Tutoriel : Installation de PyTorch sur Python

Master GBM

March 26, 2025

1 Introduction

PyTorch est une bibliothèque populaire pour le calcul scientifique et l'apprentissage automatique. Ce tutoriel explique comment installer PyTorch sur Python en utilisant `pip` ou `conda`, ainsi que les étapes pour vérifier l'installation.

2 Pré-requis

Avant de commencer, assurez-vous que :

- Vous avez installé Python (version 3.x). Vérifiez avec la commande suivante :

```
python --version
```

- Vous disposez de `pip` ou `conda`. Vérifiez leur version avec :

```
pip --version
conda --version
```

3 Installation avec pip

Étapes :

1. Créez un environnement virtuel (recommandé) :

```
python -m venv pytorch_env
source pytorch_env/bin/activate    % Sur Linux/MacOS
pytorch_env\Scripts\activate       % Sur Windows
```

2. Installez PyTorch :

- Pour une installation CPU uniquement :

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
```

- Pour une installation GPU avec CUDA (exemple : CUDA 11.8) :

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
```

3. Vérifiez l'installation : Lancez Python et exécutez le code suivant :

```
import torch
x = torch.rand(2, 3)
print(x)
```

4 Installation avec conda

Étapes :

1. Créez un environnement conda :

```
conda create -n pytorch_env python=3.10 -y
conda activate pytorch_env
```

2. Installez PyTorch : Rendez-vous sur le site officiel de PyTorch ou utilisez les commandes suivantes :

- Pour une installation CPU uniquement :

```
conda install pytorch torchvision torchaudio cpuonly -c pytorch
```

- Pour une installation GPU avec CUDA (exemple : CUDA 11.8) :

```
conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
```

3. Vérifiez l'installation : Lancez Python et exécutez ce code pour confirmer que PyTorch fonctionne correctement :

```
import torch
print(torch.cuda.is_available())  # Renvoie True si CUDA est disponible.
```

5 Dépannage

Si vous rencontrez des problèmes lors de l'installation, vérifiez que votre version de Python est compatible avec PyTorch. Assurez-vous également que vos pilotes NVIDIA et CUDA sont correctement installés si vous utilisez une version GPU.

6 Premiers pas avec Pytorch

Introduction

L'objectif de ce TP est de pratiquer les manipulations de tenseurs de base avec PyTorch, de comprendre la relation entre un tenseur et son stockage sous-jacent, et d'appréhender l'efficacité des calculs basés sur les tenseurs par rapport à leurs équivalents implémentés avec des boucles Python.

—

7 Multiples vues d'un stockage

Générez la matrice suivante sans utiliser de boucle Python :

$$\begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 \\ 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 \\ 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 \\ 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 & 3 & 3 & 1 & 2 & 1 \\ 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \end{bmatrix}$$

Indice : Utilisez les fonctions `torch.full` et l'opérateur de slicing.

—

8 Décomposition spectrale (valeurs propres)

Sans utiliser de boucles Python, créez une matrice carrée M (un tenseur bidimensionnel) de dimension 20×20, remplie avec des coefficients aléatoires suivant une distribution gaussienne. Calculez ensuite les valeurs propres de la matrice suivante :

$$M^{-1} \cdot \text{diag}(1, ..., 20)M$$

où $\text{diag}(1, ..., 20)$ est une matrice diagonale contenant les valeurs 1 à 20.

Indice : Utilisez les fonctions suivantes : `torch.empty`, `torch.normal`, `torch.arange`, `torch.diag`, `torch.mm`, `torch.inverse`, et `torch.linalg.eig`.

—

9 Flops par seconde

Générez deux matrices carrées de dimension 5000×5000, remplies avec des coefficients aléatoires suivant une distribution gaussienne. Calculez leur produit matriciel, mesurez le temps nécessaire à cette opération, et estimez combien d'opérations en virgule flottante ont été exécutées par seconde (le résultat devrait être en milliards ou dizaines de milliards).

Indice : Utilisez les fonctions suivantes : `torch.empty`, `torch.normal`, `torch.mm`, et `time.perf_counter`.

—

10 Manipulation des strides

Écrivez une fonction nommée `mul_row`, utilisant uniquement des boucles Python (sans opérateurs de slicing), qui prend un tenseur bidimensionnel en argument et retourne un nouveau tenseur de même taille. Dans ce nouveau tenseur, la première ligne est identique à celle d'origine, la deuxième ligne est multipliée par 2, la troisième par 3, etc.

Ensuite, écrivez une deuxième version appelée `mul_row_fast`, utilisant uniquement des opérations sur les tenseurs.

Appliquez ces deux versions à une matrice de taille 1000×400 et mesurez le temps que chaque version prend (il devrait y avoir une différence d'au moins deux ordres de grandeur).

Indice : Utilisez le broadcasting ainsi que les fonctions suivantes : `torch.arange`, `torch.view`, `torch.mul`, et `time.perf_counter`.

—

Conclusion

Ce TP0 vous permet d'explorer différentes manipulations efficaces avec PyTorch, tout en comparant leur performance avec des implémentations naïves en Python. Ces exercices illustrent l'importance d'utiliser des bibliothèques optimisées pour tirer parti des capacités matérielles modernes.

—