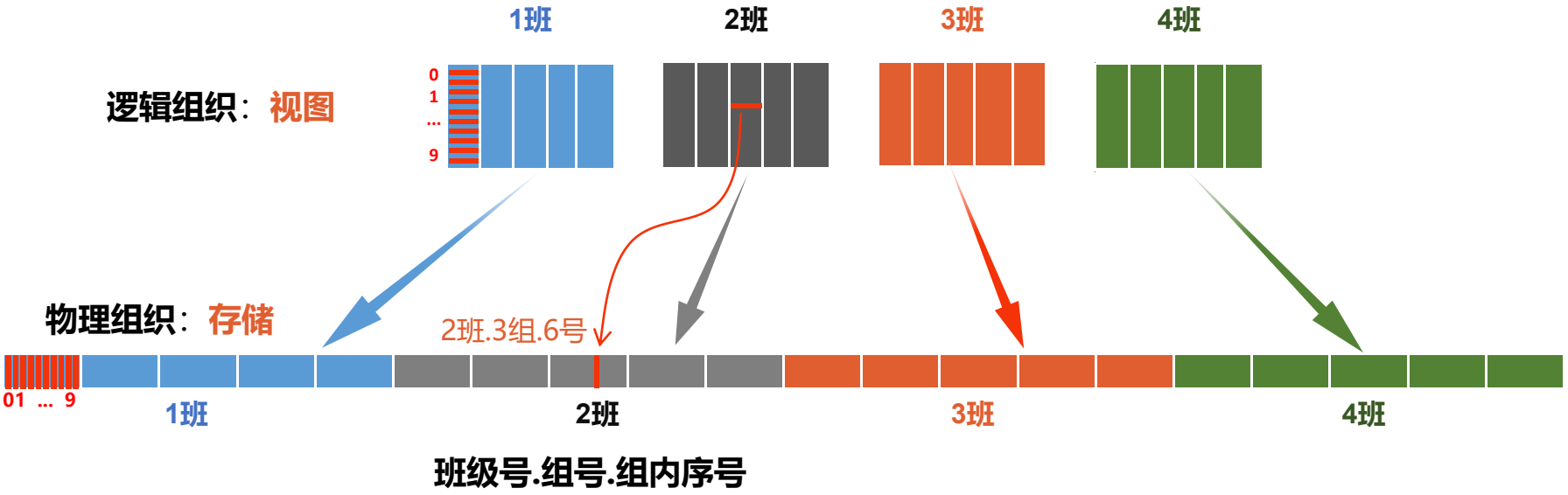

tensorflow / tensorflow
TensorFlow: Open sou
Clone this repo:
git clone https://tensor
Branches
master

8.3 维度变换

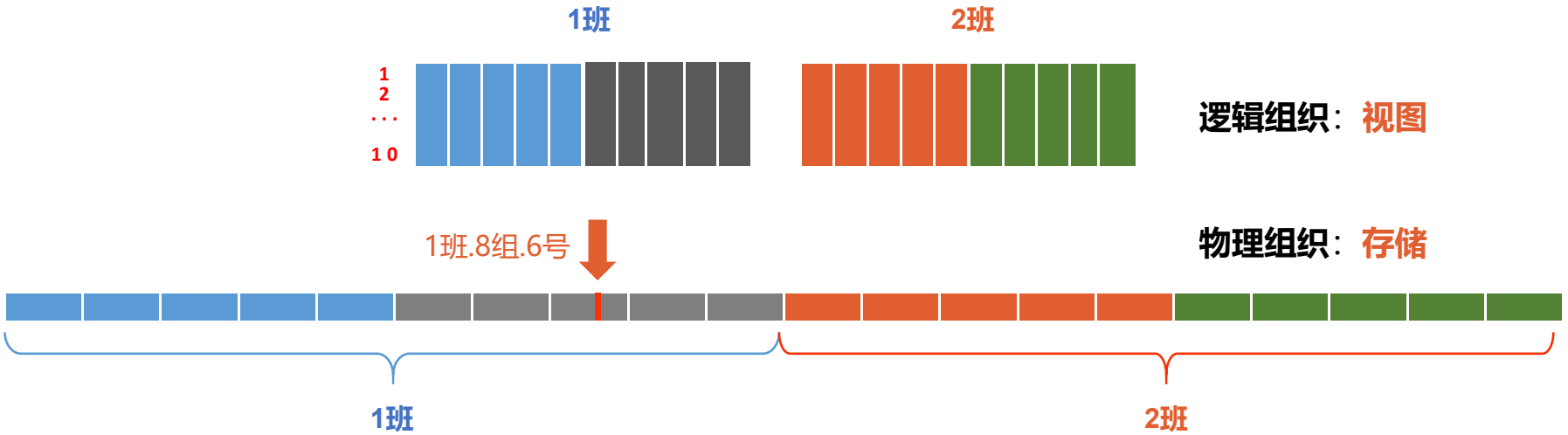
张量的存储和视图

多维张量在**物理上**以**一维**的方式连续存储
通过定义维度和形状，在**逻辑上**把它理解为**多维张量**



张量的存储和视图

当对多维张量进行维度变换时，
只是改变了逻辑上索引的方式，没有改变内存中的存储方式



改变张量的形状

tf.reshape (tensor, shape)

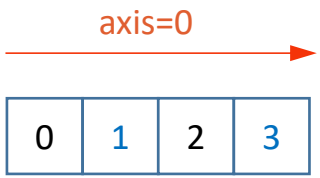
没有封装到Tensor对象中，前缀是tf，而不是张量对象

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

In [2]: a=tf.range(24)
        b=tf.reshape(a,[2,3,4])
        b

Out[2]: <tf.Tensor: id=5, shape=(2, 3, 4), 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]]])>
```

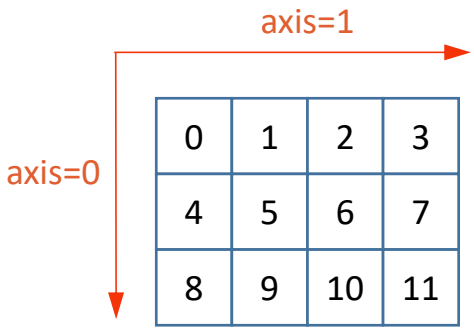
```
In [3]: import numpy as np
        tf.constant(np.arange(24).reshape(2, 3, 4))

Out[3]: <tf.Tensor: id=6, shape=(2, 3, 4), 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]]])>

In [4]: tf.reshape(b, [4, -1])

Out[4]: <tf.Tensor: id=8, shape=(4, 6), 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]])>
```

shape参数=**-1**：
自动推导出长度

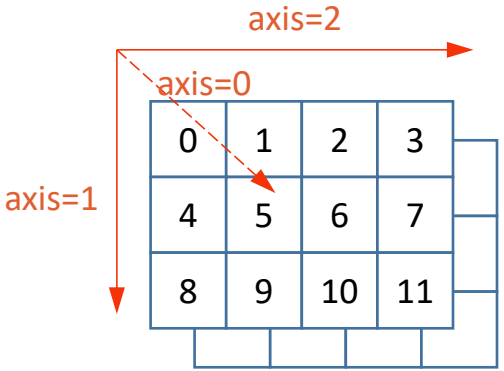
多维张量的轴：张量的维度



shape=(4,)
axis： 0



shape=(3, 4)
axis： 0, 1



shape=(2, 3, 4)
axis： 0, 1, 2
-3, -2, -1

轴也可以是负数，
表示从后向前索引

张量中的轴的概念和用法，和NumPy数组是完全一样的

增加和删除维度

增加维度

tf.expand_dims(input,axis)

增加的这个维度上，**长度为1**

```
In [5]: t = tf.constant([1,2])
        print(t.shape)

        (2,)

In [6]: t1 = tf.expand_dims(t, 1)
        print(t1.shape)

        (2, 1)

In [7]: t1

Out[7]: <tf.Tensor: id=11, shape=(2, 1), dtype=int32, numpy=
        array([[1],
               [2]])>
```

在axis=1的轴上增加维度

西安科技大学 计算机科学与技术学院

```
In [8]: t2 = tf.expand_dims(t, 0)
        print(t2.shape)

        (1, 2)

In [9]: t2

Out[9]: <tf.Tensor: id=13, shape=(1, 2), dtype=int32, numpy=ar
        ray([[1, 2]])>

In [10]: t3 = tf.expand_dims(t, -1)
         print(t3.shape)

         (2, 1)

In [11]: t3

Out[11]: <tf.Tensor: id=15, shape=(2, 1), dtype=int32, numpy=
         array([[1],
                [2]])>
```

在axis=0的轴上增加维度

```
In [12]: a=tf.range(24)
         b=tf.reshape(a,[2,3,4])
         print(b.shape)

         (2, 3, 4)

In [13]: b1=tf.expand_dims(b, 0)
         b2=tf.expand_dims(b, 1)
         b3=tf.expand_dims(b, 2)
         b4=tf.expand_dims(b, 3)
         print(b1.shape)
         print(b2.shape)
         print(b3.shape)
         print(b4.shape)

         (1, 2, 3, 4)
         (2, 1, 3, 4)
         (2, 3, 1, 4)
         (2, 3, 4, 1)
```

删除维度

```
tf.squeeze( input, axis=None )
```

原始张量　要删除的维度

只能删除**长度为 1** 的维度，
省略时删除所有长度为1的维度

例： *# t is a tensor of shape (1, 2, 1, 3, 1)*

```
>>>tf.shape(tf.squeeze(t))
(2, 3)
```

```
>>>tf.shape(tf.squeeze(t,[2,4]))
(1,2, 3)
```

增加维度和删除维度，**只是改变了张量的视图，不会改变张量的存储**

交换维度

tf.transpose(a, perm)

对二维张量交换维度，就是矩阵的转置

```
In [14]: x = tf.constant([[1, 2, 3], [4, 5, 6]])
         x

Out[14]: <tf.Tensor: id=30, shape=(2, 3), dtype=int32, numpy=
         array([[1, 2, 3],
                [4, 5, 6]])>

In [15]: tf.transpose(x)

Out[15]: <tf.Tensor: id=32, shape=(3, 2), dtype=int32, numpy=
         array([[1, 4],
                [2, 5],
                [3, 6]])>
```

交换维度

tf.transpose(a, **perm**)　调整张量中各个轴的**顺序**

```
In [16]: x = tf.constant([[1, 2, 3], [4, 5, 6]])
         x

Out[16]: <tf.Tensor: id=33, shape=(2, 3), dtype=int32, numpy=
         array([[1, 2, 3],
                [4, 5, 6]])>

In [17]: tf.transpose(x, perm=[1, 0])

Out[17]: <tf.Tensor: id=35, shape=(3, 2), dtype=int32, numpy=
         array([[1, 4],
                [2, 5],
                [3, 6]])>
```

0 1

改变轴的**顺序**，从而改变张量的形状

```
In [18]: a=tf.range(24)  0 1 2
         b=tf.reshape(a,[2,3,4])
         b
Out[18]: <tf.Tensor: id=41, shape=(2, 3, 4), 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]]])>
```

交换维度，不仅改变了张量的视图，同时也**改变了张量的存储顺序**

```
In [19]: tf.transpose(b,(1,0,2))   最后一维不变，前两维交换
Out[19]: <tf.Tensor: id=43, shape=(3, 2, 4), dtype=int32, numpy
         =
         array([[[ 0,  1,  2,  3],
                 [12, 13, 14, 15]],

                [[ 4,  5,  6,  7],
                 [16, 17, 18, 19]],

                [[ 8,  9, 10, 11],
                 [20, 21, 22, 23]]])>
```

拼接和分割

拼接张量

- 将多个张量在某个维度上**合并**
- 拼接并不会产生新的维度

tf.concat(tensors, axis)

所有需要拼接的张量列表

指定在哪个轴上进行拼接

```
In [20]: t1 = [[1, 2, 3], [4, 5, 6]]
         t2 = [[7, 8, 9], [10, 11, 12]]

In [21]: tf.concat([t1, t2], 0)
Out[21]: <tf.Tensor: id=47, shape=(4, 3), dtype=int32, numpy=
         array([[ 1,  2,  3],
                [ 4,  5,  6],
                [ 7,  8,  9],
                [10, 11, 12]])>

In [22]: tf.concat([t1, t2], 1)
Out[22]: <tf.Tensor: id=51, shape=(2, 6), dtype=int32, numpy=
         array([[ 1,  2,  3,  7,  8,  9],
                [ 4,  5,  6, 10, 11, 12]])>
```

shape: (2,3)

在axis=0的轴上拼接，2+2=4

(2, 3)
(2, 3)

axis=0

在axis=1的轴上拼接，3+3=6

(2, 3)
(2, 3)

axis=1

- 分割张量：将一个张量拆分成多个张量，分割后**维度不变**

```
tf.split( value, num_or_size_splits, axis=0 )
```

待分割张量　分割方案　指明分割的轴

分割方案：是一个**数值**时，表示**等长分割**，数值是切割的**份数**；
是一个**列表**时，表示**不等长切割**，列表中是切割后每份的长度

例：

2 ：分割成**2个**张量
[1:2:1]：就表示分割成**3个**张量，长度分别是**1,2,1**

分割张量

- 在axis=0轴上分割

```
In [23]: x=tf.range(24)
         x=tf.reshape(x,[4,6])
         x
Out[23]: <tf.Tensor: id=57, shape=(4, 6), 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]])>
```

axis=0

```
In [24]: tf.split(x,2,0)
Out[24]: [<tf.Tensor: id=60, shape=(2, 6), dtype=int32, numpy=
          array([[ 0,  1,  2,  3,  4,  5],
                 [ 6,  7,  8,  9, 10, 11]])>,
          <tf.Tensor: id=61, shape=(2, 6), dtype=int32, numpy=
          array([[12, 13, 14, 15, 16, 17],
                 [18, 19, 20, 21, 22, 23]])>]
```

在axis=0的轴上分割，平均分成2份

- 在axis=0轴上分割

```
In [23]: x=tf.range(24)
         x=tf.reshape(x,[4,6])
         x

Out[23]: <tf.Tensor: id=57, shape=(4, 6), 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]])>
```

axis=0

在axis=0的轴上分割
分割成3个张量，axis=0的轴上，长度依次为1,2,1

```
In [25]: tf.split(x,[1,2,1],0)

Out[25]: [<tf.Tensor: id=64, shape=(1, 6), dtype=int32, numpy=a
         rray([[0, 1, 2, 3, 4, 5]])>,
          <tf.Tensor: id=65, shape=(2, 6), dtype=int32, numpy=
          array([[ 6,  7,  8,  9, 10, 11],
                 [12, 13, 14, 15, 16, 17]])>,
          <tf.Tensor: id=66, shape=(1, 6), dtype=int32, numpy=a
         rray([[18, 19, 20, 21, 22, 23]])>]
```

- 在axis=1轴上分割

```
In [23]: x=tf.range(24)
         x=tf.reshape(x,[4,6])
         x
Out[23]: <tf.Tensor: id=57, shape=(4, 6), 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]])>
```

axis=1

图像的分割与拼接，**改变了张量的视图**，张量的**存储顺序并没有改变**。

```
In [26]: tf.split(x,[2,4],1)
Out[26]: [<tf.Tensor: id=69, shape=(4, 2), dtype=int32, numpy=
          array([[ 0,  1],
                 [ 6,  7],
                 [12, 13],
                 [18, 19]])>, <tf.Tensor: id=70, shape=(4, 4),
          dtype=int32, numpy=
          array([[ 2,  3,  4,  5],
                 [ 8,  9, 10, 11],
                 [14, 15, 16, 17],
                 [20, 21, 22, 23]])>]
```

在axis=1的轴上分割
分割成2个张量，axis=1的轴上，长度依次为2,4

堆叠和分解

堆叠张量

- 在合并张量时，**创建一个新的维度**
- 和NumPy中堆叠函数的功能完全一样

tf.stack(values, axis)

要堆叠的多个张量　　指定插入新维度的位置

```
In [27]: x=tf.constant([1, 2, 3])
         y=tf.constant([4, 5, 6])
```

shape：(3,)

```
In [28]: tf.stack((x, y), axis=0)
```

在axis=0的轴上堆叠

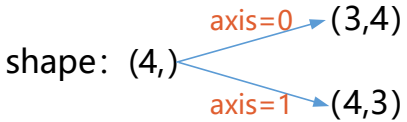
```
Out[28]: <tf.Tensor: id=73, shape=(2, 3), dtype=int32, numpy=
         array([[1, 2, 3],
                [4, 5, 6]])>
```

```
In [29]: tf.stack((x, y), axis=1)
```

在axis=1的轴上堆叠

```
Out[29]: <tf.Tensor: id=74, shape=(3, 2), dtype=int32, numpy=
         array([[1, 4],
                [2, 5],
                [3, 6]])>
```

3个张量堆叠：



分解张量

tf.unstack(values, axis)

- 是张量堆叠的逆运算
- 张量**分解为多个张量**
- 分解后得到的每个张量，和原来的张量相比，维数都少了一维

```
In [30]: c = tf.constant([[1, 2, 3], [4, 5, 6]])
         c
Out[30]: <tf.Tensor: id=75, shape=(2, 3), dtype=int32, numpy=
         array([[1, 2, 3],
                [4, 5, 6]])>

In [31]: tf.unstack(c, axis=0)
Out[31]: [<tf.Tensor: id=76, shape=(3,), dtype=int32, numpy=arr
         ay([1, 2, 3])>,
          <tf.Tensor: id=77, shape=(3,), dtype=int32, numpy=arr
         ay([4, 5, 6])>]

In [32]: tf.unstack(c, axis=1)
Out[32]: [<tf.Tensor: id=78, shape=(2,), dtype=int32, numpy=arr
         ay([1, 4])>,
          <tf.Tensor: id=79, shape=(2,), dtype=int32, numpy=arr
         ay([2, 5])>,
          <tf.Tensor: id=80, shape=(2,), dtype=int32, numpy=arr
         ay([3, 6])>]
```