1. rank为 2；rank应该译为秩，这个秩与矩阵的秩意义不同，tensor（张量）的rank指的是tensor的维度，和rank意义相同的表达方式还有order、degree、n-dimension。只有一个实数的tensor，其shape为（，），rank为0；一个向量，其shape为（n，），rank为1；一个矩阵，其shape为（n，m），rank为2，依次类推。
2. tf.matmul(A,B)是矩阵乘法运算，返回值就是两个矩阵进行矩阵乘法运算的结果。
3. tf.multiply(A,B)是tensor的element-wise product，即对应元素相加。要求A和B的shape相同。
4. tf.square(A)是element-wise square，即各个元素的平方。更多的Tensor运算可以查看tf.math包。
5. A，B不能进行矩阵乘法
6. A，B矩阵乘法结果的shape为（2,3,4,4），即除了最后两维要满足矩阵乘法的规则外，其他的维度的大小要相同。
7. （4,5）；Tensorflow 的broadcasting机制。
8. （2,3,4,5）；Tensorflow 的broadcasting机制。

config = tf.ConfigProto()

config.gpu\_options.allow\_growth = True

sess=tf.Session(config=config)

这样得到的session会动态分配显存。

config = tf.ConfigProto()

config.gpu\_options.per\_process\_gpu\_memory\_fraction = 0.4

session = tf.Session(config=config)

这样得到的session会一次占用GPU 40%的显存。

1. C=tf.where(A>B,A,B)
2. 不能使用 for i in range(tf.shape(A)[2]): 或者for i in range(numpy.shape(A)[2]):.

前者的tf.shape()没有与之相关联的节点被运算，所以不会返回A的第二个维度的大小。

而定义placeholder A时shape为[2,3,None]，所以使用numpy.shape()仍然无法得到A的动态维度的大小。

解决这种根据Tensor的参数循环的问题，应该使用tf.while\_loop(cond,body,(…)).

**def** cond(n, i, other\_param):

return i < n

**def** body(n, i, other\_param):

i += 1

other\_param=other\_param+"???"

return (n,i,other\_param)

print(sess.run(tf.while\_loop(cond, body, (tf.shape(A)[2], 0, ''))))

或者

print(sess.run(tf.while\_loop(**lambda** n, i, param: i < n,

**lambda** n, i, param:(n,i+1,param+’???’),

(tf.shape(A)[2], 0, ‘’))))

如果传入(feed)的A的shape是[2,3,4]，则输出都是（4,4,‘????????????’）

1. 与上题同理，同样不能直接使用if tf.shape(A)[2]==25: 和if numpy.shape(A)[2]==25:

应当使用tf.cond(cond,fun1,fun2)来进行条件控制。代码如下：

print(sess.run(tf.cond(tf.equal(tf.shape(A)[2],10),

**lambda**: 'yesyes',

**lambda**: 'nono'))

如果A的第二维大小是10，则应该输出yesyes。



namescope/var\_1:0

var\_2:0

variable\_scope/var\_3:0

variable\_scope/var\_4:0

1. tf\_reuse\_node\_mem\_test()的时间效率和空间效率都比另一个高。

tf\_add\_node\_mem\_test()在每次run时都会新建一个op计算节点，这样会不停的往session的graph中添加节点，使graph不断增大，所以空间效率低；而不停的申请内存和添加节点也拖慢了运行速度。

1. 2
2. [1,2,3]
3. [2,2]
4. [2,2]
5. 2
6. (3,4)
7. (2,4)
8. (2,3)
9. (4,)
10. (4,)
11. (2,3)
12. (5,7,4) (5,7,4)
13. 无法分割，因为维度1的大小7不能被分割块数2整除
14. (5,7,2) (5,7,2) (5,7,4)
15. （2,8）
16. 无法合并
17. 使用tf.add\_to\_collection()添加的集合是添加到默认图中，可以使用tf.Graph.clear\_collection(tf.get\_default\_graph(),key)来清除。