Numpy Array và Pytorch/Tensorflow Tensor

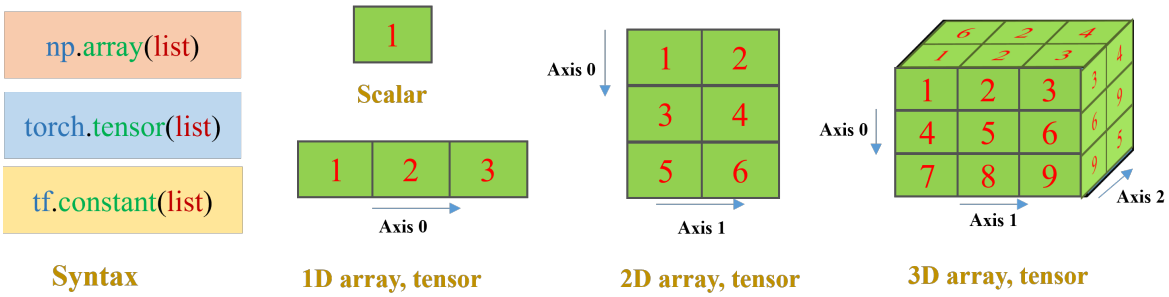
Dinh-Tiem Nguyen và Quang-Vinh Dinh

1. Mô tả

Array trong Numpy và Tensor trong các thư viện Pytorch, Tensorflow là những thành phần nền tảng cốt lõi cho việc tính toán của các thư viện này. Trong đó:

- **Array** là một cấu trúc dữ liệu đa chiều mạnh mẽ, giúp lưu trữ và thực hiện các phép toán toán học trên dữ liệu số. Array có thể là mảng 0 chiều (scalar), mảng một chiều (vector), mảng hai chiều (ma trận), hoặc nhiều chiều hơn.
- **Tensor trong Pytorch và Tensorflow** là một cấu trúc dữ liệu nhiều chiều, tương tự như array trong Numpy, nhưng được thiết kế để tương thích với học sâu và tính toán song song, đặc biệt là trên các thiết bị như GPU và TPU. Trong Pytorch và Tensorflow, tensors là đơn vị cơ bản để lưu trữ và xử lý dữ liệu.

Có nhiều cách để tạo tensor (hay array trong Nympy), ta có thể sử dụng cú pháp sau đây để khởi tạo chúng từ list Python.



Hình 1: Minh họa và cú pháp tạo Array và Tensor.

Khi làm việc với cấu trúc dữ liệu tensor, ta có thể kiểm tra thuộc tính của chúng thông qua một số phương thức sau:

- **shape** cho biết kích thước của mảng hoặc tensor, tức là số phần tử trong mỗi chiều của chúng.
- **dtype** cho biết kiểu dữ liệu của các phần tử trong mảng hoặc tensor.
- **type** cho biết kiểu của đối tượng mảng hoặc tensor.
- **device** chỉ có ở trong Pytorch và Tensorflow, và nó cho biết nơi lưu trữ tensor, có thể là CPU hoặc GPU.

2. Bài tập

Câu 1: Hãy viết chương trình tạo Numpy array, Tensorflow tensor, Pytorch tensor từ danh sách 1 chiều?

Câu 2: Hãy viết chương trình tạo Numpy array, Tensorflow tensor, Pytorch tensor từ danh sách 2 chiều. Sau đó thực hiện kiểm tra thuộc tính shape, dtype, type, device từ các array, tensor vừa tạo ?

3. Đáp án

Có nhiều cách khác nhau để ta tạo array hay tensor. Cách đầu tiên, ta tạo chúng từ list-kiểu dữ liệu quen thuộc trong Python.

```python
import numpy as np
import torch
import tensorflow as tf

# Tạo một danh sách 1 chiều
list_1D = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Tạo một mảng 1 chiều từ danh sách
arr_1D = np.array(list_1D)
print("Mảng 1 chiều NumPy:\n",
      arr_1D, type(arr_1D))

# Tạo một tensor PyTorch từ danh sách
tensor_1D_pt = torch.tensor(list_1D)
print("Tensor PyTorch 1 chiều:\n",
      tensor_1D_pt)

# Tạo một tensor TensorFlow từ danh sách
tensor_1D_tf = tf.convert_to_tensor(
                                    list_1D)
print("Tensor TensorFlow 1 chiều:\n",
      tensor_1D_tf)
```

```
================ Output ===============
Mảng 1 chiều NumPy:
 [1 2 3 4 5 6 7 8 9]
 <class 'numpy.ndarray'>

Tensor PyTorch 1 chiều:
 tensor([1, 2, 3, 4, 5, 6, 7, 8, 9])

Tensor TensorFlow 1 chiều:
 tf.Tensor([1 2 3 4 5 6 7 8 9],
           shape=(9,), dtype=int32)
=======================================
```

Ví dụ trên thực hiện tạo array, tensor từ một list Python. Để tạo array chúng ta dùng cú pháp **np.array(list_1D)**, với pytorch thì ta dùng **torch.tensor(list_1D)** và với Tensorflow ta dùng **tf.convert_to_tensor(list_1D)** hoặc **tf.constant(list_1D)**.

Đối với việc tạo array, tensor từ list 2D, chúng ta cũng thực hiện tương tự cách trên, Ví dụ Numpy:

```python
# NumPy code
import numpy as np
# Tạo một danh sách 2 chiều
list_2D = [[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9]]
# Tạo một mảng 2 chiều
arr_2D = np.array(list_2D)
# Kiểm tra hình dạng, kiểu dữ liệu và loại
print("Hình dạng của mảng: ",
      arr_2D.shape)
print("Kiểu dữ liệu của mảng: ",
      arr_2D.dtype)
print("Loại của mảng: ",
      type(arr_2D))
print("Mảng:\n", arr_2D)
```

```
================ Output ===============
Hình dạng của mảng:  (3, 3)

Kiểu dữ liệu của mảng:  int32

Loại của mảng:  <class 'numpy.ndarray'>

Mảng:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
=======================================
```

Trong đoạn mã Numpy trên, chúng ta bắt đầu bằng việc tạo ra một danh sách 2D **(list_2D)** đại diện cho một ma trận có 3 hàng và 3 cột. Sau đó, chúng ta sử dụng Numpy để chuyển danh sách này thành một mảng 2 chiều **(arr_2D)**. Dòng mã tiếp theo sử dụng các hàm tích hợp trong Numpy để in ra hình dạng (**shape**), kiểu dữ liệu (**dtype**), và kiểu của mảng (**type**). Cuối cùng, chúng ta in ra nội dung của mảng để kiểm tra kết quả

Ví dụ Pytorch:

```python
# PyTorch code
import torch

# Tạo một danh sách 2 chiều
list_2D = [[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9]]
# Tạo một tensor 2 chiều
tensor_2D_pt = torch.tensor(list_2D)

# Kiểm tra hình dạng, kiểu dữ liệu, loại,
    thiết bị lưu trữ của tensor
print("Hình dạng của tensor: ",
      tensor_2D_pt.shape)
print("Kiểu dữ liệu của tensor: ",
      tensor_2D_pt.dtype)
print("Loại của tensor: ",
      type(tensor_2D_pt))
print("Thiết bị lưu trữ của tensor: ",
      tensor_2D_pt.device)
print("Tensor:\n", tensor_2D_pt)
```

```
================ Output ===============
Hình dạng của tensor:  torch.Size([3, 3])

Kiểu dữ liệu của tensor:  torch.int64

Loại của tensor:  <class 'torch.Tensor'>

Thiết bị lưu trữ của tensor:  cpu

Tensor:
 tensor([[1, 2, 3],
         [4, 5, 6],
         [7, 8, 9]])
=======================================
```

Trong đoạn mã Pytorch trên, chúng ta bắt đầu bằng cách tạo một danh sách 2D (**list_2D**) đại diện cho ma trận 3x3. Sau đó, chúng ta sử dụng Pytorch để chuyển đổi danh sách này thành tensor 2 chiều (**tensor_2D_pt**).
Dòng mã tiếp theo sử dụng các thuộc tính tích hợp trong Pytorch để in ra hình dạng (**shape**), kiểu dữ liệu (**dtype**), kiểu của tensor (**type**), và thiết bị lưu trữ của tensor (**device**). Cuối cùng, chúng ta in ra nội dung của tensor để kiểm tra kết quả.
Ví dụ Tensorflow:

```python
import tensorflow as tf

# Tạo một danh sách 2 chiều
list_2D = [[1, 2, 3],
           [4, 5, 6],
           [7, 8, 9]]
# Tạo một tensor 2 chiều từ danh sách
tensor_2D_tf = tf.convert_to_tensor(
                                list_2D)

# Kiểm tra hình dạng, kiểu dữ liệu, loại,
# và thiết bị mà tensor được lưu trữ
print("Hình dạng của tensor: ",
      tensor_2D_tf.shape)
print("Kiểu dữ liệu của tensor: ",
      tensor_2D_tf.dtype)
print("Loại của tensor: ",
      type(tensor_2D_tf))
print("Thiết bị mà tensor được lưu trữ: ",
      tensor_2D_tf.device)
print(tensor_2D_tf)
```

```
================ Output ===============
Hình dạng của tensor:  (3, 3)

Kiểu dữ liệu của tensor:  <dtype: 'int32'>

Loại của tensor:  <class 'tensorflow.
    python.framework.ops.EagerTensor'>

Thiết bị mà tensor được lưu trữ:  /job:
    localhost/replica:0/task:0/device:CPU
    :0

tf.Tensor(
[[1 2 3]
 [4 5 6]
 [7 8 9]], shape=(3, 3), dtype=int32)
=======================================
```

Tương tự như Pytorch, trong đoạn mã Tensorflow trên, chúng ta bắt đầu bằng cách tạo một danh sách

2D (**list_2D**) đại diện cho ma trận 3x3. Sau đó, chúng ta sử dụng Tensorflow để chuyển đổi danh sách này thành một tensor 2 chiều (**tensor_2D_tf**) bằng hàm **tf.convert_to_tensor()**.
Dòng mã tiếp theo sử dụng các thuộc tính tích hợp trong Tensorflow để in ra hình dạng (**shape**), kiểu dữ liệu (**dtype**), kiểu của tensor (**type**), và thiết bị lưu trữ của tensor (**device**). Cuối cùng, chúng ta in ra nội dung của tensor để kiểm tra kết quả.