Tensor 多维数组

torch.Tensor
- data 被包装的Tensor
- dtype 张量数据类型 [float, int, bool]
- shape 张量形状
- device 张量所在设备
- requires_grad 是否需要梯度
- grad data的梯度
- grad_fn 创建Tensor的Function.
- is_leaf 是否是叶子结点

张量的创建

一、直接创建

torch.tensor(data, dtype, device, requires_grad)

↑

可以是 ndarray, list

torch.from_numpy(ndarray) 从numpy创建tensor

(创建的tensor与ndarray共享内存，改一个另一个也会变)

二. 依据数值创建

torch.zeros(size, out, layout, device, requires_grad)

out_t

out：生成张量赋给 out_t

layout：不需要处理

torch.zeros_like(input, ...)

生成与 input 形状相同的张量

torch.ones( )
torch.ones_like( )

torch.full(size, value, ...) 生成全为 value 的张量
torch.full_like(input, value, ...)

torch.arange(start, end, step, ...)
生成一维等差张量 [start, end)
torch.linspace(start, end, steps, ...)

生成均分 [start, end] 长度为 steps 的张量

torch.logspace(start, end, steps, base, ...)
生成对数均分 [start, end] 长度为 steps 的张量

torch.eye(n, ...)
生成 n, n 的对角阵

三、依概率分布创建
正态分布
torch.normal(mean, std, size)

torch.randn(size) 标准正态分布
torch.randn_like(input, ...)

torch.rand(size, ...) 在[0,1)的均匀分布
torch.rand_like(input, ...)

torch.randint(low, high, size, ...)
在[low, high)的均匀分布
torch.randint_like(input, ...)

torch.randperm(n) 生成从0到n-1的随机排列
torch.bernoulli(input)
↑ 概率值 生成概率值为input的伯努利分布