**tensor的连续性**

PyTorch中，任意维度的tensor在底层都是以一维数组(storage)的方式存于某个连续的内存中的。所谓tensor连续(contiguous)，指的是tensor的storage元素排列顺序与其按行优先(内层优先)时的元素排列顺序相同，之所以会出现不连续现象，本质上是由于不同tensor可能共用同一个storage导致的



**Tensor.contiguous()**

强制拷贝一份tensor，让它的布局和从头创建的一样，使得两个tensor没有联系，且新生成的张量存储方式为连续存储

*#不用contiguous*

x = torch.randn(3, 2)

y = torch.transpose(x, 0, 1)

print("修改前：")

print("x-", x)

print("y-", y)

print("\n修改后：")

y[0, 0] = 11

print("x-", x)

print("y-", y)

>>>

修改前：

x- tensor([[-0.3228, 0.2792],

[-0.5702, 1.1240],

[-0.5136, -1.6730]])

y- tensor([[-0.3228, -0.5702, -0.5136],

[ 0.2792, 1.1240, -1.6730]])

修改后：

x- tensor([[11.0000, 0.2792],

[-0.5702, 1.1240],

[-0.5136, -1.6730]])

y- tensor([[11.0000, -0.5702, -0.5136],

[ 0.2792, 1.1240, -1.6730]])

x = torch.randn(3, 2)

y = torch.transpose(x, 0, 1).contiguous()*#用contiguous*

print("修改前：")

print("x-", x)

print("y-", y)

print("\n修改后：")

y[0, 0] = 11

print("x-", x)

print("y-", y)

修改前：

x- tensor([[ 0.7698, -1.1048],

[-0.5661, -0.0194],

[ 0.1065, 1.3425]])

y- tensor([[ 0.7698, -0.5661, 0.1065],

[-1.1048, -0.0194, 1.3425]])

修改后：

x- tensor([[ 0.7698, -1.1048],

[-0.5661, -0.0194],

[ 0.1065, 1.3425]])

y- tensor([[11.0000, -0.5661, 0.1065],

[-1.1048, -0.0194, 1.3425]])

**Tensor.is\_contiguous()**

tensor连续的另一定义是：

对于任意的 k 维张量 t ,如果满足对于所有 i，第 i 维相邻元素间隔 = 第 i+1 维相邻元素间隔 与 第 i+1 维长度的乘积，则 t 是连续的，即stride[i] 表示第 i 维相邻元素之间间隔的位数，称为步长，可通过 stride 方法获得  
size[i] 表示固定其他维度时，第 i 维元素数量

is\_contiguous()方法据此判断张量在内存中是否连续存储