Thomas Whitcomb
thomaswhitcomb@yahoo.com
Jan 25th, 2019

Lesson 2

Code

```
import tensorflow as tf
import sys

class Tensors:
    def __init__(self):
        tf.reset_default_graph()
        self.a = tf.constant(1.12, name="a")
        self.b = tf.constant(2.34, name="b")
        self.c = tf.constant(0.72, name="c")
        self.d = tf.constant(0.81, name="d")
        self.f = tf.constant(19.83, name="f")

    def dag_x(self):
         return tf.add(tf.constant(1.0),
                       tf.add(tf.divide(self.a, self.b),
                              tf.divide(self.c,
(tf.square(self.f)))))

    def dag_s(self):
        return tf.divide(tf.subtract(self.b, self.a),
                         tf.subtract(self.d, self.c))

    def dag_r(self):
        return tf.divide(1.0,
                         tf.add(tf.divide(1.0, self.a),
                                tf.add(tf.divide(1.0,self.b),
                                       tf.add(tf.divide(1.0,self.c),
```

```
tf.divide(1.0,self.d)))))

    def dag_y(self):
        return tf.multiply(tf.multiply(self.a, self.b),
               tf.multiply(tf.divide(1.0, self.c),
               tf.multiply(self.f, tf.divide(self.f, 2))))
    def test(self):
        if run(self.dag_x()).item() != 1.4804635047912598:
            print("Tensors - x is bad: ", run(self.dag_x()).item())
        if run(self.dag_s()).item() != 13.555558204650879:
            print("Tensors - s is bad: ", run(self.dag_s()).item())
        if run(self.dag_r()).item() != 0.2535712718963623:
            print("Tensors - r is bad: ", run(self.dag_r()).item())
        if run(self.dag_y()).item() != 715.676513671875:
            print("Tensors - y s bad: ", run(self.dag_y()).item())


def problem1_setup():
    tensor_x = tf.constant(list(range(100, 110)))
    tensor_y = tf.constant([34, 28, 45, 67, 89, 93, 24, 49, 11, 7])
    return tf.add(tensor_x, tensor_y)

def problem1():
    dag = problem1_setup()
    print("Problem #1: - lazy sum is ", run(dag))

def problem1_eager():
    tf.enable_eager_execution()
    answer = problem1_setup()
    print("Problem #1: - eager sum is ", answer)

def problem2():
    x = tf.constant([[1, 2, 3, 4], [5, 6, 7, 8]])
    t = tf.stack([x, x, x, x])
    print("Problem #2", t.shape)

def problem3():
    x = tf.constant([[1, 2, 3, 4], [5, 6, 7, 8]])
    t = tf.stack([x])
    print("Problem #3", t.shape)

def problem4():
```

```
    x = tf.constant([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
    t = tf.reshape(x, [6, 2])
    print("Problem #4")
    print("\t", t.shape)
    print("\t", run(t))

def run(tensor):
    with tf.Session() as sess:
        return(sess.run(tensor))

def problem5():
    tensors = Tensors()
    tensors.test()
    print("Problem #5")
    print("\tx = ", run(tensors.dag_x()))
    print("\ts = ", run(tensors.dag_s()))
    print("\tr = ", run(tensors.dag_r()))
    print("\ty = ", run(tensors.dag_y()))

def graphit(name, tensor):
    with tf.Session() as sess:
        with tf.summary.FileWriter(name, sess.graph) as writer:
             sess.run([tensor])

def problem6():

    print("Problem #6 - graphs created")
    graphit("graphs/x", Tensors().dag_x())
    graphit("graphs/s", Tensors().dag_s())
    graphit("graphs/r", Tensors().dag_r())
    graphit("graphs/y", Tensors().dag_y())

def is_associative(matrix_a, matrix_b, matrix_c):
    x = tf.multiply(matrix_a, tf.add(matrix_b, matrix_c))
    y = tf.add(tf.multiply(matrix_a, matrix_b), tf.multiply(matrix_a,
matrix_c))
    return (run(x) == run(y)).all()

def is_distributive(matrix_a, matrix_b, matrix_c):
    x = tf.multiply(tf.multiply(matrix_a, matrix_b) , matrix_c)
    y = tf.multiply(matrix_a, tf.multiply(matrix_b, matrix_c))
    return (run(x) == run(y)).all()
```

```
def problem7():
    matrix_a = tf.constant([[4, -2, 1], [6, 8, -5], [7, 9, 10]])
    matrix_b = tf.constant([[6, 9, -4], [7, 5, 3], [-8, 2, 1]])
    matrix_c = tf.constant([[-4, -5, 2], [10, 6, 1], [3, -9, 8]])
    print("Problem #7")
    print("\tSatisfies the associative property",
is_associative(matrix_a, matrix_b, matrix_c))
    print("\tSatisfies the distributive property",
is_distributive(matrix_a, matrix_b, matrix_c))

def main():
    problem1()
    problem2()
    problem3()
    problem4()
    problem5()
    problem6()
    problem7()

if __name__ == "__main__":
    if len(sys.argv) ==  2 and sys.argv[1] == "eager":
        tf.enable_eager_execution()
        problem1_eager()
    else:
        if len(sys.argv) == 1:
            main()
        else:
            print("Usage: lesson2 [eager]")
```

Results

Run > python lesson2.py eager

```
Problem #1: - eager sum is  tf.Tensor([134 129 147 170 193 198 130
156 119 116], shape=(10,), dtype=int32)
```

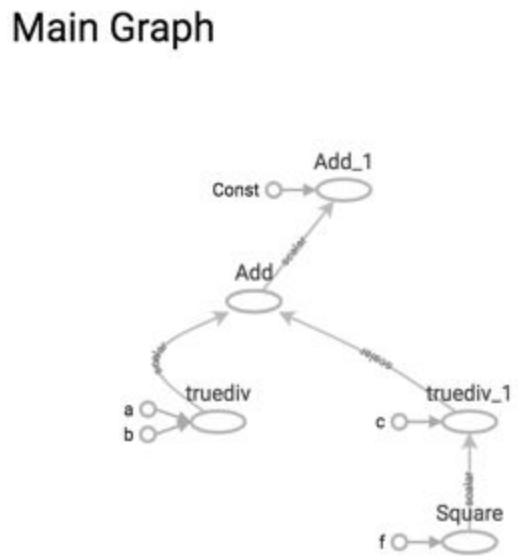
Run > python lesson2.py eager

```
Problem #1: - lazy sum is  [134 129 147 170 193 198 130 156 119 116]
Problem #2 (4, 2, 4)
Problem #3 (1, 2, 4)
Problem #4
```

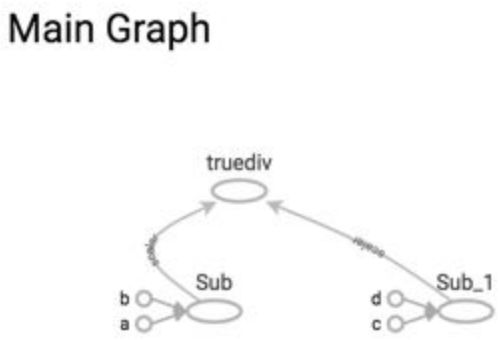
```
(6, 2)
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]]
Problem #5
        x =  1.4804635
        s =  13.555558
        r =  0.25357127
        y =  715.6765

Problem #6 - graphs created
```

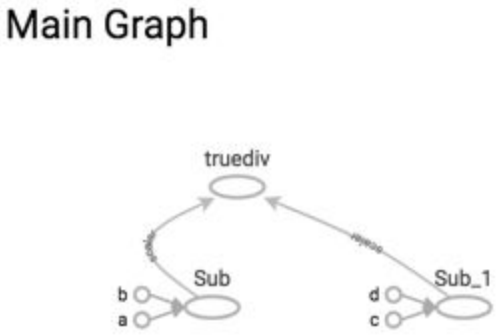
Dag X


Main Graph
Add_1
Const
Add
truediv
truediv_1
a
b
c
Square
f

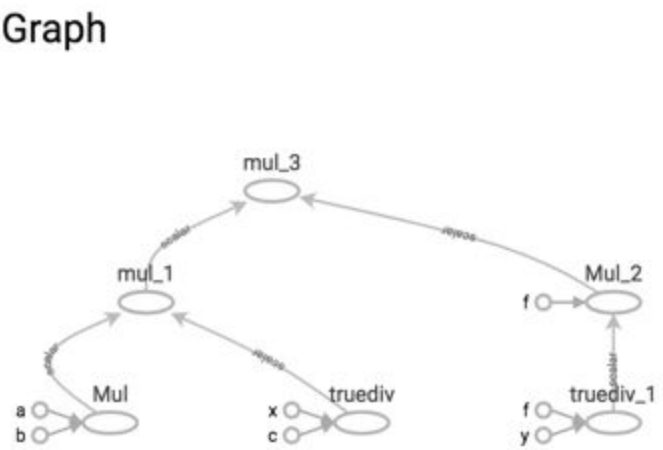
Dag S


Main Graph
truediv
Sub
Sub_1
b
a
d
c

Dag R



Dag Y




```
Problem #7
        Satisfies the associative property True
        Satisfies the distributive property True
```