exam 개인학습

Lec 1

Pytorch의 기본적인 문법을 배운다.

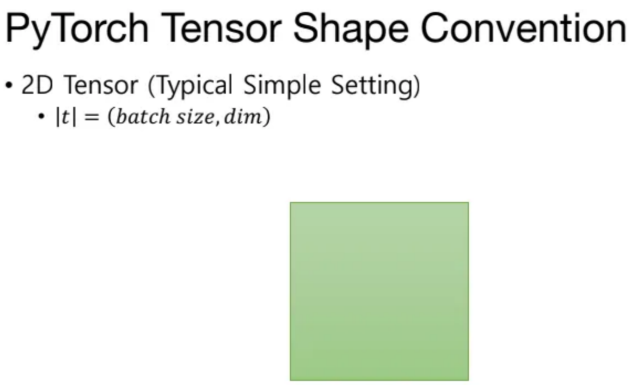
01) Tensor Manipulation 1

01_Numpy Review

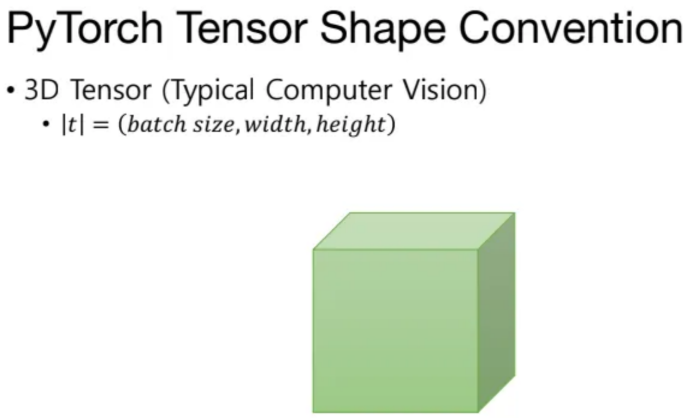
02_Pytorch learning

01) Tensor Manipulation 1

01_Numpy Review



- 일반적으로 2D Tensor의 경우 size는 (batch size, dim)으로 표현된다.



- 3D Tensor의 경우 size = (batch size, width, height)으로 표현된다.

02_Pytorch learning

2D Array with PyTorch

In [9]:
```
t = torch.FloatTensor([[1., 2., 3.],
                       [4., 5., 6.],
                       [7., 8., 9.],
                       [10., 11., 12.]
                      ])
print(t)
```

```
tensor([[ 1.,  2.,  3.],
        [ 4.,  5.,  6.],
        [ 7.,  8.,  9.],
        [10., 11., 12.]])
```

In [10]:
```
print(t.dim())  # rank
print(t.size()) # shape
print(t[:, 1])
print(t[:, 1].size())
print(t[:, :-1])
```

```
2
torch.Size([4, 3])
tensor([ 2.,  5.,  8., 11.])
torch.Size([4])
tensor([[ 1.,  2.],
        [ 4.,  5.],
        [ 7.,  8.],
        [10., 11.]])
```

- Tensor를 선언할 때는 torch.타입Tensor형식으로 선언해주면 된다.

  `t = torch.FloatTensor(array)`

  `t.dim()` = Rank

  `t.size()` = shape

Frequently Used Operations in PyTorch

Mul vs. Matmul

In [13]:
```
print()
print('-------------')
print('Mul vs Matmul')
print('-------------')
m1 = torch.FloatTensor([[1, 2], [3, 4]])
m2 = torch.FloatTensor([[1], [2]])
print('Shape of Matrix 1: ', m1.shape) # 2 x 2
print('Shape of Matrix 2: ', m2.shape) # 2 x 1
print(m1.matmul(m2)) # 2 x 1

m1 = torch.FloatTensor([[1, 2], [3, 4]])
m2 = torch.FloatTensor([[1], [2]])
print('Shape of Matrix 1: ', m1.shape) # 2 x 2
print('Shape of Matrix 2: ', m2.shape) # 2 x 1
print(m1 * m2) # 2 x 2
print(m1.mul(m2))
```

```
-------------
Mul vs Matmul
-------------
Shape of Matrix 1:  torch.Size([2, 2])
Shape of Matrix 2:  torch.Size([2, 1])
tensor([[ 5.],
        [11.]])
Shape of Matrix 1:  torch.Size([2, 2])
Shape of Matrix 2:  torch.Size([2, 1])
tensor([[1., 2.],
        [6., 8.]])
tensor([[1., 2.],
        [6., 8.]])
```

- m1, m2 Tensor 2개를 선언해주고, 행렬곱을 해주기 위해서는 `m1.matmul(m2)` → **matmul** 을 사용해야 하고, m1 * m2 를 진행하게 되면 **inner product**가 실행 된다.

You can also use `t.mean` for higher rank tensors to get mean of all elements, or mean by particular dimension.

In [19]:
```
t = torch.FloatTensor([[1, 2], [3, 4]])
print(t)
```

```
tensor([[1., 2.],
        [3., 4.]])
```

In [20]:
```
print(t.mean())
print(t.mean(dim=0))
print(t.mean(dim=1))
print(t.mean(dim=-1))
```

```
tensor(2.5000)
tensor([2., 3.])
tensor([1.5000, 3.5000])
tensor([1.5000, 3.5000])
```

- torch의 mean, sum, max, argmax를 method를 통해 쉽게 구할 수 있다.

  방향은 2가지로 dim = 0, dim = 1이 있다.

  `t.mean(dim=0)` 을 하게 되면 column 방향으로 mean 값을 구하게 된다.

  반대로 `t.mean(dim = 1)` 을 하게 되면 row 방향으로 mean 값을 구하게 된다.

  즉, dim = 0 → column 방향으로 수행, dim = 1 → row 방향으로 수행을 의미하게 된다.

Max and Argmax

In [23]:
```
t = torch.FloatTensor([[1, 2], [3, 4]])
print(t)
```

```
tensor([[1., 2.],
        [3., 4.]])
```

The `max` operator returns one value if it is called without an argument.

In [24]:
```
print(t.max()) # Returns one value: max
```

```
tensor(4.)
```

The `max` operator returns 2 values when called with dimension specified. The first value is the maximum value, and the second value is the argmax: the index of the element with maximum value.

In [25]:
```
print(t.max(dim=0)) # Returns two values: max and argmax
print('Max: ', t.max(dim=0)[0])
print('Argmax: ', t.max(dim=0)[1])
```

```
(tensor([3., 4.]), tensor([1, 1]))
Max:  tensor([3., 4.])
Argmax:  tensor([1, 1])
```

In [26]:
```
print(t.max(dim=1))
print(t.max(dim=-1))
```

```
(tensor([2., 4.]), tensor([1, 1]))
(tensor([2., 4.]), tensor([1, 1]))
```

- `t.max(dim = 0)` 을 수행하게 되면 column 방향으로 수행하여 1,3 중의 최대값, 2,4 중의 최대값을 구해주게 된다.
- `t.max(dim = 0)[0]` 0번째 인덱스에는 최대값이 들어 있고, `t.max(dim = 0)[1]` 1번째 인덱스에는 최대값의 인덱스가 들어 있다.
- `Argmax`는 최대값의 인덱스를 구해주는 method로 자주 사용하게 된다.