Subject: Re: Moving Average on Hyperspectral dataset
Posted by JD Smith on Tue, 27 Mar 2007 00:38:38 GMT
View Forum Message <> Reply to Message

On Mon, 26 Mar 2007 16:03:28 -0700, rafaloos wrote:

> Hi, all ...
> We are trying to do a moving average on our dataset. The dataset comes
> from a Hyperspectral sensor and has 296 samples, 2000 lines and 492 bands.
> We need to go through each pixel and do a moving average on the spectral
> bands. So in the end we will have a dataset with the same dimensions as
> the original but with smooth spectra. And because of the moving average
> kernel size, we will have some empty bands at the beginning and at the
> end.
>
>
> Here is the code that we are using now ...
>
>>>> > IMAGE = (296, 2000, 492)

image=smooth(image,[1,1,width])

> With the loops the code takes about 3 hours ... Is there a way to speed it
> up ?

If that 1.2GB (*2) array is pushing your memory limits, consider doing
it in "chunks", e.g. 50 samples at a time.

JD

Subject: Re: Moving Average on Hyperspectral dataset
Posted by David Fanning on Tue, 27 Mar 2007 02:14:24 GMT
View Forum Message <> Reply to Message

JD Smith writes:

>
> image=smooth(image,[1,1,width])
>
>> With the loops the code takes about 3 hours ... Is there a way to speed it
>> up ?
>
> If that 1.2GB (*2) array is pushing your memory limits, consider doing
> it in "chunks", e.g. 50 samples at a time.

I thought it might go faster if you moved the dimension
you are smoothing into contiguous memory first:

image = Transpose(image, [2,0,1])
image = Smooth(image, [width, 1, 1])

But with an image(100,200,300), it took 0.281 seconds
with and without the TRANSPOSE. Is the transposition
really negligibly fast?

I note that the LOOP method took nearly 76 seconds!! :-)

Cheers,

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

Subject: Re: Moving Average on Hyperspectral dataset
Posted by JD Smith on Tue, 27 Mar 2007 17:08:25 GMT
View Forum Message <> Reply to Message

On Mon, 26 Mar 2007 19:14:24 -0700, David Fanning wrote:

> JD Smith writes:
>
>
>> image=smooth(image,[1,1,width])
>>
>>> With the loops the code takes about 3 hours ... Is there a way to
>>> speed it up ?
>>
>> If that 1.2GB (*2) array is pushing your memory limits, consider doing
>> it in "chunks", e.g. 50 samples at a time.
>
> I thought it might go faster if you moved the dimension you are smoothing
> into contiguous memory first:
>
> image = Transpose(image, [2,0,1])
> image = Smooth(image, [width, 1, 1])
>
> But with an image(100,200,300), it took 0.281 seconds with and without the
> TRANSPOSE. Is the transposition really negligibly fast?

This might be true, depending on the kernel size. The problem is,
TRANSPOSE will require reading through and re-writing the entire block

of memory, using a number of out of order memory operations similar to what SMOOTH would use. I think the added overhead of TRANSPOSE just canceled out the savings (if any) of in order execution.

Of course, if you have a way to arrange your data so that the contiguous memory area is in order to begin with, that might help. Interestingly enough, however, I find that even this doesn't improve speed for me... i.e. smooth(image,[1,1,width]) is faster that smooth(image,[width,1,1]). The only explanation is that SMOOTH doesn't optimize itself when averaging over contiguous elements.

Another interesting finding, which sheds some light on this, is that SMOOTH's cost is almost independent of the smoothing kernel width, which might seem remarkable, until you consider that it probably works in a rolling sense, by accumulating an additional point, and subtracting the last one off of the sum. This insight probably explains the memory performance as well: by its design, SMOOTH is fetching noncontiguous pieces of memory to perform the rolling sum, independent of which dimension(s) it's smoothing.

JD
