BAYESMTH: A PROGRAM FOR MULTIVARIATE BAYESIAN INTERPOLATION

by

Christopher A. Sims

Discussion Paper No. 234, September, 1986

Center for Economic Research
Department of Economics
University of Minnesota
Minneapolis, Minn 55455

Research described herein was supported in part by NSF Grants SES 8309329 and SES 8608078 and by the IBM Woksape Grant to the University of Minnesota.

BAYESMTH: A PROGRAM FOR MULTIVARIATE BAYESIAN INTERPOLATION

This program uses a set of parameter values x and function values f, $(x_i, f_i\colon i=1,\ldots,n)$ to derive an interpolated function f^*. The interpolated function has the property that $f^*(x_i) = f(x_i)$, all i. The program has been used extensively to aid in maximizing functions for which computation of f(x) is expensive for each x and x is of high (4 or more) dimension. In such cases numerical differentiation is computationally expensive, the search for a maximum may not proceed to numerical convergence, and an estimate of the shape of the function around the maximum, including an estimate of the likely gain from further search, is important. Standard modified Newton-Raphson maximization routines are therefore inadequate, and BAYESMTH provides an alternative. The program's algorithm is likely to be useful in other contexts as well, as in computing contour diagrams for 3-d sections of multivariate functions from irregularly spaced data and in numerical integration over complex posteriors in Bayesian statistical analysis.

1. The Theory

We postulate that the function f is a random draw from a certain distribution of functions, in particular that f satisfies

$$f(x) - B = \int k[(x-y)'S]\, dW(y) \quad ,$$

where W is a zero-mean random measure satisfying

$$\mathrm{cov}(W(A),W(C)) = \int_{A\cap C} dy \quad .$$

In the univariate case, W is a martingale with stationary increments and f is a weighted average of white noise, with k the weighting function. The matrix S is a scaling constant, taken hereafter to be diagonal with $1/\tau_i$ as typical diagonal element. The scalar B is the mean of $f(x)$, called hereafter the "base value".

This framework allows us to compute directly

$$\mathrm{cov}(f(x),f(y)) = R(x,y) = \int k[(x-z)'S]k[(y-z)'S]\,dz \quad .$$

If the function k is chosen appropriately, this covariance emerges as an easily computed function of x and y.

We now pose the question: Given $\{x_i, f(x_i), i=1,\ldots,n\}$, what is $E[f(x)]$ for some arbitrarily given x? Under an assumption of joint normality of the f values, this is the same question as: What is the minimum variance linear predictor of $f(x)$ based on $\{x_i, f_i\}$? The question is easily answered. Let Ω be the matrix with typical element $R(x_i,x_j)$, i.e.

$$\Omega = \Big[\, R(x_i,x_j) \,\Big] \quad , \quad \text{and also}$$

$$\Phi = \Big[\, R(x_i,x) \,\Big] \quad ,$$

$$f_0 = \Big[\, f(x_i)-B \,\Big] \quad .$$

Then it is easily verified that

$$E[f(x)\,|\,\{x_i,f_i\}] = f_0'\Omega^{-1}\Phi \quad .$$

When n is large, inversion of Ω may be a substantial computation. When computation of $f(x)$ is very expensive, this may not matter. In any case, if considerable exploration of the interpolated f is required with a given $\{x_i\}$ sequence, the inversion of Ω need only be done once, then reused for various trial values of x.

We may experiment with different choices of S and B or even with different functions k. The Bayesian interpretation of such experimentation is that our model for f is actually a mixture of the models with different S, B and k. Appropriate Bayesian inference about f then involves weighting together results conditional on the various S, B, k choices, with weights given by the posterior p.d.f. values. The Bayesian posterior p.d.f. is the product of the prior p.d.f. over S, B, k with the likelihood function, i.e. the value of the p.d.f. for the data conditional on S, B, k. The value of the log p.d.f. for the data is thus a valuable measure of the degree to which the model fits, legitimately used to generate relative weights for various possible interpolation models. The log p.d.f. under normality is proportional to

$$-.5\log|\Omega| - .5f_0'\Omega^{-1}f_0 \ .$$

The projection formula for interpolation is invariant to scalar multiples of the R function, so it is better to have a measure of the log posterior p.d.f. with a scaling of Ω integrated out. This is, omitting a constant

$$-.5n \log f_0'\Omega^{-1}f_0 - .5 \log |\Omega| \ .$$

The version of the program now being circulated uses

$$k(x) = \exp(-x'x) \quad .$$

This implies a prior belief that f has derivatives of all orders and results in very well-behaved f^* functions. It also tends to push Ω to singularity rather rapidly as n increases for low-dimensional x. An alternative which has also been used with some success is

$$k(x) = \prod_{i=1}^{m} d(|x_i|) \text{ , where}$$

$$d(x) = \begin{cases} 1 - 1.5x^2 + .75\, x^3 , & 0<x<1 \\ .25(2-x)^3 , & 1<x<2 \\ 0 , & x>2 \end{cases} .$$

This is a product of third-order splines. It implies f is with probability one mean square differentiable to the second order, but not of higher order. It also makes observations on f separated by more than $4\tau_i$ along any co-ordinate direction independent. By making f^* more ill-behaved, this k makes analysis of it (particularly finding its peak by iterative algorithms) somewhat slower. It also tends to make Ω a better-conditioned matrix. For large problems, by putting many zeroes into the Ω matrix, this k may eventually provide scope for speeding the inversion of and shrinking the storage for Ω. The program does not exploit zeroes in Ω in its present form.

Note that, for any symmetric k, if $\{x_1,\ldots,x_n\}$ lie in a q-dimensional linear subspace, the maximum or minimum of f^* will always lie in that same subspace. This is a generalization of the intuitively obvious point that if there is only one point, $\{x_1, f_1\}$, then f^* has a unique minimum or maximum at that point. While this can be helpful in, say, exploring variations in only a subvector of x, it can lead to error. In particular, to explore variation in the whole of an m-dimensional x vector, with LOOPSMTH requires that there be at least m+1 linearly independent points in $\{x_1,\ldots,x_n\}$.

2. What the Program Will Do

The program is available in batch and interactive versions. The interactive version is more flexible and is discussed first.

The program begins by reading an input file to find (x_i, f_i), the vector of τ_i, and B. It immediately computes the likelihood function value and offers you the option of examining Ω^{-1}.

Next you are offered a list of options. One is that of altering τ and B. The first time you begin work with a set of data, it is wise to experiment here a bit, even though on an AT reinverting Ω takes a noticeable amount of time for n beyond 20 or so. If τ and B are set at very unlikely values, the program can produce bizarre results. With τ too small, the interpolated function will stay near B, with isolated peaks or pits at each x_i. With τ too large, the program will extrapolate wildly.

Another option is listing the f_i values with their sequential indexes i, sorted by f_i size. This helps in deciding which parts of the f^* surface deserve exploration.

You can also list the f_i and x_i together, in their original order. This can fill up several screens when n is substantial.

You can evaluate f^* at any x you wish to provide, including data points x_i. Once this is done, you have the option of letting the program proceed to take a Newton-Raphson step toward the peak of f^*, and you can keep this process going as long as you like.

You can invoke an automatic maximization routine, which asks for a starting point, then a maximum number of steps, and proceeds to climb f^* iteratively .

Another option allows alteration of the function-change convergence criterion of the automatic maximizer.

After any function evaluation or maximization exercise, you can have the program print the parameter vector to a file.

The batch version, LOOPSMTH, reads an input file (which must be named LOOP.DAT) of the same sort as the interactive version's, but proceeds directly to search for the maximum of f^*, starting from the x_i which gave the highest f_i. It then writes out the x at this maximum of f^*, putting the result on the file LOOP.PAR. A DOS batch file (using the COMMAND /C command to invoke a second batch file) can iteratively: invoke LOOPSMTH; pass x_n in LOOP.PAR to a second program which computes $f(x_n)$ and adds the x_n, f_n pair to the input file for LOOPSMTH; invoke LOOPSMTH again, etc. The program which computes $f(x_n)$ could even be running on a mainframe through a communications package.

3. Details

A. Choosing τ and B.

The scale factor τ_i should be a guess as to how large a change can be made in the i'th element of x with only a modest corresponding change in f. You expect f(x) to be similar for two x's which differ only in the i'th component and by no more than τ_i. B should be below at least some of the f_i values -- otherwise f^* may have no local maxima. If B is too small, a steep and unlikely rise from B to the observed f_i values may be implied, resulting in strange behavior for f^*.

When the observed f_i are very regular, commonly for example when, with n small, they all lie close to a linear function of the x_i, the likelihood may increase without bound as τ is increased. This does not mean that extremely large τ are reasonable choices, as this will imply radical extrapolation outside the observed range of x's. Furthermore, as τ is scaled up, eventually one reaches a singular Ω and numerical difficulties (on which, see below).

One should experiment with τ and B to make the likelihood large, scaling all the elements of τ by factors of 1.5 to 4 at a time. Usually the likelihood is single-peaked as a function of the scale of τ, and having that scale right to within about a factor of two is enough to get reasonable results.

B. Numerical Accuracy

The program, sadly, writes out an "Accuracy problems" message whenever it attempts to invert a non-p.s.d. matrix. During its attempts to climb to a peak in f^* this does not indicate any numerical problems at all. The same message (followed by a warning that the likelihood may be unreliable) during the calculation of the likelihood does indicate numerical problems. The program attempts to recover, and its attempts will be successful if the problem arises because of exact duplication of x_i, f_i by x_j, f_j. Otherwise, the attempts are usually inadequate. Furthermore, accuracy problems can arise even when the program does not flag them during calculation of the likelihood.

A test for accuracy problems is to evaluate $f^*(x_i)$. Any difference between this value and $f(x_i)$ reflects numerical inaccuracy. Usually, where f is a log likelihood, results are useful so long as $f^*(x_i)$ is accurate to about the third decimal place.

Another symptom of accuracy problems is "aborted search" messages from the automatic maximization option. This message emerges most often when f^* is not behaving locally like a quadratic function, as when rounding error makes f^* discontinuous. Sometimes results are useful despite "aborted search" messages, particularly if the peak of f is very close to one of the x_i's, but these messages are a warning.

Accuracy problems can sometimes be reduced by scaling down the τ vector. They can also sometimes be reduced by eliminating some of the x_i, f_i values on the input file. If one is interested in finding the maximum of f, the x's farthest from the maximum are the likeliest candidates for omission; if one is interested in the global shape of f, the x's in the most densely sampled region of x space are the likeliest candidates.

C. Input Files

The main input file has the following format:

CRIT

m

$\tau_1, \ldots, \tau_m$, B, DEL

$x_{11}, \ldots, x_{1m}$, $f(x_1)$

...

$x_{n1}, \ldots, x_{nm}$, $f(x_n)$

Data items are actually separated by spaces, not commas. The item shown beginning each line must always begin a new line, but the remaining items on the line may run over to additional lines if desired. CRIT is a positive real number defining the minimum change in f^* below which iterative maximization is declared to have converged. DEL is a positive real number which is a reasonable amount to expect f^* to increase in one iteration -- it is used to set an initial step size when the Hessian is not positive definite and a step along the gradient is taken instead. A value of about .2 seems to work for f a likelihood function. The program determines n by itself by counting the number of data points it reads. The AT version can hold a maximum of n=50.

The program in its interactive version also coummunicates with the user over standard input and output. The LOOPSMTH version requires (for no good reason) that the user input "Y" or "N" to the question of whether to work with parameters logged and to the subsequent question of whether to write out the projection matrix. Almost always it makes sense to invoke LOOPSMTH in the form "LOOPSMTH < LOOP.YN", with the appropriate Y's or N's placed on the two-line LOOP.YN file, since the main reason for using LOOPSMTH is to let the program run unattended.

D. Getting the Program

The program, whose development was supported in part by the National Science Foundation, is in the public domain and may be freely copied and altered. The latest version of the writeup and the program can be obtained, on double-sided IBM PC format diskette, by sending $10 to cover costs to

Christopher Sims
Department of Economics
University of Minnesota
Minneapolis, MN 55455.

The diskette contains BAYES0.EXE, the interactive version of the program using the exponential k, LOOPSMTH.EXE, the looping version of the program, a sample input file, fortran source code for the algorithm, and a Microsoft format object file library containing compiled code for the matrix algebra utilities invoked by the program. Thus with Microsoft Fortran version 3.0 or higher the user should be able to recompile the code with improvements or variations.

Note that this program in its present form is intended for sophisticated users. No support at all is promised for users who have questions.

4. Possible Improvements

It should be obvious that the program contains infelicities in the way it interacts with the user which ought to be eliminated. It ought also to be altered so that at least two distance functions and batch or interactive mode can be invoked as options, since keeping several complete versions of the EXE file wastes space.

More fundamentally, the program ought to be able to search for a better τ, B settings on its own and ought to recognize and circumvent numerical accuracy problems automatically.

If the program proves useful enough, it will probably be revised along these lines. Any user who makes such improvements, or others, or who finds significant bugs in the program, is urged to send his or her results to the address given above.