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.