torch_nn_flatten

March 2, 2021

Let's think that we have code

[27]:
```
import torch

a = torch.rand(1024, 100)

model = torch.nn.Sequential(
    torch.nn.Linear(100, 50),
    torch.nn.ReLU(),
    torch.nn.Linear(50, 1),
)

print(model(a).shape)
```

```
torch.Size([1024, 1])
```

The shape would be `torch.Size([1024, 1])`, but sometimes it would be more convenient to have a shape `torch.Size([1024])` instead. If we had `torch.nn.Flatten`, we can simply do:

[28]:
```
import torch

a = torch.rand(1024, 100)

model = torch.nn.Sequential(
    torch.nn.Linear(100, 50),
    torch.nn.ReLU(),
    torch.nn.Linear(50, 1),
    torch.nn.Flatten(start_dim=0)
)

print(model(a).shape)
```

```
torch.Size([1024])
```

Next, we see the difference between these two kinds of dimensions.

[29]:
```
x = torch.rand(1024)
print(x)
print(x.shape)
```

```
tensor([0.5950, 0.5283, 0.3729,  …, 0.1043, 0.9230, 0.9339])
torch.Size([1024])
```

[30]:
```
y = torch.rand(1024, 1)
print(y)
print(y.shape)
```

```
tensor([[0.3684],
        [0.7169],
        [0.7469],
        …,
        [0.9903],
        [0.2621],
        [0.2263]])
torch.Size([1024, 1])
```

[ ]:
```

```