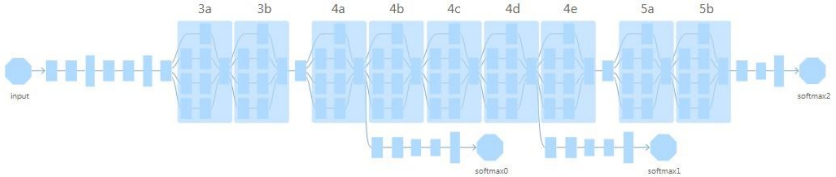
Google TensorFlow™

浙江大学城市学院 ZHEJIANG UNIVERSITY CITY COLLEGE



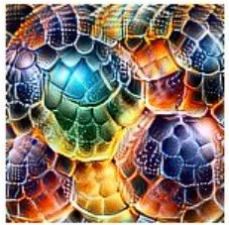
深度学习应用开发
基于TensorFlow的实践

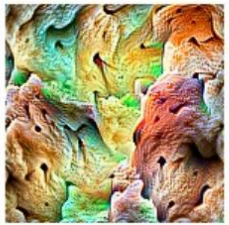
吴明晖 李卓蓉 金苍宏

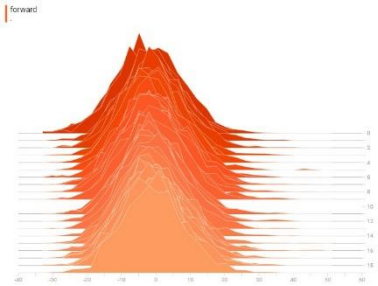
浙江大学城市学院

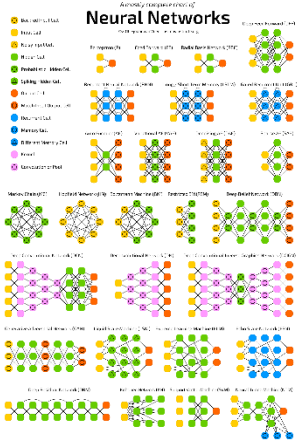
计算机与计算科学学院

Dept. of Computer Science
Zhejiang University City College









TensorFlow编程基础

浙江大学城市学院
ZHEJIANG UNIVERSITY CITY COLLEGE

A machine learning platform

for everyone

to solve real problems

TensorFlow

- TensorFlow™ 是一个开放源代码软件库，用于进行高性能数值计算
- 借助其灵活的架构，用户可以轻松地将计算工作部署到多种平台（CPU、GPU、TPU）和设备（桌面设备、服务器集群、移动设备、边缘设备等）
- TensorFlow™ 最初是由 Google Brain 团队（隶属于 Google 的 AI 部门）中的研究人员和工程师开发的，可为机器学习和深度学习提供强力支持

TensorFlow的Hello World

In [1]:
```
import tensorflow as tf

# 创建一个常值运算，将作为一个节点加入到默认计算图中
hello = tf.constant("Hello, World!")

# 创建一个TF对话
sess = tf.Session()

# 运行并获得结果
print(sess.run(hello))
```

```
b'Hello, World!'
```

输出前面的’b’ 表示Bytes literals（字节文字）

TensorFlow计算模型 - 计算图

TensorFlow的概念

TensorFlow = Tensor + Flow

Tensor 张量

数据结构：多维数组

Flow 流

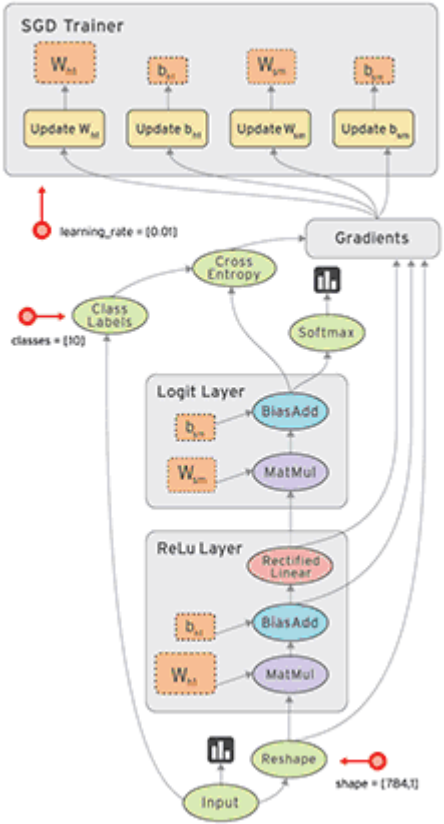
计算模型：张量之间通过计算而转换的过程

TensorFlow是一个通过**计算图**的形式表述计算的编程系统

每一个计算都是计算图上的一个节点

节点之间的边描述了计算之间的关系

计算图（数据流图）的概念



计算图是一个有向图，由以下内容构成：

- 一组节点，每个**节点**都代表一个**操作**，是一种**运算**
- 一组有向边，每条**边**代表节点之间的**关系（数据传递**和**控制依赖）**

TensorFlow有两种边：

- **常规边（实线）**：代表数据依赖关系。一个节点的运算输出成为另一个节点的输入，两个节点之间有tensor流动（**值传递**）
- **特殊边（虚线）**：不携带值，表示两个节点之间的**控制**相关性。比如，**happens-before关系**，源节点必须在目的节点执行前完成执行

计算图的实例

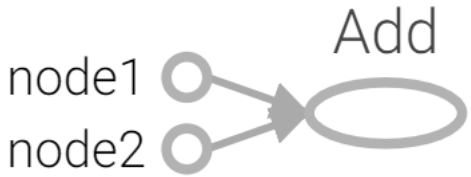
浙江大学城市学院 ZHEJIANG UNIVERSITY CITY COLLEGE

In [2]:
```
# 一个简单计算图
node1 = tf.constant(3.0,tf.float32,name="node1")
node2 = tf.constant(4.0,tf.float32,name="node2")
node3 = tf.add(node1, node2)
```

In [3]:
```
print(node3)
```

```
Tensor("Add:0", shape=(), dtype=float32)
```

输出的结果不是一个具体的数字，而是一个张量的结构



node1
Operation: Const

Attributes (2)

dtype	{"type":"DT_FLOAT"}
value	{"tensor": {"dtype":"DT_FLOAT","tensor_shape":{},"float_val":3}}

Inputs (0)

Outputs (0)

Remove from main graph

计算图的执行

- 创建计算图只是建立静态计算模型
- 执行对话才能提供数据并获得结果

In [5]:
```
# 建立对话并显示运行结果
sess = tf.Session()
print("运行sess.run(node1)的结果：", sess.run(node1))
```

```
运行sess.run(node1)的结果：  3.0
```

In [6]:
```
# 更新变量并返回计算结果
print("运行sess.run(node3)的结果：", sess.run(node3))

# 关闭session
sess.close()
```

```
运行sess.run(node3)的结果：  7.0
```

Tensor 张量

张量的概念

```
In [3]: print(node3)
```

```
Tensor("Add:0", shape=(), dtype=float32)
```

- 在TensorFlow中，所有的数据都通过张量的形式来表示
- 从功能的角度，张量可以简单理解为多维数组

  零阶张量表示标量（scalar），也就是一个数；

  一阶张量为向量（vector），也就是一维数组；

  n阶张量可以理解为一个n维数组；
- 张量并没有真正保存数字，它保存的是计算过程

张量的属性

Tensor("Add:0", shape=(), dtype=float32)

名字（name）

“node:src_output”：node 节点名称，src_output 来自节点的第几个输出

形状（shape）

张量的维度信息， shape=() ，表示是标量

类型（type）

每一个张量会有一个唯一的类型

TensorFlow会对参与运算的所有张量进行类型的检查，发现类型不匹配时会报错

张量的形状

三个术语描述张量的维度：**阶**（rank）、**形状**（shape） 、**维数**（dimension number）

阶	形状	维数	例子
0	()	0-D	4
1	(D0)	1-D	[2,3,5]
2	(D0,D1)	2-D	[[2,3],[3,4]]
3	(D0,D1,D2)	3-D	[[[7],[3]],[[2],[4]]]
N	(D0,D1,…,Dn-1)	n-D	形为(D0,D1,…,Dn-1)的张量

张量的形状

In [11]:

```
import tensorflow as tf
tens1 = tf.constant([[[1,2,2],[2,2,3]],
                     [[3,5,6],[5,4,3]],
                     [[7,0,1],[9,1,9]],
                     [[11,12,7],[1,3,14]]],name="tens1")

#语句中包含 [], {} 或 () 括号中间换行的就不需要使用多行连接符

print(tens1)
```

```
Tensor("tens1_2:0", shape=(4, 2, 3), dtype=int32)
```

张量的形状

In [12]:
```
import tensorflow as tf

scalar = tf.constant(100)
vector = tf.constant([1, 2, 3, 4, 5])
matrix = tf.constant([[1, 2, 3], [4, 5, 6]])
cube_matrix = tf.constant([[[1], [2], [3]], [[4], [5], [6]], [[7], [8], [9]]])

print(scalar.get_shape())
print(vector.get_shape())
print(matrix.get_shape())
print(cube_matrix.get_shape())
```

```
()
(5,)
(2, 3)
(3, 3, 1)
```

张量的阶

张量的阶（rank）表征了张量的维度

阶	数学实体	代码示例
0	Scalar	Scalar = 1000
1	Vector	Vector = [2,8,3]
2	Matrix	Matrix = [[4,2,1],[5,3,2],[5,5,6]]
3	3-tensor	Tensor = [[[4],[3],[2]],[[6],[100],[4]],[[5],[1],[4]]]
n	N-tensor	…

获取张量的元素

阶为1的张量等价于向量；

阶为2的张量等价于矩阵，通过 *t[i, j]* 获取元素；

阶为3的张量，通过 *t[i, j, k]* 获取元素；

例：

In [13]:
```
import tensorflow as tf
tens1 = tf.constant([[[1,2],[2,3]],[[3,4],[5,6]]])

sess = tf.Session()
print(sess.run(tens1)[1,1,0])
sess.close()
```

```
5
```

下标从 0 开始

张量的类型

TensorFlow支持14种不同的类型

实数 tf.float32, tf.float64

整数 tf.int8, tf.int16, tf.int32, tf.int64, tf.uint8

布尔 tf.bool

复数 tf.complex64, tf.complex128

默认类型：

不带小数点的数会被默认为int32

带小数点的会被默认为float32

张量的类型

```
In [14]: import tensorflow as tf
         a = tf.constant([1, 2], name="a")
         b = tf.constant([2.0, 3.0], name="b")
         result = a + b
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-ca86e03340c0> in <module>()
```

运行报错：

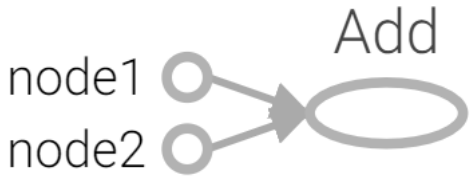
ValueError: Tensor conversion requested dtype int32 for Tensor with dtype float32: 'Tensor("b_3:0", shape=(2,), dtype=float32)'

TensorFlow会对参与运算的所有张量进行类型的检查，发现类型不匹配时会报错

Operation 操作

操作

- 计算图中的**节点**就是**操作**（Operation）
  - 一次加法是一个操作
  - 一次乘法也是一个操作
  - 构建一些变量的初始值也是一个操作
- 每个运算操作都有**属性**，它在构建图的时候需要确定下来
- 操作可以和计算**设备绑定**，指定操作在某个设备上执行
- 操作之间存在顺序关系，这些操作之间的**依赖**就是“**边**”
- 如果操作A的输入是操作B执行的结果，那么这个操作A就依赖于操作B



计算图中的操作

```
import tensorflow as tf

# 本例用到了TensorBoard，具体使用后面讲解

tf.reset_default_graph() #清除default graph和不断增加的节点

# 定义变量 a
a = tf.Variable(1, name="a")
# 定义操作b为a+1
b = tf.add(a, 1, name="b")
# 定义操作c为b*4
c = tf.multiply(b, 4, name="c")
# 定义 d为c-b
d = tf.subtract(c, b, name="d")

# logdir改为自己机器上的合适路径
logdir='D:/log'

#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(logdir,tf.get_default_graph())
writer.close()
```