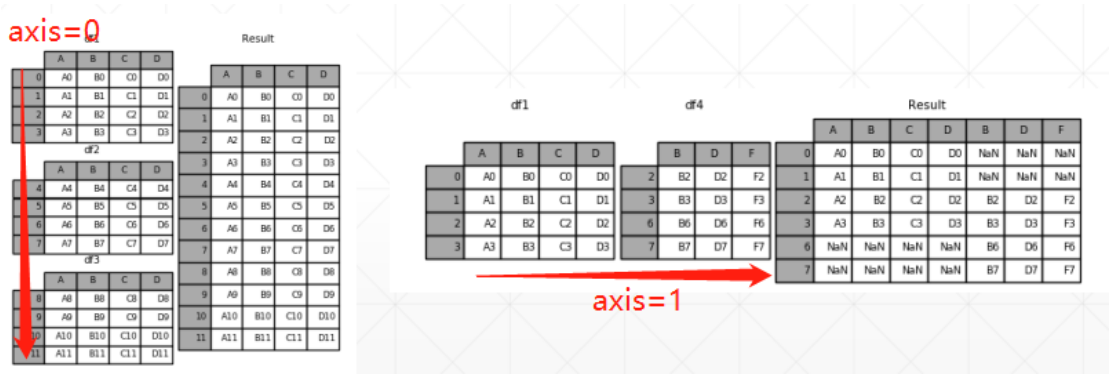
5 Tensorflow 2 高级操作

5.1 合并与分割

- ✓ tf.concat
- ✓ tf.split
- ✓ tf.stack
- ✓ tf.unstack

tf.concat

```
In [1]: import os

In [2]: os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

In [3]: import tensorflow as tf

In [4]: a = tf.ones([4, 35, 8])

In [5]: b = tf.ones([2, 35, 8])

In [6]: c = tf.concat([a, b], axis=0)

In [7]: c.shape
Out[7]: TensorShape([6, 35, 8])
```



tf.stack: create new dim

```
In [12]: a = tf.ones([4, 35, 8])

In [13]: b = tf.ones([4, 35, 8])

In [14]: a.shape, b.shape
Out[14]: (TensorShape([4, 35, 8]), TensorShape([4, 35, 8]))
```

```
In [15]: tf.concat([a,b],axis=-1).shape
Out[15]: TensorShape([4, 35, 16])

In [16]: tf.stack([a,b],axis=0).shape
Out[16]: TensorShape([2, 4, 35, 8])

In [17]: c = tf.ones([4,35,8])

In [18]: tf.stack([a,b,c], axis=3).shape
Out[18]: TensorShape([4, 35, 8, 3])
```

concat 需要满足 axis 维度可以不相等，其他维度必须相等，而 stack 则要更加严格，它要求所有的维度都必须完全一样。

tf.unstack 和 tf.split

```
In [28]: a.shape
Out[28]: TensorShape([4, 35, 8])

In [29]: b.shape
Out[29]: TensorShape([4, 35, 8])

In [30]: c = tf.stack([a,b])

In [31]: c.shape
Out[31]: TensorShape([2, 4, 35, 8])

In [32]: aa,bb = tf.unstack(c,axis=0)

In [33]: aa.shape, bb.shape
Out[33]: (TensorShape([4, 35, 8]), TensorShape([4, 35, 8]))

In [34]: res = tf.unstack(c,axis=3)

In [35]: res[0].shape, res[7].shape
Out[35]: (TensorShape([2, 4, 35]), TensorShape([2, 4, 35]))

In [36]: res = tf.unstack(c, axis=3)

In [37]: len(res)
Out[37]: 8

In [38]: res = tf.split(c,axis=3,num_or_size_splits=2)

In [39]: len(res)
Out[39]: 2

In [40]: res[0].shape
Out[40]: TensorShape([2, 4, 35, 4])

In [41]: res = tf.split(c,axis=3,num_or_size_splits=[2,2,4])

In [42]: len(res)
Out[42]: 3

In [43]: res[0].shape, res[2].shape
Out[43]: (TensorShape([2, 4, 35, 2]), TensorShape([2, 4, 35, 4]))
```

5.2 数据统计

- ✓ tf.norm
- ✓ tf.reduce_min/max
- ✓ tf.argmax/argmin
- ✓ tf.equal
- ✓ tf.unique

Vector Norm

$$\begin{aligned} &\textbf{Eukl. Norm} && \|x\|_2 = \left[\sum_k x_k^2\right]^{1/2} \\ &\textbf{Max.norm} && \|x\|_\infty = \max_k |x_k| \\ &L_1\textbf{-Norm} && \|x\|_1 = \sum_k |x_k| \end{aligned}$$

tf.norm

```
In [44]: a = tf.ones([2,2])

In [45]: tf.norm(a)
Out[45]: <tf.Tensor: id=73, shape=(), dtype=float32, numpy=2.0>

In [46]: tf.sqrt(tf.reduce_sum(tf.square(a)))
Out[46]: <tf.Tensor: id=77, shape=(), dtype=float32, numpy=2.0>

In [47]: a = tf.ones([4,28,28,3])

In [48]: tf.norm(a)
Out[48]: <tf.Tensor: id=85, shape=(), dtype=float32, numpy=96.99484>

In [49]: tf.sqrt(tf.reduce_sum(tf.square(a)))
Out[49]: <tf.Tensor: id=89, shape=(), dtype=float32, numpy=96.99484>

In [51]: a = tf.ones([2,2])

In [52]: tf.norm(a,ord=2,axis=1)
Out[52]: <tf.Tensor: id=102, shape=(2,), dtype=float32, numpy=array([1.4142135, 1.4142135], dtype=float32)>

In [53]: tf.norm(a,ord=1)
Out[53]: <tf.Tensor: id=106, shape=(), dtype=float32, numpy=4.0>
```

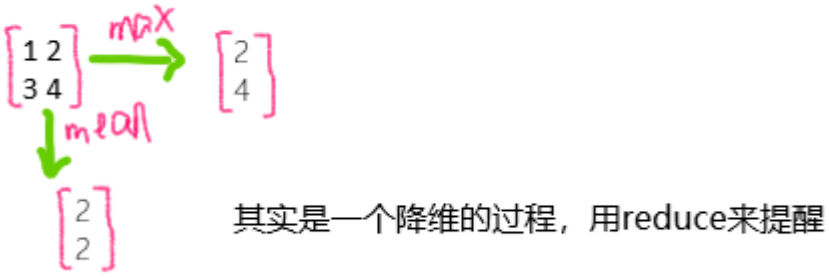
[1 1] 看成一个整体

[1 1]

1范数

```
In [55]: tf.norm(a,ord=1,axis=0)
Out[55]: <tf.Tensor: id=114, shape=(2,), dtype=float32, numpy=array([2., 2.], dtype=float32)>

In [56]: tf.norm(a,ord=1,axis=1)
Out[56]: <tf.Tensor: id=118, shape=(2,), dtype=float32, numpy=array([2., 2.], dtype=float32)>
```

tf.reduce_min/max/mean




```
In [57]: a = tf.random.normal([4,10])

In [58]: tf.reduce_min(a), tf.reduce_max(a), tf.reduce_mean(a)
Out[58]:
(<tf.Tensor: id=126, shape=(), dtype=float32, numpy=-2.4246807>,
 <tf.Tensor: id=128, shape=(), dtype=float32, numpy=1.504885>,
 <tf.Tensor: id=130, shape=(), dtype=float32, numpy=0.08340641>)

In [59]: tf.reduce_min(a,axis=1), tf.reduce_max(a,axis=1), tf.reduce_mean(a,axis=1)
Out[59]:
(<tf.Tensor: id=132, shape=(4,), dtype=float32, numpy=array([-0.91725117, -1.6169581 , -1.7778311 , -2.4246807 ], dtype=float32)>,
 <tf.Tensor: id=134, shape=(4,), dtype=float32, numpy=array([1.4219397 , 1.4947684 , 1.504885  , 0.85333383], dtype=float32)>,
 <tf.Tensor: id=136, shape=(4,), dtype=float32, numpy=array([ 0.32677358, -0.03526079,  0.09932694, -0.0572141 ], dtype=float32)>)
```

不指定轴，会flatten为一个向量
对这个向量进行_后面的操作

指定轴 [4,10]
axis=1

tf.argmax/argmin

```
In [60]: a.shape
Out[60]: TensorShape([4, 10])

In [61]: tf.argmax(a).shape
Out[61]: TensorShape([10])

In [62]: tf.argmax(a)
Out[62]: <tf.Tensor: id=140, shape=(10,), dtype=int64, numpy=array([0, 0, 2, 1, 2, 0, 2, 3, 0, 0], dtype=int64)>

In [63]: tf.argmin(a).shape
Out[63]: TensorShape([10])
```

未指定轴，默认为axis=0

返回最大/最小值的索引

tf.equal

```
In [64]: a = tf.constant([1,2,3,4,5])

In [65]: b = tf.range(5)

In [66]: tf.equal(a,b)
Out[66]: <tf.Tensor: id=148, shape=(5,), dtype=bool, numpy=array([False, False, False, False, False])>

In [67]: res = tf.equal(a,b)

In [68]: tf.reduce_sum(tf.cast(res,dtype=tf.int32))
Out[68]: <tf.Tensor: id=152, shape=(), dtype=int32, numpy=0>
```

下面以准确度的计算来说明怎么用：

```
In [71]: a = tf.constant([[0.1,0.2,0.7],[0.9,0.05,0.05]])

In [72]: a
Out[72]:
<tf.Tensor: id=159, shape=(2, 3), dtype=float32, numpy=
array([[0.1 , 0.2 , 0.7 ],
       [0.9 , 0.05, 0.05]], dtype=float32)>

In [73]: pred = tf.cast(tf.argmax(a,axis=1), dtype=tf.int32)

In [74]: pred
Out[74]: <tf.Tensor: id=162, shape=(2,), dtype=int32, numpy=array([2, 0])>
```

一个向量，既可以是行也可以是列

```
In [75]: y = tf.constant([2,1],dtype=tf.int32)

In [76]: y
Out[76]: <tf.Tensor: id=163, shape=(2,), dtype=int32, numpy=array([2, 1])>

In [77]: tf.equal(y,pred)
Out[77]: <tf.Tensor: id=164, shape=(2,), dtype=bool, numpy=array([ True, False])>

In [78]: correct = tf.reduce_sum(tf.cast(tf.equal(y,pred),dtype=tf.int32))

In [79]: correct
Out[79]: <tf.Tensor: id=168, shape=(), dtype=int32, numpy=1>

In [80]: correct/2
Out[80]: <tf.Tensor: id=172, shape=(), dtype=float64, numpy=0.5>
```

tf.unique

```
In [86]: a = tf.range(5)

In [87]: tf.unique(a)
Out[87]: Unique(y=<tf.Tensor: id=188, shape=(5,), dtype=int32, numpy=array([0, 1, 2, 3, 4])>, idx=<tf.Tensor: id=189
, shape=(5,), dtype=int32, numpy=array([0, 1, 2, 3, 4])>)

In [88]: a = tf.constant([4,2,2,4,3])

In [89]: tf.unique(a)
Out[89]: Unique(y=<tf.Tensor: id=191, shape=(3,), dtype=int32, numpy=array([4, 2, 3])>, idx=<tf.Tensor: id=192, shap
e=(5,), dtype=int32, numpy=array([0, 1, 1, 0, 2])>)
```

第0和3元素相同，1和2元素相同

5.3 张量排序

- ✓ tf.sort/argsort ←→分别是返回值和值的位置
- ✓ math.top_k

sort / argsort

```
In [93]: a = tf.random.shuffle(tf.range(5))

In [94]: a
Out[94]: <tf.Tensor: id=199, shape=(5,), dtype=int32, numpy=array([4, 3, 1, 0, 2])>

In [95]: tf.sort(a, direction='DESCENDING')
Out[95]: <tf.Tensor: id=207, shape=(5,), dtype=int32, numpy=array([4, 3, 2, 1, 0])>

In [96]: tf.argsort(a, direction='DESCENDING')
Out[96]: <tf.Tensor: id=217, shape=(5,), dtype=int32, numpy=array([0, 1, 4, 2, 3])>

In [97]: idx = tf.argsort(a, direction='DESCENDING')

In [98]: tf.gather(a, idx)
Out[98]: <tf.Tensor: id=228, shape=(5,), dtype=int32, numpy=array([4, 3, 2, 1, 0])>
```

升序：ASCENDING

```
In [109]: a = tf.random.uniform([3,3],maxval=10,dtype=tf.int32)

In [110]: a
Out[110]:
<tf.Tensor: id=283, shape=(3, 3), dtype=int32, numpy=
array([[8, 1, 7],
       [7, 1, 5],
       [7, 4, 9]])>

In [111]: tf.sort(a)
Out[111]:
<tf.Tensor: id=294, shape=(3, 3), dtype=int32, numpy=
array([[1, 7, 8],
       [1, 5, 7],
       [4, 7, 9]])>

In [112]: tf.sort(a, direction='DESCENDING')
Out[112]:
<tf.Tensor: id=302, shape=(3, 3), dtype=int32, numpy=
array([[8, 7, 1],
       [7, 5, 1],
       [9, 7, 4]])>

In [113]: idx = tf.argsort(a)

In [114]: idx
Out[114]:
<tf.Tensor: id=313, shape=(3, 3), dtype=int32, numpy=
array([[1, 2, 0],
       [1, 2, 0],
       [1, 0, 2]])>
```

Top_k：only return top-k values and indices

```
In [115]: a
Out[115]:
<tf.Tensor: id=283, shape=(3, 3), dtype=int32, numpy=
array([[8, 1, 7],
       [7, 1, 5],
       [7, 4, 9]])>

In [116]: res = tf.math.top_k(a,2)

In [117]: res.indices
Out[117]:
<tf.Tensor: id=316, shape=(3, 2), dtype=int32, numpy=
array([[0, 2],
       [0, 2],
       [2, 0]])>

In [118]: res.values
Out[118]:
<tf.Tensor: id=315, shape=(3, 2), dtype=int32, numpy=
array([[8, 7],
       [7, 5],
       [9, 7]])>
```

Top Accuracy 的实战

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import tensorflow as tf
tf.random.set_seed(2467)

def accuracy(output, target, topk=(1,)):
    maxk = max(topk)
    batch_size = target.shape[0]

    pred = tf.math.top_k(output, maxk).indices
    pred = tf.transpose(pred, [1,0])
    target_ = tf.broadcast_to(target, pred.shape)
    # [10, b]
    correct = tf.equal(pred, target_)

    res = []
    for k in topk:
        correct_k = tf.cast(tf.reshape(correct[:k],[-
1]), dtype=tf.float32)
        correct_k = tf.reduce_sum(correct_k)
        acc = float(correct_k / batch_size * 100)
        res.append(acc)
    return  res

output = tf.random.normal([10, 6])
output = tf.math.softmax(output, axis=1)
target = tf.random.uniform([10], maxval=6, dtype=tf.int32)
print('prob:', output.numpy())
pred = tf.argmax(output, axis=1)
print('pred:', pred.numpy())
print('label:', target.numpy())

acc = accuracy(output, target, topk=(1,2,3,4,5,6))
print('top-1-6 acc:', acc)
```

5.4 数据的填充与复制

- ✓ pad
- ✓ tile
- ✓ broadcast_to

tf.pad

```
In [121]: a = tf.reshape(tf.range(9), [3,3])

In [122]: a
Out[122]:
<tf.Tensor: id=323, shape=(3, 3), dtype=int32, numpy=
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])>

In [123]: tf.pad(a, [[0,0],[0,0]])
Out[123]:
<tf.Tensor: id=325, shape=(3, 3), dtype=int32, numpy=
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])>
```

行 列

```
In [124]: tf.pad(a, [[1,0],[0,0]])
Out[124]:
<tf.Tensor: id=327, shape=(4, 3), dtype=int32, numpy=
array([[0, 0, 0],
       [0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])>
```

[左行 右行] [左列 右列]

0表示不填充，1表示填充一行

```
In [125]: a = tf.random.normal([4,28,28,3])

In [126]: b = tf.pad(a, [[0,0],[2,2],[2,2],[0,0]])

In [127]: b.shape
Out[127]: TensorShape([4, 32, 32, 3])
```

tf.tile

```
In [128]: a = tf.reshape(tf.range(4),[2,2])

In [129]: a
Out[129]:
<tf.Tensor: id=341, shape=(2, 2), dtype=int32, numpy=
array([[0, 1],
       [2, 3]])>
```

1表示该维度不变，2表示复制一次

```
In [130]: tf.tile(a, [1,2])
Out[130]:
<tf.Tensor: id=343, shape=(2, 4), dtype=int32, numpy=
array([[0, 1, 0, 1],
       [2, 3, 2, 3]])>
```

复制的顺序是先从小维度开始

```
In [131]: tf.tile(a, [2,2])
Out[131]:
<tf.Tensor: id=345, shape=(4, 4), dtype=int32, numpy=
array([[0, 1, 0, 1],
       [2, 3, 2, 3],
       [0, 1, 0, 1],
       [2, 3, 2, 3]])>
```

5.5 张量限幅

- ✓ clip_by_value
- ✓ clip_by_norm
- ✓ gradient clipping

tf.clip_by_value

```
In [133]: a
Out[133]: <tf.Tensor: id=349, shape=(10,), dtype=int32, numpy=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])>

In [134]: tf.maximum(a, 2)
Out[134]: <tf.Tensor: id=351, shape=(10,), dtype=int32, numpy=array([2, 2, 2, 3, 4, 5, 6, 7, 8, 9])>

In [135]: tf.minimum(a, 8)
Out[135]: <tf.Tensor: id=353, shape=(10,), dtype=int32, numpy=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 8])>

In [136]: tf.clip_by_value(a,2,8)
Out[136]: <tf.Tensor: id=357, shape=(10,), dtype=int32, numpy=array([2, 2, 2, 3, 4, 5, 6, 7, 8, 8])>
```

类似于：max(0,x) ---> if x<0, return 0 else return x

tf.clip_by_norm

对于一个向量可能是具有方向的，不希望在缩放的时候改变方向，这时该函数应运而生。

```
In [137]: a = tf.random.normal([2,2],mean=10)

In [138]: a
Out[138]:
<tf.Tensor: id=363, shape=(2, 2), dtype=float32, numpy=
array([[11.07676  ,  9.0976925],
       [ 8.858526 , 10.378995 ]], dtype=float32)>

In [139]: tf.norm(a)
Out[139]: <tf.Tensor: id=368, shape=(), dtype=float32, numpy=19.790392>

In [140]: aa = tf.clip_by_norm(a, 15)

In [141]: aa
Out[141]:
<tf.Tensor: id=385, shape=(2, 2), dtype=float32, numpy=
array([[8.395558, 6.895538],
       [6.714263, 7.866692]], dtype=float32)>

In [142]: tf.norm(aa)
Out[142]: <tf.Tensor: id=390, shape=(), dtype=float32, numpy=15.000001>
```

a的范数约定在大约15左右

Gradient Clipping

为了防止梯度爆炸和梯度消失等，会梯度进行裁剪

new_grads, total_norm = tf.clip_by_global_norm(grads, 25)

new_grads：做缩放后的梯度值

total_norm：未作 clipping 的整体的 norm

grads：当前网络所有的 gradient

25：所有的梯度的 norm，所以一般要稍微大一点

```
print('==before==')
for g in grads:
    print(tf.norm(g))
grads,  _ = tf.clip_by_global_norm(grads, 15)

print('==after==')
for g in grads:
    print(tf.norm(g))
```

5.6 高阶操作技巧

- ✓ where
- ✓ scatter_nd
- ✓ meshgrid

where

where(tensor)：返回元素为 True 的坐标(接收一个参数)

```
In [144]: a
Out[144]:
<tf.Tensor: id=396, shape=(3, 3), dtype=float32, numpy=
array([[-0.6344489 , -2.0642662 ,  1.945435  ],
       [ 0.8565526 , -0.3025947 , -0.3369328 ],
       [ 0.35196745, -0.79820174,  0.7018128 ]], dtype=float32)>

In [145]: mask = a>0

In [146]: mask
Out[146]:
<tf.Tensor: id=398, shape=(3, 3), dtype=bool, numpy=
array([[False, False,  True],
       [ True, False, False],
       [ True, False,  True]])>

In [147]: tf.boolean_mask(a, mask)
Out[147]: <tf.Tensor: id=425, shape=(4,), dtype=float32, numpy=array([1.945435  , 0.8565526 , 0.35196745, 0.7018128
], dtype=float32)>

In [148]: indices = tf.where(mask)

In [149]: indices
Out[149]:
<tf.Tensor: id=426, shape=(4, 2), dtype=int64, numpy=
array([[0, 2],
       [1, 0],
       [2, 0],
       [2, 2]], dtype=int64)>

In [150]: tf.gather_nd(a, indices)
Out[150]: <tf.Tensor: id=427, shape=(4,), dtype=float32, numpy=array([1.945435  , 0.8565526 , 0.35196745, 0.7018128
], dtype=float32)>
```

where(cond, A, B)：根据 cond 为 True/False 从 A 和 B 中进行筛选(True 选 A,False 选 B)

```
In [151]: mask
Out[151]:
<tf.Tensor: id=398, shape=(3, 3), dtype=bool, numpy=
array([[False, False,  True],
       [ True, False, False],
       [ True, False,  True]])>

In [152]: A = tf.ones([3,3])

In [153]: B = tf.zeros([3,3])

In [154]: tf.where(mask,A,B)
Out[154]:
<tf.Tensor: id=434, shape=(3, 3), dtype=float32, numpy=
array([[0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 1.]], dtype=float32)>
```

若mask对应位置为True， 就选A的数据，若为False，就选B的数据

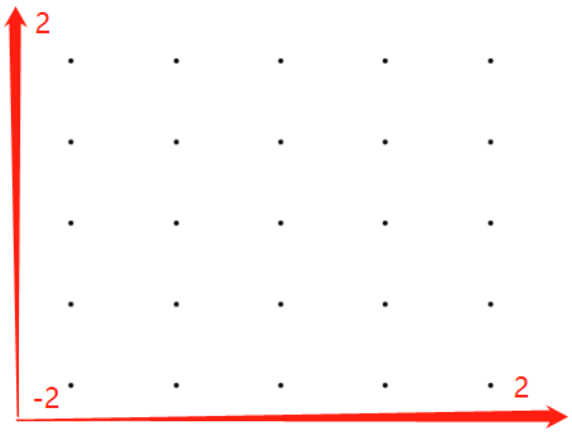
scatter_nd：根据坐标有目的性的更新


```
In [155]: indices = tf.constant([[4],[3],[1],[7]])

In [156]: updates = tf.constant([9, 10, 11, 12])

In [157]: shape = tf.constant([8])

In [158]: tf.scatter_nd(indices, updates, shape)
Out[158]: <tf.Tensor: id=438, shape=(8,), dtype=int32, numpy=array([ 0, 11,  0, 10,  9,  0,  0, 12])>
```

9更新到shape的位置4上，以此类推

meshgrid

假设要生成一个 x∈[-2,2]，y∈[-2,2]，取 5 个点



用 numpy 实现

```
points = []

for y in np.linspace(-2,2,5):
    for x in np.linspace(-2,2,5):
        points.append([x,y])
return np.array(points)
```

但是 numpy 并没有 GPU 加速，因此使用别的方法

```
In [160]: y = tf.linspace(-2.,2,5)

In [161]: y
Out[161]: <tf.Tensor: id=445, shape=(5,), dtype=float32, numpy=array([-2., -1.,  0.,  1.,  2.], dtype=float32)>

In [162]: x = tf.linspace(-2.,2,5)

In [163]: points_x, points_y = tf.meshgrid(x,y)

In [164]: points_x.shape
Out[164]: TensorShape([5, 5])


In [165]: points_x
Out[165]:
<tf.Tensor: id=467, shape=(5, 5), dtype=float32, numpy=
array([[-2., -1.,  0.,  1.,  2.],
       [-2., -1.,  0.,  1.,  2.],
       [-2., -1.,  0.,  1.,  2.],
       [-2., -1.,  0.,  1.,  2.],
       [-2., -1.,  0.,  1.,  2.]], dtype=float32)>

In [166]: points_y
Out[166]:
<tf.Tensor: id=468, shape=(5, 5), dtype=float32, numpy=
array([[-2., -2., -2., -2., -2.],
       [-1., -1., -1., -1., -1.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 2.,  2.,  2.,  2.,  2.]], dtype=float32)>

In [167]: points = tf.stack([points_x,points_y],axis=2)

In [168]: points
Out[168]:
<tf.Tensor: id=469, shape=(5, 5, 2), dtype=float32, numpy=
array([[[-2., -2.],
        [-1., -2.],
        [ 0., -2.],
        [ 1., -2.],
        [ 2., -2.]],

       [[-2., -1.],
        [-1., -1.],
        [ 0., -1.],
        [ 1., -1.],
        [ 2., -1.]],
```

所有点的x坐标

所有点的y坐标

小应用

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import tensorflow as tf
import matplotlib.pyplot as plt

def fun(x):
    z = tf.math.sin(x[...,0]) + tf.math.sin(x[...,1])
    return z

x = tf.linspace(0., 2*3.14, 500)
y = tf.linspace(0., 2*3.14, 500)
point_x, point_y = tf.meshgrid(x, y)
points = tf.stack([point_x, point_y], axis=2)
# points = tf.reshape(points, [-1, 2])
print("points", points.shape)    # (500, 500, 2)
z = fun(points)                  # (500, 500)

plt.figure("plot 2d func value")
plt.imshow(z, origin='lower', interpolation='none')
plt.colorbar()

plt.figure('plot 2d func contour')
plt.contour(point_x, point_y, z)
plt.colorbar()
plt.show()
```