Subject: Re: fast svdc for Singular Value Decomposition?
Posted by on Mon, 03 Dec 2012 20:40:11 GMT
View Forum Message <> Reply to Message

Den måndagen den 3:e december 2012 kl. 18:16:30 UTC+1 skrev ivitseva:
> Thanks, I've tried la_svd as well but I had the same problem, indeed it isn't faster.
>
>
>
> To be honest I do not understand what you mean under "calculating svd of teh blocks and then combine the results".
>
>
>
> Could you please be more specicif?

Well, it only applies if your matrix is block diagonal. If it's not, then forget it.

So, assuming your matrix A is indeed block diagonal with blocks A1, A2, ... AN along the diagonal. Consider the SVD equation A = U W VT and write it in block diagonal form. Make U and V block diagonal with blocks the same size as those of A and numbered the same way. Also split the diagonal W matrix the same way. Then you'll notice that due to the off-diagonal zeros the SVD equation holds for each block index separately, i.e., Ai = Ui Wi ViT. So you can calculate the SVD of the Ai separately and put together the U, W, and V matrices from the results, which is faster than SVD of the entire A in one go due to the non-linearity of the problem.

Note that the singular values in W will not come out sorted in order of significance, so there is one additional step of book keeping where you sort W and reorder the columns of U and V appropriately.

>
>
>
> Cheers,
>
> Eva
>
>
>
> On Monday, December 3, 2012 2:35:03 PM UTC+1, Mats Löfdahl wrote:
>
>> Den måndagen den 3:e december 2012 kl. 09:41:18 UTC+1 skrev ivitseva:
>
>>
>
>>> Dear All,
>
>>

>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I'm running a Singular Value Decomposition in IDL using the svdc routine. The goal is to perform an extended EOF analysis (or extended PCA).
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> My input is a covariance matrix built from two time series images, each with 348 bands and with a spatial dimension of ns=360 columns * nl=180 rows.
>
>>
>
>>>
>
>>
>
>>> The covariance matrix becomes an [space,space] matrix (i.e. a dimension of [(ns*nl),(ns*nl)] that is too big, my script did not finish after 3 days so I've terminated the run.
>
>>
>
>>>
>
>>
>

>>>
>
>>
>
>>>
>
>>
>
>>> Is there maybe a fast way to perform this decomposition?
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I guess there should be a mathematical workaround for Singular Value Decomposition of this large matrix but I must confess here I'm reaching my limits. I would be very grateful for an answer, our research is at halt at the moment because of this problem.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> It would be great if somebody could tell me how to improve the run time?
>
>>
>
>>>
>

>>
>
>>> Basically I read the two time series, then form a [nb,ns*nl] two dimensional array from both time series, make a subset of the two arrays ignoring NaN, and then form the covariance matrix as cov=(1/nb-1)*(array1##transpose(array2)). This becomes a very large matrix and then svdc,cov,W,U,V is almost impossible to compute.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> Thank you very much in advance,
>
>>
>
>>>
>
>>
>
>>> Eva
>
>>
>
>>
>
>>
>

>> la_svd is a better routine (see other recent thread) but I don't think it's faster by orders of magnitude. If you need that, generally your matrix has to have some special properties that you can exploit. For example, if it is block diagonal you should be able to calculate the svd of the blocks and then combine the results into the svd of the whole matrix.