PyTorch

# 统计属性

主讲人：龙良曲

# statistics

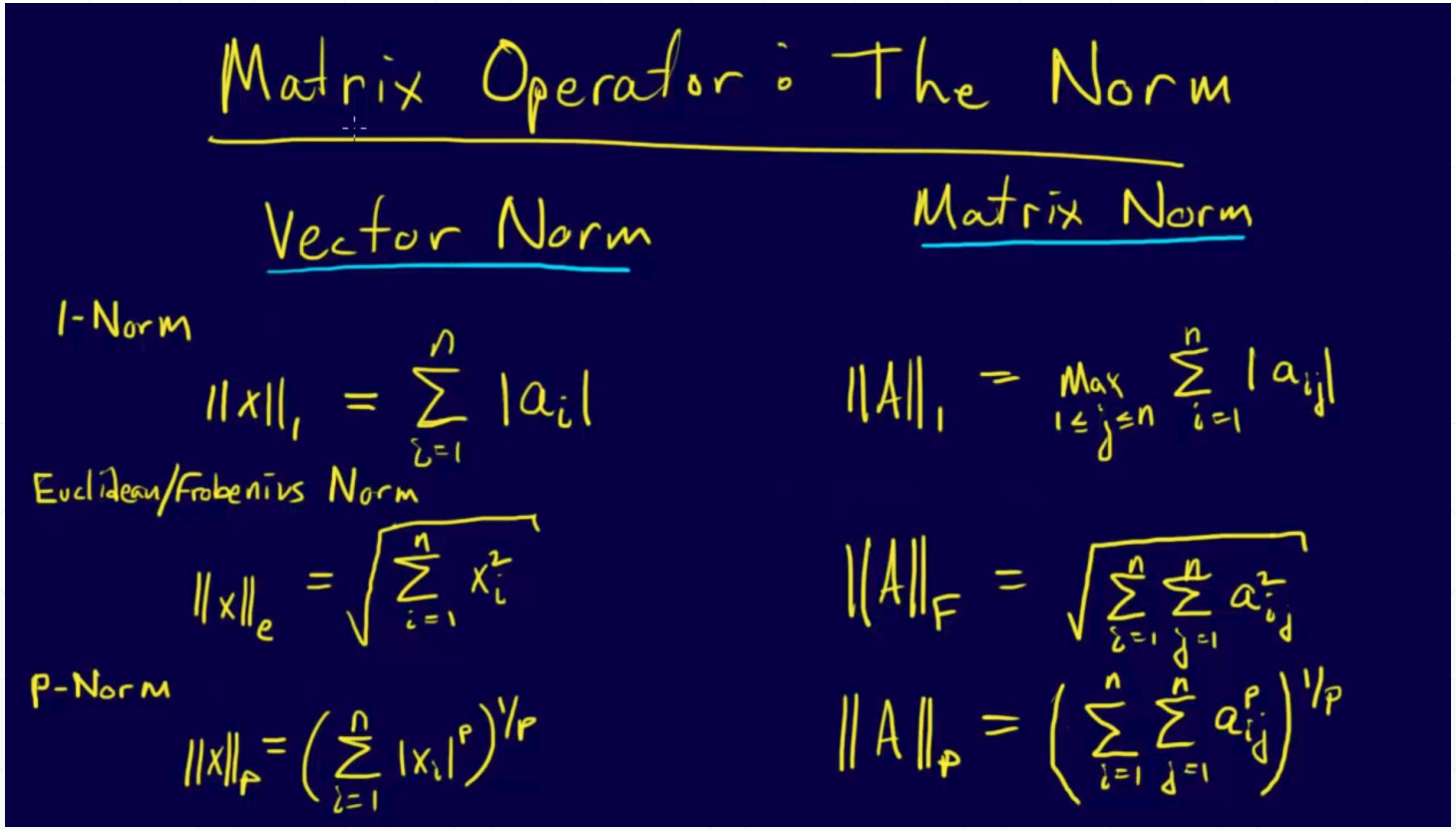
- norm
- mean sum
- prod
- max, min, argmin, argmax
- kthvalue, topk

https://blog.openai.com/generative-models/

# norm

- v.s. normalize ,e.g. batch_norm
- matrix norm v.s. vector norm

# Vector vs matrix



# norm-p

```
In [104]: a=torch.full([8], 1)
In [105]: b=a.view(2,4)
In [106]: c=a.view(2,2,2)
In [107]: b
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.]])
In [108]: c
tensor([[[1., 1.],
         [1., 1.]],
        [[1., 1.],
         [1., 1.]]])

In [109]: a.norm(1),b.norm(1),c.norm(1)
Out[109]: (tensor(8.), tensor(8.), tensor(8.))

In [110]: a.norm(2),b.norm(2),c.norm(2)
Out[110]: (tensor(2.8284), tensor(2.8284), tensor(2.8284))

In [111]: b.norm(1, dim=1)
Out[111]: tensor([4., 4.])
In [112]: b.norm(2, dim=1)
Out[112]: tensor([2., 2.])

In [113]: c.norm(1, dim=0)
tensor([[2., 2.],
        [2., 2.]])
In [114]: c.norm(2, dim=0)
tensor([[1.4142, 1.4142],
        [1.4142, 1.4142]])
```

# mean, sum, min, max, prod

```
In [143]: a=torch.arange(8).view(2,4).float()
tensor([[0, 1, 2, 3],
        [4, 5, 6, 7]])

In [144]: a.min(),a.max(),a.mean(),a.prod()
Out[144]: (tensor(0.), tensor(7.), tensor(3.5000), tensor(0.))

In [145]: a.sum()
Out[145]: tensor(28.)

In [147]: a.argmax(),a.argmin()
Out[147]: (tensor(7), tensor(0))
```

# argmin, argmax

```
In [149]: a=a.view(1,2,4)
tensor([[[0., 1., 2., 3.],
         [4., 5., 6., 7.]]])

In [151]: a.argmax()
Out[151]: tensor(7)

In [152]: a.argmin()
Out[152]: tensor(0)

In [153]: a=torch.rand(2,3,4)

In [154]: a.argmax()
Out[154]: tensor(21)
```

```
In [155]: a=torch.randn(4,10)

In [156]: a[0]
Out[156]:
tensor([-0.2636, -0.2958, -2.1356,  0.8362, -0.3137, -0.3806, -0.3547, -0.5220,
        -0.1068, -0.1281])

In [157]: a.argmax()
Out[157]: tensor(18)

In [158]: a.argmax(dim=1)
Out[158]: tensor([3, 8, 6, 4])
```

# dim, keepdim

```
In [168]: a # [4, 10]
tensor([[-0.2636, -0.2958, -2.1356,  0.8362, -0.3137, -0.3806, -0.3547, -0.5220,
         -0.1068, -0.1281],
        [ 0.2576,  1.1446, -0.4259, -0.7813,  0.1606,  1.2477,  1.4879, -1.2885,
          1.7015, -0.4974],
        [ 0.7188,  0.4481,  0.1466, -0.1624,  0.3597, -1.0365,  1.1297, -0.9084,
          0.4212,  0.8691],
        [-1.1000, -0.0078, -0.7398,  0.3747,  0.6386,  0.1516, -0.8415,  0.2227,
         -1.4960, -0.1437]])

In [170]: a.max(dim=1)
Out[170]: (tensor([0.8362, 1.7015, 1.1297, 0.6386]), tensor([3, 8, 6, 4]))
In [171]: a.argmax(dim=1)
Out[171]: tensor([3, 8, 6, 4])

In [173]: a.max(dim=1,keepdim=True)
(tensor([[0.8362],
         [1.7015],
         [1.1297],
         [0.6386]]), tensor([[3],
         [8],
         [6],
         [4]]))

In [172]: a.argmax(dim=1, keepdim=True)
tensor([[3],
        [8],
        [6],
        [4]])
```

# Top-k or k-th

- .topk
  - Largest
- kthvalue

```
tensor([[-0.2636, -0.2958, -2.1356,  0.8362, -0.3137, -0.3806, -0.3547, -0.5220,
         -0.1068, -0.1281],
        ...)

In [175]: a.topk(3,dim=1)
(tensor([[ 0.8362, -0.1068, -0.1281],
         [ 1.7015,  1.4879,  1.2477],
         [ 1.1297,  0.8691,  0.7188],
         [ 0.6386,  0.3747,  0.2227]]), tensor([[3, 8, 9],
         [8, 6, 5],
         [6, 9, 0],
         [4, 3, 7]]))

In [176]: a.topk(3,dim=1,largest=False)
(tensor([[-2.1356, -0.5220, -0.3806],
         [-1.2885, -0.7813, -0.4974],
         [-1.0365, -0.9084, -0.1624],
         [-1.4960, -1.1000, -0.8415]]), tensor([[2, 7, 5],
         [7, 3, 9],
         [5, 7, 3],
         [8, 0, 6]]))

In [180]: a.kthvalue(8,dim=1)
Out[180]: (tensor([-0.1281,  1.2477,  0.7188,  0.2227]), tensor([9, 5, 0, 7]))

In [177]: a.kthvalue(3)
Out[177]: (tensor([-0.3806, -0.4974, -0.1624, -0.8415]), tensor([5, 9, 3, 6]))

In [178]: a.kthvalue(3,dim=1)
Out[178]: (tensor([-0.3806, -0.4974, -0.1624, -0.8415]), tensor([5, 9, 3, 6]))
```

# compare

- >, >=, <, <=, !=, ==
- torch.eq(a, b)
  - torch.equal(a, b)

```
In [182]: a>0
tensor([[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [1, 1, 0, 0, 1, 1, 1, 0, 1, 0],
        [1, 1, 1, 0, 1, 0, 1, 0, 1, 1],
        [0, 0, 0, 1, 1, 1, 0, 1, 0, 0]], dtype=torch.uint8)

In [184]: torch.gt(a,0)
tensor([[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [1, 1, 0, 0, 1, 1, 1, 0, 1, 0],
        [1, 1, 1, 0, 1, 0, 1, 0, 1, 1],
        [0, 0, 0, 1, 1, 1, 0, 1, 0, 0]], dtype=torch.uint8)

In [185]: a!=0
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], dtype=torch.uint8)

In [186]: a=torch.ones(2,3)
In [187]: b=torch.randn(2,3)
In [188]: torch.eq(a,b)
tensor([[0, 0, 0],
        [0, 0, 0]], dtype=torch.uint8)

In [189]: torch.eq(a,a)
tensor([[1, 1, 1],
        [1, 1, 1]], dtype=torch.uint8)

In [190]: torch.equal(a,a)
Out[190]: True
```

# Thank You.