pytorch—索引与切片

索引

如果只给出tensor一个参数，则默认是从tensor张量从左往右的参数索引。

```
In [2]:
1 a = torch.rand(4, 3, 28, 28)

In [5]:
1 a[0].shape

Out[5]: torch.Size([3, 28, 28])

In [6]:
1 a[0,0].shape

Out[6]: torch.Size([28, 28])

In [7]:
1 #返回的是标量
2 a[0,0,2,4]

Out[7]: tensor(0.4782)
```

连续切片

切片里面的参数都是左闭右开的。

```
In [37]:
1 a.shape

Out[37]: torch.Size([4, 3, 28, 28])

In [43]:
1 #从0到2但不包括2
2 a[:2].size()

Out[43]: torch.Size([2, 3, 28, 28])

In [41]:
1 #前两块的第一通道的数据。后面参数可写可不写，冒号表示取全部
2 a[:2,:1,:,:].shape

Out[41]: torch.Size([2, 1, 28, 28])

In [44]:
1 #从1通道开始到末尾
2 a[:2,1:,:,:].shape

Out[44]: torch.Size([2, 2, 28, 28])

In [42]:
1 #反向索引
2 a[:2,-1:].shape

Out[42]: torch.Size([2, 1, 28, 28])
```

不连续切片

```
1、  :   表示全部数据
2、 :n   n:   表示从前取到n，或者取n和n之后的数据
3、 n1 :n2    [start，end)  区间取值，左闭右开
4、n1：n2：x    [start，end，steps) 后面的数字是多少表示隔几个数取1个数据
```

```
In [45]:  1  a[:,:,0:28:2,0:28:2].shape
Out[45]: torch.Size([4, 3, 14, 14])

In [46]:  1  a[:,:,::2,::2].shape
Out[46]: torch.Size([4, 3, 14, 14])

In [53]:  1  a[:,:,::10,::10].shape
Out[53]: torch.Size([4, 3, 3, 3])
```

具体索引

a.index_select(dim,index) ：a是张量，dim是维度，你选择的是第几个数据；index是包含索引的一维张量。

index_select(tensor, dim, index） ： 将张量传入到函数里面进行索引，tensor代表张量。

注意：后面的index必须要是tensor类型的一维张量。


```
In [45]:  1  a[:,:,0:28:2,0:28:2].shape
Out[45]: torch.Size([4, 3, 14, 14])

In [46]:  1  a[:,:,::2,::2].shape
Out[46]: torch.Size([4, 3, 14, 14])

In [53]:  1  a[:,:,::10,::10].shape
Out[53]: torch.Size([4, 3, 3, 3])
```


torch.take(input, index) ： input 是输入的张量，index是一维张量。这个方法会将原先的所有数据进行打散，变为一维的张量。之后根据第二个参数进行下标索引。

```
In [110]:
1 src = torch.tensor([[1,2,3],[4,5,6]])
2 print(src)
```

```
tensor([[1, 2, 3],
        [4, 5, 6]])
```

```
In [115]:
1 torch.take(src,torch.tensor([0,2,5]))
```

```
Out[115]: tensor([1, 3, 6])
```

...索引

... ：系统根据实际情况去推测有多少数据，然后全部取。

```
In [82]:
1 a.shape
```

```
Out[82]: torch.Size([4, 3, 28, 28])
```

```
In [83]:
1 a[...].shape
```

```
Out[83]: torch.Size([4, 3, 28, 28])
```

```
In [84]:
1 a[0,...].shape
```

```
Out[84]: torch.Size([3, 28, 28])
```

```
In [86]:
1 a[:,1,...].shape
```

```
Out[86]: torch.Size([4, 28, 28])
```

```
In [87]:
1 a[...,:2].shape
```

```
Out[87]: torch.Size([4, 3, 28, 2])
```

mask索引

要取出所有大于0.5的数据。

masked_select(input, mask） ：input是输入的张量，mask是bool类型的张量。

该方法输出的是一个一维的张量，具体个数为符合要求数据的个数。

In [89]:
```
x = torch.randn(3,4)
print(x)
```

```
tensor([[-0.5032,  0.4702,  1.0794, -0.5697],
        [ 0.1836,  1.6747, -0.6801,  0.5153],
        [ 1.6187, -0.0209,  0.3865, -0.4076]])
```

In [91]:
```
mask = x.ge(0.5)
print(mask)
```

```
tensor([[False, False,  True, False],
        [False,  True, False,  True],
        [ True, False, False, False]])
```

In [97]:
```
torch.masked_select(x,mask)
```

Out[97]:
```
tensor([1.0794, 1.6747, 0.5153, 1.6187])
```

In [99]:
```
torch.masked_select(x,mask).shape
```

Out[99]:
```
torch.Size([4])
```