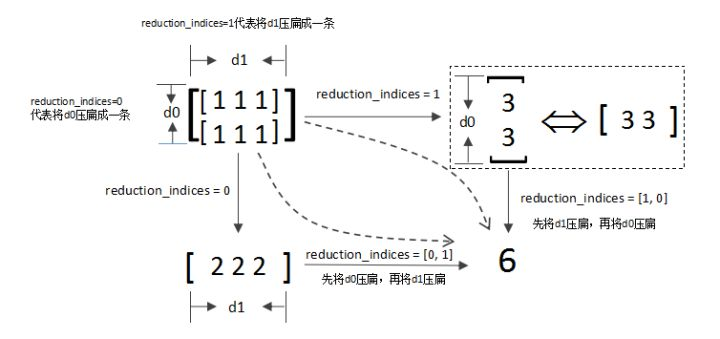
为了展示的更加清晰，我升级了一下这个图，但感觉还是不完美，[3, 3]竖起来过来显示是为了说明reduction\_indices=[1, 0]的过程中维度的信息是一直保留着的，所以它并不是一个列向量，亦即它不是[ [3], [3] ]，它本质还是[ 3, 3 ]，这也是为什么你在仅仅使用reduction\_indices=1的时候，打印出来的是[ 3, 3 ]的原因：



#############################################################################

**维度是用来索引一个多维数组中某个具体数所需要最少的坐标数量。**

把这句话多读几遍，我想你肯定会有所顿悟。这里之所以说第一个1维的例子时0维，是因为，一个数字根本不需要索引，因为就只有一个呀。所有不同维度的**形式**如下：

0维，又称0维张量，数字，标量：1

1维，又称1维张量，数组，vector：[1, 2, 3]

2维，又称2维张量，矩阵，二维数组：[[1,2], [3,4]]

3维，又称3维张量，立方（cube），三维数组：[ [[1,2], [3,4]], [[5,6], [7,8]] ]

n维：你应该get到点了吧~

**再多的维只不过是是把上一个维度当作自己的元素**

1维的元素是标量，2维的元素是数组，3维的元素是矩阵。

从0维到3维，边看边念咒语**“维度是用来索引一个多维数组中某个具体数所需要最少的坐标。”**

在纸上写写看，想要精确定位一个数字，需要几个数字呢？比如上面例子中的3维数组，我们想要3这个数字，至少要3个数字定位，它的坐标是（0为索引起点）：[0, 1, 0]

好了，现在就能说了，什么是轴（axis），如何索引axis（代码中常用的变量名，后文就用axis代表轴）。

什么是axis，编程时，你就可以简单的认为：

**axis是多维数组每个维度的坐标。**

同样，把这句话多读几遍，我想你一定有体悟。

还拿3维来说，数字3的坐标是[0, 1, 0]，那么第一个数字0的axis是0，第二个数字1的axis是1，第三个数字0的axis是2。

让我们再看看我们是如何得到3这个数字的：

1. 找到3所在的2维矩阵在这个3维立方的索引：0
2. 找到3所在的1维数组在这个2维矩阵的索引：1
3. 找到3这个数这个1维数组的索引：0

###########################################################################

下面结合自己的理解举个多维tensor例子简单说明。下面是个2\*3\*4的tensor。

[[[ 1 2 3 4]

[ 5 6 7 8]

[ 9 10 11 12]]

[[13 14 15 16]

[17 18 19 20]

[21 22 23 24]]]

如果计算tf.reduce\_sum(tensor, axis=0)，axis=0说明是按第一个维度进行求和，也就是说把

[[ 1 2 3 4]

[ 5 6 7 8]

[ 9 10 11 12]

和

[[13 14 15 16]

[17 18 19 20]

[21 22 23 24]]相加，所以第一个维度（也就是2）抹去，求和结束得到的tensor\_ans是3\*4（之前tensor是2\*3\*4）。显然tensor\_ans的元素分别是1+13;2+14;3+15……;12+24。即：

[[1+13 2+14 3+15 4+16]

[5+17 6+18 7+19 8+20]

[9+21 10+22 11+23 12+24]]。

依次类推，如果axis=1，那么求和结果shape是2\*4，即：

[[ 1 + 5 + 9 2 + 6+10 3 + 7+11 4 + 8+12]

[13+17+21 14+18+22 15+19+23 16+20+24]]

如果axis=2，那么求和结果shape是2\*3，即：

[[1+2+3+4 5+6+7+8 9+10+11+12]

[13+14+15+16 17+18+19+20 21+22+23+24]]

######################################################################

**什么是keepdims**

a=[ [[1,2], [3,4]], [[5,6], [7,8]] ]上面的reduce已经解释了，“塌缩”的是被操作的维度，那么keepdims也就是保持维度，直观来看就是**“不掉一层方括号”**，不掉哪层方括号呢？就是本来应该被塌缩的那一层（详细解释见评论）。tf.reduce\_sum(a, axis=1, keepdims=True)得出[[[ 4, 6]], [[12, 14]]]，可以看到还是3维。这种尤其适合reduce完了要和别的同维元素相加的情况。

最后来做个练习吧，对于四维情况[ [ [[1,2], [3,4]] ], [ [[3,4], [5,6]] ] ]，对axis=0，1，2，3分别在草稿纸上进行演算后，用tensorflow做个验证吧~