TensorFlow

Instalation *(requirements)*

https://www.tensorflow.org/install/

P-E-R-F-E-C-T documentation with examples

Main idea:

0) Create session
1) Build graph
1.5) Initialize variables
2) Usage „fead ang get”
3) Release of resources (optional)

Example 1

```python
#!/usr/bin/env python3

import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
import tensorflow as tf

GPU = 1
N = 3

#---------------------------------------------------------

config = tf.ConfigProto( device_count = {'GPU': GPU} )
sess = tf.InteractiveSession(config=config)
# or just sess = tf.InteractiveSession() # tf.Session()

plh = tf.placeholder(tf.float32, [None, None, None])
out = tf.matrix_determinant(plh)

sess.graph.finalize()

#---------------------------------------------------------

A = np.random.normal(0, 1, (2, N,N))

B = sess.run(out, feed_dict={plh: A})

print(A)
print("_____________________")
print(B)

# sess.close()
```

```
[[[-1.51518729  0.11046973  0.60482384]
  [-1.14696716  0.60839331  0.64082947]
  [ 0.83113608 -1.46129475  0.78067179]]

 [[-0.19216924  0.64811475 -0.19654056]
  [-1.58809467 -0.96204818 -1.98305563]
  [ 1.79761185  1.74759851  1.20165309]]]
_____________________
[-1.27289093 -1.31180155]
```

tf.placeholder

- input to the graph
- 3 dims:

  BATCH, N, M

Variables

- Created by ***tf.Variable*** or ***tf.get_variable***
- Hard to modify manually *(need to create operation as a part of graph)* op = variable.assign(value)
- Could be trained with optimizers

// don’t try to print them !!!

Example 2

```python
#!/usr/bin/env python3

import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
import tensorflow as tf

GPU = 1
N = 3
M = 4

#---------------------------------------------------

sess = tf.InteractiveSession()

plh = tf.placeholder(tf.float32, [None, M])
var = tf.Variable(tf.ones([M, N], tf.float32))
#                 tf.zeros(shape_list, dtype)
#                 tf.random_uniform(shape_list, min, max)
#     tf.Variable(np.arange(M*N, dtype=np.float32).reshape(M, N))

c = tf.constant(3.0)

out = plh @ var + c

tf.global_variables_initializer().run()

sess.graph.finalize()

#---------------------------------------------------

A = np.random.normal(0, 1, (2, M))

B = sess.run(out, feed_dict={plh: A})

print(A)
print("_____________________")
print(B)

# sess.close()
```

```
[[ 0.49067577  0.85022828  0.32803305  0.85816675]
 [-0.34574193 -0.55422166 -0.30443877  0.68917646]]
_____________________
[[ 5.5271039   5.5271039   5.5271039 ]
 [ 2.48477411  2.48477411  2.48477411]]
```

Now because of variable, placeholder has to have defined at least one dim

Few different ways to initialize variable...

tf.constant – unrecommended! *(not optimized)*

dtypes

- tf.float16: 16-bit half-precision floating-point.
- tf.float32: 32-bit single-precision floating-point.
- tf.float64: 64-bit double-precision floating-point.
- tf.bfloat16: 16-bit truncated floating-point.
- tf.complex64: 64-bit single-precision complex.
- tf.complex128: 128-bit double-precision complex.
- tf.int8: 8-bit signed integer.
- tf.uint8: 8-bit unsigned integer.
- tf.uint16: 16-bit unsigned integer.
- tf.int16: 16-bit signed integer.
- tf.int32: 32-bit signed integer.
- tf.int64: 64-bit signed integer.
- tf.bool: Boolean.
- tf.string: String.
- tf.qint8: Quantized 8-bit signed integer.
- tf.quint8: Quantized 8-bit unsigned integer.
- tf.qint16: Quantized 16-bit signed integer.
- tf.quint16: Quantized 16-bit unsigned integer.
- tf.qint32: Quantized 32-bit signed integer.
- tf.resource: Handle to a mutable resource.