Série d'exercices - V1

Formation Doctorale A1/A2

Chapitre 1 : La base

Exercice 1.1 : Fonction de moyenne

Objectif : Créer une fonction `calcul_moyenne()` pour calculer la moyenne d'une liste de flottants.

Énoncé :

```
# Compl ter la fonction suivante :

def calcul_moyenne(donnees: list[float]) -> float:
    ...
```

Questions :

- Que retourne la fonction si la liste est [1.0, 2.0, 3.0] ?
- Que se passe-t-il si la liste est vide ?

Exercice 1.2 : Gestion d'une liste vide

Améliorer la fonction pour retourner `None` si la liste est vide. Tester avec plusieurs jeux de données.

Exercice 1.3 : Analyse de données génomiques

Objectif : Utiliser un dictionnaire pour stocker les expressions de gènes.

```
donnees = {"BRCA1": 5200, "TP53": 3400, "EGFR": 4100}

#  crire  une fonction qui retourne le gene le plus exprime.
```

Exercice 1.4 : Avantages de Python

Rédiger un tableau comparatif Python vs. autre langage selon : simplicité, bibliothèques, communauté, vitesse.

Exercice 1.5 : Lecture critique de code scientifique

Trouver un article scientifique utilisant Python. Identifier les bibliothèques utilisées et leur but.

Exercice 1.6 : Création de tenseurs de base

Créer des tenseurs simples dans les deux frameworks.

```
# Avec TensorFlow
import tensorflow as tf
tensor_tf = tf.constant([[1, 2], [3, 4]])

# Avec PyTorch
import torch
tensor_pt = torch.tensor([[1, 2], [3, 4]])

# Afficher les tenseurs et leurs propri t s
```

Questions :

- Quelle est la différence entre `tf.constant` et `torch.tensor` ?
- Comment vérifier la forme (shape) d'un tenseur dans chaque framework ?

Exercice 1.7 : Opérations tensoriels de base

Effectuer des opérations mathématiques simples.

```
# Creer deux tenseurs 3x3
a = ...  #    completer
b = ...  #    completer

# Calculer:
# 1. Addition element par element
# 2. Multiplication matricielle
# 3. Produit element-wise (Hadamard)
```

Exercice 1.8 : Redimensionnement et broadcasting

Comprendre le reshape et le broadcasting.

```
# Partant d'un tenseur 1D de 12 elements:
data = [1, 2, 3, ..., 12]  #    completer

# 1. Le redimensionner en 3x4
# 2. Le redimensionner en 2x3x2
# 3. Effectuer une operation entre un tenseur 3x1 et un 1x3
```

Exercice 1.9.1 : Tenseurs sur GPU

Utiliser l'accélération matérielle.

```
# V rifier la disponibilite du GPU
# D placer les tenseurs sur GPU
# Comparer le temps d'ex cution CPU vs GPU pour une operation
```

Exercice 1.9.2 : Gradients automatiques

Comprendre le calcul automatique des gradients.

```
# Creer un tenseur necessitant le gradient
x = ...  # A compl ter
y = x**2 + 3*x + 1

# Calculer dy/dx
# Afficher le gradient
```

Exercice 1.9.3 : Conversion entre frameworks

Convertir des tenseurs entre TensorFlow et PyTorch.

```
# Cr er un tenseur dans un framework
# Le convertir dans l'autre framework
# V rifier que les donn es sont identiques
```

Exercice 1.9.4 : Tenseurs pour images

Manipuler des images comme des tenseurs.

```
# Charger une image (ex: avec matplotlib)
# La convertir en tenseur
# Normaliser les valeurs entre 0 et 1
# Ajouter une dimension de batch (B x H x W x C)
```

Exercice 1.9.5 : Tenseurs pour séquences

Créer des tenseurs pour des données séquentielles.

```
# A partir d'une sequence temporelle:
seq = [1, 3, 5, 7, 9, 11]

# Creer des fenetres glissantes pour l'entrainement:
# X = [[1,3], [3,5], [5,7]]
# y = [[5], [7], [9]]
```

Exercice 1.9.6 : Initialisation de tenseurs

Comparer différentes méthodes d'initialisation.

```
# Cr er des tenseurs avec:
# 1. Des zeros
# 2. Des valeurs aleatoires uniformes
# 3. Des valeurs aleatoires normales
# 4. Xavier/Glorot initialization
```

Exercice 1.9.7 : Tenseurs creux

Travailler avec des tenseurs creux (sparse).

```
# Creer un tenseur creux representant:
# - Une matrice diagonale
# - Une matrice avec peu de valeurs non nulles
# Convertir entre dense et sparse
```

Exercice 1.9.8 : Visualisation de tenseurs

Visualiser des tenseurs avec matplotlib.

```
# Creer un tenseur 3D (ex: 16x16x3)
# Afficher differentes coupes (slices)
# Visualiser les canaux separement
```

Exercice 1.9.9 : Tenseurs et pandas

Convertir entre DataFrames et tenseurs.

```
import pandas as pd

# Creer un DataFrame simple
df = pd.DataFrame(...)

# Convertir en tenseur
# Effectuer des operations
# Reconvertir en DataFrame
```

Exercice 1.9.1 : Lecture de fichier CSV

Utiliser `pandas` pour charger un fichier CSV et afficher la moyenne d'une colonne.

Exercice 1.9.2 : Présentation Jupyter

Réaliser un notebook expliquant l'utilisation de : NumPy, Plotly, Pandas, Polars.

Prob 1 : Optimisation Numérique et Algèbre Linéaire

Concepts : Décomposition matricielle, méthodes itératives, optimisation convexe.

1. **Décomposition LU avec pivot partiel**
   Implémentez une fonction `lu_decomposition(A)` qui prend une matrice carrée A et retourne P, L, U (matrice de permutation, triangulaire inférieure et supérieure). Testez-la sur une matrice mal conditionnée.
2. **Gradient Conjugué Préconditionné**
   Programmez l'algorithme du gradient conjugué préconditionné pour résoudre $Ax = b$. Utilisez un préconditionneur de votre choix (Jacobi, ILU, etc.).

3. **Méthode de Newton pour une fonction vectorielle**
   Implémentez la méthode de Newton pour minimiser $f(x) = \sum_{i=1}^{n}(x_i^2 - \cos(x_i))^2$. Analysez la convergence.
4. **SVD et PCA manuelle**
   Calculez la SVD d'une matrice de données synthétiques et projetez-les sur les 2 premières composantes principales sans utiliser `sklearn.decomposition.PCA`.
5. **Problème des moindres carrés contraints**
   Résolvez $\min \|Ax - b\|^2$ sous contrainte $Cx = d$ en utilisant la décomposition QR.
6. **Optimisation stochastique (SGD avancé)**
   Implémentez une variante de SGD avec momentum et taux d'apprentissage adaptatif (comme Adam) pour une régression logistique.
7. **Méthode de la puissance pour les valeurs propres**
   Estimez la plus grande valeur propre d'une matrice creuse en utilisant la méthode de la puissance avec décalage spectral.
8. **Problème de moindres carrés non linéaires**
   Utilisez l'algorithme de Levenberg-Marquardt pour ajuster un modèle exponentiel $y = a \cdot e^{bx} + c$ à des données bruitées.

Chapitre 2 : Fondamentaux de Python

Exercice 2.1 : Variables et types

Déclarer des variables de types `int`, `float`, `str`, `bool` et afficher leur type.

Exercice 2.2 : Conditionnelle if/else

Classifier une température comme : basse, normale, ou haute.

Exercice 2.3 : Pattern Matching

Utiliser `match/case` pour classer une note (excellent, passable, etc).

Exercice 2.4 : Boucle for

Afficher chaque mesure de [1.1, 1.5, 1.3] avec son index.

Exercice 2.5 : Boucle while

Afficher un compte à rebours de 5 à 1.

Exercice 2.6 : `deque`

Utiliser `collections.deque` pour ajouter et retirer des mesures.

Exercice 2.7 : Lecture avec `yield`

Écrire une fonction `lire_fichier()` utilisant `yield`.

Exercice 2.8 : Tri de dictionnaire

Trier un dictionnaire selon les valeurs.

Exercice 2.9 : Set de mesures

Identifier les valeurs uniques d'une liste avec `set()`.

Exercice 2.10 : Classe Expérience

Créer une classe `Experience` avec attributs `nom`, `domaine`, et une méthode `presenter()`.

Exercice 2.11 : Classe Simulation

Hériter de `Experience` et ajouter un paramètre `modèle`.

Prob 3 : Simulations Numériques et Équations Différentielles

Concepts : Méthodes de Runge-Kutta, différences finies, Monte Carlo.

1. **Méthode de Runge-Kutta d'ordre 4 (RK4)**
   Résolvez un système d'équations différentielles couplées (ex. Lotka-Volterra) avec RK4.
2. **Équation de la chaleur en 2D**
   Simulez la diffusion thermique sur une plaque carrée avec conditions aux limites mixtes (Dirichlet + Neumann).
3. **Schéma aux différences finies pour les EDP non linéaires**
   Résolvez l'équation de Burgers avec un schéma explicite et analysez la stabilité.
4. **Monte Carlo pour l'intégration haute dimension**
   Calculez $\int_{[0,1]^5} \sin(\sum x_i^2)\, dx$ avec une méthode de réduction de variance.
5. **Méthode de tir pour les problèmes aux limites**
   Résolvez $y'' = -k^2 y$ avec $y(0) = 0$, $y(1) = 1$ en utilisant la méthode de tir et Newton.
6. **Simulation de particules avec forces**
   Simulez N particules interagissant via un potentiel de Lennard-Jones avec intégration de Verlet.
7. **Équations différentielles stochastiques (Euler-Maruyama)**
   Simulez un processus d'Ornstein-Uhlenbeck et comparez avec la solution analytique.
8. **Méthode spectrale pour les EDP périodiques**
   Résolvez l'équation de Poisson 1D $u'' = f$ avec une transformée de Fourier discrète.

Prob 3 : Traitement de Données et Apprentissage Automatique Scientifique

Concepts : Feature engineering, modèles bayésiens, validation croisée.

1. **Feature engineering pour séries temporelles**
   Créez des features (moyenne mobile, FFT, etc.) pour un modèle de prédiction de données temporelles.
2. **Régression ridge avec validation croisée**
   Implémentez une validation croisée généralisée (GCV) pour optimiser le paramètre de régularisation.

3. **Processus gaussiens (GP) manuels**
   Programmez un GP avec noyau RBF pour interpoler des données 1D sans utiliser `sklearn.gaussian_process`.
4. **Clustering hiérarchique personnalisé**
   Implémentez un clustering agglomératif avec une métrique de linkage personnalisée.
5. **Sélection de modèle par critère d'information**
   Comparez des modèles polynomiaux avec le critère AIC/BIC sur des données synthétiques.
6. **Bayesian Optimization pour hyperparamètres**
   Utilisez une approche bayésienne (avec GP) pour optimiser les hyperparamètres d'un SVM.
7. **Autoencodeur pour réduction de dimension**
   Construisez un autoencodeur (en PyTorch/TensorFlow) pour compresser des images scientifiques (ex. MNIST).
8. **Validation croisée imbriquée**
   Évaluez un pipeline complet (prétraitement + modèle) avec une validation croisée imbriquée pour éviter le biais.

Prob 4 : Compression d'image par SVD

Comprendre la compression d'image par décomposition en valeurs singulières (SVD).

1. Lecture et affichage

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

img = Image.open("image.png").convert("L")
A = np.array(img, dtype=np.float32)
plt.imshow(A, cmap='gray')
plt.title("Image originale")
plt.show()
```

2. Faire la Décomposition et Reconstruction

3. Affichager le compressé et l'erreur

4. Trouver le taux de compression

Questions :

— Que se passe-t-il si on diminue k ?
— Quelle est la qualité visuelle perçue ?
— Quel compromis choisir pour un article scientifique ?

Notes pédagogiques

— Requiert une maîtrise avancée de `numpy`, `scipy`, et éventuellement `pytorch/tensorflow`
— Certaines questions nécessitent une lecture préalable des algorithmes
— Durée adaptable selon l'expérience du doctorant (2.5 H/séance)