TensorFlow

Prof. Gheith Abandah

Reference: *Hands-On Machine Learning with Scikit-Learn and TensorFlow* by Aurélien Géron (O'Reilly). Copyright 2017 Aurélien Géron, 978-1-491-96229-9.

Introduction

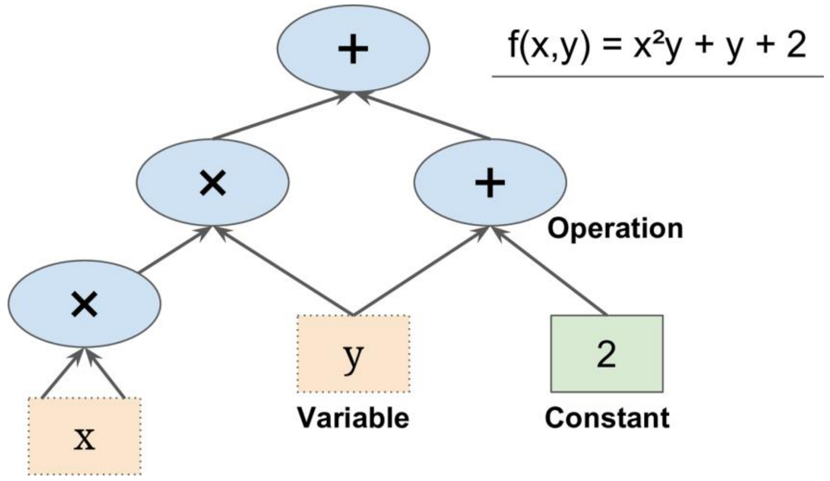
- YouTube Video: *Deep Learning with TensorFlow - Introduction to TensorFlow* from Cognitive Class

https://youtu.be/MotG3XI2qSs

Outline

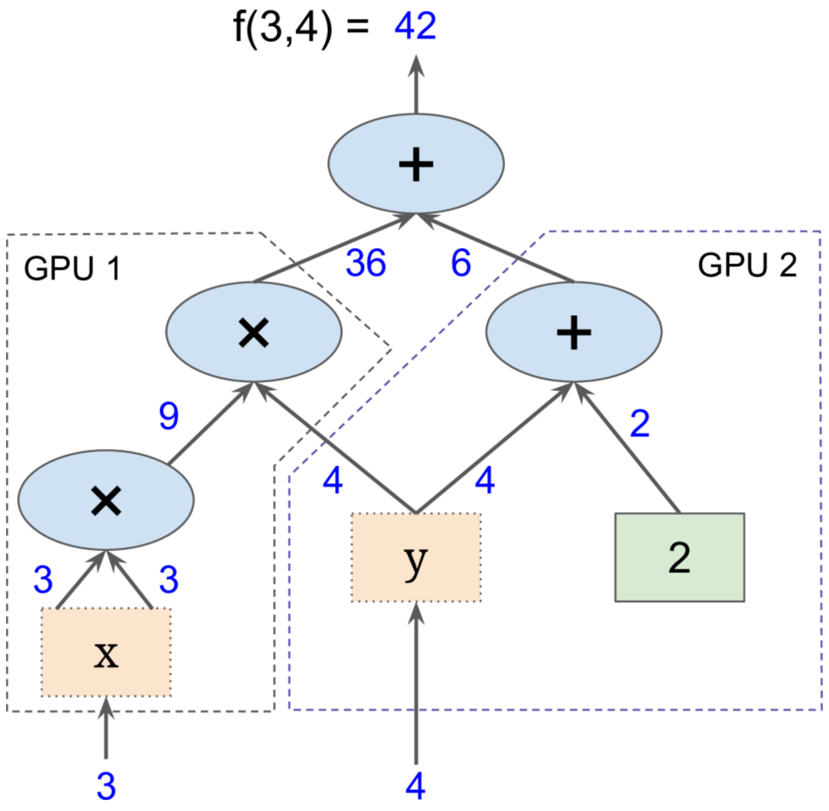
1. Introduction

- *TensorFlow* is a powerful open source, well supported, software library for numerical computation, particularly for large-scale Machine Learning.
- Its basic principle is simple:
    - you first define in Python a graph of computations to perform,
    - and then TensorFlow takes that graph and runs it efficiently using optimized C++ code.



1. Introduction

- It is possible to break up the graph into several chunks and run them in parallel across multiple CPUs or GPUs.
- TensorFlow also supports distributed computing, so you can train colossal neural networks on humongous training sets in a reasonable amount of time by splitting the computations across hundreds of servers.



1. Introduction

- Runs on Windows, Linux, and macOS, iOS, and Android.
- It provides a very simple Python APIs:
  - Keras (the only and the official high-level API in Version 2.0)
  - TF.Learn (tensorflow.contrib.learn), compatible with Scikit-Learn
  - TF-slim (tensorflow.contrib.slim)
  - Pretty Tensor
- Its main Python API offers much more flexibility (at the cost of higher complexity) to create all sorts of computations, including various neural network architectures.
- Includes a visualization tool called *TensorBoard*

2. Installation

- On a cmd window, execute:

```
pip3 install --upgrade tensorflow
```

- To verify, enter and run the following in Python:

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

- If the system outputs the following, then you are ready to begin writing TensorFlow programs:

```
Hello, TensorFlow!
```

3. Example

- The following code creates the graph represented in Slide 4.

```
import tensorflow as tf

x = tf.Variable(3, name="x")
y = tf.Variable(4, name="y")
f = x*x*y + y + 2
```

- Create session, initialize vars, evaluate, and close.

```
>>> sess = tf.Session()
>>> sess.run(x.initializer)
>>> sess.run(y.initializer)
>>> result = sess.run(f)
>>> print(result)
42
>>> sess.close()
```

3. Example

- A better way in a Python program.

```
with tf.Session() as sess:
    x.initializer.run()
    y.initializer.run()
    result = f.eval()
```

- Alternatively, you can create and run an initialization node for all variables.

```
init = tf.global_variables_initializer()  # prepare an init node

with tf.Session() as sess:
    init.run()  # actually initialize all the variables
    result = f.eval()
```

4. Lifecycle of a Node Value

- When you evaluate a node, TensorFlow automatically determines the set of nodes that it depends on and it evaluates these nodes first.

```
w = tf.constant(3)
x = w + 2
y = x + 5
z = x * 3

with tf.Session() as sess:
    print(y.eval())  # 10
    print(z.eval())  # 15
```

Note: To evaluate y and z efficiently, without evaluating w and x twice, code, use:

```
with tf.Session() as sess:
    y_val, z_val = sess.run([y, z])
    print(y_val)  # 10
    print(z_val)  # 15
```

Summary

Exercises

- Check TensorFlow exercises on: https://github.com/Kyubyong/tensorflow-exercises