bib34-saba-attar-dl-lab-1-pytorch

August 27, 2024

[1]:
```
import torch
import numpy as np
```

[3]:
```
#Step 1- using data
data = [
    [1,2],
    [3,4]
]
x_data = torch.tensor(data)
print(type(x_data))
```

```
<class 'torch.Tensor'>
```

[4]:
```
### 2 - using numpy array
np_array = np.array(data)
x_np = torch.from_numpy(np_array)
print(x_np)
print(type(x_np))
```

```
tensor([[1, 2],
        [3, 4]])
<class 'torch.Tensor'>
```

[5]:
```
### 3 - using another tensor
x_ones = torch.ones_like(x_data)
print("One Tensor: \n",x_ones)
x_rand = torch.rand_like(x_data,dtype=torch.float)
print(x_rand)
```

```
One Tensor:
 tensor([[1, 1],
        [1, 1]])
tensor([[0.5331, 0.4423],
        [0.6126, 0.1981]])
```

[6]:
```
#### more ways to create tensors
shape = (2,3)
random_tensor = torch.rand(shape)
```

```
print(random_tensor)
print(type(random_tensor))
```

```
tensor([[0.0410, 0.5287, 0.6035],
        [0.9921, 0.2403, 0.9445]])
<class 'torch.Tensor'>
```

[7]:
```
ones_tensor = torch.ones(shape)
print(ones_tensor)
print(type(ones_tensor))
```

```
tensor([[1., 1., 1.],
        [1., 1., 1.]])
<class 'torch.Tensor'>
```

[8]:
```
zeros_tensor = torch.zeros(shape)
print(zeros_tensor)
print(type(zeros_tensor))
```

```
tensor([[0., 0., 0.],
        [0., 0., 0.]])
<class 'torch.Tensor'>
```

[17]:
```
tensor = torch.rand(3,4)
print(tensor)
print(tensor.shape)
print(tensor.dtype)
print(tensor.device)
```

```
tensor([[0.0172, 0.8024, 0.4749, 0.6723],
        [0.7810, 0.9619, 0.6055, 0.8907],
        [0.0129, 0.2697, 0.6403, 0.5155]])
torch.Size([3, 4])
torch.float32
cpu
```

[12]:
```
# Tensor Operations
if torch.cuda.is_available():
  tensor = tensor.to('cuda')
  print("Device tensor is stored in ", tensor.device)
```

[13]:
```
# Indexing, Slicing
tensor = torch.ones(4,4)
print(tensor)
print(tensor)
tensor1 = torch.zeros(4,4)
print(tensor1)
tensor2 = torch.cat([tensor,tensor1])
```

```
print(tensor2)
```

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

[14]:
```
# Multiply Operation
tensor.mul(tensor1)
tensor * tensor1
tensor.T
```

[14]:
```
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
```

[15]:
```
# inplace - change the original tensor
tensor.add_(3)
print(tensor)
```

```
tensor([[4., 4., 4., 4.],
        [4., 4., 4., 4.],
        [4., 4., 4., 4.],
        [4., 4., 4., 4.]])
```

[16]:
```
# from tensor to numpy
t = torch.ones(5)
print(t)
n = t.numpy()
print(n)
```

```
print(type(n))
```

```
tensor([1., 1., 1., 1., 1.])
[1. 1. 1. 1. 1.]
<class 'numpy.ndarray'>
```