

动态图和静态图

目前神经网络框架分为静态图框架和动态图框架，PyTorch 和 TensorFlow、Caffe 等框架最大的区别就是他们拥有不同的计算图表现形式。TensorFlow 使用静态图，这意味着我们先定义计算图，然后不断使用它，而在 PyTorch 中，每次都会重新构建一个新的计算图。通过这次课程，我们会了解静态图和动态图之间的优缺点。

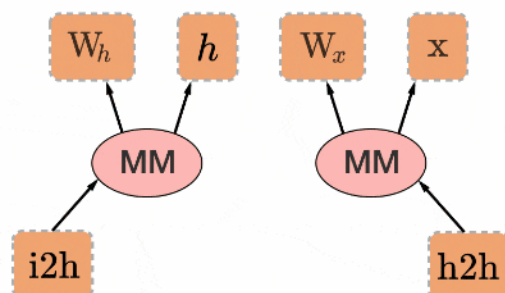
对于使用者来说，两种形式的计算图有着非常大的区别，同时静态图和动态图都有他们各自的优点，比如动态图比较方便debug，使用者能够用任何他们喜欢的方式进行debug，同时非常直观，而静态图是通过先定义后运行的方式，之后再次运行的时候就不再需要重新构建计算图，所以速度会比动态图更快。

A graph is created on the fly

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
```



下面我们比较 while 循环语句在 TensorFlow 和 PyTorch 中的定义

TensorFlow

```
# tensorflow
import tensorflow as tf

first_counter = tf.constant(0)
second_counter = tf.constant(10)
```

```
def cond(first_counter, second_counter, *args):  
    return first_counter < second_counter  
  
def body(first_counter, second_counter):  
    first_counter = tf.add(first_counter, 2)  
    second_counter = tf.add(second_counter, 1)  
    return first_counter, second_counter
```

```
c1, c2 = tf.while_loop(cond, body, [first_counter, second_counter])
```

```
with tf.Session() as sess:  
    counter_1_res, counter_2_res = sess.run([c1, c2])
```

```
print(counter_1_res)  
print(counter_2_res)
```

```
20  
20
```

可以看到 TensorFlow 需要将整个图构建为静态的，换句话说，每次运行的时候图都是一样的，是不能够改变的，所以不能直接使用 Python 的 while 循环语句，需要使用辅助函数 `tf.while_loop` 写成 TensorFlow 内部的形式

这是非常反直觉的，学习成本也是比较高的

下面我们来看看 PyTorch 的动态图机制，这使得我们能够使用 Python 的 while 写循环，非常方便

PyTorch

```
# pytorch  
import torch  
first_counter = torch.Tensor([0])  
second_counter = torch.Tensor([10])
```

```
while (first_counter < second_counter)[0]:  
    first_counter += 2  
    second_counter += 1
```

```
print(first_counter)
print(second_counter)
```

```
20
[torch.FloatTensor of size 1]
```

```
20
[torch.FloatTensor of size 1]
```

可以看到 PyTorch 的写法跟 Python 的写法是完全一致的，没有任何额外的学习成本

上面的例子展示如何使用静态图和动态图构建 while 循环，看起来动态图的方式更加简单且直观，你觉得呢？