编 程 学 习 笔 记

TensorFlow

作 者 姓 名： 颜佳

目 录

[1 版本 1](#_Toc100958387)

[1.1 tf.\_\_version\_\_ 1](#_Toc100958388)

[2 基础数据类型 1](#_Toc100958389)

[2.1 归属模块：tf.dtypes 1](#_Toc100958390)

[2.2 归属类：tf.dtypes.DType 1](#_Toc100958391)

[2.2.1 方法 1](#_Toc100958392)

[2.2.2 布尔型 1](#_Toc100958393)

[2.2.3 整形 1](#_Toc100958394)

[2.2.4 浮点型 2](#_Toc100958395)

[2.2.5 复数 2](#_Toc100958396)

[3 张量 2](#_Toc100958397)

[3.1 创建 2](#_Toc100958398)

[3.1.1 自定义张量 2](#_Toc100958399)

[3.1.2 特殊的张量 3](#_Toc100958400)

[3.2 特性 3](#_Toc100958401)

[3.2.1 形状：张量的每个轴的长度（元素数量）。 3](#_Toc100958402)

[3.2.2 秩：张量轴数。标量的秩为 0，向量的秩为 1，矩阵的秩为 2。 3](#_Toc100958403)

[3.2.3 轴或维度：张量的一个特殊维度。 3](#_Toc100958404)

[3.2.4 大小：张量的总项数，即乘积形状向量 4](#_Toc100958405)

[3.3 属性 5](#_Toc100958406)

[3.4 运算 5](#_Toc100958407)

[4 5](#_Toc100958408)

[5 5](#_Toc100958409)

[6 tf.keras 5](#_Toc100958410)

[6.1 activations 5](#_Toc100958411)

[6.1.1 第一段 5](#_Toc100958412)

[6.2 5](#_Toc100958413)

[7 5](#_Toc100958414)

查看版本：tf.\_\_version\_\_

基础数据类型

## 归属模块：tf.dtypes

**<module 'tensorflow.\_api.v2.dtypes'>**

## 归属类：tf.dtypes.DType

**<class 'tensorflow.python.framework.dtypes.DType'>**

### 方法

[as\_dtype(...)](https://tensorflow.google.cn/api_docs/python/tf/dtypes/as_dtype): Converts the given type\_value to a DType.

[cast(...)](https://tensorflow.google.cn/api_docs/python/tf/cast): Casts a tensor to a new type.

[complex(...)](https://tensorflow.google.cn/api_docs/python/tf/dtypes/complex): Converts two real numbers to a complex number.

[saturate\_cast(...)](https://tensorflow.google.cn/api_docs/python/tf/dtypes/saturate_cast): Performs a safe saturating cast of value to dtype.

### 布尔型

#### tf.dtypes.bool

### 整形

#### tf.dtypes.uint8

#### tf.dtypes.uint16

#### tf.dtypes.uint32

#### tf.dtypes.uint64

#### tf.dtypes.int8

#### tf.dtypes.int16

#### tf.dtypes.int32

#### tf.dtypes.int64

### 浮点型

#### tf.dtypes.float16

#### tf.dtypes.float32

#### tf.dtypes.float64

#### tf.dtypes.double

### 复数

#### tf.dtypes.complex64

#### tf.dtypes.complex128

张量——tf.Tensor

张量是具有**统一类型**的**多维数组**。所有张量都是不可变的：永远无法更新张量的内容，只能创建新的张量。

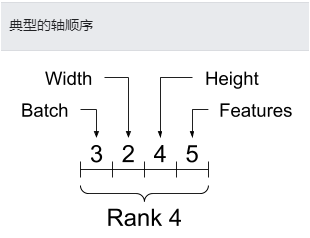
## 特性

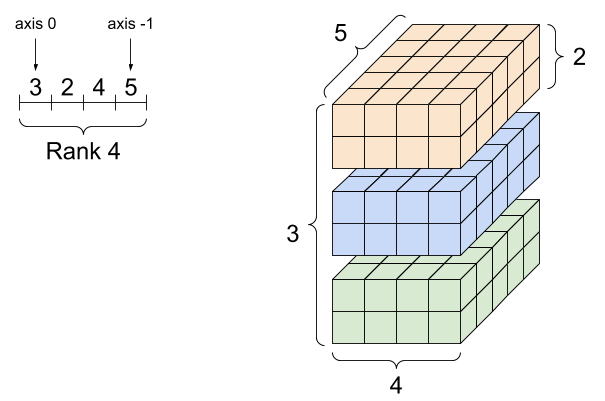
### 形状：张量的每个轴的长度（元素数量）。

### 秩：张量轴数。标量的秩为 0，向量的秩为 1，矩阵的秩为 2。

### 轴或维度：张量的一个特殊维度。

轴一般按照从全局到局部的顺序进行排序：首先是批次轴，随后是空间维度，最后是每个位置的特征。这样，在内存中，特征向量就会位于连续的区域。





### 大小：张量的总项数，即乘积形状向量

## 属性

| **属性** | **结果** |
| --- | --- |
| device | The name of the device on which this tensor will be produced, or None. |
| dtype | The DType of elements in this tensor. |
| graph | The Graph that contains this tensor. |
| name | The string name of this tensor. |
| op | The Operation that produces this tensor as an output. |
| shape | Returns a tf.TensorShape that represents the shape of this tensor. |
| value\_index | The index of this tensor in the outputs of its Operation. |

## 创建

### tf.constant(), tf.convert\_to\_tensor()——通过list, np.array构造

rank\_0\_tensor = tf.constant(4)

rank\_1\_tensor = tf.constant([2.0, 3.0, 4.0])

rank\_2\_tensor = tf.constant([[1, 2],

                             [3, 4],

                             [5, 6]], dtype=tf.float16)

rank\_3\_tensor = tf.constant([[[0, 1, 2, 3, 4],

                              [5, 6, 7, 8, 9]],

                             [[10, 11, 12, 13, 14],

                              [15, 16, 17, 18, 19]],

                             [[20, 21, 22, 23, 24],

                              [25, 26, 27, 28, 29]]])

print(rank\_0\_tensor)

print(rank\_1\_tensor)

print(rank\_2\_tensor)

print(rank\_3\_tensor)

tf.Tensor(4, shape=(), dtype=int32)

tf.Tensor([2. 3. 4.], shape=(3,), dtype=float32)

tf.Tensor(

[[1. 2.]

[3. 4.]

[5. 6.]], shape=(3, 2), dtype=float16)

tf.Tensor(

[[[ 0 1 2 3 4]

[ 5 6 7 8 9]]

[[10 11 12 13 14]

[15 16 17 18 19]]

[[20 21 22 23 24]

[25 26 27 28 29]]], shape=(3, 2, 5), dtype=int32)

### tf.zeros(), tf.zeros\_like(), tf.ones(), tf.ones\_like(), tf.fill()

### tf.random.uniform(), tf.random.normal()

## 切片索引

### [][][]——[, , ,]

### [start:end:step]

### [::]

### [, …]

### tf.gather(), tf.gather\_nd()

### tf.boolean\_mask()

## 维度变换

### tf.reshape()

### tf.transpose()

### tf.expand\_dims()

### tf.squeeze()

## 分割与合并

### tf.concat()

### tf.stack()

### tf.split()

### tf.unstack()

## 数据统计

### tf.norm()

### tf.reduce\_min()

### tf.reduce\_max()

### tf.reduce\_mean()

### tf.reduce\_sum()

### tf.argmax()

### tf.argmin()

### tf.equal()

## 排序

### tf.sort()

### tf.argsort()

## 与numpy数组之间的转换

### np.array(tensor)函数

### tensor.numpy()方法

rank\_1\_tensor = tf.constant([2.0, 3.0, 4.0])

print(np.array(rank\_1\_tensor))

print(rank\_1\_tensor.numpy())

变量——tf.Variable

## 变量与张量区别

### 变量无法重构形状，tf.reshape(my\_variable)返回一个张量而不是一个变量

### 从现有变量创建新变量会复制支持张量。两个变量不能共享同一内存空间。

### tf.Tensor 不可变。张量创建后就不能更改。它有一个值，但没有状态。目前讨论的所有运算也都无状态：tf.matmul 的输出只取决于它的输入。

### tf.Variable 具有内部状态，即它的值。使用变量时，会读取状态。计算相对于变量的梯度是正常操作，但是变量的状态会阻止梯度计算进一步向后移动

# 自动微分

## tf.GradientTape()

    w = tf.Variable(tf.random.normal((3, 2)), dtype=tf.float32, name='w')

    b = tf.Variable(tf.zeros(2, dtype=tf.float32), name='b')

    x = tf.constant([[1., 2., 3.]], dtype=tf.float32)

    with tf.GradientTape(persistent=True) as tape:

        y = x @ w + b

        loss = tf.reduce\_mean(y\*\*2)

    [dl\_dw, dl\_db] = tape.gradient(loss, [w, b])

    print(type(tape.gradient(loss, [w, b])))

    print(dl\_dw)

print(dl\_db)

### tape.gradient(loss, [w, b])——返回list，元素为tf. EagerTensor

### tape.watched\_variables()——返回tuple，元素为tf.ResourceVariable

### 默认情况下，只要调用 GradientTape.gradient 方法，就会释放 GradientTape 保存的资源。要在同一计算中计算多个梯度，请创建一个 persistent=True 的梯度带。

## 自动微分

### 默认只监视tf.Variable的梯度，设置trainable属性为False，则不被监视；

### 默认不监视tf.Tensor的梯度，设置tape.watch(my\_tensor)可以被监视。

### 要停用监视所有 tf.Variables 的默认行为，请在创建梯度带时设置 watch\_accessed\_variables=False

## 获取梯度为None的原因

### 使用张量替换变量

### 在 TensorFlow 之外进行了计算

### 通过整数或字符串获取梯度，整数和字符串不可微分。

### 通过有状态对象获取梯度