ML : Introduction to Python

TP0 Apprentissage automatique

Introduction à Python

Jérôme Buisine
jerome.buisine@univ-littoral.fr

22 septembre 2022

Durée : 1h30

L'objectif de ce TP est la prise en main de Python et de l'environnement de travail. L'utilisation de Python est très courante pour l'apprentissage automatique : dans ce TP nous allons voir pour en maîtriser les bases.

1 Travail

Ce support vous invite à configurer votre environnement afin de ne pas avoir de problèmes de dépendances et de versions de Python. Ensuite, quelques fonctions et usages utiles de Python seront proposés.

Voici un ensemble de ressources qui peuvent vous être utiles :

— 1. Documentation officielle de l'outil `Git` ;
— 2. Gitlab : interface web pour la gestion de projets versionnés sous `Git` ;
— 3. pyenv : permet la gestion d'environnement Python ;
— 4. matplotlib : librairie Python qui permet un affichage rapide de données ;
— 5. pandas : librairie Python qui permet de lire des données et les traiter ;
— 6. jupyter : un outil de travail permettant une interaction rapide et visuelle avec une console Python.

2 Configuration de l'environnement

2.1 Création du projet

Tout d'abord, depuis l'interface Gitlab il vous faudra **créer** un projet **privé** suivant la convention de nommage : « M1-IA-YOUR_NAME ». avec **`YOUR_NAME`** composé de la première de votre prénom puis votre nom. Exemple, pour Jean Dupont, on obtient : **`jdupont`**, soit « M1-IA-jdupont ». Il s'agit du projet sur lequel vous allez travailler pour tous vos TPs. Me rajouter ensuite en tant que rapporteur du projet : `jbuisine`. Puis, clonez ce projet localement dans le dossier de votre choix.

Important : vous devez à ce stade configurer un environnement de travail propre de Python. Pour cela, référez-vous à la documentation suivante : environnement Python. Elle vous permet la mise en place d'un environnement virtuel Python. Nous utiliserons une version 3.8.0 de Python et nommerons l'environnement virtuel `ml-venv`.

2.2 Installation des dépendances

Nous allons utiliser quelques librairies pour ce projet. Tout d'abord, mettez à jour `pip`. Puis ajoutez les dépendances suivantes dans un fichier nommé `requirements.txt` et installez-les :

```
numpy==1.23.2
pandas==1.4.4
matplotlib==3.5.3
jupyterlab==3.4.5
scikit-learn==1.1.2
```

Créez un dossier à la racine de votre projet nommé `tp0-python_introduction`. Dans ce dossier et depuis votre terminal, lancez les commandes suivantes :

```
# spécifie l'environnement virtuel Python à Jupyter
ipython kernel install --user --name=ml-venv
# lance l'application Jupyter
jupyter-lab
```

Remarque : pour chaque section suivante, il vous sera demandé de créer un notebook depuis Jupyter. Faites bien attention à la sélection de l'environnement Python lors de sa création. De plus, il vous sera demandé de bien documenter votre code. Vous pouvez également utiliser des cellules de type `Markdown` plutôt que de type `code`.

3 Génération de données

Créez un nouveau notebook nommé « data_generation.ipynb ». Dans ce notebook nous allons gérer la génération de données à l'aide du package `random` Python.

Commencez par réaliser un import des différents packages à utiliser :

```
1 # import des librairies avec un alias
2 import pandas as pd
3 import numpy as np
4
5 # import du module pyplot de matplotlib avec l'alias plt
6 import matplotlib.pyplot as plt
```

3.1 Génération de nombres aléatoires et gestion des listes

Pour cette première partie, vous pouvez vous aider de la documentation suivante sur les listes : https://python.sdv.univ-paris-diderot.fr/04_listes/.

Tâche 1 : Utilisez la fonction `randint` de la librairie `random` pour générer 10 notes d'étudiants aléatoires entre 0 et 20 que vous stocker dans une liste.

Proposez deux manières de réaliser cette tâche :

— À partir d'une boucle : à l'aide de la fonction `range` ;

— À partir d'une liste en compréhension (exemple : fonctionnement d'une liste en compréhension).

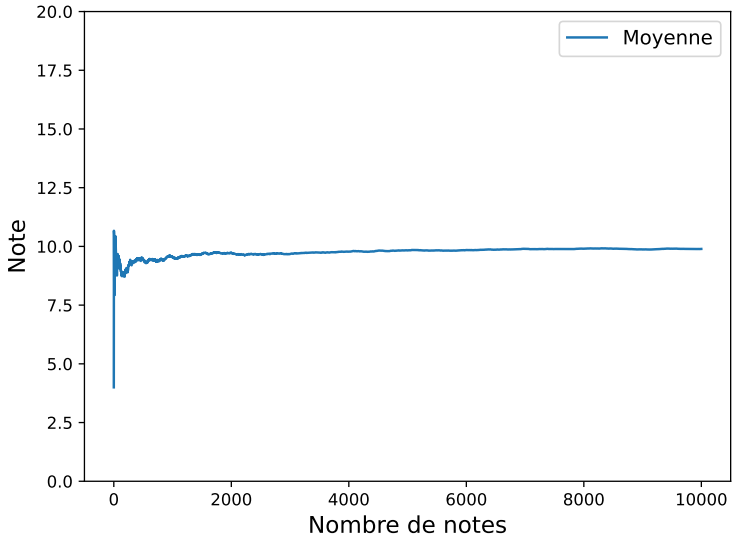
Tâche 2 : Calculez de manière itérative la moyenne obtenue pour toute nouvelle note aléatoire ajoutée :

— Vous pouvez définir une fonction qui calcule la moyenne à partir d'une liste ;

— Affichez ensuite l'évolution des moyennes successives obtenues pour chaque nouvelle note ajoutée, de 1 à 10000, à l'aide de la librairie `matplotlib` ;

— On fixe la graine aléatoire (`seed`) de la librairie `random` à 20.

Voici ce que vous devriez obtenir :



Remarque : étant donné l'utilisation d'une loi uniforme, on se rend compte que plus le nombre de valeurs sont présentes pour calculer la moyenne empirique, plus cette moyenne tend vers 10. De manière générale, on appelle cela, la loi des grands nombres :

$$\lim_{n \to +\infty} \frac{1}{n} \sum_{i=0}^{n} x_i \approx \mu$$

où la moyenne calculée se rapproche de l'espérance de la loi de probabilité X (aussi notée μ, la moyenne), plus le nombre de valeurs obtenues est élevé.

3.2 Gestion des indices d'une liste

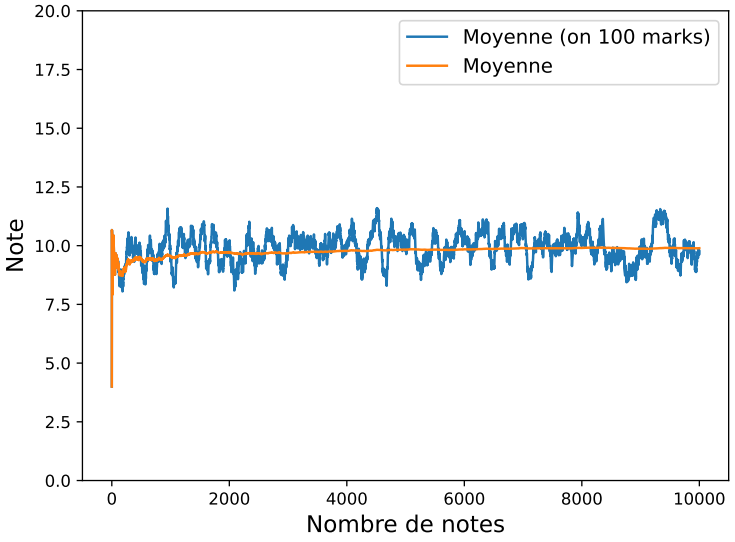
Il est également important de connaître l'utilisation des indices d'une liste. Notamment pour accès à des éléments particulier de cette liste.

Par exemple :

```python
# création d'une liste
l = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

l[0] # returns 10
l[0:2] # returns [10, 9]
l[5:] # returns [5, 4, 3, 2, 1]
l[-1] # returns [1]
l[-3:] # returns [3, 2, 1]
```

Tâche 3 : en vous inspirant de ces exemples, ajoutez la moyenne calculée sur les 100 dernières notes obtenues (en fixant de nouveau la graine aléatoire à 20). De cette manière vous devez obtenir :

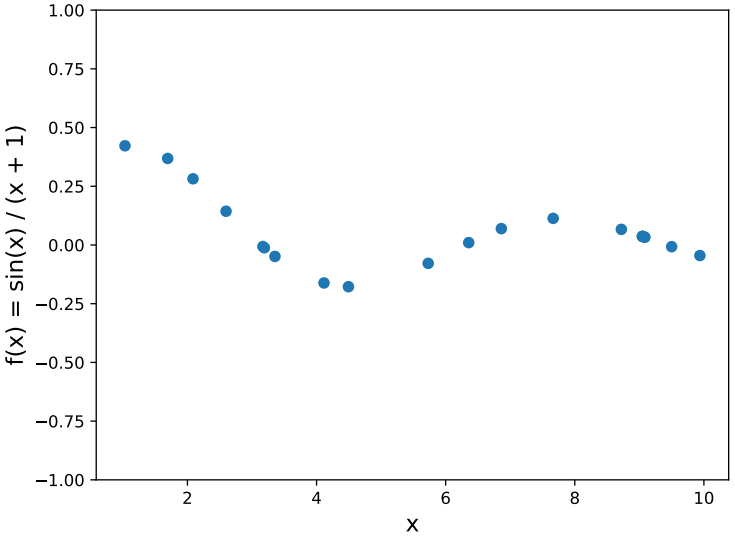


3.3 Données aléatoires d'une fonction

Tâche 4 : Vous allez écrire une fonction qui pour toute donnée x fournit $f(x) = \frac{sin(x)}{x+1}$ (utiliser la librairie `math`).

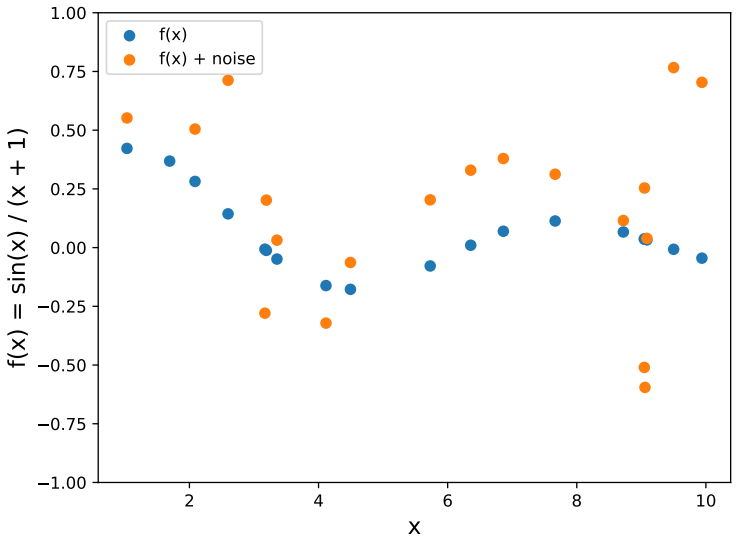
Générez ensuite 20 points de cette fonction pour des valeurs aléatoires (d'une loi uniforme) de x dans l'intervalle $[0, 10]$. La seed peut-être de nouveau fixée à 20.

Vous devez obtenir quelque assez proche de :



Tâche 5 : Vous allez ajouter un bruit gaussien de moyenne 0 et de variance 0.5 à chaque valeur $f(x)$ calculée.

Si l'on visualise le bruit ajouté, on devrait obtenir :



Remarque : la notion de bruit est assez courante car un bruit est généralement présent dans les données récoltées. Un capteur soumis à un contexte extérieur récolte assez souvent des données composées de bruit.

4 Lecture de base de données

Dans la suite du TP, nous allons nous initier à l'utilisation de la librairie *pandas* pour de la lecture de données. Créez un notebook « data_reading.ipynb » pour ces travaux.

Tâche 6 : Récupérez la base de données opossum. Ajoutez-la dans un dossier `datasets` relatif à ce TP.

À l'aide la librairie `pandas`, lire cette base de données (voir méthode d'ouverture d'un fichier relative à son extension), puis affichez quelques valeurs de cette base à l'aide de la méthode `head`.

Vous devriez obtenir :

	sex	age	hdlngth	skullw	totlngth	taill	footlgth	earconch	eye	chest	belly
0	m	1.0	85.9	52.4	80.5	35.0	62.0	42.4	14.1	25.5	30.0
1	m	1.0	86.7	52.6	84.0	38.0	62.3	44.8	15.0	23.5	30.5
2	m	1.0	85.8	50.0	81.0	36.5	62.8	43.0	14.8	22.0	28.5
3	m	1.0	86.5	51.0	81.0	36.5	63.0	44.3	13.2	23.0	28.0
4	m	1.0	88.6	54.7	82.5	39.0	64.4	48.0	14.0	25.0	33.0

Tâche 7 : Il est possible d'accéder aux valeurs des lignes et colonnes de différentes manières en traitant notre « dataframe ». Familiarisez-vous avec les commandes suivantes :

```
# df est notre dataframe
len(df.index) # nombre de lignes
df.columns # liste les colonnes
df["age"] # retourne une Serie pandas
df["age"].values # retourne les valeurs de la colonne

```

```python
df.iloc[0] # récupère la première lignes

# identique à df["age"] dans notre cas
# (accès à toutes les lignes de la colonne 0)
df.iloc[:, 0]

# vérifie si une colonne contient au moins une valeur nulle
df.isnull().any()

# supprime les lignes avec des valeurs manquantes (null ou NaN)
df = df.dropna(subset=['age'])
```

Tâche 8 : Supprimez les lignes avec des valeurs manquantes pour vous assurer que le dataframe est propre.

Afficher ensuite la matrice de correlation des colonnes entre-elles à l'aide de la méthode `corr`.

Vous pouvez également afficher graphiquement cette matrice :

```python
im = plt.imshow(matrix, cmap=plt.cm.RdBu)
plt.colorbar(im)

pticks = np.arange(len(matrix.columns))
plt.xticks(ticks=pticks, labels=matrix.columns, rotation=45)
plt.yticks(ticks=pticks, labels=matrix.columns)
```

Le coefficient de correlation permet de mesurer le lien entre certains descripteurs. Sa valeur est comprise entre -1 et 1 :

- Entre 0 et 1 : on observe une corrélation positive (lorsqu'une variable change, l'autre variable change dans la même direction) ;
- 0 : aucune corrélation ;
- Entre 0 et -1 : on observe une corrélation négative (lorsqu'une variable change, l'autre variable change dans la direction opposée).

Tâche 9 : À partir de cette matrice de corrélation, identifiez 5 descripteurs qui semblent être intéressants pour donner des indications sur la largeur du crâne d'un opossum (`skullw`).

5 Remise des travaux

N'oubliez pas de réaliser un commit de vos travaux. Taguez également votre projet avec le tag « `tp0` » et soumettez-le sur le serveur Gitlab. Il fera office de rendu.