# Piscine Machine Learning | Jour 1

## Dataset loading et
## Introduction à MatPlotLib

Nom de repo: ml_d01

Droits: florian.bacho@epitech.eu, voravo_d

Langage : Python

## Introduction

Le Machine Learning est une branche de l'intelligence artificielle dont le principe est de donner à un ordinateur/agent la capacité d'apprendre à effectuer une tâche sans être explicitement programmé. Il permet de résoudre des problèmes trop complexes pour être exprimés en terme de formules et de conditions (reconnaissance d'image, contrôle moteur, classification, ect...).

Il existe 2 types majeurs d'apprentissage en Machine Learning : l'apprentissage supervisé et l'apprentissage non supervisé.

En apprentissage supervisé, on dit qu'un agent « apprend » lorsqu'il est capable d'améliorer son comportement à partir de l'observation de données fournies. On appelle cet ensemble d'observations un « dataset ». Il contient des paires entrées / sorties_attendues. L'agent peut corriger son comportement à partir de la différence entre ses estimations et les résultats attendus dans le dataset (feedbacks).

Tandis qu'en apprentissage non supervisé, l'agent va chercher le meilleur comportement sans connaître son environnement dans lequel il est plongé. L'agent n'a donc pas besoin d'exemples pour apprendre.

Durant cette semaine, nous allons nous intéresser à l'apprentissage supervisé. Nous verrons aujourd'hui comment charger et visualiser des données, puis dans les jours suivants comment les utiliser à travers deux algorithmes:

1) La méthode des K plus proches voisins: intuitive et simple à implémenter, elle va vous permettre d'avoir une première intuitions des espaces à dimensions multiples (nommés « hyperplans »).

2) Les réseaux de neurones artificiels: plus complexes mais également plus puissants.

# Exercice 00 :

Nom du fichier: DatasetLoader.py

En apprentissage supervisé, vous voulez que votre agent apprenne à partir d'exemples que l'on fournit. Dans la grande majorité des cas, les exemples d'entraînement se trouvent dans un fichier représentant les paires Entrées-Sorties désirées.

Durant la durée de la piscine, nous allons considérer le format suivant :

[nb_exemple] [nb_entry] [nb_out]
[entry_ex1$_1$] [entry_ex1$_2$] … [entry_ex1$_{nb_entry}$]
[out_ex1$_1$] [out_ex1$_2$] … [out_ex1$_{nb_out}$]
[entry_ex2$_1$] [entry_ex2$_2$] … [entry_ex2$_{nb_entry}$]
[out_ex2$_1$] [out_ex2$_2$] … [out_ex2$_{nb_out}$]
…
[entry_ex_nb_exemple$_1$] [entry_ex_nb_exemple$_2$] … [entry_ex_nb_exemple$_{nb_entry}$]
[out_ex_nb_exemple$_1$] [out_ex_nb_exemple$_2$] … [out_ex_nb_exemple$_{nb_out}$]

Créer une class nommée *Exemple* contenant :

- Une liste correspondant aux entrées de votre exemple

- Une liste correspondant aux sorties de votre exemple

Créer une class nommée *Dataset* contenant :

- Le nombre d'exemples

- Le nombre d'entrées

- Le nombre de sorties

- Les exemples chargés de type *Exemple*

Son constructeur doit prendre un nom de fichier en paramètre et charger les données.

**!** Inutile de faire de la gestion d'erreur de parsing. Considérez que tous les dataset fournis seront valides.

## Ex01 :

Dossier : Aucun

MatPlotLib est un outil de visualisation de données et de fonctions en python.

NumPy est une bibliothèque destinée à manipuler les matrices ou tableaux multidimensionnels.

En apprentissage automatique, il est très important de visualiser ses données, ses paramètres ou certaines fonctions assez complexes… Cette bibliothèque vous permettra de faire toutes sortes de visualisations de fonctions tout le long de cette piscine.

Installez MatPlotLib et NumPy si vous ne les avez pas.

## Ex02 :

Fichier: ex02.py

Afficher la fonction f(x) = 3x + 4 avec l'axe des abscisses définie sur [-10; 10].

## Ex03 :

Fichier: ex03.py

Afficher la fonction f(x) = 1 / x avec l'axe des abscisses définie sur [-10; 10].

> **!** Attention : 1 / 0 n'existe pas !

## Ex04 :

Fichier: ex04.py

Afficher la fonction f(x) = 1 + exp(-x) avec l'axe des abscisses définie sur [-5; 3]

On observe que f(x) = 1 + exp(-x) tend vers 1 lorsque x augmente.
En effet :

$\lim_{x \to +\infty} \exp(-x) = 0$ par conséquent : $\lim_{x \to +\infty} 1 + \exp(-x) = 1$

## Ex05 :

Fichier: ex05.py

Afficher la fonction f(x) = 1 / (1 + exp(-x)) avec l'axe des abscisses définie sur [-10; 10]

> ? f(x) peut-elle avoir un dénominateur égal à 0 ?

Cette fonction se nomme *fonction Logistique* ou *Sigmoïde*. Elle a la particularité de fournir une valeur comprise entre 0 et 1, d'être continue et dérivable sur ℝ. On remarque surtout que cette fonction sépare l'espace de X en 2: Une partie inférieure à 0.5, l'autre supérieure à 0.5. On peut interpréter cela comme du binaire: 0 lorsque Logistic(x) est inférieure 0.5, 1 lorsque Logistic(x) est supérieure à 0.5. Dans le jargon, on appelle ce genre de fonction *une fonction d'activation*.

Gardez la dans un coin de votre tête.

## Ex06 :

Fichier: ex06.py

Un dataset « xor.ds » a été fourni avec le sujet. Chargez le grâce à votre package DatasetLoader puis affichez les exemples dans un repère en 3D sous forme de points.

## Ex07 :

Fichier: Dataset.py

Lorsque vous travaillez sur des datasets à plusieurs entrées, il arrive souvent que vos variables prennent des valeurs très différentes. Prenons le cas d'un dataset contenant des caractéristiques de maisons:

| Caractéristique | Prix (euros) | Superficie (m²) | Nombre de pièces |
|---|---|---|---|
| Intervalle | 0 – 1 million | 0 - 500 | 1 - 10 |

On constate que les valeurs peuvent être très grandes pour le prix, moyennes pour la superficie, et petites pour le nombre de pièces.

Or, il se trouve qu'en apprentissage supervisé, la majorité des algorithmes ne fonctionnent pas (ou mal/lentement) si les données sont situées sur des intervalles trop différents. (Vous comprendrez dans les prochains jours pourquoi).

Il est donc très important de normaliser votre dataset: la normalisation est une méthode qui consiste à modifier les valeurs de vos variables afin de les placer sur une même échelle, relativement petite (entre -1 et 1 environ).

On peut diviser la normalisation en 3 étapes:

Pour chaque caractéristique/type d'entrée,

1) Calculer la moyenne μ de ce type d'entrée:

$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$ où m est le nombre d'exemples et $\sum_{i=1}^{m} x_i$ la somme de toutes les entrées de ce type (du 1er exemple au dernier exemple).

2) Calculer l'écart type s de ce type d'entrée:

$s = \sqrt{\sum_{i=1}^{m} (x_i - \mu)^2 / m}$ où μ est la moyenne calculée précédemment.

3) Mettre à jour cette entrée pour chaque exemple:

x = (x - μ) / s

Rajouter dans la classe *Dataset* une fonction permettant de normaliser les données chargées.

# Conclusion

L'apprentissage supervisé dépend entièrement des paires entrées-sorties afin de modifier et d'améliorer son comportement. Si les données sont incohérentes, mal chargées, ou non normalisées, il est possible et même probable que l'agent ne réussisse pas à apprendre correctement.

Charger correctement les données et les visualiser afin de s'assurer de que tout est en ordre doit donc être la première étape de tout projet d'apprentissage supervisé.

Plus généralement, la visualisation de données et de fonctions est extrêmement importante en Machine Learning. Elle vous permettra de vérifier l'implémentation de vos algorithmes, puis de comprendre où et comment optimiser l’apprentissage.