南京大学 人工智能学院
SCHOOL OF ARTIFICIAL INTELLIGENCE, NANJING UNIVERSITY

Tencent 腾讯

智能应用建模

第二部分：TensorFlow入门

章宗长

2019年7月3日

内容安排

TensorFlow环境搭建

TensorBoard可视化

TensorFlow计算模型——计算图

TensorFlow数据模型——张量

TensorFlow运行模型——会话

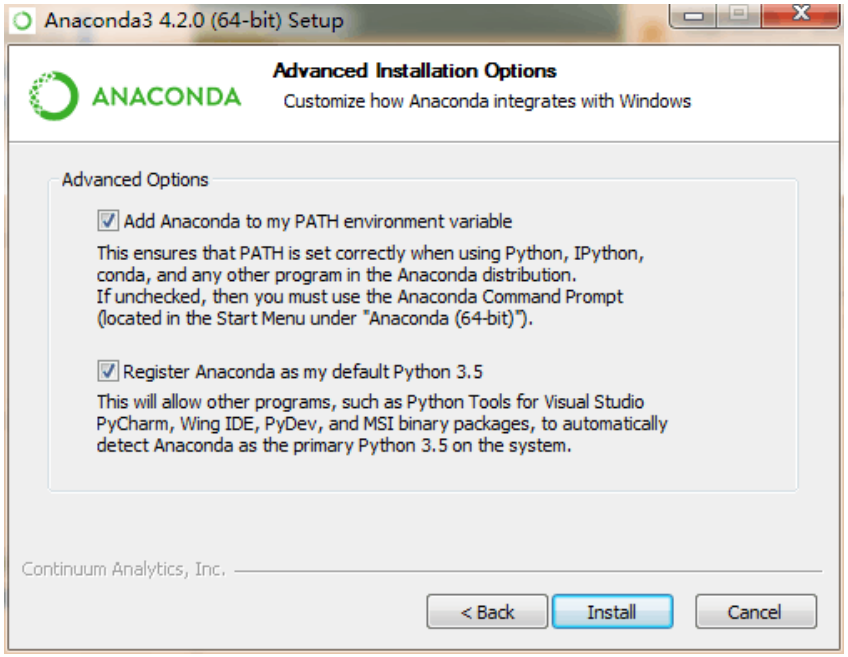
相关程序包和资料已存放在如下ftp地址中：

ftp://172.26.184.2/Download/AI Class

Anaconda3 4.2.0安装

https://repo.anaconda.com/archive/

Anaconda3-4.2.0-Linux-x86.sh	373.9M
Anaconda3-4.2.0-Linux-x86_64.sh	455.9M
Anaconda3-4.2.0-MacOSX-x86_64.sh	349.5M
Anaconda3-4.2.0-Windows-x86.exe	333.4M
Anaconda3-4.2.0-Windows-x86_64.exe	391.4M



TensorFlow安装

使用pip安装（Mac OS X环境，仅支持CPU的TensorFlow）

第一步：安装pip

```
$ sudo easy_install pip
$ sudo easy_install  -upgrade six
```

第二步：找到合适的安装包URL

```
$ export TF_BINARY_URL=
https://storage.googleapis.com/tensorflow/mac/cpu/tenso
rflow-1.4.0-py3-none-any.whl
```

第三步：通过pip安装TensorFlow

```
$ sudo pip3 install  -upgrade $TF_BINARY_URL
```

TensorFlow安装（续）

使用pip安装（Ubuntu/Linux 64-BIT环境，仅支持CPU的TensorFlow ）

第一步：安装pip

```
$ sudo apt-get install python-pip python-dev
```

第二步：找到合适的安装包URL

```
$ export TF_BINARY_URL=
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.4.0-py35-cp35m-linux_x86_64.whl
```

第三步：通过pip安装TensorFlow

```
$ sudo pip3 install  -upgrade $TF_BINARY_URL
```

TensorFlow安装（续）

使用pip安装（Windows环境，仅支持CPU的TensorFlow）

参考博客：https://zhuanlan.zhihu.com/p/33705179

第一步：升级pip

```
python -m pip install --upgrade pip
```

第二步：在Anaconda Prompt中输入：

```
conda create -n tensorflow_cpu python=3.5
```

看到Proceed ([y]/n)?时，输入y

回车运行后，输入activate tensorflow_cpu

然后，输入pip install tensorflow==1.4.0

TensorFlow安装（续）

Docker安装

https://docs.docker.com/install/

当Docker安装完成之后，可以通过以下命令来启动一个TensorFlow容器

$ docker run -it tensorflow/tensorflow:1.4.0

TensorFlow测试样例

（1）进入Python交互界面

$ python

（2）通过import操作加载TensorFlow

```
>>> import tensorflow as tf
```

（3）定义两个向量：a和b

```
>>> a = tf.constant([1.0,2.0], name='a')
>>> b = tf.constant([2.0,3.0], name='b')
```

（4）将这两个向量加起来

```
>>> result = a + b
```

TensorFlow测试样例（续）

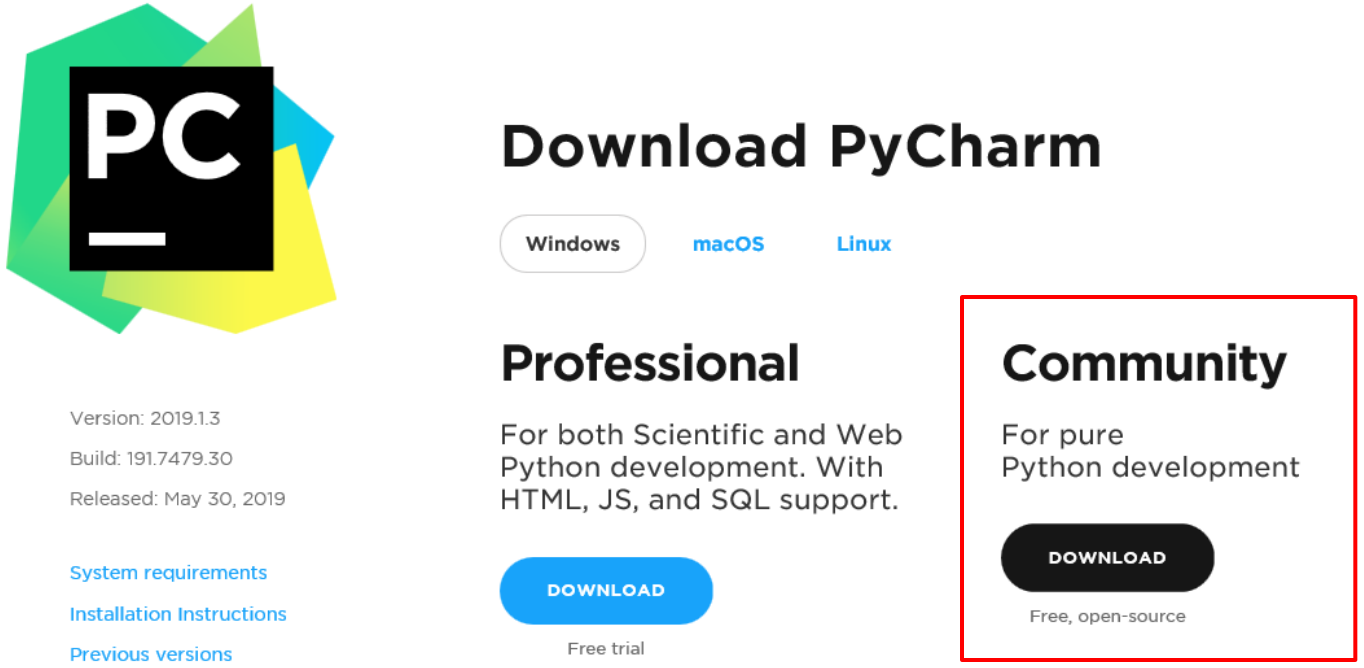
（5）输出相加得到的结果

```
>>> sess = tf.Session()
>>> print(sess.run(result))
>>> sess.close()

[3.  5.]
```

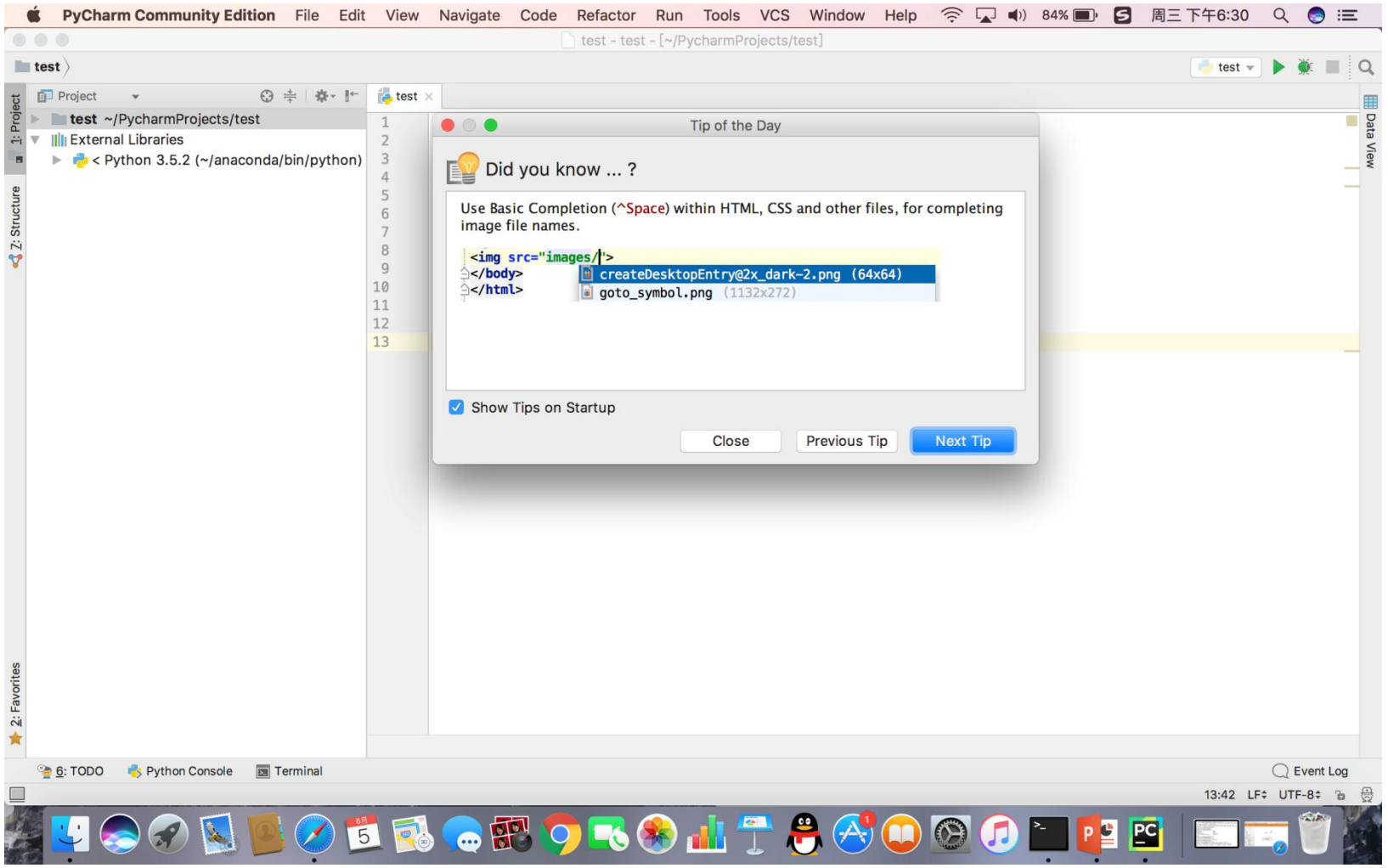
PyCharm安装

PyCharm是一种Python IDE，带有一整套可以帮助用户在使用Python语言开发时提高其效率的工具

https://www.jetbrains.com/pycharm/

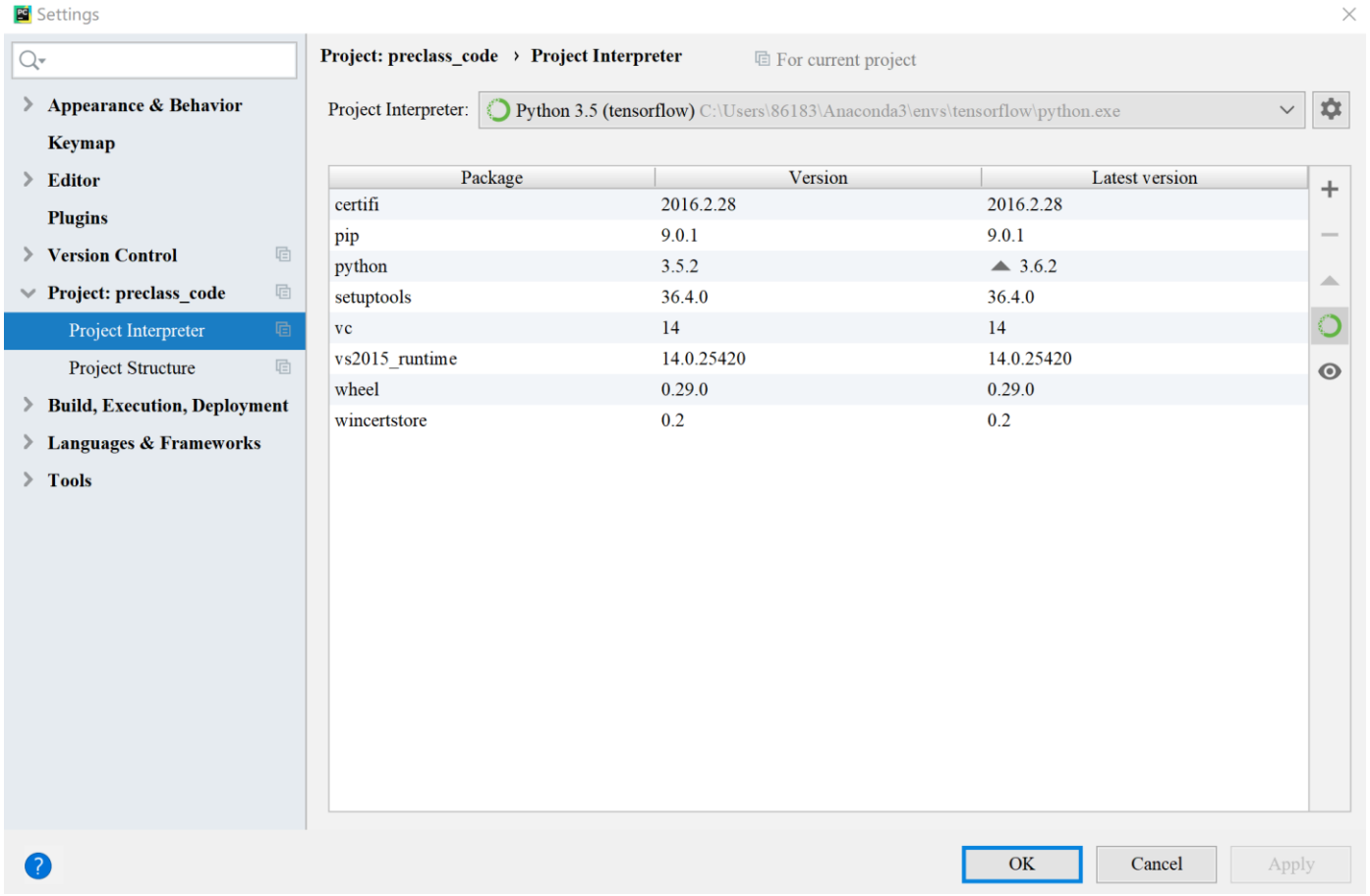


PyCharm安装（续）


PyCharm Community Edition File Edit View Navigate Code Refactor Run Tools VCS Window Help
test - test - [~/PycharmProjects/test]
Project
test ~/PycharmProjects/test
External Libraries
< Python 3.5.2 (~/anaconda/bin/python)
Tip of the Day
Did you know ... ?
Use Basic Completion (^Space) within HTML, CSS and other files, for completing image file names.
<img src="images/">
</body>
</html>
createDesktopEntry@2x_dark-2.png (64x64)
goto_symbol.png (1132x272)
Show Tips on Startup
Close
Previous Tip
Next Tip
6: TODO
Python Console
Terminal
Event Log
13:42 LF UTF-8

PyCharm安装（续）

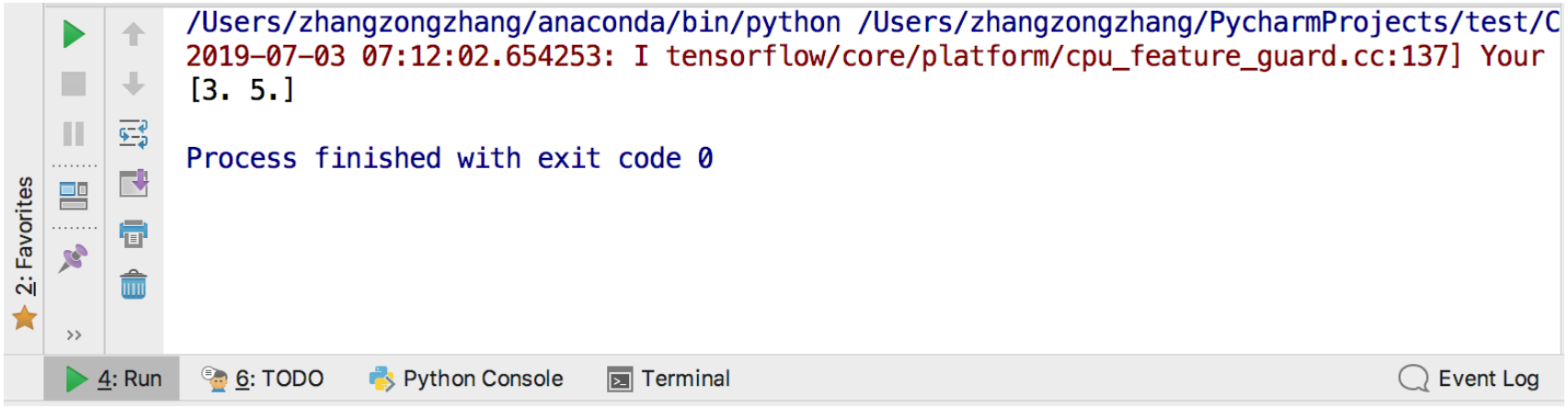
File-Settings–Project Interpreter选择tensorflow下的Python解释器



课堂作业

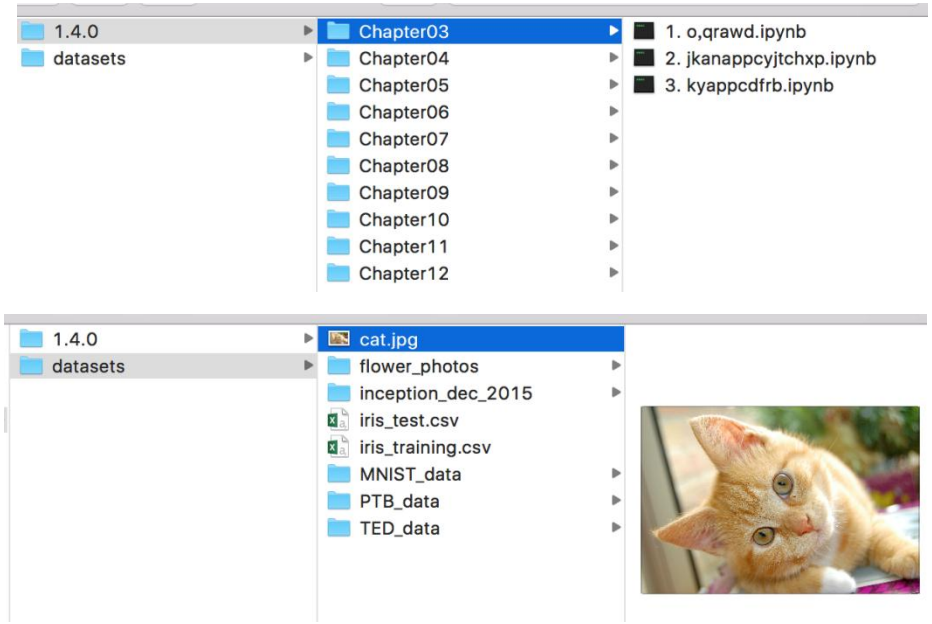
在PyCharm中输入第一个TensorFlow测试样例，运行程序

```
1  import tensorflow as tf
2
3  a = tf.constant([1.0,2.0], name="a")
4  b = tf.constant([2.0,3.0], name="b")
5
6  result = a + b
7
8  sess = tf.Session()
9  print(sess.run(result))
10
11 sess.close()
```



代码和资料文件下载

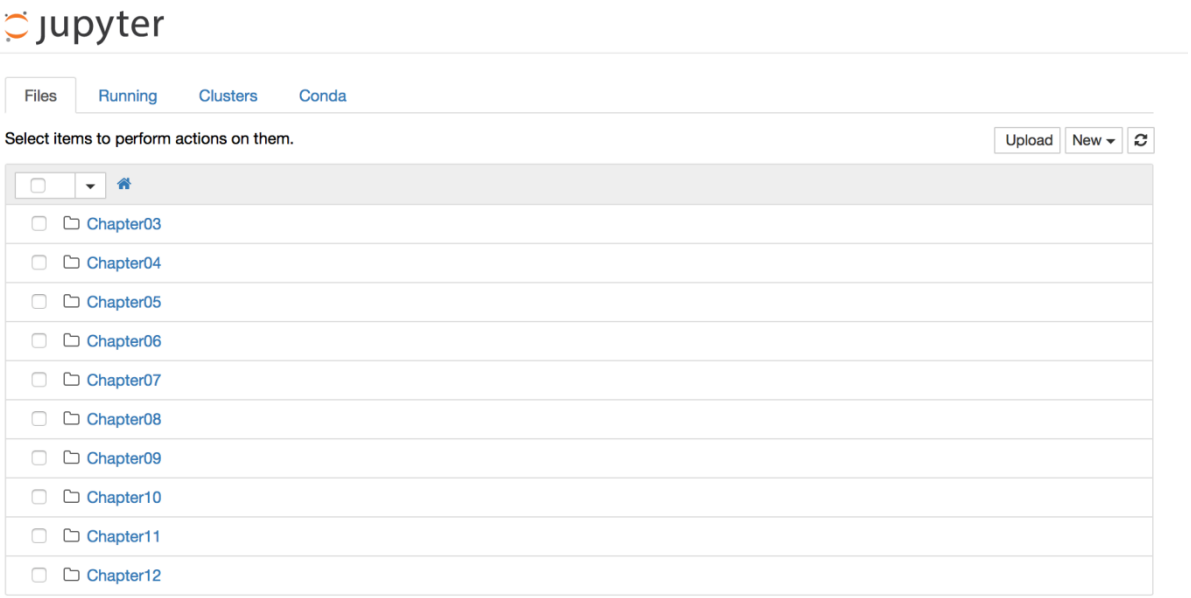
http://www.broadview.com.cn/book/5360



Jupyter Notebook

此前被称为Ipython notebook，是一个交互式笔记本，支持运行40多种编程语言

```
$ cd ~/Desktop/智能应用建模-课程/201806-githubtpuadl/1.4.0/
$ jupyter notebook
```



内容安排

- TensorFlow环境搭建
- **TensorBoard可视化**
- TensorFlow计算模型——计算图
- TensorFlow数据模型——张量
- TensorFlow运行模型——会话

TensorBoard可视化

TensorBoard是TensorFlow的可视化工具，它可以通过程序运行过程中输出的日志文件可视化TensorFlow程序的运行状态

TensorBoard不需要额外的安装过程，TensorFlow安装完成时，TensorBoard会自动安装

运行以下命令便可以启动TensorBoard：

```
$ tensorboard --logdir=/path/to/log       (Mac OS X)
> tensorboard --logdir=D:\path\to\log      (Windows)
```

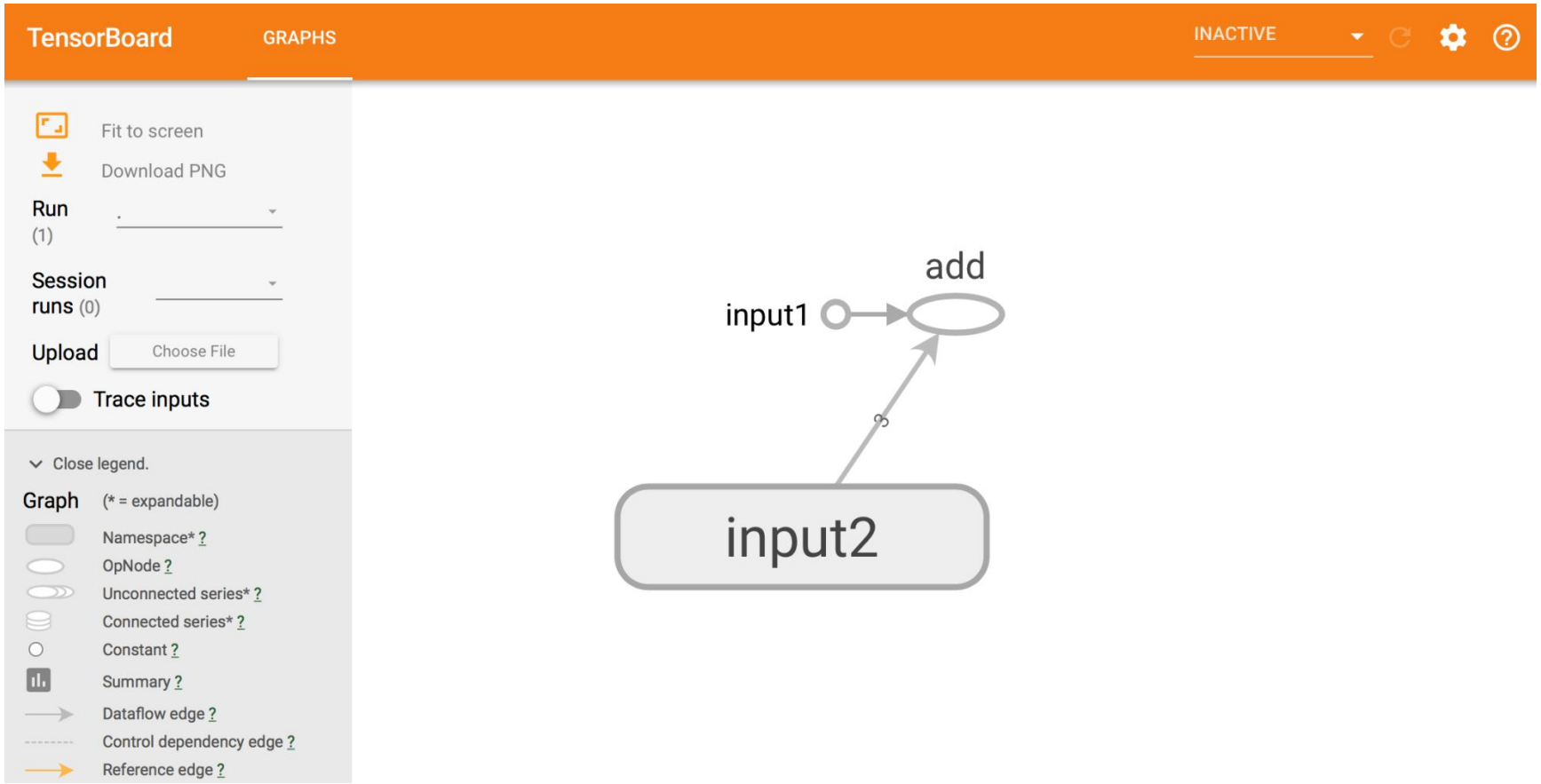
通过浏览器打开 **localhost:6006**

TensorBoard可视化（续）

以下代码展示了一个简单的TensorFlow程序，在这个程序中完成了TensorBoard日志输出的功能

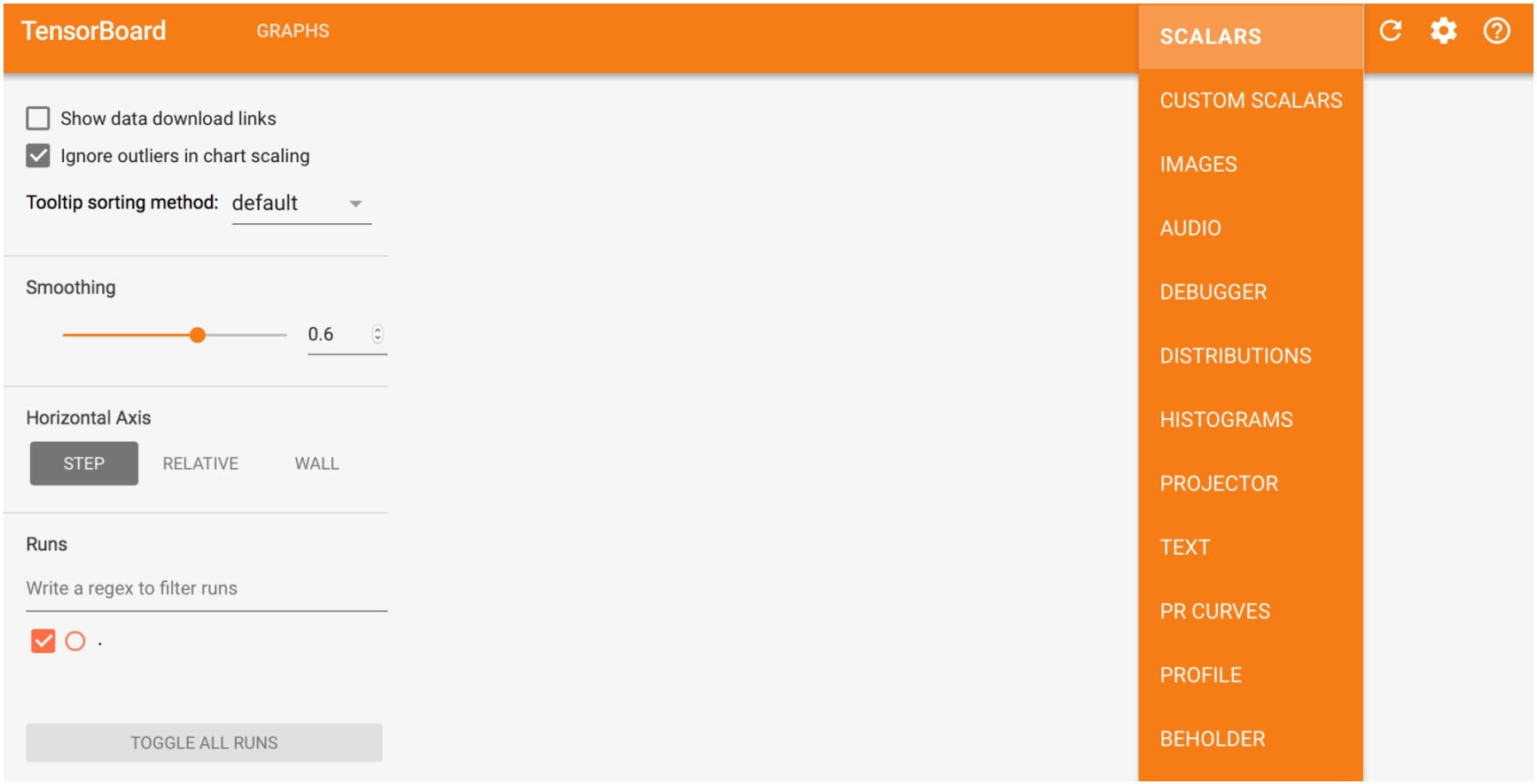
```
import tensorflow as tf

# 定义一个简单的计算图，实现向量加法的操作
with tf.name_scope("input1"):
    input1 = tf.constant([1.0, 2.0, 3.0], name="input1")
with tf.name_scope("input2"):
    input2 = tf.Variable(tf.random_uniform([3]), name="input2")
output = tf.add_n([input1, input2], name="add")

# 生成一个写日志的writer，并将当前的TensorFlow计算图写入日志
writer = tf.summary.FileWriter("D:\\path\\to\\log", tf.get_default_graph())
writer.close()
```

TensorBoard可视化（续）



TensorBoard可视化向量相加程序的TensorFlow计算图结果

TensorBoard可视化（续）



TensorBoard可视化内容选项

内容安排

- TensorFlow环境搭建
- TensorBoard可视化
- **TensorFlow计算模型——计算图**
- TensorFlow数据模型——张量
- TensorFlow运行模型——会话

计算图的使用

TensorFlow程序一般可以分为两个阶段：

1. 定义计算图中所有的计算
2. 执行计算

通过tf.get_default_graph函数可以获得当前默认的计算图

```
import tensorflow as tf

a = tf.constant([1.0,2.0], name="a")
b = tf.constant([3.0,3.0], name="b")
result = a + b

# 通过a.graph查看张量所属的计算图，该图应该等于当前默认的计算图，
# 所以下面这个操作输出值为True
print(a.graph is tf.get_default_graph())
```

计算图的使用（续）

除了使用默认的计算图，TensorFlow支持通过tf.Graph函数来生成新的计算图

不同计算图上的张量和运算都不会共享

```
import tensorflow as tf

g1 = tf.Graph()
with g1.as_default():
    # 在计算图g1中定义变量“v”，并设置初始值为0
    v = tf.get_variable("v", shape=[1], initializer=tf.zeros_initializer)

g2 = tf.Graph()
with g2.as_default():
    # 在计算图g2中定义变量“v”，并设置初始值为1
    v = tf.get_variable("v", shape=[1], initializer=tf.ones_initializer)
```

计算图的使用（续）

```
# 在计算图g1中读取变量“v”的取值
with tf.Session(graph=g1) as sess:
    tf.initialize_all_variables().run()
    with tf.variable_scope("", reuse=True):
        # 在计算图g1中，变量“v”的取值应该为0，所以下面这行会输出[0.]
        print(sess.run(tf.get_variable("v")))

# 在计算图g2中读取变量“v”的取值
with tf.Session(graph=g2) as sess:
    tf.initialize_all_variables().run()
    with tf.variable_scope(“”, reuse=True):
        # 在计算图g2中，变量“v”的取值应该为1，所以下面这行会输出[1.]
        print(sess.run(tf.get_variable("v")))

writer = tf.summary.FileWriter("/path/to/log", g1)
writer = tf.summary.FileWriter("/path/to/log", g2)
writer.close()
```

内容安排

- TensorFlow环境搭建
- TensorBoard可视化
- TensorFlow计算模型——计算图
- **TensorFlow数据模型——张量**
- TensorFlow运行模型——会话

张量

张量在TensorFlow中的实现不是直接采用数组的形式，它只是对TensorFlow中运算结果的引用

一个张量中主要保存了三个属性：名字、维度和类型

```
import tensorflow as tf

# tf.constant是一个计算，计算的结果是一个张量，保存在变量a中
a = tf.constant([1.0,2.0], name="a")
b = tf.constant([3.0,3.0], name="b")

result = tf.add(a, b, name="add")
print(result)
'''
输出：
Tensor("add:0", shape=(2,), dtype=float32)
'''
```

张量的使用

张量的使用可以总结为两大类：

1. 对中间计算结果的引用

```
# 使用张量记录中间结果
a = tf.constant([1.0,2.0], name="a")
b = tf.constant([3.0,3.0], name="b")
result = a + b

# 直接计算向量的和，可读性差
result = tf.constant([1.0, 2.0], name="a") + tf.constant([2.0, 3.0], name="b")
```

2. 当计算图构造完成之后，用来获得计算结果

```
print(tf.Session().run(result))
```

内容安排

- TensorFlow环境搭建
- TensorBoard可视化
- TensorFlow计算模型——计算图
- TensorFlow数据模型——张量
- **TensorFlow运行模型——会话**

会话

使用TensorFlow中的会话来执行定义好的运算

会话拥有并管理TensorFlow程序运行时的所有资源

使用会话的第一种模式：明确调用会话生成函数和关闭会话函数

```
# 创建一个会话
sess = tf.Session()

#使用创建好的会话来得到关心的运算的结果
sess.run(...)

#关闭会话使得本次运行中使用到的资源可以被释放
sess.close()
```

会话（续）

使用会话的第二种模式：通过Python的上下文管理器来使用会话

```
# 创建一个会话，并通过Python中的上下文管理器来管理这个会话
with tf.Session() as sess:
    #使用创建好的会话来得到关心的运算的结果
    sess.run(...)

#不需要再调用"Session.close()"函数来关闭会话
# 当上下文退出时会话关闭和资源释放也自动完成了
```

会话（续）

TensorFlow不会自动生成默认的会话，而是需要手动指定

当默认的会话被指定之后，可以通过tf.Tensor.eval函数来计算一个张量的取值

```
sess = tf.Session()
with sess.as_default():
    print(result.eval())
```

下面代码也可以完成相同的功能

```
sess = tf.Session()

#以下两个命令有相同的功能
print(sess.run(result))
print(result.eval(session=sess))
```

会话（续）

在交互式环境下（比如Python脚本或者Jupyter的编辑器），通过设置默认会话的方式来获取张量的取值更加方便

TensorFlow提供了一种在交互式环境下直接构造默认会话的函数：tf.InteractiveSession

```
sess = tf.InteractiveSession()
print(result.eval())
sess.close()
```