ningz Lv1
2018年09月14日 阅读 1247

PyTorch 自动求导机制（2）

torch.autograd.backward

当进行如下操作时，`''RuntimeError: grad can be implicitly created only for scalar outputs''` 的错误。

```
In [1]: import torch
        a = torch.tensor([[1,2],[3,4]], requires_grad=True)
        b = torch.tensor([[1,2,3],[4,5,6]], requires_grad=True)
        c = a.mm(b)
        c.backward()

---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-1-79c8d1c06ff7> in <module>()
      3 b = torch.tensor([[1,2,3],[4,5,6]], requires_grad=True)
      4 c = a.mm(b)
----> 5 c.backward()

~/anaconda3/lib/python3.6/site-packages/torch/tensor.py in backward(self, gradient, retain_graph, create_graph)
     91                 products. Defaults to ``False``.
     92         """
---> 93         torch.autograd.backward(self, gradient, retain_graph, create_graph)
     94 
     95     def register_hook(self, hook):

~/anaconda3/lib/python3.6/site-packages/torch/autograd/__init__.py in backward(tensors, grad_tensors, retain_graph, create_graph, grad_variables)
     81         grad_tensors = list(grad_tensors)
     82 
---> 83     grad_tensors = _make_grads(tensors, grad_tensors)
     84     if retain_graph is None:
     85         retain_graph = create_graph

~/anaconda3/lib/python3.6/site-packages/torch/autograd/__init__.py in _make_grads(outputs, grads)
     25             if out.requires_grad:
     26                 if out.numel() != 1:
---> 27                     raise RuntimeError("grad can be implicitly created only for scalar outputs")
     28                 new_grads.append(torch.ones_like(out))
     29             else:

RuntimeError: grad can be implicitly created only for scalar outputs
```

当改为：

```
In [2]: c.backward(torch.ones_like(c))

In [3]: a.grad
Out[3]: tensor([[  6,  15],
                [  6,  15]])

In [4]: b.grad
Out[4]: tensor([[ 4,  4,  4],
                [ 6,  6,  6]])
```

没有报错。

假设 $\mathbf{c}, \mathbf{a}, \mathbf{b}$ 分别是大小为 $m \times n, m \times k, k \times n$ 的矩阵，且有 $\mathbf{c}^{m\times n} = \mathbf{a}^{m\times k} \times \mathbf{b}^{k\times n}$，即：

$$\begin{bmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,k} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,k} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{k,1} & \cdots & b_{k,n} \end{bmatrix} \tag{1}$$

PyTorch的自动求导过程如下：

$$\frac{\partial c_{1,1}}{\partial \mathbf{b}^{k\times n}} = \begin{bmatrix} \frac{\partial c_{1,1}}{\partial b_{1,1}} & \cdots & \frac{\partial c_{1,1}}{\partial b_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial c_{1,1}}{\partial b_{k,1}} & \cdots & \frac{\partial c_{1,1}}{\partial b_{k,n}} \end{bmatrix} \tag{2}$$

$$\nabla_{\mathbf{b}^{k\times n}} \mathbf{c} = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{\partial c_{i,j}}{\partial \mathbf{b}^{k\times n}} \tag{3}$$

回到最开始的代码，有

$$\mathbf{a} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \tag{4}$$

$$\mathbf{b} = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \tag{5}$$

$$\mathbf{c} = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \end{bmatrix}$$

$$c_{1,1} = a_{1,1}b_{1,1} + a_{1,2}b_{2,1}, \tag{7}$$

$$c_{1,2} = a_{1,1}b_{1,2} + a_{1,2}b_{2,2}, \tag{8}$$

$$c_{1,3} = a_{1,1}b_{1,3} + a_{1,2}b_{2,3}, \tag{9}$$

$$c_{2,1} = a_{2,1}b_{1,1} + a_{2,2}b_{2,1}, \tag{10}$$

$$c_{2,2} = a_{2,1}b_{1,2} + a_{2,2}b_{2,2}, \tag{11}$$

$$c_{2,3} = a_{2,1}b_{1,3} + a_{2,2}b_{2,3}, \tag{12}$$

所以根据(3)式有：

$$\frac{\partial c_{1,1}}{\partial \mathbf{b}} = \begin{bmatrix} a_{1,1} & 0 & 0 \\ a_{1,2} & 0 & 0 \end{bmatrix}, \tag{13}$$

$$\frac{\partial c_{1,2}}{\partial \mathbf{b}} = \begin{bmatrix} 0 & a_{1,1} & 0 \\ 0 & a_{1,2} & 0 \end{bmatrix}, \tag{14}$$

$$\frac{\partial c_{1,3}}{\partial \mathbf{b}} = \begin{bmatrix} 0 & 0 & a_{1,1} \\ 0 & 0 & a_{1,2} \end{bmatrix}, \tag{15}$$

$$\frac{\partial c_{2,1}}{\partial \mathbf{b}} = \begin{bmatrix} a_{2,1} & 0 & 0 \\ a_{2,2} & 0 & 0 \end{bmatrix}, \tag{16}$$

$$\frac{\partial c_{2,2}}{\partial \mathbf{b}} = \begin{bmatrix} 0 & a_{2,1} & 0 \\ 0 & a_{2,2} & 0 \end{bmatrix}, \tag{17}$$

$$\frac{\partial c_{2,3}}{\partial \mathbf{b}} = \begin{bmatrix} 0 & 0 & a_{2,1} \\ 0 & 0 & a_{2,2} \end{bmatrix}, \tag{18}$$

将式(13)~(18)加起来即得到 `b.grad` ，同理可得到 `a.grad` 。 其中 `c.backward(torch.ones_like(c))` 中 `backward()` 的参数是与$\mathbf{c}^{m\times n}$ 大小相同且全为1的矩阵，其中矩阵每个位置的值**【相对应】**的是式(13)~(18)的系数。 即：

$$b.grad = 1\times(13) + 1\times(14) + 1\times(15) + 1\times(16) + 1\times(17) + 1\times(18) = \begin{bmatrix} 4 & 4 & 4 \\ 6 & 6 & 6 \end{bmatrix}$$

【相应的】如果 `c.backward()` 的【参数】为任意与矩阵$\mathbf{c}^{m\times n}$ 【形状一致】的矩阵都可，所得到
各元素梯度乘上对应位置的系数即可。