Dot products of ndarray and transposition

November 22, 2017

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

1 One dimensional ndarray

if $\vec{A}$ and $\vec{B}$ are in R^n. The dot product (produit scalaire) is defined by :

$$c = \vec{A} \cdot \vec{B} = \sum_i^n A_i B_i$$

c is a scalar

```
In [97]: A = array([0, 1, 2, 3])
         B = array([1, 2, 3, 4])
         C = A.dot(B)
         print(C)
         print(C.shape)
         print(type(C))
```

```
20
()
<class 'numpy.int64'>
```

2 N dimensional ndarray versus 1 dimensional ndarray

A is three dimensional (N=3)

```
In [98]: A = arange(24).reshape(2, 3, 4)
         print(A)
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]
```

```
 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

B has one dimension with, for example, 4 elements.

```
In [101]: B = array([1, 2, 3, 4])
          C = A.dot(B)
          print(C)
          print(C.shape)
```

```
[[ 20  60 100]
 [140 180 220]]
(2, 3)
```

A shape is (2, 3, 4)
B shape is (4,)
C shape is (2, 3) since (4,)dot(4,) is a scalar
Every dot products are done on the last dimension of A through here, the first dimension of B.

3 N dimensional ndarray versus P dimensional ndarray

To be compatible for dot products, the number of elements in the antepenultimate (before last) dimension of B must be the same as the number of elements in the last dimension of A.
if A is the same as before, the antepenultimate dimension of B must contains 4 elements.

```
In [87]: B = arange(8).reshape(4, 2)
         print(B)
```

```
[[0 1]
 [2 3]
 [4 5]
 [6 7]]
```

In this example, the dot product will be done with the two vectors $[0, 2, 4, 6]$ and $[1, 3, 5, 7]$.
A is shape (2, 3, 4) ans B is shape (4, 2).
The resulting shape will be (2, 3, 2), the since the (4,) dot (4,) operation results in a scalar.

```
In [88]: D = A.dot(B)
         print(D)
         print(D.shape)
```

```
[[[ 28  34]
  [ 76  98]
  [124 162]]
```

```
 [[172 226]
  [220 290]
  [268 354]]]
(2, 3, 2)
```

if A is (2, 3, 4) and B is (3, 4, 5), the result will be (2, 3, 3, 5)

```
In [89]: B = arange(60).reshape(3, 4, 5)
         print(B)
```

```
[[[ 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]
  [35 36 37 38 39]]

 [[40 41 42 43 44]
  [45 46 47 48 49]
  [50 51 52 53 54]
  [55 56 57 58 59]]]
```

```
In [90]: C = A.dot(B)
         print(C)
         print(C.shape)
```

```
[[[[  70   76   82   88   94]
   [ 190  196  202  208  214]
   [ 310  316  322  328  334]]

  [[ 190  212  234  256  278]
   [ 630  652  674  696  718]
   [1070 1092 1114 1136 1158]]

  [[ 310  348  386  424  462]
   [1070 1108 1146 1184 1222]
   [1830 1868 1906 1944 1982]]]


 [[[ 430  484  538  592  646]
   [1510 1564 1618 1672 1726]
   [2590 2644 2698 2752 2806]]

  [[ 550  620  690  760  830]
```

```
   [1950 2020 2090 2160 2230]
   [3350 3420 3490 3560 3630]]

  [[ 670  756  842  928 1014]
   [2390 2476 2562 2648 2734]
   [4110 4196 4282 4368 4454]]]]
(2, 3, 3, 5)
```

4 Transposition

A has shape (2, 3, 4)

```
In [91]: print(A)
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

A.T has shape (4, 3, 2)

```
In [92]: print(A.T)
```

```
[[[ 0 12]
  [ 4 16]
  [ 8 20]]

 [[ 1 13]
  [ 5 17]
  [ 9 21]]

 [[ 2 14]
  [ 6 18]
  [10 22]]

 [[ 3 15]
  [ 7 19]
  [11 23]]]
```

```
In [93]: A = arange(120).reshape(2, 3, 4, 5)
```

```
In [94]: print(A.shape)
         print(A.T.shape)
```

```
(2, 3, 4, 5)
(5, 4, 3, 2)
```

```
In [95]: print(A)
         print(A.T)
```

```
[[[[  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]
   [ 35  36  37  38  39]]

  [[ 40  41  42  43  44]
   [ 45  46  47  48  49]
   [ 50  51  52  53  54]
   [ 55  56  57  58  59]]]


 [[[ 60  61  62  63  64]
   [ 65  66  67  68  69]
   [ 70  71  72  73  74]
   [ 75  76  77  78  79]]

  [[ 80  81  82  83  84]
   [ 85  86  87  88  89]
   [ 90  91  92  93  94]
   [ 95  96  97  98  99]]

  [[100 101 102 103 104]
   [105 106 107 108 109]
   [110 111 112 113 114]
   [115 116 117 118 119]]]]
[[[[  0  60]
   [ 20  80]
   [ 40 100]]

  [[  5  65]
   [ 25  85]
   [ 45 105]]

  [[ 10  70]
   [ 30  90]
   [ 50 110]]
  [[ 15  75]
   [ 35  95]
   [ 55 115]]]


 [[[  1  61]
   [ 21  81]
   [ 41 101]]

  [[  6  66]
   [ 26  86]
   [ 46 106]]

  [[ 11  71]
   [ 31  91]
   [ 51 111]]

  [[ 16  76]
   [ 36  96]
   [ 56 116]]]


 [[[  2  62]
   [ 22  82]
   [ 42 102]]

  [[  7  67]
   [ 27  87]
   [ 47 107]]

  [[ 12  72]
   [ 32  92]
   [ 52 112]]

  [[ 17  77]
   [ 37  97]
   [ 57 117]]]


 [[[  3  63]
   [ 23  83]
   [ 43 103]]

  [[  8  68]
   [ 28  88]
   [ 48 108]]
  [[ 13  73]
   [ 33  93]
   [ 53 113]]

  [[ 18  78]
   [ 38  98]
   [ 58 118]]]


 [[[  4  64]
   [ 24  84]
   [ 44 104]]

  [[  9  69]
   [ 29  89]
   [ 49 109]]

  [[ 14  74]
   [ 34  94]
   [ 54 114]]

  [[ 19  79]
   [ 39  99]
   [ 59 119]]]]
```