

TensorFlow 基本使用

2016-04-05

DeepMilan

阅 262

分享：

微信

▼

转藏到我的图书馆

使用 TensorFlow ，你必须明白 TensorFlow ：

使用图（graph）来表示任务

被称之为会话（Session）的上下文（context）中执行图

使用 tensor 表示数据

通过变量（Variable）维护状态

使用 feed 和 fetch 可以为任意操作（arbitrary operation）赋值或者从其中获取数据

综述

TensorFlow 是一个编程系统，使用图来表示计算任务，图中的节点被称之为 op（operation 的缩写），一个 op 获得0个或多个 tensor，执行计算，产生0个或多个 tensor。每个 tensor 是一个类型化的多维数组。例如，你可以将一组图像素集表示为一个四维浮点数数组，这四个维度分别是 [batch, height, width, channels]。

一个 TensorFlow 图描述了计算的过程，为了进行计算，图必须在 会话 里被启动，会话 将图的 op 分发到诸如CPU或GPU之类的设备上，同时提供执行 op 的方法，这些方法执行后，将产生的 tensor 返回。在python语言中，返回的 tensor 是 numpy ndarray 对象；在C/C++语言中，返回的是 tensorflow::Tensor 实例。

计算图

Tensorflow 程序通常被组织成一个构建阶段和一个执行阶段，在构建阶段，op 的执行步骤被描述成为一个图，在执行阶段，使用会话执行图中的 op。

例如，通常在构建阶段创建一个图来表示和训练神经网络，然后在执行阶段反复执行图中的训练 op。

Tensorflow 支持C/C++，python编程语言。目前，TensorFlow 的python库更易使用，它提供了大量的辅助函数来简化构建图的工作，这些函数尚未被C/C++库支持。

三种语言的会话库（session libraries）是一致的。

构建图

构件图的第一步是创建源 op（source op）。源 op 不需要任何输入。源 op 的输出被传递给其它 op 做运算。

python库中，op 构造器的返回值代表被构造出的 op 输出，这些返回值可以传递给其它 op 作为输入。

TensorFlow Python库中有一个默认图（default graph），op 构造器可以为其增加节点。这个默认图对许多程序来说已经足够用了，可以阅读 Graph 类文档，来了解如何管理多个视图。

```
1 import tensorflow as tf
2 # 创建一个常量op，产生一个1x2矩阵，这个op被作为一个节点
3 # 加到默认视图中
4 # 构造器的返回值代表该常量op的返回值
5 matrix1 = tr.constant([[3., 3.]])
6
7 # 创建另一个常量op,产生一个2x1的矩阵
8 matrix2 = tr.constant([[2.], [2.]])
9
```

TA

永远成

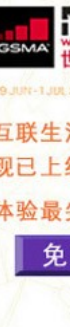
我们将

我们将

[转] 赞

他们还

基督徒



推荐阅读

BetaC

揪出b

深度学

简易的

国外公

enum

再谈：

是还没

帧缓存



- 1 美亚保险
- 2 美亚保险
- 3 公司邮箱
- 4 中老年妈
- 5 企业邮箱
- 6 led亮化照

```
10
11 # 创建一个矩阵乘法matmul op，把matrix1和matrix2作为输入：
product = tf.matmul(matrix1, matrix2)
```

默认图现在有三个节点，两个 `constant()` op 和 `matmul()` op。为了真正进行矩阵乘法的结果，你必须在会话里启动这个图。

在一个会话中启动图

构造阶段完成后，才能启动图。启动图的第一步是创建一个 `Session` 对象，如果无任何创建参数，会话构造器将无法启动默认图。

欲了解完整的会话API，请阅读 [Session 类](#)。

```
1 # 启动默认图
2 sess = tf.Session()
3
4 # 调用sess的'run()'方法来执行矩阵乘法op，传入'product'作为该方法的参数
5 # 上面提到，'product'代表了矩阵乘法op的输出，传入它是向方法表明，我们希望取回
6 # 矩阵乘法op的输出。
7 #
8 # 整个执行过程是自动化的，会话负责传递op所需的全部输入。op通常是并发执行的。
9 #
10 # 函数调用'run(product)'触发了图中三个op（两个常量op和一个矩阵乘法op）的执行。
11 # 返回值'result'是一个numpy 'ndarray'对象。
12 result = sess.run(product)
13 print result
14 # ==>[[12.]]
15
16 # 完成任务，关闭会话
17 sess.close()
```

`Session` 对象在使用完成后需要关闭以释放资源，除了显式调用 `close` 外，也可以使用 `with` 代码来自动完成关闭动作：

```
1 with tf.Session() as sess:
2     result = sess.run([product])
3     print result
```

在实现上，`Tensorflow` 将图形定义转换成分布式执行的操作，以充分利用可以利用的计算资源（如CPU或GPU）。一般你不需要显式指定使用CPU还是GPU，`Tensorflow` 能自动检测。如果检测到GPU，`Tensorflow` 会尽可能地使用找到的第一个GPU来执行操作。

如果机器上有超过一个可用的GPU，除了第一个外的其他GPU是不参与计算的。为了让 `Tensorflow` 使用这些GPU，你必须将 `op` 明确地指派给它们执行。`with...Device` 语句用来指派特定的CPU或GPU操作：

```
1 with tf.Session() as sess:
2     with tf.device("/gpu:1"):
3         matrix1 = tf.constant([[3., 3.]])
4         matrix2 = tf.constant([[2.], [2.]])
5         product = tf.matmul(matrix1, matrix2)
```

设备用字符串进行标识，目前支持的设备包括：

`/cpu:0` :机器的CPU

`/gpu:0` :机器的第一个GPU，如果有的话

`/gpu:1` :机器的第二个GPU，以此类推

交互式使用

文档中的python示例使用一个会话 `Session` 来启动图，并调用 `Session.run()` 方法执行操作。

为了便于使用诸如IPython之类的python交互环境，可以使用 `InteractiveSession` 代替 `Session` 类，使用 `Tensor.eval()` 和 `Operation.run()` 方法代替 `Session.run()`。这样可以避免使用一个变量来持有会话：

```
1 # 进入一个交互式Tensorflow会话
2 import tensorflow as tf
3 sess = tf.InteractiveSession()
4
```

```
5 x = tf.Variable([1.0, 2.0])
6 a = tf.constant([3.0, 3.0]);
7
8 # 使用初始化器initializer op的run()方法初始化x
9 x.initializer.run()
10
11 # 增加一个减法sub op，从x减去a。运行减法op，输出结果
12 sud = tf.sub(x, a)
13 print sub.eval()
14 # ==>[-2. -1.]
```

Tensor

Tensorflow 程序使用 tensor 数据结构来代表所有的数据，计算图中，操作间传递数据都是 tensor。你可以把 Tensorflow 的 tensor 看做是一个 n 维的数组或列表。一个 tensor 包含一个静态类型 rank 和一个 shape。

阶

在 Tensorflow 系统中，张量的维数被描述为阶。但是张量的阶和矩阵的阶并不是同一个概念。张量的阶是张量维数的一个数量描述，下面的张量（使用python中 list 定义的）就是2阶：

```
1 t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

你可以认为一个二阶张量就是我们平常所说的矩阵，一阶张量可以认为是一个向量。对于一个二阶张量，你可以使用语句 t[i, j] 来访问其中的任何元素。而对于三阶张量你可以通过 t[i, j, k] 来访问任何元素：

阶	数学实例	python例子
0	纯量（只有大小）	s=483
1	向量（大小和方向）	v=[1.1, 2.2, 3.3]
2	矩阵（数据表）	m=[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
3	3 阶张量	t=[[2], [4], [6]], [[8], [9], [10]], [[11], [12], [13]]]
n	n 阶	

形状

Tensorflow 文档中使用了三种记号来方便地描述张量的维度：阶，形状以及维数。以下展示了它们之间的关系：

阶	形状	维数	实例
0	[]	0-D	一个0维张量，一个纯量
1	[D0]	1-D	一个1维张量的形式 [5]
2	[D0, D1]	2-D	一个2维张量的形式 [3, 4]
3	[D0, D1, D2]	3-D	一个3维张量的形式 [1, 4, 3]
n	[D0, D1, ..., Dn]	n-D	一个n维张量的形式 [D0, D1, ..., Dn]

数据类型

除了维度， tensor 有一个数据类型属性。你可以为一个张量指定下列数据类型中的任意一个类型：

数据类型	python类型	描述
DT_FLOAT	tf.float32	32位浮点数
DT_DOUBLE	tf.float64	64位浮点数
DT_INT64	tf.int64	64位有符号整型

DT_INT32	tf.int32	32位有符号整型
DT_INT16	tf.int16	16位有符号整型
DT_INT8	tf.int8	8位有符号整型
DT_UINT8	tf.uint8	8位无符号整型
DT_STRING	tf.string	可变长度的字节数组，每一个张量元素都是一个字节数组
DT_BOOL	tf.bool	布尔型
DT_COMPLEX64	tf.complex64	由32位浮点数组成的复数：实数和虚数
DT_QINT32	tf.qint32	用于量化Ops的32位有符号整型
DT_QINT8	tf.qint8	用于量化Ops的8位有符号整型
DT_QUINT8	tf.quint8	用于量化Ops的8位无符号整型

变量

在Variables 中查看更多细节。变量维护图执行过程中的状态信息。下面的例子演示了如何使用变量实现一个简单的计数器：

```
1  # 创建一个变量，初始为标量0
2  state = tf.Variable(0, name="counter")
3
4  # 创建一个op，其作用是使`state`增加1
5  one = tf.constant(1)
6  new_value = tf.add(state, one)
7  update = tf.assign(state, new_value)
8
9  # 启动图后，变量必须先经过init op初始化
10 # 首先先增加一个初始化op到图中
11 init_op = tf.initialize_all_variables()
12
13 # 启动图
14 with tf.Session() as sess:
15     # 运行init op
16     sess.run(init_op)
17     # 打印 state 的初始值
18     print sess.run(state)
19     # 运行op，更新state 并打印
20     for _ in range(3):
21         sess.run(update)
22         print sess.run(state)
23
24 # 输出：
25 # 0
26 # 1
27 # 2
28 # 3
```

代码中 assign() 操作是图所描述的表达式的一部分，正如 add() 操作一样，所以在调用 run() 执行表达式之前，它并不会真正执行赋值操作。

通常会将一个统计模型中的参数表示为一组变量。例如，你可以将一个神经网络的权重作为某个变量存储在一个 tensor 中。在训练过程中，通过反复训练图，更新这个 tensor。

Fetch

为了取回操作的输出内容，可以在使用 Session 对象的 run() 调用执行图时，传入一些 tensor，这些 tensor 会帮助你取回结果。在之前的例子里，我们只取回了单个节点 state，但是你也可以取回多个 tensor：

```
1  input1 = tf.constant(3.0)
2  input2 = tf.constant(4.0)
3  input3 = tf.constant(5.0)
4  intermed = tf.add(input2, input3)
5  mul = tf.mul(input1, intermed)
6
7  with tf.Session() as sess:
```

```
8 result = sess.run([mul, intermed])
9 print result
10
11 # print
12 # [27.0, 9.0]
```

需要获得更多个 `tensor` 值，在 `op` 的依次运行中获得（而不是逐个去获得 `tenter`）。

Feed

上述示例在计算图中引入 `tensor`，以常量或变量的形式存储。`Tensorflow` 还提供了 `feed` 机制，该机制可以临时替代图中的任意操作中的 `tensor` 可以对图中任何操作提交补丁，直接插入一个 `tensor`。

`feed` 使用一个 `tensor` 值临时替换一个操作的输出结果，你可以提供 `feed` 数据作为 `run()` 调用的参数。`feed` 只在调用它的方法内有效，方法结束，`feed` 就会消失。最常见的用例是将某些特殊的操作指定为 `feed` 操作，标记的方法是使用 `tf.placeholder()` 为这些操作创建占位符。

```
input1 = tf.placeholder(tf.types.float32)
input2 = tf.placeholder(tf.types.float32)
output = tf.mul(input1, input2)

with tf.Session() as see:
    print sess.run([output], feed_dict={input:[7.], input2:[2.]})

# print
# [array([ 14.], dtype=float32)]
```

转藏到我的图书馆

献花（0）

分享：

微信

来自：DeepMilan > 《Tensorflow》

以文找文 | 举报

上一篇：在崇尚极简主义的未来，我们不再需要APP

下一篇：【机器学习】Tensorflow基本使用

猜你喜欢

360doc 个人图书馆

首页

阅览室

馆友

我的图书馆

搜文章 找馆友

登录

注册

≡



六房间直播



角色扮演页游



小生意项目



霸业传奇



自我护理能力



现货白银



秀色直播间



原油模拟交易



原油分析师



原油交易点差

- 类似文章
- Word长篇文档排版技巧（一）
- 怎样才能成为亿万富翁
- 台湾美女画家董晓蕙画展作品
- 大学毕业前应该拿到的七类证书
- 秀色西湖
- 2011中国大学排行榜
- 拿花篮的美丽女孩（清纯美女）
- 难得一见的风景...【极品美图】
- 精选文章
- 看图学国际象棋
- 九部让你开怀大笑的电影
- 西北厨房的风水问题
- 你应该给女儿的忠告：想有个气质优雅的女儿...
- 电影告诉你生活是什么
- 读书无用，但为什么还要读？
- 凉皮的调料配方
- 死前1秒看到什么



鼠标手写输入



六房间直播




人有三个错误不能



lol竞猜的首页



《算命字典》举例



调查问卷与量表的



中国居民膳食营养



苏果lol的首页



lol职业联赛2f的首



霍兰德职业兴趣测

- 1 看老股民如何从15万变300万

2 远方驾校培训学校欢迎您!

3 外贸营销,日搜200国家外贸..
- 1 美亚保险官网

2 美亚保险

3 公司邮箱
- 4 北京口腔医院

5 企业邮箱注册

6 英语学习

发表评论：
请 [登录](#) 或者 [注册](#) 后再进行评论 社交帐号登录：