PT complex tensor examples

In [13]:
```
import numpy as np
import torch
from complex_tensors import ComplexTensor
```

Creation

In [2]:
```
# numpy complex tensor
np_c = np.asarray([[1+3j, 1+3j, 1+3j], [2+4j, 2+4j, 2+4j]]).astype(np.complex64)
np_c
```

Out[2]:
```
array([[1.+3.j, 1.+3.j, 1.+3.j],
       [2.+4.j, 2.+4.j, 2.+4.j]], dtype=complex64)
```

In [4]:
```
# torch equivalent
pt_c = ComplexTensor([[1, 1, 1], [2,2,2], [3,3,3], [4,4,4]])
pt_c
```

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

In [6]:
```
# verify reals match
print(np_c.real)
print(pt_c.real)
```

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

In [7]:
```
# verify imag match
print(np_c.imag)
print(pt_c.imag)
```

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

Verify complex addition

In [8]:
```
np_c + (3+2j)
```

Out[8]:
```
array([[4.+5.j, 4.+5.j, 4.+5.j],
       [5.+6.j, 5.+6.j, 5.+6.j]], dtype=complex64)
```

In [9]:
```
pt_c + (3 + 2j)
```

Out[9]:
```
tensor([[4., 4., 4.],
        [5., 5., 5.],
        [5., 5., 5.],
        [6., 6., 6.]])
```

verify abs

In [10]:
```
np.abs(np_c)
```

Out[10]:
```
array([[3.1622777, 3.1622777, 3.1622777],
       [4.472136 , 4.472136 , 4.472136 ]], dtype=float32)
```

In [11]:
```
pt_c.abs()
```

Out[11]:
```
tensor([[3.1623, 3.1623, 3.1623],
        [4.4721, 4.4721, 4.4721]])
```

verify complex vs real matrix multiply

In [14]:
```
np_x = np.asarray([[3, 3], [4, 4], [2, 2]])
pt_x = torch.Tensor([[3, 3], [4, 4], [2, 2]])

print(np_x)
print(pt_x)
```

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

In [17]:
```
np_mm_out = np.matmul(np_c, np_x)
np_mm_out
```

Out[17]:
```
array([[ 9.+27.j,  9.+27.j],
       [18.+36.j, 18.+36.j]])
```

In [18]:
```
pt_mm_out = pt_c.mm(pt_x)
pt_mm_out
```

Out[18]:
```
tensor([[ 9.,  9.],
        [18., 18.],
        [27., 27.],
        [36., 36.]])
```

In [19]:
```
# verify reals
print(np_mm_out.real)
print(pt_mm_out.real)
```

```
[[ 9.  9.]
 [18. 18.]]
tensor([[ 9.,  9.],
        [18., 18.]])
```

In [20]:
```
# verify imags
print(np_mm_out.imag)
print(pt_mm_out.imag)
```

```
[[27. 27.]
 [36. 36.]]
tensor([[27., 27.],
        [36., 36.]])
```

---

verify transpose

In [21]:
```
np_c.T
```

Out[21]:
```
array([[1.+3.j, 2.+4.j],
       [1.+3.j, 2.+4.j],
       [1.+3.j, 2.+4.j]], dtype=complex64)
```

In [22]:
```
pt_c.t()
```

Out[22]:
```
tensor([[1., 2.],
        [1., 2.],
        [1., 2.],
        [3., 4.],
        [3., 4.],
        [3., 4.]])
```

show pytorch grads still work

In [29]:
```
pt_c2 = ComplexTensor([[2, 2, 2], [1,1,1], [4,4,4], [3,3,3]])
pt_c2.requires_grad = True
```

In [33]:
```
out = pt_c2 + 4
out = out.mm(pt_c.t())
print(out)
```

```
tensor([[-18., -12.],
        [-12.,  -6.],
        [ 66.,  96.],
        [ 54.,  78.]], grad_fn=<CatBackward>)
```

In [34]:
```
y = out.sum()
print(y)
```

```
tensor(246., grad_fn=<SumBackward0>)
```

In [35]:
```
y.backward()
```

In [36]:
```
pt_c2.grad
```

Out[36]:
```
tensor([[10., 10., 10.],
        [10., 10., 10.],
        [-4., -4., -4.],
        [-4., -4., -4.]])
```