Apprentissage-Automatisé

Release 1.0.0-beta0

IVIA-AF Team

Jul 30, 2023

Contents

1 Pré-requis **1**
1.1 Langage Python et ses Librairies 1
1.2 Les Bases Mathématiques pour l'Apprentissage Automatique 3

2 Bibliography **7**

Bibliography **9**

1 | Pré-requis

Python est le langage de programmation préféré des Data Scientistes. Ils ont besoin d'un langage facile à utiliser, avec une disponibilité décente des bibliothèques et une grande communauté. Les projets ayant des communautés inactives sont généralement moins susceptibles de mettre à jour leurs plates-formes. Mais alors, pourquoi Python est populaire en Data Science ?

Python est connu depuis longtemps comme un langage de programmation simple à maîtriser, du point de vue de la syntaxe. Python possède également une communauté active et un vaste choix de bibliothèques et de ressources. Comme résultat, vous disposez d'une plate-forme de programmation qui est logique d'utiliser avec les technologies émergentes telles que l'apprentissage automatique (Machine Learning) et la Data Science.

1.1 Langage Python et ses Librairies

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts quand on fait de l'apprentissage automatique et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plate-formes.

1.1.1 Installation de Python et Anaconda

L'installation de Python peut-être un vrai challenge. Déjà il faut se décider entre les versions 2.X et 3.X du langage, par la suite, choisir les librairies nécessaires (ainsi que les versions compatibles) pour faire de l'apprentissage automatique (Machine Learning); sans oublier les subtilités liées aux différents Systèmes d'exploitation (Windows, Linux, Mac…) qui peuvent rendre l'installation encore plus *"douloureuse"*.

Dans cette partie nous allons installer pas à pas un environnement de développement Python en utilisant Anaconda[^1]. A l'issue de cette partie, nous aurons un environnement de développement fonctionnel avec les librairies (packages) nécessaires pour faire de l'apprentissage automatique (Machine Learning).

Qu'est ce que Anaconda ?

L'installation d'un environnement Python complet peut-être assez complexe. Déjà, il faut télécharger Python et l'installer, puis télécharger une à une les librairies (packages) dont on a besoin. Parfois, le nombre de ces librairies peut-être grand. Par ailleurs, il faut s'assurer de la compatibilité entre les versions des différents packages qu'on a à télécharger. *Bref, ce n'est pas amusant*!

Alors Anaconda est une distribution Python. À son installation, Anaconda installera Python ainsi qu'une multitude de packages dont vous pouvez consulter la liste[1]. Cela nous évite de nous ruer dans les problèmes

[1] https://docs.anaconda.com/anaconda/packages/pkg-docs/#Python-3-7

d'incompatibilités entre les différents packages. Finalement, Anaconda propose un outil de gestion de packages appelé Conda. Ce dernier permettra de mettre à jour et installer facilement les librairies dont on aura besoin pour nos développements.

Téléchargement et Installation de Anaconda

Note: Les instructions qui suivent ont été testées sur Linux/Debian. Le même processus d'installation pourra s'appliquer pour les autres systèmes d'exploitation.

Pour installer Anaconda sur votre ordinateur, vous allez vous rendre sur le site officiel[2] depuis lequel l'on va télécharger directement la dernière version d'Anaconda. Prenez la version du binaire qu'il vous faut :

- Choisissez le système d'exploitation cible (Linux, Windows, Mac, etc…)
- Sélectionnez la version 3.X (à l'heure de l'écriture de ce document, c'est la version 3.8 qui est proposée, surtout pensez à toujours installer la version la plus récente de Python), compatible (64 bits ou 32 bits) avec l'architecture de votre ordinateur.

Après le téléchargement, si vous êtes sur Windows, alors rien de bien compliqué double cliquez sur le fichier exécutable et suivez les instructions classique d'installation d'un logiciel sur Windows.

Si par contre vous êtes sur Linux, alors suivez les instructions qui suivent:

- Ouvrez votre terminal et rassurez vous que votre chemin accès est celui dans lequel se trouve votre fichier d'installation.
- Exécutez la commande: $ bash Anaconda3-2020.02-Linux-x86_64.sh, rassurez vous du nom du fichier d'installation, il peut changer selon la version que vous choisissez.

Après que l'installation se soit déroulée normalement, éditez le fichier caché **.bashrc** pour ajouter le chemin d'accès à Anaconda. Pour cela exécutez les commandes suivantes:

- $ cd ~
- $ gedit .bashrc
- Ajoutez cette commande à la dernière ligne du fichier que vous venez d'ouvrir
- export PATH= ~/anaconda3/bin:$PATH

Maintenant que c'est fait, enregistrez le fichier et fermez-le. Puis exécutez les commandes suivantes

- $ conda init
- $ Python

Pour ce qui est de l'installation sur Mac, veuillez suivre la procédure d'installation dans la documentation d'Anaconda[3].

Il existe une distribution appelée Miniconda[4] qui est un programme d'installation minimal gratuit pour conda. Il s'agit d'une petite version bootstrap d'Anaconda qui inclut uniquement conda, Python, les packages dont ils dépendent, et un petit nombre d'autres packages utiles.

Terminons cette partie en nous familiarisant avec quelques notions de la programmation Python.

Première utilisation de Anaconda

La distribution Anaconda propose deux moyens d'accéder à ses fonctions: soit de manière graphique avec Anaconda-Navigator, soit en ligne de commande (depuis Anaconda Prompt sur Windows, ou un terminal pour

[2] http://docs.anaconda.com/anaconda/navigator/
[3] https://docs.anaconda.com/anaconda/install/mac-os/
[4] https://docs.conda.io/en/latest/miniconda.html

Linux ou MacOS). Sous Windows ou MacOs, démarrez Anaconda-Navigator dans le menu des programmes. Sous Linux, dans un terminal, tapez la commande : $ anaconda-navigator (cette commande est aussi disponible dans le prompt de Windows). Anaconda-Navigator propose différents services (déjà installés, ou à installer). Son onglet Home permet de lancer le service désiré. Les principaux services à utiliser pour développer des programmes Python sont :

- Spyder
- IDE Python
- Jupyter notebook et jupyter lab : permet de panacher des cellules de commandes Python (code) et des cellules de texte (Markdown).

Pour la prise en main de Python nous allons utiliser jupyter lab.

1.1.2 Prise en main de Python

Nous avons préparé un notebook qui nous permettra d'aller de zèro à demi Héros en Python. Le notebook se trouve ici[5].

1.2 Les Bases Mathématiques pour l'Apprentissage Automatique

Dans cette section, nous allons présenter les notions mathématiques essentielles à l'apprentissage automatique (machine learning). Nous n'aborderons pas les théories complexes des mathématiques afin de permettre aux débutants (en mathématiques) ou mêmes les personnes hors du domaine mais intéressées à l'apprentissage automatique de pouvoir en profiter.

1.2.1 Algèbre linéaire et Analyse

Définition d'espaces vectoriels. Un espace vectoriel est un triplet $(V, +, *)$ formé d'un ensemble V muni de deux lois,

$$\begin{aligned} + : \quad & V \times V \longrightarrow V \\ & (u, v) \mapsto u + v \\ \text{et} & \\ * : \quad & \mathbb{K} \times V \longrightarrow V, \text{ avec } \mathbb{K} \text{ un corps commutatif} \\ & (\lambda, v) \mapsto \lambda * v = \lambda v \end{aligned} \tag{1.2.1}$$

qui vérifient:

1. associativité de $+ : \forall\, u, v, w \in V, \quad (u + v) + w = u + (v + w)$
2. commutativité de $+ : \forall\, u, v \in V, \quad u + v = v + u$
3. existence d'élément neutre pour $+ : \exists\, e \in V : \forall\, u \in V, \quad u + e = e + u = u$
4. existence d'élément opposé pour $+ : \forall\, u \in V, \exists\, v \in V : u + v = v + u = 0$. On note $v = -u$ et v est appelé l'opposé de u
5. existence de l'unité pour $* : \exists\, e \in \mathbb{K}$ tel que $\forall\, u \in V, \quad e * u = u$

[5] https://colab.research.google.com/drive/1zILtNrCmPDFyQQ1Ev1H4jeHx7FuyEZ27?usp=sharing

6. associativité de $* : \forall\, (\lambda_1, \lambda_2, u) \in \mathbb{K} \times \mathbb{K} \times V, \quad (\lambda_1 \lambda_2) * u = \lambda_1 * (\lambda_2 * u)$
7. somme de vecteurs (distributivité de * sur +) : $\forall\, (\lambda, u, v) \in \mathbb{K} \times V \times V, \quad \lambda * (u + v) = \lambda * u + \lambda * v$
8. : $\forall\, (\lambda_1, \lambda_2, u) \in \mathbb{K} \times \mathbb{K} \times V, \quad (\lambda_1 + \lambda_2) * u = \lambda_1 * u + \lambda_2 * u.$

Remarque 1: Les éléments de V sont appelés des **vecteurs**, ceux de $\mathbb{K}$ sont appelés des **scalaires** et l'élément neutre pour $+$ est appelé **vecteur nul**. Finalement, V est appelé $\mathbb{K}$-espace vectoriel ou espace vectoriel sur $\mathbb{K}$.
Base d'un espace vectoriel. Soit V un $\mathbb{K}$-espace vectoriel. Une famille de vecteurs
$\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ est appelée base de V si les deux propriétés suivantes sont satisfaites:

- $\forall\, u \in V, \exists\, c_1, \ldots, c_n \in \mathbb{K}$ tels que $u = \sum_{i=1}^{n} c_i b_i$ (On dit que $\mathcal{B}$ est **une famille génératrice** de V).
- $\forall\, \lambda_1, \ldots, \lambda_n \in \mathbb{K}, \quad \sum_{i=1}^{n} \lambda_i b_i = 0 \Longrightarrow \lambda_i = 0 \quad \forall\, i.$ (On dit que les éléments de $\mathcal{B}$ sont *linéairement indépendants*).

Lorsque $u = \sum_{i=1}^{n} c_i b_i$, on dit que $c_1, \ldots, c_n$ sont les coordonnées de u dans la base $\mathcal{B}$. Si de plus aucune confusion n'est à craindre, on peut écrire:

$$\mathbf{u} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}. \tag{1.2.2}$$

Définition. Le nombre d'éléments dans une base d'un espace vectoriel est appelé **dimension** de l'espace vectoriel.

NB: Un espace vectoriel ne peut être vide (il contient toujours le vecteur nul). L'**espace vectoriel nul** $\{0\}$ n'a pas de base et est **de dimension nulle**. Tout **espace vectoriel non nul** de dimension finie admet une infinité de bases mais sa **dimension est unique**.
Exemples d'espaces vectoriels: Pour tous $n, m \geq 1$, l'ensemble des matrices $\mathcal{M}_{nm}$ à coefficients réels et l'ensemble $\mathbb{R}^n$ sont des $\mathbb{R}$-espace vectoriels. En effet, il est très facile de vérifier que nos exemples satisfont les huit propriétés énoncées plus haut. Dans le cas particulier $V = \mathbb{R}^n$, toute famille d'exactement n vecteurs linéairement indépendants en est une base. En revanche, toute famille de moins de n vecteurs ou qui contient plus que n vecteurs ne peut être une base de $\mathbb{R}^n$.

Matrices: Soit $\mathbb{K}$ un corps commutatif. Une matrice en mathématiques à valeurs dans $\mathbb{K}$ est un tableau de nombres, où chaque nombre est un élément de $\mathbb{K}$. Chaque ligne d'une telle matrice est un vecteur (élément

d'un $\mathbb{K}$-espace vectoriel). Une matrice est de la forme:

$$\mathbf{M} = \begin{bmatrix} a_{11} & a_{12} \dots a_{1n} \\ a_{21} & a_{22} \dots a_{2n} \\ \vdots & \\ a_{m1} & a_{m2} \dots a_{mn} \end{bmatrix}. \tag{1.2.3}$$

On note aussi $\mathbf{M} = \big(a_{ij}\big)_{1 \leq i \leq m, 1 \leq j \leq n}$.

La matrice ci-dessus est carrée si $m = n$. Dans ce cas, la suite $[a_{11}, a_{22}, \dots, a_{mm}]$ est appelée **diagonale** de M. Si tous les coefficients hors de la diagonale sont zéro, on dit que la matrice est diagonale. Une matrice avec tous ses coefficients nuls est dite matrice **nulle**.

Produit de matrices. Soient $\mathbf{A} = \big(a_{ij}\big)_{1 \leq i \leq m, 1 \leq j \leq n}, \mathbf{B} = \big(b_{ij}\big)_{1 \leq i \leq n, 1 \leq j \leq q}$ deux matrices.
On définit le produit de **A** par **B** et on note $\mathbf{A} \times \mathbf{B}$ ou simplement **AB**, la matrice M définie par:

$$\mathbf{M}_{ij} = \sum_{\ell=1}^{n} a_{i\ell} b_{\ell j}, \text{ pour tout } i \text{ et } j. \tag{1.2.4}$$

Important.

- Le produit **AB** est possible si et seulement si le nombre de colonnes de **A** est égal au nombre de lignes de **B**.
- Dans ce cas, **AB** a le même nombre de lignes que **A** et le même nombre de colonnes que **B**.
- Un autre point important à noter est que le produit matriciel n'est pas commutatif (**AB** n'est pas toujours égal à **BA**).

Exemple. Soient les matrices **A** et **B** définies par:

$$\mathbf{A} = \begin{bmatrix} 2 & -3 & 0 \\ 5 & 11 & 5 \\ 1 & 2 & 3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 3 \\ -5 & 1 \\ 1 & 2 \end{bmatrix}, \mathbf{A} + \mathbf{B} = \begin{bmatrix} 1 & 3 \\ -5 & 1 \\ 1 & 2 \end{bmatrix} \tag{1.2.5}$$

Le nombre de colonnes de la matrice **A** est égal au nombre de lignes de la matrice **B**.

$$\mathbf{AB} = \begin{bmatrix} 2 \times 1 + (-3) \times (-5) + 0 \times 1 & 2 \times 3 + (-3) \times 1 + 0 \times 2 \\ 5 \times 1 + 11 \times (-5) + 5 \times 1 & 5 \times 3 + 11 \times 1 + 5 \times 2 \\ 1 \times 1 + 2 \times (-5) + 3 \times 1 & 1 \times 3 + 2 \times 1 + 3 \times 2 \end{bmatrix} = \begin{bmatrix} 17 & 3 \\ -45 & 33 \\ -6 & 11 \end{bmatrix}. \tag{1.2.6}$$

Le produit **BA** n'est cependant pas possible.

Somme de matrices et multiplication d'une matrice par un scalaire.
La somme de matrices et multiplication d'une matrice par un scalaire se font coefficients par coefficients.
Avec les matrice $\mathbf{A}, \mathbf{B}$ de l'exemple précédent, et $\mathbf{C} = \begin{bmatrix} -2 & -7 & 3 \\ 5 & 10 & 5 \\ 12 & 9 & 3 \end{bmatrix}$, on a:

$$\mathbf{A}+\mathbf{C}=\begin{bmatrix}2+(-2) & -3+(-7) & 0+3\\ 5+5 & 11+10 & 5+5\\ 1+12 & 2+9 & 3+3\end{bmatrix}=\begin{bmatrix}0 & -10 & 3\\ 10 & 21 & 10\\ 13 & 11 & 6\end{bmatrix}, \text{ et pour tout } \lambda\in\mathbb{R}, \quad \lambda\mathbf{B}=\begin{bmatrix}\lambda & 3\lambda\\ -5\lambda & \lambda\\ \lambda & 2\lambda\end{bmatrix}. \tag{1.2.7}$$

2 | Bibliography

The breakthrough of deep learning origins from (Krizhevsky *et al.*, 2017) for computer vision, there is a rich of following up works, such as (He *et al.*, 2016). NLP is catching up as well, the recent work (Devlin *et al.*, 2018) shows significant improvements.

Two keys together (Devlin *et al.*, 2018, He *et al.*, 2016). Single author , two authors Newell and Rosenbloom (1980)

Bibliography

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Newell, A., & Rosenbloom, P. S. (1980). *Mechanisms of skill acquisition and the law of practice.* CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.