numpy python: vectorize distance function to calculate pairwise distance of 2 matrix with a dimension of (m, 3)

Asked 2 years, 2 months ago Modified 1 year, 1 month ago Viewed 860 times

I have two numpy arrays A and B. Shape of A is (m,3) and shape of B is (n, 3).

Those matrix look like this:

```
A
#output
array([[  9.227,  -4.698, -95.607],
   [ 10.294,  -4.659, -94.606],
   [ 11.184,  -5.906, -94.675],
   ...,
   [ 19.538, -91.572, -45.361],
   [ 20.001, -92.655, -45.009],
   [ 19.271, -92.726, -45.79 ]])
```

So it contains for each row the coordinates x,y,z of a 3D point. B follows the same format.

I have this function (np is numpy):

```
def compute_dist(point1, point2):
    squared = (point1-point2)**2
    return (np.sqrt(np.sum(squares)))
```

I want to compute a pairwise distance between A and B by using a vectorized function.

I try this:

```
v = np.vectorize(compute_dist)
v(A, B)
#output
matrix([[37.442, 42.693, 72.705],
   [37.442, 42.693, 72.705],
   [37.442, 42.693, 72.705],
   ...,
   [37.442, 42.693, 72.705],
   [37.442, 42.693, 72.705],
   [37.442, 42.693, 72.705]])
```

I don't understand how to use vectorize even if I read the doc. How can I compute a matrix which contains pairwise distance between A and B? I know there is **scipy.distance.cdist** but I want to do it myself with **np.vectorize**.

I don't care about the format of the output (list, array, matrix ...). At the end I just want to find the minimal distance.

python python-3.x numpy distance

Share Edit Follow Flag

edited Feb 3, 2020 at 13:15

Nicolas Gervais

asked Feb 3, 2020 at 13:13

Adam Bellaïche

Vectorize doesn't do what you think. What output are you expecting? An MxN matrix? – Mad Physicist Feb 3, 2020 at 13:53

@MadPhysicist I don't expect any particular output. The usual form is a matrix of course. But in my code I try to get a list of all distances. – Adam Bellaïche Feb 3, 2020 at 14:04

I don't necessarily mean the numbers in the output. Just the expected shape and conceptual content. – Mad Physicist Feb 3, 2020 at 14:12

I hoped to get a list of length = m*n – Adam Bellaïche Feb 3, 2020 at 14:17

If by list you mean array (they're different in python), then you should upvote and accept the posted answer. It's pretty much the canonical way to do it, and a good introduction to broadcasting. – Mad Physicist Feb 3, 2020 at 14:28

1 Answer

Sorted by:

You can use `np.newaxis` to *expand* the dimensions of your two arrays `A` and `B` to enable broadcasting and then do your calculations.

Pairwise distance means every point in `A (m, 3)` should be compared to every point in `B (n, 3)`. This results in a `(m, n)` matrix of distances. With numpy one can use broadcasting to achieve the wanted result. By using `A=A[:, np.newaxis, :]` and `B=B[np.newaxis, :, :]` the resulting shapes are `A (m, 1, 3)` and `B(1, n, 3)` respectivley. If you then perform a calculation like `C = A-B` numpy automatically broadcasts. This means you get a copy of all m rows of A for all n columns of B and a copy of all n columns of B for all m rows of A.

```
  A (m, 1, 3)
- B (1, n, 3)
--------------
= C (m, n, 3)
```

To get the distance matrix you can then use `numpy.linalg.norm()`:

```
import numpy as np
m = 10
n = 12
A = np.random.random((m, 3))
B = np.random.random((n, 3))

# Add newaxis on second axis of A and on first axis on B
# shape: (m, n, 3) = (m, 1, 3) - (1, n, 3)
C = A[:, np.newaxis, :] - B[np.newaxis, :, :]

C = np.linalg.norm(C, axis=-1)
# shape: (m, n)
```

Share Edit Follow Flag

edited Mar 1, 2021 at 10:49

answered Feb 3, 2020 at 13:30

scleronomic

4,065 1 11 40

I don't understand how it can help me. I miss something I think. Can you explain your mine?
– Adam Bellaïche Feb 3, 2020 at 13:36

It is not a solution using `np.vectorize`, but rather `broadcasting`. A look in the documentation might help you. – scleronomic Feb 3, 2020 at 13:53

1 @Adam. This is about as spelled out as it can reasonably get. You need to sit down and walk yourself through the steps line by line. Play with each operation until you understand it. It's a painful process in the beginning, but absolutely necessary if you want to learn. – Mad Physicist Feb 3, 2020 at 13:55

1 Finally you want an `(n, m)` matrix where the `element (i, j)` stands for the scalar distance between point i of A and point j of B. As intermediate result you have an array `(m, n, 3)` where an `element (i, j, :)` stands for the distance `vector` between point i of A and point j of B. – scleronomic Feb 3, 2020 at 14:31

1 @AdamBellaïche The last line in my answer `np.linalg.norm(C, axis=-1)` does exactly that. If you want to compute it yourself its fine, but instead of using `apply_over_axis` you can just use `np.sum((A-B)**2, axis=2)`. (Remember python uses `zero based indexing`) – scleronomic Feb 3, 2020 at 15:07