Exemple de questions de programmation, PHYS-231, 2024/25

January 22, 2025

Questions de programmation

1. La fonction `f` suivante prend en entrée deux tableaux numpy bidimensionnels `a` et `b`. Expliquez brièvement ce qu'elle retourne. Proposez une ligne de code qui effectue le même calcul de manière plus efficace.

```
def f(a, b):
  K, L = a.shape
  M, N = b.shape
  assert L == N
  r = []
  for k in range(K):
    r.append([])
    for m in range(M):
      t = 0
      for l in range(L):
        t += a[k, l] * b[m, l]
      r[k].append(t)
  return r
```

> **Solution:** `f` calcule le produit matriciel ab^T. Cela peut être réalisé de manière plus efficace en utilisant la ligne de code suivante : `np.dot(a, b.T)`.

2. L'extrait de code suivant implémente la mise à jour des paramètres W_1 et W_2 d'un modèle après une itération de descente de gradient.

```
W_2 -= learning_rate * gradient_W1(W_1, W_2)
W_1 -= learning_rate * gradient_W2(W_1, W_2)
```

En supposant que les deux fonctions `gradient_W1` et `gradient_W2` calculent correctement les gradients et que les variables sont initialisées de manière appropriée, ce code est-il correct ? Justifiez brièvement.

> **Solution:** Le code n'est pas correct, car W_1 devrait être mis à jour en utilisant l'ancienne valeur de W_2, et non sa valeur après un pas de gradient.

3. Le code suivant calcule les $k = 2$ vecteurs propres de X associés aux k plus grandes valeurs propres. Complétez la dernière ligne.

```
import numpy as np

matrix = np.random.randn(10, 10)

eigenvalues, eigenvectors = np.linalg.eig(matrix)

sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvalues_sorted = eigenvalues[sorted_indices]
eigenvectors_sorted = eigenvectors[:, sorted_indices]

k = 2
eigenvectors_top_k = ...
```

Indication : `[::-1]` inverse l'ordre d'une liste ou d'un tableau.

Solution:

```
eigenvectors_top_k = eigenvectors_sorted[:, :k]
```

4. La fonction suivante calcule le déterminant d'une matrice de manière récursive :

```
import numpy as np

def determinant_recursive(matrix):
    if len(matrix) == 1:
        return matrix[0][0]
    if len(matrix) == 2:
        return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]
    det = 0
    for i in range(len(matrix)):
        minor = matrix[1:, :i] + matrix[1:, i+1:]
        det += ((-1) ** i) * matrix[0][i] * determinant_recursive(minor)
    return det
```

Une erreur est rencontrée lorsque vous utilisez la fonction ci-dessus. Quelle ligne pose problème ? Expliquez brièvement pourquoi.

Solution: La ligne

`minor = matrix[1:, :i] + matrix[1:, i+1:]`

provoque une erreur. En effet, les matrices extraites n'ont pas les bonnes tailles pour être additionnées.

Remarque additionnelle : l'idée ici est que les tableaux doivent être concaténés, par exemple en utilisant `np.concatenate`.