PyConCA2018

November 3, 2018

1 Numpy To PyTorch

```
In [0]: import numpy as np
        import torch
        import time
```

1.1 Numpy

Numpy vs Python List

```
In [4]: given_list = [1, 2, 3]
        new_array = np.array(given_list)

        print(type(new_array))

        # <class 'numpy.ndarray'>
```

```
<class 'numpy.ndarray'>
```

```
In [5]: given_list = [24, 12, 57]
        new_array = np.array(given_list)

        print([x+3 for x in given_list])

        print(new_array+3)
```

```
[27, 15, 60]
[27 15 60]
```

```
In [6]: first_array = np.random.rand(128, 5)

        second_array = np.random.rand(5, 128)

        print(np.matmul(first_array, second_array))
```

```
[[1.14621608 1.98757359 1.64628498 ... 0.81847937 1.2301337  1.14188524]
 [1.37717482 2.64040113 2.52680352 ... 1.31106759 1.83547388 1.3020587 ]
 [0.4858669  0.96603831 1.02596539 ... 0.66230938 0.70703009 0.30740704]
 ...
 [1.56416053 2.74988515 2.79702175 ... 1.90621481 2.27962372 1.3136982 ]
 [1.36626942 2.0879698  1.80864462 ... 1.48238634 1.53634107 1.05710297]
 [0.89681003 1.82609787 1.94508328 ... 0.91342258 1.51778127 0.99043572]]
```

1.2 PyTorch

```
In [7]: torch.Tensor

Out[7]: torch.Tensor
```

Creation Ops

```
In [8]: shape = (2, 3)
        print(np.ones(shape))
        print(torch.ones(shape))

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


In [9]: shape = (3, 3)
        x = 2 * np.ones(shape)
        y = np.eye(shape[0])
        print(x+y)

        x = 2 * torch.ones(shape)
        y = torch.eye(shape[0])
        print(x+y)

[[3. 2. 2.]
 [2. 3. 2.]
 [2. 2. 3.]]
tensor([[3., 2., 2.],
        [2., 3., 2.],
        [2., 2., 3.]])


In [10]: x = np.array([[1, 2], [3, 4]])
         print(x)

         x = torch.tensor([[1, 2], [3, 4]])
         print(x)
```

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

Variable Assignment Numpy supports two styles:

- Compute and assign to a variable using the `assignment` operator.
- Compute and assign value using a function call

```
In [11]: shape = (3, 3)
         x = 2 * np.ones(shape)
         y = np.eye(shape[0])
         y = np.add(x, y)
         print(y)



         x = 2 * np.ones(shape)
         y = np.eye(shape[0])
         np.add(x, y, out=y)
         print(y)
```

```
[[3. 2. 2.]
 [2. 3. 2.]
 [2. 2. 3.]]
[[3. 2. 2.]
 [2. 3. 2.]
 [2. 2. 3.]]
```

```
In [12]: shape = (3, 3)
         x = 2 * torch.ones(shape)
         y = torch.eye(shape[0])
         y = torch.add(x, y)
         print(y)



         x = 2 * torch.ones(shape)
         y = torch.eye(shape[0])
         torch.add(x, y, out=y)
         print(y)
```

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

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

Advanced Indexing

```
In [13]: x = np.arange(10)
         print(x[1:7:2])


         y = np.arange(35).reshape(5,7)
         print(y[1:5:2,::3])
```

```
[1 3 5]
[[ 7 10 13]
 [21 24 27]]
```

```
In [14]: x = torch.arange(10)
         print(x[1:7:2])

         y = torch.arange(35).reshape(5,7)
         print(y[1:5:2,::3])
```

```
tensor([1, 3, 5])
tensor([[ 7, 10, 13],
        [21, 24, 27]])
```

```
In [15]: print(np.random.random(shape))
         torch.rand(shape)
```

```
[[0.30984779 0.37485521 0.26480706]
 [0.09522194 0.18916697 0.7615523 ]
 [0.11044389 0.28357937 0.91886845]]
```

```
Out[15]: tensor([[0.1870, 0.0262, 0.5876],
                 [0.9313, 0.6382, 0.2628],
                 [0.3238, 0.7973, 0.1585]])
```

```
In [16]: x = np.arange(10,1,-1)
         indexing_array = np.array([3,3,-3,8])
         print(x[indexing_array])

         indexing_array = np.array([[1,1],[2,3]])
         print(x[indexing_array])
```

```
[7 7 4 2]
[[9 9]
 [8 7]]
```

```
In [17]: x = torch.arange(10,1,-1)
         indexing_array = torch.tensor([3,3,-3,8])
         print(x[indexing_array])


         indexing_array = torch.tensor([[1,1],[2,3]])
         print(x[indexing_array])
```

```
tensor([7, 7, 4, 2])
tensor([[9, 9],
        [8, 7]])
```

Other comparisons for the APIs

```
In [18]: shape = (2000, 1000)

         np_array = np.ones(shape)
         np.sum(np_array, axis=1)
```

```
Out[18]: array([1000., 1000., 1000., ..., 1000., 1000., 1000.])
```

```
In [19]: torch_array = torch.ones(shape)
         torch.sum(torch_array, dim=1)
```

```
Out[19]: tensor([1000., 1000., 1000.,  ..., 1000., 1000., 1000.])
```

```
In [20]: x = np.linspace(start=10.0, stop=20, num=5)
         print(x)

         x = torch.linspace(start=10, end=20, steps=5)
         print(x)
```

```
[10.  12.5 15.  17.5 20. ]
tensor([10.0000, 12.5000, 15.0000, 17.5000, 20.0000])
```

GPU Acceleration

```
In [21]: %%timeit
         np.random.seed(1)
         n = 10000
         x = np.array(np.random.randn(n,n), dtype = np.float32)
         y = np.matmul(x, x)
```

```
1 loop, best of 3: 29 s per loop
```

```
In [22]: %%timeit
         torch.manual_seed(1)
         n = 10000
         device = torch.device('cuda:0')
         x = torch.rand(n, n, dtype=torch.float32, device=device)
         y = torch.matmul(x, x)
```

```
1 loop, best of 3: 860 ms per loop
```

```
In [23]: %%timeit
         torch.manual_seed(1)
         n = 10000
         x = torch.rand(n, n, dtype=torch.float32)
         y = torch.matmul(x, x)
```

```
1 loop, best of 3: 23.3 s per loop
```

PyTorch to Numpy to PyTorch

```
In [24]: shape = (5, 3)
         numpy_array = np.array(shape)
         torch_array = torch.from_numpy(numpy_array)
         print(torch_array)
         recreated_numpy_array = torch_array.numpy()
         print(recreated_numpy_array)
         if((recreated_numpy_array == numpy_array).all()):
           print("Numpy -> Torch -> Numpy")
```

```
tensor([5, 3])
[5 3]
Numpy -> Torch -> Numpy
```

Tensors to GPU

```
In [0]: tensor = torch_array
```

```
In [0]: gpu_device = torch.device('cuda:0')
        cpu_device = torch.device('cpu')
        tensor_on_gpu = tensor.to(gpu_device)
        tensor_on_cpu = tensor.to(cpu_device)
```

1.3 Pitfalls

Pitfall1

```
In [27]: numpy_array = np.array([1, 2, 3])
         torch_array = torch.from_numpy(numpy_array)
         torch_array[0] = -100
         print(numpy_array[0])
```

```
-100
```

```
In [28]: numpy_array = np.array([1, 2, 3])
         torch_array = torch.Tensor(numpy_array)
         torch_array[0] = -100
         print(numpy_array[0])
```

```
1
```

Pitfall2

```
In [29]: numpy_array = np.array([1, 2, 3])
         torch_array = torch.from_numpy(numpy_array).to(gpu_device)
         torch_array[0] = -100
         print(numpy_array[0])
```

```
1
```

Pitfall3

```
In [30]: torch_array = torch.tensor([1, 2, 3],
                     device = gpu_device)
         numpy_array = torch_array.numpy()
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-30-12c809eb4ad1> in <module>()
          1 torch_array = torch.tensor([1, 2, 3],
          2             device = gpu_device)
    ----> 3 numpy_array = torch_array.numpy()


        TypeError: can't convert CUDA tensor to numpy. Use Tensor.cpu() to copy the tensor to
```

```
In [31]: torch_array = torch.tensor([1, 2, 3],
                                    device = gpu_device)

         numpy_array = torch_array
                     .to(cpu_device)
                     .numpy()


          File "<ipython-input-31-eb25a08989af>", line 5
        .to(cpu_device)
        ^
    IndentationError: unexpected indent


In [32]: shape = (128, 1000)
         np_array = np.random.random(shape)
         np.sum(np_array, axis=1)

Out[32]: array([498.59712162, 502.52849752, 503.57117359, 488.07623026,
                495.6646413 , 501.18841552, 502.73261359, 504.21015344,
                501.86042666, 499.5747481 , 483.84437326, 481.45765204,
                491.02671955, 482.9447564 , 509.91074875, 503.39466817,
                496.14817687, 509.15691603, 491.87977037, 490.99489233,
                499.15861804, 487.3149828 , 506.962333  , 498.32899877,
                519.03384374, 502.41867744, 506.09145746, 519.36073811,
                493.244471  , 500.05835267, 490.06825235, 494.46028917,
                503.26328107, 501.25197965, 508.68742517, 514.95956824,
                521.2921211 , 496.51557246, 498.04266599, 508.44416484,
                509.1711359 , 478.82011639, 504.85985054, 490.21507097,
                499.46736878, 524.60855069, 500.02714249, 512.29558163,
                502.07606393, 497.8230504 , 500.8593151 , 488.50639928,
                504.7286064 , 502.86484579, 498.23609011, 511.69882821,
                504.23075273, 512.16885496, 491.89449766, 496.84376843,
                502.33907818, 516.2477405 , 502.16443656, 505.39996406,
                490.19242331, 512.63794612, 499.74999987, 499.1022014 ,
                502.58554766, 507.40975013, 497.84605395, 501.4238301 ,
                495.33024065, 500.40737249, 494.68165447, 502.56517216,
                507.24752445, 475.3198302 , 490.68972528, 496.04986353,
                487.48407952, 517.57266527, 503.03728608, 502.12876388,
                480.13451861, 509.95604448, 491.79212749, 495.62045963,
                503.92845225, 503.88169354, 495.73011105, 493.87849392,
                487.38705573, 503.02681419, 504.84295737, 500.62651625,
                500.60303453, 489.99515902, 509.559676  , 483.72110276,
                511.77761332, 485.44093909, 495.79816309, 493.71850697,
                513.5217938 , 492.57733115, 493.72782342, 492.12947812,
                502.95766563, 503.28058837, 495.32478734, 499.75906194,
                495.34034223, 512.32393363, 505.14868806, 499.22313741,
```

```
                494.7618941 , 482.17552851, 489.79058029, 487.40383174,
                500.18445541, 514.81150132, 505.59146184, 509.22661368,
                502.22901128, 495.67435816, 480.2240037 , 493.42428088])
```

```
In [33]: shape = (128, 1000)
         torch_array = torch.rand(shape)
         torch.sum(torch_array, dim=1)
```

```
Out[33]: tensor([492.3059, 494.1634, 497.9210, 513.8610, 481.6728, 496.2572, 491.1049,
                 499.0293, 501.8353, 501.8107, 488.3193, 484.0969, 502.3377, 505.6399,
                 505.1918, 499.3962, 491.9419, 498.9113, 495.9724, 501.9955, 511.6537,
                 499.5494, 495.1823, 493.0666, 500.8804, 497.6123, 509.1573, 510.7198,
                 498.3613, 498.5643, 509.0970, 495.3907, 505.9768, 489.7607, 502.9781,
                 491.3438, 509.1869, 501.3050, 489.1332, 482.0582, 503.9869, 489.8748,
                 507.7838, 512.3679, 503.4282, 478.4197, 509.6520, 487.4262, 507.8362,
                 505.2957, 497.6619, 499.1406, 506.4666, 493.7106, 484.3940, 498.6255,
                 507.9043, 500.8365, 498.8157, 505.2602, 495.7344, 501.3155, 510.9013,
                 495.9234, 505.2554, 494.4229, 519.5248, 506.7155, 488.7425, 496.3576,
                 496.3604, 511.3221, 505.2527, 506.4363, 502.2169, 508.3695, 514.0903,
                 482.9781, 508.0064, 505.3618, 488.0728, 500.7408, 487.0506, 504.9445,
                 489.9784, 496.4337, 507.2312, 506.5978, 502.0357, 490.7647, 517.3200,
                 482.3507, 507.0240, 497.7324, 494.9421, 498.1074, 492.7851, 504.1136,
                 493.5891, 496.3753, 498.6391, 509.7771, 500.0313, 515.2452, 506.2283,
                 503.0292, 493.8932, 499.8261, 499.1005, 504.7088, 498.7352, 510.3496,
                 497.5545, 494.7366, 502.9202, 499.0169, 503.1243, 496.9809, 492.3491,
                 496.5404, 493.9608, 503.1400, 504.4873, 492.0404, 494.0301, 494.2773,
                 503.1441, 501.2573])
```

1.4 More Indexing Examples

```
In [34]: x = np.arange(10)
         print(x[2:5])
         print(x[:-7])
         print(x[1:7:2])
         y = np.arange(35).reshape(5,7)
         print(y)
         print(y[1:5:2,::3])
```

```
[2 3 4]
[0 1 2]
[1 3 5]
[[ 0  1  2  3  4  5  6]
 [ 7  8  9 10 11 12 13]
 [14 15 16 17 18 19 20]
 [21 22 23 24 25 26 27]
 [28 29 30 31 32 33 34]]
[[ 7 10 13]
 [21 24 27]]
```

```
In [35]: x = torch.arange(10)
         print(x[2:5])
         print(x[:-7])
         print(x[1:7:2])
         y = torch.arange(35).reshape(5,7)
         print(y)
         print(y[1:5:2,::3])
```

```
tensor([2, 3, 4])
tensor([0, 1, 2])
tensor([1, 3, 5])
tensor([[ 0,  1,  2,  3,  4,  5,  6],
        [ 7,  8,  9, 10, 11, 12, 13],
        [14, 15, 16, 17, 18, 19, 20],
        [21, 22, 23, 24, 25, 26, 27],
        [28, 29, 30, 31, 32, 33, 34]])
tensor([[ 7, 10, 13],
        [21, 24, 27]])
```

```
In [36]: x = np.arange(10,1,-1)
         print(x)
         indexing_array = np.array([3, 3, 1, 8])
         print(x[indexing_array])
         indexing_array = [np.array([3,3,-3,8])]
         print(x[indexing_array])
         indexing_array = np.array([[1,1],[2,3]])
         print(x[indexing_array])
```

```
[10  9  8  7  6  5  4  3  2]
[7 7 9 2]
[7 7 4 2]
[[9 9]
 [8 7]]
```

```
In [37]: x = torch.arange(10,1,-1)
         print(x)
         indexing_array = torch.tensor([3, 3, 1, 8])
         print(x[indexing_array])
         indexing_array = [torch.tensor([3,3,-3,8])]
         print(x[indexing_array])
         indexing_array = torch.tensor([[1,1],[2,3]])
         print(x[indexing_array])
```

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

```
        [8, 7]])
```

```
In [38]: x = np.arange(10,1,-1)
         indexing_array = np.array([3,3,-3,8])
         print(x[indexing_array])

         indexing_array = np.array([[1,1],[2,3]])
         print(x[indexing_array])
```

```
[7 7 4 2]
[[9 9]
 [8 7]]
```

```
In [39]: x = torch.arange(10,1,-1)
         indexing_array = torch.tensor([3,3,-3,8])
         print(x[indexing_array])
         # [7 7 4 2]


         indexing_array = torch.tensor([[1,1],[2,3]])
         print(x[indexing_array])
         # [[9 9]
         # [8 7]]
```

```
tensor([7, 7, 4, 2])
tensor([[9, 9],
        [8, 7]])
```

```
In [40]: shape = (2, 3)
         print(np.ones(shape))
         print(torch.ones(shape))
         # [[1. 1. 1.]
         # [1. 1. 1.]]
```

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