记录一个切片导致的小问题

- fancy index 和 bool index 返回原数组的一个copy
- slice 返回原数组的一个view，这点和列表不一样.
- https://liwt31.github.io/2019/02/28/numpy_view/ (警惕NumPy切片视图(Slice View)中的“内存泄漏”陷阱)说得太好了，这样memoryview的意义也就可以理解了.
- 可以用numba加速pytorch自定义dataset的数据预处理.

```
import numpy as np
import torch


class Toy1:
    def __init__(self):
        self.x_list = np.zeros((10,10))
    def _pre_process(self,x):
        x += np.random.normal(10,1,size=x.shape)
        return x
    def __getitem__(self,index):
        x = self.x_list[index]
        x = torch.Tensor(x)
        x = self._pre_process(x)
        return x


toy1 = Toy1()


for i in range(10):
    print(toy1[0])
```

输出为：

```
tensor([11.1936,  9.6680, 10.3589,  9.3128, 12.1851,  9.8324, 11.1655,  9.7072,
         9.1300,  9.8151], dtype=torch.float64)
tensor([10.8049, 10.8619,  9.7030, 10.9868,  9.6527,  9.8182, 11.0079, 10.9283,
         9.8499, 10.4441], dtype=torch.float64)
```

```
tensor([ 7.8097,  9.5682,  9.5932,  9.5863,  8.8383,  9.2816, 11.5156,  8.6302,
         8.9394, 10.0598], dtype=torch.float64)
tensor([ 8.1653, 10.6674, 10.4430,  9.8594,  9.1327,  9.4114, 11.4207, 10.0508,
        11.4400,  9.3245], dtype=torch.float64)
tensor([10.2448,  9.6530,  9.7135, 10.2490, 10.7117,  9.0196,  9.7127, 10.0913,
        10.2286, 11.1697], dtype=torch.float64)
tensor([11.4281,  9.4863,  9.3562, 10.6482,  8.8311,  8.5337, 10.7233,  9.5303,
        10.2731, 10.5429], dtype=torch.float64)
tensor([10.5949,  9.5061,  9.5509, 10.3889, 10.5734, 10.5830,  8.3087, 10.3692,
         9.9943,  9.7930], dtype=torch.float64)
tensor([ 9.9605, 10.3794, 10.3654, 10.1450,  9.2584,  9.2737, 10.0656,  9.5827,
        10.2378,  8.1681], dtype=torch.float64)
tensor([ 9.9973,  9.7198,  9.5997,  8.9854, 11.1076,  9.1467, 10.4797,  8.8637,
        10.0195,  9.1357], dtype=torch.float64)
tensor([ 9.7966,  7.8419, 10.1753,  9.9243,  9.9067, 10.3850, 11.5918,  8.0932,
         8.5211,  8.4153], dtype=torch.float64)
```

```
class Toy2:
    def __init__(self):
        self.x_list = np.zeros((10,10))
    def _pre_process(self,x):
        x += np.random.normal(10,1,size=x.shape)
        return x
    def __getitem__(self,index):
        x = self.x_list[index]
        x = self._pre_process(x)
        x = torch.Tensor(x)
        return x


toy2 = Toy2()


for i in range(10):
    print(toy2[0])
```

输出为：

```
tensor([10.4171, 10.9843,  8.0187,  7.5639, 10.5345,  9.5257, 12.2044,  8.4229,
        10.6969,  9.7231])
tensor([19.3723, 20.0729, 19.3233, 18.5441, 20.7856, 18.8565, 21.9052, 20.0978,
        19.8563, 20.9260])
tensor([29.9699, 30.8167, 30.5387, 29.8222, 31.8951, 29.1343, 30.8827, 30.7759,
        29.1300, 31.3557])
tensor([39.3184, 40.6728, 40.7825, 38.5859, 43.7154, 39.2851, 40.2388, 40.5890,
        40.6375, 41.6656])
tensor([49.7643, 48.6515, 52.2233, 47.9980, 53.0037, 50.5212, 49.9058, 51.5314,
        51.2658, 51.3687])
tensor([59.7970, 58.0771, 63.9988, 59.4151, 62.6010, 59.3163, 59.7800, 61.1103,
        61.3739, 62.4363])
tensor([70.6014, 69.6588, 75.8430, 69.8010, 73.7324, 67.8835, 69.4310, 70.7775,
        71.9213, 73.1251])
tensor([81.7533, 79.6750, 85.2563, 79.6907, 83.2710, 78.6026, 79.5761, 80.7719,
        82.3329, 82.9063])
tensor([90.0995, 88.8826, 94.4563, 87.7133, 92.6143, 87.6910, 88.8625, 88.8491,
        92.3851, 93.1900])
tensor([ 99.7364, 100.6963, 104.7957,  97.4885, 103.3829,  99.3400,  98.1868,
         99.1119, 103.9565, 104.5152])
```

```
class Toy3:
    def __init__(self):
        self.x_list = np.zeros((10,10))
    def _pre_process(self,x):
        x += np.random.normal(10,1,size=x.shape)
        return x
    def __getitem__(self,index):
        x = self.x_list[index].copy()
        x = self._pre_process(x)
        x = torch.Tensor(x)
        return x


toy3 = Toy3()


for i in range(10):
    print(toy3[0])
```

输出为：

```
tensor([ 9.2556,  8.6402,  9.7651,  9.7772, 11.2082, 11.1656, 11.5032, 11.0477,
         9.1664, 11.4338])
tensor([ 9.2510, 11.9188,  8.5967, 10.1560, 11.1240,  9.9262, 10.6061, 10.3028,
        10.6116, 11.5722])
tensor([10.1320, 11.3630,  9.4132, 10.1754, 10.2467, 10.2389,  9.5375, 10.8479,
        10.1020, 11.9192])
tensor([ 8.2110,  9.3971, 10.0790, 10.1578,  9.8869,  8.8328,  7.9871,  9.1023,
         8.6052,  8.1066])
tensor([ 9.5481,  9.9247,  9.7306,  8.6508,  9.9780, 10.2113, 10.3141, 10.6318,
         9.2056, 12.7433])
tensor([ 9.5609, 10.0979,  9.8096, 10.5516, 11.7037, 11.5780,  8.1963,  8.3716,
        10.8255, 10.9552])
tensor([10.3634, 11.6631,  9.7124,  9.9065, 10.2127, 12.2161, 13.0025, 11.1271,
         9.4598,  8.6592])
tensor([11.6567, 10.4351, 10.2801, 12.6541,  9.6427,  9.3057, 12.0668,  8.6563,
         9.1362, 11.1149])
tensor([ 7.7418, 11.6473, 10.5215,  9.6892,  9.7522,  8.7857, 11.0384, 10.3307,
        10.4203,  9.7805])
tensor([ 9.5924, 10.4769,  8.3236,  9.8965,  9.8669,  8.4239,  8.6173, 10.5610,
        10.4306, 10.7248])
```

```
a = [1,2,3]
b = a[1:]
b[0] = 100
print(a)
print(b)
```

输出为：

```
[1, 2, 3]
[100, 3]
```