80.03 np.pad

原贴：https://blog.csdn.net/hustqb/article/details/77726660

np.pad（）常用与深度学习中的数据预处理，可以将**numpy**数组按指定的方法填充成指定的形状(**填充方式：在原数组之外，包上其他数据**)。
声明：

np.pad()

对一维数组的填充

```
import numpy as np
arr1D = np.array([1, 1, 2, 2, 3, 4])
'''不同的填充方法'''
print 'constant: ' + str(np.pad(arr1D, (2, 3), 'constant'))
print 'edge: ' + str(np.pad(arr1D, (2, 3), 'edge'))
print 'linear_ramp: ' + str(np.pad(arr1D, (2, 3), 'linear_ramp'))
print 'maximum: ' + str(np.pad(arr1D, (2, 3), 'maximum'))
print 'mean: ' + str(np.pad(arr1D, (2, 3), 'mean'))
print 'median: ' + str(np.pad(arr1D, (2, 3), 'median'))
print 'minimum: ' + str(np.pad(arr1D, (2, 3), 'minimum'))
print 'reflect: ' + str(np.pad(arr1D, (2, 3), 'reflect'))
print 'symmetric: ' + str(np.pad(arr1D, (2, 3), 'symmetric'))
print 'wrap: ' + str(np.pad(arr1D, (2, 3), 'wrap'))
```

```
constant:  [0 0 1 1 2 2 3 4 0 0 0]
edge:  [1 1 1 1 2 2 3 4 4 4 4]
linear_ramp:  [0 0 1 1 2 2 3 4 3 1 0]
maximum:  [4 4 1 1 2 2 3 4 4 4 4]
mean:  [2 2 1 1 2 2 3 4 2 2 2]
median:  [2 2 1 1 2 2 3 4 2 2 2]
minimum:  [1 1 1 1 2 2 3 4 1 1 1]
reflect:  [2 1 1 1 2 2 3 4 3 2 2]
symmetric:  [1 1 1 1 2 2 3 4 4 3 2]
wrap:  [3 4 1 1 2 2 3 4 1 1 2]
```

解释：

第一个参数是待填充数组

第二个参数是填充的形状，（2，3）表示前面两个，后面三个

第三个参数是填充的方法

填充方法：

constant连续一样的值填充，有关于其填充值的参数。
`constant_values=（x, y）`时前面用x填充，后面用y填充。缺参数是为0000。。。

edge用边缘值填充：即数组边上是什么值，就用这个边上的值填充。比如上例，就在前面填两个1，后面填3个4 。

linear_ramp边缘递减的填充方式。比如上例，在前面填0 0 ，后面填3 2 1 0

maximum, **mean**, **median**, **minimum**分别用最大值、均值、中位数和最小值填充

reflect, **symmetric**都是对称填充。前一个是关于边缘对称，后一个是关于边缘外的空气对称ヽ(ゝ ▽ ゝ)ノ

wrap用原数组后面的值填充前面，前面的值填充后面

也可以有其他自定义的填充方法

对多维数组的填充

```
import numpy as np
arr3D = np.array([[[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4]],
                  [[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]],
                  [[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4]]])
                  '''对于多维数组'''
print 'constant: \n' + str(np.pad(arr3D, ((0, 0), (1, 1), (2, 2)), 'constant'))
print 'edge: \n' + str(np.pad(arr3D, ((0, 0), (1, 1), (2, 2)), 'edge'))
print 'linear_ramp: \n' + str(np.pad(arr3D, ((0, 0), (1, 1), (2, 2)), 'linear_ramp'))
print 'maximum: \n' + str(np.pad(arr3D, ((0, 0), (1, 1), (2, 2)), 'maximum'))
print 'mean: \n' + str(np.pad(arr3D, ((0, 0), (1, 1), (2, 2)),
```

```
        'mean'))
print 'median: \n' + str(np.pad(arr3D, ((0, 0), (1, 1), (2, 2))
```

```
constant:
[[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 0 0 0 0 0 0 0 0]]

 [[0 0 0 0 0 0 0 0 0 0]
  [0 0 0 1 2 3 4 5 0 0]
  [0 0 0 1 2 3 4 5 0 0]
  [0 0 0 1 2 3 4 5 0 0]
  [0 0 0 0 0 0 0 0 0 0]]

 [[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 0 0 0 0 0 0 0 0]]]
edge:
[[[1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]]

 [[0 0 0 1 2 3 4 5 5 5]
  [0 0 0 1 2 3 4 5 5 5]
  [0 0 0 1 2 3 4 5 5 5]
  [0 0 0 1 2 3 4 5 5 5]
  [0 0 0 1 2 3 4 5 5 5]]

 [[1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]
  [1 1 1 1 2 2 3 4 4 4]]]
linear_ramp:
[[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 2 0]
  [0 0 1 1 2 2 3 4 2 0]
  [0 0 1 1 2 2 3 4 2 0]
  [0 0 0 0 0 0 0 0 0 0]]

 [[0 0 0 0 0 0 0 0 0 0]
  [0 0 0 1 2 3 4 5 2 0]
  [0 0 0 1 2 3 4 5 2 0]
  [0 0 0 1 2 3 4 5 2 0]
  [0 0 0 0 0 0 0 0 0 0]]

 [[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 2 0]
  [0 0 1 1 2 2 3 4 2 0]
  [0 0 1 1 2 2 3 4 2 0]
  [0 0 0 0 0 0 0 0 0 0]]]
maximum:
[[[4 4 1 1 2 2 3 4 4 4]
  [4 4 1 1 2 2 3 4 4 4]
  [4 4 1 1 2 2 3 4 4 4]
  [4 4 1 1 2 2 3 4 4 4]
  [4 4 1 1 2 2 3 4 4 4]]

 [[5 5 0 1 2 3 4 5 5 5]
  [5 5 0 1 2 3 4 5 5 5]
  [5 5 0 1 2 3 4 5 5 5]
  [5 5 0 1 2 3 4 5 5 5]
  [5 5 0 1 2 3 4 5 5 5]]

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

```
mean:
[[[2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]]

 [[2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]]

 [[2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]]]
minimum:
[[[1 1 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 1 1]]

 [[0 0 0 1 2 3 4 5 0 0]
  [0 0 0 1 2 3 4 5 0 0]
  [0 0 0 1 2 3 4 5 0 0]
  [0 0 0 1 2 3 4 5 0 0]
  [0 0 0 1 2 3 4 5 0 0]]

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

```
median:
[[[2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]]

 [[2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]
  [2 2 0 1 2 3 4 5 2 2]]

 [[2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]
  [2 2 1 1 2 2 3 4 2 2]]]
reflect:
[[[2 1 1 1 2 2 3 4 3 2]
  [2 1 1 1 2 2 3 4 3 2]
  [2 1 1 1 2 2 3 4 3 2]
  [2 1 1 1 2 2 3 4 3 2]
  [2 1 1 1 2 2 3 4 3 2]]

 [[2 1 0 1 2 3 4 5 4 3]
  [2 1 0 1 2 3 4 5 4 3]
  [2 1 0 1 2 3 4 5 4 3]
  [2 1 0 1 2 3 4 5 4 3]
  [2 1 0 1 2 3 4 5 4 3]]

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

```
symmetric:                    wrap:
[[[1 1 1 1 2 2 3 4 4 3]      [[[3 4 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 4 3]        [3 4 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 4 3]        [3 4 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 4 3]        [3 4 1 1 2 2 3 4 1 1]
  [1 1 1 1 2 2 3 4 4 3]]       [3 4 1 1 2 2 3 4 1 1]]

 [[1 0 0 1 2 3 4 5 5 4]       [[4 5 0 1 2 3 4 5 0 1]
  [1 0 0 1 2 3 4 5 5 4]        [4 5 0 1 2 3 4 5 0 1]
  [1 0 0 1 2 3 4 5 5 4]        [4 5 0 1 2 3 4 5 0 1]
  [1 0 0 1 2 3 4 5 5 4]        [4 5 0 1 2 3 4 5 0 1]
  [1 0 0 1 2 3 4 5 5 4]]       [4 5 0 1 2 3 4 5 0 1]]

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

补充：

```
arr3D = np.array([[[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3,
4]],
                  [[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3,
4]],
                  [[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3,
4]]])
print('constant:  \n' + str(np.pad(arr3D, ((1, 1), (1, 1), (2, 2)),
'constant')))
```

生成：在z维上下，加入全0的二维矩阵。

```
constant:
[[[0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]]


[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
```

```
  [0 0 0 0 0 0 0 0 0 0]]


[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 0 0 0 0 0 0 0 0]]


[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 0 0 0 0 0 0 0 0]]


[[0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]]]
```

如果z维改成(0, 0)，补充：

```
arr3D = np.array([[[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3,
4]],
                  [[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3,
4]],
                  [[1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3, 4], [1, 1, 2, 2, 3,
4]]])
print('constant:  \n' + str(np.pad(arr3D, ((0, 0), (1, 1), (2, 2)),
'constant')))
```

生成：

```
constant:

[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 0 0 0 0 0 0 0 0]]


[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
```

```
  [0 0 0 0 0 0 0 0 0 0]]


[[0 0 0 0 0 0 0 0 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 1 1 2 2 3 4 0 0]
  [0 0 0 0 0 0 0 0 0 0]]
```