Subject: Re: how to get the index of the maximum element in each row for a 2D matrix efficiently
Posted by JD Smith on Tue, 02 Sep 2003 17:12:42 GMT
View Forum Message <> Reply to Message

On Sat, 30 Aug 2003 13:33:46 -0700, Henry Roe wrote:

>> Starting with v5.6, MAX will do this for you with its own DIMENSION
>> keyword. Remember that in IDL dimensions start, perversely, with 1, as
>> opposed to everything else, which is 0 based. As of now we can thread
>> the following:
>>
>> TOTAL
>> PRODUCT
>> MIN
>> MAX
>> MEDIAN
>>
>> The only remaining operators I wish could be dimensionally threaded in
>> this way are AND, OR, and especially the new && and || short-circuiting
>> ops.
>>
>> JD
>
> Standard deviation or variance is also a useful function; e.g. if you
> have a stack of images and want the stdev in each pixel. (I wrote a
> bunch of call_external fortran routines to do all these before v5.6; I
> think stdev_in_1d is the only one that has not been made redundant.)

Luckily, the variance can be computed directly from its definition as
the mean square deviation, using only TOTAL, and a little REBIN magic,
e.g.:

```
IDL> a=randomu(sd,10,10,10)
IDL>  var=total((a-rebin(total(a,3)/10.,10,10,10,/SAMPLE))^2,3)/(1 0.-1)
IDL> print,variance(a[0,0,*])
    0.190512
IDL> print,var[0,0]
    0.190512
```

To make this general for any array size and any summed dimension `d',
you need a way to expand an arbitrary array along an arbitrary
dimension (the selfsame one over which it was just collapsed). REBIN
and REFORM together can do this for you:

```
s1=(s2=size(a,/dimensions)) & s2[d-1]=1L
 var=total((a-rebin(reform(total(a,d)/s1[d-1],s2),s1,/SAMPLE) )^2,d)/(s1[d-1]-1.)
```

The key ingredient above is the formidable looking:

rebin(reform(total(a,d)/s1[d-1],s2),s1,/SAMPLE)

All this is doing is taking the mean of an array over a given dimension, and then inflating this result to match the original array dimensions. Notice the use of vector inputs to REBIN and REFORM (s2 & s1), which makes treating general arrays/dimensions trivial. This same approach could be used for any such generic collapse/inflate operation, e.g.:

s1=(s2=size(a,/dimensions)) & s2[d-1]=1L
median_inflated=rebin(reform(median(a,DIMENSION=d),s2),s1,/S AMPLE)

JD
