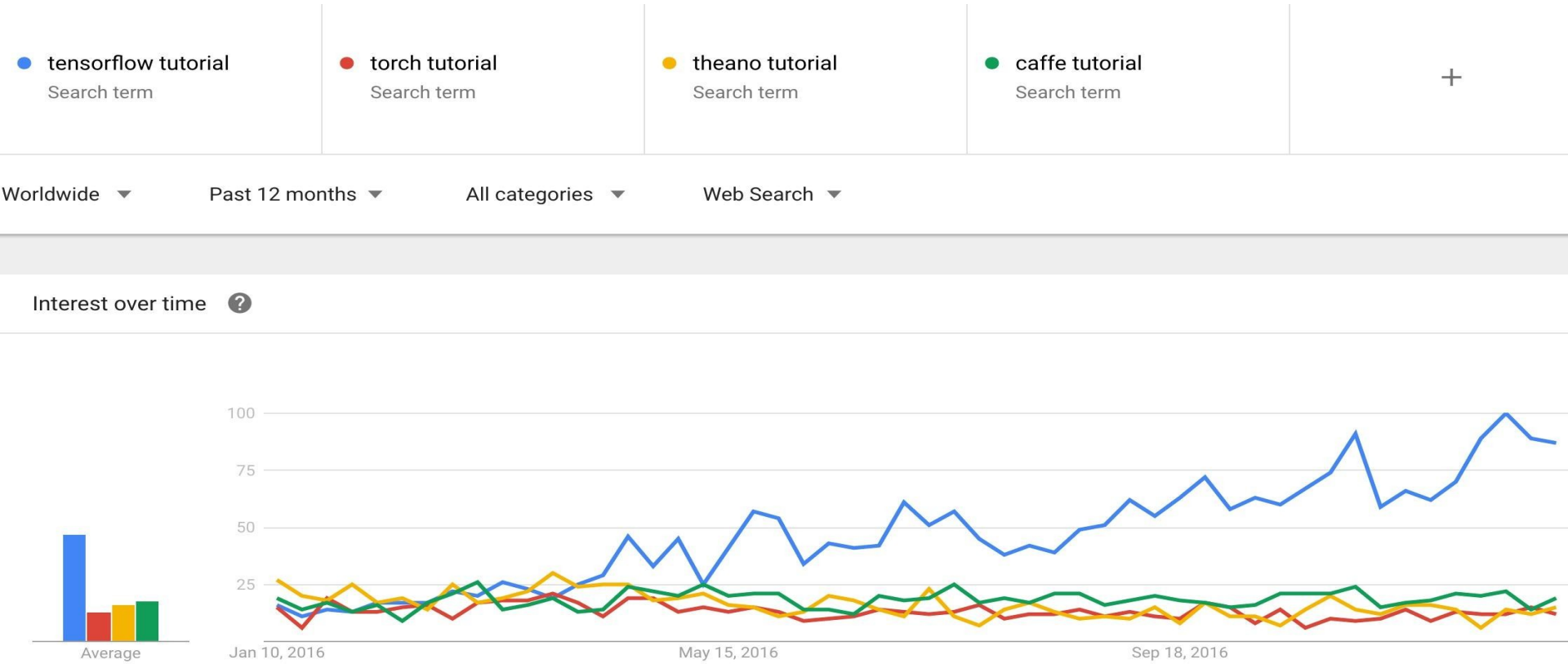
Tensorflow基础

寒小阳
2017-07-29

为什么选Tensorflow



为什么选Tensorflow

- Python 接口
- 高移植性：在一个/多个CPUs 或GPUs 的笔记本电脑，服务器，甚至移动端，都可以用同样的API
- 灵活性：适用Android, Windows, iOS, Linux 等
- 可视化：TensorBoard简直太赞
- Checkpoints：实验状态保存与恢复
- 自动微分/求导(复杂网络也不怕了)
- 强大的社区 (> 20,000 commits， > 6000 TF-related repos in 1 year)
- 有很多公司和项目已经是用的TensorFlow

为什么选Tensorflow

使用Tensorflow的公司

- Google
- OpenAI
- DeepMind
- Snapchat
- Uber
- Airbus
- eBay
- Dropbox
- BAT部分team
- 各种创业公司

进入正题

import tensorflow as tf

如果有同学想要更简单的用法

1. TF Learn (tf.contrib.learn): 习惯scikit-learn的fit函数的同学们，这是tensorflow界的scikit-learn

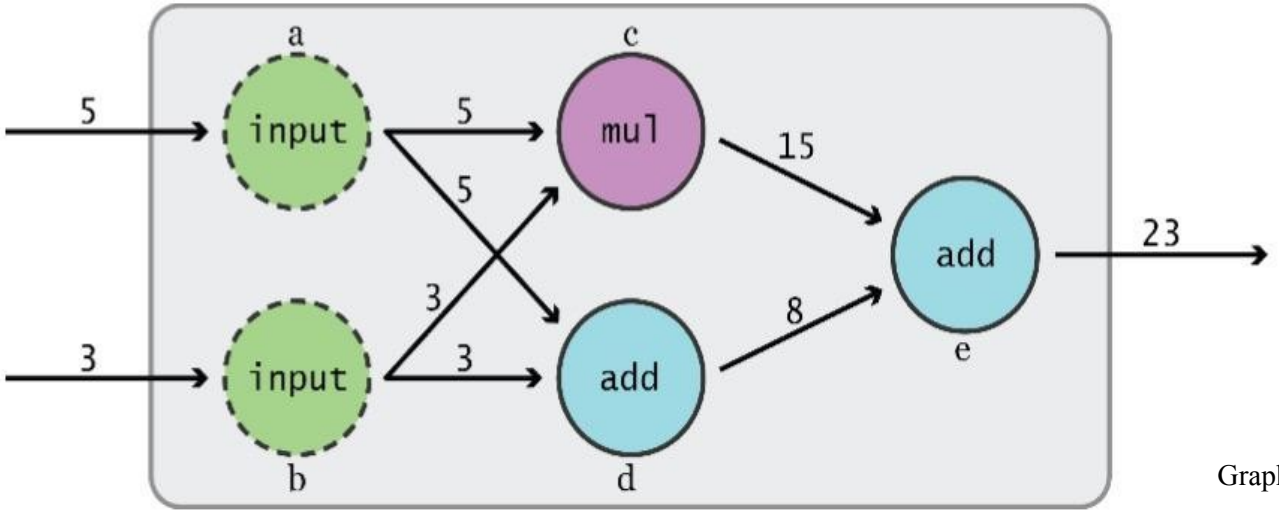
2. TF Slim (tf.contrib.slim): 这是TensorFlow里的一个轻量浓缩版本高级接口，可以很方便地定义/训练/评估浮躁的网络结构模型

3. 还想更简单？参考
   - Keras
   - TFLearn
   - Tensorlayer

核心概念Graph和Session

Data Flow Graph

数据流图



Graph by TFFMI

计算定义≠执行计算

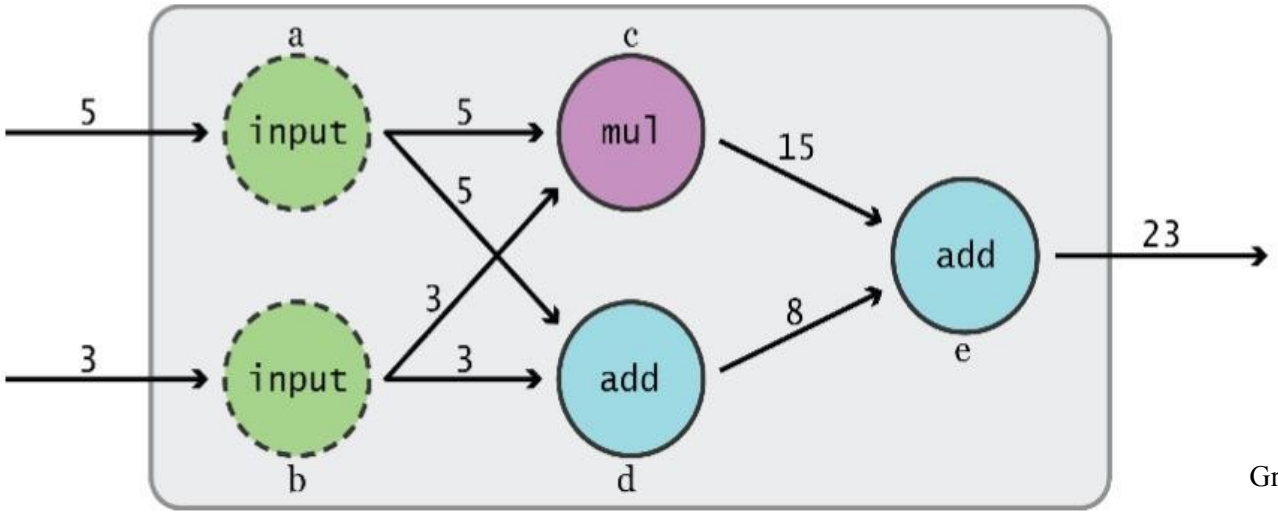
计算的定义和执行，被很好地分离开了

核心概念Graph和Session

模型跑起来，你需要2步：

❶ 描绘整幅图(定义计算)

❷ 在session当中执行图中的运算

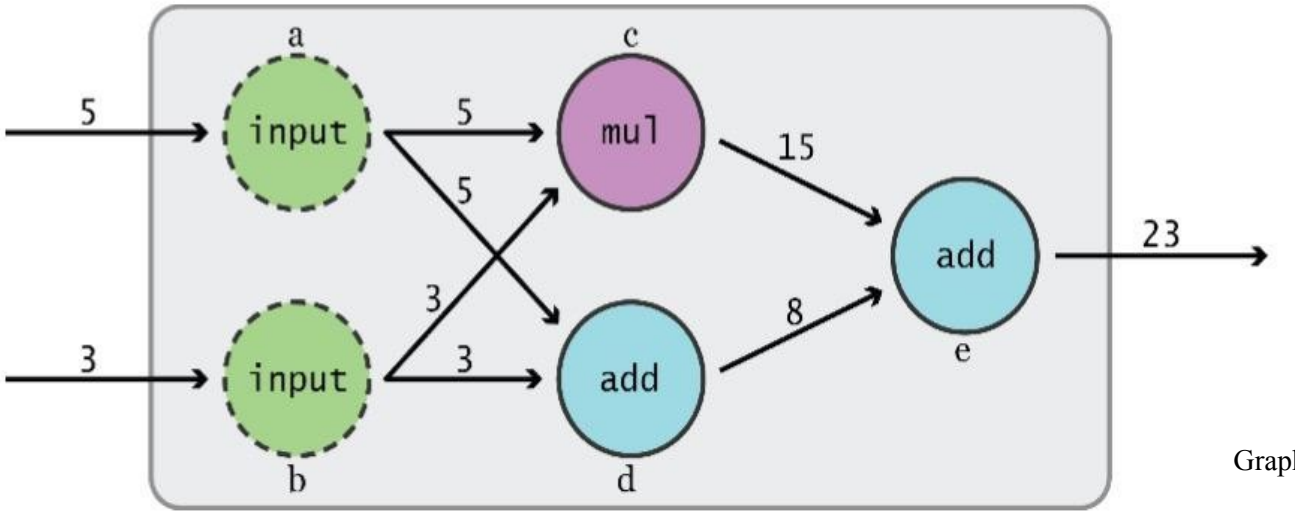


Graph by TFFMI

TensorFlow

Tensor + Flow

张量在图中通过运算(op)进行传递和变换



Graph by TFFMI

Tensor是什么

Tensor/张量：

在tensorflow里，大家可以理解成一个n维的矩阵

0-d tensor: 标量/数 scalar (number)

1-d tensor: 向量 vector

2-d tensor: 矩阵 matrix

...

Tensor

Numpy vs Tensorflow

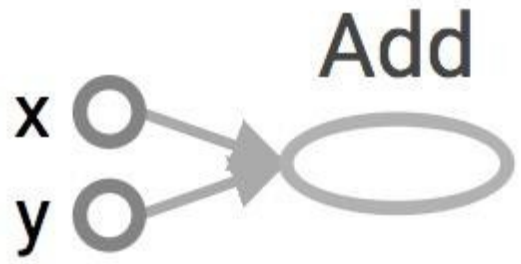
Numpy	Tensorflow
a = np.zeros((2,2)); b = np.ones((2,2))	a = tf.zeros((2,2)), b = tf.ones((2,2))
np.sum(b, axis=1)	tf.reduce_sum(a,reduction_indices=[1])
a.shape	a.get_shape()
np.reshape(a, (1,4))	tf.reshape(a, (1,4))
b*5+1	b*5+1
np.dot(a,b)	tf.matmul(a, b)
a[0,0], a[:,0], a[0,:]	a[0,0], a[:,0], a[0,:]

Tensor

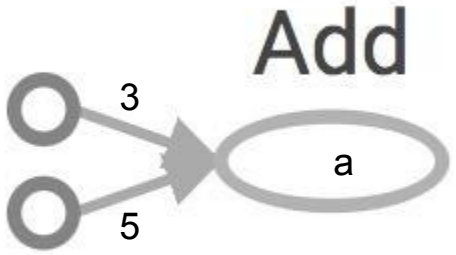
在Tensorflow的计算图里看

```
import tensorflow as tf
a = tf.add(3, 5)
```

TensorBoard的可视化结果



x,y是什么？
Tensorflow在你没有指定名称的时候会自动命名
x = 3
y = 5

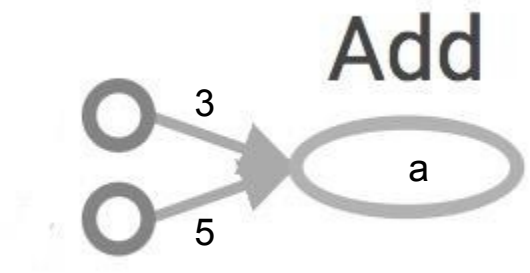


在“这幅图”里
节点: operators, variables, and constants
边: tensors

Tensor

在Tensorflow的计算图里看

```
import tensorflow as tf
a = tf.add(3, 5)
print a
```




```
>> Tensor("Add:0", shape=(), dtype=int32)
```

结果不是8
如何取到结果？

Session

初始化session，完成操作

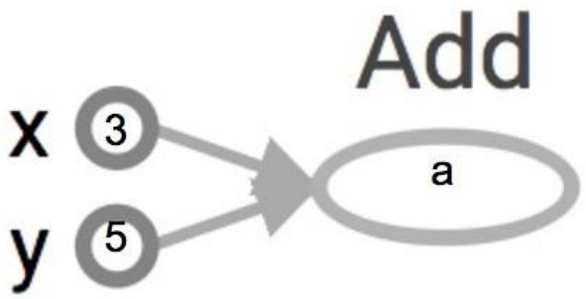
```
import tensorflow as tf

a = tf.add(3, 5)

sess = tf.Session()

print sess.run(a)

sess.close()
```

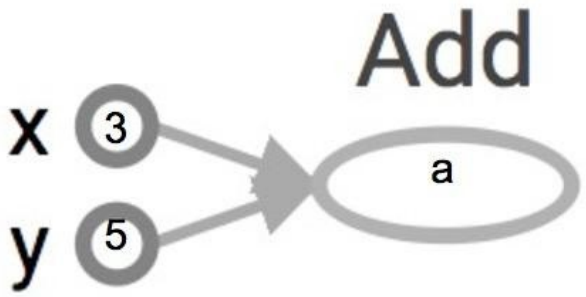


Session会在计算图里找到a的依赖，把依赖的节点都进行计算

Session

建议的session写法如下

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
with tf.Session() as sess:
    print sess.run(a)
sess.close()
```

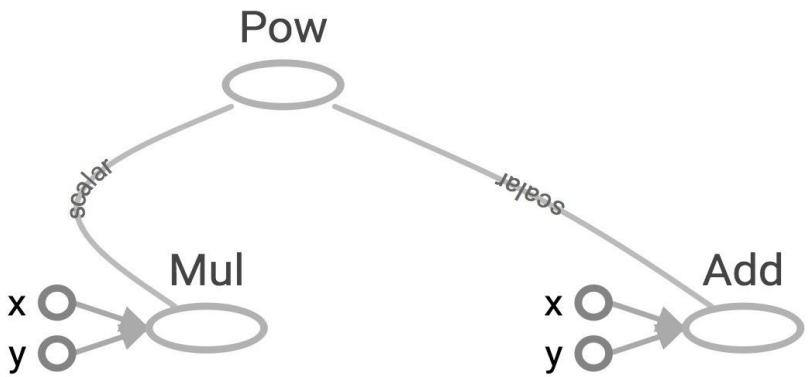


Session会在计算图里找到a的依赖，把依赖的节点都进行计算

Graph && Session

我们来看一个复杂一点点的例子

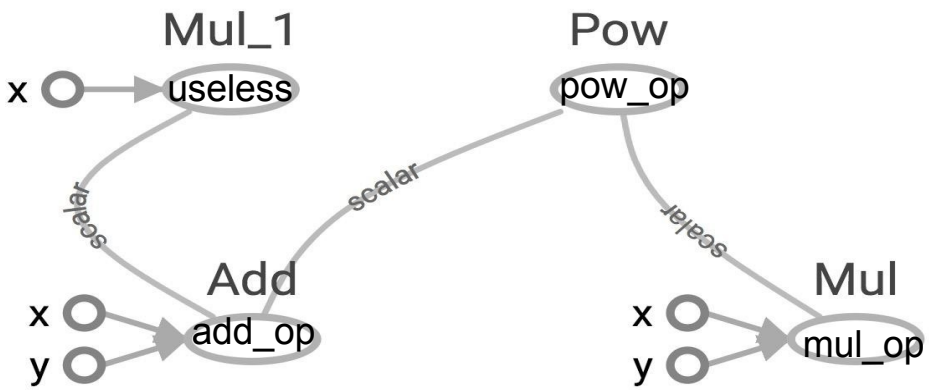
```
x = 2
y = 3
op1 = tf.add(x, y)
op2 = tf.mul(x, y)
op3 = tf.pow(op2, op1)
with tf.Session() as sess:
   op3 = sess.run(op3)
```



Graph && Session

我们不需要的依赖，其实不会进行计算，比如下面的例子：

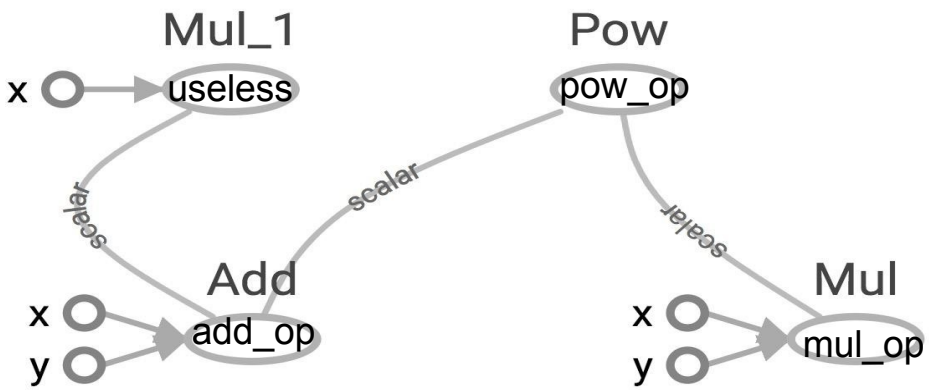
```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.mul(x, y)
useless = tf.mul(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
   z = sess.run(pow_op)
```



Graph && Session

如果我需要运行几个运算节点，比如上例中的useless：

```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.mul(x, y)
useless = tf.mul(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
   z, not_useless = sess.run([pow_op, useless])
```

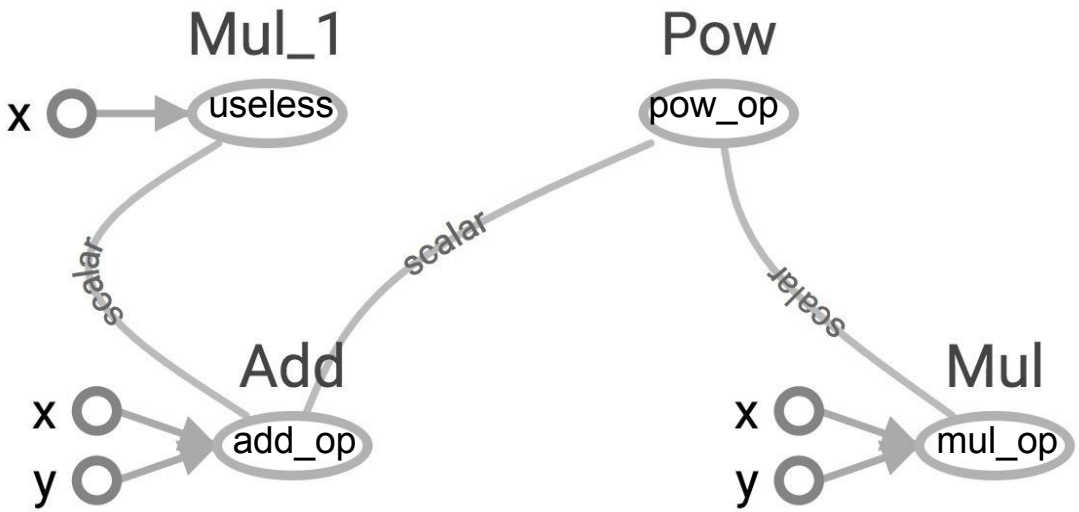


Graph && Session

更全的格式是下面这样，我们把所有需要的变量编成list放到fetches里：

```
x = 2
y = 3
…

tf.Session.run(fetches, feed_dict=None,
               options=None, run_metadata=None)
```



指定CPU/GPU

指定CPU or GPU去完成session里的运算:

```
# 构建graph.
with tf.device('/gpu:2'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='b')  c
  = tf.matmul(a, b)

# 构建session，设置log_device_placement为True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

# 运行op定义的运算
print sess.run(c)
```

思考一下：为什么需要Graph

1. 节省资源高效运算(我们只会计算你需要的结果依赖的子图
   - 回想useless例子
2. 把整个运算分解成子环节，方便自动求导
3. 对分布式运算很友好，计算工作可以分给多个GPU或者多个CPU或者多个设备运算
4. 很多机器学习的模型本身也非常适合组织成图格式

先了解一下tesnorboard

了解图结构/可视化的利器

在定义完计算图 和 运行session之前使用summary writer

```
import tensorflow as tf
a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a, b)
with tf.Session() as sess:
    #写到日志文件里
    writer = tf.summary.FileWriter('./graphs, sess.graph)
    print sess.run(x)
    writer.close() # 关闭writer
```

先了解一下tesnorboard

了解图结构/可视化的利器
命令行解析日志，浏览器端可视化

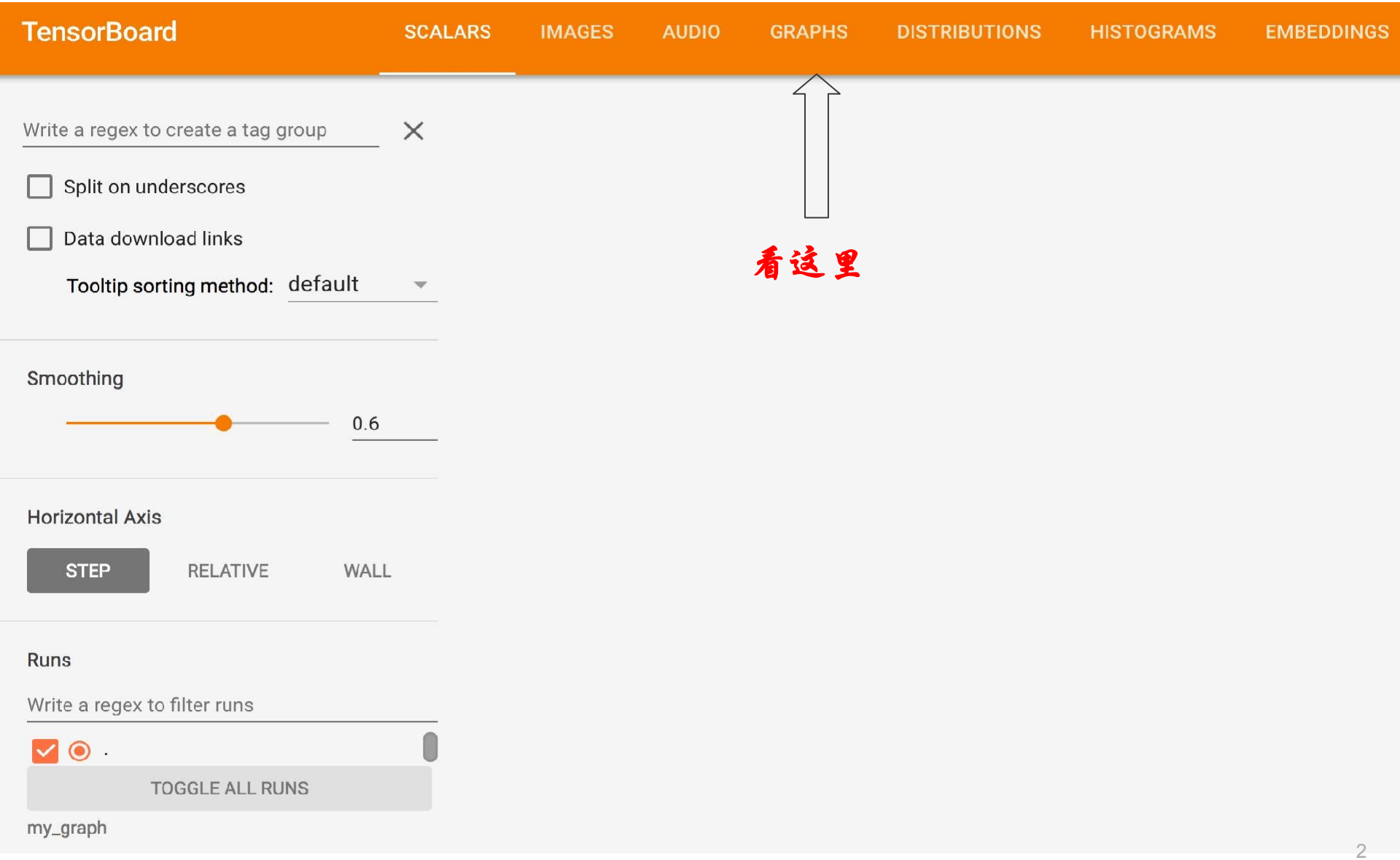
在命令行端运行:

```
$ python [yourprogram].py

$ tensorboard --logdir="./graphs" -port 7001
```

打开google浏览器访问: http://localhost:7001/

先了解一下tesnorboard



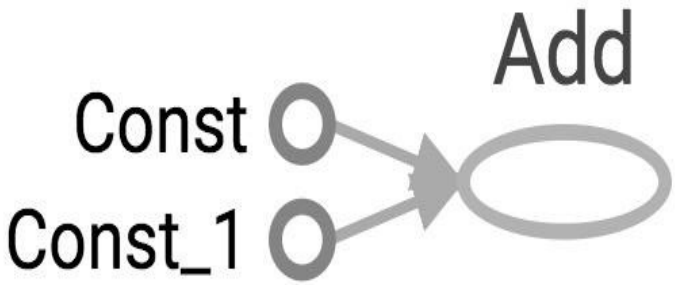
看这里

先了解一下tesnorboard

了解图结构/可视化的利器

在定义完计算图 和 运行session之前使用summary writer

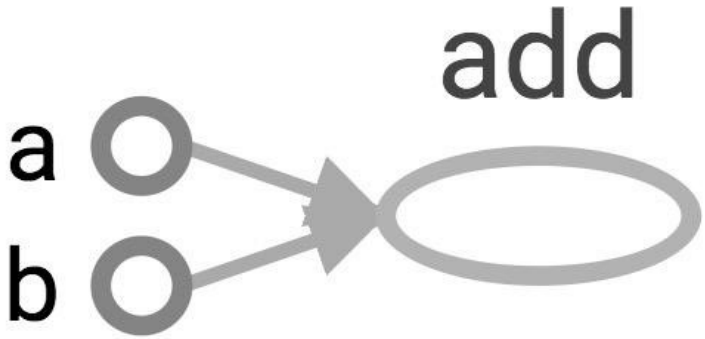
```
import tensorflow as tf
a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a, b)
with tf.Session() as sess:
    #写到日志文件里
    writer = tf.summary.FileWriter('./graphs, sess.graph)
    print sess.run(x)
    writer.close() # 关闭writer
```



先了解一下tesnorboard

自定义常量名称？
可以自己指定

```
import tensorflow as tf
a = tf.constant(2, name="a")
b = tf.constant(3, name="b")
x = tf.add(a, b, name="add")
with tf.Session() as sess:
    #写到日志文件里
    writer = tf.summary.FileWriter('./graphs, sess.graph)
    print sess.run(x)
    writer.close() # 关闭writer
```



关于constant

和numpy其实很像

```
tf.constant(value, dtype=None, shape=None,
    name='Const', verify_shape=False)
```

```
import tensorflow as tf

a = tf.constant([2, 2], name="a")
b = tf.constant([[0, 1], [2, 3]], name="b")
x = tf.add(a, b, name="add")
y = tf.mul(a, b, name="mul")
with tf.Session() as sess:
        x, y = sess.run([x, y])
        print x, y
# >> [5 8] [6 12]
```

关于constant

TensorFlow provides several operations that you can use to generate constants.

- `tf.zeros`
- `tf.zeros_like`
- `tf.ones`
- `tf.ones_like`
- `tf.fill`
- `tf.constant`

Sequences

- `tf.linspace`
- `tf.range`

https://www.tensorflow.org/api_guides/python/constant_op

关于constant

随机常量

```
tf.random_normal(shape, mean=0.0, stddev=1.0,
                 dtype=tf.float32, seed=None, name=None)

tf.truncated_normal(shape, mean=0.0, stddev=1.0,
                 dtype=tf.float32, seed=None,  name=None)

tf.random_uniform(shape, minval=0, maxval=None,
                 dtype=tf.float32, seed=None,  name=None)

tf.random_shuffle(value, seed=None, name=None)

tf.random_crop(value, size, seed=None, name=None)

tf.multinomial(logits, num_samples, seed=None, name=None)

tf.random_gamma(shape, alpha, beta=None, dtype=tf.float32, seed=None, name=None)
```

https://www.tensorflow.org/api_guides/python/constant_op

关于operations

在tensor上可以进行各种运算/变换

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural network building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

https://www.tensorflow.org/api_guides/python/constant_op

关于operations

在tensor上可以进行各种运算/变换

```
a = tf.constant([3, 6])
b = tf.constant([2, 2])
tf.add(a, b) #>> [5 8]
tf.add_n([a, b, b]) #>> [7 10]
tf.mul(a, b) #>> [6 12]
tf.matmul(a, b) #>> ValueError
tf.matmul(tf.reshape(a,[1, 2]), tf.reshape(b,[2, 1])) #>> [[18]]
tf.div(a, b) #>> [1 3]
tf.mod(a, b) #>> [1 0]
```

https://www.tensorflow.org/api_guides/python/math_ops

Tensorflow数据类型

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.

https://www.tensorflow.org/programmers_guide/dims_types#data_types

Tensorflow变量

tf.constant是op，而tf.Variable是一个类，初始化的对象有多个op

```
# create variable a with scalar value
a = tf.Variable(2, name="scalar")

# create variable b as a vector
b = tf.Variable([2, 3], name="vector")

# create variable c as a 2x2 matrix
c = tf.Variable([[0, 1], [2, 3]], name="matrix")

# create variable W as 784 x 10 tensor, filled with zeros
W = tf.Variable(tf.zeros([784,10]))
```

https://www.tensorflow.org/programmers_guide/variables

Tensorflow变量

tf.constant是op，而tf.Variable是一个类，初始化的对象有多个op

```
x = tf.Variable(...)

x.initializer # 初始化
x.value() # 读取的op
x.assign(...) # 写入的op
x.assign_add(...) # 更多
```

https://www.tensorflow.org/programmers_guide/variables

Tensorflow变量

变量使用之前一定要初始化！！

最简单的初始化全部变量方法:

```
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
```

初始化一个变量子集:

```
init_ab = tf.variables_initializer([a, b], name="init_ab")
with tf.Session() as sess:
    sess.run(init_ab)
```

初始化单个变量：

```
W = tf.Variable(tf.zeros([784,10]))
with tf.Session() as sess:
    sess.run(W.initializer)
```

Tensorflow变量

输出变量内容：Eval()函数

```
# W 是一个700 x 100 随机变量
W = tf.Variable(tf.truncated_normal([700, 10]))
with tf.Session() as sess:
    sess.run(W.initializer)
    print W
    print W.eval()
```

```
>> Tensor("Variable/read:0", shape=(700, 10),
dtype=float32)

>> [[-0.76781619 -0.67020458……
```

Tensorflow placeholder

通过placeholder可以存放用于训练的数据

tf.placeholder(dtype, shape=None, name=None)

```
# create a placeholder of type float 32-bit, shape is a vector of 3 elements
a = tf.placeholder(tf.float32, shape=[3])

# create a constant of type float 32-bit, shape is a vector of 3 elements
b = tf.constant([5, 5, 5], tf.float32)

# use the placeholder as you would a constant or a variable
c = a + b  # Short for tf.add(a, b)

with tf.Session() as sess:
    print sess.run(c)
    # Error because a doesn't have any value
```

Tensorflow placeholder

通过placeholder可以存放用于训练的数据

tf.placeholder(dtype, shape=None, name=None)

```
# create a placeholder of type float 32-bit, shape is a vector of 3 elements
a = tf.placeholder(tf.float32, shape=[3])

# create a constant of type float 32-bit, shape is a vector of 3 elements
b = tf.constant([5, 5, 5], tf.float32)

# use the placeholder as you would a constant or a variable
c = a + b  # Short for tf.add(a, b)

with tf.Session() as sess:
    # feed [1, 2, 3] to placeholder a via the dict {a: [1, 2, 3]}
    print sess.run(c, {a: [1, 2, 3]})
    # the tensor a is the key, not the string 'a'

# >> [6, 7, 8]
```

Tensorflow placeholder

通过placeholder可以存放用于训练的数据

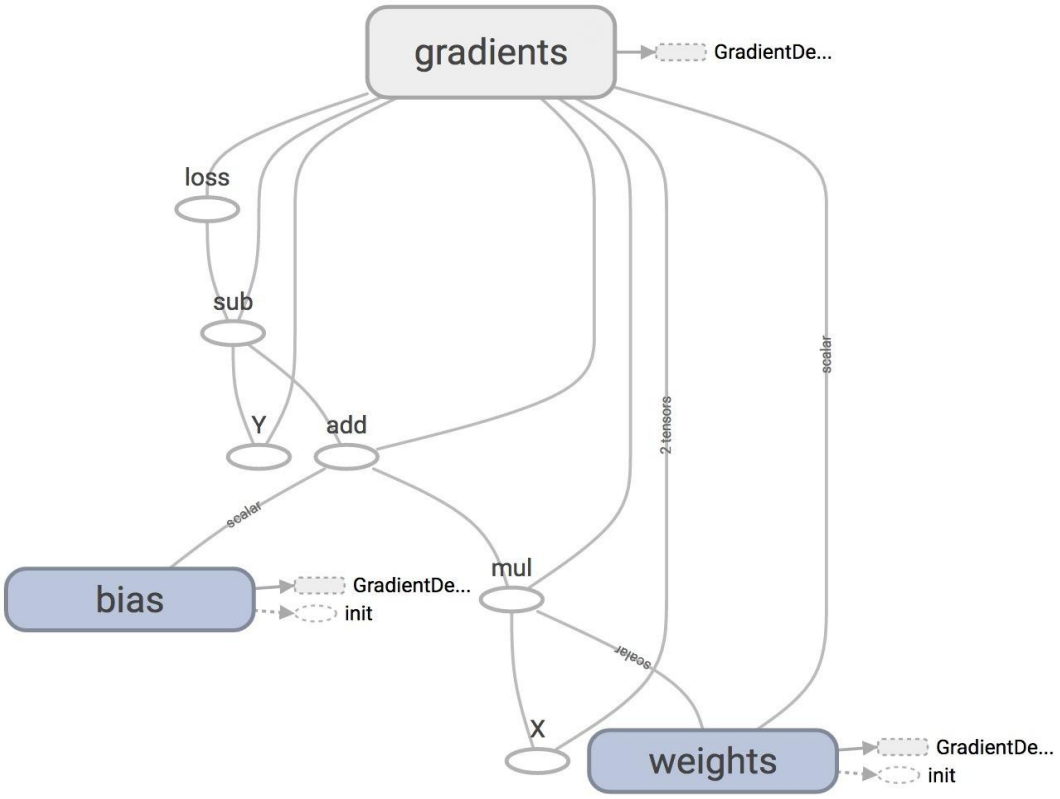
tf.placeholder(dtype, shape=None, name=None)

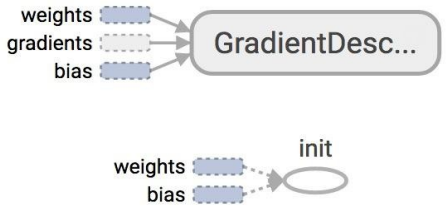
```
# create operations, tensors, etc (using the default graph)
a = tf.add(2, 5)
b = tf.mul(a, 3)

with tf.Session() as sess:
    # define a dictionary that says to replace the value of 'a' with 15
    replace_dict = {a: 15}

    # Run the session, passing in 'replace_dict' as the value to 'feed_dict'
    sess.run(b, feed_dict=replace_dict)
    # returns 45
```

试试构建逻辑回归


gradients
GradientDe...
loss
sub
Y
add
scalar
2 tensors
scalar
bias
GradientDe...
init
mul
scalar
X
weights
GradientDe...
init


weights
gradients
bias
GradientDesc...
init
weights
bias

感谢大家！

恳请大家批评指正！

寒小阳
hanxiaoyang.ml@gmail.com