



# 统计属性

---

主讲人：龙良曲

# statistics

- norm
- mean sum
- prod
- max, min, argmin, argmax
- kthvalue, topk

# norm

- v.s. normalize ,e.g. batch\_norm
- matrix norm v.s. vector norm



# Vector vs matrix

## Matrix Operator: The Norm

### Vector Norm

1-Norm

$$\|x\|_1 = \sum_{i=1}^n |a_i|$$

Euclidean/Frobenius Norm

$$\|x\|_e = \sqrt{\sum_{i=1}^n x_i^2}$$

p-Norm

$$\|x\|_p = \left( \sum_{i=1}^n |x_i|^p \right)^{1/p}$$

### Matrix Norm

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$$

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n a_{ij}^2}$$

$$\|A\|_p = \left( \sum_{i=1}^n \sum_{j=1}^n a_{ij}^p \right)^{1/p}$$

# norm-p

```
1 In [104]: a=torch.full([8], 1)
2 In [105]: b=a.view(2,4)
3 In [106]: c=a.view(2,2,2)
4 In [107]: b
5 tensor([[1., 1., 1., 1.],
6         [1., 1., 1., 1.]])
7 In [108]: c
8 tensor([[[1., 1.],
9          [1., 1.]],
10         [[1., 1.],
11          [1., 1.]]])
12
13 In [109]: a.norm(1),b.norm(1),c.norm(1)
14 Out[109]: (tensor(8.), tensor(8.), tensor(8.))
15
16 In [110]: a.norm(2),b.norm(2),c.norm(2)
17 Out[110]: (tensor(2.8284), tensor(2.8284), tensor(2.8284))
18
19 In [111]: b.norm(1, dim=1)
20 Out[111]: tensor([4., 4.])
21 In [112]: b.norm(2, dim=1)
22 Out[112]: tensor([2., 2.])
23
24 In [113]: c.norm(1, dim=0)
25 tensor([[2., 2.],
26         [2., 2.]])
27 In [114]: c.norm(2, dim=0)
28 tensor([[1.4142, 1.4142],
29         [1.4142, 1.4142]])
```

# mean, sum, min, max, prod



```
1 In [143]: a=torch.arange(8).view(2,4).float()
2 tensor([[0, 1, 2, 3],
3         [4, 5, 6, 7]])
4
5 In [144]: a.min(),a.max(),a.mean(),a.prod()
6 Out[144]: (tensor(0.), tensor(7.), tensor(3.5000), tensor(0.))
7
8 In [145]: a.sum()
9 Out[145]: tensor(28.)
10
11 In [147]: a.argmax(),a.argmin()
12 Out[147]: (tensor(7), tensor(0))
13
```

---

# argmin, argmax

```
1 In [149]: a=a.view(1,2,4)
2 tensor([[[0., 1., 2., 3.],
3          [4., 5., 6., 7.]]])
4
5 In [151]: a.argmax()
6 Out[151]: tensor(7)
7
8 In [152]: a.argmin()
9 Out[152]: tensor(0)
10
11 In [153]: a=torch.rand(2,3,4)
12
13 In [154]: a.argmax()
14 Out[154]: tensor(21)
```

```
1 In [155]: a=torch.randn(4,10)
2
3 In [156]: a[0]
4 Out[156]:
5 tensor([-0.2636, -0.2958, -2.1356,  0.8362, -0.3137, -0.3806, -0.3547, -0.5220,
6         -0.1068, -0.1281])
7
8 In [157]: a.argmax()
9 Out[157]: tensor(18)
10
11 In [158]: a.argmax(dim=1)
12 Out[158]: tensor([3, 8, 6, 4])
```

# dim, keepdim

```
1 In [168]: a # [4, 10]
2 tensor([[ -0.2636, -0.2958, -2.1356,  0.8362, -0.3137, -0.3806, -0.3547, -0.5220,
3          -0.1068, -0.1281],
4          [ 0.2576,  1.1446, -0.4259, -0.7813,  0.1606,  1.2477,  1.4879, -1.2885,
5            1.7015, -0.4974],
6          [ 0.7188,  0.4481,  0.1466, -0.1624,  0.3597, -1.0365,  1.1297, -0.9084,
7            0.4212,  0.8691],
8          [-1.1000, -0.0078, -0.7398,  0.3747,  0.6386,  0.1516, -0.8415,  0.2227,
9            -1.4960, -0.1437]])
10
11 In [170]: a.max(dim=1)
12 Out[170]: (tensor([0.8362, 1.7015, 1.1297, 0.6386]), tensor([3, 8, 6, 4]))
13 In [171]: a.argmax(dim=1)
14 Out[171]: tensor([3, 8, 6, 4])
15
16 In [173]: a.max(dim=1, keepdim=True)
17 (tensor([[0.8362],
18          [1.7015],
19          [1.1297],
20          [0.6386]]), tensor([[3],
21                             [8],
22                             [6],
23                             [4]]))
24
25 In [172]: a.argmax(dim=1, keepdim=True)
26 tensor([[3],
27         [8],
28         [6],
29         [4]])
```



# Top-k or k-th

- `.topk`
  - Largest

- `kthvalue`

```
1 tensor([[ -0.2636, -0.2958, -2.1356,  0.8362, -0.3137, -0.3806, -0.3547, -0.5220,
2          -0.1068, -0.1281],
3          ...])
4
5 In [175]: a.topk(3,dim=1)
6 (tensor([[ 0.8362, -0.1068, -0.1281],
7          [ 1.7015,  1.4879,  1.2477],
8          [ 1.1297,  0.8691,  0.7188],
9          [ 0.6386,  0.3747,  0.2227]]), tensor([[3, 8, 9],
10         [8, 6, 5],
11         [6, 9, 0],
12         [4, 3, 7]]))
13
14 In [176]: a.topk(3,dim=1,largest=False)
15 (tensor([[ -2.1356, -0.5220, -0.3806],
16          [-1.2885, -0.7813, -0.4974],
17          [-1.0365, -0.9084, -0.1624],
18          [-1.4960, -1.1000, -0.8415]]), tensor([[2, 7, 5],
19         [7, 3, 9],
20         [5, 7, 3],
21         [8, 0, 6]]))
22
23 In [180]: a.kthvalue(8,dim=1)
24 Out[180]: (tensor([-0.1281,  1.2477,  0.7188,  0.2227]), tensor([9, 5, 0, 7]))
25
26 In [177]: a.kthvalue(3)
27 Out[177]: (tensor([-0.3806, -0.4974, -0.1624, -0.8415]), tensor([5, 9, 3, 6]))
28
29 In [178]: a.kthvalue(3,dim=1)
30 Out[178]: (tensor([-0.3806, -0.4974, -0.1624, -0.8415]), tensor([5, 9, 3, 6]))
```

# compare

- $>$ ,  $>=$ ,  $<$ ,  $<=$ ,  $\neq$ ,  $=$

- `torch.eq(a, b)`

- `torch.equal(a, b)`

```
1 In [182]: a>0
2 tensor([[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
3         [1, 1, 0, 0, 1, 1, 1, 0, 1, 0],
4         [1, 1, 1, 0, 1, 0, 1, 0, 1, 1],
5         [0, 0, 0, 1, 1, 1, 0, 1, 0, 0]], dtype=torch.uint8)
6
7 In [184]: torch.gt(a,0)
8 tensor([[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
9         [1, 1, 0, 0, 1, 1, 1, 0, 1, 0],
10        [1, 1, 1, 0, 1, 0, 1, 0, 1, 1],
11        [0, 0, 0, 1, 1, 1, 0, 1, 0, 0]], dtype=torch.uint8)
12
13 In [185]: a!=0
14 tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
15        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
16        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
17        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], dtype=torch.uint8)
18
19 In [186]: a=torch.ones(2,3)
20 In [187]: b=torch.randn(2,3)
21 In [188]: torch.eq(a,b)
22 tensor([[0, 0, 0],
23        [0, 0, 0]], dtype=torch.uint8)
24
25 In [189]: torch.eq(a,a)
26 tensor([[1, 1, 1],
27        [1, 1, 1]], dtype=torch.uint8)
28
29 In [190]: torch.equal(a,a)
30 Out[190]: True
```

**Thank You.**

---