TensorFlow

Introduction

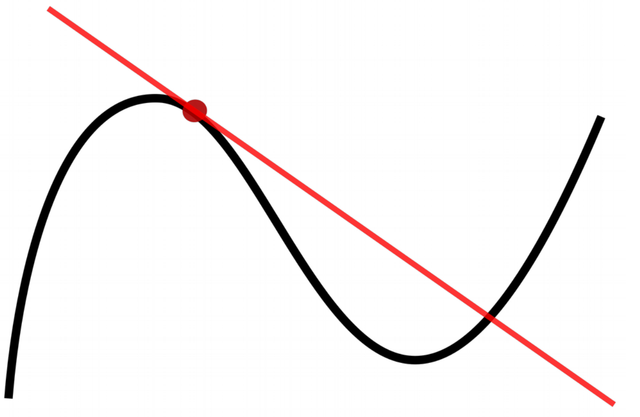
Intro

- Intro
- Computation Graph
- Gradients
- Parallelization

Intro

- Intro
- Computation Graph
- Gradients
- Parallelization

Intro



Tensorflow (TF)

- ~~Machine Learning (ML) Framework~~
- Numeric Computation Framework
- Plethora of Numeric Functions and Algorithms
- Computes Gradients (analytically & automatically)
- Helps with Parallelization & Distribution
  (CPU, GPU, Compute Clusters, ...)
- Executable on many Platforms
  (Android, iOS, Browser, ...)

Intro/Overview

Steps of using TF:

1) Create Numeric Program
   (Compuation Graph)

2) Run Program (in a Session)
   - Train/Optimize (optional)
   - Test/Execute

3) Compile to Binary (optional)

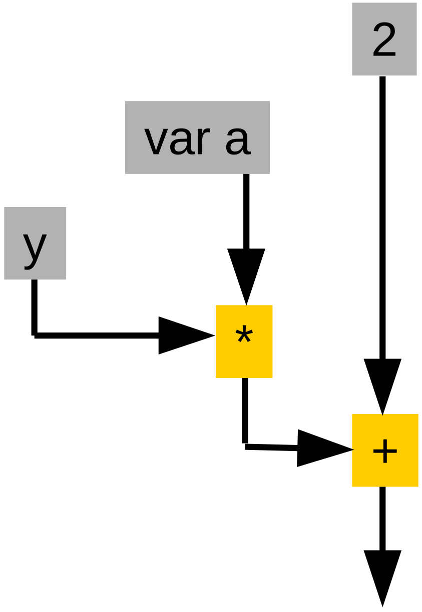
Computation Graph

- Intro
- Computation Graph
- Gradients
- Parallelization

Computation Graph

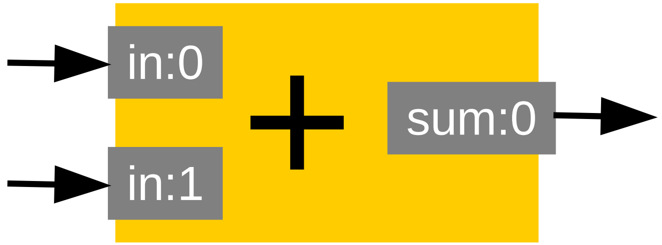
Consists of:

- Constants: Unmodifiable computation input
- Variables: Re-assignable, stored in Session, used for configuration & training
- Placeholders: Must be specified every run, comparable to (named) function parameters
- Operations
  - Functions: sin, cos, … (recursion possible)
  - Control Flow: (loops, conditional, …)



Computation Graph/Operations

- Operations may be stateful
- Ports:
  - Inputs and Outputs of Operations
  - 1 ≤ ports per operation
  - Control ports for synchronization
- Tensors:
  - Data flowing between ports
  - N-dimensional arrays of values (float, int, complex, ...)
  - Similar to NumPy arrays



Computation Graph is combined Control- & Data-Flow Graph
(iss.ices.utexas.edu/Publications/Papers/ICPP1990b.pdf)

Intro/Example

NumPy

```python
import numpy as np

y = np.array([[1, 2],
              [3, 4]])

def x_plus_y( x ):
  return x+y

sum_1 = x_plus_y( x=[[10, 20],
                     [30, 40]] )
print('\nsum_1:\n%s' % sum_1)

y = np.array([[5, 6],
              [7, 8]])

sum_2 = x_plus_y( x=[[100, 200],
                     [300, 400]] )
print('\nsum_2:\n%s' % sum_2)
```

Tensorflow

Graph Creation

```python
import tensorflow as tf

x = tf.placeholder(tf.float32, shape=[2,2], name='x')
y = tf.Variable([[1, 2],
                 [3, 4]], dtype=tf.float32, name='y')
x_plus_y = x+y

init_all_vars = tf.global_variables_initializer()

with tf.Session() as sess:
  sess.run(init_all_vars)

  sum_1 = sess.run(x_plus_y, feed_dict={
    x: [[10, 20],
        [30, 40]]
  })
  print('\nsum_1:\n%s' % sum_1)

  sess.run( y.assign([[5, 6],
                      [7, 8]]) )

  sum_2 = sess.run(x_plus_y, feed_dict={
    x: [[100, 200],
        [300, 400]]
  })
  print('\nsum_2:\n%s' % sum_2)
```

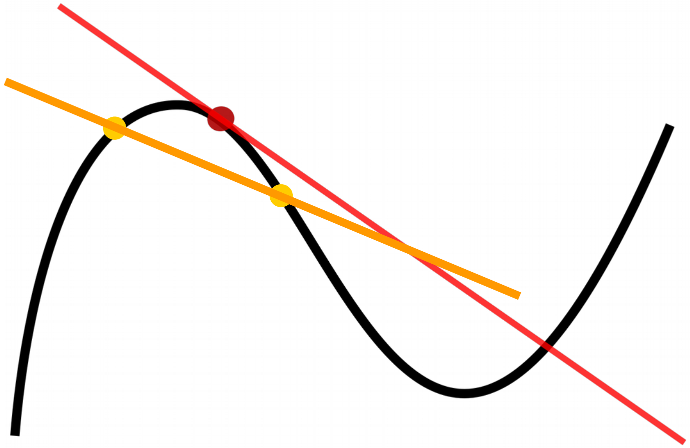
Execution

Gradients

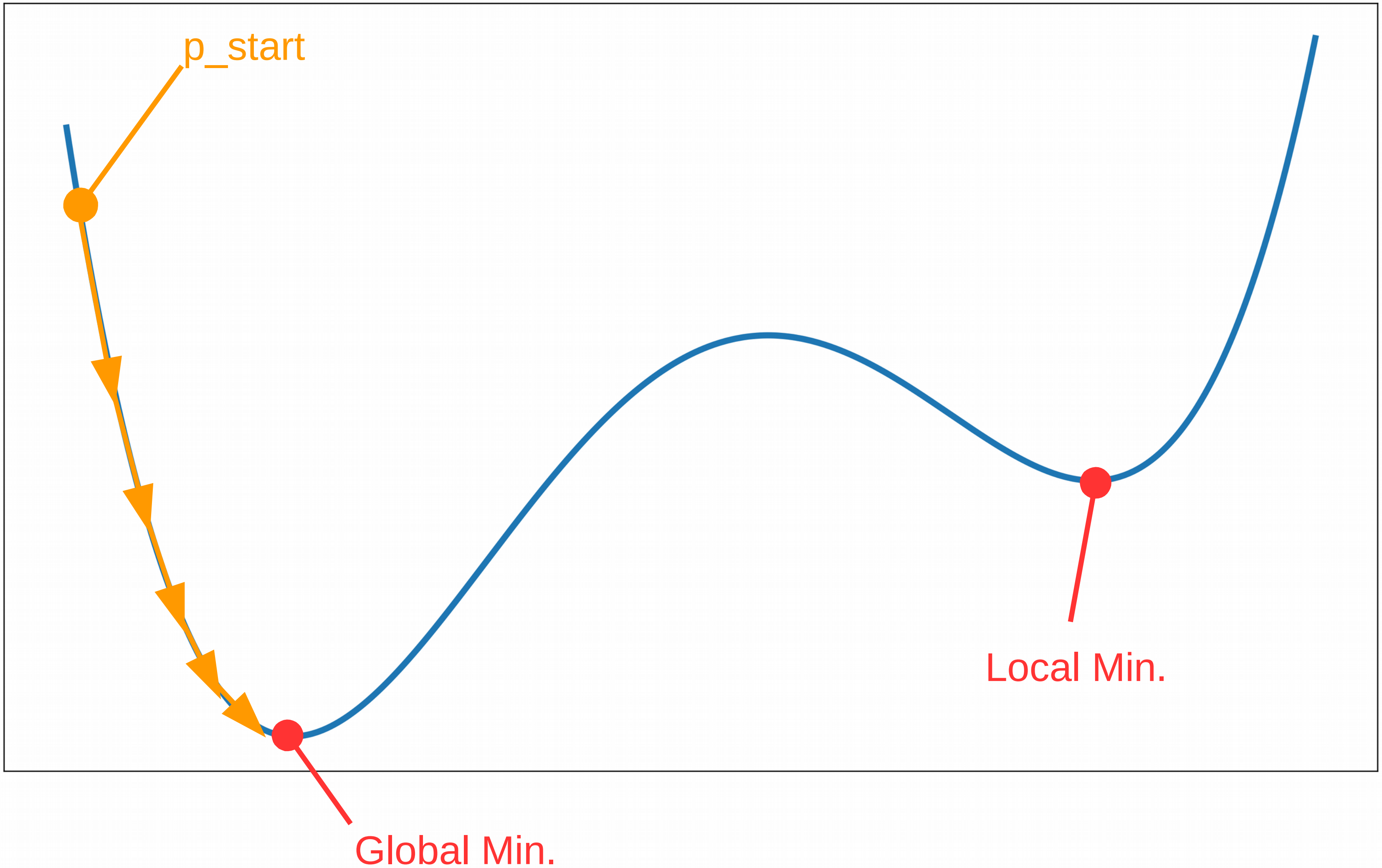
- Intro
- Computation Graph
- Gradients
- Parallelization

Gradients/Application

- Optimization & Equation Solving is ubiquitous
  - ML: Training ≙ Loss Function Minimization
  - Engineering: Stiffness Maximization, …
  - Finance: Cost Optimization, …
- Most Problems non-linear
  - ➔ Iterative solution (Newton, BFGS, …)
  - Usually requires derivative/gradients
  - Approximation (Finite Differences) slow and imprecise
  - TF computes gradients quickly and precisely
    (Using Back Propagation)



Gradients/Application



Gradients/Example

```python
import tensorflow as tf

a = tf.Variable(7, dtype=tf.float32,  name='a')
x = tf.placeholder(tf.float32, shape=[], name='x')
f = a * x**2 # <- MUST HAVE SCALAR OUTPUT

df_dx, df_da = tf.gradients(f, [x,a])

ddf_dx_da, = tf.gradients(df_dx, [a])

init_vars = tf.global_variables_initializer()

with tf.Session() as sess:
  sess.run(init_vars)

  df_dx_3 = sess.run(df_dx, feed_dict={ x: 3 })
  print('\n df/dx = 2*a*x\n(df/dx)(a=7, x=3) =', df_dx_3)

  ddf_dx_da_3 = sess.run(ddf_dx_da, feed_dict={ x: 3 })
  print('\n ddf/dx/da = 2*x\n(ddf/dx/da)(x=3) =',ddf_dx_da_3)
```

```
 df/dx = 2*a*x
(df/dx)(a=7, x=3) = 42.0

 ddf/dx/da = 2*x
(ddf/dx/da)(x=3) = 6.0
```

Parallelization

- Intro
- Computation Graph
- Gradients
- Parallelization

Parallelization

- TF automatically parallel
- TF guesses best device placement (GPU, CPU, ...)
- Manual device placement possible
- NVIDIA GPU owners: pip install tensorflow-gpu

```
from tensorflow.python.client import device_lib

for dev in device_lib.list_local_devices():
  print( 'name: "%s"' % dev.name )
  print( 'type: "%s"' % dev.device_type )
  print( ' mem: %.2f GB' % (dev.memory_limit / 1e9) )
  print( '----')

with tf.device('/cpu:0'):
  a = tf.constant(1.0)
  b = tf.constant(2.0)
  a_plus_b = a+b

with tf.Session() as sess:
  print( sess.run(a_plus_b) )
```

TensorFlow

That‘s it! Questions?

Image source: commons.wikimedia.org

Sources

- Tensorflow Logo
  http://hilite.me/
  (Presenter is in no way affiliated)
- Tangent Sketch [Slide 4]:
  https://en.wikipedia.org/wiki/Tangent
- Python code examples:
  http://hilite.me/