Subject: Re: how to get the index of the maximum element in each row for a 2D matrix efficiently
Posted by JD Smith on Fri, 29 Aug 2003 19:05:37 GMT
View Forum Message <> Reply to Message

On Fri, 29 Aug 2003 10:34:52 -0700, Xiaoying Jin wrote:

> Hi, there,
>
> Suppose we have a 2D dimension matrix A(i,j), I want to get the index of
> the max element in each row.
>
> Basicly, we can do a for loop on each row, then use 'max' function to
> get the index for each row. But for a matrix having many rows, that will
> be pretty slow-->not efficient. Is there a function that can return what
> I want if I input a 2D matrix.
>
> In the previous version, 'total' function can only get the total values
> of the whole matrix. Now it can get the total values on each dimension
> by setting the keyword 'dimension'.
>
> So how is 'max' working?

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

Subject: Re: how to get the index of the maximum element in each row for a 2D matrix efficiently
Posted by xje4e on Fri, 29 Aug 2003 22:23:10 GMT
View Forum Message <> Reply to Message

> 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

Oh, Yeah! Finally they did it, although it has been implemented in
matlab for a long time. And I found it in IDL 5.5 in ENVI3.5 too.

Cheers,

Jin

Subject: Re: how to get the index of the maximum element in each row for a 2D matrix efficiently
Posted by the_cacc on Sat, 30 Aug 2003 00:33:50 GMT
View Forum Message <> Reply to Message

I just *know* histogram is going to be useful here... gurus, please step up :)

Subject: Re: how to get the index of the maximum element in each row for a 2D matrix efficiently
Posted by henrygroe on Sat, 30 Aug 2003 20:33:46 GMT
View Forum Message <> Reply to Message

>
> 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.)

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