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

自动求导常用的 API 有 2 个函数、1 个类：torch::autograd::backward()、torch::autograd::grad()、torch::autograd::function::Function< >。这些是原始的 API，我们一般使用Tensor::backward()函数完成反向求导。

2.1 Tensor::backward()

函数介绍如下：原型

```
inline void Tensor::backward( const Tensor &gradient = {},
    std::optional<bool> retain_graph = std::nullopt,
    bool create_graph = false,
    std::optional<TensorList> inputs = std::nullopt ) const
```

功能：自动求解当前标量相对于计算图中所有叶子节点的梯度，否则要指定gradient参数。它内部会调用torch::autograd::backward()。函数会累计（加法）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。

参数：一共有四个

- gradient 非标量时，这个函数会计算雅可比向量积，这个参数接收雅可比矩阵，描述分量的上游梯度关系。当前张量非标量，必须指定某种运算关系，把分量规约为新的标量，新标量对分量的梯度矩阵就是这个参数，也称为上游梯度。

- retain_graph 这个函数求导时，依赖已经存在的计算图，求导结束后，会默认销毁该计算图。设置retain_graph=true即可保留，方便重复使用。
- create_graph 求出来的梯度是一个张量，它也是被运算出来的，那么就会创建另一张计算图描述梯度和自变量的关系。设置create_graph=true即可保留，方便高阶求导。
- inputs 我们一般用不到，在 API 页面有说明。

返回值：void

2.2 torch::autograd::grad()

函数介绍如下：原型

```
variable_list torch::autograd::grad(const variable_list &outputs,
    const variable_list &inputs,
    const variable_list &grad_outputs = {},
    std::optional<bool> retain_graph = std::nullopt,
    bool create_graph = false,
    bool allow_unused = false)
```

功能：不累计梯度，仅仅计算并返回梯度值。

参数：

- outputs
- inputs
- grad_outputs
- retain_graph 同上
- create_graph 同上
- allow_unused

返回值：variable_list类型，可以用torch::Tensor接收。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。

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。

2.3 torch::autograd::function::Function< >

这是一个模板类。如果我们有额外需求的计算，可以自定义类，公有继承它，重写forward()和backward()函数，这样自定义运算就变成“可微分运算”了，计算图就能记录此运算，方便自动求导。使用方法在官方仓库中有例子。

2.4 计算图与自动求导的关系和注意事项

计算图是由张量计算过程中自动创建的，自动求导函数只是使用现成的计算图，不会创建新的“函数计算图”，而且还会默认销毁这个图。

根据计算图的原理，假设我们已经得到最终变量，下一语句要求导了，在这两语句之间，不可以改变任何图内变量，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，再修改则不会破坏计算图。