




莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

# Tensorflow

 Python基础 ▼

 机器学习 ▼

 数据处理 ▼

 其他 ▼

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说 (神经网络 教学教程 tutorial)



切换到 优酷 视频 ( 如优酷播放出现问题, 请 [点击这里](#) )

[<< 上一个](#)[下一个 >>](#)

---

## scope 命名方法

作者: Morvan 编辑: Morvan

- 学习资料:
  - 不同 scope [对比代码](#)
  - reuse variable [RNN 代码](#)
  - sharing variable [tensorflow 官网介绍](#)

scope 能让你命名变量的时候轻松很多. 同时也会在 reusing variable 代码中常常见到. 所以今天我们会来讨论下 tensorflow 当中的两种定义 scope 的方式. 最后并附加一个 RNN 运用 reuse variable 的例子.

- [tf.name\\_scope\(\)](#)
- [tf.variable\\_scope\(\)](#)
- [RNN应用例子](#)

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

## tf.name\_scope()

在 Tensorflow 当中有两种途径生成变量 variable, 一种是 `tf.get_variable()`, 另一种是 `tf.Variable()`. 如果在 `tf.name_scope()` 的框架下使用这两种方式, 结果会如下.

```
import tensorflow as tf

with tf.name_scope("a_name_scope"):
    initializer = tf.constant_initializer(value=1)
    var1 = tf.get_variable(name='var1', shape=[1], dtype=tf.float32, initializer=initializer)
    var2 = tf.Variable(name='var2', initial_value=[2], dtype=tf.float32)
    var2l = tf.Variable(name='var2', initial_value=[2.1], dtype=tf.float32)
    var22 = tf.Variable(name='var2', initial_value=[2.2], dtype=tf.float32)

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    print(var1.name)      # var1:0
    print(sess.run(var1)) # [ 1.]
```

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

```
print(var21.name)      # a_name_scope/var2_1:0
print(sess.run(var21)) # [ 2.09999999]
print(var22.name)      # a_name_scope/var2_2:0
print(sess.run(var22)) # [ 2.20000005]
```

可以看出使用 `tf.Variable()` 定义的时候, 虽然 `name` 都一样, 但是为了不重复变量名, Tensorflow 输出的变量名并不是一样的. 所以, 本质上 `var2`, `var21`, `var22` 并不是一样的变量. 而另一方面, 使用 `tf.get_variable()` 定义的变量不会被 `tf.name_scope()` 当中的名字所影响.

## tf.variable\_scope()

如果想要达到重复利用变量的效果, 我们就要使用 `tf.variable_scope()`, 并搭配 `tf.get_variable()` 这种方式产生和提取变量. 不像 `tf.Variable()` 每次都会产生新的变量, `tf.get_variable()` 如果遇到了同样名字的变量时, 它会单纯的提取这个同样名字的变量(避免产生新变量). 而在重复使用的时候, 一定要在代码中强调 `scope.reuse_variables()`, 否则系统将会报错, 以为你只是单纯的不小心重复使用到了一个变量.

## 莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

```
initializer = tf.constant_initializer(value=3)

var3 = tf.get_variable(name='var3', shape=[1], dtype=tf.float32, initializer=initializer)
scope.reuse_variables()
var3_reuse = tf.get_variable(name='var3',)
var4 = tf.Variable(name='var4', initial_value=[4], dtype=tf.float32)
var4_reuse = tf.Variable(name='var4', initial_value=[4], dtype=tf.float32)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    print(var3.name)          # a_variable_scope/var3:0
    print(sess.run(var3))     # [ 3.]
    print(var3_reuse.name)    # a_variable_scope/var3:0
    print(sess.run(var3_reuse)) # [ 3.]
    print(var4.name)          # a_variable_scope/var4:0
    print(sess.run(var4))     # [ 4.]
    print(var4_reuse.name)    # a_variable_scope/var4_1:0
    print(sess.run(var4_reuse)) # [ 4.]
```

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

## RNN 应用例子

RNN 例子的代码在[这里](#), 整个 RNN 的结构已经在这里定义好了. 在 training RNN 和 test RNN 的时候, RNN 的 `time_steps` 会有不同的取值, 这将会影响到整个 RNN 的结构, 所以导致在 test 的时候, 不能单纯地使用 training 时建立的那个 RNN. 但是 training RNN 和 test RNN 又必须是有同样的 weights biases 的参数. 所以, 这时, 就是使用 reuse variable 的好时机.

首先定义training 和 test 的不同参数.

```
class TrainConfig:
    batch_size = 20
    time_steps = 20
    input_size = 10
    output_size = 2
    cell_size = 11
    learning_rate = 0.01
```

## 莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

```
train_config = TrainConfig()  
test_config = TestConfig()
```

然后让 `train_rnn` 和 `test_rnn` 在同一个 `tf.variable_scope('rnn')` 之下. 并且定义 `scope.reuse_variables()`, 使我们能把 `train_rnn` 的所有 `weights`, `biases` 参数全部绑定到 `test_rnn` 中. 这样, 不管两者的 `time_steps` 有多不同, 结构有多不同, `train_rnn` `W`, `b` 参数更新成什么样, `test_rnn` 的参数也更新成什么样.

```
with tf.variable_scope('rnn') as scope:  
    sess = tf.Session()  
    train_rnn = RNN(train_config)  
    scope.reuse_variables()  
    test_rnn = RNN(test_config)  
    sess.run(tf.global_variables_initializer())
```

如果你觉得这篇文章或视频对你的学习很有帮助, 请你也分享它, 让它能再次帮助到更多的需要学习的人.


莫烦没有正式的经济来源, 如果你也想支持 莫烦Python 并看到更好的教学内容, 请拉倒屏幕最下方, 赞助他一点点, 作为鼓励他继续开源的动力.



莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

评论内容

使用社交网站账户登录

或使用来必力便捷评论 

邮件

写评论

总评论数 0

按时间正序

还没有评论，快来抢沙发吧！

来必力是？ | 询问

莫烦PYTHON 教程 ▼ 关于我 赞助 大家说

支持 让教学变得更优秀

点我 赞助 莫烦

关注我的动向:

[Youtube频道](#) [优酷频道](#) [Github](#) [微博](#)

**Email:** [morvanzhou@hotmail.com](mailto:morvanzhou@hotmail.com)

© 2016 [morvanzhou.github.io](https://morvanzhou.github.io). All Rights Reserved