23 ms ± 69.3 µs (mean ± std. dev. of 7 runs, 100 loops each)
```

```
In [11]: A, B = npr.rand(1000, 1000), npr.rand(1000, 1000)

In [12]: %%timeit
         npl.norm(A-B)      # Definition de la norme en NumPy.
```

2.98 ms ± 77.1 µs (mean ± std. dev. of 7 runs, 100 loops each)

La méthode des moindres-carrés

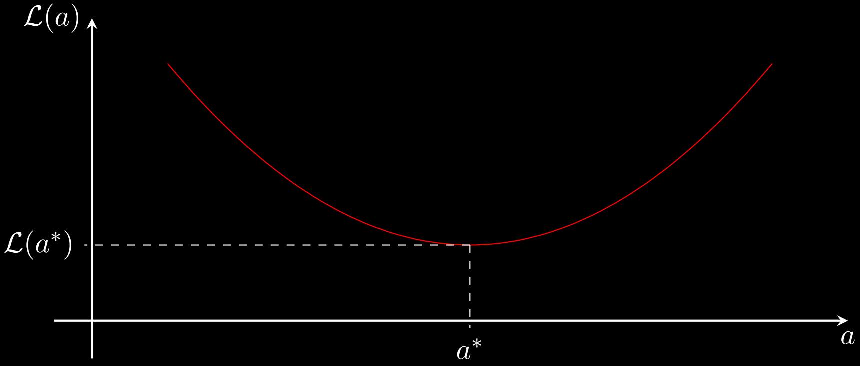

Fit : $y = 0.99 + 1.96x$
Mesures
6
5
4
3
2
1
0
y
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x

Comment trouver l'équation de la droite $\hat{y} = ax + b$
qui approxime au mieux les données $(\boldsymbol{x}, \boldsymbol{y})$?

Mesure de l'erreur : Pour des valeurs de a et b données, quelle est l'erreur faite par mon modèle, i.e. à quel point $\boldsymbol{y}$ et $\hat{\boldsymbol{y}}$ sont différents ?

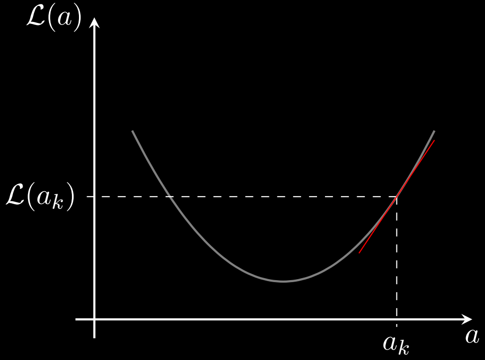
Erreur maximum	$\max_i \ \lvert y_i - \hat{y}_i \rvert$
Erreur absolue moyenne	$\frac{1}{m}\sum_{i=1}^{m} \lvert y_i - \hat{y}_i \rvert$
Erreur quadratique moyenne	$\frac{1}{m}\sum_{i=1}^{m} (y_i - \hat{y}_i)^2$

Réduction de l'erreur : Connaissant l'erreur commise par mon modèle pour des valeurs de a et b données, comment dois-je les modifier de façon à réduire cette erreur ?


$\mathcal{L}(a)$
$\mathcal{L}(a^*)$
a^*
a

$$\underset{a,b}{\text{minimiser}} \;\; \mathcal{L}(a, b)$$

$$\mathcal{L}(a_k + \epsilon) \simeq \mathcal{L}(a_k) + \epsilon \frac{\partial \mathcal{L}}{\partial a}\Big|_{a=a_k}$$



Données d'entrée : La fonction $\mathcal{L}(\boldsymbol{\theta})$ à minimiser, le vecteur initial $\boldsymbol{\theta}$, le pas d'optimisation η, la tolérance ϵ et `maxiter` le nombre maximum d'itérations possibles.

Algorithme : Descente de gradient

1. Evaluer la fonction $\mathcal{L}(\boldsymbol{\theta})$ et son gradient $\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ pour la valeur actuelle de $\boldsymbol{\theta}$.
2. Mettre à jour $\boldsymbol{\theta}$ via $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$.
3. Si $\left\| \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \right\| \leq \epsilon$, on arrête le calcul, sinon retour à l'étape 1.

Données de sortie : Les paramètres $\boldsymbol{\theta}$ du modèle minimisant l'erreur.

Erreur maximum	$\max_i \; \lvert y_i - \hat{y}_i \rvert$
Erreur absolue moyenne	$\frac{1}{m} \sum_{i=1}^{m} \lvert y_i - \hat{y}_i \rvert$
Erreur quadratique moyenne	$\frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$

Modèle : $$\hat{y}_i = ax_i + b$$

Erreur : $$\mathcal{L}(a, b) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Modèle : $$\hat{y}_i = ax_i + b$$

Erreur : $$\mathcal{L}(a,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\mathcal{L}(a,b) = \sum_{i=1}^{m}\mathcal{L}_i(a,b)$$

Modèle : $\hat{y}_i = ax_i + b$

Erreur : $$\mathcal{L}(a,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\mathcal{L}_i(a,b) = \frac{1}{m}(y_i - \hat{y}_i)^2$$

Modèle : $\hat{y}_i = ax_i + b$

Erreur : $\mathcal{L}(a,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$

$$\mathcal{L}_i(a,b) = \frac{1}{m}(y_i - ax_i - b)^2$$

Modèle : $$\hat{y}_i = ax_i + b$$

Erreur : $$\mathcal{L}(a,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\frac{\partial \mathcal{L}_i}{\partial a} = -\frac{2}{m}\,(y_i - ax_i - b)\,x_i$$

Modèle : $$\hat{y}_i = ax_i + b$$

Erreur : $$\mathcal{L}(a,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\frac{\partial \mathcal{L}_i}{\partial b} = -\frac{2}{m}(y_i - ax_i - b)$$

Modèle : $$\hat{y}_i = ax_i + b$$

Erreur : $$\mathcal{L}(a,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\frac{\partial \mathcal{L}}{\partial a} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}_i}{\partial a}$$

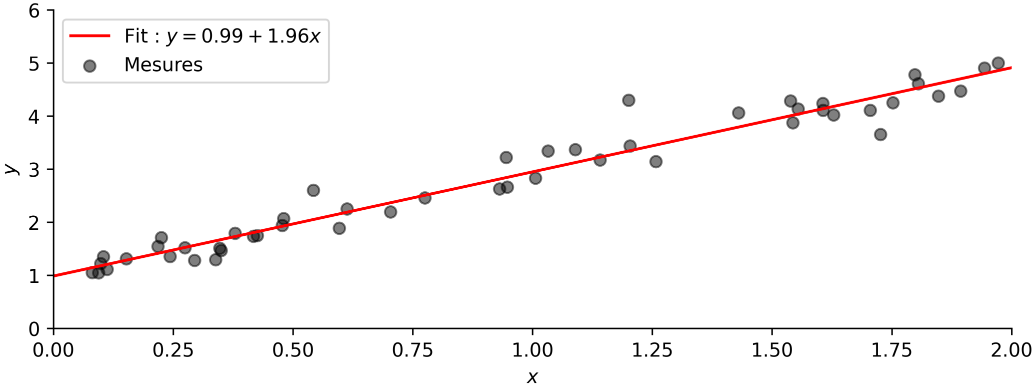
Objectif du TP : Reformuler le code de calcul qui vous est fourni en utilisant au mieux les bonnes pratiques `NumPy` qui vous ont été présentées.

```
In [13]: x, y = donnes_synthetiques()

In [14]: %%timeit
         moindres_carres(x, y, eta=0.1)
```

2.98 ms ± 77.1 µs (mean ± std. dev. of 7 runs, 100 loops each)

```
In [15]: %%timeit
         moindres_carres_opt(x, y, eta=0.1)
```

2.98 ms ± 77.1 µs (mean ± std. dev. of 7 runs, 100 loops each)



Fit : $y = 0.99 + 1.96x$
Mesures
0
1
2
3
4
5
6
y
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x

Pour en savoir plus, rendez-vous sur `https://numpy.org/`