backward

2023 年 7 月 15 日

1 PyTorch 中的 backward 背后的数学概念

1.1 前置知识

设

$$X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}, \qquad A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}$$

令矩阵函数 F 为

$$\begin{aligned} F(X) &= AX & (1) \\ &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} & (2) \\ &= \begin{pmatrix} f_{11}(X) & f_{12}(X) \\ f_{21}(X) & f_{22}(X) \\ f_{31}(X) & f_{32}(X) \end{pmatrix} & (3) \end{aligned}$$

则

$$\frac{\partial F}{\partial X} = \begin{pmatrix} \frac{\partial f_{11}}{\partial X} & \frac{\partial f_{12}}{\partial X} \\ \frac{\partial f_{21}}{\partial X} & \frac{\partial f_{22}}{\partial X} \\ \frac{\partial f_{31}}{\partial X} & \frac{\partial f_{32}}{\partial X} \end{pmatrix}$$

若想求 $\frac{\partial F}{\partial X}$, 则需要求出每一项 $\frac{\partial f_{ij}}{\partial X}$

以 $f_{22}(X)$ 为例，已知

$$f_{22}(X) = a_{21}x_{12} + a_{22}x_{22}$$

则有

$$\frac{\partial f_{22}}{\partial x_{12}} = a_{21}, \quad \frac{\partial f_{22}}{\partial x_{22}} = a_{22}$$

进而，有

$$\frac{\partial f_{22}}{\partial X} = \begin{pmatrix} \frac{\partial f_{22}}{\partial x_{11}} & \frac{\partial f_{22}}{\partial x_{12}} \\ \frac{\partial f_{22}}{\partial x_{21}} & \frac{\partial f_{22}}{\partial x_{22}} \end{pmatrix} \tag{4}$$

$$= \begin{pmatrix} 0 & a_{21} \\ 0 & a_{22} \end{pmatrix} \tag{5}$$

1.2 代码实践

令

$$X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \qquad A = \begin{pmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{pmatrix}$$

首先计算实值函数 $f_{11} = a_{11}x_{11} + a_{12}x_{21}$ 的导数

直接计算可得,

$$\frac{\partial f_{11}}{\partial X} = \begin{pmatrix} a_{11} & 0 \\ a_{12} & 0 \end{pmatrix} \tag{6}$$

$$= \begin{pmatrix} 2 & 0 \\ 3 & 0 \end{pmatrix} \tag{7}$$

[9]:
```
import torch
X = torch.tensor([[1.,2.],[3.,4.]])
X.requires_grad_(True)
A = torch.tensor([[2.,3.],[4.,5.],[6.,7.]])
y_11 = (A@X)[0,0]
y_11.backward()
X.grad
```

[9]:
```
tensor([[2., 0.],
        [3., 0.]])
```

显然，上述是符合要求的，我们求得了实值函数的导数。接下来，开始求矩阵值函数 $F(X)$ 的导数。

由先前的讨论我们知道, $\frac{\partial F}{\partial X}$ 中的元素都是相同大小的矩阵, 此时, $\frac{\partial F}{\partial X}$ 可以看作是一个四维的向量.

针对这种情况，pytorch 要求我们传入一个和 $F(X)$ 相同大小的矩阵 *gradient*, 然后在求导时, 使用这个矩阵点乘 $\frac{\partial F}{\partial X}$ 中的元素, 最后将结果相加返回, 即

$$G = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \rightarrow \mathrm{X.grad} = \frac{\mathrm{f}_{11}}{\mathrm{X}}$$

$$G = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \rightarrow \mathrm{X.grad} = \frac{\mathrm{f}_{11}}{\mathrm{X}} + \frac{\mathrm{f}_{21}}{\mathrm{X}}$$

若还不清楚，可以先手动算出每一个 $\frac{\partial f_{ij}}{\partial X}$, 然后调整 G 来感受一下

[10]:
```python
import torch
x = torch.tensor([[1.,2.],[3.,4.]])
x.requires_grad_(True)
A = torch.tensor([[2.,3.],[4.,5.],[6.,7.]])
Y = A@X
G = torch.tensor([[0,0],[0,1],[0,0]])
Y.backward(G)
X.grad
```

[10]:
```
tensor([[2., 4.],
        [3., 5.]])
```