Subject: fast svdc for Singular Value Decomposition?
Posted by ivitseva on Mon, 03 Dec 2012 08:41:18 GMT
View Forum Message <> Reply to Message

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

Subject: Re: fast svdc for Singular Value Decomposition?
Posted by Craig Markwardt on Mon, 03 Dec 2012 18:49:34 GMT
View Forum Message <> Reply to Message

On Monday, December 3, 2012 3:41:18 AM UTC-5, ivitseva wrote:
> 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, ...
>
...
> 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.

You say your matrix is array(ns*nl,ns*nl), which is array(64800,64800), which I think is nearly impossible. If what you say is true, then your matrix occupies 4 billion elements and 134 GB of storage space if you are using floating point arithmetic (or 268 GB for double precision). If that really is the case, then you have entered the realm of massive matrix algebra, and IDL is way underpowered. Give up now and find a massively parallel supercomputer, and a numerical analyst to help you.

In the hopes that you didn't define the problem correctly, and your problem really is smaller than 64800x64800, I have some other comments.

Singular value decomposition typically takes about 10*N^3 operations where N is the dimension of your square matrix. (See the classic Golub & van Loan textbook.)

If you really have a positive definite symmetric covariance matrix, then Cholesky factorization will be faster, more like N^3/3 operations, which is potentially a factor of 30 faster than SVD. [This does depend on how efficiently the algorithm is implemented.] Cholesky is faster because it exploits the symmetry of the system, whereas SVD does not. On the other hand, Cholesky is less capable of dealing with nearly-singular matrices than SVD.

The real killer for either technique is still the N^3 term. Which is to say, every time you double the size of your problem, it will take eight times longer to process. That is why I say you should give up if N really is 64800.

If your covariance matrix is banded - only has non-zero terms within a few cells of the diagonal - then the algorithm can be be made *much* faster. If you have non-zero terms far off-diagonal, then you have no choice but to use the full factorization. But, it's worth seeing if you can reorganize rows & columns to make the pattern more banded.

Craig