pandas基础教程六-广播操作

一、为什么要讲述广播操作

“广播”指的是在不同维度的数组之间进行算术运算的一种执行机制，其通过将数据矢量化进行高效的运算。在使用numpy和pandas以及一些机器学习或者深度学习的一些模块的时候。我们都会涉及到一些加减乘除的操作，这些操作会有一种特殊的效果，这个效果就是我们要讲述的广播操作即broadcast。弄懂广播操作有利于我们进行数据处理操作。

二、pandas广播操作

在panda中广播操作是指一个Series数组和另一个DataFrame进行运算，举例如下：

import pandas as pd  
  
a = pd.Series([0,1,2])  
b = pd.DataFrame([[0,0,0],[10,10,10],[20,20,20],[30,30,30]],  
 index=[**'A'**,**'B'**,**'C'**,**'D'**])  
print(a)  
print(b)  
print(a + b)

#输出如下

0 0

1 1

2 2

dtype: int64

0 1 2

A 0 0 0

B 10 10 10

C 20 20 20

D 30 30 30

0 1 2

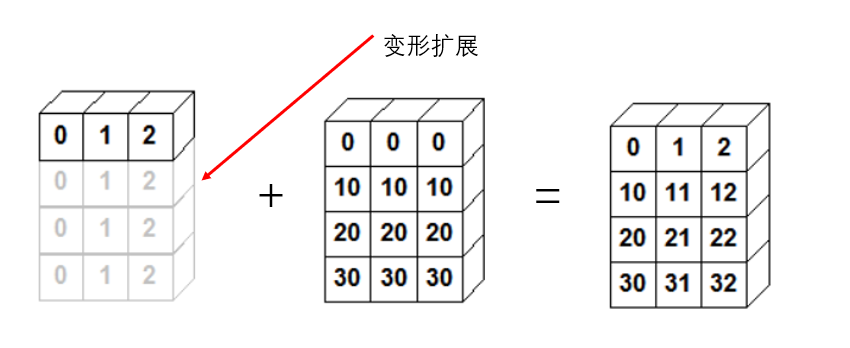
A 0 1 2

B 10 11 12

C 20 21 22

D 30 31 32

从结果上来看是很神奇的，从原理上来说就是a进行了维度的扩展，变成了[[0,1,2], [0,1,2], [0,1,2], [0,1,2]]，然后与b进行相加的操作。示意图如下：



当然了其他的运算也会有同样的效果：

c = pd.Series([0.1,0.2,0.3])  
d = 10  
print(c\*d)  
print(c/10)

维度不一致的时候，会怎样呢？

import pandas as pd  
  
a\_1 = pd.Series([0,1])  
b\_1 = pd.DataFrame([[0,0,0],[10,10,10],[20,20,20],[30,30,30]],  
 index=[**'A'**,**'B'**,**'C'**,**'D'**])print(a\_1 + b\_1)

#输出

0 1 2

A 0.0 1.0 NaN

B 10.0 11.0 NaN

C 20.0 21.0 NaN

D 30.0 31.0 NaN

结果还是很好理解的，总维度与高维度的数据维度一致。

二、numpy广播操作

numpy两个数组的相加、相减以及相乘都是对应元素之间的操作,比如说：

import numpy as np  
  
A = np.array([  
 [2,2,3],  
 [1,2,3]  
])  
B = np.array([  
 [1,1,3],  
 [2,2,4]  
])  
  
print(A + B)  
print(A \* B)

# 输出：

[[3 3 6]

[3 4 7]]

[[ 2 2 9]

[ 2 4 12]]

上述例子是维度一致的，我们来看维度不一致的情况

import numpy as np  
E = np.array([  
 [0, 0, 0],  
 [1, 1, 1],  
 [2, 2, 2],  
 [3, 3, 3]  
])  
  
F = np.array([1, 2, 3])  
sum = E + F  
print(E.shape)  
print(F.shape)  
print(sum.shape)  
print(sum)

# 输出

(4, 3)

(3,)

(4, 3)

[[1 2 3]

[2 3 4]

[3 4 5]

[4 5 6]]

通过广播机制，将F进行扩展，扩展之后就可以与原来的数据进行相加的操作。

我们将F数据进行变换一下：

import numpy as np  
E = np.array([  
 [0, 0, 0],  
 [1, 1, 1],  
 [2, 2, 2],  
 [3, 3, 3]  
])  
F\_1 = np.array([  
 [1],  
 [2],  
 [3],  
 [4]  
])  
  
sum2 = E + F\_1  
print(sum2)

#输出

[[1 1 1]

[3 3 3]

[5 5 5]

[7 7 7]]

总而言之，就是将低维度扩展到高纬度。

我们来总结一下广播的机制：

如果两个数组的**后缘维度(从末尾开始算起的维度)**的**轴长度相符**或**其中一方的长度为1**，则认为它们是广播兼容的。广播会在**缺失维度**和(或)**轴长度为1的维度**上进行。

三、Pytorch广播操作

深度学习框架Pytorch中也使用了广播机制, pytorch中的广播机制和numpy中的广播机制一样, 因为都是数组的广播机制，当然了Pandas也是。

import torch  
a = torch.arange(0,6).reshape((6,))  
b = torch.arange(0,12).reshape((2,6))  
res = torch.mul(a,b)  
  
print(a.shape)  
print(b.shape)  
print(res.shape)  
print(res)

# 输出

torch.Size([6])

torch.Size([2, 6])

torch.Size([2, 6])

tensor([[ 0, 1, 4, 9, 16, 25],

[ 0, 7, 16, 27, 40, 55]])

我们来看一个不能进行广播的例子：

import torch  
x=torch.empty((0,))  
y=torch.empty(2,2)  
  
print(x)  
print(y)  
print(x.shape)  
print(y.shape)

# 输出如下：

tensor([])

tensor([[0., 0.],

[0., 0.]])

torch.Size([0])

torch.Size([2, 2])

如果执行：

print(x + y)

就会报错如下：

The size of tensor a (0) must match the size of tensor b (2) at non-singleton dimension 1

也就是说不能进行广播的操作。总结一下Pytorch的广播机制：

1. 每个张量至少具有一个维度；

2. 广播时，从尾部维度开始，满足尺寸大小相等或其中一个为1或其中一个不存在。