
tensorflow / tensorflow
TensorFlow: Open sou
Clone this repo:
git clone https://tensor
Branches
master

8.5 张量运算

基本数学运算

加减乘除运算

算术操作	描 述
tf.add(x, y)	将x和y逐元素相加
tf.subtract(x, y)	将x和y逐元素相减
tf.multiply(x, y)	将x和y逐元素相乘
tf.divide(x, y)	将x和y逐元素相除
tf.math.mod(x, y)	对x逐元素取模

参数分别是参与运算的两个张量

```
In [1]: import tensorflow as tf

In [2]: a = tf.constant([0,1,2])
        b = tf.constant([3,4,5])
        tf.add(a, b)

Out[2]: <tf.Tensor: id=2, shape=(3,), dtype=int32, numpy=array([3, 5, 7])>
```

幂指对数运算

算术操作	描 述
tf.pow(x, y)	对x求y的幂次方
tf.square(x)	对x逐元素求计算平方
tf.sqrt(x)	对x逐元素开平方根
tf.exp(x)	计算e的x次方
tf.math.log(x)	计算自然对数,底数为e

一维张量幂运算

```
In [3]: x = tf.range(4)
        x
Out[3]: <tf.Tensor: id=6, shape=(4,), dtype=int32, numpy=array([0, 1, 2, 3])>

In [4]: tf.pow(x, 2)
Out[4]: <tf.Tensor: id=8, shape=(4,), dtype=int32, numpy=array([0, 1, 4, 9])>
```

对x张量中的每个元素计算了2次方

二维张量幂运算

```
In [5]: x=tf.constant([[2, 2], [3, 3]])
        y=tf.constant([[8, 16], [2, 3]])
        tf.pow(x, y)
Out[5]: <tf.Tensor: id=11, shape=(2, 2), dtype=int32, numpy=
        array([[  256, 65536],
               [    9,    27]])>

In [6]: x=tf.constant([1., 4., 9., 16.])
        pow(x, 0.5)
Out[6]: <tf.Tensor: id=14, shape=(4,), dtype=float32, numpy=array([1., 2., 3., 4.], dtype=float32)>
```

张量的元素必须是浮点数类型

平方根

平方和平方根运算

```
In [7]: x=tf.constant([1,2,3,4])
        tf.square(x)
Out[7]: <tf.Tensor: id=16, shape=(4,), dtype=int32, numpy=array([ 1,  4,  9, 16])>

In [8]: x=tf.constant([1.,4.,9.,16.])
        tf.sqrt(x)
Out[8]: <tf.Tensor: id=18, shape=(4,), dtype=float32, numpy=array([1., 2., 3., 4.], dtype=float32)>

In [9]: f = tf.constant([[1., 9.], [16., 100.]])
        tf.sqrt(f)
Out[9]: <tf.Tensor: id=20, shape=(2, 2), dtype=float32, numpy=
        array([[ 1.,  3.],
               [ 4., 10.]], dtype=float32)>
```

自然指数和自然对数运算

TensorFlow中只有以e为底的自然对数，没有提供以其他数值为底的对数运算函数

```
In [10]: tf.exp(1.)
Out[10]: <tf.Tensor: id=22, shape=(), dtype=float32, numpy=2.7182817>

In [11]: x=tf.exp(3.)
         tf.math.log(x)
Out[11]: <tf.Tensor: id=25, shape=(), dtype=float32, numpy=3.0>
```

张量的元素必须是浮点数类型

自然对数在math模块中

- 对数运算

$$\log_a b = \frac{\log_c b}{\log_c a}$$

要计算其他底数的对数，可以利用对数的换底公式，间接的通过 tf.math.log(x)函数来实现。

```
In [12]: x=tf.constant(256.)
         y=tf.constant(2.)
         tf.math.log(x)/tf.math.log(y)
Out[12]: <tf.Tensor: id=30, shape=(), dtype=float32, numpy=8.0>
```

$$\log_2 256 = \frac{\log_e 256}{\log_e 2}$$

```
In [13]: x = tf.constant([[1., 9.], [16., 100.]])
         y = tf.constant([[2., 3.], [2., 10.]])
         tf.math.log(x) / tf.math.log(y)
Out[13]: <tf.Tensor: id=35, shape=(2, 2), dtype=float32, numpy=
         array([[0., 2.],
                [4., 2.]], dtype=float32)>
```

真数

底数

其他运算

函 数	描 述
tf.sign(x)	返回x的符号
tf.abs(x)	对x逐元素求绝对值
tf.negative(x)	对x逐元素求相反数，y = -x
tf.reciprocal(x)	取x的倒数
tf.logical_not(x)	对x逐元素求的逻辑非
tf.ceil(x)	向上取整
tf.floor(x)	向下取整
tf.rint(x)	取最接近的整数
tf.round(x)	对x逐元素求舍入最接近的整数
tf.maximum(x, y)	返回两tensor中的最大值
tf.minimum(x, y)	返回两tensor中的最小值

三角函数和反三角函数运算

函 数	描 述
tf.cos(x)	三角函数cos
tf.sin(x)	三角函数sin
tf.tan(x)	三角函数tan
tf.acos(x)	反三角函数arccos
tf.asin(x)	反三角函数arcsin
tf.atan(x)	反三角函数arctan

重载运算符

运算符	构造方法	运算符	构造方法
x+y	tf.add( )	x&y	tf.logical_and( )
x-y	tf.subtract( )	x|y	tf.logical_or( )
x*y	tf.multiply( )	x^y	tf.logical_xor( )
x/y(python2.0)	tf.divide( )	~x	tf.logical_not()
x/y(python3.0)	tf.truediv( )	x<y	tf.less( )
X//y(python3.0)	tf.floordiv( )	x<=y	tf.less_equal( )
x%y	tf.math.mod( )	x>y	tf.greater( )
x**y	tf.pow( )	x>=y	tf.greater_equal( )
-x	tf.neg( )		
abs(x)	tf.abs()		

```
In [14]: a = tf.constant([0, 1, 2, 3])
         b = tf.constant([4, 5, 6, 7])
         a+b
Out[14]: <tf.Tensor: id=38, shape=(4,), dtype=int32, numpy=array([ 4,  6,  8, 10])>

In [15]: a-b
Out[15]: <tf.Tensor: id=39, shape=(4,), dtype=int32, numpy=array([-4, -4, -4, -4])>

In [16]: a*b
Out[16]: <tf.Tensor: id=40, shape=(4,), dtype=int32, numpy=array([ 0,  5, 12, 21])>

In [17]: a/b
Out[17]: <tf.Tensor: id=43, shape=(4,), dtype=float64, numpy=array([0.        , 0.2
               , 0.33333333, 0.42857143])>
```

```
In [18]: a=tf.constant([0,1,2,3])
         b = 2
         a%b
Out[18]: <tf.Tensor: id=46, shape=(4,), dtype=int32, numpy=array([0, 1, 0, 1])>

In [19]: a=tf.constant([[0,1,2,3],[0,-1,-2,-3]])
         a//b
Out[19]: <tf.Tensor: id=49, shape=(2, 4), dtype=int32, numpy=
         array([[ 0,  0,  1,  1],
                [ 0, -1, -1, -2]])>

In [20]: a=tf.constant([0,1,2,3])
         b=2
         a**b
Out[20]: <tf.Tensor: id=52, shape=(4,), dtype=int32, numpy=array([0, 1, 4, 9])>
```

广播机制（broadcasting）

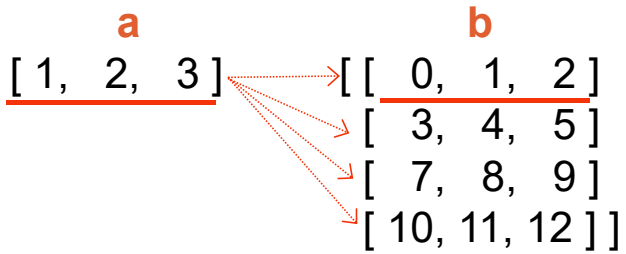
```
In [21]: a = tf.constant([1,2,3])
         a

Out[21]: <tf.Tensor: id=53, shape=(3,), dtype=int32, numpy=array([1, 2, 3])>

In [22]: import numpy as np
         b=tf.constant(np.arange(12).reshape(4,3))
         b

Out[22]: <tf.Tensor: id=54, shape=(4, 3), dtype=int32, numpy=
         array([[ 0,  1,  2],
                [ 3,  4,  5],
                [ 6,  7,  8],
                [ 9, 10, 11]])>
```

一维张量+二维张量



两个张量**最后一个维度**的长度必须相等

```
In [23]: a+b

Out[23]: <tf.Tensor: id=55, shape=(4, 3), dtype=int32, numpy=
         array([[ 1,  3,  5],
                [ 4,  6,  8],
                [ 7,  9, 11],
                [10, 12, 14]])>
```

- 一维张量+三维张量

```
In [24]: a = tf.constant([1,2,3])
         b = tf.constant(np.arange(12).reshape(2,2,3))
         b
```

(3,)

最后一个维度的长度相等

```
Out[24]: <tf.Tensor: id=57, shape=(2, 2, 3), dtype=int32, numpy=
         array([[[ 0,  1,  2],
                 [ 3,  4,  5]],

                [[ 6,  7,  8],
                 [ 9, 10, 11]]])>
```

[1, 2, 3]

```
In [25]: a+b
```

```
Out[25]: <tf.Tensor: id=58, shape=(2, 2, 3), dtype=int32, numpy=
         array([[[ 1,  3,  5],
                 [ 4,  6,  8]],

                [[ 7,  9, 11],
                 [10, 12, 14]]])>
```

```
In [26]: a*b

Out[26]: <tf.Tensor: id=59, shape=(2, 2, 3), dtype=int32, numpy=
         array([[[ 0,  2,  6],
                 [ 3,  8, 15]],

                [[ 6, 14, 24],
                 [ 9, 20, 33]]])>
```

[1, 2, 3]

- 数字+N维张量

  当张量和一个**数字**进行运算时，会将这个数字值广播到张量的各个元素

张量和NumPy数组之间的相互转换

NumPy数组转化为张量：tf.constant()； tf.convert_to_tensor

张量转换为NumPy数组：Tensor.numpy()

当张量和NumPy数组共同参与运算时:

- 执行TensorFlow操作，TensorFlow将自动的把NumPy数组转换为张量

- 执行NumPy操作，NumPy将自动的张量转换为NumPy数组

```
In [27]: nd = np.ones([2,2])
         t = tf.multiply(nd, 36)
         t

Out[27]: <tf.Tensor: id=62, shape=(2, 2), dtype=float64, numpy=
         array([[36., 36.],
                [36., 36.]])>

In [28]: np.add(nd,t)

Out[28]: array([[37., 37.],
                [37., 37.]])
```

使用运算符操作

只要操作数中**有一个Tensor对象**，就把所有的操作数都**转化为张量**，然后再进行运算。

```
In [29]: a=tf.constant([1,2,3])
         b=np.array((4,5,6))
         a+b
Out[29]: <tf.Tensor: id=65, shape=(3,), dtype=int32, numpy=array([5, 7, 9])>

In [30]: a=np.array((4,5,6))
         b=np.ones(3)
         a+b
Out[30]: array([5., 6., 7.])

In [31]: a=1
         b=2
         a+b
Out[31]: 3
```

张量

运算符重载为张量加法

2个NumPy数组相加

Python整数相加

张量乘法

- **元素乘法**：tf.multiply()，*运算符

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \times \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 0 & 3 \\ 4 & 12 \end{bmatrix}$$

- **向量乘法**：tf.matmul()，@运算符

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 10 & 13 \\ 28 & 40 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 10 & 13 \\ 28 & 40 \end{bmatrix}$$

创建张量

```
In [1]: import tensorflow as tf
        print("TensorFlow version:", tf.__version__)

        TensorFlow version: 2.0.0

In [2]: import numpy as np
        a = tf.constant(np.arange(6), shape=(2, 3))
        a

Out[2]: <tf.Tensor: id=2, shape=(2, 3), dtype=int32, numpy=
        array([[0, 1, 2],
               [3, 4, 5]])>

In [3]: b = tf.constant(np.arange(6), shape=(3, 2))
        b

Out[3]: <tf.Tensor: id=5, shape=(3, 2), dtype=int32, numpy=
        array([[0, 1],
               [2, 3],
               [4, 5]])>
```

定义张量a

定义张量b

向量乘法

tf.matmul()，@运算符

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 10 & 13 \\ 28 & 40 \end{bmatrix}$$

```
In [4]: tf.matmul(a, b)
Out[4]: <tf.Tensor: id=6, shape=(2, 2), dtype=int32, numpy=
        array([[10, 13],
               [28, 40]])>

In [5]: a@b
Out[5]: <tf.Tensor: id=7, shape=(2, 2), dtype=int32, numpy=
        array([[10, 13],
               [28, 40]])>
```

多维向量乘法—— 三维张量×二维张量

- 最后两维做向量乘法
- 高维采用广播机制

```
In [6]: a = tf.random.normal([2,3,5])
        b = tf.random.normal([5,4])
        tf.matmul(a,b)
```

(3,5)×(5,4)→(3,4)——广播→(2,3,4)

```
Out[6]: <tf.Tensor: id=20, shape=(2, 3, 4), dtype=float32, numpy=
        array([[[ 0.13672318, -0.3784006 , -0.2231093 , -1.2624099 ],
                [ 2.196111  ,  1.2696512 , -4.518535  ,  3.6673522 ],
                [ 0.62343776, -2.2611952 ,  1.4009535 ,  0.22248505]],

               [[-0.7526479 , -0.5529423 ,  2.294831  , -2.4406729 ],
                [-0.12710276,  1.5016136 ,  1.420251  ,  0.81047237],
                [ 1.014232  ,  1.3020521 , -4.1052294 ,  1.0766649 ]]],
              dtype=float32)>
```

■ 多维向量乘法—— 三维张量×三维张量

创建张量

```
In [7]: a = tf.constant(np.arange(12),shape=(2,2,3))
        a

Out[7]: <tf.Tensor: id=23, shape=(2, 2, 3), dtype=int32, numpy=
        array([[[ 0,  1,  2],
                [ 3,  4,  5]],

               [[ 6,  7,  8],
                [ 9, 10, 11]]])>

In [8]: b = tf.constant(np.arange(12),shape=(2,3,2))
        b

Out[8]: <tf.Tensor: id=26, shape=(2, 3, 2), dtype=int32, numpy=
        array([[[ 0,  1],
                [ 2,  3],
                [ 4,  5]],

               [[ 6,  7],
                [ 8,  9],
                [10, 11]]])>
```

■ 多维向量乘法—— 三维张量×三维张量

张量乘法过程

- 最后两维做向量乘法
- 高维采用广播机制

(2,3)×(3,2)→(2,2)

广播→(2,2,2)

```
In [7]: a = tf.constant(np.arange(12),shape=(2,2,3))
        a

Out[7]: <tf.Tensor: id=23, shape=(2, 2, 3), dtype=int32, numpy=
        array([[[ 0,  1,  2],
                [ 3,  4,  5]],

               [[ 6,  7,  8],
                [ 9, 10, 11]]])>

In [8]: b = tf.constant(np.arange(12),shape=(2,3,2))
        b

Out[8]: <tf.Tensor: id=26, shape=(2, 3, 2), dtype=int32, numpy=
        array([[[ 0,  1],
                [ 2,  3],
                [ 4,  5]],

               [[ 6,  7],
                [ 8,  9],
                [10, 11]]])>
```

$$\begin{bmatrix} 10 & 13 \\ 28 & 40 \end{bmatrix}$$

$$\begin{bmatrix} 172 & 193 \\ 244 & 274 \end{bmatrix}$$

■ 多维向量乘法—— 三维张量×三维张量

运行结果

```
In [9]: a @ b
Out[9]: <tf.Tensor: id=27, shape=(2, 2, 2), dtype=int32, numpy=
        array([[[ 10,  13],
                [ 28,  40]],

               [[172, 193],
                [244, 274]]])>
```

■ 多维向量乘法—— 四维张量×四维张量

```
In [10]: a = tf.constant(np.arange(24), shape=(2, 2, 2, 3))
         a

Out[10]: <tf.Tensor: id=30, shape=(2, 2, 2, 3), dtype=int32, numpy=
         array([[[[ 0,  1,  2],
                  [ 3,  4,  5]],

                 [[ 6,  7,  8],
                  [ 9, 10, 11]]],


                [[[12, 13, 14],
                  [15, 16, 17]],

                 [[18, 19, 20],
                  [21, 22, 23]]]])>
```

创建张量

```
In [11]: b = tf.constant(np.arange(24), shape=(2, 2, 3, 2))
         b

Out[11]: <tf.Tensor: id=33, shape=(2, 2, 3, 2), dtype=int32, numpy=
         array([[[[ 0,  1],
                  [ 2,  3],
                  [ 4,  5]],

                 [[ 6,  7],
                  [ 8,  9],
                  [10, 11]]],


                [[[12, 13],
                  [14, 15],
                  [16, 17]],

                 [[18, 19],
                  [20, 21],
                  [22, 23]]]])>
```

■ 多维向量乘法—— 四维张量×四维张量

```
In [10]: a = tf.constant(np.arange(24), shape=(2, 2, 2, 3))
         a
Out[10]: <tf.Tensor: id=30, shape=(2, 2, 2, 3), dtype=int32, numpy=
         array([[[[ 0,  1,  2],
                  [ 3,  4,  5]],

                 [[ 6,  7,  8],
                  [ 9, 10, 11]]],


                [[[12, 13, 14],
                  [15, 16, 17]],

                 [[18, 19, 20],
                  [21, 22, 23]]]])>
```

张量乘法过程

```
In [11]: b = tf.constant(np.arange(24), shape=(2, 2, 3, 2))
         b
Out[11]: <tf.Tensor: id=33, shape=(2, 2, 3, 2), dtype=int32, numpy=
         array([[[[ 0,  1],
                  [ 2,  3],
                  [ 4,  5]],

                 [[ 6,  7],
                  [ 8,  9],
                  [10, 11]]],


                [[[12, 13],
                  [14, 15],
                  [16, 17]],

                 [[18, 19],
                  [20, 21],
                  [22, 23]]]])>
```

$$\begin{bmatrix} 10 & 13 \\ 28 & 40 \end{bmatrix} \quad \begin{bmatrix} 172 & 193 \\ 244 & 274 \end{bmatrix} \quad \begin{bmatrix} 550 & 589 \\ 676 & 724 \end{bmatrix} \quad \begin{bmatrix} 1144 & 1201 \\ 1324 & 1390 \end{bmatrix}$$

(2,3)×(3,2)→(2,2)

广播→(2,2,2,2)

■ 多维向量乘法—— 四维张量×四维张量

运行结果

```
In [12]: a@b
Out[12]: <tf.Tensor: id=34, shape=(2, 2, 2, 2), dtype=int32, numpy=
array([[[[  10,   13],
         [  28,   40]],

        [[ 172,  193],
         [ 244,  274]]],


       [[[ 550,  589],
         [ 676,  724]],

        [[1144, 1201],
         [1324, 1390]]]])>
```

数据统计：求张量在某个维度上、或者全局的统计值

函 数	描 述
tf.reduce_sum(input_tensor,axis)	求和
tf.reduce_mean(input_tensor,axis)	求平均值
tf.reduce_max(input_tensor,axis)	求最大值
tf.reduce_min(input_tensor,axis)	求最小值

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

全局最大值：5

每行的最大值：[2,5]

每列的最大值：[3,4,5]

■ 求和函数——tf.reduce_sum()

```
In [13]: a = tf.constant([[1, 5, 3], [4, 2, 6]])
         a

Out[13]: <tf.Tensor: id=35, shape=(2, 3), dtype=int32, numpy=
         array([[1, 5, 3],
                [4, 2, 6]])>

In [14]: tf.reduce_sum(a, axis=0)

Out[14]: <tf.Tensor: id=37, shape=(3,), dtype=int32, numpy=array([5, 7, 9])>

In [15]: tf.reduce_sum(a, axis=1)

Out[15]: <tf.Tensor: id=39, shape=(2,), dtype=int32, numpy=array([ 9, 12])>

In [16]: tf.reduce_sum(a)

Out[16]: <tf.Tensor: id=41, shape=(), dtype=int32, numpy=21>
```

■ 求均值函数——tf.reduce_mean()

$$\underset{0}{\downarrow}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 6\end{bmatrix}$$

2.5 3.5 4.5

```
In [17]: tf.reduce_mean(a,axis=0)
Out[17]: <tf.Tensor: id=43, shape=(3,), dtype=int32, numpy=array([2, 3, 4])>

In [18]: a = tf.constant([[1.,5.,3.], [4.,2.,6.]])
         tf.reduce_mean(a,axis=0)
Out[18]: <tf.Tensor: id=46, shape=(3,), dtype=float32, numpy=array([2.5, 3.5, 4.5],
          dtype=float32)>

In [19]: a = tf.constant([[1,5,3], [4,2,6]])
         tf.reduce_mean(tf.cast(a, tf.float32), axis=0)
Out[19]: <tf.Tensor: id=50, shape=(3,), dtype=float32, numpy=array([2.5, 3.5, 4.5],
          dtype=float32)>
```

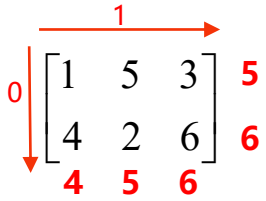
张量元素的数据类型是int32，因此求得的均值也是int32

张量元素采用浮点数

得到浮点数的均值

将张量的数据类型转换为浮点数，再求均值

■ 求最大值、最小值函数——tf.max(), tf.min()

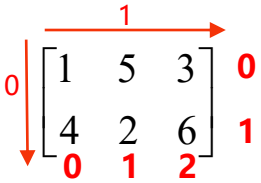



```
In [20]: tf.reduce_max(a,axis=0)
Out[20]: <tf.Tensor: id=52, shape=(3,), dtype=int32, numpy=array([4, 5, 6])>

In [21]: tf.reduce_max(a,axis=1)
Out[21]: <tf.Tensor: id=54, shape=(2,), dtype=int32, numpy=array([5, 6])>

In [22]: tf.reduce_max(a)
Out[22]: <tf.Tensor: id=56, shape=(), dtype=int32, numpy=6>
```

■ 求最值的索引——tf.argmax(), tf.argmin()




```
In [23]: a = tf.constant([[1, 5, 3], [4, 2, 6]])
         tf.argmax(a, axis=0)
Out[23]: <tf.Tensor: id=59, shape=(3,), dtype=int64, numpy=array([1, 0, 1], dtype=int64)>

In [24]: tf.argmax(a, axis=1)
Out[24]: <tf.Tensor: id=61, shape=(2,), dtype=int64, numpy=array([1, 2], dtype=int64)>

In [25]: tf.argmax(a)
Out[25]: <tf.Tensor: id=63, shape=(3,), dtype=int64, numpy=array([1, 0, 1], dtype=int64)>
```

没有指定axis参数时，默认axis=0