pytorch-基本运算

basic

+、-、*、/

torch.add(a,b)、torch.sub(a,b)、torch.mul(a,b)、torch.div(a,b)

基本运算，上面的都是等效的。

上面的乘法运算表示的是相同位置的元素进行相乘。

```
In [2]: a=torch.rand(3,4)
In [3]: b=torch.rand(4)

In [4]: a+b
Out[4]:
tensor([[0.7114, 1.7711, 1.1941, 0.8124],
        [0.8159, 0.8690, 1.2145, 0.5990],
        [0.3545, 1.3203, 0.4354, 0.1347]])

In [5]: torch.add(a,b)
Out[5]:
tensor([[0.7114, 1.7711, 1.1941, 0.8124],
        [0.8159, 0.8690, 1.2145, 0.5990],
        [0.3545, 1.3203, 0.4354, 0.1347]])

In [6]: torch.all(torch.eq(a-b, torch.sub(a,b)))
Out[6]: tensor(1, dtype=torch.uint8)

In [7]: torch.all(torch.eq(a*b, torch.mul(a,b)))
Out[7]: tensor(1, dtype=torch.uint8)

In [8]: torch.all(torch.eq(a/b, torch.div(a,b)))
Out[8]: tensor(1, dtype=torch.uint8)
```

矩阵相乘

这是按照矩阵的运算规则进行相乘。

有三种方式：

1、torch.mm　只适用于2d的矩阵相乘

2、torch.matmul (a,b)　适用于两个矩阵相乘

3、 @　　第二种方式等价，一种简写

```
In [80]:
1 a = torch.ones([2,2]) * 3
2 a
```

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

```
In [81]:
1 b = torch.ones(2,2)
2 b
```

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

```
In [82]:
1 torch.mm(a,b)
```

```
tensor([[6., 6.],
        [6., 6.]])
```

```
In [83]:
1 torch.matmul(a,b)
```

```
tensor([[6., 6.],
        [6., 6.]])
```

```
In [84]:
1 a@b
```

```
tensor([[6., 6.],
        [6., 6.]])
```

实例：

原始张量为【4.784】，我们希望将其降维为【4,512】这样类型的数据。

w张量是pytorch的习惯写法，第一个数据是channel-out，第二个数据是channel-in。也就是结果是512，进来的是784.

```
In [84]:
1 a@b

tensor([[6., 6.],
        [6., 6.]])
```

```
In [86]:
1 #原始张量，我们希望将其转换为【4,512】这样类型的数据
2 x = torch.rand(4,784)
```

```
In [87]:
1 #这里是pytorch的习惯写法，第一个数据是channel-out，第二个数据是channel-in
2 w = torch.rand(512,784)
```

```
In [88]:
1 (x@w.t()).shape            #和w的转置矩阵相乘

torch.Size([4, 512])
```

torch.matmul()

适用于多维张量的运算。

a = torch.rand(4,3,28,64)
b = torch.rand(4,3,64,32)

当用于这两个张量的计算时，该函数会计算最后两个维度的相乘，前两个维度不变。得到结果：torch.Size([4, 3, 28, 32])

c = torch.rand(4,1,64,32)

torch.matmul(a,c).shape

当上述a和c进行张量相乘的时候，会首先将c进行广播维度扩展，之后进行运算。

如果前面的维度数据无法扩展到相应的数据，则无法张量相乘。

```
In [90]:
1 a = torch.rand(4,3,28,64)
2 b = torch.rand(4,3,64,32)
```

```
In [91]:
1 torch.matmul(a,b).shape

torch.Size([4, 3, 28, 32])
```

```
In [92]:
1 c = torch.rand(4,1,64,32)
```

```
In [93]:
1 torch.matmul(a,c).shape

torch.Size([4, 3, 28, 32])
```

```
In [94]:
1 d = torch.rand(4,64,32)
2 torch.matmul(a,d).shape

---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-94-79f2206f9822> in <module>
      1 d = torch.rand(4,64,32)
----> 2 torch.matmul(a,d).shape

RuntimeError: The size of tensor a (3) must match the size of tensor b (4) at non-singleton dimension
```

次方计算

1、pow(input, exponent)　:将张量根据exponent参数进行次方运算。

2、**　：次方运算

3、sqrt ：开方运算

4、rsqrt ：返回一个新张量，包含输入张量每个元素的平方根倒数。

```
In [112]:
1  a = torch.full([2,2],3.)
```

```
In [113]:
1  a.pow(2)
```

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

```
In [114]:
1  a**2
```

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

```
In [120]:
1  aa = a**2
2  aa.sqrt()
```

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

```
In [116]:
1  aa.rsqrt()
```

```
tensor([[0.3333, 0.3333],
        [0.3333, 0.3333]])
```

```
In [119]:
1  aa**0.5
```

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

指数、对数

torch.exp（）：指数函数

torch.log（）　：对数函数

```
In [116]: a=torch.exp(torch.ones(2,2))

In [117]: a
Out[117]:
tensor([[2.7183, 2.7183],
        [2.7183, 2.7183]])

In [118]: torch.log(a)
Out[118]:
tensor([[1., 1.],
        [1., 1.]])
```

近似值

a.floor()　　#向下取整函数
a.ceil()　　#向上取整函数
a.trunc()　　#数据的整数部分
a.frac()　　#数据的小数部分
a.round()　　#数据的四舍五入计算

```
In [121]: a=torch.tensor(3.14)

In [124]: a.floor(),a.ceil(),a.trunc(),a.frac()
Out[124]: (tensor(3.), tensor(4.), tensor(3.), tensor(0.1400))

In [125]: a=torch.tensor(3.499)

In [126]: a.round()
Out[126]: tensor(3.)

In [127]: a=torch.tensor(3.5)

In [128]: a.round()
Out[128]: tensor(4.)
```