CS 20SI：TensorFlow for Deep Learning Lecture 1 summary

Email：lightaime@gmail.com
Github: https://github.com/lightaime

3/15/2017

Outline

Graphs and Sessions Concept

Code and Tricks

Graphs and Sessions Concept

TensorFlow separates definition of computations from their execution

1. definite your computations by assembling a graph
2. execute your computations by running a (sub)graph in a session

```
import tensorflow as tf
a = tf.add(3, 5)
print(a)
```

Guess what we would get?

Graphs and Sessions Concept

1

```
a = tf.add(3, 5) # tf.int32 implicitly
print(a)
print(a.op.name)
# Not 8: We just defince a graph. To get the value of a, we nend to create a session and
# run the graph in the sesssion
```

```
Tensor("Add:0", shape=(), dtype=int32)
Add
```

```
b = tf.add(3.0, 5.0) # tf.float32 implicitly
print(b)
```

```
Tensor("Add_1:0", shape=(), dtype=float32)
```

3

```
sess = tf.Session() # Without arguments the session constructor launches the default graph
print(sess.run(a)) # run the graph, now we get the value of a
#print(sess.run(b))
sess.close()
```

```
8
```

```
<bound method Session.close of <tensorflow.python.client.session.Session object at 0x7f
bc54384050>>
```

4

```
# use 'with as' statement to make your code neat
with tf.Session() as sess:
    print(sess.run(a))
```

```
8
```

2

```
# understand print tensor
a_1 = tf.constant(3)
a_2 = tf.constant([1, 2, 3])
a_3 = tf.constant([[1, 2],
                   [3, 4]])
print(a_1)
print(a_1.op.name)
print(a_2)
print(a_2.op.name)
print(a_3)
print(a_3.op.name)
```

```
Tensor("Const_12:0", shape=(), dtype=int32)
Const_12
Tensor("Const_13:0", shape=(3,), dtype=int32)
Const_13
Tensor("Const_14:0", shape=(2, 2), dtype=int32)
Const_14
```

Tricks：

1. x.op.name help your to understand operation
2. with…as statement make your code neat

Graphs and Sessions Concept

1

```
# defining a graph
x = 2
y = 3
op1 = tf.add(x, y)
op2 = tf.mul(x, y)
op3 = tf.pow(op2, op1)
# tf.mul, tf.sub and tf.neg are replaced by tf.multiply, tf.substract and tf.negative
# since my tensorflow version is 1.0.0, it raise a error
# you might replace tf.mul by tf.multiply if you are using 1.0.0
# to get your version by:
# python -c 'import tensorflow as tf; print(tf.__version__)'  # for Python 2
# python3 -c 'import tensorflow as tf; print(tf.__version__)'  # for Python 3
```

2

```
# replace tf.mul by tf.multiply if you are using 1.0.0
graph = tf.Graph()
with graph.as_default():
    x = tf.constant(2) # tf.constant() will build a node
    y = tf.constant(3)
    op1 = tf.add(x, y)
    op2 = tf.multiply(x, y)
    op3 = tf.pow(op2, op1)
print(graph.get_operations()) # Return the list of operations in the graph.
print(graph.version) #Returns a version number that increases as ops are added to the grap
```

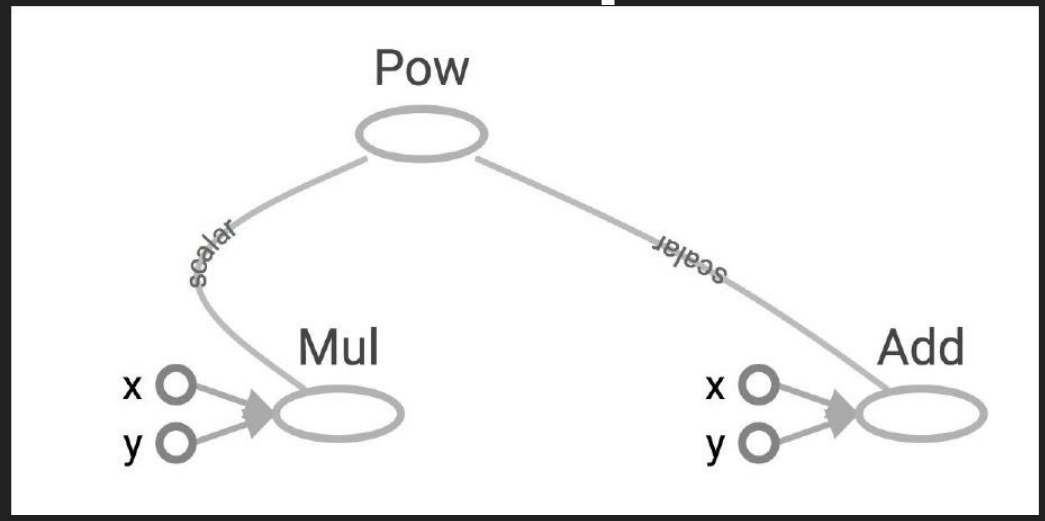
3

```
[<tf.Operation 'Const' type=Const>, <tf.Operation 'Const_1' type=Const>, <tf.Operation
 'Add' type=Add>, <tf.Operation 'Mul' type=Mul>, <tf.Operation 'Pow' type=Pow>]
5
```

```
# run the graph
with tf.Session(graph = graph) as sess:
    op3 = sess.run(op3)
    print(op3) # 6^5 = 7776
```

```
7776
```

Tensorflow will find what make op3 and only compute the necessary operations



class tf.Graph

A TensorFlow computation, represented as a dataflow graph.

A `Graph` contains a set of `Operation` objects, which represent units of computation; and `Tensor` objects, which represent the units of data that flow between operations.

Tricks：

1. Graph.get_operations() return a list of operations in the graph
2. Graph.version return the number of operations

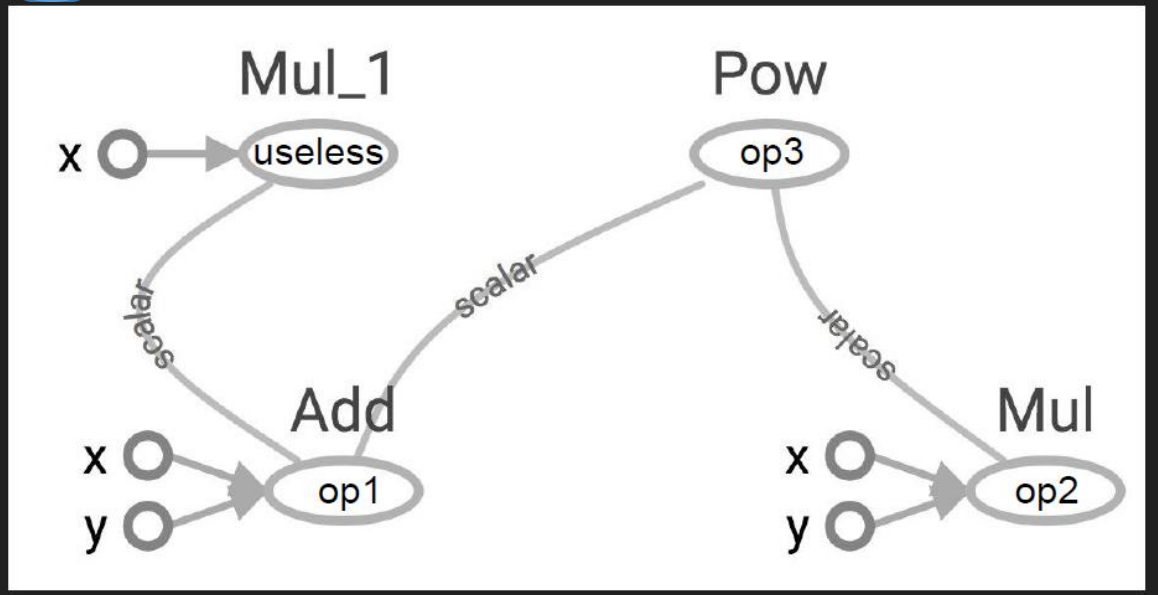
Graphs and Sessions Concept

1

```
graph = tf.Graph()
with graph.as_default():
    x = tf.constant(2)
    y = tf.constant(3)
    op1 = tf.add(x, y)
    op2 = tf.multiply(x, y)
    useless = tf.multiply(x, op1) # this multipy operation wolud not be run in case 1
    op3 = tf.pow(op2, op1)
print(graph.get_operations())
print(graph.version)
```

```
[<tf.Operation 'Const' type=Const>, <tf.Operation 'Const_1' type=Const>, <tf.Operation
 'Add' type=Add>, <tf.Operation 'Mul' type=Mul>, <tf.Operation 'Mul_1' type=Mul>, <tf.O
peration 'Pow' type=Pow>]
6
```

2



Graphs and Sessions Concept

running a subgraph example

```
graph = tf.Graph()
with graph.as_default():
    x = tf.constant(2)
    y = tf.constant(3)
    op1 = tf.add(x, y)
    op2 = tf.multiply(x, y)
    useless = tf.multiply(x, op1) # this multipy operation wolud not be run in case 1
    op3 = tf.pow(op2, op1)
print(graph.get_operations())
print(graph.version)
```

TensorFlow will find what make op3 and only compute the necessary operations

```
[<tf.Operation 'Const' type=Const>, <tf.Operation 'Const_1' type=Const>, <tf.Operation
 'Add' type=Add>, <tf.Operation 'Mul' type=Mul>, <tf.Operation 'Mul_1' type=Mul>, <tf.O
peration 'Pow' type=Pow>]
6
```

```
# case 1
with tf.Session(graph = graph) as sess:
    op3 = sess.run(op3)
    print(op3)
    #print(useless)
```

```
7776
```

1

```
graph = tf.Graph()
with graph.as_default():
    x = tf.constant(2)
    y = tf.constant(3)
    op1 = tf.add(x, y)
    op2 = tf.multiply(x, y)
    useless = tf.multiply(x, op1) # run this operation
    op3 = tf.pow(op2, op1)
print(graph.get_operations())
print(graph.version)
```

```
[<tf.Operation 'Const' type=Const>, <tf.Operation 'Const_1' type=Const>, <tf.Operation
 'Add' type=Add>, <tf.Operation 'Mul' type=Mul>, <tf.Operation 'Mul_1' type=Mul>, <tf.O
peration 'Pow' type=Pow>]
6
```

```
# case 2
with tf.Session(graph = graph) as sess:
    op3, not_useless = sess.run([op3, useless])
    print(op3)
    print(not_useless)
```

```
7776
10
```

2

Graphs and Sessions Concept

1

```
with tf.device('/gpu:2'):
    graph = tf.Graph()
    with graph.as_default():
        a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='a', shape=[2, 3])
        b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='b', shape=[3, 2])
        c = tf.matmul(a, b)
    print(graph.get_operations())
    print(graph.version)
```

```
[<tf.Operation 'a' type=Const>, <tf.Operation 'b' type=Const>, <tf.Operation 'MatMul' t
ype=MatMul>]
3
```

```
with tf.Session(graph=graph, config=tf.ConfigProto(log_device_placement=True)) as sess:
    c = sess.run(c)
    print(c)
```

```
[[ 22.  28.]
 [ 49.  64.]]
```

```
c = []
for d in ['/gpu:2', '/gpu:3']:
  with tf.device(d):
    a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3])
    b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2])
    c.append(tf.matmul(a, b))
with tf.device('/cpu:0'):
  sum = tf.add_n(c)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print sess.run(sum)
```

```
[[  44.   56.]
 [  98.  128.]]
```

2

```
Device mapping:
/job:localhost/replica:0/task:0/gpu:0 -> device: 0, name: GeForce GTX 1080, pci bus id: 0000:02:00.0
/job:localhost/replica:0/task:0/gpu:1 -> device: 1, name: GeForce GTX 1080, pci bus id: 0000:03:00.0
/job:localhost/replica:0/task:0/gpu:2 -> device: 2, name: GeForce GTX 1080, pci bus id: 0000:83:00.0
/job:localhost/replica:0/task:0/gpu:3 -> device: 3, name: GeForce GTX 1080, pci bus id: 0000:84:00.0
I tensorflow/core/common_runtime/direct_session.cc:257] Device mapping:
/job:localhost/replica:0/task:0/gpu:0 -> device: 0, name: GeForce GTX 1080, pci bus id: 0000:02:00.0
/job:localhost/replica:0/task:0/gpu:1 -> device: 1, name: GeForce GTX 1080, pci bus id: 0000:03:00.0
/job:localhost/replica:0/task:0/gpu:2 -> device: 2, name: GeForce GTX 1080, pci bus id: 0000:83:00.0
/job:localhost/replica:0/task:0/gpu:3 -> device: 3, name: GeForce GTX 1080, pci bus id: 0000:84:00.0

MatMul_1: (MatMul): /job:localhost/replica:0/task:0/gpu:3
I tensorflow/core/common_runtime/simple_placer.cc:841] MatMul_1: (MatMul)/job:localhost/replica:0/task:0/gpu:3
MatMul: (MatMul): /job:localhost/replica:0/task:0/gpu:2
I tensorflow/core/common_runtime/simple_placer.cc:841] MatMul: (MatMul)/job:localhost/replica:0/task:0/gpu:2
AddN: (AddN): /job:localhost/replica:0/task:0/cpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] AddN: (AddN)/job:localhost/replica:0/task:0/cpu:0
Add_1: (Add): /job:localhost/replica:0/task:0/gpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] Add_1: (Add)/job:localhost/replica:0/task:0/gpu:0
Add: (Add): /job:localhost/replica:0/task:0/gpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] Add: (Add)/job:localhost/replica:0/task:0/gpu:0
Const_4: (Const): /job:localhost/replica:0/task:0/gpu:3
I tensorflow/core/common_runtime/simple_placer.cc:841] Const_4: (Const)/job:localhost/replica:0/task:0/gpu:3
Const_3: (Const): /job:localhost/replica:0/task:0/gpu:3
I tensorflow/core/common_runtime/simple_placer.cc:841] Const_3: (Const)/job:localhost/replica:0/task:0/gpu:3
Const_2: (Const): /job:localhost/replica:0/task:0/gpu:2
I tensorflow/core/common_runtime/simple_placer.cc:841] Const_2: (Const)/job:localhost/replica:0/task:0/gpu:2
Const_1: (Const): /job:localhost/replica:0/task:0/gpu:2
I tensorflow/core/common_runtime/simple_placer.cc:841] Const_1: (Const)/job:localhost/replica:0/task:0/gpu:2
Const: (Const): /job:localhost/replica:0/task:0/gpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] Const: (Const)/job:localhost/replica:0/task:0/gpu:0
Add_1/y: (Const): /job:localhost/replica:0/task:0/gpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] Add_1/y: (Const)/job:localhost/replica:0/task:0/gpu:0
Add_1/x: (Const): /job:localhost/replica:0/task:0/gpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] Add_1/x: (Const)/job:localhost/replica:0/task:0/gpu:0
Add/y: (Const): /job:localhost/replica:0/task:0/gpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] Add/y: (Const)/job:localhost/replica:0/task:0/gpu:0
Add/x: (Const): /job:localhost/replica:0/task:0/gpu:0
I tensorflow/core/common_runtime/simple_placer.cc:841] Add/x: (Const)/job:localhost/replica:0/task:0/gpu:0
```

Graphs and Sessions Concept

1

```
g = tf.Graph()
with g.as_default():
    a = tf.constant(3)
    b = tf.constant(5)
    x = tf.add(a, b)
print(g.get_operations())
print(g.version)

with tf.Session(graph = g) as sess:
    x = sess.run(x)
    print(x)
```

```
[<tf.Operation 'Const' type=Const>, <tf.Operation 'Const_1' type=Const>, <tf.Operation
 'Add' type=Add>]
3
8
```

2

```
# Do not mix default graph and user created graphs
g = tf.Graph()
a = tf.constant(3) # this op will be placed into default graph
print(a)
with g.as_default():
    b = tf.constant(5)
print(g.get_operations())
print(g.version)
# we only get one operation in g
```

```
Tensor("Const:0", shape=(), dtype=int32)
[<tf.Operation 'Const' type=Const>]
1
```

Why case 2 happens?

More details please go deep into default graph

Thank you