June 8, 2024

[ ]:
```python
import torch
from torch.nn.parameter import Parameter

m = torch.nn.Linear(2,3)  #
w=torch.tensor([[0.1,0.15],[0.2,0.25],[0.3,0.35]]  )   #     tensor
m.weight = Parameter(w)   #w               m.weight = torch.normal(0, 0.01,␣
 ↪size=(3,2), requires_grad=True)
b = torch.tensor([0.35,0.35,0.35])
m.bias = Parameter(b)
print(m.weight)
print(m.bias)
```

```
Parameter containing:
tensor([[0.1000, 0.1500],
        [0.2000, 0.2500],
        [0.3000, 0.3500]], requires_grad=True)
Parameter containing:
tensor([0.3500, 0.3500, 0.3500], requires_grad=True)
```

[ ]:
```python
import torch.nn.functional



x = torch.tensor([5.0,10.0])
x = m(x)
print(x)
x = torch.nn.functional.sigmoid(x)
print(x)
```

```
tensor([2.3500, 3.8500, 5.3500], grad_fn=<AddBackward0>)
tensor([0.9129, 0.9792, 0.9953], grad_fn=<SigmoidBackward0>)
```

```
/Users/lihuitao/anaconda3/envs/fasternet/lib/python3.9/site-
packages/torch/nn/functional.py:1944: UserWarning: nn.functional.sigmoid is
deprecated. Use torch.sigmoid instead.
  warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid
instead.")
```

[ ]:
```python
m_2 = torch.nn.Linear(3,2)  #
w=torch.tensor([[0.4,0.5,0.6],[0.45,0.55,0.65]] )   #     tensor
```

```
m_2.weight = Parameter(w)   #w               m.weight = torch.normal(0, 0.01,␣
 ↪size=(3,2), requires_grad=True)
b = torch.tensor([0.65,0.65])
m_2.bias = Parameter(b)
print(m_2.weight)
print(m_2.bias)
```

```
Parameter containing:
tensor([[0.4000, 0.5000, 0.6000],
        [0.4500, 0.5500, 0.6500]], requires_grad=True)
Parameter containing:
tensor([0.6500, 0.6500], requires_grad=True)
```

[ ]:
```
x = m_2(x)
print(x)
x = torch.nn.functional.sigmoid(x)
print(x)
```

```
tensor([2.1019, 2.2463], grad_fn=<AddBackward0>)
tensor([0.8911, 0.9043], grad_fn=<SigmoidBackward0>)
```

[ ]:
```
m_3 = torch.nn.Linear(2,1)  #
w=torch.tensor([[0.7,0.75]] )   #      tensor
m_3.weight = Parameter(w)   #w               m.weight = torch.normal(0, 0.01,␣
 ↪size=(3,2), requires_grad=True)
b = torch.tensor([1.0])
m_3.bias = Parameter(b)
print(m_3.weight)
print(m_3.bias)
```

```
Parameter containing:
tensor([[0.7000, 0.7500]], requires_grad=True)
Parameter containing:
tensor([1.], requires_grad=True)
```

[ ]:
```
x = m_3(x)
print(x)
x = torch.nn.functional.relu(x)
print(x)
```

```
tensor([2.3020], grad_fn=<AddBackward0>)
tensor([2.3020], grad_fn=<ReluBackward0>)
```

[ ]:
```
```