Subject: array manipulation problem
Posted by Martin B. Schmidt on Wed, 07 Jun 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Hi,

Given:

```
  n=500
  a=FltArr(n,n,n)
```

How can I assign
  a[i,j,k] = i^2+j^2+k^2   for all i,j,k (i,j,k=0,1,2,..,n-1)
without using any loop?

Any idea?

Martin

Subject: Re: Array manipulation
Posted by John-David T. Smith on Wed, 01 Nov 2000 17:00:24 GMT
View Forum Message <> Reply to Message

Leon Majewski wrote:
>
> Hello
> I was wondering whether any array minded person could suggest a way of using array
> indicies to chop up a large array into ordered windows.
> I can't think of a way to do it with reform, translate (though i'm sure this is my
> limitation not a reform translate limitation)
>
> ---------
> ie given an array of 30*30 elements
> return 100 3*3 elements
> or 36 5*5
> or...
>
> in=
>
> 00 01 02 03 04 05..
> 30 31 32 33 34 35..
> 60 61 62 63 64 65..
>
> out = blocks such as
> 00 01 02
> 30 31 32
> 60 61 62

>
> each block is then processed to one representative number (ie mean or median....) and
> returned
>

If you just want mean or median you should use:

mean=smooth(arr,n)
med=median(arr,n)

where n=3 or 5 or whatever odd box size you want. If you really were only interested in windows centered at [2,2],[5,2], etc., you can trivially extract those elements. Usually the full boxcar is more interesting though.

There are other things you can do without resorting to individually extracting "windows" as you call them. For instance:

```
; the nxn window total
total=smooth(arr,n)*n^2
; the nxn window total not including central pixel
neighbors=smooth(arr,n)*n^2-arr
; the mean of the neighboring pixels (excluding central)
neighmean=(smooth(arr,n)*n^2-arr)/(n^2-1)
; the square deviation from that mean
sqdev=(arr-neighmean)^2
; the variance of an nxn window of data, excluding central pixel
imvar=(smooth(sqdev,n)*n^2-sqdev)/(n^2-2)
```

With repeated applications of smooth you can generate a boxcar of any statistical moment you desire, and I guarantee it will be much faster than your code. See the EDGE_TRUNCATE keyword for dealing with edge issues. Cutoffs can be enforced by zeroing outliers, which will not affect the mean or average, but will affect higher moments.

As a bonus exercise, it's quite possible to do full exclude-certain-value (above max, below min, NaN, whatever) boxcar statistics using the same method. Precompute a mask of included elements, and use it judiciously throughout the computations. How can you count good elements in a given window? Simply total the mask as above. Be careful of integer truncation effects: (1+1)/3=0. Overcome this with a floating mask. You'll also have to add an fallback step if you expect boxcar windows with fewer valid elements than the highest moment number you wish to compute (i.e. 0 elements for a mean is pretty obvious).

JD

--
J.D. Smith | WORK: (607) 255-6263
Cornell Dept. of Astronomy | (607) 255-5842
304 Space Sciences Bldg. | FAX: (607) 255-5875
Ithaca, NY 14853 |

Subject: Re: Array manipulation
Posted by R.G.S. on Wed, 01 Nov 2000 17:44:35 GMT
View Forum Message <> Reply to Message

Leon Majewski <majewski@cygnus.uwa.edu.au> wrote in message
news:39ffa4ae.1813674@news.uwa.edu.au...
> Hello
> I was wondering whether any array minded person could suggest a way of
using array
> indicies to chop up a large array into ordered windows.
> I can't think of a way to do it with reform, translate (though i'm sure
this is my
> limitation not a reform translate limitation)
>
> ---------
> ie given an array of 30*30 elements
> return 100 3*3 elements
> or 36 5*5
> or...
>
> in=
>
> 00 01 02 03 04 05..
> 30 31 32 33 34 35..
> 60 61 62 63 64 65..
>
> out = blocks such as
> 00 01 02
> 30 31 32
> 60 61 62
>
> each block is then processed to one representative number (ie mean or
median....) and
> returned
>
> ------------
> What i've used so far is attached below (it does what i want, just slowly)
>
>
> leon

>
(snip)

> Leon Majewski
>
> Remote Sensing & Satellite Research Group
> Curtin University of Technology, Perth, Australia
>
> email: majewski@ses.curtin.edu.au

Hi Leon,
I am guessing that you can't do this with "reform".

Here is a quick array-index solution to extract a (that is, ONE) subarray.
Thus, if you have array "a" here which is 12 by 16, and you want
a 3 by3 subarray (set piecesize to 3 as done below), then run the code
below.
The srow and scol are the indices of the subarrays
(i.e. your 12 by 15 array can be thought of an array of 4 by 5 blocks,
and srow is 0..4-1 and scol is 0..5-1, get it?)

It is easy enough to turn this into a function.
If you want, this can be turned into a 3d array of indices, if you
need to produces indices for all subarrays. If you want that,
I can take another look at the indices function.

Cheers,
bob stockwell
stockwell (at) co-ra.com

```
; create a test case here
a = indgen(12,15)
ncols = (size(a))(1)


; Assumes square pieces
piecesize = 3 ; size of the square subarray (i.e. piecesize  by piecesize)
srow = 1 ; which row of the possible subsets
scol = 2 ; which column of the possible subsets

indices = indgen(piecesize,piecesize) +(intarr(piecesize) + 1) #
indgen(piecesize) * (ncols - piecesize) +piecesize*srow*ncols +
scol*piecesize
subarray = a(indices)
help,b
print,b
```

end

Subject: Re: Array manipulation
Posted by Jaco van Gorkom on Wed, 01 Nov 2000 17:59:16 GMT
View Forum Message <> Reply to Message

Hi,
I cannot suggest a general trick for chopping the array using subscripting.
If the mean is all that you are after, using IDL's built-in "rebin" function
for neighbourhood averaging is probably the fastest:

```
ArrDims = size(InputArr, /dimensions)
Average = rebin(InputArr, ArrDims[0]/Width, ArrDims[1]/Width)
```

This assumes a two-dimensional InputArr of which you have already forced the
dimensions to be a multiple of Width, like in your code.
This method does not allow to leave certain elements (e.g. <minval, >maxval,
or NaN) out of the calculation: NaN in one of the elements will cause the
average for an entire block to be NaN. I suppose this could be avoided by
setting the unwanted elements to zero so they do not really add to the mean,
and then applying an array of correction factors. So if e.g. 3 out of 9
elements in a block were set to zero, the calculated average from rebin
should be multiplied by 1.5 or, alternatively, divided by the fraction of
valid elements in the block. For just NaN-correction the code would look
something like this:

```
ArrDims = size(InputArr, /dimensions)
CorrArr = $
  rebin(float(finite(InputArr)), ArrDims[0]/Width, ArrDims[1]/Width)
InputArr[where(finite(InputArr, /NaN))] = 0
Average = rebin(InputArr, ArrDims[0]/Width, ArrDims[1]/Width)
CorrArr[where(CorrArr eq 0.)] = !values.f_nan     ; to avoid division by zero
Average = Average / CorrArr
```

I don't know if this will still be faster than smart subscripting in a loop,
it probably depends on the size of the arrays. It also uses somewhat more
memory. I hope that someone else knows a better solution. It would be best
for this type of NaN-handling to be a built-in option of IDL-routines like
rebin, smooth, etc. I would certainly be using it!

Rebin works for getting the mean value of each box. Maybe for getting the
median of each box one could use the built-in "median" filter function,
followed by a resampling of the array with rebin(,/sample). Median
calculates a median filter for each element of the original array, which is

much more and maybe much slower than what you need. Because of the way rebin resamples, you may need to shift the array first by half the box width.
Alternatively, do the resampling yourself by, once again, clever subscripting.

Jaco

Jaco van Gorkom
FOM-Instituut voor Plasmafysica "Rijnhuizen", The Netherlands
e-mail: gorkom@rijnh.nl

"Leon Majewski" <majewski@cygnus.uwa.edu.au> wrote in message news:39ffa4ae.1813674@news.uwa.edu.au...
> Hello
> I was wondering whether any array minded person could suggest a way of using array
> indicies to chop up a large array into ordered windows.
> I can't think of a way to do it with reform, translate (though i'm sure this is my
> limitation not a reform translate limitation)
>
> ---
> ie given an array of 30*30 elements
> return 100 3*3 elements
> or 36 5*5
> or...
>
> in=
>
> 00 01 02 03 04 05..
> 30 31 32 33 34 35..
> 60 61 62 63 64 65..
>
> out = blocks such as
> 00 01 02
> 30 31 32
> 60 61 62
>
> each block is then processed to one representative number (ie mean or median....) and
> returned
>
> ---
> What i've used so far is attached below (it does what i want, just slowly)
>

>
> leon

Subject: Re: Array manipulation
Posted by Vince Hradil on Wed, 01 Nov 2000 21:42:45 GMT
View Forum Message <> Reply to Message

How's this:

```
IDL> a = indgen(30,30)
IDL> r=reform(a,3,10,3,10)
IDL> print, a[0:2,0:2], r[*,0,*,0]
      0       1       2
     30      31      32
     60      61      62
      0       1       2

     30      31      32

     60      61      62

IDL> r2 = reform(a,5,6,5,6)
IDL> print, r2[*,0,*,0]
      0       1       2       3       4

     30      31      32      33      34

     60      61      62      63      64

     90      91      92      93      94

    120     121     122     123     124
```

"Leon Majewski" <majewski@cygnus.uwa.edu.au> wrote in message
news:39ffa4ae.1813674@news.uwa.edu.au...
> Hello
> I was wondering whether any array minded person could suggest a way of
using array
> indicies to chop up a large array into ordered windows.
> I can't think of a way to do it with reform, translate (though i'm sure
this is my
> limitation not a reform translate limitation)
>
> ---------
> ie given an array of 30*30 elements

> return 100 3*3 elements
> or 36 5*5
> or...
>
> in=
>
> 00 01 02 03 04 05..
> 30 31 32 33 34 35..
> 60 61 62 63 64 65..
>
> out = blocks such as
> 00 01 02
> 30 31 32
> 60 61 62
>
> each block is then processed to one representative number (ie mean or
median....) and
> returned
>
> ----------
> What i've used so far is attached below (it does what i want, just slowly)
>
>
> leon
>
>
>
> FUNCTION QMean2, DATA, $
> WIDTH=WIDTH, $
> MINVAL=MINVAL, $
> MAXVAL=MAXVAL, $
> SILENT=SILENT
>
> ;+
> ;NAME:
> ; QMean2
> ;
> ;PURPOSE:
> ; Calculate the mean of an array, excluding values 0 and NaN.
> ;
> ;CATEGORY:
> ; Array Manipulation
> ; Statistics
> ;
> ;CALLING SEQUENCE:
> ; Result = QMean(DATA)
> ;
> ;ARGUMENTS:

```
>  ; DATA
>  ; A 2-dimensional array to be averaged
>  ;
>  ;KEYWORDS:
>  ; MAXVAL
>  ; If the data is above this value it is omitted.
>  ; MINVAL
>  ; Only data greater than this value is included in the mean.
>  ; If this keyword is not set, it defaults to 0.
>  ; WIDTH
>  ; The width of the averaging box. If this keyword is not set then
>  ; it defaults to a value of 4.
>  ;
>  ;OUTPUTS:
>  ; An array holding the average of the input data.
>  ; This array is of size (Xdim/Width, Ydim/Width).
>  ;
>  ;SIDE EFFECTS;
>  ; If the size of the array holding the data is not a multiple of the
>  ; Width of the filter, then the Array is truncated to a size that is a
>  ; Multiple of the Width - The program Outputs a warning if this is the
>  ; case.
>  ;
>  ;EXAMPLES:
>  ; Result = QMean(findgen(20,20), Width = 5)
>  ; Result = QMean(findgen(20,20), Width = 5)
>  ; Result = QMean(findgen(20,20), Width = 15, MAXVAL = 210.3)
>  ;
>  ;MODICIFACTION HISTORY:
>  ; Created, Leon Majewski, 3rd August 2000
>  ;
>  ;-
>
>  IF Size(MINVAL, /n_elements) EQ 0 THEN MINVAL = 0
>  IF Size(WIDTH, /n_elements) EQ 0 THEN WIDTH = [4,4]
>  IF Size(WIDTH, /n_elements) EQ 1 THEN WIDTH = [WIDTH,WIDTH]
>  IF NOT KEYWORD_SET(SILENT) THEN SILENT = 1 ELSE SILENT = 0
>  ;start_time = SYSTIME(1)
>
>  DATA_IN = DATA
>  Size_Data = SIZE(DATA_IN)
>
>  xdim = Size_Data[1]/Width[0]
>  xdim_less1 = xdim-1
>  ydim = Size_Data[2]/Width[1]
>  ydim_less1 = ydim-1
>
>  IF Size_Data[1] NE xdim*width[0] THEN BEGIN
```

```
>  if silent then begin
>  print, 'The x dimension is not a multiple of the specified WIDTH.'
>  print, 'Reducing the size of the array from'
>  help,  DATA_IN
>  endif
>
>  DATA_IN = DATA_IN[0:(xdim)*width[0]-1, *]
>
>  if silent then help,  DATA_IN
>  ENDIF
>
>  IF Size_Data[2] NE ydim*width[1] THEN BEGIN
>  if silent then begin
>  print, 'The y dimension is not a multiple of the specified WIDTH.'
>  print, 'Reducing the size of the array from'
>  help,  DATA_IN
>  endif
>
>  DATA_IN = DATA_IN[*,0:(ydim)*width[1]-1]
>
>  if silent then help,  DATA_IN
>  ENDIF
>
>  IF KEYWORD_SET(MAXVAL) THEN $
>  IF MAXVAL GT MINVAL THEN $
>  DATA_IN = DATA_IN*(DATA_IN lt MAXVAL)
>
>  Size_Data = SIZE(DATA_IN)
>  n_els = Size_Data[4]
>
>  DATA_IN = REFORM(DATA_IN, Width[0], n_els/Width[0], /overwrite)
>  DATA_IN = REFORM(DATA_IN, Width[0], xdim, Width[1], ydim, /overwrite)
>
>  Average = FLTARR(xdim,ydim)
>
>  FOR j = 0, ydim_less1 DO BEGIN
>  FOR i = 0, xdim_less1 DO BEGIN
>
>  data_sub = REFORM(DATA_IN[*,i,*,j])
>  goodpos = WHERE(data_sub GT MINVAL AND $
>  FINITE(data_sub) EQ 1, c_goodpos)
>
>  IF c_goodpos NE 0 THEN BEGIN
>  data_sub = data_sub[goodpos]
>  Average[i,j] = TOTAL(data_sub)/N_ELEMENTS(data_sub)
>  ENDIF ELSE Average[i,j]=0
>
>  ENDFOR
```

> ENDFOR
>
> ; print, SYSTIME(1) - start_time, 's'
>
> RETURN, Average
> END
> ------------------------
> Leon Majewski
>
> Remote Sensing & Satellite Research Group
> Curtin University of Technology, Perth, Australia
>
> email: majewski@ses.curtin.edu.au

Subject: Re: Array manipulation
Posted by Martin Schultz on Thu, 02 Nov 2000 08:58:43 GMT
View Forum Message <> Reply to Message

Leon Majewski wrote:
>
> Hello
> I was wondering whether any array minded person could suggest a way of using array
> indicies to chop up a large array into ordered windows.
> I can't think of a way to do it with reform, translate (though i'm sure this is my
> limitation not a reform translate limitation)
>
> ---------
> ie given an array of 30*30 elements
> return 100 3*3 elements
> or 36 5*5
> or...
>
> in=
>
> 00 01 02 03 04 05..
> 30 31 32 33 34 35..
> 60 61 62 63 64 65..
>
> out = blocks such as
> 00 01 02
> 30 31 32
> 60 61 62
>
> each block is then processed to one representative number (ie mean or median....) and
> returned
>
> ------------

> What i've used so far is attached below (it does what i want, just slowly)
> [snipped][/color]

For the extraction of subportions of an array you could try out my
arrex routine which you can find at
 http://www.mpimet.mpg.de/~schultz.martin/idl/html/libmartin_ schultz.html
. It's fairly general, but I am not sure if it is the fastest for
extracting similar array portions over and over again.

Cheers,
Martin

--

 [[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
 [[[[[[[[[

Subject: Re: Array manipulation
Posted by majewski on Sat, 04 Nov 2000 04:28:22 GMT
View Forum Message <> Reply to Message

Thanks Jaco (& everyone else) this works very well for my purposes (much faster than my
for loop stuff).
I'll look into the median aspect latter

leon

On Wed, 1 Nov 2000 17:59:16 -0000, "Jaco van Gorkom" <gorkom@rijnh.nl> wrote:
> <snip>
> For just NaN-correction the code would look
> something like this:
>
> ArrDims = size(InputArr, /dimensions)
> CorrArr = $
>   rebin(float(finite(InputArr)), ArrDims[0]/Width, ArrDims[1]/Width)
> InputArr[where(finite(InputArr, /NaN))] = 0
> Average = rebin(InputArr, ArrDims[0]/Width, ArrDims[1]/Width)
> CorrArr[where(CorrArr eq 0.)] = !values.f_nan   ; to avoid division by zero
> Average = Average / CorrArr
>
> I don't know if this will still be faster than smart subscripting in a loop,

> it probably depends on the size of the arrays. It also uses somewhat more
> memory. I hope that someone else knows a better solution. It would be best
> for this type of NaN-handling to be a built-in option of IDL-routines like
> rebin, smooth, etc. I would certainly be using it!
> <snip>
> Jaco
>
> ----------
> Jaco van Gorkom
> FOM-Instituut voor Plasmafysica "Rijnhuizen", The Netherlands
> e-mail: gorkom@rijnh.nl
>

Leon Majewski

Remote Sensing & Satellite Research Group
Curtin University of Technology, Perth, Australia

email: majewski@ses.curtin.edu.au
