Các Hàm Khởi Tạo Numpy Array và Pytorch/Tensorflow Tensor - Phần 2

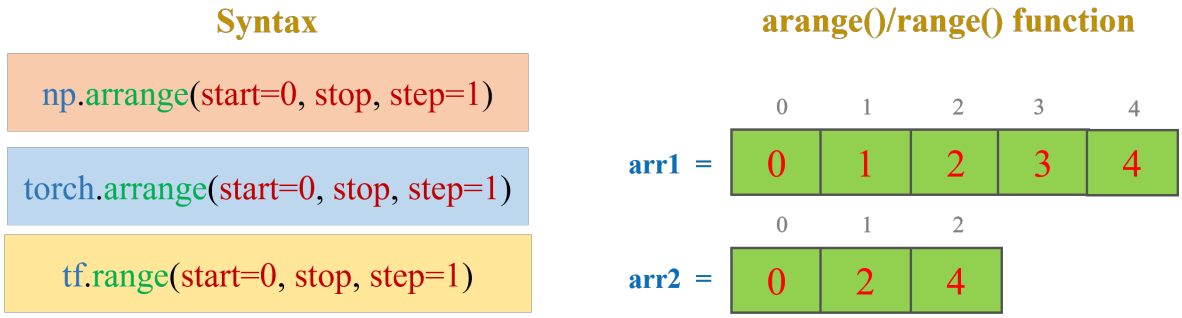
Dinh-Tiem Nguyen và Quang-Vinh Dinh

1. Mô tả

Trong bài tập này, chúng ta tiếp tục tìm hiểu các hàm có chức năng tạo array, tensor đặc biệt. Trong bài tập này chúng ta sẽ tìm hiểu và sử dụng các hàm **arrange**, **range**, **eye**, **random**.

a) ***arrange, range***

Trong Numpy, Pytorch, hàm **arange** được sử dụng để tạo một mảng chứa dãy số có giá trị tăng dần với khoảng cách cố định. Trong Tensorflow chúng ta sẽ sử dụng hàm **range**. Cú pháp như sau:

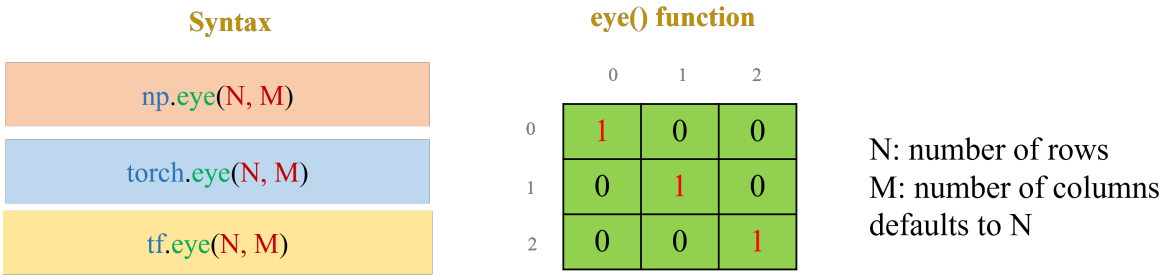


Hình 1: Minh họa và cú pháp sử dụng hàm arrange, range.

Hàm **arange** (hoặc **range** trong Tensorflow) rất hữu ích khi ta muốn tạo ra một dãy số tăng dần với các giá trị cách đều nhau.

b) ***eye***

Hàm **eye** được sử dụng để tạo ma trận đơn vị, tức là một ma trận vuông có tất cả các phần tử trên đường chéo chính có giá trị 1, các phần tử còn lại là 0. Cú pháp sử dụng như sau:



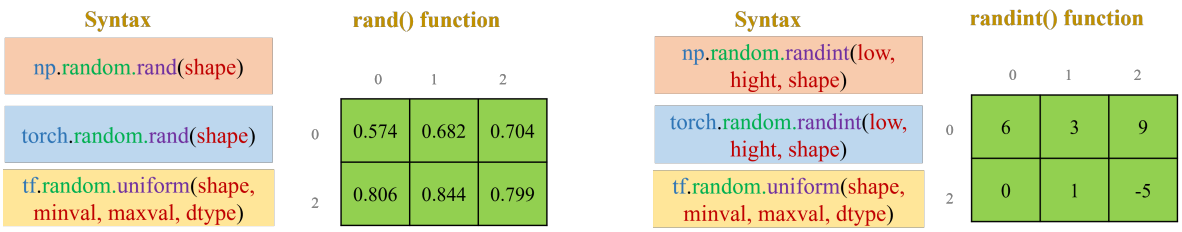
Hình 2: Minh họa và cú pháp sử dụng hàm eye.

Hàm **eye** giúp tạo ma trận hoặc tensor đơn vị, đặc biệt hữu ích trong nhiều ứng dụng toán học và xử lý dữ liệu.

c) ***random***

Trong thư viện Numpy, Pytorch cung cấp hàm **random.rand** để tạo mảng với các giá trị ngẫu nhiên trong khoảng [0.0, 1.0) với kích cỡ (shape) được chỉ định. Ngoài ra có thể sử dụng hàm **random.randint**

để tạo mảng với các số nguyên ngẫu nhiên. Đối với thư viện Tensorflow chúng ta sử dụng hàm **random.uniform**.



Hình 3: Minh họa và cú pháp sử dụng hàm random.

Các hàm này giúp việc tạo dữ liệu ngẫu nhiên trở nên dễ dàng trong quá trình phát triển mô hình Machine Learning, Deep Learning.

2. Bài tập

Câu 1: Hãy viết chương trình tạo Numpy array, Tensorflow tensor, Pytorch tensor trong khoảng [0, 10) với step=1?

Câu 2: Hãy viết chương trình tạo Numpy array, Tensorflow tensor, Pytorch tensor là ma trận đơn vị với kích thước (3, 3).

Câu 3: Hãy viết chương trình tạo Numpy array, Tensorflow tensor, Pytorch tensor ngẫu nhiên trong khoảng giá trị [0, 1] với kích thước (3, 4). Tiếp theo hãy tạo array, tensor với các giá trị là số nguyên trong khoảng [-10, 10]. Lưu ý trong bài tập này, các bạn sẽ sử dụng seed = 2024 để dảm bảo ra kết quả giống đáp án, cách sử dụng như sau:

```
# Numpy code
import numpy as np
np.random.seed(2024)

# Pytorch code
import torch
torch.manual_seed(2024)

# Tensorflow code
import tensorflow as tf
tf.random.set_seed(2024)
```

3. Đáp án

Câu 1: ***arrange, range***

Trong numpy chúng ta sử dụng hàm **np.arange**

```
# Numpy code
import numpy as np

# Tạo array trong khoảng [0, 10),
# bước nhảy 1
arr_arange = np.arange(10)
print(arr_arange)
```

```
================ Output ===============
[0 1 2 3 4 5 6 7 8 9]
=======================================
```

Trong PyTorh, để tạo tensor chứa dãy số, ta sử dụng hàm **torch.arange**:

```
#Pytorch code
import torch

# Tạo tensor trong khoảng [0, 10),
# bước nhảy 1
tensor_arange = torch.arange(10)
print(tensor_arange)
```

```
================ Output ===============
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])



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

Trong Tensorflow, ta có thể sử dụng hàm **tf.range** để thực hiện chức năng tương tự:

```
# Tensorflow code
import tensorflow as tf

# Tạo tensor trong khoảng [0, 10),
# bước nhảy 1
tensor_arange = tf.range(10)
print(tensor_arange)
```

```
================ Output ===============
tf.Tensor([0 1 2 3 4 5 6 7 8 9],
shape=(10,), dtype=int32)



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

Câu 2: ***eye***

Trong numpy chúng ta sử dụng hàm np.eye để tạo ma trận đơn vị, trong đó giá trị chúng ta truyền vào là số lượng hàng của ma trận đầu ra, số lượng cột mặc định bằng số lượng hàng, nếu muốn số lượng cột khác bạn cần truyền vào hàm eye số lượng cột chỉ định.

```
# Numpy code
import numpy as np

# Tạo array với đường chéo chính là 1,
# còn lại là 0
arr_eye = np.eye(3)
print(arr_eye)
```

```
================ Output ===============
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]


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

Trong Pytorch, để tạo tensor đơn vị, ta sử dụng hàm **torch.eye**:

```
#Pytorch code
import torch

# Tạo tensor với đường chéo chính là 1,
# còn lại là 0
tensor_eye = torch.eye(3)
print(tensor_eye)
```

```
================ Output ===============
tensor([[1., 0., 0.],
        [0., 1., 0.],
        [0., 0., 1.]])


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

Trong Tensorflow, hàm **eye** cũng được sử dụng để tạo constant tensor đơn vị:

```
# Tensorflow code
import tensorflow as tf

# Tạo tensor với đường chéo chính là 1,
# còn lại là 0
tensor_eye = tf.eye(3)
print(tensor_eye)
```

```
================ Output ===============
tf.Tensor(
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]], shape=(3, 3), dtype=float32)

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

Câu 3: ***random***

Trong Numpy, chúng ta sẽ sử dụng hàm random.rand để tạo mảng với các giá trị ngẫu nhiên trong khoảng giá trị mặc định là [0, 1). Để tạo mảng với giá trị ngẫu nhiên là số nguyên chúng ta dùng random.randint, khi sử dụng hàm này ta cần truyền vào khoảng giá trị lấy ngẫu nhiên và kích thước đầu ra mong muốn.

```python
#Numpy code
import numpy as np

# Đặt seed để đảm bảo kết quả ngẫu nhiên
# có thể tái tạo được
np.random.seed(2024)

# Tạo một mảng với các giá trị ngẫu nhiên
    trong khoảng [0, 1) với kích thước (3,
     4)
arr_rand = np.random.rand(3, 4)
print("Mảng ngẫu nhiên trong khoảng
    [0, 1):\n", arr_rand)

# Tạo một mảng với các giá trị ngẫu nhiên
    trong khoảng [-10, 10)
arr_randint = np.random.randint(-10, 10,
    size=(3, 4))
print("Mảng ngẫu nhiên trong khoảng
[-10, 10):\n", arr_randint)
```

```
================ Output ===============
Mảng ngẫu nhiên trong khoảng [0, 1):
 [[0.58801452 0.69910875 0.18815196
   0.04380856]
 [0.20501895 0.10606287 0.72724014
   0.67940052]
 [0.4738457  0.44829582 0.01910695
   0.75259834]]

Mảng ngẫu nhiên trong khoảng [-10, 10):
 [[ 5  1 -3  8]
 [-1 -4  0 -9]
 [-5  9 -6 -2]]

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

Trong Pytorch, các hàm tương tự cũng có sẵn thông qua module torch. **torch.rand** tạo tensor với các giá trị ngẫu nhiên từ phân phối đều trong khoảng [0.0, 1.0), **torch.randint** dùng để tạo tensor với các số nguyên ngẫu nhiên trong một khoảng cụ thể.

```python
# Pytorch code
import torch

# Thiết lập seed để đảm bảo kết quả ngẫu
    nhiên có thể tái tạo được
torch.manual_seed(2024)

# Tạo tensor với các giá trị ngẫu nhiên
    trong khoảng [0, 1) với kích thước (3,
     4)
tensor_rand = torch.rand((3, 4))
print("Tensor ngẫu nhiên trong khoảng
    [0, 1):\n", tensor_rand)

# Tạo tensor với các giá trị ngẫu nhiên
    trong khoảng [-10, 10)
tensor_randint = torch.randint(-10, 10,
    size=(3, 4))
print("Tensor ngẫu nhiên trong khoảng
    [-10, 10):\n", tensor_randint)
```

```
================ Output ===============
Tensor ngẫu nhiên trong khoảng [0, 1):
 tensor([[0.5317, 0.8313, 0.9718, 0.1193],
        [0.1669, 0.3495, 0.2150, 0.6201],
        [0.4849, 0.7492, 0.1521, 0.5625]])

Tensor ngẫu nhiên trong khoảng [-10, 10):
 tensor([[  1,   8,   0,  -2],
        [ -9, -10,  -9,   0],
        [ -3,  -7,  -4,   8]])

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

Trong Tensorflow, ta cũng có các hàm tương tự là **tf.random.uniform** để tạo tensor với các giá trị ngẫu nhiên trong khoảng [0.0, 1.0), và sử dụng **tf.random.uniform** để tạo tensor với các số nguyên ngẫu nhiên trong một khoảng cụ thể.

```tensorflow
#TensorFlow code
import tensorflow as tf


# Thiết lập seed để đảm bảo kết quả ngẫu nhiên có thể tái tạo được
tf.random.set_seed(2024)

# Tạo tensor với các giá trị ngẫu nhiên trong khoảng [0, 1)
tensor_rand = tf.random.uniform((3, 4))
print("Tensor ngẫu nhiên trong khoảng [0, 1):\n", tensor_rand)

# Tạo tensor với các giá trị ngẫu nhiên trong khoảng [-10, 10)
tensor_randint = tf.random.uniform((3, 4), minval=-10, maxval=10, dtype=tf.dtypes.int32)
print("Tensor ngẫu nhiên trong khoảng [-10, 10):\n", tensor_randint)
```

```
================= Output ===============
Tensor ngẫu nhiên trong khoảng [0, 1):

 tf.Tensor(
[[0.90034294 0.19453335 0.36069036
   0.66361904]
 [0.76605344 0.2159369  0.6261736
   0.07380784]
 [0.22062695 0.934368   0.93327904
   0.69267046]],
 shape=(3, 4), dtype=float32)

Tensor ngẫu nhiên trong khoảng [-10, 10):
 tf.Tensor(
[[-3 -7  9  3]
 [ 2 -5 -3 -5]
 [ 4 -3 -3  5]],
 shape=(3, 4), dtype=int32)


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