Subject: Re: Generalized Least Squares? (LONG POST!)
Posted by Gernot Hassenpflug on Tue, 20 May 2008 09:00:34 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@REMOVEcow.physics.wisc.edu> writes:

> Gernot Hassenpflug <gernot@nict.go.jp> writes:
>> Dear all,
>>
>> I'm involved in ongoing research on a problem that I solved to a first
>> approximation with weighted least squares (using MPFIT) but for a real
>> solution I require generalized least squares: WLS uses a diagonal
>> covariance matrix, i.e., the data errors are uncorrelated; GLS uses a
>> full covariance matrix, i.e., the data errors can be correlated.
>>
>> I have not found any ready solution in IDL yet, and I am under the
>> impression that there is no analytical solution to GLS, so fairly
>> complicated numerical methods are required.
>>
>> I have actually found a routine in MATLAB called "lscov" which can
>> solve this problem, and wonder whether there is a chance to hobble
>> something together in IDL? I'd be happy to try and modify MPFIT to be
>> able to deal with GLS too, could anyone comment on possibilities?
>
> Greetings Gernot--

Hello Craig,

Yoiur post is much appreciated. I've been reading up on fitting,
correlated errors and matrix computations for this, so I'll indicate
below what I understand, and what I maybe missed before.

> Solving the generalized least squares problem is reasonably
> straightforward. Basically it involves transforming the original
> (correlated) variables to new (uncorrelated) variables. Ironically I

I did not know that the transformation goes to uncorrelated
variables. From my reading, it looked as though the objective function
S, which is Chi-squared for the case of uncorrelated errors (diagonal
matrix), becomes a full matrix: S= trans(r) inv(V) r, where V is the
covariance matrix, and r is the residual vector.

Edit: I see you give this explanation below.

Then, it seems to me though rather hazy, that the assumptions about
unbiased estimator may not hold anymore for GLS.

> The least-squares problem can be expressed as the following normal

> equation, in matrix notation,
>
> A^T A x = A^T v

OK.

> where A is the pattern matrix (= "Jacobian"), v is the vector of
> normalized residuals, and x is the vector of parameters (and "^T"
> indicates the matrix transpose). MPFIT solves this equation, given an
> initial estimate of x, to get a new estimate of x.

OK.

> I say that v is the vector of normalized residuals, because typically
> we compute it something like this,
>
> v = (DATA - MODEL)/ERROR

OK.

> This explicit definition shows that we were assuming *uncorrelated*
> errors, i.e. for each vector element, there is a single well defined
> uncertainty which does not depend on neighboring elements. The
> chi-squared value is defined as,
>
> CHI2 = v^T v

OK.

> However, in the case of correlated errors, the chi-square value is
> defined as,
>
> CHI2 = v^T (COVAR^(-1)) v

OK.

> where COVAR is the covariance matrix and "^(-1)" indicates the matrix
> inverse. In this case, v is no longer the normalized residuals, but
> just the raw residuals. The units are correct since COVAR has units
> of v^2, so (COVAR^(-1)) has units of v^(-2). The normal equation
> becomes,
>
> A^T (COVAR^(-1)) A x = A^T (COVAR^(-1)) v
>
> Unfortunately, MPFIT does *not* solve this problem. Are we out of
> luck? No, actually it's still a reasonably straightforward problem to
> solve.

Ah, the plot thickens!

> For example, consider if we can factor the COVAR matrix like this,
>
> COVAR = L L^T

OK, this is where it got hazy for me: general optimization solutions to systems of linear equations. I much appreciate your solution description below, as I am sure that I could not have reliably duplicated that with assurance that I was doing the right thing. You describe two methods below, and the SVD one seems to be the one that lscov uses in MATLAB.

> where L is lower-triangular. This is the well-known Cholesky
> factorization. As long as COVAR is positive-definite /../

Yes, I had the same problem as you in previous work with 95-channel radar interferometric imaging: large matrices, noise and statistical error, and boom, no more positive-deifnite...

> OK, this all may sound great. Unfortunately, in my particular
> application, I did not succeed. The problem was that my particular
> covariance matrix was not positive-definite. I had huge correlations
> between points, which caused the CHOLSOL stage to fail.
>
> There is theoretically a way around *this* problem as well. Instead
> of using the Cholesky factorization, one can use the SVD
> factorization, which is far more robust against singular matrices.
> The SVD factorization looks like this,
>
> COVAR = UMAT WMAT VMAT^T
>
> where UMAT, VMAT and WMAT are matrices with special properties. In
> IDL, the SVDC procedure computes this factorization.
>
> The benefit of this method is that the singular values are sorted by
> magnitude within the WMAT matrix. The first values are the strongest,
> and the last values are small, or zero. One can effectively "zero
> out" the insignificant singular values, which results in a more robust
> effective inverse, COVAR^(-1). This is described in more detail by
> Numerical Recipes. However, a more intuitive way to think about this
> is that if you start with N measurements, but M of the singular values
> are insignificant, then your data set really had N-M uncorrelated
> degrees of freedom to begin with (whereas M of the measurements were
> effectively totally dependent values).

Yes, understood I think: dependent given the effects of the existing noise and statistical error, right?

> Proceeding, one can compute the revised values,
>
> B = (WMAT^(-1/2)) VMAT^T A
> u = (WMAT^(-1/2)) VMAT^T v
>
> and then the problem is reduced again to the "uncorrelated" normal
> equations described earlier. Since WMAT is a diagonal matrix, all of
> the equations above are very easy to compute, and can be substituted
> into MYFUNCT_CORREL.

OK, I begin to see why it is "uncorrelated" error now.

> Again, for my problem, I implemented this method in the C language,
> but to be honest, the method did not improve the situation. I believe
> that my data was so correlated that even SVD was not appropriate./../

Many thanks Craig. My 2 cents worth:

- when the covariance matrix of the data is so poor, then
  pre-conditioning may be necessary, which will bias the results
  unavoidably.

- better methods of experiment may be necessary to get more malleable
  data.

- the Numerical Recipes algorithm is quick and dirty, and not usable
  on anything except well-behaved data; MATLAB forum posts imply that
  the lscov algorithm is very well structured and can deal with
  this. I will post my results here within a week I hope, and give my
  ideas for what kind of solutions might be possible (other than LU
  and SVD).

Best regards,
Gernot

--
BOFH excuse #98:

The vendor put the bug there.