Subject: Nearest Neighbor ... again!
Posted by Fabzi on Thu, 08 Jul 2010 13:11:37 GMT
View Forum Message <> Reply to Message

Hi everybody,

You probably have been asked many times, but once again this
apparently simple problem is driving me crazy. The problem is famous:

I have a 2D grid defined by x1 (dim 2 array, for example lons) and y1
(dim2 array, for example lats). And I want to fit it to a second grid
x2, y2. More precisely, I want to know the indexes in GRID1 that are
the closest to each of my points in GRID2. The output of my function
has then the same dimension as GRID2.

After using a long time the (very) inefficient "for loop":

```
-----
n2 = N_ELEMENTS(x2)
for i = 0l, n2 - 1 do begin
        quad = (x2[i] - x1)^2 + (y2[i]- y1)^2
        minquad = min(quad, p)
        if N_ELEMENTS(p) gt 1 then p = p[0] ; it happens.....
        out[i] = p
 endfor
---
```

I finaly found the best solution:

```
---
  n1 = n_elements(ilon)
  triangulate, x1, y1, c ; Compute Delaunay triangulation
  out = GRIDDATA(x1,y1, LINDGEN(n1), XOUT=x2, YOUT=y2, /NEAREST_N,
TRIANGLES =c)
---
```

Which is very fast.

However, I cannot find a quick solution to get the FOUR nearest points
in my GRID1.

The stupid solution is (sorry for the very very ugly code):

```
---
      for i = 0l, n2 - 1 do begin
        quad = (x2[i] - x1)^2 + (y2[i]- y1)^2
        for j=0, 3 do begin
          minquad = min(quad, p)
          if N_ELEMENTS(p) gt 1 then p = p[0] ; it happens.....
          out[j,i] = p
          quad[p] = max(quad) * 2. ;dummy large distance
        endfor
      endfor
---
```

But I just cannot find a cleverer solution with triangulation...
Someone clever than me to help ?

Thanks a lot!

Fabz

Subject: Re: Nearest Neighbor ... again!
Posted by Wout De Nolf on Mon, 12 Jul 2010 08:56:23 GMT

View Forum Message <> Reply to Message

On Mon, 12 Jul 2010 01:14:09 -0700 (PDT), Fabzi
<fabien.maussion@gmail.com> wrote:

>> These two methods give a different result. Try running the code below.
>> The red and green lines connect the nearest neighbours from method 1
>> and method 2. You should see only red lines, but you see some green
>> lines too...
>
> Yes, the first method is actually the fastest (especially when working
> on
> huge datasets).

Well, fast or not, the results are different so what's going on? Maybe
someone can shed some light on this, it would be interresting to know.

Subject: Re: Nearest Neighbor ... again!
Posted by Fabzi on Tue, 13 Jul 2010 09:31:36 GMT

View Forum Message <> Reply to Message

You're right, I didn't notice!

The GRIDDATA procedure definitively produce doubtful nearest neighbors
for a few points...

On Jul 12, 10:56 am, Wox <s...@nomail.com> wrote:
> On Mon, 12 Jul 2010 01:14:09 -0700 (PDT), Fabzi
>

> <fabien.mauss...@gmail.com> wrote:
>>> These two methods give a different result. Try running the code below.
>>> The red and green lines connect the nearest neighbours from method 1
>>> and method 2. You should see only red lines, but you see some green
>>> lines too...
>
>> Yes, the first method is actually the fastest (especially when working
>> on
>> huge datasets).
>
> Well, fast or not, the results are different so what's going on? Maybe
> someone can shed some light on this, it would be interresting to know.