calculating distance matrix of a given row-wise vector matrix

Suppose I have an matrix nxm accommodating row vectors. I want to have an distance matrix nxn that presents the distance of each vector to each other. How can I do it in Python as I am using Numpy. I know Scipy does it but I want to dirst my hands. I already write a cosine similarity function `cos_dist(a,b)` where a and b two different vectors. Now I need a caller function that is doing it for each couple of items efficiently. How would I do it?

matrix numpy distance

asked May 9 '13 at 20:13
erogol
3,861 15 61 116

If scipy does it, why not look at the source code of the relevant function? It might only depend on numpy already. – fgb May 9 '13 at 20:26

is not installed and I am not SUDO – erogol May 9 '13 at 20:37

Do you know the name of the relevant function? You could just google for the source code. – fgb May 9 '13 at 20:47

not exactly ... – erogol May 9 '13 at 20:57

You might want to look into python `virtualenv` , which allows you to install python and its dependencies anywhere, without the need to have `sudo` privileges. – fgb May 9 '13 at 21:35

2 Answers

The following code shows two option to do what you are after. One looping over the array twice and using a Python function to calculate the cos_dist. The second uses a vectorized approach and broadcasting to get the same result x1000 faster.

```
from __future__ import division
import numpy as np

def cos_dist(a, b):
    mod_a = np.sqrt(a.dot(a))
    mod_b = np.sqrt(b.dot(b))
    return a.dot(b) / mod_a / mod_b

a = np.random.rand(100, 4)

# Slow option
def slow_dist(a):
    items = a.shape[0]
    out_slow = np.ones((items,items))
    for j in xrange(items):
        for k in xrange(j+1, items):
            out_slow[j, k] = cos_dist(a[j], a[k])
            out_slow[k, j] = out_slow[j, k]
    return out_slow

# Faster option
from numpy.core.umath_tests import inner1d
def fast_dist(a):
    mod_a = np.sqrt(inner1d(a ,a))
    norm_a = a / mod_a[:, None]
    out_fast = inner1d(norm_a[:, None, :],
                       norm_a[None, :, :])
    return out_fast
```

And here are the timings:

```
In [2]: %timeit slow_dist(a)
10 loops, best of 3: 67.6 ms per loop

In [3]: %timeit fast_dist(a)
10000 loops, best of 3: 60.5 us per loop

In [4]: np.allclose(slow_dist(a), fast_dist(a))
Out[4]: True
```

edited Jun 15 '14 at 3:34
Amyunimus
708 2 12 36

answered May 10 '13 at 16:00
Jaime
44k 6 65 118

Why don't you check on scipy's `spatial.distance.pdist()` , which computes pairwise distances between observations in n-dimensional space and has a vast number of distance functions to choose from?

Since you don't have scipy installed and want to code this using numpy, I suggest you study its source code, which is linked at the top-left of its documentation page.

answered May 9 '13 at 21:09
fgb
1,379 10 21

thanks for the effort but not enough since there are lots of wrappers – erogol May 9 '13 at 21:21

np. I was hoping that it would at least get you started in the right direction. – fgb May 9 '13 at 21:26