Pytorch张量介绍

2022.07.18

```
import torch
import numpy as np
# Tensor可理解为多维数组
# （1）type()是python内置的函数。type() 返回数据结构类型（list、dict、numpy.ndarray 等）
# （2）dtype 返回数据元素的数据类型（int、float等）
# （3）astype() 改变np.array中所有数据元素的数据类型。
```

```
# 列举常用的，其实更多的还是用的时候看官方文档,yyds
```

Tensor基础

```
a = [1,2,3]
b = torch.Tensor(a)
print(type(a))
print(type(b))
print(b.dtype) # Tensor中的元素类型为 float32
```

```
<class 'list'>
<class 'torch.Tensor'>
torch.float32
```

```
a = np.random.normal((2,3)) # 正态分布，均值为2，方差为3
b = torch.Tensor(a)
print(a)
print(b)
print(b.dtype)
c = torch.ones_like(b) # 1
print(c)
d = torch.rand_like(b) # 随机
print(d)
```

```
[1.86075708 4.90797958]
tensor([1.8608, 4.9080])
torch.float32
tensor([1., 1.])
tensor([0.3748, 0.7736])
```

```python
a = torch.rand((2,2)) # 随机生成2*2的Tensor,或传入列表 ， 2*2看成两行两列
print(a)
print(a.dtype)

# 属性
print(a.shape)
print(a.device)

print(torch.is_tensor(a))
```

```
tensor([[0.1908, 0.1239],
        [0.1794, 0.2687]])
torch.float32
torch.Size([2, 2])
cpu
True
```

```python
a = torch.Tensor(1)
print(torch.is_nonzero(a))
# b = torch.Tensor(0)
# print(torch.is_nonzero(b))
```

```
True
```

```python
a = torch.rand((2,2))
print(torch.numel(a)) # 这个张量中所有元素的总数目
```

```
4
```

```python
a = torch.zeros([5,5])
print(a)
print(a.dtype)
b = torch.zeros([5,5],dtype=torch.int32)
print(b)
```

```
tensor([[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]])
torch.float32
tensor([[0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0]], dtype=torch.int32)
```

arange函数

```
# arange函数 总数目：(end - start)/step
a = torch.arange(5) # [Start,End)
print(a)
print(a.dtype) # 默认是Int64
b = torch.arange(0,5,2) # 5取不到
print(b)
```

```
tensor([0, 1, 2, 3, 4])
torch.int64
tensor([0, 2, 4])
```

range函数

```
# range函数 总数目：(end - start)/step + 1
a = torch.range(start=0,end=5) # 5取得到
print(a)
print(a.dtype)  # 默认是float32
```

```
tensor([0., 1., 2., 3., 4., 5.])
torch.float32
```

```
C:\Users\17435\anaconda3\lib\site-packages\ipykernel_launcher.py:2: UserWarning:
torch.range is deprecated and will be removed in a future release because its
behavior is inconsistent with Python's range builtin. Instead, use torch.arange,
which produces values in [start, end).
```

eye函数

```
a = torch.eye(3)
print(a)
```

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

full函数

```
a = torch.full([2,2],5)
print(a)
```

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

cat函数

```
# cat函数
a = torch.rand([2,2])
b = torch.rand([2,3])
print(a)
print(b)
# 拼接
c = torch.cat([a,b],dim=1) # 按第2维拼接
print(c)
```

```
tensor([[0.5050, 0.7638],
        [0.4250, 0.1312]])
tensor([[0.5087, 0.5983, 0.1191],
        [0.3641, 0.1201, 0.4325]])
tensor([[0.5050, 0.7638, 0.5087, 0.5983, 0.1191],
        [0.4250, 0.1312, 0.3641, 0.1201, 0.4325]])
```

```
a = torch.rand([2,2])
b = torch.rand([3,2])
print(a)
print(b)
# 拼接
c = torch.cat([a,b],dim=0) # 按第2维拼接
print(c)
```

```
tensor([[0.1499, 0.6340],
        [0.6928, 0.9849]])
tensor([[4.3438e-01, 2.6518e-01],
        [3.0065e-04, 6.7917e-01],
        [3.0370e-01, 8.5561e-01]])
tensor([[1.4987e-01, 6.3396e-01],
        [6.9276e-01, 9.8493e-01],
        [4.3438e-01, 2.6518e-01],
        [3.0065e-04, 6.7917e-01],
        [3.0370e-01, 8.5561e-01]])
```

chunk函数（用于分割）

```
a = torch.rand([3,2])
print(a)
b = torch.chunk(a,chunks=2) # 默认按dim=0,chunks表示分成2个
print(b)
c,d = torch.chunk(a,chunks=2)
print(c)
```

```
tensor([[0.2344, 0.4459],
        [0.5224, 0.8762],
        [0.0453, 0.8944]])
(tensor([[0.2344, 0.4459],
        [0.5224, 0.8762]]), tensor([[0.0453, 0.8944]]))
tensor([[0.2344, 0.4459],
        [0.5224, 0.8762]])
```

reshape函数

```
a = torch.arange(4)
print(a)
b = torch.reshape(a,(2,2))
print(b)
c = torch.reshape(b,[-1]) # 变成一维的,就和a一样了
print(c)
```

```
tensor([0, 1, 2, 3])
tensor([[0, 1],
        [2, 3]])
tensor([0, 1, 2, 3])
```

squeeze函数（降低维度）

```
a = torch.rand([3,2])
print(a.shape)
print(a)
b = torch.reshape(a,[3,1,2]) # 3*1*2 = 3*2
print(b)
# 删除 1 这一维 ,也就是说squeeze是把维数为1的给移除，如A*1*B*C*1*D --->  A*B*C*D
c = torch.squeeze(b)
print(c)
```

```
torch.Size([3, 2])
tensor([[0.6259, 0.2592],
        [0.4027, 0.6188],
        [0.5671, 0.9000]])
tensor([[[0.6259, 0.2592]],

        [[0.4027, 0.6188]],

        [[0.5671, 0.9000]]])
tensor([[0.6259, 0.2592],
        [0.4027, 0.6188],
        [0.5671, 0.9000]])
```

```
a = torch.rand([3,2])
b = torch.reshape(a,[3,1,2,1,1])
print(b)
c = torch.squeeze(b)
print(c)
```

```
tensor([[[[[0.4311]],

          [[0.4349]]]],
```

[[[[0.9111]],

```
          [[0.1867]]]],
```

[[[[0.8477]],

```
          [[0.6680]]]])
tensor([[0.4311, 0.4349],
        [0.9111, 0.1867],
        [0.8477, 0.6680]])
```

```
# 指定维度
a = torch.rand([3,2])
b = torch.reshape(a,[3,1,2,1,1])
c = torch.squeeze(b,dim=1)
print(c.shape)
```

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

unsqueeze函数（增加维度）

```
a = torch.tensor([1,2,3,4])
print(a)
print(a.shape)
b = torch.unsqueeze(a,0)
print(b)
print(b.shape)
c = torch.unsqueeze(a,1)
print(c)
print(c.shape)
```

```
tensor([1, 2, 3, 4])
torch.Size([4])
tensor([[1, 2, 3, 4]])
torch.Size([1, 4])
tensor([[1],
        [2],
        [3],
        [4]])
torch.Size([4, 1])
```

tile函数（用于复制）

```
a = torch.Tensor([1,2,3])
print(a.tile((2)))
b = torch.Tensor([[1,2],[3,4]]) # 2*2
print(b)
c = torch.tile(b,(2,2)) # 第一个维度复制两份，第二个也是
print(c)
# 若传入的维度比Tensor的要少，如 Tensor为(8,6,4,2)，而传入维度为(2,2)，则会自动填充为(1,1,2,2)
```

```
tensor([1., 2., 3., 1., 2., 3.])
tensor([[1., 2.],
        [3., 4.]])
tensor([[1., 2., 1., 2.],
        [3., 4., 3., 4.],
        [1., 2., 1., 2.],
        [3., 4., 3., 4.]])
```

```
a = torch.rand(4,3)
print(a)
a_tiled = torch.tile(a,[2,1])
print(a_tiled) # 8*3
a_tiled1 = torch.tile(a,[1,3])
print(a_tiled1) # 4*9
a_tiled2 = torch.tile(a,[2,3])
print(a_tiled2) # 8*9
```

```
tensor([[0.2200, 0.8335, 0.9748],
        [0.9201, 0.7036, 0.0800],
        [0.0902, 0.9818, 0.0802],
        [0.3845, 0.9041, 0.5785]])
tensor([[0.2200, 0.8335, 0.9748],
        [0.9201, 0.7036, 0.0800],
        [0.0902, 0.9818, 0.0802],
        [0.3845, 0.9041, 0.5785],
        [0.2200, 0.8335, 0.9748],
        [0.9201, 0.7036, 0.0800],
        [0.0902, 0.9818, 0.0802],
        [0.3845, 0.9041, 0.5785]])
tensor([[0.2200, 0.8335, 0.9748, 0.2200, 0.8335, 0.9748, 0.2200, 0.8335,
0.9748],
        [0.9201, 0.7036, 0.0800, 0.9201, 0.7036, 0.0800, 0.9201, 0.7036,
0.0800],
        [0.0902, 0.9818, 0.0802, 0.0902, 0.9818, 0.0802, 0.0902, 0.9818,
0.0802],
        [0.3845, 0.9041, 0.5785, 0.3845, 0.9041, 0.5785, 0.3845, 0.9041,
0.5785]])
tensor([[0.2200, 0.8335, 0.9748, 0.2200, 0.8335, 0.9748, 0.2200, 0.8335,
0.9748],
        [0.9201, 0.7036, 0.0800, 0.9201, 0.7036, 0.0800, 0.9201, 0.7036,
0.0800],
        [0.0902, 0.9818, 0.0802, 0.0902, 0.9818, 0.0802, 0.0902, 0.9818,
0.0802],
        [0.3845, 0.9041, 0.5785, 0.3845, 0.9041, 0.5785, 0.3845, 0.9041,
0.5785],
        [0.2200, 0.8335, 0.9748, 0.2200, 0.8335, 0.9748, 0.2200, 0.8335,
0.9748],
        [0.9201, 0.7036, 0.0800, 0.9201, 0.7036, 0.0800, 0.9201, 0.7036,
0.0800],
        [0.0902, 0.9818, 0.0802, 0.0902, 0.9818, 0.0802, 0.0902, 0.9818,
0.0802],
        [0.3845, 0.9041, 0.5785, 0.3845, 0.9041, 0.5785, 0.3845, 0.9041,
0.5785]])
```

transpose函数

```
a = torch.randn(2,3) # randn表示满足标准正态分布(0,1)
print(a) # 2*3
b = torch.transpose(a,0,1) # 1,0也行
print(b) # 3*2
```

```
tensor([[ 0.4865, -0.6912,  0.1034],
        [-0.6256,  0.5780,  0.2711]])
tensor([[ 0.4865, -0.6256],
        [-0.6912,  0.5780],
        [ 0.1034,  0.2711]])
```

unbind函数

```
# 2维张量
a = torch.rand((4,3))
print(a)
b = torch.unbind(a,dim=0)
print(b)
c = torch.unbind(a,dim=1)
print(c)
```

```
tensor([[0.5718, 0.5241, 0.5091],
        [0.6381, 0.4077, 0.7330],
        [0.0455, 0.6433, 0.0024],
        [0.3128, 0.9635, 0.5507]])
(tensor([0.5718, 0.5241, 0.5091]), tensor([0.6381, 0.4077, 0.7330]),
tensor([0.0455, 0.6433, 0.0024]), tensor([0.3128, 0.9635, 0.5507]))
(tensor([0.5718, 0.6381, 0.0455, 0.3128]), tensor([0.5241, 0.4077, 0.6433,
0.9635]), tensor([0.5091, 0.7330, 0.0024, 0.5507]))
```

```
# 3维张量
a = torch.rand((2,3,4)) # 要
print(a)
print("================================")
b = torch.unbind(a,dim=0)
print(b)
print("================================")
c = torch.unbind(a,dim=1)
print(c)
```

```
tensor([[[0.5349, 0.5428, 0.5302, 0.6086],
         [0.4699, 0.7760, 0.4407, 0.5393],
         [0.6315, 0.4695, 0.1710, 0.9065]],

        [[0.6641, 0.7211, 0.1332, 0.8443],
         [0.8008, 0.3894, 0.5648, 0.8209],
         [0.8653, 0.8879, 0.2399, 0.9747]]])
================================
(tensor([[0.5349, 0.5428, 0.5302, 0.6086],
        [0.4699, 0.7760, 0.4407, 0.5393],
        [0.6315, 0.4695, 0.1710, 0.9065]]), tensor([[0.6641, 0.7211, 0.1332,
0.8443],
        [0.8008, 0.3894, 0.5648, 0.8209],
        [0.8653, 0.8879, 0.2399, 0.9747]]))
================================
```

```
(tensor([[0.5349, 0.5428, 0.5302, 0.6086],
        [0.6641, 0.7211, 0.1332, 0.8443]]), tensor([[0.4699, 0.7760, 0.4407,
0.5393],
        [0.8008, 0.3894, 0.5648, 0.8209]]), tensor([[0.6315, 0.4695, 0.1710,
0.9065],
        [0.8653, 0.8879, 0.2399, 0.9747]]))
```

```

```

```

```