In [1]:
```
import torch
print(torch.__version__)
```

```
1.9.1
```

In [ ]:
```

```

自动求导机制

In [2]:
```
x = torch.randn(4,1, requires_grad=True)
y = torch.randn(4,1, requires_grad=True)
W = torch.randn(4,4)
print(x)
print(y)
print(W)
```

```
tensor([[-1.1258],
        [-0.5773],
        [-0.1061],
        [-0.9505]], requires_grad=True)
tensor([[-3.1289],
        [ 0.5473],
        [-1.8960],
        [ 0.6313]], requires_grad=True)
tensor([[ 0.2428,  0.5767,  0.4123,  1.7777],
        [ 0.0687,  1.0518,  0.2852, -1.0909],
        [-0.2580,  0.0690, -0.0691, -2.1543],
        [-1.8731,  1.9711,  1.7972, -0.8069]])
```

In [3]:
```
z = torch.mm(torch.mm(torch.t(x), W),y)
print(z)
```

```
tensor([[-2.2146]], grad_fn=<MmBackward>)
```

In [4]:
```
z = x.t().mm(W).mm(y).abs()
print(z)
```

```
tensor([[2.2146]], grad_fn=<AbsBackward>)
```

In [5]:
```
print(x.grad)
```

```
None
```

In [6]:
```
z.backward()
```

In [7]:
```
print(x.grad)
```

```
tensor([[ 0.1033],
        [ 0.8686],
        [ 0.3841],
        [-3.0227]])
```

In [8]:
```
print(W.mm(y))
```

```
tensor([[-0.1033],
        [-0.8686],
```

```
        [-0.3841],
        [ 3.0227]], grad_fn=<MmBackward>)
```

In [ ]:

In [ ]:

默认情况下，定义的tensor属性requires_grad为false

In [9]:

```
x = torch.randn(4,1)
print(x)
y = torch.mm(torch.t(x),x)
#  requires_grad=True
y.requires_grad = True
y.backward()
```

```
tensor([[ 0.4440],
        [ 2.2153],
        [ 0.9607],
        [-0.7670]])
```

In [10]:

```
print(y)

print(x.grad)

print(2*x)
```

```
tensor([[6.6161]], requires_grad=True)
None
tensor([[ 0.8879],
        [ 4.4307],
        [ 1.9215],
        [-1.5341]])
```

PyTorch Tensor 与 Numpy 转换

In [11]:

```
import numpy as np
import torch

a = np.random.randn(3,4)
print(a)
print(type(a))
```

```
[[-0.89046738 -1.33144818  0.58736942  0.12961395]
 [-0.66517408 -0.49432405 -0.07731219  0.95735545]
 [ 0.49108709 -0.51189006  2.32828968  1.75018753]]
<class 'numpy.ndarray'>
```

In [12]:

```
a_tensor = torch.from_numpy(a)
print(a_tensor)
print(type(a_tensor))
```

```
tensor([[-0.8905, -1.3314,  0.5874,  0.1296],
        [-0.6652, -0.4943, -0.0773,  0.9574],
        [ 0.4911, -0.5119,  2.3283,  1.7502]], dtype=torch.float64)
<class 'torch.Tensor'>
```

In [13]:

```
b = a_tensor.numpy()
```

```
print(b)
print(type(b))
```

```
[[-0.89046738 -1.33144818  0.58736942  0.12961395]
 [-0.66517408 -0.49432405 -0.07731219  0.95735545]
 [ 0.49108709 -0.51189006  2.32828968  1.75018753]]
<class 'numpy.ndarray'>
```

In [14]:

```
d_tensor = torch.randn(3, 4, requires_grad=False)
d_numpy = d_tensor.numpy()
print(d_numpy)
print(type(d_numpy))
```

```
[[ 0.2963589   2.5849059   0.5525259  -0.9820206 ]
 [ 0.6916591  -0.39802215 -0.34922287  0.69784075]
 [ 0.49665242  0.1621982   0.8358983   0.07762461]]
<class 'numpy.ndarray'>
```

In [15]:

```
d_tensor = torch.randn(3, 4, requires_grad=True)
print(d_tensor)
# d_numpy = d_tensor.numpy()
d_numpy = d_tensor.detach().numpy()
print(d_numpy)
print(type(d_numpy))
```

```
tensor([[ 0.7342,  0.3885, -0.3788,  0.1474],
        [ 0.4050, -0.0277, -1.8610,  0.1441],
        [ 1.0670, -0.0192,  0.7793, -0.2527]], requires_grad=True)
[[ 0.73417985  0.38852325 -0.37883264  0.14740653]
 [ 0.40495276 -0.02765906 -1.8610157   0.14414687]
 [ 1.0669876  -0.01918314  0.77927184 -0.25268313]]
<class 'numpy.ndarray'>
```

In [16]:

```
d_tensor = torch.randn(3, 4, requires_grad=True)
d_numpy = d_tensor.data.numpy()
print(d_numpy)
print(d_tensor.data)
print(type(d_numpy))
```

```
[[-2.8518267  -1.0040312   0.6788978  -0.8308356 ]
 [-0.29047233 -1.2727869  -0.7147835   0.36596352]
 [ 1.0417142   0.04265385  0.69055223 -0.5379234 ]]
tensor([[-2.8518, -1.0040,  0.6789, -0.8308],
        [-0.2905, -1.2728, -0.7148,  0.3660],
        [ 1.0417,  0.0427,  0.6906, -0.5379]])
<class 'numpy.ndarray'>
```

In [17]:

```
# Kan Horst - PKU - 干皓丞
```