深度学习工具及实践

计算机学院并行与分布处理国家重点实验室

主要内容

- 讲授内容
  - Python 基础
  - Tensorflow基础：概念与编程模型
- 要求
  - 理解Python编程语言与Tensorflow相关概念、框架
  - 能够使用Tensorflow编写简单的计算模型并进行计算

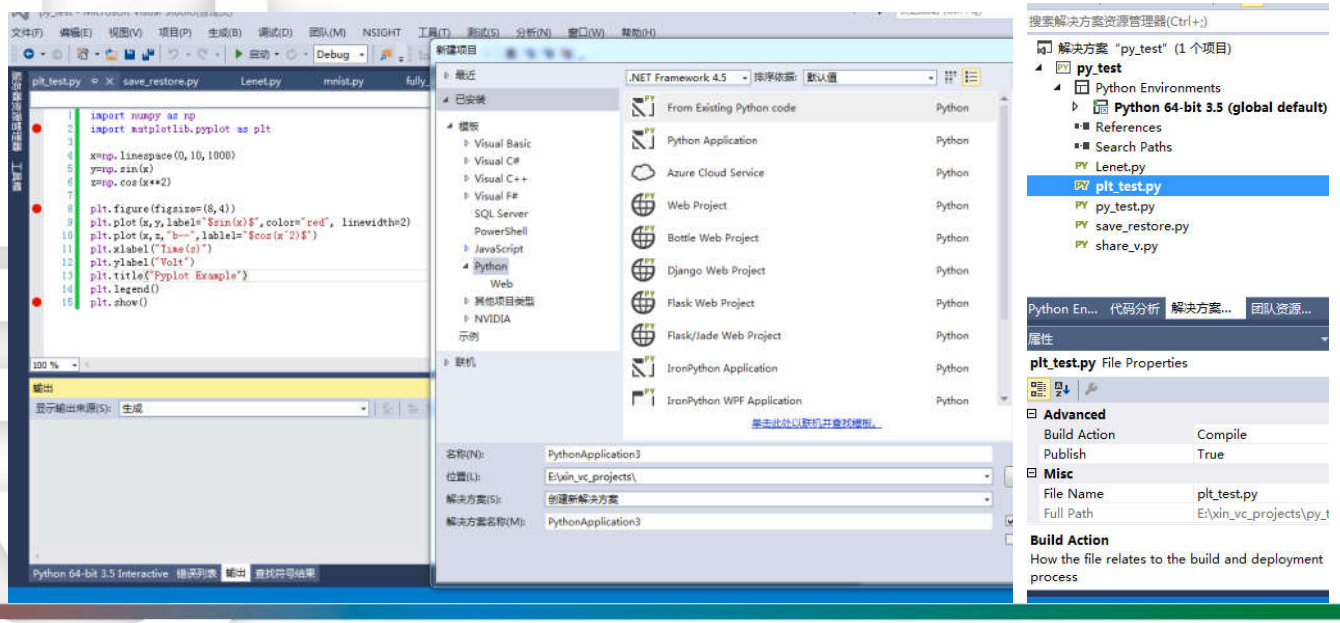
PYTHON 基础

Python基础

- Python特性
  - 开源的、解释性，面向对象编程语言
  - 优雅的语法，可读性强
  - 丰富的库，可处理各种工作
  - 支持类和多层继承等面向对象编程技术
  - 跨平台，如Unix，Windows，MacOS等等

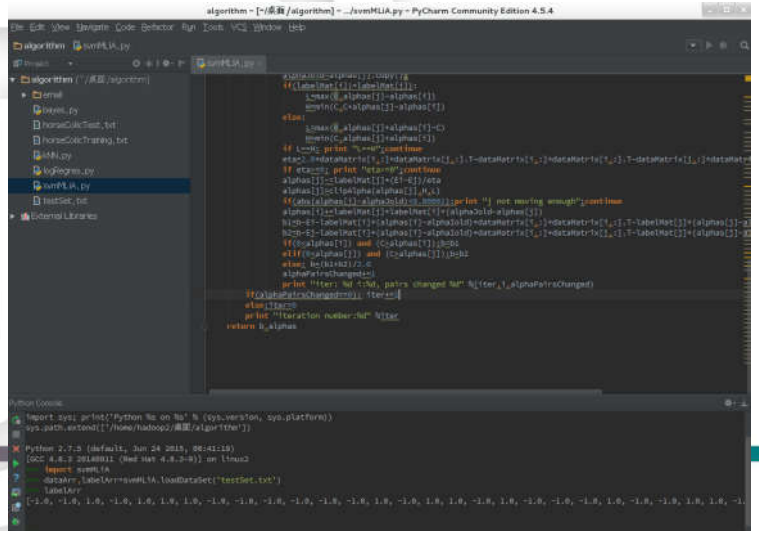
Python 基础

- 集成开发环境
  - Visual Studio
    - VS 下集成python开发环境, 目前仅支持Windows
    - 用惯VS的同学，调试、编辑和C++在VS中类似



Python 基础

- 集成开发环境
  - PyCharm
    - JetBrains开发的Python IDE，支持MacOS、Windows、Unix系统
    - 功能：调试、语法高亮、Project管理、代码跳转、自动完成、单元测试、版本控制…



NumPy

- NumPy: numerical python的简称，是python科学计算的基础包
  - 快速高效的多维数组对象ndarray
  - 用于对数组执行元素级计算及直接对数组执行数学运算的函数
  - 用于读写磁盘上基于数组的数据集工具
  - 线性代数运算、傅里叶变换、随机数生成
  - 将C、C++、Fortran代码集成到Python工具
  - 算法之间传递数据的容器

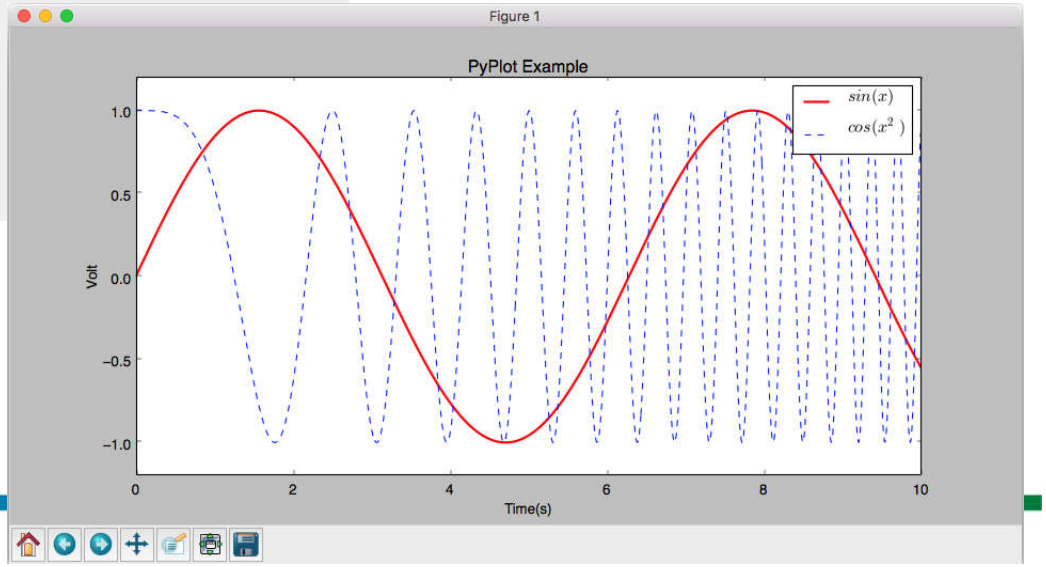
Matplotlib

- 流行的数据图表绘制Python库，实现数据可视化

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x)
z = np.cos(x**2)

plt.figure(figsize=(8,4))
plt.plot(x, y, label="$sin(x)$", color="red", linewidth=2)
plt.plot(x, z, "b--", label="$cos(x^2)$")
plt.xlabel("Time(s)")
plt.ylabel("Volt")
plt.title("PyPlot Example")
plt.ylim(-1.2, 1.2)
plt.legend()
plt.show()
```



其他Python库

- Pandas
  - 提供快捷处理结构化数据的大量数据结构和函数
  - 兼具NumPy的高性能数据计算功能以及电子表格和关系型数据库的灵活数据处理功能
  - 提供大量适用于金融数据的高性能时间序列分析功能和工具
- SciPy
  - 专门解决科学计算中各种标准问题域的包的集合
  - 数值积分、矩阵分解、信号处理、稀疏矩阵和线性系统求解器、统计工具等

第三方Python库的安装和导入

- 安装
  - 下载Anaconda，它附带了预安装的库
    - Anaconda是完全免费、跨平台、企业级的Python发行大规模数据处理、预测、分析和科学计算工具
  - Pycharm集成了NumPy、Pandas、Matplotlib等常用库
- 导入
  - import numpy as np
  - from socket import gethostname, socket

TENSORFLOW概念与编程模型

TensorFlow简介

- 开源的基于数据流图的数学计算库
- Google brain开发用来做机器学习、深度学习研究的工具
- 多平台：支持CPU/GPU，服务器、个人电脑、移动设备
- 分布并行：方便多GPU，分布式训练
- 可扩展：支持OP扩展，kernal扩展
- 接口丰富：支持python, java以及c++接口
- 可视化：使用TensorBoard可视化计算图
- 易用性：相比Caffe等易于学习掌握，文档资料丰富
- 社区支持：开源项目支持最多的几种框架：Tensorflow、caffe、pyTorch

TensorFlow易于学习

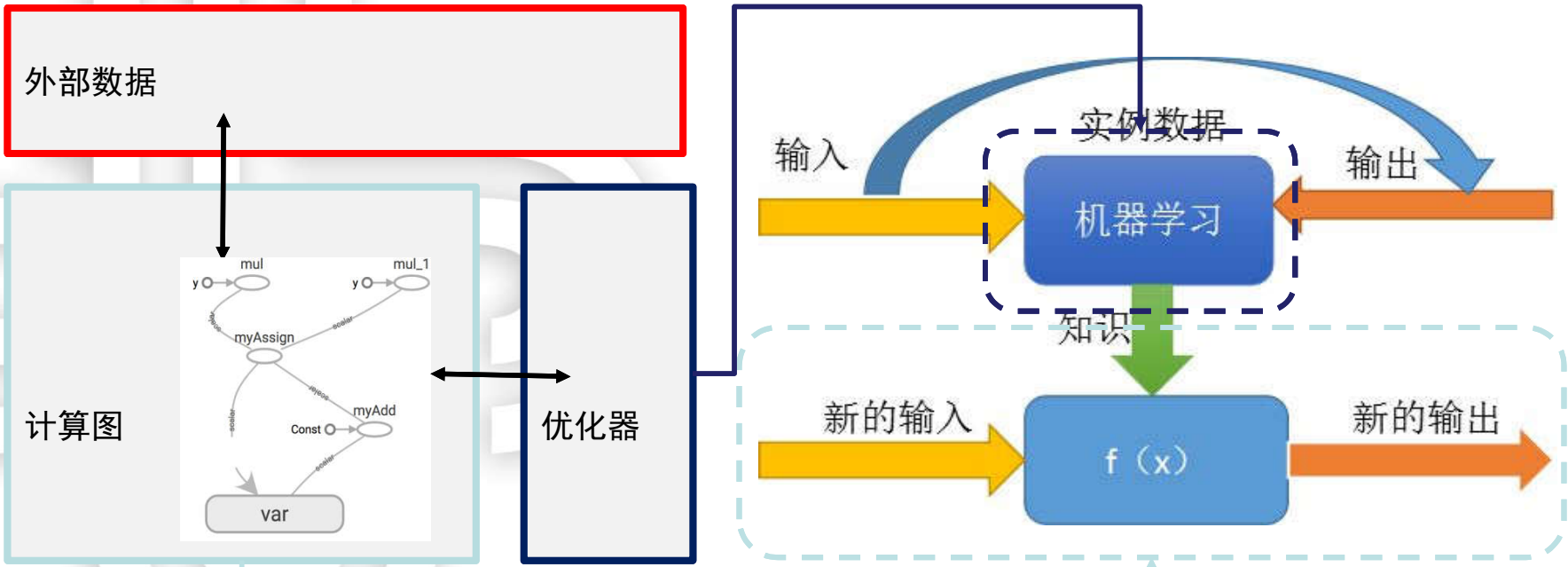
- 易学：
  - Python 接口：适于非计算机专业学习编程
  - 安装方便: Anaconda+pip install
  - 多平台：windows, linux
  - 模型设计方便：
    - 1. 自动求导，方便自定义层 (不用自己写求导公式)
    - 2. 不用算每层的参数维度，自动计算
  - 多GPU, 分布式训练支持方便：简单一两个命令就可以并行训练

Tensorflow 和 Python

- 简单理解：Tensorflow（TF）就是Python中调用的一个库
- 怎样熟练TF库?和其他库一样
  - 数据结构：TF定义数据如何和Python中其他数据进行交互
  - 算法思想：TF进行计算的编程思想、编程模型
  - 熟练TF库中常用的函数、工具
- 调用TF库和Numpy库进行计算对比举例：

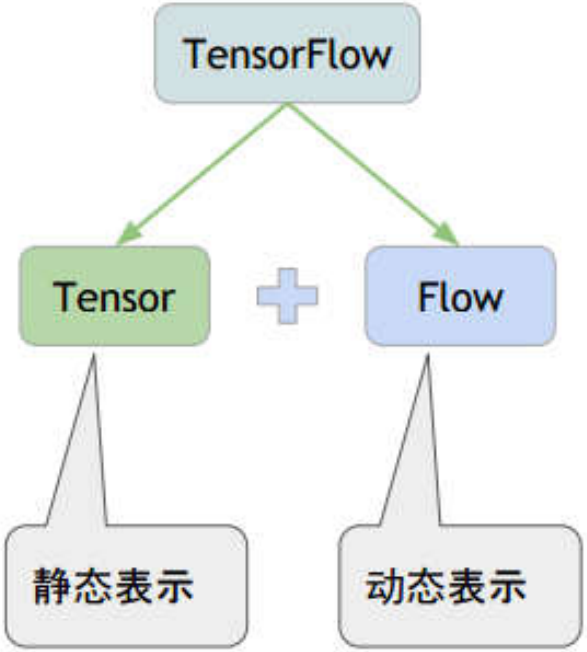
Tensorflow 编程框架和机器学习模型对应关系

- 输入、输出、模型计算过程用计算图Graph描述，并用优化器和训练数据对模型参数进行优化
- 模型设计围绕Graph展开



TensorFlow 基本原理

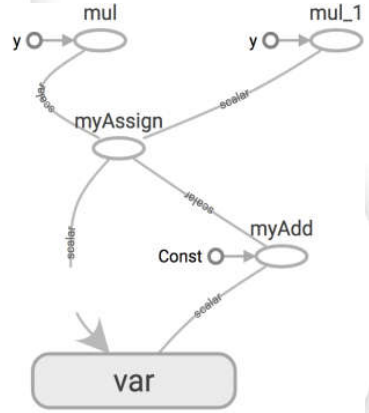
- TensorFlow=Tensor+flow
- Tensor:张量
  - 张量数据，流动（flow）的数据
  - Tensor在哪里flow？
- Graph:计算图
  - Tensor 流动的路径图
  - Graphs定义了模型和计算任务
  - Tensor在图里怎么flow起来？
- Session:会话
  - 管理计算资源，驱动Tensor 在Graph里面转起来
  - 怎么取数据？怎么取结果？怎么训练？



编程模型

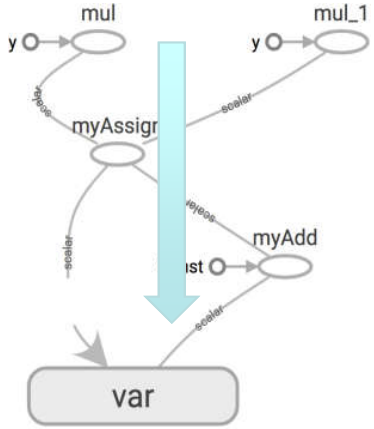
- 编程模型：
  - 1.画个图——定义模型计算图（Graph）
  - 2.写个执行剧本——定义会话（Session），设计并执行计算过程

1. def Graph()=



2. Sess:

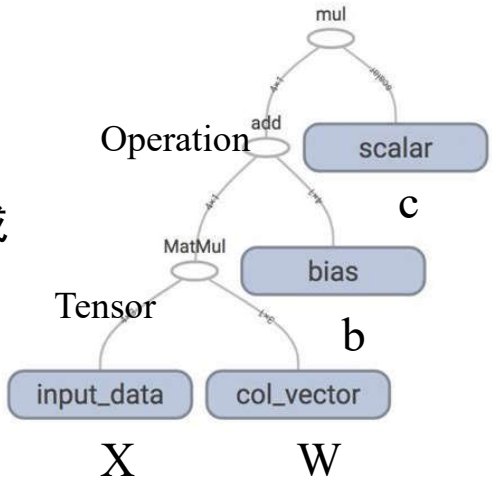
1. 取数据
2. 放数据入图
3. run()
4. 取结果

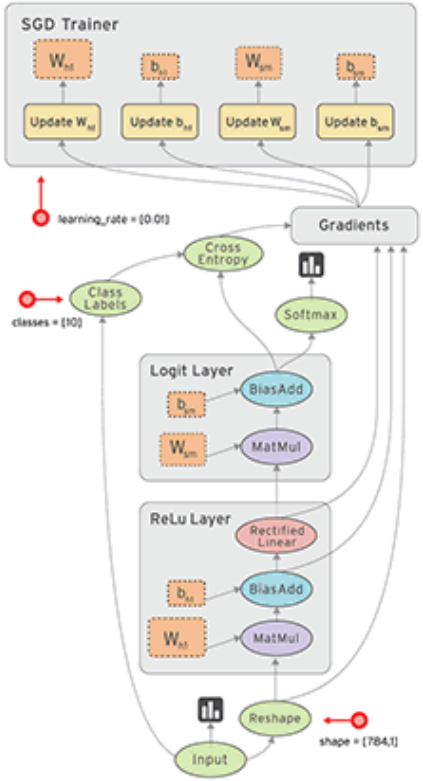


- Graph怎么定义？Session 怎么写？

计算图定义

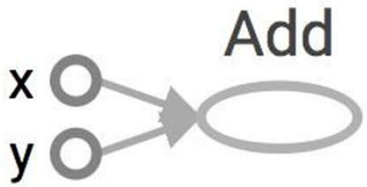
- Graph:描述数学计算的有向图(有向无环图)
  - 结构：
    - 节点: 算子(Operation)：+-/…
    - 连接节点的边:Operation的输出或者称作(Tensor)
    - 边缘点：（数据）输入和（参数）变量
  - 计算过程：
    - 输入沿计算路径（有向）逐个结点激活
    - 每个节点的激活需要所有前驱节点激活
- 举例：（（$W*X$）$+b$）*c





Graph的边-张量

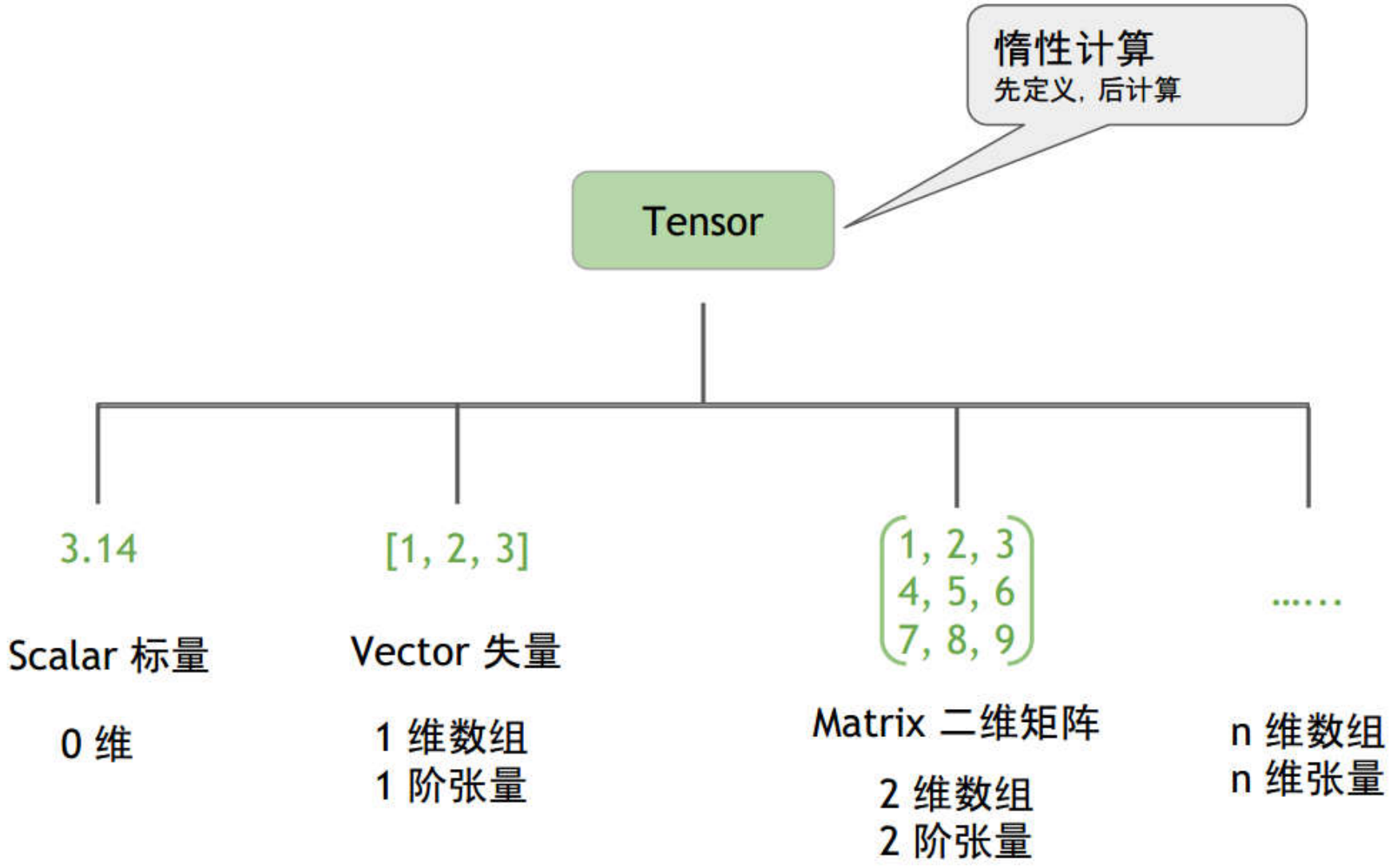
- 张量（Tensor）——Graph节点(Operation)之间传递的数据都为可以看作n维的数组
  - 0维张量: 标量（数）
  - 1维张量: 向量
  - 2维向量：矩阵
  - n维向量等等




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

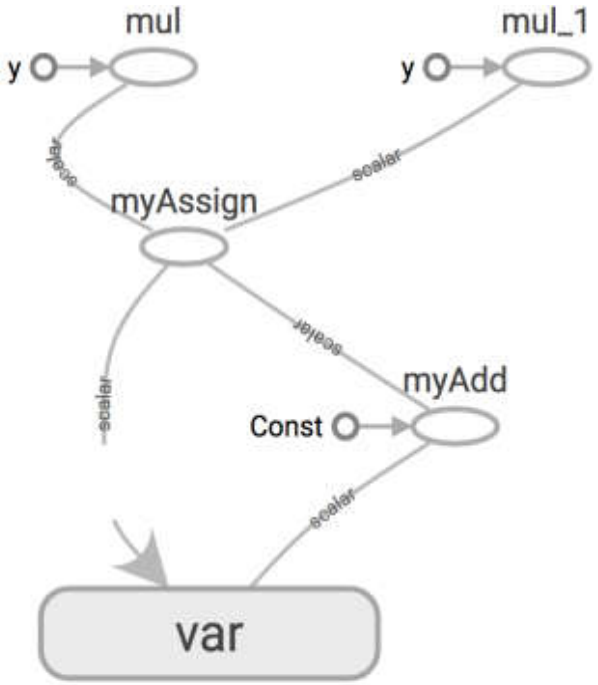
- 张量Tensor——算子Operation的输出
  - 引用中间计算结果

多维张量举例


惰性计算
先定义，后计算
Tensor
3.14
Scalar 标量
0 维
[1, 2, 3]
Vector 失量
1 维数组
1 阶张量
1, 2, 3
4, 5, 6
7, 8, 9
Matrix 二维矩阵
2 维数组
2 阶张量
.....
n 维数组
n 维张量

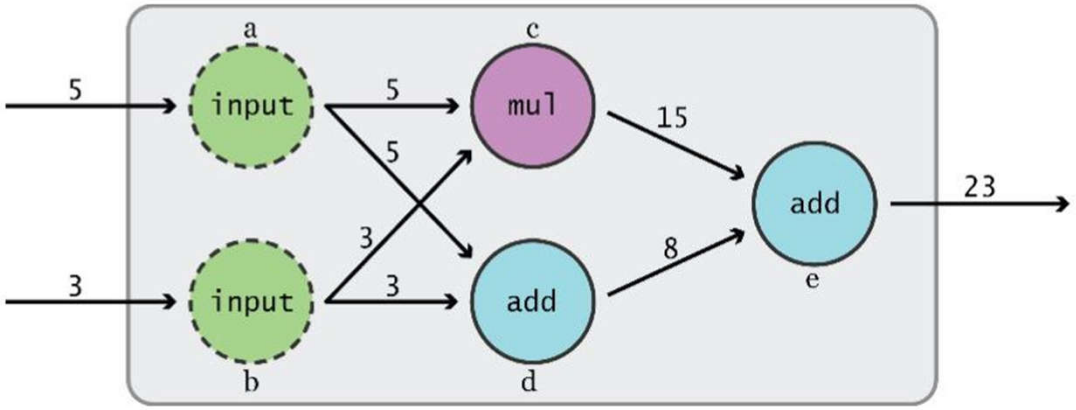
Graph终端节点：输入、模型参数

- tf.Variable。变量节点
  - 用来存在图执行过程中需要更新的量，
  - 在神经网络中用来存储权值
- tf.constant。常量节点
  - 在建图时值已经确定，所以要传一个python 值，而不能传一个tensor
- tf.placeholder。占位节点
  - 在运行时要给它喂/feed 一个值
- tf.zeros, tf.ones, tf.zeros_like, tf.ones_like, tf.random , ...

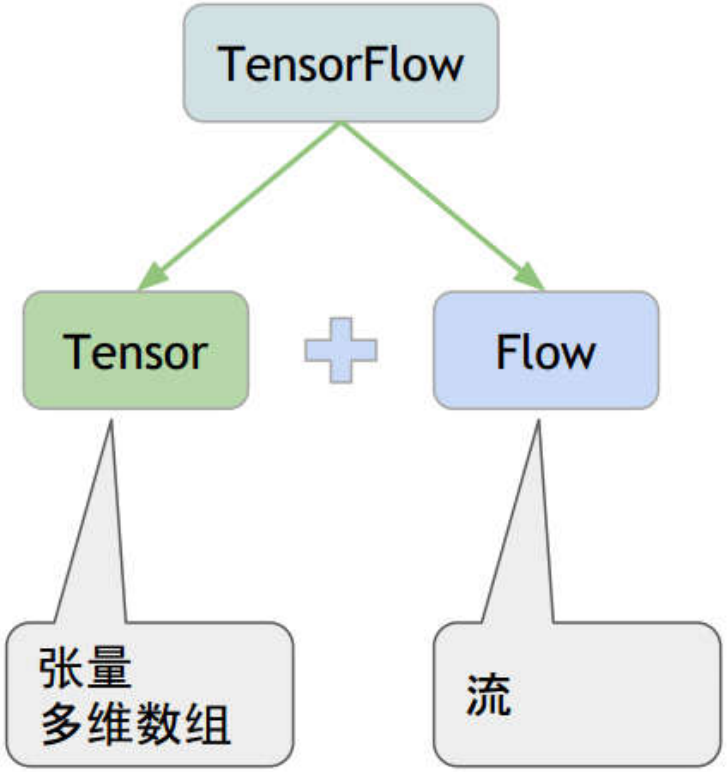


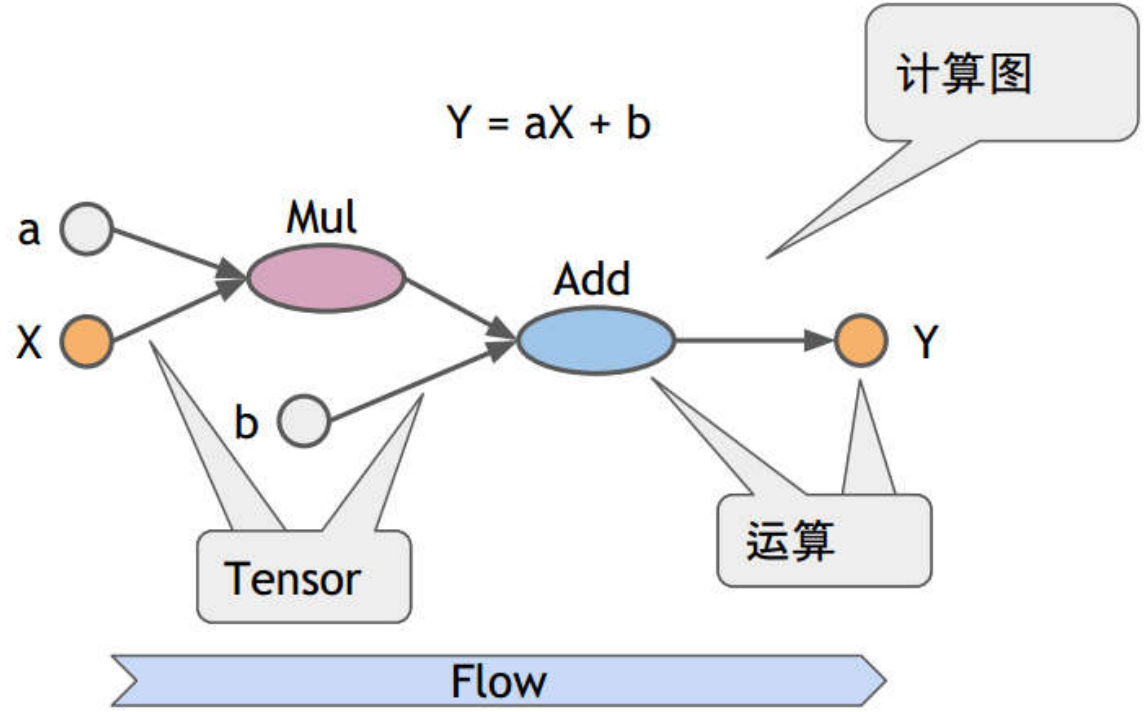
Graph=Operation+Tensor +终端节点

- Graph = Set{Operation} + Set{Tensor}+Set{终端节点}
  - 图中间的节点：定义算子（Operation）
  - 节点可以拥有零条或多条边：多输入，多输出
  - 每条边表示节点的输入/输出，并以张量（Tensor）传递数据



计算图（Graph）


TensorFlow
Tensor
Flow
张量
多维数组
流


Y = aX + b
计算图
a
X
b
Mul
Add
Y
Tensor
运算
Flow

TF中Graph构建方式

- 1. 像写函数一样，使用tf中的数据结构和算子（tf.xxx）直接描述 (整个描述成为模型默认图)
- 2. 可以指定多个图， 并分别定义

```
a=tf.constant(2,name='a')
print(a.name, a)
b=tf.constant(3,name='b')
print(b.name, b)
x=tf.add(a,b,name='add')
print (x.name,x)
```

```
g = tf.Graph()
with g.as_default():
    a = tf.constant(2,name='a')
    print(a.name,a)
    b = tf.constant(3,name='b')
    print(b.name,b)
    x = tf.add(a,b,name='add')
    print(x.name,x)
```

Graph 惰性计算

- Graph 构图与pytho函数计算区别：
  - 1. 使用tf中的数据结构和算子（tf.xxx），把整个模型的所有连接都写一遍
  - 2. 写一遍：像函数定义不行，还得像函数运行一样执行一遍
  - 3. 执行结果只是创建了Graph，就算终端节点都给了定值了，运行一遍也不像python其他函数一样出结果
  - 4. 要得计算结果，必须在Session中用特定方式启动

```
17 import tensorflow as tf
18 #定义两个常量向量a b
19 def graph_a():
20     a = tf.constant([1.0,2.0],name="a")
21     b = tf.constant([2.0,3.0],name="b")
22     #将两个向量加起来
23     result = a+b
24 #生成一个会话
25 sess = tf.Session()
26 #通过会话来计算结果
27 xxx = sess.run(result)
```

```
1 import tensorflow as tf
2 #定义两个常量向量a b
3 a = tf.constant([1.0,2.0],name="a")
4 b = tf.constant([2.0,3.0],name="b")
5 #将两个向量加起来
6 result = a+b
```

运行模型-会话Session

- Session:
  - 执行Graph计算
  - 拥有并管理Tensorflow程序运行时的所有资源
  - 资源：硬件（CPU,GPU），数据
- Tensorflow使用会话的模式有两种:
  - **1. 明确调用会话生成函数和关闭会话函数**

```
#创建会话
sess = tf.Session()
#运行,得到张量中的结果
sess.run(...)
#关闭会话,释放资源
sess.close()
```

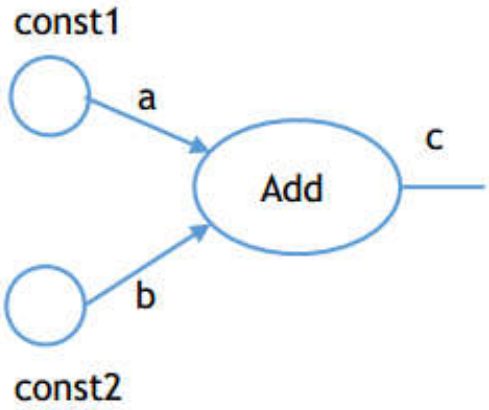
运行模型-会话

- Tensorflow使用会话的模式有两种:
  - **2. 通过Python的上下文管理器来使用会话**

```
#创建会话并通过Python的上下文管理器来管理会话
with tf.Session() as sess:
    #使用这个创建好的会话来计算关系的结果
    sess.run()
#不需要调用Session.close()函数来关闭会话,上下文结束时资源会自动的释放.
```

TensorFlow编程基本流程

- 画个图（Graph）+执行剧本（Session）




```
import tensorflow as tf
import numpy as np

a = tf.constant(1., name='const1')
b = tf.constant(2., name='const2')
c = tf.add(a, b)
with tf.Session() as sess:
    print sess.run(c)
    print c.eval()
```

示例：随机数生成

```python
import numpy as np
aa = np.random.rand(1)

for i in range(5):
    print aa
```

```
[ 0.12619646]
[ 0.12619646]
[ 0.12619646]
[ 0.12619646]
[ 0.12619646]
```

```python
import tensorflow as tf
import numpy as np

a = tf.random_normal([1], name='random')
with tf.Session() as sess:
    for i in range(5):
        print sess.run(a)
```

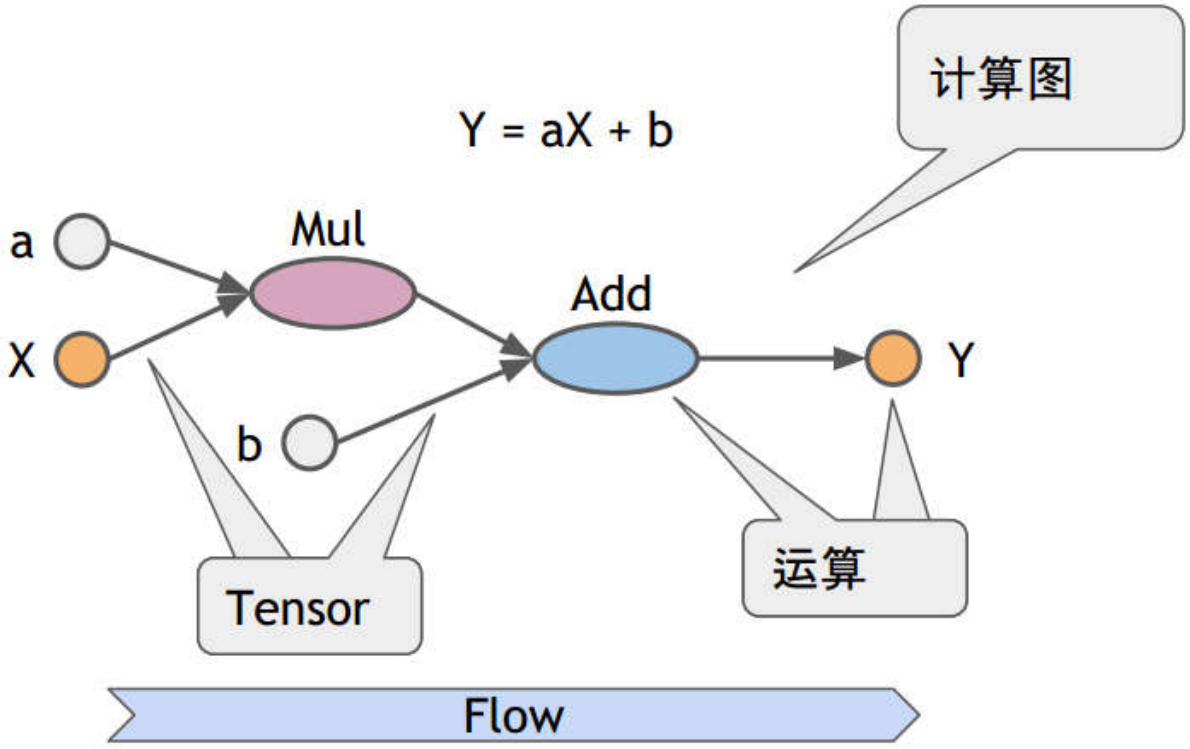
```
[ 0.38659823]
[-0.6114915]
[ 0.22402158]
[ 0.48937175]
[-0.63917536]
```

输入输出：Feed & Fetch

- feed和fetch是python与TensorFlow数据交流的途径
  - Feed，将数据喂进Tensorflow实例图(给placeholder节点)
  - Fetch，Tensorflow实例图中取数据
    - 取哪个节点输出的值，就只计算计算图哪个部分
    - Fetch一下，计算图就算一次，图里的tensor就更新一次
  - 怎么喂？怎么拿？
    - 先定义一个输入接口a=tf.placeholder()
    - 在运行中告诉要拿什么？拿什么喂？
    - Sess.run ([c, …], feed_dict={a: xxxx, ….})

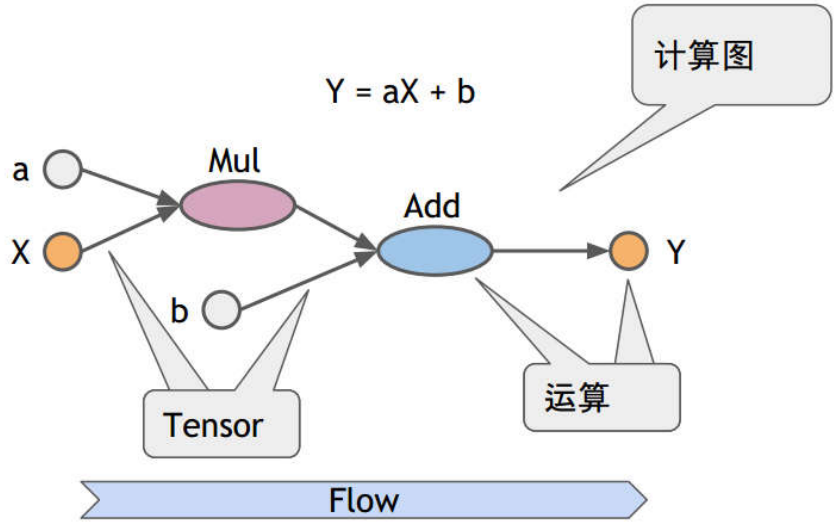
示例：

- Y=aX+b



示例：

- Y=aX+b




```
a = tf.placeholder(tf.int16)
x = tf.placeholder(tf.int16)

mul = tf.multiply(a, x)
y = tf.add(mul, b)

# Launch the default graph.
with tf.Session() as sess:
    # Run every operation with variable input
    print "Addition with variables: %i" % sess.run(y, feed_dict={a: 2, b: 3})
    print "Multiplication with variables: %i" % sess.run(mul, feed_dict={a: 2, b: 3})
```

总结

- 1. Tensorflow是Python接口的深度学习计算库
- 2. TF采用计算图描述模型，并用会话运行计算实例
- 3. TF基本编程模式：计算图+会话