2018.08.15 linear\_multi.py

Numpy.stack

Numpy.stack(arrays,axis=0)

函数功能：堆叠数组

参数说明：

arrays，需要进行堆叠的数组

axis,堆叠时使用的轴，axis=i时，表示在堆叠时首先选取第i+1维进行打包

具体例子：

eg1:

arrays = [[1,2,3,4],[5,6,7,8]]

axis=0表示的是第一维，即arrays[0]=[1,2,3,4]或arrays[1]=[5,6,7,8]。

eg2:

array = [[[1,2,3,4],[1,2,3,4],[1,2,3,4]]]

>>>print array

[[[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4]]]

arrays = np.array(array)

>>>print arrays (注意其中已没有逗号进行分隔)

[[[1 2 3 4]

[1 2 3 4]

[1 2 3 4]]]

print np.stack(arrays,axis = 0)

>>>

#取出第一维的[[1,2,3,4],[1,2,3,4],[1,2,3,4]]进行打包

[[[1 2 3 4]

[1 2 3 4]

[1 2 3 4]]]

print np.stack(arrays,axis = 1)

>>>

#先对第二维进行打包，即[1,2,3,4]进行打包，其余类似。

[[[1 2 3 4]]

[[1 2 3 4]]

[[1 2 3 4]]]

print np.stack(arrays,axis = 2)

>>>

#先对第三维进行打包。

[[[1]

[2]

[3]

[4]]

[[1]

[2]

[3]

[4]]

[[1]

[2]

[3]

[4]]]

eg3:

a = np.array([[1,2,3,4],[5,6,7,8]])

arrays2 = np.asarray([a,a,a])

>>>

arrays2为：

[[[1 2 3 4]

[5 6 7 8]]

[[1 2 3 4]

[5 6 7 8]]

[[1 2 3 4]

[5 6 7 8]]]

print np.stack(arrays2,axis = 0)

#注意stack()函数会先把参数arrays中的每个元素变为numpy的数组，数组之间时没有逗号的。

>>>

[[[1 2 3 4]

[5 6 7 8]]

[[1 2 3 4]

[5 6 7 8]]

[[1 2 3 4]

[5 6 7 8]]]

print np.stack(arrays2,axis = 1)

>>>

[[[1 2 3 4]

[1 2 3 4]

[1 2 3 4]]

[[5 6 7 8]

[5 6 7 8]

[5 6 7 8]]]

print np.stack(arrays2,axis = 2)

>>>

[[[1 1 1]

[2 2 2]

[3 3 3]

[4 4 4]]

[[5 5 5]

[6 6 6]

[7 7 7]

[8 8 8]]]

参考文章：<https://blog.csdn.net/csdn15698845876/article/details/73380803>

https://blog.csdn.net/qq\_26414307/article/details/79317965

pytorch squeeze(),unsqueeze()

squeeze()

函数功能：去除size为1的维度，包括行和列。当维度大于等于2时，squeeze()无作用。

其中squeeze(0)代表若第一维度值为1则去除第一维度，squeeze(1)代表若第二维度值为1则去除第二维度。

eg1：

**a = torch.Tensor(1,3)**

print a

>>>

tensor([[-1.37,4.56,-3.57]])

print **a.squeeze(0)**

>>>

tensor([-1.37,4.56,-3.57])

print **a.squeeze(1)**

>>>

tensor([[-1.37,4.56,-3.57]])

eg2：

**b = torch.Tensor(2,3)**

print b

>>>

tensor([[-3.17,3.09,1.43],

[0.00,0.00,0.00]])

print b.squeeze(0)

>>>

tensor([[-3.17,3.09,1.43],

[0.00,0.00,0.00]])

print b.squeeze(1)

>>>

tensor([[-3.17,3.09,1.43],

[0.00,0.00,0.00]])

eg3：

**c = torch.Tensor(3,1)**

print c

>>>

tensor([[-3.54],

[3.09],

[0.00]])

print c.squeeze(0)

>>>

tensor([[-3.54],

[3.09],

[0.00]])

print c.squeeze(1)

>>>

tensor([-3.54,3.09,0.00])

eg4：

**d = torch.rand(4,1,3)**

print d

>>>

tensor([[[0.19,0.25,0.23]],

[[0.91,0.66,0.12]],

[[0.82,0.07,0.73]],

[[0.35,0.06,0.10]]])

print **d.squeeze()**

>>>

tensor([[0.19,0.25,0.23],

[0.91,0.66,0.12],

[0.82,0.07,0.73],

[0.35,0.06,0.10]])

eg5：

**e = torch.rand(4,3,1)**

print e

>>>

tensor([[[0.97],

[0.86],

[0.52]],

[[0.88],

[0.76],

[0.54]],

[[0.61],

[0.27],

[0.56]],

[[0.56],

[0.66],

[0.53]]])

print e.squeeze()

>>>

tensor([[0.97, 0.86, 0.52],

[0.88, 0.76, 0.54],

[0.61, 0.27, 0.56],

[0.56, 0.66, 0.53]])

eg6：

**f = torch.rand(4,3,2)**

print f

>>>

tensor([[[0.90,0.26],

[0.78,0.33],

[0.45,0.71]],

[[0.25,0.87],

[0.36,0.37],

[0.60,0.88]],

[[0.32,0.06],

[0.63,0.23],

[0.13,0.08]],

[[0.53,0.92],

[0.56,0.27],

[0.41,0.08]]])

print f.squeeze()

>>>

tensor([[[0.90,0.26],

[0.78,0.33],

[0.45,0.71]],

[[0.25,0.87],

[0.36,0.37],

[0.60,0.88]],

[[0.32,0.06],

[0.63,0.23],

[0.13,0.08]],

[[0.53,0.92],

[0.56,0.27],

[0.41,0.08]]])

unsqueeze()

函数功能：与squeeze()函数功能相反，用于添加维度。

eg：

**g = torch.Tensor(3)**

print g

>>>

tensor([3.27,4.56,-4.84])

print **g.unsqueeze(0)**

>>>

tensor([[3.27,4.56,-4.84]])

print **g.unsqueeze(1)**

>>>

tensor([[3.27],

[4.56],

[-4.84]])

#print g.unsqueeze() 必须指明维度

参考文章：

<https://blog.csdn.net/jacke121/article/details/80595928>

<https://blog.csdn.net/abc781cba/article/details/79663190>

2018.8.16 logistic.py

python 中的lambda filter map reduce

lambda 为关键字，是实现python中单行最小函数。

lambda x:y

说明： x是入参，y是表达式

g = lambda x:x \* 2

相当于

def g(x):

return x\*2

Eg1：

print [(lambda x : x\*x - 4)(x) for x in range(4)]

>>>

[-4, -3, 0, 5]

print (lambda x:x\*x - 4)(3)

>>> 5

g = lambda x: x\*x-4

for i in range(10):

print g(i)

>>>

-4 -3 0 5 12 21 32 45 60 77

Filter

filter(function, sequence)

函数功能：对sequence 中的item依次执行function(item)，将执行结果为True的item组成一个List/String/Tuple(取决于sequence的类型)

参数说明：

function 为函数

sequence 为列表/元组/字符串

返回值：

item列表/元组/字符串

Eg2：

print filter(lambda x:x\*x-4,range(10))

>>>

[0, 1, 3, 4, 5, 6, 7, 8, 9]

Map

map(function, sequence)

函数功能：对sequence中的item依次执行function(item)，将执行结果function(item)组成一个List返回。

参数说明：

function 为函数

sequence 为列表/元组/字符串,有多少个列表就应该有多少个入参，具体例子见Eg3.2.

返回值：

function(item)列表

Eg3.1:

print map(lambda x:x\*x-4,range(10))

>>>

[-4, -3, 0, 5, 12, 21, 32, 45, 60, 77]

Eg3.2:

print map(lambda x, y:x\*y-4,range(3),[8,9,10])

>>>

[-4, 5, 16]

Reduce

reduce(function, sequence, starting\_value)

函数说明：

对sequence中的item顺序迭代调用function，如果有starting\_value，可作为初始值调用。

function接收的参数个数只能为2，先把sequence中的第一个和第二个值当参数传给function，再把function的返回值和第三个值当参数传给function，然后只返回一个结果。

参数说明：

function 为函数

sequence 为列表/元组/字符串

starting\_value 为初始值

返回值：

数值结果

Eg4：

print reduce(lambda x,y:x\*y-4,range(4))

>>>

-40

print reduce(lambda x,y:x+y, range(101))

#计算0到100的和

>>>

5050

print reduce(lambda x,y:x+y, range(101),100)

>>>

5150

<https://www.cnblogs.com/yufeihlf/p/6179982.html>

http://www.runoob.com/python/python-func-filter.html