How to count the occurrence of certain item in an ndarray in Python?

256

60

In Python, I have an ndarray `y` that is printed as `array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])`

I'm trying to count how many `0` s and how many `1` s are there in this array.

But when I type `y.count(0)` or `y.count(1)` , it says

> `numpy.ndarray` object has no attribute `count`

What should I do?

python numpy multidimensional-array count

edited Jun 21 '18 at 9:16

Ivan
5,507 3 15 45

asked Feb 22 '15 at 22:05

mflowww
1,520 3 11 16

7 Can't you use sum and length function, since you only have aces and zeros? – nikaltipar Feb 22 '15 at 22:07

In this case, it is also possible to simply use `numpy.count_nonzero` . – Mong H. Ng Mar 31 at 17:50

24 Answers

423

```
>>> a = numpy.array([0, 3, 0, 1, 0, 1, 2, 1, 0, 0, 0, 0, 1, 3, 4])
>>> unique, counts = numpy.unique(a, return_counts=True)
>>> dict(zip(unique, counts))
{0: 7, 1: 4, 2: 1, 3: 2, 4: 1}
```

Non-numpy way:

Use `collections.Counter` ;

```
>> import collections, numpy

>>> a = numpy.array([0, 3, 0, 1, 0, 1, 2, 1, 0, 0, 0, 0, 1, 3, 4])
>>> collections.Counter(a)
Counter({0: 7, 1: 4, 3: 2, 2: 1, 4: 1})
```

edited Jan 24 '18 at 9:31

answered Feb 22 '15 at 22:10

ozgur
29.7k 14 56 85

2 That would be ``` unique, counts = numpy.unique(a, return_counts=True) dict(zip(unique, counts)) ``` – shredding Mar 16 '16 at 13:14

13 If you want the dictionary, `dict(zip(*numpy.unique(a, return_counts=True)))` – Seppo Enarvi Apr

28 '16 at 13:19

2 What if I want to access the number of occurences of each unique elements of the array without assigning to the variable - counts. Any hints on that ? – sajis997 Dec 24 '16 at 23:08

I have the same goal as @sajis997. I want to use 'count' as an aggregating function in a groupby – p_sutherland Mar 15 '18 at 16:34

@sajis997 if you do a groupby on the desired level of aggregation and use np.count_nonzero as the aggregate function it will return the number of occurrences of a each unique value – p_sutherland Mar 15 '18 at 16:52

183

What about using `numpy.count_nonzero` , something like

```
>>> import numpy as np
>>> y = np.array([1, 2, 2, 2, 2, 0, 2, 3, 3, 3, 0, 0, 2, 2, 0])

>>> np.count_nonzero(y == 1)
1
>>> np.count_nonzero(y == 2)
7
>>> np.count_nonzero(y == 3)
3
```

edited Oct 12 '17 at 14:33

answered Feb 22 '16 at 9:14

Aziz Alto

9,114 3 47 46

12 This answer seems better than the one with the most upvotes. – Alex Dec 31 '17 at 17:16

1 I don't think this would work for `numpy.ndarray` as OP originally asked. – LYu Jul 28 '18 at 20:55

3 @LYu - the y is an np.ndarray in this answer. Also - most if not all np.something functions work on ndarrays without problem. – mmagnuski Jul 29 '18 at 19:34

102

Personally, I'd go for: `(y == 0).sum()` and `(y == 1).sum()`

E.g.

```
import numpy as np
y = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
num_zeros = (y == 0).sum()
num_ones = (y == 1).sum()
```

answered May 5 '16 at 20:51

Gus Hecht

1,021 1 5 2

1 It's definitely the easiest to read. The question is which is fastest, and most space efficient – frank May 30 '18 at 19:02

Mightbe less space efficient than numpy.count_nonzero(y==0), since it evaluates the vector (y==0) –

Sridhar Thiagarajan Oct 27 '18 at 22:21

30

For your case you could also look into numpy.bincount

```
In [56]: a = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])

In [57]: np.bincount(a)
Out[57]: array([8, 4])  #count of zeros is at index 0 : 8
                        #count of ones is at index 1 : 4
```

answered Feb 22 '15 at 23:45

Akavall
43.6k 33 148 187

This code may be one of the fastest solutions for larger arrays I experimented. Getting the result as a list is a bonus, too. Thanx! – Youngsup Kim Oct 24 '18 at 22:56

1 Not that `numpy.bincount` works for integers only. – Skippy le Grand Gourou Feb 24 at 10:39

17

Convert your array `y` to list `l` and then do `l.count(1)` and `l.count(0)`

```
>>> y = numpy.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
>>> l = list(y)
>>> l.count(1)
4
>>> l.count(0)
8
```

edited Feb 22 '15 at 22:19

answered Feb 22 '15 at 22:12

Milind Dumbare
2,078 2 13 31

15

```
y = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
```

If you know that they are just `0` and `1` :

```
np.sum(y)
```

gives you the number of ones. `np.sum(1-y)` gives the zeroes.

For slight generality, if you want to count `0` and not zero (but possibly 2 or 3):

```
np.count_nonzero(y)
```

gives the number of nonzero.

But if you need something more complicated, I don't think numpy will provide a nice `count` option. In that case, go to collections:

```
import collections
collections.Counter(y)
> Counter({0: 8, 1: 4})
```

This behaves like a dict

```
collections.Counter(y)[0]
> 8
```

edited Feb 22 '15 at 22:15

answered Feb 22 '15 at 22:10

Joel
11.4k 28 59

13

If you know exactly which number you're looking for, you can use the following;

```
lst = np.array([1,1,2,3,3,6,6,6,3,2,1])
(lst == 2).sum()
```

returns how many times 2 is occurred in your array.

answered Nov 22 '17 at 10:31

CanCeylan
1,400 7 28 46

6

```
y.tolist().count(val)
```

with val 0 or 1

Since a python list has a native function `count` , converting to list before using that function is a simple solution.

answered May 19 '16 at 19:14

michael
147 1 8

6

Honestly I find it easiest to convert to a pandas Series or DataFrame:

```
import pandas as pd
import numpy as np

df = pd.DataFrame({'data':np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])})
print df['data'].value_counts()
```

Or this nice one-liner suggested by Robert Muil:

```
pd.Series([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1]).value_counts()
```

edited Sep 11 '17 at 17:41

answered Oct 28 '16 at 17:58

wordsforthewise
3,888 3 31 53

4 Just a note: don't need the DataFrame or numpy, can go directly from a list to a Series: `pd.Series([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1]).value_counts()` – Robert Muil Feb 2 '17 at 20:14

Awesome, that's a nice one-liner. Big up – wordsforthewise Sep 11 '17 at 17:43

6

What about `len(y[y==0])` and `len(y[y==1])` ?

answered Mar 11 '16 at 18:29

Anas
637 11 20

5

Yet another simple solution might be to use **numpy.count_nonzero()**:

```
import numpy as np
y = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
y_nonzero_num = np.count_nonzero(y==1)
y_zero_num = np.count_nonzero(y==0)
y_nonzero_num
4
y_zero_num
8
```

Don't let the name mislead you, if you use it with the boolean just like in the example, it will do the trick.

answered Oct 4 '16 at 9:30

NaZo
51 1 1

5

To count the number of occurrences, you can use `np.unique(array, return_counts=True)` :

```
In [75]: boo = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])

# use bool value `True` or equivalently `1`
In [77]: uniq, cnts = np.unique(boo, return_counts=1)
In [81]: uniq
Out[81]: array([0, 1])   #unique elements in input array are: 0, 1

In [82]: cnts
Out[82]: array([8, 4])   # 0 occurs 8 times, 1 occurs 4 times
```

edited Oct 4 '18 at 23:37

answered Dec 23 '16 at 19:57

kmario23

21.7k 5 72 83

4

No one suggested to use `numpy.bincount(input, minlength)` with `minlength = np.size(input)`, but it seems to be a good solution, and definitely the *fastest*:

```
In [1]: choices = np.random.randint(0, 100, 10000)

In [2]: %timeit [ np.sum(choices == k) for k in range(min(choices),
max(choices)+1) ]
100 loops, best of 3: 2.67 ms per loop

In [3]: %timeit np.unique(choices, return_counts=True)
1000 loops, best of 3: 388 µs per loop

In [4]: %timeit np.bincount(choices, minlength=np.size(choices))
100000 loops, best of 3: 16.3 µs per loop
```

That's a crazy speedup between `numpy.unique(x, return_counts=True)` and `numpy.bincount(x, minlength=np.max(x))` !

edited Oct 24 '18 at 8:19

answered Mar 17 '17 at 16:19

Næreen

448 6 14

hows it compare to histogram? – john ktejik Oct 22 '18 at 3:02

@johnktejik `np.histogram` does not compute the same thing. No point comparing the three approaches I propose with the `histogram` function, sorry. – Næreen Oct 24 '18 at 8:20

1 @Næreen `bincount` only works for integers though, so it works for the OP's problem, but maybe not for the generic problem described in the title. Also have you tried using `bincount` with arrays with very big ints? – Imperishable Night Oct 27 '18 at 13:19

@ImperishableNight no I haven't tried with large ints, but anyone is welcome to do so and post their own benchmark :-) – Næreen Oct 30 '18 at 17:47

4

I'd use np.where:

```
how_many_0 = len(np.where(a==0.)[0])
how_many_1 = len(np.where(a==1.)[0])
```

answered Oct 19 '15 at 14:15

MaxG

78 7

You can use dictionary comprehension to create a neat one-liner. More about dictionary

2

comprehension [can be found here]

```
>>>counts = {int(value): list(y).count(value) for value in set(y)}
>>>print(counts)
{0: 8, 1: 4}
```

This will create a dictionary with the values in your ndarray as keys, and the counts of the values as the values for the keys respectively.

This will work whenever you want to count occurences of a value in arrays of this format.

edited Dec 3 '18 at 16:18

answered Dec 3 '18 at 16:12

CB Madsen

61 3

1

A general and simple answer would be:

```
numpy.sum(MyArray==x)   # sum of a binary list of the occurence of x (=0 or 1)
in MyArray
```

which would result into this full code as exemple

```
import numpy
MyArray=numpy.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])  # array we want to
search in
x=0   # the value I want to count (can be iterator, in a list, etc.)
numpy.sum(MyArray==0)   # sum of a binary list of the occurence of x in MyArray
```

Now if MyArray is in **multiple dimensions** and you want to count the occurence of a distribution of values in line (= pattern hereafter)

```
MyArray=numpy.array([[6, 1],[4, 5],[0, 7],[5, 1],[2, 5],[1, 2],[3, 2],[0, 2],[2,
5],[5, 1],[3, 0]])
x=numpy.array([5,1])   # the value I want to count (can be iterator, in a list,
etc.)
temp = numpy.ascontiguousarray(MyArray).view(numpy.dtype((numpy.void,
MyArray.dtype.itemsize * MyArray.shape[1])))  # convert the 2d-array into an
array of analyzable patterns
xt=numpy.ascontiguousarray(x).view(numpy.dtype((numpy.void, x.dtype.itemsize *
x.shape[0])))  # convert what you search into one analyzable pattern
numpy.sum(temp==xt)  # count of the searched pattern in the list of patterns
```

edited Nov 18 '16 at 16:19

answered Nov 18 '16 at 16:04

sol

537 2 7 25

1

Since your ndarray contains only 0 and 1, you can use sum() to get the occurrence of 1s and len()-sum() to get the occurrence of 0s.

```
num_of_ones = sum(array)
num_of_zeros = len(array)-sum(array)
```

answered Jan 12 '17 at 17:40

Sabeer Ebrahim
55 1 1 7

1

This can be done easily in the following method

```
y = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
y.tolist().count(1)
```

edited Oct 22 '16 at 0:25

Eli Sadoff
4,890 6 23 44

answered Oct 21 '16 at 21:37

user7055304
11 2

0

For generic entries:

```
x = np.array([11, 2, 3, 5, 3, 2, 16, 10, 10, 3, 11, 4, 5, 16, 3, 11, 4])
n = {i:len([j for j in np.where(x==i)[0]]) for i in set(x)}
ix = {i:[j for j in np.where(x==i)[0]] for i in set(x)}
```

Will output a count:

```
{2: 2, 3: 4, 4: 2, 5: 2, 10: 2, 11: 3, 16: 2}
```

And indices:

```
{2: [1, 5],
3: [2, 4, 9, 14],
4: [11, 16],
5: [3, 12],
10: [7, 8],
11: [0, 10, 15],
16: [6, 13]}
```

edited Nov 6 '18 at 11:21

answered Nov 6 '18 at 11:12

deckard
335 3 7

0

It involves one more step, but a more flexible solution which would also work for 2d arrays and more complicated filters is to create a boolean mask and then use .sum() on the mask.

```
>>>>y = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
>>>>mask = y == 0
```

```
>>>>mask.sum()
8
```

answered Dec 24 '15 at 22:35

Thomas
198 3 10

---

take advantage of the methods offered by a Series:

0

```
>>> import pandas as pd
>>> y = np.array([0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
>>> pd.Series(y).value_counts()
0    8
1    4
dtype: int64
```

answered Jun 13 at 10:33

Sébastien Wieckowski
158 1 8

You have a special array with only 1 and 0 here. So a trick is to use

0

```
np.mean(x)
```

which gives you the percentage of 1s in your array. Alternatively, use

```
np.sum(x)
np.sum(1-x)
```

will give you the absolute number of 1 and 0 in your array.

answered May 6 at 16:28

CathyQian
185 13

If you don't want to use numpy or a collections module you can use a dictionary:

0

```
d = dict()
a = [0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1]
for item in a:
    try:
        d[item]+=1
    except KeyError:
        d[item]=1
```

result:

```
>>>d
{0: 8, 1: 4}
```

Of course you can also use an if/else statement. I think the Counter function does almost the same thing but this is more transparant.

answered Jul 8 '16 at 14:41

JLT
440 4 13

-1

Numpy has a module for this. Just a small hack. Put your input array as bins.

```
numpy.histogram(y, bins=y)
```

The output are 2 arrays. One with the values itself, other with the corresponding frequencies.

answered Apr 26 '17 at 10:37

Ishan Tomar
591 1 7 16

isn't 'bins' supposed to be a number? – john ktejik Oct 22 '18 at 3:07

Yes @johnktejik you're right. This answer does *not* work. – Næreen Oct 24 '18 at 8:21