Subject: Re: algorithm question. Can I get rid of the for loop?
Posted by Jeremy Bailin on Fri, 22 Mar 2013 03:39:29 GMT
View Forum Message <> Reply to Message

On 3/21/13 4:32 PM, Sï¿½ren Frimann wrote:
> Hi All.
>
> I have an implementation of a hampel filter (see e.g. http://exploringdatablog.blogspot.dk/2012/01/moving-window-f ilters-and-pracma.html) in IDL.
>
> My implementation looks like this:
>
> ##
> FUNCTION hampel, x, y, dx, THRESHOLD=threshold
>
> Compile_Opt idl2
>
> IF N_Elements(threshold) EQ 0 THEN $
> threshold = 3
>
> ;initialize arrays
> s0 = FltArr(N_Elements(y))
> y0 = FltArr(N_Elements(y))
> yy = y
>
> FOR i=0,N_Elements(y)-1 DO BEGIN
> index = Where((x GE x[i] - dx) AND (x LE x[i] + dx))
> y0[i] = Median(y[index]) ; Median filtering
> s0[i] = 1.4826*Median(Abs(y[index] - y0[i])) ;estimating uncertainty
> ENDFOR
>
> ol = Where(Abs(y - y0) GE threshold*s0) ;Index of outliers
> yy[ol] = y0[ol]
>
> result = Create_Struct('y',yy, $
> 'sigma',s0)
>
> RETURN, result
>
> END
> ##
>
> the filter runs a moving window of width 2*dx measured in the same units as x.
> x is generally not uniformly spaced (so there's not a constant number of points inside the window as it moves).
> x and y can be quite long vectors so the filter takes a long time to run.
> Can anyone see any method for speeding the code up?
> Any help would be much appreciated!

>
> Cheers,
> Sï¿½ren
>

If your x elements are spaced regularly (so that your window always includes the same number of elements), and the width of the moving window isn't too large (so you don't run into memory problems), then you can use this trick to do vector operations on the moving window:

Let's say the half-width of the window is k (equivalent to your dx if the x spacing is 1). Then the width of the window is

nwindow = 2*k+1

If your data series "data" has ndata elements, then you can create an index array into data where the first dimension is where along the series the window is centered, and the second dimension is where within the window you are.

```
datawindowindex = rebin(lindgen(ndata,1), ndata, nwindow, /sample) + $
   rebin(lindgen(1,nwindow), ndata, nwindow, /sample) - k
```

For example, for an 8-element data series with a k=2 (5-element) window:

```
IDL> print, datawindowindex
      -2      -1       0       1       2       3       4       5
      -1       0       1       2       3       4       5       6
       0       1       2       3       4       5       6       7
       1       2       3       4       5       6       7       8
       2       3       4       5       6       7       8       9
```

So the values within the window when evaluating element i are data[datawindowindex[i,*]]. Note that at the edges (values of -2, -1, 8, and 9), this will effectively replicate the end elements because of the way IDL truncates out-of-bound indices, which is generally acceptable behaviour.

You can then do your array operations over the window by doing them over the second dimension. For example, the median value is simply:

```
y0 = median(data[datawindowindex], dimension=2)
```

Similarly, the sigma calculation is

```
s0 = 1.4826 * median( abs( data[datawindowindex] - rebin(y0, ndata,$
  nwindow, /sample) ), dimension=2)
```

If your x values are *not* evenly spaced, then I don't have any obvious ideas that aren't horrendous memory hogs.

-Jeremy.
