3 自动求导

LibTorch 简单教程

2025-12-01

目录

1 计算图

1.1 创建计算图

假设有一个运算表达式：$f(x,y)=(x+y)(y+1)$，要对 x和y 分别求偏导数。这是一个多元函数，我们可以把计算过程用图表示出来，这个图就称为计算图。在代码中，张量经过任何“可微分”的运算后，张量和“运算”都会被记录在这个图中，运算结束后图就创建成功了。有了这个图，我们就可以实现自动求导。自动求导函数会依赖已存在的计算图求解梯度，并且结束后默认销毁这里创建的计算图。

1.2 节点

计算过程中，类比多元函数的自变量（叶子节点）和中间变量（非叶子节点），有的张量是被其他张量计算出来的临时变量，这种叫非叶子节点。不由任何张量计算而来，直接存在的叫叶子节点。在计算图中，要重点注意张量的这些属性：是否叶子、是否需要梯度、梯度值、梯度函数。它们可以由这些函数获取：Tensor::is_leaf() 、Tensor::requires_grad() 、Tensor::grad()、Tensor::grad_fn()。由叶子节点计算出的节点默认需要梯度。

1.3 静态图和动态图

PyTorch 是动态图，具有容易理解，灵活的特点。

2 自动求导函数

自动求导自动求导常用的 API 有 2 个函数、1 个类：torch::autograd::backward()、torch::autograd::grad()、torch::autograd::function::Function。

backward()函数介绍如下：

功能：自动求解计算图中所有节点的梯度。我们一般使用Tensor::backward()函数求导，它内部会调用torch::autograd::backward()。

参数：

返回值：

backward()函数会累计梯度值到叶子节点，不会自动清零梯度值，所以训练过程需要手动清零梯度值。grad()函数不累计梯度，仅仅计算并返回梯度值。

retain_graph和create_graph参数的区别是前者控制函数计算图，后者控制梯度计算图，联系是后者利用前者完成创建，创建后两个图就互相独立，销毁与否都不影响对方。

根据计算图的原理，在求导语句和最终变量语句之间，不可以改变任何变量，才能保证结果正确。如果某个变量被改变，计算图会记录改变后的值，导致结果错误。如果想要修改，使用Tensor::detach()把变量剥离计算图，得到数据引用，和原变量共享同一块内存，但是修改不会破坏计算图。

还可以局部控制计算图搭建过程的临时变量是否保存。使用torch::no_grad()函数。