Tensor数据类型

浮点类型

dtype类型	Tensor对象	含义
torch.float torch.float32	torch.FloatTensor	浮点型
torch.float64 torch.double	torch.DoubleTensor	双精度浮点型
torch.float16 torch.half	torch.HalfTensor	半精度浮点型

整数类型

dtype类型	Tensor对象	含义
torch.uint8	torch.ByteTensor	无符号8位整型
torch.int8	torch.CharTensor	8位整型
torch.int16 torch.short	torch.ShortTensor	16位整型
torch.int32 torch.int	torch.IntTensor	整型
torch.int64 torch.long	torch.LongTensor	64位长整型

量化整数类型（quantized integer）：通常用于量化神经网络中，以减少模型的内存占用和计算成本。通过将浮点数参数和激活值量化为整数，可以显著减少存储和计算的需求。

dtype类型	Tensor对象	含义
torch.quint8	torch.ByteTensor	量化无符号8位整型
torch.qint8	torch.CharTensor	量化8位整型
torch.qint32	torch.IntTensor	量化32位整型
torch.quint4x2	torch.IntTensor	量化无符号4位整型

布尔及复数类型

dtype类型	Tensor对象	含义
torch.bool	torch.BoolTensor	布尔
torch.complex32 torch.chalf		32位浮点数组成的复数类型，其中实部和虚部都是32位浮点数。
torch.complex64 torch.cfloat		64位复数

dtype类型	Tensor对象	含义
torch.complex128 torch.cdouble		128位复数

代码复现

```
import torch
import numpy as np

tensor1 = torch.Tensor([1, 2, 3])
print(tensor1.data.dtype)

tensor2 = torch.FloatTensor([1, 2, 3])
print(tensor2.data.dtype)

# tensor 默认数据类型是float32

print(torch.float64)
print(torch.double)

# %%

a = torch.tensor([i for i in range(12)], dtype=torch.int32)

print(a)

a = a.reshape(2, 6)
print(a)

a = a.reshape(-1, 3, 2)
print(a)

print(a.view(-1, 6))
```

```
# tensor 和numpy互转

data = [[1, 2, 3], [4, 5, 6]]
n = np.array(data)
print(type(n))
# numpy转tensor
t = torch.tensor(n)
print(type(t))

# tensor转np
n1 = t.numpy()
print(type(n1))

# np转tensor
n2 = torch.from_numpy(n1)
print(type(n2))
```

```
# %%
t1 = torch.tensor([[1, 2, 3], [4, 5, 6]])
t2 = torch.tensor([[1, 2, 3], [4, 5, 6]])

# 查看cuda是否可用，
print(torch.cuda.is_available())
print(t1.device)

# 调用cuda  方法1
t1 = t1.cuda()
t2 = t2.to('cuda')  # 方法2

print(t1.device)
print(t2.device)

# 放回cpu
t1 = t1.cpu()
t2 = t2.to('cpu')

print(t1.device)
print(t2.device)
```

```
# %%
a = torch.tensor([
    [[1, 2, 3], [7, 8, 9]],
    [[4, 5, 6], [2, 2, 2]]
], dtype=torch.int32)

"""
2轴的元素求和  1+2+3，7+8+9 4+5+6 2+2+2
"""
print(a.sum(dim=2))

"""
1轴的元素求和
[1, 2, 3] + [7, 8, 9] , [4, 5, 6] + [2, 2, 2]
"""
print(a.sum(dim=1))

"""
0轴的元素求和
[[1, 2, 3], [7, 8, 9]]  + [[4, 5, 6], [2, 2, 2]]
"""
print(a.sum(dim=0))

#
data = torch.Tensor([i for i in range(12)]).reshape(-1, 2, 3)
# 张量中元素个数
print(torch.numel(data))
# 对角线为1的n*n张量
print(torch.eye(10))
# 交换维度
print(torch.transpose(data, 1, 0))
# item()获取元素值
```

```python
print(data[0, 0, 1].item())

# %% 矩阵相乘 mm二维 matmul高维
a = torch.arange(1, 7).reshape(2, 3)
b = torch.arange(7, 13).reshape(3, 2)
c = torch.mm(a, b)
# c = torch.matmul(a, b)
print(a)
print(b)

print(c)

#  dot点乘（只能一维，且元素数一样）
t1 = torch.tensor([1, 2, 3])
t2 = torch.tensor([4, 5, 7])

t3 = torch.dot(t1, t2)
print(t3)

#  内积
a = torch.arange(1, 5).reshape(2, 2)
b = torch.arange(5, 9).reshape(2, 2)
print(a)
print(b)
"""
[[1, 2],
[3, 4]]
    inner
[[5, 6],
 [7, 8]]
    =
[
    [1*5+2*6, 1*7+2*8]
    [3*5+4*6, 3*7+4*8]
]
"""
print(torch.inner(a, b))

# 转置
a = torch.arange(6).reshape(2, 3)

print(a.T)

#  cat连接

a = torch.tensor([[1], [2]])
b = torch.tensor([[3], [4]])

c = torch.cat([a, b], dim=0)
d = torch.cat([a, b], dim=1)

# print(c)
print(c.shape)
# print(d)
print(d.shape)
```

```
# 数据格式 nchw
data1 = torch.randn(128, 3, 28, 28)
data2 = torch.randn(128, 3, 28, 28)

# dim为0，连接n批次
print(torch.cat([data1, data2], dim=0).shape)
# dim为1，连接c通道
print(torch.cat([data1, data2], dim=1).shape)

#  压缩squeeze
x = torch.zeros(2, 1, 2, 1, 2)
print(x.size())
y = torch.squeeze(x)
print(y.size())

# 升维
y = torch.unsqueeze(y, 1)
print(y.size())

# %% 拆分split
import torch

x = torch.arange(1, 10).reshape(3, 3)

# 0轴进行拆分，步长为1
a, b, c = x.split(1, dim=0)
print(a)
print(b)
print(c)
# 1轴进行拆分
a, b, c = x.split(1, dim=1)
print(a)
print(b)
print(c)
```

```
# %% argmax()，eq()，mean()实际应用：网络计算结果的评分计算
import torch

# 模拟生成批次为128的十分类结果
input_data = torch.randn(128, 10)
# 使用softmax归一化
out = torch.nn.Softmax(dim=1)(input_data)
# 提取最大概率的索引
out_idx = torch.argmax(out, dim=1)

print(out_idx)
# 模拟label
label = torch.nn.Softmax(dim=1)(torch.randn(128, 10))
label_idx = torch.argmax(label, dim=1)

# 计算评分
score = torch.mean(torch.eq(out_idx, label_idx).float())
print(score)
```