4.6 GPU计算

到目前为止，我们一直在使用CPU计算。对复杂的神经网络和大规模的数据来说，使用CPU来计算可能不够高效。在本节中，我们将介绍如何使用单块NVIDIA GPU来计算。所以需要确保已经安装好了PyTorch GPU版本。准备工作都完成后，下面就可以通过 `nvidia-smi` 命令来查看显卡信息了。

```
!nvidia-smi  # 对Linux/macOS用户有效
```

输出：

```
Sun Mar 17 14:59:57 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.48                 Driver Version: 390.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1050    Off  | 00000000:01:00.0 Off |                  N/A |
| 20%   36C    P5    N/A /  75W |   1223MiB /  2000MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1235      G   /usr/lib/xorg/Xorg                           434MiB |
|    0      2095      G   compiz                                       163MiB |
|    0      2660      G   /opt/teamviewer/tv_bin/TeamViewer              5MiB |
|    0      4166      G   /proc/self/exe                               416MiB |
|    0     13274      C   /home/tss/anaconda3/bin/python               191MiB |
+-----------------------------------------------------------------------------+
```

可以看到我这里只有一块GTX 1050，显存一共只有2000M（太惨了😭）。

4.6.1 计算设备

PyTorch可以指定用来存储和计算的设备，如使用内存的CPU或者使用显存的GPU。默认情况下，PyTorch会将数据创建在内存，然后利用CPU来计算。

用 `torch.cuda.is_available()` 查看GPU是否可用:

```
import torch
from torch import nn

torch.cuda.is_available() # 输出 True
```

查看GPU数量:

```
torch.cuda.device_count() # 输出 1
```

查看当前GPU索引号，索引号从0开始:

```
torch.cuda.current_device() # 输出 0
```

根据索引号查看GPU名字:

```
torch.cuda.get_device_name(0) # 输出 'GeForce GTX 1050'
```

4.6.2 `Tensor` 的GPU计算

默认情况下， `Tensor` 会被存在内存上。因此，之前我们每次打印 `Tensor` 的时候看不到GPU相关标识。

```
x = torch.tensor([1, 2, 3])
x
```

输出:

```
tensor([1, 2, 3])
```

使用 `.cuda()` 可以将CPU上的 `Tensor` 转换（复制）到GPU上。如果有多块GPU，我们用 `.cuda(i)` 来表示第 i 块GPU及相应的显存（i从0开始）且 `cuda(0)` 和 `cuda()` 等价。

```
x = x.cuda(0)
x
```

输出：

```
tensor([1, 2, 3], device='cuda:0')
```

我们可以通过 `Tensor` 的 `device` 属性来查看该 `Tensor` 所在的设备。

```
x.device
```

输出：

```
device(type='cuda', index=0)
```

我们可以直接在创建的时候就指定设备。

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

x = torch.tensor([1, 2, 3], device=device)
# or
x = torch.tensor([1, 2, 3]).to(device)
x
```

输出：

```
tensor([1, 2, 3], device='cuda:0')
```

如果对在GPU上的数据进行运算，那么结果还是存放在GPU上。

```
y = x**2
y
```

输出：

```
tensor([1, 4, 9], device='cuda:0')
```

需要注意的是，存储在不同位置中的数据是不可以直接进行计算的。即存放在CPU上的数据不可以直接与存放在GPU上的数据进行运算，位于不同GPU上的数据也是不能直接进行计算的。

```
z = y + x.cpu()
```

会报错:

```
RuntimeError: Expected object of type torch.cuda.LongTensor but found type torch.Long
```

4.6.3 模型的GPU计算

同 `Tensor` 类似，PyTorch模型也可以通过 `.cuda` 转换到GPU上。我们可以通过检查模型的参数的 `device` 属性来查看存放模型的设备。

```
net = nn.Linear(3, 1)
list(net.parameters())[0].device
```

输出：

```
device(type='cpu')
```

可见模型在CPU上，将其转换到GPU上:

```
net.cuda()
list(net.parameters())[0].device
```

输出:

```
device(type='cuda', index=0)
```

同样的，我么需要保证模型输入的 `Tensor` 和模型都在同一设备上，否则会报错。

```
x = torch.rand(2,3).cuda()
net(x)
```

输出:

```
tensor([[-0.5800],
        [-0.2995]], device='cuda:0', grad_fn=<ThAddmmBackward>)
```

小结

- PyTorch可以指定用来存储和计算的设备，如使用内存的CPU或者使用显存的GPU。在默认情况下，PyTorch会将数据创建在内存，然后利用CPU来计算。
- PyTorch要求计算的所有输入数据都在内存或同一块显卡的显存上。