4.5 读取和存储

到目前为止，我们介绍了如何处理数据以及如何构建、训练和测试深度学习模型。然而在实际中，我们有时需要把训练好的模型部署到很多不同的设备。在这种情况下，我们可以把内存中训练好的模型参数存储在硬盘上供后续读取使用。

4.5.1 读写 `Tensor`

我们可以直接使用 `save` 函数和 `load` 函数分别存储和读取 `Tensor` 。 `save` 使用Python的pickle实用程序将对象进行序列化，然后将序列化的对象保存到disk，使用 `save` 可以保存各种对象,包括模型、张量和字典等。而 `load` 使用pickle unpickle工具将pickle的对象文件反序列化为内存。

下面的例子创建了 `Tensor` 变量 `x` ，并将其存在文件名同为 `x.pt` 的文件里。

```
import torch
from torch import nn

x = torch.ones(3)
torch.save(x, 'x.pt')
```

然后我们将数据从存储的文件读回内存。

```
x2 = torch.load('x.pt')
x2
```

输出：

```
tensor([1., 1., 1.])
```

我们还可以存储一个 `Tensor` 列表并读回内存。

```
y = torch.zeros(4)
torch.save([x, y], 'xy.pt')
```

```
xy_list = torch.load('xy.pt')
xy_list
```

输出：

```
[tensor([1., 1., 1.]), tensor([0., 0., 0., 0.])]
```

存储并读取一个从字符串映射到 `Tensor` 的字典。

```
torch.save({'x': x, 'y': y}, 'xy_dict.pt')
xy = torch.load('xy_dict.pt')
xy
```

输出：

```
{'x': tensor([1., 1., 1.]), 'y': tensor([0., 0., 0., 0.])}
```

4.5.2 读写模型

4.5.2.1 `state_dict`

在PyTorch中， `Module` 的可学习参数(即权重和偏差)，模块模型包含在参数中(通过 `model.parameters()` 访问)。 `state_dict` 是一个从参数名称隐射到参数 `Tesnor` 的字典对象。

```
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.hidden = nn.Linear(3, 2)
        self.act = nn.ReLU()
        self.output = nn.Linear(2, 1)

    def forward(self, x):
        a = self.act(self.hidden(x))
```

```
        return self.output(a)

net = MLP()
net.state_dict()
```

输出：

```
OrderedDict([('hidden.weight', tensor([[ 0.2448,  0.1856, -0.5678],
                      [ 0.2030, -0.2073, -0.0104]])),
             ('hidden.bias', tensor([-0.3117, -0.4232])),
             ('output.weight', tensor([[-0.4556,  0.4084]])),
             ('output.bias', tensor([-0.3573]))])
```

注意，只有具有可学习参数的层(卷积层、线性层等)才有 `state_dict` 中的条目。优化器(`optim`)也有一个 `state_dict` ，其中包含关于优化器状态以及所使用的超参数的信息。

```
optimizer = torch.optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
optimizer.state_dict()
```

输出：

```
{'param_groups': [{'dampening': 0,
   'lr': 0.001,
   'momentum': 0.9,
   'nesterov': False,
   'params': [4736167728, 4736166648, 4736167368, 4736165352],
   'weight_decay': 0}],
 'state': {}}
```

4.5.2.2 保存和加载模型

PyTorch中保存和加载训练模型有两种常见的方法:

1. 仅保存和加载模型参数(`state_dict`);
2. 保存和加载整个模型。

1. 保存和加载 `state_dict` (推荐方式)

保存:

```
torch.save(model.state_dict(), PATH) # 推荐的文件后缀名是pt或pth
```

加载:

```
model = TheModelClass(*args, **kwargs)
model.load_state_dict(torch.load(PATH))
```

2. 保存和加载整个模型

保存:

```
torch.save(model, PATH)
```

加载:

```
model = torch.load(PATH)
```

我们采用推荐的方法一来实验一下:

```
X = torch.randn(2, 3)
Y = net(X)

PATH = "./net.pt"
torch.save(net.state_dict(), PATH)

net2 = MLP()
net2.load_state_dict(torch.load(PATH))
Y2 = net2(X)
Y2 == Y
```

输出：

```
tensor([[1],
        [1]], dtype=torch.uint8)
```

因为这 `net` 和 `net2` 都有同样的模型参数，那么对同一个输入 `X` 的计算结果将会是一样的。上面的输出也验证了这一点。

此外，还有一些其他使用场景，例如GPU与CPU之间的模型保存与读取、使用多块GPU的模型的存储等等，使用的时候可以参考**官方文档**。

小结

- 通过 `save` 函数和 `load` 函数可以很方便地读写 `Tensor` 。
- 通过 `save` 函数和 `load_state_dict` 函数可以很方便地读写模型的参数。