Subject: Point Matching...
Posted by aisiteru31 on Tue, 26 Aug 2014 19:47:35 GMT
View Forum Message <> Reply to Message

Hi, everyone

I have a problem that I'm trying to find an efficient way to solve.

I have a large list of points (about 2750L*2750L), that I need to match to a second list of points (about 1000L*1000L).
The points are located on the surface of the Earth, so they are given in lat/lon coordinates. I have looked at functions (like map_2points), but brute force way of matching them would be way too slow.

I noticed a user written library in the documents of Exelis called match_sph that seem to do what I need, but I can't find where to download it.

Any help is greatly appreciated,

Thanks!

Subject: Re: Point Matching...
Posted by Russell Ryan on Wed, 27 Aug 2014 17:49:52 GMT
View Forum Message <> Reply to Message

Ok, so this is a very long answer to a simple question. I am an astronomer that matches catalogs of objects all the time. I wrote an object to handle these catalogs and do the matching, I've attached a snippet of code that does the matching. The top (which I've omitted) just extracts the relevant fields from the self structure, but should be obvious.

x1,y1 are the coordinates from the first catalog
x2,y2 are the coordinates from the second catalog

You'll need hist_nd from JD Smith, it's easy to find.

Basically, you tell the algorithm the "search radius" to look for matches. It then histograms the points in bins that are this size, then only computes the distance metric (in your case ti's a spierical one) for points which are in adjacent grid cells. Then applies the matching criterion to those adjacent cells. THis is many orders of magnitude faster than brute force matching. I didn't create this algorithm, I've seen various versions on line, but found my implementation is faster.

Good luck,
Russell

```
;get the number of objects to match
```

```
nobj=n_elements(x1)

;get the ranges
xr=[min(x1)<min(x2),max(x1)>max(x2)]
yr=[min(y1)<min(y2),max(y1)>max(y2)]
mn=[xr(0),yr(0)]
mx=[xr(1),yr(1)]


;property of the bins
nbins=ceil([xr(1)-xr(0),yr(1)-yr(0)]/rad)
binsz=[rad,rad]

;histogram the data
 h1=hist_nd([transpose(x1),transpose(y1)],binsz,min=mn,max=mx ,reverse=ri1)
 h2=hist_nd([transpose(x2),transpose(y2)],binsz,min=mn,max=mx ,reverse=ri2)

;compute the indices
match=replicate({g1:0L,g2:-1L,dist:!values.f_nan},nobj)
match.g1=lindgen(nobj)

;only process the bins with data in them
guse=where(h1 ne 0,nuse)
for i=0L,nuse-1 do begin
   ;coordinates from set 1
   xx1=x1[ri1[ri1[guse(i)]:ri1[guse(i)+1]-1]]
   yy1=y1[ri1[ri1[guse(i)]:ri1[guse(i)+1]-1]]

   ;2-d indices of the bin of interest
   cc=array_indices(nbins,guse(i),/dim)

   ;get the 3x3 box around bin of interest
   i0=(cc(0)-1)>0 & i1=(cc(0)+1)<(nbins(0)-1)
   j0=(cc(1)-1)>0 & j1=(cc(1)+1)<(nbins(1)-1)

   ;number of potential matches in set 2
   n2=total(h2(i0:i1,j0:j1),/preserve)

   ;only operate if 3x3 bins have data from set 2 in them
   if n2 ne 0 then begin

      ;dimensions of the subgrid to search
      dims=[h1(guse(i)),n2]
      d1=[h1(guse(i)),1]
      d2=[1,n2]

      ;store the data from set 2
      xx2=replicate(0.0,n2)
```

```
  yy2=replicate(0.0,n2)
  ii2=replicate(-1L,n2)
  k=0L

  ;loop the 3x3 set for the bin of interest
  for jj=j0,j1 do begin
    for ii=i0,i1 do begin

      ;compute bin ID
      bb=ii+jj*nbins(0)

      ;number of points in one of the 3x3 bins
      nt=ri2[bb+1]-ri2[bb]
      if nt gt 0 then begin
        ;record the ID and (x,y) pair
        ii2[k:k+nt-1]=ri2[ri2[bb]:ri2[bb+1]-1]
        xx2[k:k+nt-1]=x2[ii2[k:k+nt-1]]
        yy2[k:k+nt-1]=y2[ii2[k:k+nt-1]]
        k+=nt
      endif
    endfor
  endfor

  ;compute the deltas
   dx=rebin(reform(xx2,d2,/over),dims)-rebin(reform(xx1,d1,/ove r),dims)
   dy=rebin(reform(yy2,d2,/over),dims)-rebin(reform(yy1,d1,/ove r),dims)

  ;the distance metric
  if keyword_set(SPHERICAL) then begin
    cdec=cos(((cc(1)-0.5)*binsz(0)+yr(0))/!radeg)
    dr2=dy*dy+dx*dx*cdec
  endif else begin
    dr2=dy*dy+dx*dx
  endelse

  ;check if IDL drops a last dimension.  if so, put it back.
  if n2 eq 1 then dr2=reform(dr2,d1,/over)

  ;compute the minimum
   match[ri1[ri1[guse(i)]:ri1[guse(i)+1]-1]].dist=sqrt(min(dr2, id,dim=2))

  ;compute the bin of the best distnace
  c2=array_indices(dims,id,/dim)

  ;compute the index of the set 2
   match[ri1[ri1[guse(i)]:ri1[guse(i)+1]-1]].g2=ii2[reform(c2(1 ,*))]

endif
```

```
  endfor

  ;check to make sure something got matched
  g=where(match.g2 ne -1,count)
  if count ne 0 then match=match(g) else begin
     self->Message,'There were no matches.'
     match=-1b
     return
  endelse


  ;ok, matches at this point are best matches, but best is not
  ;necessarily smaller than requested search radius, so trim those.
  if keyword_set(STRICT) then begin
     g=where(match.dist le rad,count)
     if count ne 0 then match=match(g) else $
        self->Message,'STRICT rendered no viable matches.'
  endif

  if keyword_set(SELECT) then begin
     ;now select only the good matches
     self->SelectRows,match.g1
     cat2->SelectRows,match.g2
  endif
```

O
n Tuesday, August 26, 2014 3:47:35 PM UTC-4, aisiteru31 wrote:
> Hi, everyone
>
>
>
> I have a problem that I'm trying to find an efficient way to solve.
>
>
>
> I have a large list of points (about 2750L*2750L), that I need to match to a second list of points (about 1000L*1000L).
>
> The points are located on the surface of the Earth, so they are given in lat/lon coordinates. I have looked at functions (like map_2points), but brute force way of matching them would be way too slow.
>
>

>
> I noticed a user written library in the documents of Exelis called match_sph that seem to do what I need, but I can't find where to download it.
>
>
>
> Any help is greatly appreciated,
>
>
>
> Thanks!

Subject: Re: Point Matching...
Posted by aisiteru31 on Wed, 27 Aug 2014 19:54:49 GMT
View Forum Message <> Reply to Message

Thanks, Russell !

You have been of great help !
