张量

- 张量是对矢量和矩阵向潜在的更高维度的泛化
- 对内，TensorFlow 将张量表现为基本数据类型的 n 维数组
- tf.Tensor 有以下属性：
  - 数据类型（例如 float32，int32 或 string）
  - 形状：张量的维数和每个维度的大小
- 主要的特殊张量：
  - tf.Variable：张量的值可变
  - tf.constant
  - tf.placeholder
  - tf.SparseTensor

阶（Rank）

- 也称为："order", "degree", or "ndims."
- 0阶

```
mammal = tf.Variable("Elephant", tf.string)
ignition = tf.Variable(451, tf.int16)
floating = tf.Variable(3.14159265359, tf.float64)
its_complicated = tf.Variable(12.3 - 4.85j, tf.complex64)
# 注意：字符串在 TensorFlow 中被视为单一项，而不是一连串字符。
```

- 1阶，可以传递一个列表作为初始值

```
mystr = tf.Variable(["Hello"], tf.string)
cool_numbers  = tf.Variable([3.14159, 2.71828], tf.float32)
first_primes = tf.Variable([2, 3, 5, 7, 11], tf.int32)
its_very_complicated = tf.Variable([12.3 - 4.85j, 7.5 - 6.23j], tf.complex64)
```

- 更高阶

```
mymat = tf.Variable([[7],[11]], tf.int16)
myxor = tf.Variable([[False, True],[True, False]], tf.bool)
linear_squares = tf.Variable([[4], [9], [16], [25]], tf.int32)
squarish_squares = tf.Variable([ [4, 9], [16, 25] ], tf.int32)
rank_of_squares = tf.rank(squarish_squares)
mymatC = tf.Variable([[7],[11]], tf.int32)
```

- 例如：四阶张量

```
my_image = tf.zeros([10, 299, 299, 3])  # batch x height x width x color
```

- 获取张量的阶：tf.rank

```
r = tf.rank(my_image)
# After the graph runs, r will hold the value 4.
```

- 张量切片，类似python切片

形状（shape）

- 获取张量的形状：
  - my_matrix.shape
  - my_matrix.shape[1] # 获取某个维度上的值

- 改变张量的形状：reshape

```
rank_three_tensor = tf.ones([3, 4, 5])
matrix = tf.reshape(rank_three_tensor, [6, 10])  # 改为（6，10）

matrixB = tf.reshape(matrix, [3, -1])  # 改为（3，20），-1表示自动计算该维度

matrixAlt = tf.reshape(matrixB, [4, 3, -1])  # 改为（4，3， 5）

yet_another = tf.reshape(matrixAlt, [13, 2, -1])  # ERROR，不匹配
```

数据类型

- 一个张量只可能有一种数据类型
- 但可以将任意数据结构序列化为 string 并将其存储在 tf.Tensor 中
- tf.cast 来将 tf.Tensor 从一个数据类型转换到另一种：

```
# Cast a constant integer tensor into floating point.
float_tensor = tf.cast(tf.constant([1, 2, 3]), dtype=tf.float32)
```

- 查看数据类型：Tensor.dtype 属性

评估张量（Evaluating Tensors）

- Tensor.eval方法：
- eval 方法仅在默认 tf.Session 值处于活动状态时才起作用

```
constant = tf.constant([1, 2, 3])
tensor = constant * constant
print tensor.eval()
```

- Tensor.eval 会返回一个与张量内容相同的 Numpy 数组

打印张量

- TensorFlow 提供 tf.Print 指令

```
t = <<some tensorflow operation>>
tf.Print(t, [t])  # This does nothing
t = tf.Print(t, [t])  # Here we are using the value returned by tf.Print
result = t + 1  # Now when result is evaluated the value of `t` will be printed.
```

更新时间：2018年05月09日10:22:22