Subject: Technique to find maximum in 100x100 element moving box
Posted by sam.tushaus on Wed, 12 Oct 2016 21:23:20 GMT
View Forum Message <> Reply to Message

I have a 3200x3248 array (satellite swath), and I would like to find the maximum value in a 100x100 element moving box. It's exactly like using smooth or convol to find the mean in a 100x100 element moving box, but I need the max. Right now I have it hard-coded with FOR loops to take into account the edges, but it's veeeeery slow.

Does anyone have any advice on how to speed this up using IDL functions (rebin, histogram, anything of that sort)?

Thanks for your time!

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by sam.tushaus on Wed, 12 Oct 2016 21:26:15 GMT
View Forum Message <> Reply to Message

For reference, this is my current code in which I essentially use a truncated edge method.

```
FOR i = 0, nx-1 DO BEGIN
 FOR j = 0, ny-1 DO BEGIN
  IF (i-50) LT 0 THEN low_ind_i = 0 ELSE low_ind_i = i-50
  IF (j-50) LT 0 THEN low_ind_j = 0 ELSE low_ind_j = j-50
  IF (i+50) GT (nx-1) THEN hi_ind_i = nx-1 ELSE hi_ind_i = i+50
  IF (j+50) GT (ny-1) THEN hi_ind_j = ny-1 ELSE hi_ind_j = j+50
  data_max[i,j] = max(data[low_ind_i:hi_ind_i,low_ind_j:hi_ind_j])
 ENDFOR
ENDFOR
```

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by lecacheux.alain on Wed, 12 Oct 2016 21:38:31 GMT
View Forum Message <> Reply to Message

Le mercredi 12 octobre 2016 23:23:22 UTC+2, Samantha Tushaus a écrit :
> I have a 3200x3248 array (satellite swath), and I would like to find the maximum value in a 100x100 element moving box. It's exactly like using smooth or convol to find the mean in a 100x100 element moving box, but I need the max. Right now I have it hard-coded with FOR loops to take into account the edges, but it's veeeeery slow.
>
> Does anyone have any advice on how to speed this up using IDL functions (rebin, histogram, anything of that sort)?
>
> Thanks for your time!

max_array = dilate(your_array, replicate(1,100,100), /GRAY) ?

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by Markus Schmassmann on Thu, 13 Oct 2016 10:36:50 GMT
View Forum Message <> Reply to Message

On 10/12/2016 11:26 PM, Samantha Tushaus wrote:
> For reference, this is my current code in which I essentially use a truncated edge method.
>
> FOR i = 0, nx-1 DO BEGIN
> FOR j = 0, ny-1 DO BEGIN
> IF (i-50) LT 0 THEN low_ind_i = 0 ELSE low_ind_i = i-50
> IF (j-50) LT 0 THEN low_ind_j = 0 ELSE low_ind_j = j-50
> IF (i+50) GT (nx-1) THEN hi_ind_i = nx-1 ELSE hi_ind_i = i+50
> IF (j+50) GT (ny-1) THEN hi_ind_j = ny-1 ELSE hi_ind_j = j+50
> data_max[i,j] = max(data[low_ind_i:hi_ind_i,low_ind_j:hi_ind_j])
> ENDFOR
> ENDFOR
still looping, but faster without the ifs:

```
data_max=make_array(size(data,/dim),type=size(data,/type))
FOR i = 0, nx-1 DO FOR j = 0, ny-1 DO data_max[i,j] = $
    max(data[(i-50)>0:(i+50)<nx-1,(j-50)>0:(j+50)<ny-1])
```

There might be a way with histogram and it's reverse index (possibly
using ORD beforehand if data is float), but depending on the data
(mostly how many different values end up being in data_max and how many
values larger than it's smallest one exist) it might not be faster.

You can also try looping twice over -50 to 50, but i doubt that would be
faster.

Good luck, Markus

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by sam.tushaus on Thu, 13 Oct 2016 15:02:12 GMT
View Forum Message <> Reply to Message

Interestingly, my method, the "dilute" method, and the no-ifs loop all take the same amount of time (63, 64, and 62 seconds, respectively).

Subject: Re: Technique to find maximum in 100x100 element moving box

Posted by Lajos Foldy on Thu, 13 Oct 2016 17:01:54 GMT
View Forum Message <> Reply to Message

On Thursday, October 13, 2016 at 5:02:14 PM UTC+2, Samantha Tushaus wrote:
> Interestingly, my method, the "dilute" method, and the no-ifs loop all take the same amount of time (63, 64, and 62 seconds, respectively).

Try this:

```
tmp1=transpose(data)
tmp2=fltarr(ny,nx,/nozero)
FOR i = 0, nx-1 DO BEGIN
      FOR j = 0, ny-1 DO BEGIN
            IF (j-m2) LT 0 THEN low_ind_j = 0 ELSE low_ind_j = j-m2
            IF (j+m2) GT (ny-1) THEN hi_ind_j = ny-1 ELSE hi_ind_j = j+m2
            tmp2[j,i] = max(tmp1[low_ind_j:hi_ind_j, i])
         ENDFOR
   ENDFOR
tmp2=transpose(tmp2)
data_max=fltarr(nx,ny,/nozero)
FOR j = 0, ny-1 DO BEGIN
      FOR i = 0, nx-1 DO BEGIN
            IF (i-m2) LT 0 THEN low_ind_i = 0 ELSE low_ind_i = i-m2
            IF (i+m2) GT (nx-1) THEN hi_ind_i = nx-1 ELSE hi_ind_i = i+m2
            data_max[i,j] = max(tmp2[low_ind_i:hi_ind_i,j])
         ENDFOR
   ENDFOR
```

regards,
Lajos

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by Lajos Foldy on Thu, 13 Oct 2016 17:58:48 GMT
View Forum Message <> Reply to Message

On Thursday, October 13, 2016 at 7:01:56 PM UTC+2, fawltyl...@gmail.com wrote:

> Try this:

Revised version, faster and without ifs:

```
pro test
nx=3200
ny=3248
m=100

m2=m/2
```

```
seed=123
data=randomu(seed,nx,ny)

tic
tmp1=transpose(data)
tmp2=fltarr(ny,nx,/nozero)
FOR i = 0, nx-1 DO BEGIN
       FOR j = 0, m2         DO tmp2[j,i] = max(tmp1[0   :j+m2, i])
       FOR j = m2+1, ny-m2-1 DO tmp2[j,i] = max(tmp1[j-m2:j+m2, i])
       FOR j = ny-m2, ny-1   DO tmp2[j,i] = max(tmp1[j-m2:ny-1, i])
    ENDFOR
tmp2=transpose(tmp2)
data_max=fltarr(nx,ny,/nozero)
FOR j = 0, ny-1 DO BEGIN
       FOR i = 0, m2         DO data_max[i,j] = max(tmp2[0   :i+m2, j])
       FOR i = m2+1, nx-m2-1 DO data_max[i,j] = max(tmp2[i-m2:i+m2, j])
       FOR i = nx-m2, nx-1   DO data_max[i,j] = max(tmp2[i-m2:nx-1, j])
    ENDFOR
toc

end
```

regards,
Lajos

ps: the i-50:i+50 subscript range has 101 elements, not 100.

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by Lajos Foldy on Thu, 13 Oct 2016 18:56:46 GMT
View Forum Message <> Reply to Message

On Thursday, October 13, 2016 at 7:58:50 PM UTC+2, fawltyl...@gmail.com wrote:
> On Thursday, October 13, 2016 at 7:01:56 PM UTC+2, fawltyl...@gmail.com wrote:
>
>> Try this:
>
> Revised version, faster and without ifs:
>
> pro test
> nx=3200
> ny=3248
> m=100
>
> m2=m/2
> seed=123
> data=randomu(seed,nx,ny)

```
>
>  tic
>  tmp1=transpose(data)
>  tmp2=fltarr(ny,nx,/nozero)
>  FOR i = 0, nx-1 DO BEGIN
>         FOR j = 0, m2         DO tmp2[j,i] = max(tmp1[0   :j+m2, i])
>         FOR j = m2+1, ny-m2-1 DO tmp2[j,i] = max(tmp1[j-m2:j+m2, i])
>         FOR j = ny-m2, ny-1   DO tmp2[j,i] = max(tmp1[j-m2:ny-1, i])
>      ENDFOR
>  tmp2=transpose(tmp2)
>  data_max=fltarr(nx,ny,/nozero)
>  FOR j = 0, ny-1 DO BEGIN
>         FOR i = 0, m2         DO data_max[i,j] = max(tmp2[0   :i+m2, j])
>         FOR i = m2+1, nx-m2-1 DO data_max[i,j] = max(tmp2[i-m2:i+m2, j])
>         FOR i = nx-m2, nx-1   DO data_max[i,j] = max(tmp2[i-m2:nx-1, j])
>      ENDFOR
>  toc
>
>  end
>
>
>  regards,
>  Lajos
>
>  ps: the i-50:i+50 subscript range has 101 elements, not 100.
```

Last version, with a real sliding window. This one is about 30x faster than the original code for random data.

```
pro test
nx=3200
ny=3248
m=100
m2=m/2
seed=123
data=randomu(seed,nx,ny)

tic
tmp1=transpose(data)
tmp2=fltarr(ny,nx,/nozero)
FOR i = 0, nx-1 DO BEGIN
       FOR j = 0, m2         DO tmp2[j,i] = max(tmp1[0     :j+m2, i])
       maxi=tmp2[m2,i]
       FOR j = m2+1, ny-m2-1 DO begin
          if maxi eq tmp1[j-m2-1, i] then begin
             maxi=max(tmp1[j-m2:j+m2, i])
          endif else maxi=maxi>tmp1[j+m2, i]
          tmp2[j,i]=maxi
```

```
      endfor
      FOR j = ny-m2, ny-1   DO tmp2[j,i] = max(tmp1[j-m2  :ny-1, i])
    ENDFOR
tmp2=transpose(tmp2)
data_max=fltarr(nx,ny,/nozero)
FOR j = 0, ny-1 DO BEGIN
      FOR i = 0, m2         DO data_max[i,j] = max(tmp2[0     :i+m2, j])
      maxi=data_max[m2,j]
      FOR i = m2+1, nx-m2-1 DO begin
        if maxi eq tmp2[i-m2-1, j] then begin
          maxi=max(tmp2[i-m2:i+m2, j])
        endif else maxi=maxi>tmp2[i+m2, j]
        data_max[i,j]=maxi
      endfor
      FOR i = nx-m2, nx-1   DO data_max[i,j] = max(tmp2[i-m2  :nx-1, j])
    ENDFOR
toc
end
```

regards,
Lajos

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by Heinz Stege on Thu, 13 Oct 2016 21:59:52 GMT
View Forum Message <> Reply to Message

On Thu, 13 Oct 2016 08:02:12 -0700 (PDT), Samantha Tushaus wrote:

> Interestingly, my method, the "dilute" method, and the no-ifs loop all take the same amount of time (63, 64, and 62 seconds, respectively).

Congratulation first, you have a very fast computer. My one needs
273 s if data is choosed by calculated by
  data=randomn(seed,3200,3248).

However your code can be optimized. The following needs about 18 s (on
my slow computer):

```
function sam_v02,data
;--------------------
;
temp=size(data,/dimensions)
nx=temp[0]
ny=temp[1]
;
```

```
low_ind_x=indgen(nx)-50
low_ind_x[0:49]=0
low_ind_y=indgen(ny)-50
low_ind_y[0:49]=0
hi_ind_x=indgen(nx)+50
hi_ind_x[-50:*]=nx-1
hi_ind_y=indgen(ny)+50
hi_ind_y[-50:*]=ny-1
;
data_type=size(data,/type)
data_max=make_array(nx,ny,type=data_type)
temp_max=make_array(ny,type=data_type)
FOR i=0,nx-1 DO BEGIN
  low_i=low_ind_x[i]
  hi_i=hi_ind_x[i]
  for j=0,ny-1 do temp_max[j]=max(data[low_i:hi_i,j])
  for j=0,ny-1 do $
    data_max[i,j]=max(temp_max[low_ind_y[j]:hi_ind_y[j]])
ENDFOR
;
return,data_max
end
```

First I removed the index calculation from the loop. But that made only a slow difference. Then I splitted the 2-dimensional calculation of the maximum into two 1-dimensional calculations (the two loops over j).

This calculation is more than 15 times faster than the original one.

Cheers, Heinz

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by Heinz Stege on Fri, 14 Oct 2016 17:25:08 GMT
View Forum Message <> Reply to Message

On Thu, 13 Oct 2016 23:59:52 +0200, Heinz Stege wrote:

[...]

Ooops, Lajos used this ideas hours before. Sorry for not seeing this.

Look for his last post on yesterday. His optimization is more comprehensive and makes the calculation one more faster.

Heinz

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by wlandsman on Sat, 15 Oct 2016 23:23:40 GMT
View Forum Message <> Reply to Message

On Thursday, October 13, 2016 at 11:02:14 AM UTC-4, Samantha Tushaus wrote:
> Interestingly, my method, the "dilute" method, and the no-ifs loop all take the same amount of time (63, 64, and 62 seconds, respectively).

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by wlandsman on Sat, 15 Oct 2016 23:32:30 GMT
View Forum Message <> Reply to Message

On Thursday, October 13, 2016 at 11:02:14 AM UTC-4, Samantha Tushaus wrote:
> Interestingly, my method, the "dilute" method, and the no-ifs loop all take the same amount of time (63, 64, and 62 seconds, respectively).

I surprised that you find DILATE() takes about the same amount of time as the other algorithms. I find DILATE() to be even faster than the fast algorithms presented by Lajos and Heinz. This is not surprising, since DILATE() is an intrinsic IDL function coded in C, and so will always be faster than the same algorithm coded in IDL.

However, there are two drawbacks to DILATE(). First, it only works on non-negative integer data, including ULONG. The default is to return byte data and you have to set the /ULONG or /PRESERVE_TYPE keywords to get it to return 32 bit data. The second problem is that DILATE() does not seem to handle the edges (where the moving box goes off the edge of the array). In fact, I can't figure out what DILATE() does with the edge values, but it is not the obvious of trimming the size of the box so it fits within the array.

Subject: Re: Technique to find maximum in 100x100 element moving box
Posted by sam.tushaus on Fri, 28 Oct 2016 18:49:54 GMT
View Forum Message <> Reply to Message

The solutions suggested by Lajos and Heinz are so much faster, thank you! The results are not exactly identical to my original version, but I'm going to double check everything to see if I can find the source of the error.

Thanks again! I learn something new every time I come here!