

5.2 数组变形

Markdown by 董慧杰

1. 直接变形

a) reshape()函数

numpy的Ndarray具有附带的reshape()函数

- 输入参数为要变形的各个维度大小
- -1的作用：在所有维度均确定后，表示剩下的

代码示例：

```
In [1]: import numpy as np      #导入Numpy
x=np.zeros((30,73,144)) #生成一个30X73X144的数组
print(x.shape)          #查看数组结构
```

(30, 73, 144)

```
In [2]: y=x.reshape(-1,144)  #将x前两个维度合并
print(y.shape)
```

(2190, 144)

也可以用 **np.reshape()**函数

- 第1个参数是要变形的数组
- 第2个参数是一个列表或元组，表达要变成的形状

```
In [4]: y=np.reshape(x,(-1,144))
print(y.shape)
```

(2190, 144)

```
In [3]: y = x.reshape(-1,15,73,144)
print(y.shape)
```

(2, 15, 73, 144)

```
In [5]: y=x.reshape(2,15,73,144)
print('y shape:', y.shape)
y2=x.flatten()
print('y2 shape:', y2.shape)
```

y shape: (2, 15, 73, 144)

y2 shape: (315360,)

注意：

reshape () 与flatten () 函数一样，**变形后数组总元素数必须与变形前一致**，当确定了其余维度后，剩余维度可以用-1表示

b) 转置及维度变换

```
In [6]: x=np.zeros((2190,144))
x=x.T      #实现二维数组转置
```

```
print(x.shape)
```

(144, 2190)

```
In [7]: x=np.zeros((17,520,73,144))
y=np.moveaxis(x,0,1) #将数组x的第0维移动到第1维
print(y.shape)
```

(520, 17, 73, 144)

2. 拓展变形

a) np.meshgrid()

```
In [8]: import numpy as np
lon = np.arange(144)
lat = np.arange(73)
x, y = np.meshgrid(lon,lat) # 把lon在0维复制73次形成x; 把lat在1维复制144次形成y
print(x.shape,y.shape)
```

(73, 144) (73, 144)

b) np.tile()

示例1:

```
In [9]: y=np.tile(lon,(73,1)) #将lon一维数组在0维复制73次，形成73*144的数组
print('y shape:',y.shape)
y2=np.tile(lon,(73,2))#将lon一维数组在0维复制73次，一维复制2次，形成73*288的数组
print('y2 shape:',y2.shape)
```

y shape: (73, 144)

y2 shape: (73, 288)

示例2:

```
In [10]: #将lon逐渐拓展维520X17X73X144矩阵
y = np.tile(lon,(73,1))
print('1st:',y.shape)
y = np.tile(y,(17,1,1))
print('2nd:',y.shape)
y = np.tile(y,(520,1,1,1))
print('3rd:',y.shape)
```

1st: (73, 144)

2nd: (17, 73, 144)

3rd: (520, 17, 73, 144)

c) np.expand_dims()

示例1:

```
In [11]: x = np.zeros((520,17,73,144))
y1 = np.expand_dims(x,1) #将数组x在增加1维，增加位置位于第1维
print('y1.shape:',y1.shape)
y2 = np.expand_dims(x,0) #将数组x在增加1维，增加位置位于第0维
print('y2.shape:',y2.shape)
```

y1.shape: (520, 1, 17, 73, 144)

y2.shape: (1, 520, 17, 73, 144)

d) np.squeeze()

```
In [12]: print('before:', y2.shape)
y3 = np.squeeze(y2) # 将数组中长度维1的维度去掉
print('after:', y3.shape)
```

before: (1, 520, 17, 73, 144)

after: (520, 17, 73, 144)

3. 数组拼接

a) np.hstack()

示例1:

```
In [13]: a = np.zeros((3,5))
b = np.zeros((3,1))
c = np.zeros((3,2))
x = np.hstack((a,b,c)) #将数组"列"拼接，"行"维度必须一致
x.shape
```

Out[13]: (3, 8)

示例2:

```
In [14]: a = a.flatten() #将数组转换1维数组
b = b.flatten()
c = c.flatten()
x = np.hstack((a,b,c)) #将a,b,c数组拼接为x
print(a.shape,b.shape,c.shape,x.shape)
```

(15,) (3,) (6,) (24,)

b) np.vstack()

```
In [15]: a = np.zeros((2,3))
b = np.zeros((1,3))
c = np.zeros((5,3))
x = np.vstack((a,b,c)) #将数组"行"拼接，"列"维度必须一致
print(x.shape)
```

(8, 3)

c) np.concatenate()

```
In [16]: a = np.zeros((1,3,5))
b = np.zeros((1,4,5))
c = np.zeros((1,6,5))
x = np.concatenate((a,b,c),axis=1) #将a,b,c合并，指定拼接的维度为1维
x.shape
```

Out[16]: (1, 13, 5)

课堂练习

给定一个一维数组 `A = np.array([1, 2, 3, 4, 5, 6])`，通过以下步骤进行一系列数组变形操作，最终生成一个符合要求的数组：

1. 利用 `np.reshape`：将 `A` 数组变形为一个二维数组，形状为 (2, 3)。
2. 利用 `np.moveaxis`：将该二维数组的维度顺序从 (2, 3) 调整为 (3, 2)。
3. 通过 `np.tile` 扩展该数组，使其沿第一个轴重复 3 次，得到一个形状为 (9, 2) 的数组。
4. 利用 `np.expand_dims`：使用 `np.expand_dims` 增加一个维度，将形状从 (9, 2) 改为 (1, 9, 2)。
5. 使用 `np.squeeze` 去掉所有单维度，将其变回 (9, 2)。
6. 使用 `np.hstack` 将该数组与自己拼接，得到一个形状为 (9, 4) 的数组。
7. 使用 `np.vstack` 将拼接后的数组垂直堆叠两次，得到一个形状为 (18, 4) 的最终数组。

► 参考答案

注意：在Python 编程中，能找到函数完成的任务一般先用函数；能用数组运算进行的计算不使用循环进行计算。因为Python 的函数和数组运算是基于标准的C语言编写的，相较于Python 本身，运行速度更快。