Subject: set all elements in 2d array between some range to 1
Posted by havok2063 on Fri, 22 May 2015 21:14:43 GMT
View Forum Message <> Reply to Message

So I'm trying to set all elements of a 2d-array that are between some padding, based off elements in another vector, to 1. Creating a mask of 1's and 0's.

I want to turn this bit of code, which runs in 30 seconds, into a non-loop bit of code that runs faster.

wave = 2d array of floats - size [4112,709]
skywave = 1d array of floats - size [739]

```
  nx = 4112
  ny =709
  nlines = 739
  skylinemask = intarr(nx,ny)  ; output 2d array of 1's and 0's

  for j = 0, nlines-1 do begin
    index = where( (wave gt skywave[j]-3) and (wave lt skywave[j]+3), nindex)
    if (nindex gt 0) then skylinemask[index] = 1
  endfor
```

I've started tackling this with value_locate but I got stuck.

```
waved = wave[*]
uniwave = sort(waved)
minskywave = skywave - 3
maxskywave = skywave + 3

v1 = value_locate(minskywave, waved[uniwave])
v2 = value_locate(maxskywave, waved[uniwave])
```

Any ideas on how to finish this? Or a simpler way than what I'm attempting. Thanks.

Subject: Re: set all elements in 2d array between some range to 1
Posted by David Fanning on Fri, 22 May 2015 21:22:33 GMT
View Forum Message <> Reply to Message

Brian Cherinka writes:

>
> So I'm trying to set all elements of a 2d-array that are between some padding, based off elements in another vector, to 1. Creating a mask of 1's and 0's.
>
> I want to turn this bit of code, which runs in 30 seconds, into a non-loop bit of code that runs faster.

```
>
>  wave = 2d array of floats - size [4112,709]
>  skywave = 1d array of floats - size [739]
>
>    nx = 4112
>    ny =709
>    nlines = 739
>    skylinemask = intarr(nx,ny)  ; output 2d array of 1's and 0's
>
>    for j = 0, nlines-1 do begin
>      index = where( (wave gt skywave[j]-3) and (wave lt skywave[j]+3), nindex)
>      if (nindex gt 0) then skylinemask[index] = 1
>    endfor
>
>  I've started tackling this with value_locate but I got stuck.
>
>  waved = wave[*]
>  uniwave = sort(waved)
>  minskywave = skywave - 3
>  maxskywave = skywave + 3
>
>  v1 = value_locate(minskywave, waved[uniwave])
>  v2 = value_locate(maxskywave, waved[uniwave])
>
>  Any ideas on how to finish this?  Or a simpler way than what I'm attempting.  Thanks.
```

Two ideas:

   http://www.idlcoyote.com/code_tips/valuelocate.html
   http://www.idlcoyote.com/code_tips/partition.html

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: set all elements in 2d array between some range to 1
Posted by havok2063 on Fri, 22 May 2015 21:58:20 GMT

View Forum Message <> Reply to Message

My attempted solution was at the bottom. I love value_locate, and I've started tackling this with that but I got stuck. My problem seems a bit more complicated than the uses described on your page.

```
waved = wave[*]
uniwave = sort(waved)
minskywave = skywave - 3
maxskywave = skywave + 3

v1 = value_locate(minskywave, waved)
v2 = value_locate(maxskywave, waved)
```

This gives me the positions in minskywave and maxskywave where the elements in waved lie, but it's not the final answer. I can't combine my "values" array into one big array, because my skywave ranges overlap, and that will give me incorrect binning.

For each element in skywave, I need to find where waved is between skywave+-3, and set those indices to 1. The rest should be set to 0.

Any ideas on how to finish this? Or a simpler way than what I'm attempting. Thanks.

>
> Two ideas:
>
>   http://www.idlcoyote.com/code_tips/valuelocate.html
>   http://www.idlcoyote.com/code_tips/partition.html
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
> Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: set all elements in 2d array between some range to 1
Posted by David Fanning on Fri, 22 May 2015 23:25:39 GMT

View Forum Message <> Reply to Message

Brian Cherinka writes:

>
> My attempted solution was at the bottom. I love value_locate, and I've started tackling this with that but I got stuck. My problem seems a bit more complicated than the uses described on your page.
>

> waved = wave[*]
> uniwave = sort(waved)
> minskywave = skywave - 3
> maxskywave = skywave + 3
>
> v1 = value_locate(minskywave, waved)
> v2 = value_locate(maxskywave, waved)
>
> This gives me the positions in minskywave and maxskywave where the elements in waved lie, but it's not the final answer. I can't combine my "values" array into one big array, because my skywave ranges overlap, and that will give me incorrect binning.
>
> For each element in skywave, I need to find where waved is between skywave+-3, and set those indices to 1. The rest should be set to 0.
>
> Any ideas on how to finish this? Or a simpler way than what I'm attempting. Thanks.

Here is how I would do this for one value in skywave:

```
   skywave = Randomu(-3L, 20)*20-10
   skywaveValue= skywave[1]
   cuts = [skywaveValue-3., skywaveValue+3]
   sortedIndices = Sort(skywave)
   sortedData = skywave[sortedIndices]
   indices = Value_Locate(cuts, sortedData)
   h = cgHistogram(indices, MIN=-1, REVERSE_INDICES=ri)
   oneIndices = cgReverseIndices(ri, 1)
   ones = skywave*0
   solnIndices = (sortedIndices)[oneIndices]
   ones[solnIndices] = 1
   Print, ""
   Print, 'Original Data:'
   Print, skywave
   Print, ""
   Print, 'Find data between ', String(cuts[0], Format='(F0.2)'), $
      ' and ', String(cuts[1], Format='(F0.2)')
   Print, ""
   Print, 'Data Between Endpoints: '
   Print, skywave[solnIndices]
   Print, ""
   Print, 'Array of 0s and 1s'
   Print, ones
   END
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: set all elements in 2d array between some range to 1
Posted by Jeremy Bailin on Wed, 27 May 2015 03:46:37 GMT

View Forum Message <> Reply to Message

On Friday, May 22, 2015 at 4:14:45 PM UTC-5, Brian Cherinka wrote:
> So I'm trying to set all elements of a 2d-array that are between some padding, based off elements in another vector, to 1. Creating a mask of 1's and 0's.
>
> I want to turn this bit of code, which runs in 30 seconds, into a non-loop bit of code that runs faster.
>
```
> wave = 2d array of floats - size [4112,709]
> skywave = 1d array of floats - size [739]
>
>   nx = 4112
>   ny =709
>   nlines = 739
>   skylinemask = intarr(nx,ny)  ; output 2d array of 1's and 0's
>
>   for j = 0, nlines-1 do begin
>     index = where( (wave gt skywave[j]-3) and (wave lt skywave[j]+3), nindex)
>     if (nindex gt 0) then skylinemask[index] = 1
>   endfor
>
```
> I've started tackling this with value_locate but I got stuck.
>
```
> waved = wave[*]
> uniwave = sort(waved)
> minskywave = skywave - 3
> maxskywave = skywave + 3
>
> v1 = value_locate(minskywave, waved[uniwave])
> v2 = value_locate(maxskywave, waved[uniwave])
>
```
> Any ideas on how to finish this? Or a simpler way than what I'm attempting. Thanks.

Did someone say value_locate? ;-)

I have two solutions to this. The obvious IDL Way is easy but requires many GB of memory for arrays of this size because it requires building several NX x NY x NLINES arrays, each of which has over 4 billion elements:

```
skywave_sorted = skywave[sort(skywave)]
minskywave = skywave_sorted - 3
maxskywave = skywave_sorted + 3

; memory-intensive version
minmask = rebin(wave, nx, ny, nlines, /sample) gt rebin(reform(minskywave,1,1,nlines), nx, ny,
nlines)
maxmask = rebin(wave, nx, ny, nlines, /sample) lt rebin(reform(maxskywave,1,1,nlines), nx, ny,
nlines)
skylinemask_v1 = total(minmask * maxmask, 3) ne 0
```

The value_locate way is actually ridiculously easy once you merge the overlapping sky regions. Here is how I would first do that:

```
; 1. merge overlapping skyline regions
;   First figure out where the overlaps are, and label them uniquely
non_overlap = [1, minskywave[1:*] gt maxskywave]   ; this is 0 if it overlaps with previous one, 1 if
it's
 ; a new skyline region
skyregionlabel = total(non_overlap, /cumulative, /int)  ; a unique integer for each skyline region
skyreghist = histogram(skyregionlabel, min=1, omax=nlineregions, reverse_indices=skyri)
;   Second, create new non-overlapping arrays. In this new region array, the even indices
;   indicate the beginning of a sky region and the odd indices indicate the end of a sky region.
;   For example, the first region goes from skyregion_bounds[0] to skyregion_bounds[1].
skyregion_bounds = fltarr(2L * nlineregions)
for i=0l, nlineregions-1 do begin
   skyregion_bounds[i*2] = minskywave[skyri[skyri[i]]]
   skyregion_bounds[i*2+1] = maxskywave[skyri[skyri[i+1]-1]]
endfor
```

...and then the reason for sticking both the minimum and maximum bounds into a single array becomes clear once you use the value_locate magic to do the real work:

```
; now use value_locate to see where wave lies with respect to these boundaries
wave_regions = value_locate(skyregion_bounds, wave)
; now if wave_regions is even then it is within a sky region, and if it is odd then it is between
regions
skylinemask_v2 = (wave_regions+1) mod 2
```

-Jeremy.

Subject: Re: set all elements in 2d array between some range to 1
Posted by Jeremy Bailin on Wed, 27 May 2015 04:05:02 GMT
View Forum Message <> Reply to Message

On Tuesday, May 26, 2015 at 10:46:39 PM UTC-5, Jeremy Bailin wrote:
> On Friday, May 22, 2015 at 4:14:45 PM UTC-5, Brian Cherinka wrote:
>> So I'm trying to set all elements of a 2d-array that are between some padding, based off elements in another vector, to 1. Creating a mask of 1's and 0's.
>>
>> I want to turn this bit of code, which runs in 30 seconds, into a non-loop bit of code that runs faster.
>>
>> wave = 2d array of floats - size [4112,709]
>> skywave = 1d array of floats - size [739]
>>
>>   nx = 4112
>>   ny =709
>>   nlines = 739
>>   skylinemask = intarr(nx,ny) ; output 2d array of 1's and 0's
>>
>>   for j = 0, nlines-1 do begin
>>   index = where((wave gt skywave[j]-3) and (wave lt skywave[j]+3), nindex)
>>   if (nindex gt 0) then skylinemask[index] = 1
>>   endfor
>>
>> I've started tackling this with value_locate but I got stuck.
>>
>> waved = wave[*]
>> uniwave = sort(waved)
>> minskywave = skywave - 3
>> maxskywave = skywave + 3
>>
>> v1 = value_locate(minskywave, waved[uniwave])
>> v2 = value_locate(maxskywave, waved[uniwave])
>>
>> Any ideas on how to finish this? Or a simpler way than what I'm attempting. Thanks.
>
> Did someone say value_locate? ;-)
>
>
> I have two solutions to this. The obvious IDL Way is easy but requires many GB of memory for arrays of this size because it requires building several NX x NY x NLINES arrays, each of which has over 4 billion elements:
>
>
> skywave_sorted = skywave[sort(skywave)]
> minskywave = skywave_sorted - 3
> maxskywave = skywave_sorted + 3
>

> ; memory-intensive version
> minmask = rebin(wave, nx, ny, nlines, /sample) gt rebin(reform(minskywave,1,1,nlines), nx, ny, nlines)
> maxmask = rebin(wave, nx, ny, nlines, /sample) lt rebin(reform(maxskywave,1,1,nlines), nx, ny, nlines)
> skylinemask_v1 = total(minmask * maxmask, 3) ne 0
>
>
>
> The value_locate way is actually ridiculously easy once you merge the overlapping sky regions. Here is how I would first do that:
>
> ; 1. merge overlapping skyline regions
> ;   First figure out where the overlaps are, and label them uniquely
> non_overlap = [1, minskywave[1:*] gt maxskywave]   ; this is 0 if it overlaps with previous one, 1 if it's
>   ; a new skyline region
> skyregionlabel = total(non_overlap, /cumulative, /int) ; a unique integer for each skyline region
> skyreghist = histogram(skyregionlabel, min=1, omax=nlineregions, reverse_indices=skyri)
> ;   Second, create new non-overlapping arrays. In this new region array, the even indices
> ;   indicate the beginning of a sky region and the odd indices indicate the end of a sky region.
> ;   For example, the first region goes from skyregion_bounds[0] to skyregion_bounds[1].
> skyregion_bounds = fltarr(2L * nlineregions)
> for i=0l, nlineregions-1 do begin
>   skyregion_bounds[i*2] = minskywave[skyri[skyri[i]]]
>   skyregion_bounds[i*2+1] = maxskywave[skyri[skyri[i+1]-1]]
> endfor
>
>
> ...and then the reason for sticking both the minimum and maximum bounds into a single array becomes clear once you use the value_locate magic to do the real work:
>
> ; now use value_locate to see where wave lies with respect to these boundaries
> wave_regions = value_locate(skyregion_bounds, wave)
> ; now if wave_regions is even then it is within a sky region, and if it is odd then it is between regions
> skylinemask_v2 = (wave_regions+1) mod 2
>
>
> -Jeremy.

I just did some timing tests. I get your original for loop code running in 2.9 seconds (I'm not sure why you're finding that it takes 30 -- are you sure that's just this piece of code? Or is it embedded in an outer loop that runs 10-ish times?), and the value_locate version runs in 0.14 seconds, so 20 times faster.

-Jeremy.

Subject: Re: set all elements in 2d array between some range to 1
Posted by Jeremy Bailin on Wed, 27 May 2015 04:12:58 GMT
View Forum Message <> Reply to Message

On Tuesday, May 26, 2015 at 10:46:39 PM UTC-5, Jeremy Bailin wrote:
> On Friday, May 22, 2015 at 4:14:45 PM UTC-5, Brian Cherinka wrote:
>> So I'm trying to set all elements of a 2d-array that are between some padding, based off elements in another vector, to 1. Creating a mask of 1's and 0's.
>>
>> I want to turn this bit of code, which runs in 30 seconds, into a non-loop bit of code that runs faster.
>>

```
>>  wave = 2d array of floats - size [4112,709]
>>  skywave = 1d array of floats - size [739]
>>
>>    nx = 4112
>>    ny =709
>>    nlines = 739
>>    skylinemask = intarr(nx,ny)  ; output 2d array of 1's and 0's
>>
>>    for j = 0, nlines-1 do begin
>>      index = where( (wave gt skywave[j]-3) and (wave lt skywave[j]+3), nindex)
>>      if (nindex gt 0) then skylinemask[index] = 1
>>    endfor
>>
>>  I've started tackling this with value_locate but I got stuck.
>>
>>  waved = wave[*]
>>  uniwave = sort(waved)
>>  minskywave = skywave - 3
>>  maxskywave = skywave + 3
>>
>>  v1 = value_locate(minskywave, waved[uniwave])
>>  v2 = value_locate(maxskywave, waved[uniwave])
>>
```

>> Any ideas on how to finish this? Or a simpler way than what I'm attempting. Thanks.
>
> Did someone say value_locate? ;-)
>
>
> I have two solutions to this. The obvious IDL Way is easy but requires many GB of memory for arrays of this size because it requires building several NX x NY x NLINES arrays, each of which has over 4 billion elements:
>
>

```
> skywave_sorted = skywave[sort(skywave)]
> minskywave = skywave_sorted - 3
> maxskywave = skywave_sorted + 3
>
```

> ; memory-intensive version
> minmask = rebin(wave, nx, ny, nlines, /sample) gt rebin(reform(minskywave,1,1,nlines), nx, ny, nlines)
> maxmask = rebin(wave, nx, ny, nlines, /sample) lt rebin(reform(maxskywave,1,1,nlines), nx, ny, nlines)
> skylinemask_v1 = total(minmask * maxmask, 3) ne 0
>
>
>
> The value_locate way is actually ridiculously easy once you merge the overlapping sky regions. Here is how I would first do that:
>
> ; 1. merge overlapping skyline regions
> ;   First figure out where the overlaps are, and label them uniquely
> non_overlap = [1, minskywave[1:*] gt maxskywave]   ; this is 0 if it overlaps with previous one, 1 if it's
>   ; a new skyline region
> skyregionlabel = total(non_overlap, /cumulative, /int) ; a unique integer for each skyline region
> skyreghist = histogram(skyregionlabel, min=1, omax=nlineregions, reverse_indices=skyri)
> ;   Second, create new non-overlapping arrays. In this new region array, the even indices
> ;   indicate the beginning of a sky region and the odd indices indicate the end of a sky region.
> ;   For example, the first region goes from skyregion_bounds[0] to skyregion_bounds[1].
> skyregion_bounds = fltarr(2L * nlineregions)
> for i=0l, nlineregions-1 do begin
>   skyregion_bounds[i*2] = minskywave[skyri[skyri[i]]]
>   skyregion_bounds[i*2+1] = maxskywave[skyri[skyri[i+1]-1]]
> endfor
>
>
> ...and then the reason for sticking both the minimum and maximum bounds into a single array becomes clear once you use the value_locate magic to do the real work:
>
> ; now use value_locate to see where wave lies with respect to these boundaries
> wave_regions = value_locate(skyregion_bounds, wave)
> ; now if wave_regions is even then it is within a sky region, and if it is odd then it is between regions
> skylinemask_v2 = (wave_regions+1) mod 2
>
>
> -Jeremy.

...and you can actually completely get rid of the remaining for loop, now that I think about it. Replace the for loop with these two lines (not that it really matters, but nature abhors a for loop):

```
skyregion_bounds[0:*:2] = minskywave[skyri[skyri[0:nlineregions-1]]]
skyregion_bounds[1:*:2] = maxskywave[skyri[skyri[1:nlineregions]-1]]
```

-Jeremy.

Subject: Re: set all elements in 2d array between some range to 1
Posted by havok2063 on Wed, 03 Jun 2015 03:37:57 GMT
View Forum Message <> Reply to Message

Hi Jeremy,

Thanks a lot for your help. This is great. Exactly the solution I'm looking for. Hmm..my original loop is taking me ~30 seconds. Not sure why yours is much faster. Here is my code. Actually it seems it's because the 10.^loglam is being done every loop iteration. Changing this to a stored variable, the loop takes 2.1 seconds.

```
starttime = systime(/seconds)
for j=0,nlines-1 do begin
   index=where((10.^loglam gt skylines[j].skywave-skypad)and(10.^loglam lt
skylines[j].skywave+skypad),nindex)
  f (nindex gt 0) then skylinemask[index]=1
endfor
print, (systime(/seconds)-starttime)
       30.812672
```

I really like your code, since it's faster and no loops. However, when I run it, I'm getting some small differences between the output from the original vs the new method.

```
IDL> help, where(skylinemask)
<Expression>    LONG      = Array[226935]

IDL> help, where(skylinemask_v2)
<Expression>    LONG      = Array[226933]

IDL> help, where(skylinemask ne skylinemask_v2)
<Expression>    LONG      = Array[14]
```

Printing these 14 elements for both the old and new mask shows that are flipped from each other. Could it be a boundary issue?

Cheers, Brian

Subject: Re: set all elements in 2d array between some range to 1
Posted by Jeremy Bailin on Wed, 03 Jun 2015 19:31:33 GMT
View Forum Message <> Reply to Message

> Thanks a lot for your help. This is great. Exactly the solution I'm looking for. Hmm..my original loop is taking me ~30 seconds. Not sure why yours is much faster. Here is my code. Actually it seems it's because the 10.^loglam is being done every loop iteration. Changing this to a stored variable, the loop takes 2.1 seconds.

Ah, that definitely makes sense! Yes, avoid doing redundant calculations in loops. :) This is always Good Advice... it looks like you got 15x faster just by doing that, which is almost as much as the speedup of going to the more sophisticated algorithm!

> I really like your code, since it's faster and no loops. However, when I run it, I'm getting some small differences between the output from the original vs the new method.
>
> IDL> help, where(skylinemask)
> <Expression>   LONG   = Array[226935]
>
> IDL> help, where(skylinemask_v2)
> <Expression>   LONG   = Array[226933]
>
> IDL> help, where(skylinemask ne skylinemask_v2)
> <Expression>   LONG   = Array[14]
>
> Printing these 14 elements for both the old and new mask shows that are flipped from each other. Could it be a boundary issue?

Quite possibly. The Value_Locate code will mask when wave lies exactly at a lower boundary (but not at an upper boundary), while your code doesn't mask at either boundary. So I'd suggest checking what the exact wavelengths of those 14 discrepancies are -- I suspect you'll find that they're exactly at the lower boundary of one of the mask regions.

-Jeremy.

Subject: Re: set all elements in 2d array between some range to 1
Posted by havok2063 on Mon, 08 Jun 2015 22:03:31 GMT

View Forum Message <> Reply to Message

It looks like that's exactly what is happening. Some are matched to the lower boundary, and some are actually matched to the upper boundary. And it seems like a precision issue. My wavelengths are to 4 sigfigs, but my skyline mask values are to 2 sigfigs, e.g. 3827.1299 vs 3827.13, so that kicks them to a different bin.

On Wednesday, June 3, 2015 at 3:31:36 PM UTC-4, Jeremy Bailin wrote:
>> Thanks a lot for your help. This is great. Exactly the solution I'm looking for. Hmm..my original loop is taking me ~30 seconds. Not sure why yours is much faster. Here is my code. Actually it seems it's because the 10.^loglam is being done every loop iteration. Changing this to a stored variable, the loop takes 2.1 seconds.
>
> Ah, that definitely makes sense! Yes, avoid doing redundant calculations in loops. :) This is always Good Advice... it looks like you got 15x faster just by doing that, which is almost as much as the speedup of going to the more sophisticated algorithm!
>
>> I really like your code, since it's faster and no loops. However, when I run it, I'm getting some small differences between the output from the original vs the new method.
>>

>> IDL> help, where(skylinemask)
>> <Expression>   LONG   = Array[226935]
>>
>> IDL> help, where(skylinemask_v2)
>> <Expression>   LONG   = Array[226933]
>>
>> IDL> help, where(skylinemask ne skylinemask_v2)
>> <Expression>   LONG   = Array[14]
>>
>> Printing these 14 elements for both the old and new mask shows that are flipped from each other. Could it be a boundary issue?
>
> Quite possibly. The Value_Locate code will mask when wave lies exactly at a lower boundary (but not at an upper boundary), while your code doesn't mask at either boundary. So I'd suggest checking what the exact wavelengths of those 14 discrepancies are -- I suspect you'll find that they're exactly at the lower boundary of one of the mask regions.
>
> -Jeremy.