Các Hàm Khởi Tạo Numpy Array và Pytorch/Tensorflow Tensor - Phần 1

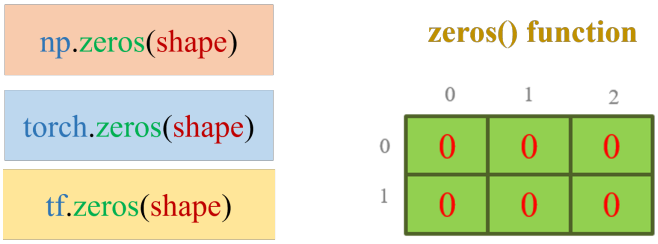
Dinh-Tiem Nguyen và Quang-Vinh Dinh

1. Mô tả

Khi lập trình với các thư viện Numpy, Pytorch, Tensorflow có một số cách để nhanh chóng tạo ra các array với giá trị, kích thước khác nhau. Trong bài tập này, chúng ta sẽ tìm hiểu các sử dụng 3 hàm **zeros**, **ones**, **full**.

a) ***zeros***

zeros là hàm thực hiện chức năng tạo array, tensor toàn giá trị 0 với đầu vào là kích thước và kiểu dữ liệu ta muốn. Cả 3 thư viện đều sử dụng hàm trên, với cú pháp tương tự nhau.

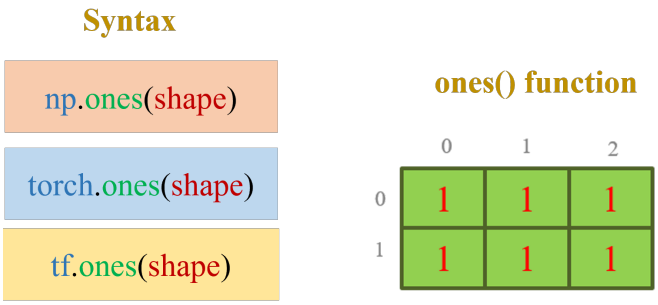


Hình 1: Minh họa và cú pháp sử dụng hàm zeros.

Nhìn chung, hàm **zeros** chủ yếu dùng để khởi tạo mảng hoặc tensor với các giá trị 0, thường được dùng trong quá trình xây dựng các mô hình máy học và thực hiện các phép toán số học.

b) ***ones***

Tương tự với **zeros**, hàm **ones** tạo mảng chứa toàn số 1 với đầu vào là kích thước do người dùng chỉ định.

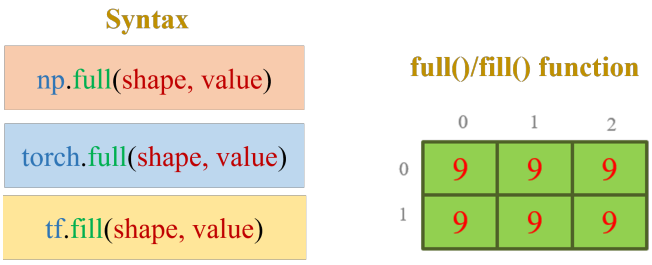


Hình 2: Minh họa và cú pháp sử dụng hàm ones.

c) ***full, fill***

Hàm **full** giúp chúng ta tạo array, tensor(Pytorch) với phần tử là giá trị chỉ định, đầu vào của hàm **full** gồm kích thước array và giá trị hằng số muốn tạo. Khác với hai thư viện trên, Tensorflow sử dụng hàm **fill**.

Hàm **full** với Numpy, Pytorch hay **fill** với Tensorflow giúp tạo array hoặc tensor với kích thước và giá trị được chỉ định, có ích khi ta cần khởi tạo các cấu trúc dữ liệu với giá trị đồng nhất.



Hình 3: Minh họa và cú pháp sử dụng hàm full/fill.

2. Bài tập

Câu 1: Hãy viết chương trình sử dụng hàm zeros tạo Numpy array, Tensorflow tensor, Pytorch tensor chỉ chứa giá trị là số 0 với kích thước (3, 4)?

Câu 2: Hãy viết chương trình sử dụng hàm ones tạo Numpy array, Tensorflow tensor, Pytorch tensor chỉ chứa giá trị là số 1 với kích thước (3, 4)?

Câu 3: Hãy viết chương trình tạo Numpy array, Tensorflow tensor, Pytorch tensor chỉ chứa giá trị là số 5 với kích thước (3, 4)?

3. Đáp án

Câu 1: ***zeros***

Chương trình sau tạo array với kích thước (3, 4), đối với thư viện Numpy sử dụng cú pháp **np.zeros**, bên trong hàm này chúng ta truyền vào kích thước array chúng ta mong muốn là (3, 4).

```
# Numpy code
import numpy as np

# Tạo array toàn số 0
arr_zeros = np.zeros((3, 4))
print(arr_zeros)
```

```
================ Output ===============
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

=======================================
```

Tương tự như thư viện Numpy, Pytorch sử dụng cú pháp ***torch.zeros***

```
#Pytorch code
import torch

# Tạo tensor toàn số 0
tensor_zeros = torch.zeros((3, 4))
print(tensor_zeros)
```

```
================ Output ===============
tensor([[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]])

=======================================
```

Tương tự, Tensorflow sử dụng cú pháp **tf.zeros**

```
# Tensorflow code
import tensorflow as tf

# Tạo tensor toàn số 0
tensor_zeros = tf.zeros((3, 4))
print(tensor_zeros)
```

```
================ Output ===============
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]], shape=(3, 4), dtype=
   float32)
=======================================
```

Câu 2: ***ones***

Chương trình sau tạo array với kích thước (3, 4), đối với thư viện Numpy sử dụng cú pháp ***np.ones***

```python
# Numpy code
import numpy as np

# Tạo array toàn số 1
arr_ones = np.ones((3, 4))
print(arr_ones)
```

```
================ Output ===============
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]

=======================================
```

Với Pytorch và Tensorflow cú pháp thực hiện tương tự với cú pháp torch.ones, tf.ones

```python
#PyTorch code
import torch

# Tạo tensor toàn số 1
tensor_ones = torch.ones((3, 4))
print(tensor_ones)
```

```
================ Output ===============
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])

=======================================
```

```python
#TensorFlow code
import tensorflow as tf


# Tạo tensor toàn số 1
tensor_ones = tf.ones((3, 4))
print(tensor_ones)
```

```
================ Output ===============
tf.Tensor(
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]], shape=(3, 4), dtype=
   float32)
=======================================
```

Câu 3: ***full, fill***

Chương trình sau tạo array với kích thước (3, 4), giá trị hằng số là 5. Đối với thư viện Numpy sử dụng cú pháp ***np.full***

```python
# Numpy code
import numpy as np

# Tạo array toàn số 5
arr_full = np.full((3, 4), 5)
print(arr_full)
```

```
================ Output ===============
[[5 5 5 5]
 [5 5 5 5]
 [5 5 5 5]]

=======================================
```

Tương tự như thư viện Numpy, Pytorch sử dụng cú pháp ***torch.full***

```python
#Pytorch code
import torch

# Tạo tensor toàn số 5
tensor_full = torch.full((3, 4), 5)
print(tensor_full)
```

```
================ Output ===============
tensor([[5, 5, 5, 5],
        [5, 5, 5, 5],
        [5, 5, 5, 5]])

=======================================
```

Khác với hai thư viện trên, Tensorflow sử dụng hàm **fill** để tạo một tensor với giá trị được chỉ định.

```python
# Tensorflow code
import tensorflow as tf

# Tạo tensor toàn số 5
tensor_full = tf.fill((3, 4), 5)
print(tensor_full)
```

```
================ Output ===============
tf.Tensor(
[[5 5 5 5]
 [5 5 5 5]
 [5 5 5 5]], shape=(3, 4), dtype=int32)
=======================================
```