Numpy:zero mean data and standardization

Asked 2 years, 7 months ago   Active 2 months ago   Viewed 19k times

▲

11

▼

★

1

↺

I saw in tutorial (there were no further explanation) that we can process data to zero mean with `x -= np.mean(x, axis=0)` and normalize data with `x /= np.std(x, axis=0)`. Can anyone elaborate on these two pieces on code, only thing I got from documentations is that `np.mean` calculates arithmetic mean calculates mean along specific axis and `np.std` does so for standard deviation.

python   numpy   image-preprocessing

edited Jun 21 '18 at 6:19   asked Aug 23 '17 at 8:16

Jonas Adler   econ
7,384 2 25 61   **295** 3 4 14

---

3   arithmetic operations are vectorized in `numpy` – juanpa.arrivillaga Aug 23 '17 at 8:19

---

4 Answers

Active | Oldest | Votes

▲

14

▼

✔

↺

This is also called `zscore`.

SciPy has a utility for it:

```
>>> from scipy import stats
>>> stats.zscore([ 0.7972,  0.0767,  0.4383,  0.7866,  0.8091,
...                0.1954,  0.6307,  0.6599,  0.1065,  0.0508])
array([ 1.1273, -1.247 , -0.0552,  1.0923,  1.1664, -0.8559,  0.5786,
        0.6748, -1.1488, -1.3324])
```

edited Jan 15 at 18:51   answered Aug 23 '17 at 8:48

grisaitis   Jonas Adler
2,180 1 19 25   **7,384** 2 25 61

7

```python
import numpy as np

# create x
x = np.asarray([1,2,3,4], dtype=np.float64)

np.mean(x) # calculates the mean of the array x
x-np.mean(x) # this is euivalent to subtracting the mean of x from each value in x
x-=np.mean(x) # the -= means can be read as x = x- np.mean(x)

np.std(x) # this calcualtes the standard deviation of the array
x/=np.std(x) # the /= means can be read as x = x/np.std(x)
```

edited Sep 11 '18 at 23:14

ritiek
1,214   2   11   20

answered Aug 23 '17 at 8:24

Clock Slave
4,771   7   40   78

3

From the given syntax you have I conclude, that your array is multidimensional. Hence I will first discuss the case where your x is just a linear array:

`np.mean(x)` will compute the mean, by broadcasting `x-np.mean(x)` the mean of `x` will be subtracted form all the entries. `x -=np.mean(x,axis = 0)` is equivalent to `x = x-np.mean(x,axis = 0)`. Similar for x/np.std(x)`.

In the case of multidimensional arrays the same thing happens, but instead of computing the mean over the entire array, you just compute the mean over the first "axis". Axis is the `numpy` word for dimension. So if your `x` is two dimensional, then `np.mean(x,axis =0) = [np.mean(x[:,0]), np.mean(x[:,1])...]` . Broadcasting again will ensure, that this is done to all elements.

Note, that this only works with the first dimension, otherwise the shapes will not match for broadcasting. If you want to normalize wrt another axis you need to do something like:

```python
x -= np.expand_dims(np.mean(x,axis = n),n)
```

answered Aug 23 '17 at 8:29

Jürg Merlin Spaak
1,459   8   27

2

Key here are the assignment operators. They actually performs some operations on the original variable. a += c is actually equal to a=a+c.

So indeed a (in your case x) has to be defined beforehand.

Each method takes an array/iterable (x) as input and outputs a value (or array if a multidimensional array was input), which is thus applied in your assignment operations. The axis parameter means that you apply the mean or std operation over the rows. Hence, you take values for each row in a given column and perform the mean or std. Axis=1 would take values of each column for a given row.

data around this zero, and now it should roughly be in a [-1, +1] interval around 0.

So now, each of your column values is centered around zero and standardized.

There are other scaling techniques, such as removing the minimal or maximal value and dividing by the range of values.

answered Aug 23 '17 at 8:28

Ando Jurai

758   1   8   19