numpy数组和pytorch数组

numpy数组

In [7]:

```
import numpy as np
```

NumPy 的前身 Numeric 最早是由 Jim Hugunin 与其它协作者共同开发，2005 年，Travis Oliphant 在 Numeric 中结合了另一个同性质的程序库 Numarray 的特色，并加入了其它扩展而开发了 NumPy。 Numpy是高性能科学计算和数据分析的基础包，其官网为https://numpy.org/ (https://numpy.org/) 。

数组创建

创建一个一维数组

In [5]:

```
np.array([1,2,3,4,5])
```

Out[5]:

```
array([1, 2, 3, 4, 5])
```

创建一个二维数组

In [ ]:

```
np.array([[1,2,3],[4,5,6]])
```

创建一个元素为零的二维数组

In [8]:

```
np.zeros((2,3))
```

Out[8]:

```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

创建一个元素为1的二维数组

In [9]:

```
np.ones((2,3))
```

Out[9]:

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

生成随机的二维数组

In [ ]:

```
np.random.rand(3,4) # 均匀分布
```

In [ ]:

```
np.random.randn(3,4) # 标准正态分布
```

矩阵的属性

In [ ]:

```
a = np.zeros((3,4))
print(a.shape)
print(np.shape(a))
print(a.ndim)
print(np.ndim(a))
print(a.size)
print(np.size(a,0))
print(np.size(a,1))
```

每个维度称为矩阵的一个axis，如上的矩阵第一维的长度是3，第二维的长度是4。而(3,4)这个tuple称为矩阵的shape。

数组拼接

In [3]:

```
m1 = np.ones((2,2))
m2 = np.full((2,2), 2.0)
m3 = np.full((2,2), 3.0)
```

通过vstack（vertically）函数竖直得合成数组

In [6]:

```
m4 = np.vstack((m1, m2, m3))
print(m4.shape)
print(m4)
```

```
(6, 2)
[[1. 1.]
 [1. 1.]
 [2. 2.]
 [2. 2.]
 [3. 3.]
 [3. 3.]]
```

通过hstack（horizontally）函数水平得合成数组

In [20]:

```
m5 = np.hstack((m1, m2))
print(m5.shape)
print(m5)
```

```
(2, 4)
[[1. 1. 2. 2.]
 [1. 1. 2. 2.]]
```

注意stack函数的区别

In [17]:

```
m7 = np.stack((m1,m2), axis=0)
print(m7.shape)
print(m7)
m8 = np.stack((m1,m2), axis=1)
print(m8.shape)
print(m8)
m9 = np.stack((m1,m2), axis=2)
print(m9.shape)
print(m9)
```

```
(2, 2, 2)
[[[1. 1.]
  [1. 1.]]

 [[2. 2.]
  [2. 2.]]]
(2, 2, 2)
[[[1. 1.]
  [2. 2.]]

 [[1. 1.]
  [2. 2.]]]
(2, 2, 2)
[[[1. 2.]
  [1. 2.]]

 [[1. 2.]
  [1. 2.]]]
```

我们也可以用concatenate函数自由指定合成的axis

In [4]:

```
m10 = np.concatenate((m1, m2, m3), axis=0)  # 等同于vstack
print(m6.shape)
print(m6)
```

```
(6, 2)
[[1. 1.]
 [1. 1.]
 [2. 2.]
 [2. 2.]
 [3. 3.]
 [3. 3.]]
```

In [19]:

```
m11 = np.concatenate((m1, m2), axis=1)  # 等同于hstack
print(m7.shape)
print(m7)
```

```
(2, 2, 2)
[[[1. 1.]
  [1. 1.]]

 [[2. 2.]
  [2. 2.]]]
```

数组拆分

In [8]:

```
r = np.arange(24).reshape(6,4)
print(r)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
```

vsplit将数组沿纵向分割成三个数组

In [ ]:

```
r1, r2, r3 = np.vsplit(r, 3)
print(r1)
print(r1.shape)
print(r2)
print(r2.shape)
print(r3)
print(r3.shape)
```

hsplit将数组沿横向分割成两个数组

In [ ]:

```
r4, r5 = np.hsplit(r, 2)
print(r4)
print(r4.shape)
print(r5)
print(r5.shape)
```

split函数对数组进行分割

In [22]:

```
 r6,r7,r8=np.split(r,[1,3],axis=1)
print(r6)
print(r7)
print(r8)
print(r6.shape)
print(r7.shape)
print(r8.shape)
```

```
[[ 0]
 [ 4]
 [ 8]
 [12]
 [16]
 [20]]
[[ 1  2]
 [ 5  6]
 [ 9 10]
 [13 14]
 [17 18]
 [21 22]]
[[ 3]
 [ 7]
 [11]
 [15]
 [19]
 [23]]
(6, 1)
(6, 2)
(6, 1)
```

pytorch tensor

pytorch的官方网站为https://pytorch.org/ (https://pytorch.org/) 。 Pytorch是基于科学计算包的Python，目标两类人群

1. 为利用 GPU 的能力取代 NumPy
2. 提供最大灵活性和速度的深度学习研究平台

In [2]:

```
import torch
```

Tensors类似于Numpy的narrays，只是Tensors可以使用GPU加速计算

Tensor创建

创建一个2*3的矩阵

In [4]:

```
x = torch.Tensor(2, 3)
print(x)
print(x.size())
```

```
tensor([[-5.9951e-27,  4.5779e-41, -5.9951e-27],
        [ 4.5779e-41,  4.4842e-44,  0.0000e+00]])
torch.Size([2, 3])
```

创建一个2*3的空矩阵

In [5]:

```
x = torch.empty(2, 3)
print(x)
print(x.size())
```

```
tensor([[-5.9951e-27,  4.5779e-41, -5.9951e-27],
        [ 4.5779e-41,  4.4842e-44,  0.0000e+00]])
torch.Size([2, 3])
```

创建一个2*3的全为1的矩阵

In [9]:

```
x = torch.ones(2, 3)
print(x)
print(x.size())
```

```
tensor([[1., 1., 1.],
        [1., 1., 1.]])
torch.Size([2, 3])
```

创建一个2*3的随机矩阵

In [10]:

```
x = torch.rand(2, 3) # 均匀分布
print(x)
```

```
tensor([[0.5113, 0.2495, 0.6437],
        [0.4773, 0.4164, 0.6892]])
```

In [11]:

```
x = torch.rand(2, 3) # 标准正态分布
print(x)
```

```
tensor([[0.2031, 0.1493, 0.1148],
        [0.1006, 0.6589, 0.4320]])
```

创建一个单位矩阵

In [12]:

```
x = torch.eye(2)
print(x)
```

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

Tensor 拼接

In [14]:

```
t1 = torch.full([2,2],1) # 必须赋值
t2 = torch.full([2,2],2)
t3 = torch.full([2,2],3)
print(t1)
print(t2)
print(t3)
```

```
tensor([[1., 1.],
        [1., 1.]])
tensor([[2., 2.],
        [2., 2.]])
tensor([[3., 3.],
        [3., 3.]])
```

用cat函数自由指定合成的axis

In [21]:

```
t4 = torch.cat([t1,t2,t3], dim=0)
print(t4)

t5 = torch.cat([t1,t2], dim=1)
print(t5)
```

```
tensor([[1., 1.],
        [1., 1.],
        [2., 2.],
        [2., 2.],
        [3., 3.],
        [3., 3.]])
tensor([[1., 1., 2., 2.],
        [1., 1., 2., 2.]])
```

stack增加维度拼接

In [25]:

```
t6 = torch.stack([t1,t2], dim=0)
print(t6)
print(t6.size())
t7 = torch.stack([t1,t2], dim=1)
print(t7)
print(t7.size())
t8 = torch.stack([t1,t2], dim=2)
print(t8)
print(t8.size())
```

```
tensor([[[1., 1.],
         [1., 1.]],

        [[2., 2.],
         [2., 2.]]])
torch.Size([2, 2, 2])
tensor([[[1., 1.],
         [2., 2.]],

        [[1., 1.],
         [2., 2.]]])
torch.Size([2, 2, 2])
tensor([[[1., 2.],
         [1., 2.]],

        [[1., 2.],
         [1., 2.]]])
torch.Size([2, 2, 2])
```

Tensor 拆分

In [30]:

```
t9 = torch.arange(1,25)
t10 = t9.reshape(4,6)
print(t9)
print(t10)
```

```
tensor([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 1
6, 17, 18,
        19, 20, 21, 22, 23, 24])
tensor([[ 1,  2,  3,  4,  5,  6],
        [ 7,  8,  9, 10, 11, 12],
        [13, 14, 15, 16, 17, 18],
        [19, 20, 21, 22, 23, 24]])
```

使用split函数

In [43]:

```
t11,t12 = torch.split(t10,[1,3],dim=0) # 自定义长度
print(t11)
print(t12)
```

```
tensor([[1, 2, 3, 4, 5, 6]])
tensor([[ 7,  8,  9, 10, 11, 12],
        [13, 14, 15, 16, 17, 18],
        [19, 20, 21, 22, 23, 24]])
```

In [42]:

```
t13,t14 = torch.split(t10,[2,2],dim=0) # t13,t14 = torch.split(t10,2,dim=0)  平均分
print(t13)
print(t14)
```

```
tensor([[ 1,  2,  3,  4,  5,  6],
        [ 7,  8,  9, 10, 11, 12]])
tensor([[13, 14, 15, 16, 17, 18],
        [19, 20, 21, 22, 23, 24]])
```

In [54]:

```
t15,t16 = torch.split(t10,3,dim=0) # 若不能整除，则最后一个分块会小于其它分块
print(t15)
print(t16)
```

```
tensor([[ 1,  2,  3,  4,  5,  6],
        [ 7,  8,  9, 10, 11, 12],
        [13, 14, 15, 16, 17, 18]])
tensor([[19, 20, 21, 22, 23, 24]])
```

使用chunk函数，把一个tensor均匀分割成若干个小tensor,若不能整除，则最后一个分块会小于其它分块,或为空

In [64]:

```
t17,t18, = torch.chunk(t10,2,dim=0)
print(t17)
print(t18)
```

```
tensor([[ 1,  2,  3,  4,  5,  6],
        [ 7,  8,  9, 10, 11, 12]])
tensor([[13, 14, 15, 16, 17, 18],
        [19, 20, 21, 22, 23, 24]])
```

array和Tensor互相转换

tensor转换为array

In [67]:

```
x = torch.ones(5);
y = x.numpy()
print(x)
print(y)
```

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

array转换为tensor

In [69]:

```
y = np.ones(5)
x = torch.from_numpy(y)
print(y)
print(x)
```

```
[1. 1. 1. 1. 1.]
tensor([1., 1., 1., 1., 1.], dtype=torch.float64)
```

小结

功能	Numpy	Pytorch
创建矩阵	np.array()	torch.Tensor()
创建随机矩阵	np.random.rand() / np.random.randn()	torch.rand() / torch.randn()
增加维度拼接	np.stack()	torch.stack()
不增加维度拼接	np.concatenate()	torch.cat()
矩阵分割	np.split()	torch.split()

注意有些函数的输入参数之间是有区别的，如np.split()和torch.split()