链滴

Python 数据科学 (1)——NumPy(4.Arrays 数据结构化)

作者：hsxian

原文链接：https://ld246.com/article/1556009607185

来源网站：链滴

在 numpy 中，提供了基本的数据类型，如int64、str。显然，基本数据类型是不能解决复杂的业务题的，大多数情况下，我们使用的都是自定义的结构化数据。例如，描述一个人，用str存储其名字，int64存储其地址等等。如果是一个公司，有很多员工，每一个属性将有大量的值。我们可以把这些值立地存储在不同的 Arrays 中，用 numpy 仍然可以进行计算，但却把相关性给丢弃了，这显然得不失。我们需要呼唤更高级的数据结构。

类似于定义np.zeros(4, dtype=int)基本数据类型，我们可以在矩阵中定义复合类型：

```
In [3]: data = np.zeros(4, dtype={'names':('name', 'age', 'weight'),
   ...:                           'formats':('U10', 'i4', 'f8')})
In [4]: data
Out[4]:
array([('', 0,  0.), ('', 0,  0.), ('', 0,  0.), ('', 0,  0.)],
      dtype=[('name', '<U10'), ('age', '<i4'), ('weight', '<f8')])
```

紧接着，把数据导入到定义好的结构中

```
In [5]: name = ['Alice', 'Bob', 'Cathy', 'Doug']
   ...: age = [25, 45, 37, 19]
   ...: weight = [55.0, 85.5, 68.0, 61.5]

In [6]: data['name'] = name
   ...: data['age'] = age
   ...: data['weight'] = weight
```

作为结果，我们可以观察到 data 中存储了结构化的数据

```
In [7]: data
Out[7]:
array([('Alice', 25,  55. ), ('Bob', 45,  85.5), ('Cathy', 37,  68. ),
       ('Doug', 19,  61.5)],
      dtype=[('name', '<U10'), ('age', '<i4'), ('weight', '<f8')])
```

这方便了我们对数据的操作

```
In [8]: data['name']
Out[8]:
array(['Alice', 'Bob', 'Cathy', 'Doug'],
      dtype='<U10')

In [9]: data[0]
Out[9]: ('Alice', 25,  55.)

In [10]: data[-1]['name']
Out[10]: 'Doug'
```

创建结构化矩阵

使用字典

```
In [11]: np.dtype({'names':('name', 'age', 'weight'),
    ...:           'formats':('U10', 'i4', 'f8')})
Out[11]: dtype([('name', '<U10'), ('age', '<i4'), ('weight', '<f8')])
```

或者，为了清晰，使用 numpy 的指定类型

```
np.dtype({'names':('name', 'age', 'weight'),
          'formats':((np.str_, 10), int, np.float32)})
```

使用元组

```
In [12]: np.dtype([('name', 'S10'), ('age', 'i4'), ('weight', 'f8')])
Out[12]: dtype([('name', 'S10'), ('age', '<i4'), ('weight', '<f8')])
```

如过名称不重要的话，直接使用类型定义

```
In [13]: np.dtype('S10,i4,f8')
Out[13]: dtype([('f0', 'S10'), ('f1', '<i4'), ('f2', '<f8')])
```

以下是类型简写表：

字符	描述	例
'b'	字节	np.dtype('b')
'i'	有符号整数	np.dtype('i4') == np.int32
'u'	无符号整数	np.dtype('u1') = np.uint8
'f'	浮点	np.dtype('f8') == np.int4
'c'	复数	np.dtype('c16') == np.omplex128
'S'，'a'	字符串	np.dtype('S5')

'U' == np.str_	Unicode 字符串	np.dtype('U')
'V') == np.void	原始数据（无效）	np.dtype('

更高级的复合类型

使用嵌套，可以创建出更高级高级的数据类型：

```
In [14]: tp = np.dtype([('id', 'i8'), ('mat', 'f8', (3, 3))])
In [15]: np.zeros(1, dtype=tp)
Out[15]:
array([(0, [[ 0.,  0.,  0.], [ 0.,  0.,  0.], [ 0.,  0.,  0.]])],
      dtype=[('id', '<i8'), ('mat', '<f8', (3, 3))])
```

记录矩阵

回顾之前的数据结构物data，如果要获取所有的age，我们只能

```
In [16]: data['age']
Out[16]: array([25, 45, 37, 19], dtype=int32)
```

此外，我们可以使用类recarray来创建视图，这样就可以像访问属性一样访问字典数据：

```
In [17]: data_rec = data.view(np.recarray)
    ...: data_rec.age
Out[17]: array([25, 45, 37, 19], dtype=int32)
```

此外，我们还将获得额外的效率提升：

```
In [18]: %timeit data['age']
    ...: %timeit data_rec['age']
    ...: %timeit data_rec.age
138 ns ± 1.25 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops each)
2.92 µs ± 35.7 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
3.72 µs ± 18 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

迎接 pandas

对于处理结构化数据，pandas包是一个更好的选择。