链滴

# pytorch 入门笔记 -05- 数据并行

作者：zyk

原文链接：https://ld246.com/article/1639724622060

来源网站：链滴

# 数据并行

在这个教程里，我们将学习如何使用 DataParallel 来使用多 GPU。

PyTorch 非常容易就可以使用多 GPU，用如下方式把一个模型放到 GPU 上：

```
device = torch.device("cuda:0")
model.to(device)
```

GPU：然后复制所有的张量到 GPU 上：

```
mytensor = my_tensor.to(device)
```

请注意，只调用 my_tensor.to(device) 并没有复制张量到 GPU 上，而是返回了一个副本 。所以你要把它赋值给一个新的张量并在 GPU 上使用这个张量。

在多 GPU 上执行前向和反向传播是自然而然的事。

但是 PyTorch 默认将只使用一个 GPU。

使用 DataParallel 可以轻易的让模型并行运行在多个 GPU 上。

```
model = nn.DataParallel(model)
```

这才是这篇教程的核心，接下来我们将更详细的介绍它。

# 导入和参数

导入 Pytorch 模块和定义参数

```
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

input_size = 5
output_size = 2

batch_size = 30
data_size = 100
```

Device

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

print(device)
```

运行结果为：

```
cuda:0
```

# 虚拟数据集

制作一个虚拟（随机）数据集，你只需实现 __getitem__

```
class RandomDataset(Dataset):
    def __init__(self, size, length):
        self.len = length
        self.data = torch.randn(length, size)

    def __getitem__(self, index):
        return self.data[index]

    def __len__(self):
        return self.len


rand_loader = DataLoader(dataset=RandomDataset(
    input_size, data_size), batch_size=batch_size, shuffle=True)
```

# 简单模型

作为演示，我们的模型只接受一个输入，执行一个线性操作，然后得到结果。说明：DataParallel 能任何模型（CNN，RNN，Capsule Net 等）上使用。

```
class Model(nn.Module):
    def __init__(self, input_size, output_size):
        super(Model, self).__init__()
        self.fc = nn.Linear(input_size, output_size)

    def forward(self, input):
        output = self.fc(input)
        print("\tIn Models: input size", input.size(),
              "output size", output.size())

        return output
```

# 创建一个模型和数据并行

首先，我们需要创建一个模型实例和检测我们是否有多个 GPU。若有多个 GPU，使用 nn.DataParallel 来包装我们的模型。然后通过 model.to(device) 把模型放到 GPU 上。

```
model = Model(input_size, output_size)
if torch.cuda.device_count() > 1:
    print("Let's use", torch.cuda.device_count(), "GPUs")
    model = nn.DataParallel(model)

model.to(device)
print(model)
```

运行结果为：

```
Model(
    (fc): Linear(in_features=5, out_features=2, bias=True)
)
```

# 运行模型

现在可以看到输入和输出张量的大小。

```
for data in rand_loader:
    input = data.to(device)
    output = model(input)
    print("Outside: input size", input.size(), "output_size", output.size())
```

运行结果为：

```
In Models: input size torch.Size([30, 5]) output size torch.Size([30, 2])
Outside: input size torch.Size([30, 5]) output_size torch.Size([30, 2])
In Models: input size torch.Size([30, 5]) output size torch.Size([30, 2])
Outside: input size torch.Size([30, 5]) output_size torch.Size([30, 2])
In Models: input size torch.Size([30, 5]) output size torch.Size([30, 2])
Outside: input size torch.Size([30, 5]) output_size torch.Size([30, 2])
In Models: input size torch.Size([10, 5]) output size torch.Size([10, 2])
Outside: input size torch.Size([10, 5]) output_size torch.Size([10, 2])
```