CS 20SI: TensorFlow for Deep Learning Lecture 2 summary

3/21/2017

outline

1. TensorBoard
2. Randomly Generated Constants
3. Variables
4. Control Dependencies

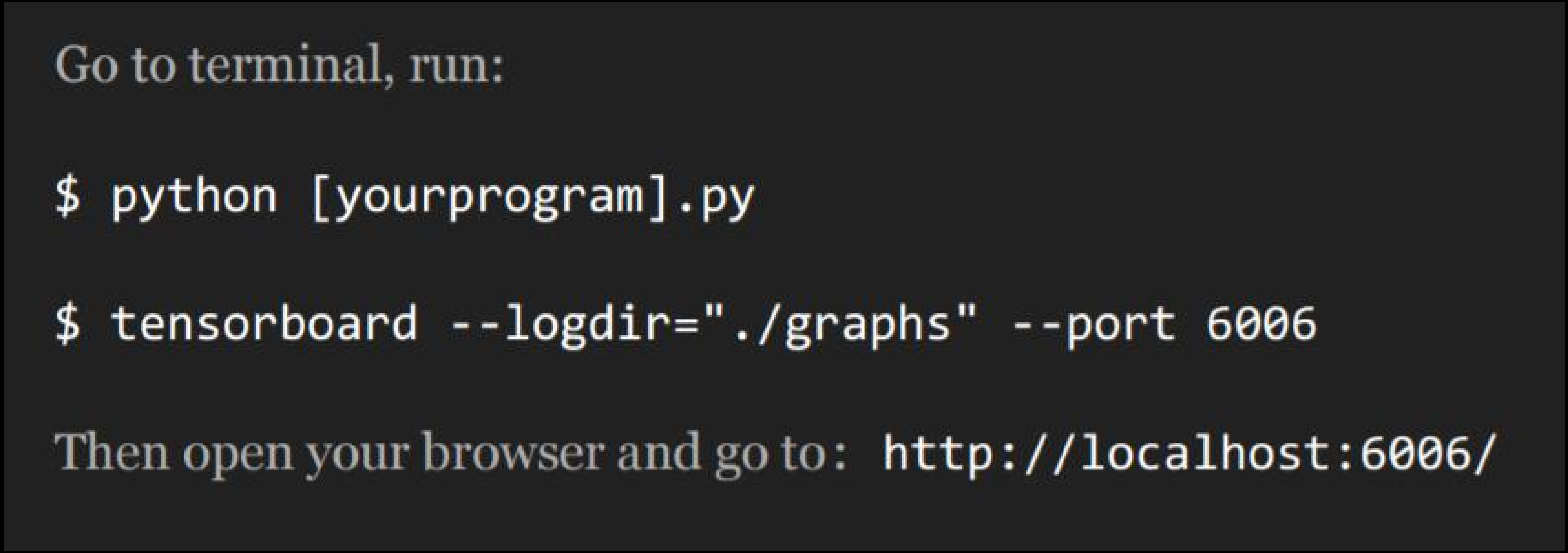
TensorBoard

```
import tensorflow as tf
a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a, b)
with tf.Session() as sess:
        # add this line to use TensorBoard.

        (**) writer = tf.summary.FileWriter('./graphs, sess.graph)

        print sess.run(x)
writer.close()        # close the writer when you're done using it
```

TensorBoard



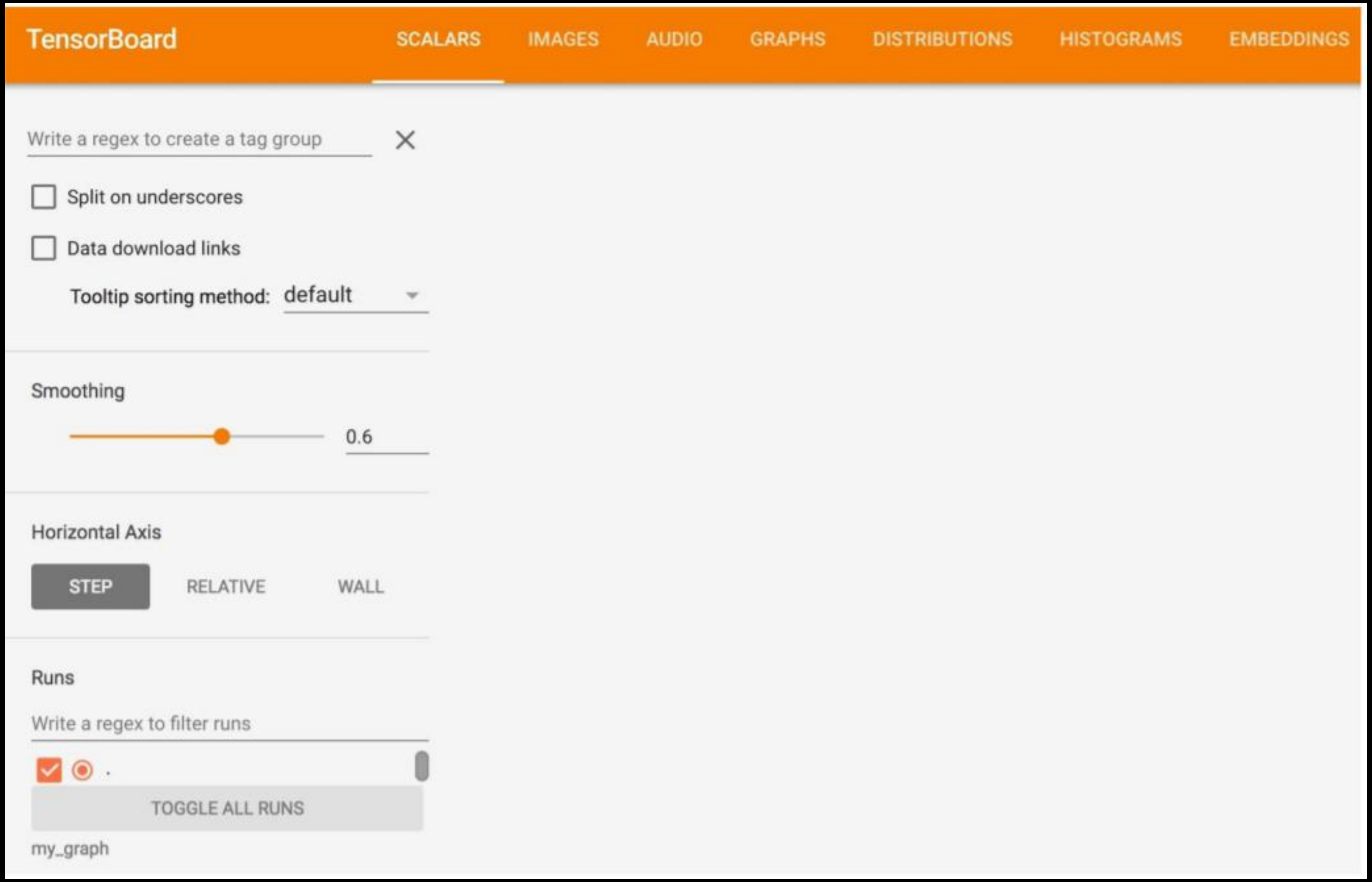
Go to terminal, run:

```
$ python [yourprogram].py

$ tensorboard --logdir="./graphs" --port 6006
```

Then open your browser and go to: `http://localhost:6006/`

TensorBoard



Write a regex to create a tag group

☐ Split on underscores

☐ Data download links

Tooltip sorting method: default

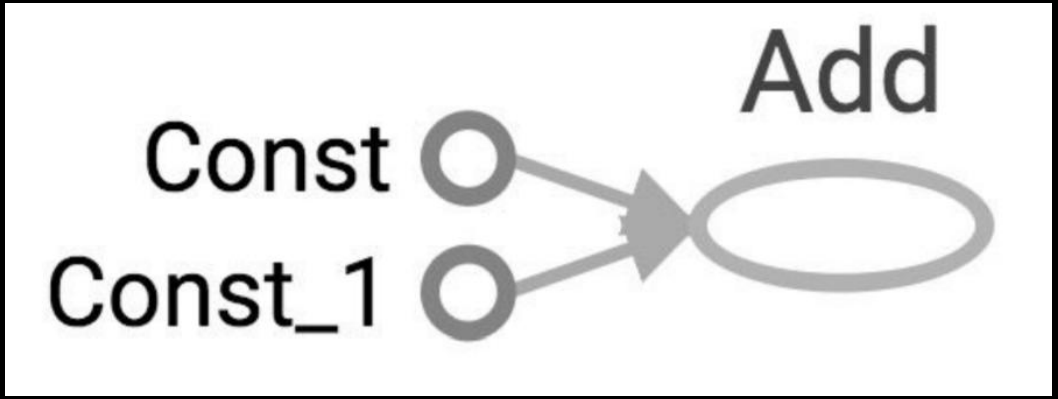
Smoothing 0.6

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

my_graph



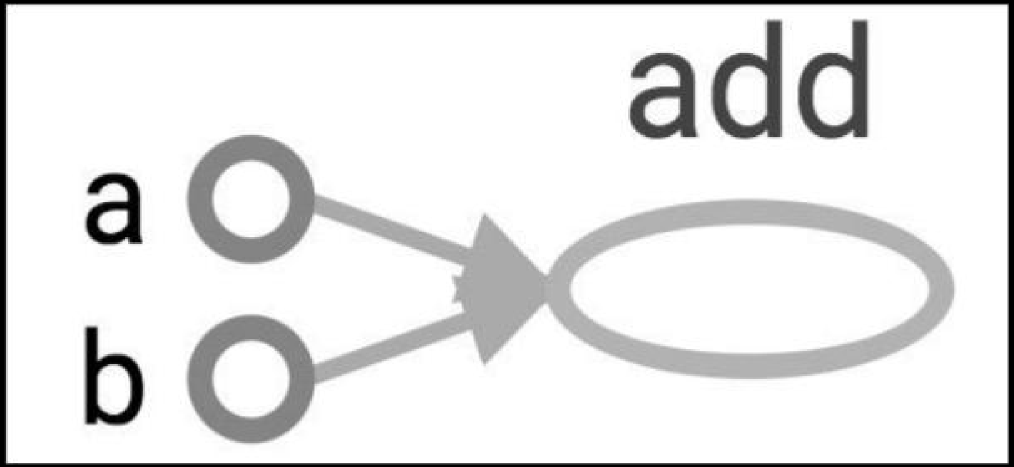
TF automatically names the nodes when you don't explicitly name them. More about this next lecture!

Nodes: operators, variables, and constants
Edges: tensors
Tensors are data.

TensorBoard

```
a = tf.constant(2, name="a")
b = tf.constant(3, name="b")
x = tf.add(a, b, name="add")

writer = tf.summary.FileWriter("./graphs", sess.graph)
with tf.Session() as sess:
print sess.run(x)     # >> 5
```



Randomly Generated Constants

```
tf.set_random_seed(seed)
```

Randomly Generated Constants

To generate different sequences across sessions, set neither graph-level nor op-level seeds:

```
a = tf.random_uniform([1])
b = tf.random_normal([1])
with tf.Session() as sess1:
    print(sess1.run(a))      # generates 'A1'
    print(sess1.run(a))      # generates 'A2'
    print(sess1.run(b))      # generates 'B1'
    print(sess1.run(b))      # generates 'B2'

with tf.Session() as sess2:
    print(sess2.run(a))      # generates 'A3'
    print(sess2.run(a))      # generates 'A4'
    print(sess2.run(b))      # generates 'B3'
    print(sess2.run(b))      # generates 'B4'
```

Randomly Generated Constants

To generate the same repeatable sequence for an op across sessions, set the seed for the op:

```
a = tf.random_uniform([1], seed=1)
b = tf.random_normal([1])
# Repeatedly running this block with the same graph will generate the same
# sequence of values for 'a', but different sequences of values for 'b'.
```

```
with tf.Session() as sess1:
   print(sess1.run(a)) # generates 'A1'
   print(sess1.run(a)) # generates 'A2'
   print(sess1.run(b)) # generates 'B1'
   print(sess1.run(b)) # generates 'B2'
```

```
with tf.Session() as sess2:
   print(sess2.run(a)) # generates 'A1'
   print(sess2.run(a)) # generates 'A2'
   print(sess2.run(b)) # generates 'B3'
   print(sess2.run(b)) # generates 'B4'
```

Randomly Generated Constants

To make the random sequences generated by all ops be repeatable across sessions, set a graph-level seed:

```
tf.set_random_seed(1234)
a = tf.random_uniform([1])
b = tf.random_normal([1])
# Repeatedly running this block with the same graph will generate different
# sequences of 'a' and 'b'.
```

```
with tf.Session() as sess1:
  print(sess1.run(a))  # generates 'A1'
  print(sess1.run(a))  # generates 'A2'
  print(sess1.run(b))  # generates 'B1'
  print(sess1.run(b))  # generates 'B2'
```

```
with tf.Session() as sess2:
  print(sess2.run(a))  # generates 'A1'
  print(sess2.run(a))  # generates 'A2'
  print(sess2.run(b))  # generates 'B1'
  print(sess2.run(b))  # generates 'B2'
```

Variables

tf.Variable holds several ops:

```
(1) x = tf.Variable(...)
(2) x.initializer          # init op
(3) x.value()              # read op
(4) x.assign(...)          # write op
(5) x.assign_add(...) # and more
```

Variables

The easiest way is initializing all variables at once:

```
init = tf.global_variables_initializer()
with tf.Session() as sess:
	sess.run(init)
```

Initialize only a subset of variables:

```
init_ab = tf.variables_initializer([a, b], name="init_ab")
with tf.Session() as sess:
	sess.run(init_ab)
```

Initialize a single variable:

```
W = tf.Variable(tf.zeros([784,10]))
with tf.Session() as sess:
	sess.run(W.initializer)
```

Variables

```
W = tf.Variable(10)
assign_op = W.assign(100)
with tf.Session() as sess:
        sess.run(W.initializer)
        print W.eval()  # >> 10
        sess.run(assign_op)
print W.eval() # >> 100
```

Control Dependencies

```
tf.Graph.control_dependencies(control_inputs)
# defines which ops should be run first
# your graph g have 5 ops: a, b, c, d, e

with g.control_dependencies([a, b, c]):
	# 'd' and 'e' will only run after 'a', 'b', and 'c' have executed.
	d = ...
	e = …
```

谢谢大家！