Up and Running with TensorFlow

Kyubyong Park

kyubyong.park@kakaobrain.com
www.github.com/kyubyong

- Note: This file is for the hands-on session *Up and Running with TensorFlow*
  in the first Deep Learning Conference held in Seoul on Feb. 17, 2017.
  Most of the slides were adapted from the TensorFlow official web site:
  https://www.tensorflow.org/get_started/basic_usage.
  Explanations on each slide will be paired with brief demonstrations and exercises.
  All materials are available at my github repository: https://github.com/Kyubyong/up_and_running_with_tensorflow

Deep Learning Hurdles

- Theories (Back-prop, activation, normalization, …)
- Speed of progression
- Programming skills (Python, NumPy, TensorFlow, Theano, …)
- Math (statistics, probabilities, linear algebra, …)
- English (paper reading, lecture listening, …)
- Money (GPUs, decent computer environment, …)

Content

- Graph, Ops, Tensors
- Handling Arrays and Tensors
- Variables, TensorFlow Fundamentals
- Simple Linear Regression with TensorFlow
- Regression using Neural networks

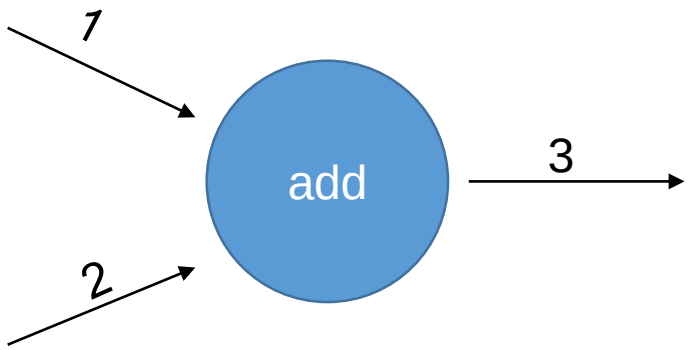
- TensorFlow is a programming system in which you represent computations as graphs.
- Nodes in the graph are called *Ops* (short for operations).
- An op takes zero or more Tensors, performs some computation, and produces zero or more Tensors.
- In TensorFlow terminology, a Tensor is a typed multi-dimensional array. For example, you can represent a mini-batch of images as a 4-D array of floating point numbers with dimensions [batch, height, width, channels].

- A TensorFlow graph is a *description* of computations.
- To compute anything, a graph MUST be launched in a Session.
- A Session places the graph ops onto Devices, such as CPUs or GPUs, and provides methods to execute them.
- These methods return tensors produced by ops as numpy ndarray objects in Python.

- TensorFlow programs are usually structured into a construction phase, that assembles a graph, and an execution phase that uses a session to execute ops in the graph.

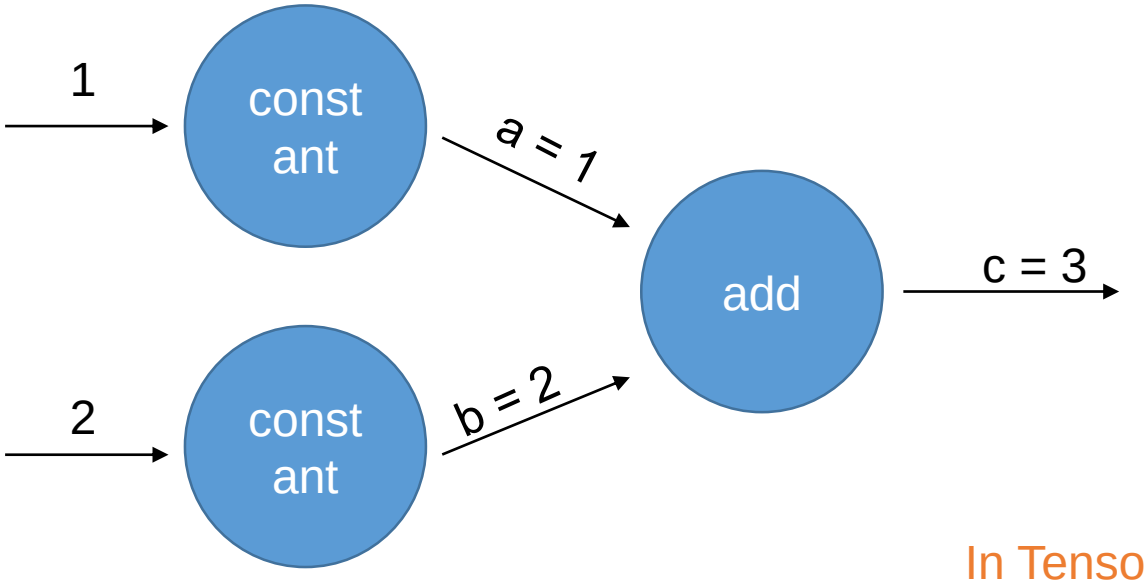
- To fetch the outputs of Ops, execute the graph with a run() call on the Session object and pass in the tensors to retrieve.

Ops and Tensors




```
>> def add(a, b):
…     return a + b
>> print(add(1, 2))
3
```

In Python




```
>> import tensorflow as tf
# Build a graph
>> a = tf.constant(1)
>> b = tf.constant(2)
>> c = tf.add(a, b)
# Launch the graph
>> with tf.Session() as sess:
…       print(sess.run(c))
3
```

In TensorFlow

Challenge

- Open and challenge `**Ch0. Graph.ipynb**`.

- TensorFlow programs use a tensor data structure to represent all data -- only tensors are passed between operations in the computation graph.
- You can think of a TensorFlow tensor as an n-dimensional array or list. A tensor has a type, a rank, and a shape.
- Tensor rank or n-dimension is the number of didmensions of the tensor.
- Tensor shape is the size of every dimension.

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

- Tensors have a data type.

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.
DT_UINT8	tf.uint8	8 bits unsigned integer.
DT_UINT16	tf.uint16	16 bits unsigned integer.
DT_STRING	tf.string	Variable length byte arrays. Each element of a Tensor is a byte array.
DT_BOOL	tf.bool	Boolean.
DT_COMPLEX64	tf.complex64	Complex number made of two 32 bits floating points: real and imaginary parts.
DT_COMPLEX128	tf.complex128	Complex number made of two 64 bits floating points: real and imaginary parts.
DT_QINT8	tf.qint8	8 bits signed integer used in quantized Ops.
DT_QINT32	tf.qint32	32 bits signed integer used in quantized Ops.
DT_QUINT8	tf.quint8	8 bits unsigned integer used in quantized Ops.

```
>> import tensorflow as tf
>>
>> x = tf.zeros([3, 5])
# Let's peek at x
>> print(x)
Tensor("zeros:0", shape=(3, 5), dtype=float32)

# Get shape
>> print(x.get_shape().as_list())
[3, 5] # Python list
>> shp = tf.shape(x) # Tensor

# Get rank
>> print(x.get_shape().ndims)
2 # Python scalar
>> rnk = tf.rank(x) # Tensor

# Get data type
>> print(x.dtype)
tf.float32
```

NumPy

- is the fundamental package for scientific computing with Python. (www.numpy.org)
- has very similar APIs with TensorFlow's.
  e.g. np.reshape() == tf.reshape()

Arrays Are

- typically used when feeding data in.
- returned when fetching the value of tensors.
- extensively used particulary in preprocessing and postprocessing.

- Open and check `**Demo0. Create Arrays and Tensors.ipynb**`.
- Download files from my another github repository https://github.com/Kyubyong/tensorflow-exercises.
- Challenge questions 1-12 in `**Constants_Sequences_and_Random_Values.ipynb**`
- Open and check `**Demo1. Slicing and Indexing.ipynb**`.
- Open and check `**Demo2. Math.ipynb**`.
- Challenge questions 1, 3, and 13 in `**Math Part I.ipynb**`.
- Challenge questions 7 and 14 in `**Math Part II.ipynb**`.
- Challenge question 9 in `**Math Part III.ipynb**`.
- Challenge questions 5, 10, and 13 in `**Tensor Transformation.ipynb**`.

- A Variable is constructed by tf.Variable(<initial-value>).

```
>> a = tf.Variable(3, dtype=tf.int32)
>> b = tf.Variable(tf.random_normal([3, 2]))
```

- A Variable is updated by tf.assign(<current-value>, <new-value>).

```
>> update_op = tf.assign(a, a + 2)
```

- Variables must be initialized before you run Ops that use their value.

```
>> a = tf.Variable(3, dtype=tf.int32)
>> init = a.initializer
>> sess = tf.Session()
>> sess.run(init)
>> print(sess.run(a))
```

- You typically represent the parameter of a statistical model as a set of Variables
- During training you update parameters or weights represented as a Variable by running a training graph repeatedly.

```
>> … Definition of Graph …
>> update_op = …
>> sess = tf.Session()
>> for step in range(100):
…     sess.run(update_op)
```

- You can save and restore variables by tf.train.Saver.save and tf.train.Saver.restore.

```
>> a = tf.Variable(3, dtype=tf.int32)
>> b = tf.Variable(tf.random_normal([3, 2]))
>> saver = tf.train.Saver([a, b])
>> sess = tf.Session()
>> saver.save(sess, ‘filename’)
>> saver.restore(sess, ‘filename’)
```

Challenge

- Open and challenge `**Ch1. Variables.ipynb**`.

- A placeholder exists solely to serve as the target of feeds.

```
>> x_pl0 = tf.placeholder(tf.float32) # Any shape
>> x_pl1 = tf.placeholder(tf.float32, []) # 0-D
>> x_pl2 = tf.placeholder(tf.int32, [None]) # 1-D
>> x_pl3 = tf.placeholder(tf.float64, [None, 3]) # 2-D
```

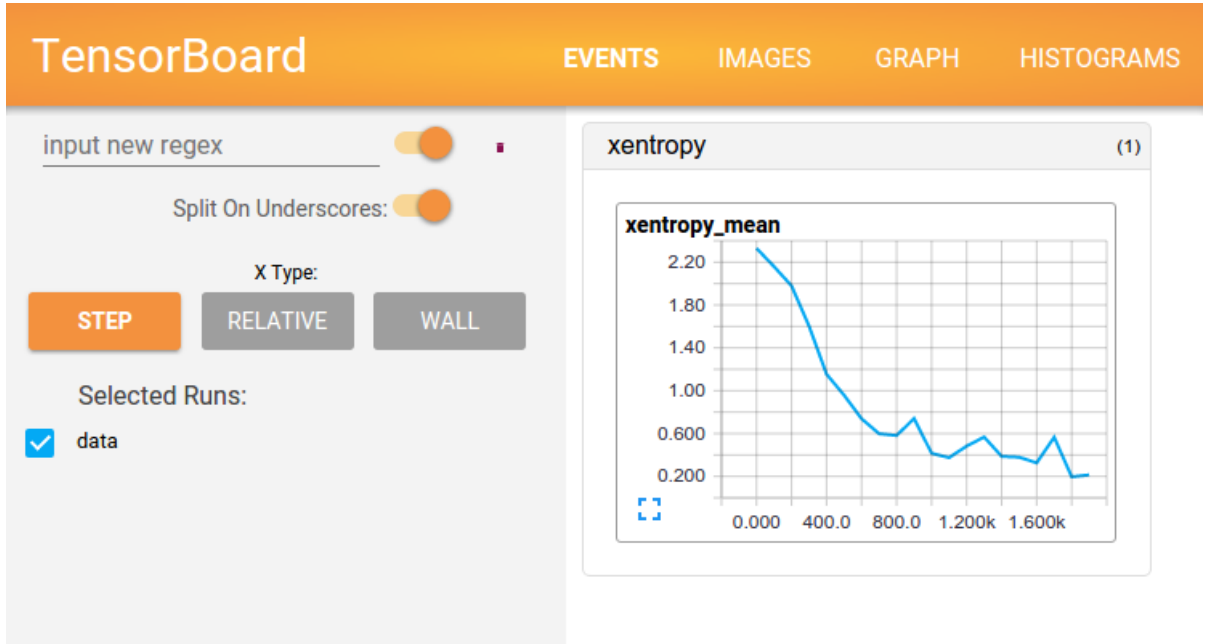
- A placeholder is not initialized and contains no data.
- A placeholder generates an error if it is executed without a feed.

```
>> sess = tf.Session()
>> sess.run(x_pl0)
You must feed a value for placeholder
```

- TensorFlow also provides a feed mechanism for patching a tensor directly into any operation in the graph.
- A feed temporarily replaces the output of an operation with a tensor value. You supply feed data as an argument to a run() call. The feed is only used for the run call to which it is passed. The most common use case involves designating specific operations to be "feed" operations by using tf.placeholder() to create them:

You can use TensorBoard to

- visualize your TensorFlow graph,
- plot quantitative metrics about the execution of your graph
- show additional data like images that pass through it.



- Collect Variables or (0-D)Tensors you want to record with tf.summary.scalar(name, tensor).
- Combine all summary Ops with tf.summary.merge_all().

```
>> summaries = tf.summary.merge_all()
```

- Create a FileWriter with tf.summary.FileWriter(logdir, graph). Then an event file will be generated.

```
>> writer = tf.summary.FileWriter('asset', tf.get_default_graph())
```

- Fetch the content of merged summaries.

```
>> summary_content = sess.run(summaries)
```

- Write the summary content to TensorBoard.

```
>> writer.add_summary(summary_content)
```

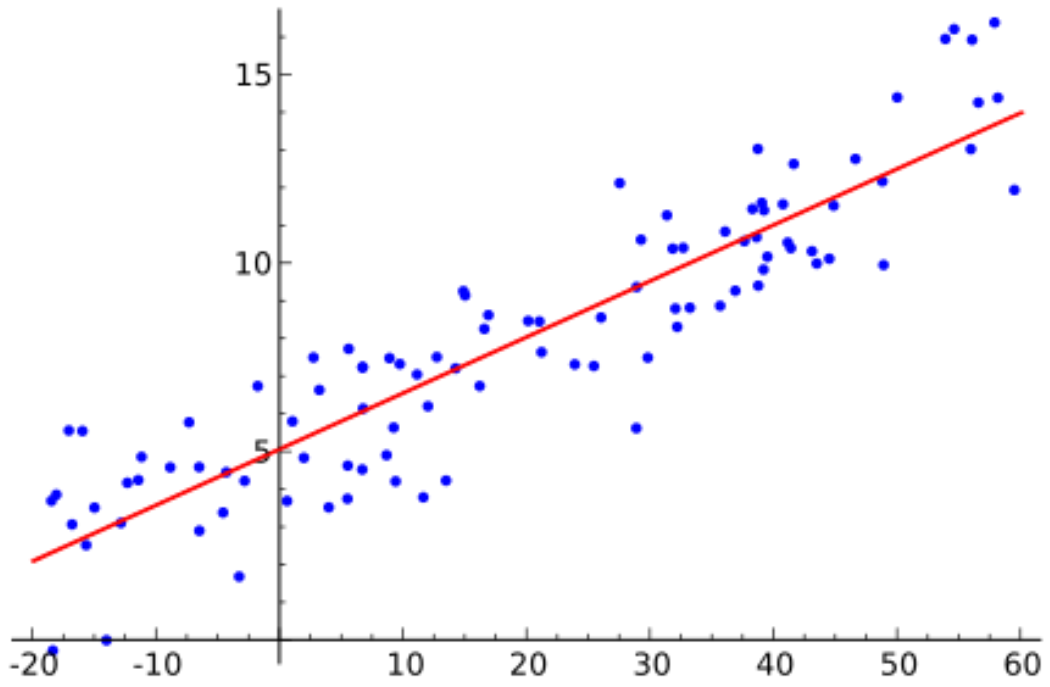
- Launch the TensorBoard.

```
>> tensorboard -logdir=asset
```

Challenge

- Open and challenge `**Ch2. Placeholder.ipynb**`.

- In statistics, linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. (https://en.wikipedia.org/wiki/Linear_regression)



Challenge

- Open and check `**Demo3. Simple Linear Regression.ipynb**`.
- Open and challenge `**Ch3. Linear Regression.ipynb**`.

- Open and check `**Demo4. Regression with Neural Networks**`.

Thank you.

+ We’re hiring.