All is about Tensor

python	PyTorch
Int	IntTensor of size()
float	FloatTensor of size()
Int array	IntTensor of size [d1, d2 ,...]
Float array	FloatTensor of size [d1, d2, ...]
string	--

How to denote string

- One – hot
  - [0, 1, 0, 0, …]

- Embedding
  - Word2vec
  - glove

Data type

Data type	dtype		CPU tensor	GPU tensor
32-bit floating point	`torch.float32`	or `torch.float`	`torch.FloatTensor`	`torch.cuda.FloatTensor`
64-bit floating point	`torch.float64`	or `torch.double`	`torch.DoubleTensor`	`torch.cuda.DoubleTensor`
16-bit floating point	`torch.float16`	or `torch.half`	`torch.HalfTensor`	`torch.cuda.HalfTensor`
8-bit integer (unsigned)	`torch.uint8`		`torch.ByteTensor`	`torch.cuda.ByteTensor`
8-bit integer (signed)	`torch.int8`		`torch.CharTensor`	`torch.cuda.CharTensor`
16-bit integer (signed)	`torch.int16`	or `torch.short`	`torch.ShortTensor`	`torch.cuda.ShortTensor`
32-bit integer (signed)	`torch.int32`	or `torch.int`	`torch.IntTensor`	`torch.cuda.IntTensor`
64-bit integer (signed)	`torch.int64`	or `torch.long`	`torch.LongTensor`	`torch.cuda.LongTensor`

Type check

```
 1 In [5]: a = torch.randn(2, 3)
 2
 3 In [6]: a.type()
 4 Out[6]: 'torch.FloatTensor'
 5
 6 In [7]: type(a)
 7 Out[7]: torch.Tensor
 8
 9 In [8]: isinstance(a, torch.FloatTensor)
10 Out[8]: True
```

```
1 In [21]: isinstance(data, torch.cuda.DoubleTensor)
2 Out[21]: False
3
4 In [22]: data=data.cuda()
5
6 In [23]: isinstance(data, torch.cuda.DoubleTensor)
7 Out[23]: True
```

Dimension 0 / rank 0

```
1 In [3]: torch.tensor(1.)
2 Out[3]: tensor(1.)
3
4 In [3]: torch.tensor(1.3)
5 Out[3]: tensor(1.300)
```

Loss

Dim 0

```
1 In [25]: a=torch.tensor(2.2)
2
3 In [26]: a.shape
4 Out[26]: torch.Size([])
5
6 In [27]: len(a.shape)
7 Out[27]: 0
8
9 In [28]: a.size()
10 Out[28]: torch.Size([])
```

Dim 1 / rank 1

Bias

Linear Input

```
1 In [8]: torch.tensor([1.1])
2 Out[8]: tensor([1.1000])
3
4 In [9]: torch.tensor([1.1, 2.2])
5 Out[9]: tensor([1.1000, 2.2000])
6
7 In [10]: torch.FloatTensor(1)
8 Out[10]: tensor([3.2239e-25])
9
10 In [11]: torch.FloatTensor(2)
11 Out[11]: tensor([3.2239e-25, 4.5915e-41])
12
13 In [12]: data = np.ones(2)
14
15 In [13]: data
16 Out[13]: array([ 1.,  1.])
17
18 In [14]: torch.from_numpy(data)
19 Out[14]: tensor([1., 1.], dtype=torch.float64)
```

Dim 1

```
1 In [29]: a=torch.ones(2)
2
3 In [30]: a.shape
4 Out[30]: torch.Size([2])
```

Dim 2

Linear Input batch

```
 1 In [31]: a=torch.randn(2,3)
 2
 3 In [32]: a
 4 Out[32]:
 5 tensor([[-0.4423,  0.5949,  1.1440],
 6         [-2.0935,  0.2051,  1.2781]])
 7
 8 In [33]: a.shape
 9 Out[33]: torch.Size([2, 3])
10
11 In [34]: a.size(0)
12 Out[34]: 2
13
14 In [35]: a.size(1)
15 Out[35]: 3
16
17 In [38]: a.shape[1]
18 Out[38]: 3
```

Dim 3

RNN Input Batch

```
1 In [39]: a=torch.rand(1,2,3)
2
3 In [40]: a
4 Out[40]:
5 tensor([[[0.0764, 0.2590, 0.9816],
6          [0.6798, 0.1568, 0.7919]]])
7
8 In [41]: a.shape
9 Out[41]: torch.Size([1, 2, 3])
10
11 In [42]: a[0]
12 Out[42]:
13 tensor([[0.0764, 0.2590, 0.9816],
14         [0.6798, 0.1568, 0.7919]])
15
16 In [43]: list(a.shape)
17 Out[43]: [1, 2, 3]
```

Dim 4

CNN:
[b, c, h, w]

```
1 In [44]: a=torch.rand(2,3,28,28)
2
3 In [45]: a
4 Out[45]:
5 tensor([[[[0.0509, 0.0420, 0.2934,  ..., 0.6700, 0.1302, 0.9558],
6           [0.9358, 0.7044, 0.6030,  ..., 0.4887, 0.7318, 0.2061],
7           [0.8381, 0.8006, 0.0413,  ..., 0.8347, 0.7955, 0.0314],
8           ...,
9
10          ...,
11          [0.3959, 0.1904, 0.4436,  ..., 0.1279, 0.8817, 0.1984],
12          [0.8796, 0.7907, 0.4319,  ..., 0.1975, 0.0611, 0.1149],
13          [0.3238, 0.4519, 0.0493,  ..., 0.6546, 0.8963, 0.4967]]]])
14
15 In [46]: a.shape
16 Out[46]: torch.Size([2, 3, 28, 28])
```

Mixed

```
 1 In [46]: a.shape
 2 Out[46]: torch.Size([2, 3, 28,
 3 28])
 4 In [47]: a.numel()
 5 Out[47]: 4704
 6
 7 In [48]: a.dim()
 8 Out[48]: 4
 9
10 In [49]: a=torch.tensor(1)
11
12 In [50]: a.dim()
13 Out[50]: 0
```

下一课时

创建Tensor

Thank You.