TensorFlow

目录

安装

跟往常一样，我们用 Conda 来安装 TensorFlow。你也许已经有了一个 TensorFlow 环境，但要确保你安装了所有必要的包。

OS X 或 Linux

运行下列命令来配置开发环境

```
conda create -n tensorflow python=3.5
source activate tensorflow
conda install pandas matplotlib jupyter notebook scipy scikit-learn
conda install -c conda-forge tensorflow
```

Windows

Windows系统，在你的 console 或者 Anaconda shell 界面，运行

```
conda create -n tensorflow python=3.5
activate tensorflow
conda install pandas matplotlib jupyter notebook scipy scikit-learn
conda install -c conda-forge tensorflow
Hello, world!
```

在 Python console 下运行下列代码，检测 TensorFlow 是否正确安装。如果安装正确，Console 会打印出 "Hello, world!"。这可以帮你检测是否·

```
import tensorflow as tf

# Create TensorFlow object called tensor
hello_constant = tf.constant('Hello World!')

with tf.Session() as sess:
    # Run the tf.constant operation in the session
    output = sess.run(hello_constant)
    print(output)
```

Hello, Tensor World!

让我们来分析一下你刚才运行的 Hello World 的代码。代码如下：

```
import tensorflow as tf

# Create TensorFlow object called hello_constant
hello_constant = tf.constant('Hello World!')

with tf.Session() as sess:
    # Run the tf.constant operation in the session
    output = sess.run(hello_constant)
    print(output)
```

Tensor

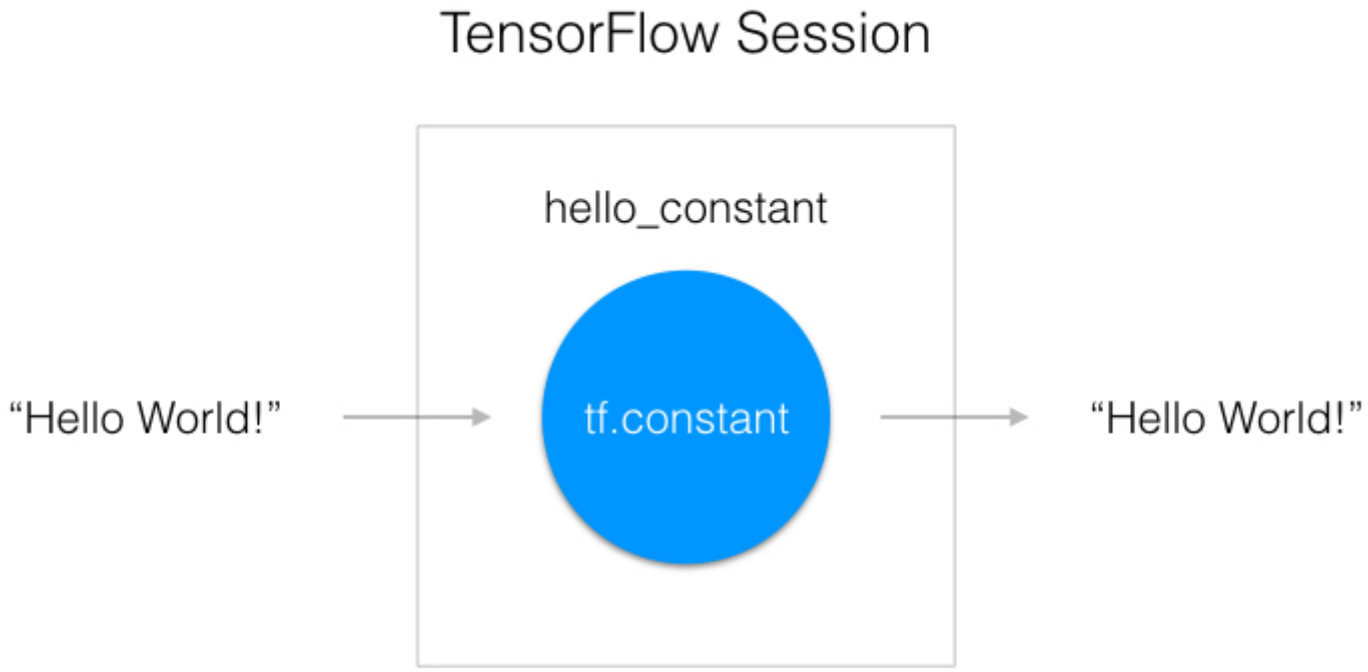
在 TensorFlow 中，数据不是以整数、浮点数或者字符串形式存储的。这些值被封装在一个叫做 tensor 的对象中。在 hello_constant = tf.constant('Hello World!') 代码中，hello_constant 是一个 0 维度的字符串 tensor，tensor 还有很多不同大小：

```
# A is a 0-dimensional int32 tensor
A = tf.constant(1234)
# B is a 1-dimensional int32 tensor
B = tf.constant([123,456,789])
 # C is a 2-dimensional int32 tensor
C = tf.constant([ [123,456,789], [222,333,444] ])
```

tf.constant() 是你在本课中即将使用的多个 TensorFlow 运算之一。tf.constant() 返回的 tensor 是一个常量 tensor，因为这个 tensor 的值不会变。

Session

TensorFlow 的 api 构建在 computational graph 的概念上，它是一种对数学运算过程进行可视化的方法（在 MiniFlow 这节课中学过）。让我们把你刚才运行的 TensorFlow 代码变成一个图：



如上图所示，一个 "TensorFlow Session" 是用来运行图的环境。这个 session 负责分配 GPU(s) 和 / 或 CPU(s)，包括远程计算机的运算。让我们看看如何使用它：

```
with tf.Session() as sess:
    output = sess.run(hello_constant)
```

代码已经从之前的一行中创建了 tensor hello_constant。接下来是在 session 里对 tensor 求值。

这段代码用 tf.Session 创建了一个 sess 的 session 实例。然后 sess.run() 函数对 tensor 求值，并返回结果。

输入

在上一小节中，你向 session 传入一个 tensor 并返回结果。如果你想使用一个非常量（non-constant）该怎么办？这就是 `tf.placeholder()` 和 `feed_dict` 派上用场的时候了。这一节将向你讲解向 TensorFlow 传输数据的基础知识。

tf.placeholder()

很遗憾，你不能把数据集赋值给 x 再将它传给 TensorFlow。因为之后你会想要你的 TensorFlow 模型对不同的数据集采用不同的参数。你需要的是 tf.placeholder()！

数据经过 tf.session.run() 函数得到的值，由 tf.placeholder() 返回成一个 tensor，这样你可以在 session 运行之前，设置输入。

Session 的 feed_dict

```
x = tf.placeholder(tf.string)

with tf.Session() as sess:
    output = sess.run(x, feed_dict={x: 'Hello World'})
```

用 tf.session.run() 里的 feed_dict 参数设置占位 tensor。上面的例子显示 tensor x 被设置成字符串 "Hello, world"。如下所示，也可以用 feed_dict 设置多个 tensor。

```
x = tf.placeholder(tf.string)
y = tf.placeholder(tf.int32)
z = tf.placeholder(tf.float32)

with tf.Session() as sess:
    output = sess.run(x, feed_dict={x: 'Test String', y: 123, z: 45.67})
```

注意：

如果传入 feed_dict 的数据与 tensor 类型不符，就无法被正确处理，你会得到 “ValueError: invalid literal for...”。