TensorFlow

Tensorflow

- We are very used to procedural programming
  - Take some value, transform it in some way to obtain a new value, …
- Tensorflow breaks this model:
  - Although our code looks procedural, no computation is actually being done
  - Instead, we are defining a dataflow graph that implements the operations

TensorFlow Operations

- Node in the dataflow graph
- In general, perform some service or computation
- In many cases, they produce a result
  - Can have any number of channels (0, 1, …). But: in most cases, there are zero channels or one channel as output
- Results are edges in the graph

Operations and Tensors

```
x = tf.constant(ins_training, dtype=tf.float32, name = "x")
```

- ins_training: numpy array in this case
- name: name of the node

This line creates:

- Operation node
- Output tensor, which is returned and stored in variable x

Tensors

Objects that:

- Contain information about how to evaluate them (i.e. a reference to the associated node)
- Have a value (a mathematical tensor), which can be obtained using the eval() method

Variable Nodes

```
w = tf.Variable(tf.random_uniform([n,1], -1.0,
      1.0), name = "w")
```

- Creates a “variable” type node (for storage of a value)
- Creates the tensor that evaluates to the value of the variable

```
a = w.eval()
```

- Evaluate the tensor
- In this case, a is a numpy array

General Operation Nodes

```
error = y_pred - y
```

- Create a general operation node (a name will be assigned)
- Creates an output tensor, which is assigned to variable error

- The TensorFlow class provides a set of functions that will perform various kinds of mathematical operations (and they can be named):

```
error = tf.subtract(y_pred, y, name="error")
```

Placeholders

```
n = number of samples
x = tf.placeholder(dtype=tf.float32,
                   shape=(None, n), name="x")
```

- Create a node that has a (partially) defined shape and whose value will be defined later
- x is the corresponding Tensor

Placeholders

- All evaluation / running involves touching a subset of the graph
- If this subgraph involves a placeholder, then its value must be defined at time of evaluation / running

```
feed_dict_validation = {x: ins_validation,
                        y: outs_validation}
f = fvaf.eval(feed_dict = feed_dict_validation)
```

Not all Operation Nodes Produce a Tensor

```
training_op =
    tf.assign(w, w - alpha * gradients)
```

- Return value is a node (not a Tensor!)
- Nodes with no output can be “run” but not eval’d:

```
sess.run(training_op)
```

Graph Evaluation

Evaluating or running a node is generally recursive:

- Evaluation in the graph will stop at a variable, constant and placeholder nodes
- But: general operations will recursively evaluate their input Tensors before performing their operation
- This means that separate calls to eval() or run() will cause reevaluation …

Graph Evaluation

Evaluating multiple variables:

```
[fvaf_training, mse_training] =
    sess.run([fvaf, mse], feed_dict=feed_dict)
```