Subject: data inside a circle
Posted by angela on Fri, 10 Jun 2005 21:01:37 GMT
View Forum Message <> Reply to Message

Hello,
I have a list of x,y coordinates and I am trying to make circles of different centers and radii and find how many of the data points are inside in the different circles. Do you happen to know an elegant way to do that?
I have tried to do that using FOR loops but my data points are about 10,000 and the number of circles are about 160,000, so it takes about 2-3 hours for my program to run. I was wondering if you know of a faster way to do that.

Thank you,
Angela

Subject: Re: data inside a circle
Posted by Michael Wallace on Mon, 13 Jun 2005 16:20:02 GMT
View Forum Message <> Reply to Message

> Thank you. I was wondering if there is a function or procedure in IDL
> that will find the coordinates of the bounding box for a circle and/or
> ellipse.

If the origin is at [x, y] and you have a radius r, the bounding box is defined by [x - r, y - r] for the lower left-hand corner and [x + r, y + r] for the upper-right hand corner. If you have an ellipse, just replace r with the radius in the x direction and radius in the y direction. This assumes that you don't have any ellipses that are tilted.

-Mike

Subject: Re: data inside a circle
Posted by Michael Wallace on Tue, 14 Jun 2005 00:10:17 GMT
View Forum Message <> Reply to Message

I've been thinking about this a little more as well and wanted to throw out this suggestion for how to think about things. For this example, I'm going to ignore the elliptical case, but you should see how to extend this idea to cover ellipses as well.

Here's the approach in brief. Create a sorted X -> index mapping and a sorted Y -> index mapping. Calculate bounding box for a given circle. Search sorted X and Y lists for appropriate indexes. Intersect the

indexes returned -- this gives all points within the bounding box.
Check distance of point from the center of the circle to determine
inside or outside.

Now, here's the longer explanation. What you can do is create mappings into your unordered list of data points. You can create a mapping of X value -> index in unordered list and a mapping of Y value -> index in unordered list. These mappings can be sorted by X value and Y value respectively. You have to do this because you can't sort the list of data points in place such that you can search on X equally as fast as you can search on Y. You can create both the X mapping and the Y mapping at the same time, so when it's all said and done, you should only have a running time of O(n log n) so far.

No matter what, you must iterate over the list of circles, O(n). For each circle, you first calculate the X and Y ranges of the bounding box, O(1). Now you have to search the mappings for the range just calculated, O(log n). Calculate the intersection of the set of indexes that satisfy the X range and the set of indexes that satisfy the Y range, O(n). Finally, calculate the distance from the center of the circle to the resultant set of data points to determine if the particular data point is inside or outside the circle, O(n).

Finally, if multiple circles have the same center, you can sort that list, O(n log n), such that circles with the same center are grouped together and are then sub-ordered from the largest radius to the smallest. Get the bounding box for the largest circle and find the data points within the circle. Instead of searching the data point list again, just use the data points you already have and do the inside/outside check for each smaller circle, culling out more data points each time.

HTH,
Mike

Subject: Re: data inside a circle
Posted by btt on Mon, 20 Jun 2005 13:18:09 GMT
View Forum Message <> Reply to Message

Michael Wallace wrote:
> I've been thinking about this a little more as well and wanted to throw
> out this suggestion for how to think about things. For this example,
> I'm going to ignore the elliptical case, but you should see how to
> extend this idea to cover ellipses as well.
>
> Here's the approach in brief. Create a sorted X -> index mapping and a

> sorted Y -> index mapping. Calculate bounding box for a given circle.
> Search sorted X and Y lists for appropriate indexes. Intersect the
> indexes returned -- this gives all points within the bounding box. Check
> distance of point from the center of the circle to determine inside or
> outside.
>
>
> Now, here's the longer explanation. What you can do is create mappings
> into your unordered list of data points. You can create a mapping of X
> value -> index in unordered list and a mapping of Y value -> index in
> unordered list. These mappings can be sorted by X value and Y value
> respectively. You have to do this because you can't sort the list of
> data points in place such that you can search on X equally as fast as
> you can search on Y. You can create both the X mapping and the Y
> mapping at the same time, so when it's all said and done, you should
> only have a running time of O(n log n) so far.
>
> No matter what, you must iterate over the list of circles, O(n). For
> each circle, you first calculate the X and Y ranges of the bounding box,
> O(1). Now you have to search the mappings for the range just
> calculated, O(log n). Calculate the intersection of the set of indexes
> that satisfy the X range and the set of indexes that satisfy the Y
> range, O(n). Finally, calculate the distance from the center of the
> circle to the resultant set of data points to determine if the
> particular data point is inside or outside the circle, O(n).
>
> Finally, if multiple circles have the same center, you can sort that
> list, O(n log n), such that circles with the same center are grouped
> together and are then sub-ordered from the largest radius to the
> smallest. Get the bounding box for the largest circle and find the data
> points within the circle. Instead of searching the data point list
> again, just use the data points you already have and do the
> inside/outside check for each smaller circle, culling out more data
> points each time.
>
> HTH,
> Mike

Hi,

I guess I am a bit late with this but the following is from the online
help to IDL 6.1

" The IDLanROI::ContainsPoints function method determines whether the
given data coordinates are contained within the closed polygon region."

So, you could define the circle boundary as the ROI and pass the points
to the object method.

Ben

Subject: Re: data inside a circle
Posted by James Kuyper on Tue, 21 Jun 2005 12:01:35 GMT
View Forum Message <> Reply to Message

Ben Tupper wrote:
...
> I guess I am a bit late with this but the following is from the online
> help to IDL 6.1
>
> " The IDLanROI::ContainsPoints function method determines whether the
> given data coordinates are contained within the closed polygon region."
>
> So, you could define the circle boundary as the ROI and pass the points
> to the object method.

A circle is not a closed polygon. It can be approximated with arbitrary accuracy by a closed polygon with a sufficiently large number of sides. However, as long as you use a finite number of sides, there will always be a certain amount of inaccuracy in that approximation. The more sides you use, the slower the comparison; at some desired level of accuracy, it's quicker to perform the correct test for being inside a circle, than it is to test for being inside a polygon approximation to a circle.

In any event, the key problem in this particular problem is not the test for being inside a single cirle; the problem is that the test is against a very large number of circles. Doing that efficiently in IDL is tricky; and it's not clear to me that IDLanROI::ContainsPoints helps address that problem.

Subject: Re: data inside a circle
Posted by btt on Tue, 21 Jun 2005 20:09:57 GMT
View Forum Message <> Reply to Message

kuyper@wizard.net wrote:
> Ben Tupper wrote:
> ...
>
>> I guess I am a bit late with this but the following is from the online
>> help to IDL 6.1
>>
>> " The IDLanROI::ContainsPoints function method determines whether the

>> given data coordinates are contained within the closed polygon region."
>>
>> So, you could define the circle boundary as the ROI and pass the points
>> to the object method.
>
>
> A circle is not a closed polygon. It can be approximated with arbitrary
> accuracy by a closed polygon with a sufficiently large number of sides.
> However, as long as you use a finite number of sides, there will always
> be a certain amount of inaccuracy in that approximation. The more sides
> you use, the slower the comparison; at some desired level of accuracy,
> it's quicker to perform the correct test for being inside a circle,
> than it is to test for being inside a polygon approximation to a
> circle.
>
> In any event, the key problem in this particular problem is not the
> test for being inside a single cirle; the problem is that the test is
> against a very large number of circles. Doing that efficiently in IDL
> is tricky; and it's not clear to me that IDLanROI::ContainsPoints helps
> address that problem.
>

Yes and yes. I get it now.

I have cobbled together a test procedure that shows, as you hint, that IDLanROI::ContainsPoints is not a light-footed solution.

Cheers,
Ben

```
*****START
PRO testHoop, $
   nIter = nIter, $  ;number of iterations
   nP = nP, $        ;number of scatter points
   nC = nC           ;number of points that make
                     ;up the circular polygon

if n_elements(nIter) EQ 0 then nIter = 2
if n_elements(nP) EQ 0 Then nP = 10000
if n_elements(nC) EQ 0 then nC = 1000
x = RANDOMU(s, nP[0])
y = RANDOMU(s, nP[0])
roi = OBJ_NEW('IDLanROI' )

   ;make the points on the circle
;(offset = [0,0] and radius = 1 to start)
points = ((2.0 * !Pi )/(nC[0]-1.0) ) * FINDGEN(nC[0])
```

```
xp = COS(points )
yp = SIN(points)

    ;the elapsed time over all iterations
dt = 0.0d

For i = 0, nIter[0]-1 do Begin

    start = systime(/sec)

    rxy = RANDOMU(s,3)

    cx = rxy[1] + rxy[0] * xP
    cy = rxy[2] + rxy[0] * yP
    roi->setproperty, data = transpose([[cx],[cy]])

    ok = roi->ContainsPoints(x,y)

    fini = systime(/sec)
    dt  += (fini-start)
    print, 'iter, radius, x0, y0 ', i,  rxy
EndFor

print, 'elapsed time (s)  = ', dt
print, 'time per iter (s) = ', dt/niter

OBJ_DESTROY, roi
end

*****END
```