# TensorFlow学习

## TensorFlow的执行时的基本对象及其具体理解：

分为Tensor、Variable和operation三大类，他们共同构成执行时的graph，执行可以执行graph中的一个子节点，子节点依照DAG顺序执行期上游全部父节点。

示例：下面讲解概念时有具体示例

## 常量操作和变量、另外也理解一下执行时的基本对象

**import** tensorflow **as** tf

m1 **=** tf.constant([[3, 3]]) *# 常量operation，生成一个Tensor*

m2 **=** tf.constant([[1], [2]]) *# 常量operation，生成一个Tensor*

product **=** tf.matmul(m1, m2) *# 加法operation，输入两个Tensor，生成一个Tensor*

​

**with** tf.Session() **as** session: *# 利用Session执行上述代码中生成的DAG*

res **=** session.run(product)

print(res)

[[9]]

In [2]:

**import** tensorflow **as** tf

state **=** tf.Variable([0, 0], "counter") *# 变量*

new\_value **=** tf.add(state, [1, 2]) *# 加法operation 生成一个Tensor*

upstate **=** tf.assign(state, new\_value) *# 赋值operation 生成一个Tensor*

​

init **=** tf.global\_variables\_initializer() *# 变量初始化*

​

**with** tf.Session() **as** session:

session.run(init) *# 变量初始化*

**for** \_ **in** range(10):

session.run(upstate) *# 执行更新赋值也就是“+=”10次*

print(session.run(state)) *# 查看每次执行的效果*

[1 2]

[2 4]

[3 6]

[4 8]

[ 5 10]

[ 6 12]

[ 7 14]

[ 8 16]

[ 9 18]

[10 20]

## Feed和fetch

*# fetch*

**import** tensorflow **as** tf

m1 **=** tf.constant([[3, 3]]) *# 常量operation，生成一个Tensor*

m2 **=** tf.constant([[1], [2]]) *# 常量operation，生成一个Tensor*

m3 **=** tf.constant([3])

product **=** tf.matmul(m1, m2) *# 加法operation，输入两个Tensor，生成一个Tensor*

add **=** tf.add(product, m3)

​

**with** tf.Session() **as** session: *# 利用Session执行上述代码中生成的DAG*

res **=** session.run([product, add]) *# 同一个run中运行多个operation，operation之间可以有依赖关系*

print(res)

[array([[9]]), array([[12]])]

[array([[9]]), array([[12]])]

*# feed*

**import** tensorflow **as** tf

input1 **=** tf.placeholder(tf.float32)

input2 **=** tf.placeholder(tf.float32)

add **=** tf.add(input1, input2)

​

**with** tf.Session() **as** session: *# 利用Session执行上述代码中生成的DAG*

res **=** session.run(add, feed\_dict **=** {input1: [32.1], input2: 78.2}) *# 同一个run中运行多个operation，operation之间可以有依赖关系*

print(res)

[110.299995]

## Tensorflow进行机器学习的一个简单案例 ——梯度下降求解线性回归模型

*# 构造一个线性回归的数据集*

**import** numpy **as** np

x\_data **=** np.random.rand(1000)

y\_data **=** x\_data **\*** 2.0 **+** 1.5

​

*# 利用tensorflow中的梯度下降法训练该线性回归*

**import** tensorflow **as** tf

​

*# 定义线性回归的两个变量：系数和常数项*

k **=** tf.Variable(0.0)

b **=** tf.Variable(0.0)

y **=** x\_data **\*** k **+** b

​

*# 梯度下降需要：学习率、损失函数 --梯度下降会自动根据损失函数求导，这一点很方便*

loss **=** tf.reduce\_mean(tf.square(y\_data **-** y))

optimizer **=** tf.train.GradientDescentOptimizer(0.2)

train **=** optimizer.minimize(loss)

​

init **=** tf.global\_variables\_initializer()

​

*# 运行梯度下降4000次并打印每次的结果和对应的损失函数*

**with** tf.Session() **as** session:

session.run(init)

**for** step **in** range(4000):

session.run(train)

print(session.run([k, b, loss]))

运行结果：

[0.5802862, 1.0091515, 1.6426241]

[0.87479293, 1.4959297, 0.4374238]

[1.0294542, 1.7277478, 0.14771365]

[1.1155472, 1.8351988, 0.076466516]

[1.1678884, 1.8820568, 0.057430845]

[1.203488, 1.8994639, 0.0509492]

… … …

[1.9999962, 1.5000021, 1.2278321e-12]

[1.9999963, 1.500002, 1.1365984e-12]

[1.9999964, 1.500002, 1.0793286e-12]

[1.9999965, 1.5000019, 9.928555e-13]

[1.9999967, 1.5000019, 9.46713e-13]

[1.9999968, 1.5000018, 8.66649e-13]

[1.9999969, 1.5000018, 8.3117866e-13]

[1.9999969, 1.5000017, 7.964047e-13]

[1.999997, 1.5000017, 7.523795e-13]

[1.9999971, 1.5000017, 7.226504e-13]

[1.9999971, 1.5000015, 6.7916517e-13]

[1.9999973, 1.5000015, 6.461818e-13]

[1.9999973, 1.5000015, 6.461818e-13]

[1.9999973, 1.5000015, 6.461818e-13]

[1.9999973, 1.5000015, 6.461818e-13]

[1.9999973, 1.5000015, 6.461818e-13]

最后稳定在6.461818e-13处