
Subject: Re: counting points falling inside a Voronoi cell
Posted by [penteado](#) on Sun, 03 May 2015 20:49:34 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello,

I expect the most efficient solution is dependent on some parameters of your problem, particularly the number of points in data1 and data2, how much overlap there is in the area covered by a polygon surrounding each of them, and how homogeneous/clustered the points are.

Going for a third alternative - not the double loop or the masking method:

Since your polygons are Voronoi cells, each point in data2 will be inside either 0 or 1 polygon in data1. Therefore, I see the point in doing a while loop, to stop searching for a polygon once one is found. But calling a function to test a single data point/polygon pair is so inherently inefficient, that I suspect this approach would be better:

1) Loop over data1's polygons, since there are much fewer polygons than points in data2. For each polygon, call a function that accepts an array of points to test, so that there is only one function call per polygon, testing for all of data2's points. Such as IDLanROI::ContainsPoints (www.exelisvis.com/docs/idlanroi__containspoints.html).

2) If data2 has many points, and the polygons in data1 cover a large fraction of the area covered by a polygon encircling data2's points, this can be significantly sped up if, after searching for data2's points on each data1 polygon, the matches are removed from data2, since we know a point in data2 cannot be in more than one Voronoi cell. Since data2 will be decreasing in size, the loop iterations will speedup as the code moves through the polygons.

It would be something like

```
data2_work=data2
matches=long64arr(N1)
foreach roi,rois,iroi do begin
  print,'processing polygon ',iroi
  match=roi.containspoints(data2_work)
  w=where(match eq 1,count,complement=wc,ncomplement=nc)
  matches[iroi]=count
  if (nc gt 0) then data2_work=data2_work[*,wc] else break
endforeach
```

Where N1 is the number of polygons in data1, and I am guessing that data2 is a 2xN2 array, with x,y for each of the N2 points. And rois is an array of N1 IDLanROIs, created from the Voronoi cells you have found.

If data2 still has too many points, so that the first loop iterations are taking too long (or, worse, memory fills up), this can be easily parallelized by splitting data2 into smaller chunks. The combined result is just a sum over the matches array produced on testing each chunk of data2.

The code above is only looking for points falling inside each Voronoi polygon. You must also consider that a point can be on the edge or on a vertex of the polygon (in which case ContainsPoints returns 2 or 3), and decide what to do with those, since they do not clearly belong to just one polygon. The way I wrote it, edge and vertex points will not be counted on any polygons. Another possible approach is to count them only in the first polygon where they hit the edge/vertex, which would be done just changing the test to

```
w=where(match ne 0,count,complement=wc,ncomplement=nc)
```

Paulo

On Friday, May 1, 2015 at 12:33:04 PM UTC-3, Paula wrote:

```
> hey there,  
> i'm looking for ideas on how to best solve this.  
>  
> I have 2 sets of x, y data that I will, non-creatively, call 'data1' and 'data2'. Using data1 I have  
built Voronoi cells/polygons to represent the x,y space. Now I need to discover how many points  
from data2 fall in each of the Voronoi cells built from data1. In other words, I need to be able to  
assign each of the data1's Voronoi cell to all points in data2 falling inside that cell.  
>  
> Firstly I was suggested to use INSIDE function, but that doesn't look good to me because it  
involves going through a double loop: for every x,y point in data2, test each of the voronoi  
polygons until I discover a match. That FOR+WHILE structure will be slow, specially given that  
data2 is much larger than the number of Voronoi polygons (I'm "FOR-looping" through the largest  
of the data arrays, besides I know IDL "hates" loops :)).  
>  
> I'm wondering how I can build an algorithm differently... if I think about the two datasets as if  
they were 2D images, I could FOR-loop through the Voronoi cells (the smallest dataset) and use  
them as a mask. It would be something like:  
>  
> 1) build a 2D image from data2  
> 2) FOR loop: for each voronoi polygon, make a 2D image where the inside of the loop polygon  
equals to 1, everything else equals to 0  
> 3) multiply the "voronoi mask" in step 2) by the image built in step 1)  
> The non-zero results of the multiplication in step above are exactly the points in data2 that I'm  
looking for (in true, I only need the total number of points, so i don't even need to track back to the  
original x,y data2 values).  
> 4) Write out the number of non-zero points, close the FOR loop.  
> Done! (done in only 1 FOR loop, looping through the smallest dataset)  
>  
> Does that reasoning make sense?  
> How can I efficiently build an image from a) a set of x,y points and b) a polygon?  
> If I don't find a way to efficiently built the images, better to stick to the inefficient loop over the  
larger x,y array using INSIDE I guess... (or not?)  
>  
> thanks,  
> Paula
```
