
Applied Machine Learning with Big Data “EE 6973”



Topic:
TensorFlow

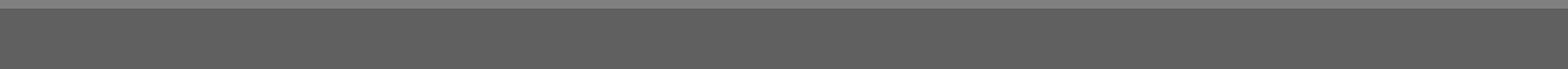
Paul Rad, Ph.D.

Chief Research Officer
UTSA Open Cloud Institute(OCI)
University of Texas at San Antonio

Outline

What does TensorFlow do?

TensorFlow lets us do all our processing in TensorFlow environment, by creating the whole **computational graph first**, then **passing in data and actually computing the result at a later time**

- TensorFlow computations define a computation graph that has no numerical value until evaluated! (Symbolic presentation)
 - Computational Graph: Build the graph, then run it later
 - Benefits include distributed processing, drastic speedups, focus on algorithm not implementation
 - Theano and Torch have done this before, TensorFlow builds on these approaches
 - Tensor can be represented as a multidimensional array of numbers
- 

TensorFlow variables

When you train a model you use variables to hold and update parameters. Variables are in-memory buffers containing tensors – TensorFlow Docs.

Variable tensor has to be initialized

Numpy vs. Tensorflow

```
In [2]: import numpy as np
import tensorflow as tf
```

```
In [5]: a = np.zeros((2,2))
print (a)
```

```
[[ 0.  0.]
 [ 0.  0.]]
```

```
In [6]: ta = tf.zeros((2,2))
```

```
In [7]: print (ta)
```

```
Tensor("zeros:0", shape=(2, 2), dtype=float32)
```

```
In [14]: with tf.Session() as session:
print (ta.eval())
print (session.run(ta))
```

```
[[ 0.  0.]
 [ 0.  0.]]
[[ 0.  0.]
 [ 0.  0.]]
```

Constant

```
In [1]: import tensorflow as tf
```

```
In [2]: input1 = tf.constant(3.0)
input2 = tf.constant(2.0)
```

```
In [3]: input3 = tf.constant(5.0)
```

```
In [4]: intermed = tf.add(input2, input3)
mul = tf.mul(input1, intermed)
```

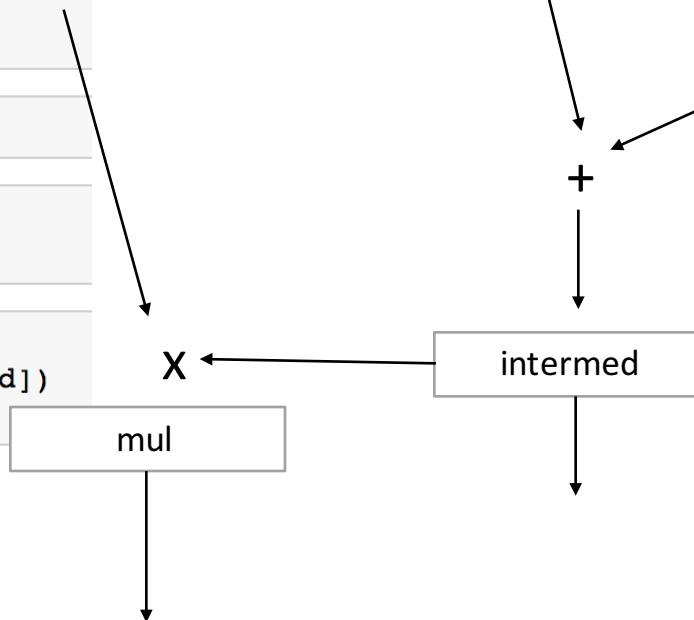
```
In [5]: with tf.Session() as session:
    result = session.run([mul, intermed])
    print (result)

[21.0, 7.0]
```

Input1 = tf.constant (3.0)

Input2 = tf.constant (2)

Input3 = tf.constant (5)



```
In [1]: import tensorflow as tf
```

```
In [2]: x = tf.constant (35, name='x')
```

```
In [3]: y = x + 5
```

```
In [4]: print (y)
```

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

so far, we have the graph, how do we work out its value?

```
In [5]: with tf.Session() as session:  
        print (session.run(y))
```

```
40
```

Exercises

```
In [11]: import tensorflow as tf
```

```
In [12]: x = tf.constant([35, 40, 45], name='x')  
y = tf.Variable(x + 5, name='y')
```

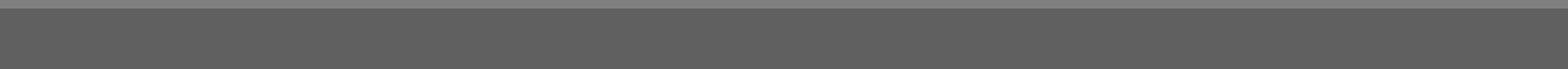
```
In [15]: print (y)
```

```
<tensorflow.python.ops.variables.Variable object at 0x1072ee4e0>
```

```
In [16]: model = tf.initialize_all_variables()
```

```
In [19]: with tf.Session() as session:  
    session.run(model)  
    print(session.run(y))
```

```
[40 45 50]
```



Generate a NumPy array of 10,000 random numbers (called x) and create a Variable storing the equation

$$y = 5x^2 - 3x + 15$$

```
In [1]: import tensorflow as tf
import numpy as np
```

```
In [2]: data = np.random.randint(1000,size=10000)
print (data)
x = tf.constant(data)
print (x)

[525 744 559 ..., 245 471    7]
Tensor("Const:0", shape=(10000,), dtype=int64)
```

```
In [3]: y = 2*(x*x) -3 * x + 15
```

```
In [4]: model = tf.initialize_all_variables()
```

```
In [5]: print (y)

Tensor("add:0", shape=(10000,), dtype=int64)
```

so far, we have the graph, how do we work out its value? Nothing will be computed until we create a tensorflow session

```
In [6]: with tf.Session() as session:
        session.run(model)
        print (session.run(y))

[ 549690 1104855  623300 ..., 119330  442284    92]
```


Placeholders

A placeholder is simply a variable (dummy variable) that we will assign data to at a later date.

It allows us to create our operations and build our computation graph, without needing the data.

In TensorFlow terminology, we then *feed* data into the graph through these placeholders.



Placeholder

```
In [2]: import tensorflow as tf
```

```
In [6]: input1 = tf.placeholder (tf.float32)
```

```
In [7]: input2 = tf.placeholder (tf.float32)
```

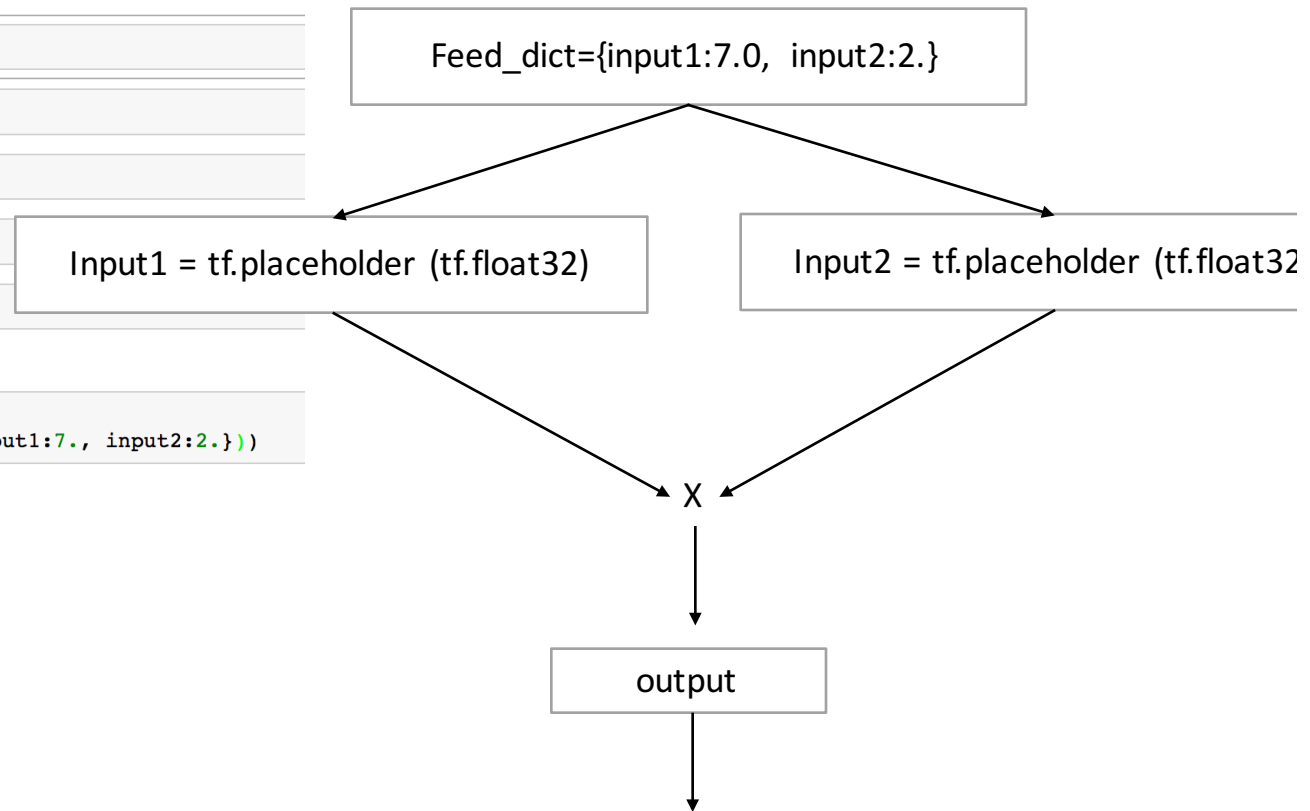
```
In [8]: output = tf.mul(input1, input2)
```

```
In [9]: print (output)
```

```
Tensor("Mul:0", dtype=float32)
```

```
In [13]: with tf.Session() as session:  
         print (session.run(output, feed_dict={input1:7., input2:2.}))
```

```
14.0
```



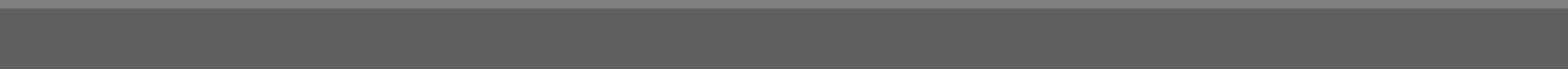
```
In [2]: import tensorflow as tf
```

```
In [3]: x = tf.placeholder("float", 3)
```

```
In [4]: y = x * 2
```

```
In [5]: with tf.Session() as session:  
        result = session.run(y, feed_dict={x: [1, 2, 3]})  
        print(result)
```

```
[ 2.  4.  6.]
```



Placeholders do not need to be statically sized.

```
In [5]: import tensorflow as tf
```

```
In [6]: x = tf.placeholder("float", [None, 3])
```

```
In [7]: y = x * 2
```

```
In [8]: with tf.Session() as session:
          x_data = [[1, 2, 3],
                    [4, 5, 6],]
          result = session.run(y, feed_dict={x: x_data})
          print(result)
```

```
[[ 2.  4.  6.]
 [ 8. 10. 12.]]
```

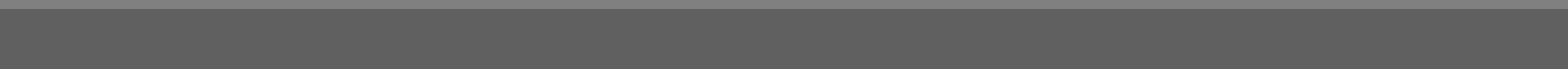
Interactive Sessions

TensorFlow allows us to create a *graph* of operations and variables.

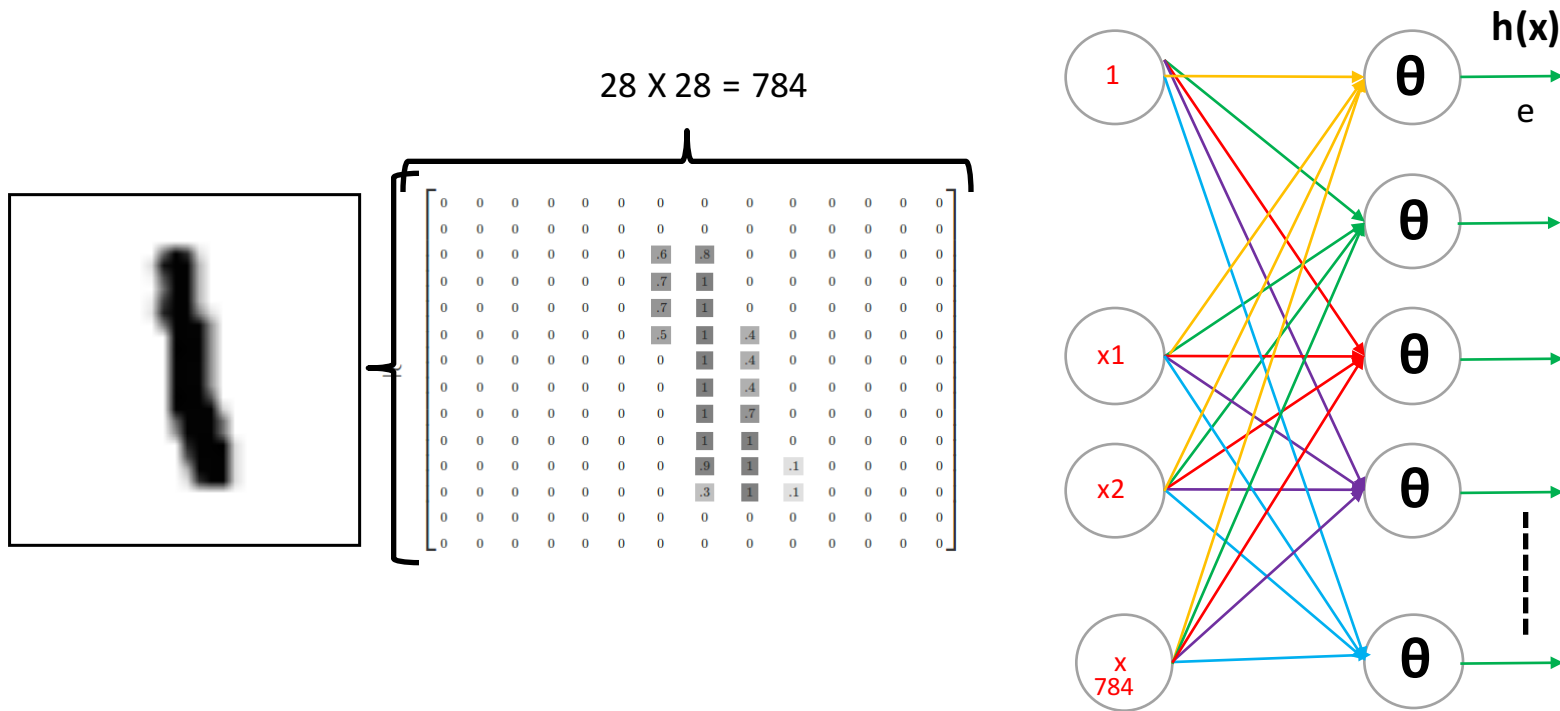
These variables are called Tensors, and represent data, whether that is a single number, a string, a matrix, or something else.

Tensors are combined through operations, and this whole process is modelled in a graph.

One major change is the use of an InteractiveSession, which allows us to run variables without needing to constantly refer to the session object (less typing!)



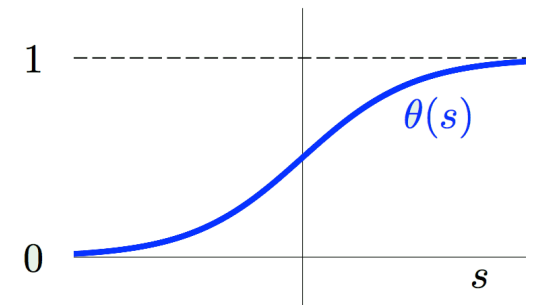
MNIST Number Detection Problem



Formula for likelihood

$$P(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1; \\ 1 - h(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

Substitute $h(\mathbf{x}) = \theta(\mathbf{w}^\top \mathbf{x})$, noting $\theta(-s) = 1 - \theta(s)$



$$P(y \mid \mathbf{x}) = \theta(y \mathbf{w}^\top \mathbf{x})$$

Likelihood of $\mathcal{D} = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ is

$$\prod_{n=1}^N P(y_n \mid \mathbf{x}_n) = \prod_{n=1}^N \theta(y_n \mathbf{w}^\top \mathbf{x}_n)$$

Maximizing the likelihood

Minimize
$$-\frac{1}{N} \ln \left(\prod_{n=1}^N \theta(y_n \mathbf{w}^\top \mathbf{x}_n) \right)$$

$$= \frac{1}{N} \sum_{n=1}^N \ln \left(\frac{1}{\theta(y_n \mathbf{w}^\top \mathbf{x}_n)} \right)$$

$$\left[\theta(s) = \frac{1}{1 + e^{-s}} \right]$$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \underbrace{\ln \left(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n} \right)}_{e(h(\mathbf{x}_n), y_n)}$$

“cross-entropy” error