Neural-Network -Environnement de travail

☰ TOC

Environnement de travail

A. Utiliser Anaconda et Spyder sous Windows

Nous allons utiliser **Anaconda** qui est une distribution Python très utilisée en data sciences. Elle gère les environnements virtuels et intègre de nombreux packages scientifiques, largement plus complet que R Studio. En complement vous pouvez utiliser **Spyder** qui est un IDE (environnement de développement) graphique orienté science des données, inclus par défaut dans Anaconda.

- Sur les postes de travail de 3il
  - Tout est installer
- Installation Anaconda :
  - Télécharger le fichier d'installation sur le site officiel Anaconda
  - Lancer l'installeur et suivre les étapes (choisir "Just Me" recommandé).
  - Après installation, Anaconda Navigator permet de gérer graphiquement les environnements et les packages.
- Lancement de Spyder :
  - Via le menu Démarrer Windows : chercher "Anaconda Navigator", puis cliquer sur "Launch" sous Spyder.

B. Utiliser Poetry sous Ubuntu (ou WSL/Linux)

- **Poetry** est un gestionnaire moderne d'environnements et de dépendances Python, adapté au développement professionnel et reproductible.

Installation de Poetry :

1. Installer pip (si ce n'est pas déjà fait) :
   - `sudo apt update ; sudo apt install python3-pip`
2. Installer Poetry via le script officiel :
   - `curl -sSL https://install.python-poetry.org | python3 -`
3. Vérifier l'installation : `poetry --version`
4. Initialiser l'environement projet `poetry init`

Créer un projet avec Poetry :

```
poetry new mon_projet
cd mon_projet
poetry add numpy matplotlib
poetry shell  # pour activer l'environnement virtuel
```

Proposition de structure de dossier pour les TPs

Voici une structure adaptée pour des TPs Python en data science, inspirée des bonnes pratiques :

```
nom_du_tp/
├── main.py              # Interface ligne de commande avec l'application ecrite
├── src/                 # Code source Python (modules, classes, fonctions)
│   ├── __init__.py
│   └── perceptron.py    # Class perceptron
├── tests/               # Tests unitaires pour le code
│   └── test_perceptron.py
├── data/                # Données utilisées pour les TPs
│   ├── raw/             # Données brutes
│   └── processed/       # Données prétraitées
├── figures/             # Graphiques générés
├── README.md            # Rapport
├── requirements.txt     # Liste des dépendances (pour pip)
├── pyproject.toml       # Fichier de configuration Poetry (si utilisé)
└── environment.yml      # Fichier d'environnement Conda (pour vous)
```

Cette structure facilite la clarté, la réutilisation et la reproductibilité des TP.