numpy notes

June 19, 2019

```
In [2]: import numpy as np
```

1 numpy.newaxis

numpy differentiates between arrays an matrices in the sense that a matrix is essentially a 2D-array. This implies that a row vector is not same as a 1D array. We can use numpy.newaxis to convert numpy arrays into row and column vectors.

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

Here 'a' is a 1D numpy array. We can convert it to a column vector as follows:

```
In [12]: ac=a[:, np.newaxis]; print(ac)
```

```
[[1]
 [2]
 [3]]
```

Similarly, we can convert 'a' into a row vector as follows:

```
In [13]: ar=a[np.newaxis,:]; print(ar)
```

```
[[1 2 3]]
```

Note that the operator for matrix multiplication in numpy is given by '@'

```
In [16]: print(ar@ac)
```

```
[[14]]
```

```
In [17]: print(ac@ar)
```

```
[[1 2 3]
 [2 4 6]
 [3 6 9]]
```

We can also use the numpy functions 'dot' and 'matmul' to carry out matrix multiplication. The function 'dot' can be applied to general numpy array and not just 2d matrices.

```
In [18]: amult1=np.matmul(ar,ac); print(amult1)

[[14]]


In [19]: amult2=np.matmul(ac,ar); print(amult2)

[[1 2 3]
 [2 4 6]
 [3 6 9]]
```

In general the command newaxis will increase the dimentionality of an array by 1

```
In [8]: b=np.array([[1,2],[3,4]])

In [9]: print(b)

[[1 2]
 [3 4]]


In [20]: print(b[:,np.newaxis])

[[[1 2]]

 [[3 4]]]


In [21]: print(b[np.newaxis,:])

[[[1 2]
  [3 4]]]
```

2 Concatinating two numpy arrays

```
In [22]: a =np.array([[1,2],[3,4]]); print(a)

[[1 2]
 [3 4]]


In [43]: b =np.array([6,9]); print(b)

[6 9]
```

```
In [34]: ct=np.c_[a,b]; print(ct)

[[1 2 6]
 [3 4 9]]


In [37]: b2 = np.array([[6],[9]]); print(b)

[[6]
 [9]]


In [38]: ct2=np.c_[a,b2]; print(ct2)

[[1 2 6]
 [3 4 9]]


In [40]: b3=np.array([6,7,8]);print(b3)

[6 7 8]


In [41]: ct3=np.c_[a,b3]; print(ct3)


        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-41-2db71b8973b9> in <module>
    ----> 1 ct3=np.c_[a,b3]; print(ct3)


        C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\index_tricks.py in __getitem__(sel
        333                 objs[k] = objs[k].astype(final_dtype)
        334
    --> 335         res = self.concatenate(tuple(objs), axis=axis)
        336
        337         if matrix:


        ValueError: all the input array dimensions except for the concatenation axis must matc


In [48]: b=np.array([[7,6]]);print(b)

[[7 6]]
```

```
In [49]: rt=np.r_[a,b];print(rt)

[[1 2]
 [3 4]
 [7 6]]


In [51]: b2=np.array([7,6]);print(b2)

[7 6]


In [52]: rt2=np.r_[a,b2];print(rt2)


        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-52-3044a273e10b> in <module>
    ----> 1 rt2=np.r_[a,b2];print(rt2)


        C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\index_tricks.py in __getitem__(sel
        333                 objs[k] = objs[k].astype(final_dtype)
        334
    --> 335         res = self.concatenate(tuple(objs), axis=axis)
        336
        337         if matrix:


        ValueError: all the input arrays must have same number of dimensions
```

3 reshape

```
In [4]: a=np.array([1,3,5]);print(a)

[1 3 5]


In [5]: print(a.reshape(-1,1))

[[1]
 [3]
 [5]]


In [6]: b=np.array([[1,5,9]]);print(b)
```

```
[[1 5 9]]
```

```
In [8]: print(b.reshape(-1,1))
```

```
[[1]
 [5]
 [9]]
```

```
In [9]: c=np.array([[1,2],[3,4]]);print(c)
```

```
[[1 2]
 [3 4]]
```

```
In [10]: print(c.reshape(-1,1))
```

```
[[1]
 [2]
 [3]
 [4]]
```

```
In [11]: print(c.reshape(1,4))
```

```
[[1 2 3 4]]
```