

WAVELET-BASED FUNCTIONAL MIXED MODEL ANALYSIS: COMPUTATION CONSIDERATIONS

Richard C. Herrick and Jeffrey S. Morris
Department of Biostatistics and Applied Mathematics
The University of Texas M. D. Anderson Cancer Center, Houston, TX

ABSTRACT

Objective: Optimize a C++ implementation of the wavelet-based functional mixed model methodology of Morris and Carroll (2006) and test its performance on a variety of functional data sets representative of biomedical applications.

Methods: Wavelet-based Functional Mixed Models is a new Bayesian method extending mixed models to irregular functional data (Morris and Carroll, 2006). These data sets are typically very large and can quickly run into memory and time constraints unless these issues are carefully managed in the software.

Results: We reduced runtime and memory use by 1.) identifying and optimizing hotspots, 2.) using wavelet compression and other approximations to do less computation with minimal impact on results, and 3.) dividing the code into multiple executables to be run in parallel using a grid computing resource. We present rules of thumb for estimating memory requirements and computation times in terms of model and data set parameters.

Conclusion: We present examples and benchmarks demonstrating that it is practical to analyze very large data sets with readily available computing resources. This code is freely available on our website.

INTRODUCTION

Wavelet-based Functional Mixed Models can present a number of computational challenges when applied to the large spectral and imaging data sets that are frequently generated by modern biomedical experiments.

Although input and output files can be quite large, on the order of gigabytes, they can be readily handled by today's low-cost hard drives.

Physical memory, on the other hand, is still expensive and limited, and therefore has to be managed carefully. Larger data sets can easily require gigabytes of RAM and exceed the physical memory of the system.

Depending on the data set, the design matrices applied, and number of MCMC samples required, execution time on a single processor can range from a few hours to several days.

This poster presents a C++ implementation of WFMM. Although other scripting languages such as those provided by Matlab, Splus, and R may be easier to develop statistical applications in, C++ provides a greater opportunity to achieve efficiencies and more economical use of memory since it is closer to the hardware. This makes it possible to write more efficient code and do more performance tuning. Moreover, much of the convenience of the aforementioned

scripting languages can be brought to C++ through the construction of a good class library.

Representative Data Sets

SELDI-TOF proteomics mass spectroscopy data set from organ-by-cell line experiment. Tumors from either A375P human melanoma cell line or PC3MM2 prostate cancer cell line were implanted in either the brain or lung of 16 nude mice. 32 spectra were obtained from blood serum samples, modeled by 5 fixed effects for organ-cell-line effects and laser intensity.

MALDI-TOF proteomics mass spectroscopy data set from organ-by-cell-line experiment. Tumors from one of 3 cell lines were implanted in one of 4 organs of 147 nude mice. Spectra were obtained from blood serum of these mice as well as 37 controls. Design matrix includes cancer vs. normal effect, 12 effects for the organ-by-cell line groups, and gender effect.

MALDI-TOF proteomics mass spectroscopy data from pancreatic cancer experiment. Blood serum was taken from 139 pancreatic cancer patients and 117 controls. Design matrix allows for block effect as well as cancer effect. No random effects were included.

Planet Health Children’s Activity Study – Data set was recorded with TriTrac activity monitor (Hemokentics, Inc. Madison, WI), which provides minute-by-minute acceleration counts from motion sensors in three-dimensions. Two data sets were benchmarked consisting of weekday and weekend complete accelerometer profiles.

Rat colon carcinogenesis study – 30 rats were fed two different diets rich in either fish-oil or corn oil, then sacrificed at 5 different time intervals after exposure to a carcinogen. MGMT levels were measured as a function of cell depth.

2D gel data sets – This study consisted of 7 mice in each of three groups: “abstainers”, “social drinkers”, and “alcoholics”. After a period of time, each animal was sacrificed, brain tissue from the amygdale region extracted, and run on 3 replicate 2d gels. The goal of this study is to identify proteins whose expression levels reflect changes in the brain due to alcoholism.

Linear Mixed Models

Suppose we have a sample of N observations, represented by the vector Y . From Laird and Ware (1982), the linear mixed model equation is:

$$Y = X\beta + Z u + e, \text{ where } u \sim N(0, Q) \text{ and } e \sim N(0, S)$$

- The general linear form of the fixed effects, $X\beta$, accommodates a broad class of possible structures, including main effects and interactions from ANOVA, linear regression, and ANCOVA.
- The random effects Zu provide a convenient mechanism for modeling correlation among the N observations, e.g. when they come from the same experimental unit or block.

Functional Mixed Models

Suppose we have a sample of N “observed functions”, e.g. mass spectra. Let $Y(t) = \{Y_1(t), \dots, Y_N(t)\}'$ be the N observed functions, “stacked” as rows. Morris and Carroll (2006) present the following functional mixed model, which generalizes the linear mixed model to functional data.

$$\underbrace{Y(t)}_{N \text{ functions}} = \underbrace{\sum_{N \times p} B(t)}_{N \times p \text{ functions}} + \underbrace{\sum_{N \times m} U(t)}_{N \times m \text{ functions}} + \underbrace{E(t)}_{N \text{ functions}}, U(t) \sim MGP(P, Q), E(t) \sim MGP(R, S)$$

- P and R are correlation matrices ($m \times m$ and $N \times N$, respectively)
- Q and S are covariance surfaces defined on the functions' domain space
- $U(t) \sim MGP(P, Q)$ means that $P^{-1/2}U(t)$ is mean zero Gaussian Process with covariance surface Q . This is the functional generalization of the Matrix Normal distribution of Dawid(1981).

Suppose we observe all functions on a common grid $\mathbf{t} = \{t_1, \dots, t_T\}$ of length T

Discrete Version of Functional Mixed Model:

$$\underbrace{Y}_{N \times T} = \underbrace{\sum_{N \times p} B}_{N \times p \times T} + \underbrace{\sum_{N \times m} U}_{N \times m \times T} + \underbrace{E}_{N \times T}, U \sim MN(P, Q), E \sim MN(R, S)$$

Q and S are $T \times T$ covariance matrices, and MN refers to the Matrix Normal distribution of Dawid(1981).

- For mass spectrometry data, think of each row of Y containing the intensities from one spectrum for a grid of m/z values \mathbf{t} .

Wavelet-based Functional Mixed Model

Rather than fitting this model to the data Y directly, we instead project the data into the wavelet space, and do the modeling there.

Wavelets: orthogonal basis functions that can be used to represent functions.

The projection can be represented by orthogonal projection $D = YW'$, but is more efficiently done by applying an $O(n)$ algorithm (DWT) to each row of Y .

Wavelet Space Version of Functional Mixed Model:

$$\underbrace{D}_{N \times T} = \underbrace{\sum_{N \times p} B^*}_{N \times p \times T} + \underbrace{\sum_{N \times m} U^*}_{N \times m \times T} + \underbrace{E^*}_{N \times T}, U^* \sim MN(P, Q^*), E^* \sim MN(R, S^*)$$

- $B^* = BW'$ & $U^* = UW'$ are wavelet coefficients for fixed/random effect functions
- We assume Q^* and S^* are diagonal, implicitly assuming wavelet coefficients within a given function are independent. This assumption is commonly used in wavelet regression, and is heuristically justified by the whitening property of the wavelet transform. While the wavelet coefficients are independent, this structure allows varying degree of correlation within a function.
- We put shrinkage priors on the elements of B^* that consist of a mixture of a point mass at zero and a normal distribution. When placed on wavelet coefficients, this type of prior results in functional estimates that are denoised but without substantial attenuation of the peaks of the function.
- We put vague priors on the Ω , the set of parameters indexing the covariance matrices P , Q^* , R , and S^* .
- Given estimates of B^* and U^* , we can easily obtain $B = B^*W$ and $U = U^*W$.

Fitting the Model

Given Y, X, Z , wavelet basis, and choice of structure for P and R , we can use MCMC to obtain posterior samples of the fixed and random effect functions $B(t)$ and $U(t)$ and covariance matrices/surfaces P, Q, R , and S on the grid \mathbf{t} :

Working with the likelihood w/ U^* integrated out; Repeat for $g=1, \dots, G$ iterations:

1. Sample from $f(B^*|D, \Omega)$, which are mixtures of normals and point masses at 0.
2. Sample from $f(\Omega|D, B^*)$, which is done using Metropolis-Hastings steps.
 - Use IDWT to obtain posterior samples of $B(t)$ on grid \mathbf{t} from B^* .

Application Architecture

The computation readily separates into three components: Initialization, MCMC Loop, and Summarization. These are linked into a single executable, WFMM, for serial execution on a single PC or into three executables, WFMM1, WFMM2, and WFMM3 for parallel computing. A Windows bat file runs WFMM1 to compute initial parameters, then submits WFMM2 to N processors using the grid computing facility Condor (www.cs.wisc.edu/condor) and waits for them to finish. It then runs WFMM3 to summarize the results.

Initialization

Apply discrete wavelet transform (DWT) to each function to obtain sets of wavelet coefficients.

Compute starting values for fixed effect functions and variance components using maximum likelihood; estimate variance of MLE for variance components to use as proposal variance in random-walk Metropolis-Hastings step. These automatic proposals are important, since they allow the MCMC to proceed automatically.

Use Empirical Bayes method to estimate smoothing parameters from the data, and also to set up vague, proper priors with 1 unit of information for each variance component.

MCMC Loop

Random Walk Metropolis Hastings sampling of variance components

Gibbs sampling of Beta (mixtures of point masses at zero and Normals)

Summarization

Apply inverse DWT (IDWT) to MCMC samples of wavelet coefficients for fixed and, if desired, random effect functions to obtain MCMC samples for fixed and random effect functions.

Compute mean, pointwise variance, and pointwise credible intervals for fixed effect functions.

METHODS

The WFMM application was implemented as a Windows console application in C++, using Visual Studio 2005 as the IDE. A class library developed internally by the Biostatistics department (BiostatGeneral) was used to provide the vector and matrix handling as well as random-number generation. Linear algebra

operations were provided by Intel's Math Kernel Library 8.1. Benchmarks were run on a Dell Precision Workstation 650 with Intel Xeon 2.4 GHz processor and 2 GB RAM.

Software Tools

The following applications and system functions were used to improve performance and minimize memory usage:

- VTune Performance Analyzer (Intel Corp.): This application can analyze a test run of the application down to the instruction level and estimate how much time is spent in each function and even how much time is spent on each instruction within a function. It can quickly identify hot spots and CPU time wasters.
- Timing Macros (Windows system) : The code is instrumented throughout with system time calls to compute elapsed time for each component and major functions within each subcomponent in order to provide benchmarks.
- Perfmon: (part of Windows): Performance monitor was run concurrently with the WFMM application to monitor available memory along with many other system parameters in real-time.
- GetMemoryStatus function (Windows system call): Allows instrumentation of application to determine how much memory is available.
- Memory Leak detection with `_CrtDumpMemoryLeaks()` (Windows system call): Allows for checking for memory that is not being deallocated from the heap and to determine which line of code allocated the memory.
- Condor: Some of the applications with a large runtime and/or memory requirement benefited from running multiple MCMC loops in parallel. Condor, which is a grid computing application using multiple networked but not dedicated PC's, was used to manage the parallel jobs.

Optimizing Runtime

VTune or timing macros were used to identify the most time-consuming function, which was computation of the generalized cross products (GCP) on each step.

We then:

- Identified the most efficient algorithms for computing the function. A key step in keeping the calculations computationally feasible is that the generalized cross products $X'\Sigma^{jk-1}X$, $X'\Sigma^{jk-1}d_{jk}$, etc. are computed using a SWEEP operator (a la PROC MIXED, Wolfinger, et al. 1994) and stored at each iteration. Use of the SWEEP operator prevents the need to compute high-dimensional inverses or matrix multiplications. These GCP are sufficient pivotal quantities for updating the fixed effect functions, and are also the key quantities in the Metropolis-Hastings for the variance components, as well.
- Made sure the 10% of code that takes up 90% of CPU time (the SWEEP step for updating the GCP) is coded as efficiently as possible using the most efficient C++ coding.

- Used VTune to identify the top 10 to 15 most time consuming functions for any clues to time-wasting practices; for instance this method identified the new operator as a major time consumer because large buffers were repeatedly allocated and deallocated because the objects allocating them were instantiated locally. These were made member variables so memory would be persistent.

Optimizing Memory Usage

The point where memory use is maximum was first determined by monitoring available memory with perfmon during a memory intensive run. Breakpoints can be set in the debugger if necessary to freeze execution at a specific line. Then the following steps were taken to maximize available memory at that point:

- Checked for memory leaks using `_CRTDumpMemoryLeaks()` system call in case some lower level functions are not releasing memory.
- Ensured memory buffers were deallocated when no longer needed so that subsequent objects could use it.
- Avoided double buffering when possible, such as passing by reference rather than returning a large array on the stack.

Parallel Computing

Once reasonably optimized, the code was split into three executables for initialization, MCMC loop, and summarization so that the MCMC loop, which was the most time-consuming, could be run as multiple independent chains in parallel on multiple processors. We used the grid utility Condor to run this on non-dedicated PC's, but any grid computing utility or distributed memory cluster will work.

Computational Shortcuts

premise that significant computation is spent on elements that have little influence on the final results and will not change the inference. Since their effect is data dependent, the degree of application or whether to use them at all is controlled by the user:

- Wavelet compression; Sort wavelet coefficients in decreasing magnitude, then keep only the K' largest coefficients that preserve a certain percentage (typically ~99%) of the total energy (sum of squared data values). Set others to 0 and drop them from MCMC computation, then fill with 0's on reconstruction. This easily speeds up computations by a factor of 5 to 100, without substantially impacting results in many cases.
- Set $q_{jk} = 0$ and do not update in MCMC if the MLE starting value is less than some prespecified threshold, say 10^{-6} . This saves considerable time and does not strongly impact results, since we expect parsimony in the wavelet space for the covariance parameters.
- For other q_{jk} that are updated in the MCMC, if the current value is very small, especially relative to s_{jk} , then when using the sweep operator to update the generalized cross-products $X'\Sigma^{-1}X$, $X'\Sigma^{-1}D$, etc., substitute

$q_{jk}=0$. Again, this results in substantial savings, since updating the GCP represents the most costly computational step, and in these cases, q_{jk} is so small that the GCP remain virtually the same when substituting $q_{jk}=0$.

RESULTS

Estimating Memory and Runtime

The enormously large data sets typically fit with this method present tremendous computational challenges, both in terms of memory requirements and computational times. One must check these requirements before implementing this method on a given system. Also, the large number of MCMC samples kept for each parameter present some challenges in terms of hard disk space, which are also important to consider up front. Memory requirements and computation times can be estimated from data set size and design matrix parameters:

N = Number of Functions T = Number of observations/function
 B = Number of MCMC samples K = Number of wavelet coefficients
 p = Number of fixed effect functions K' = Number of non-thresholded wavelet coefficients
 m = Number of random effect functions H = Number of levels of random effect functions
 c = Number of strata for residual error functions
 B' = Number of samples in memory

$$\text{Disk Usage} \approx 8 [BK'(p+H+c+1) + pT(B+4) + 3NT + 2NK' + K'(p^2+m^2+pm+m+3+7p+7(H+c))]]$$

$$\text{RAM Usage} \approx 8[B'(2pT+(H+c+2)K'+T) + (0.05B+4)pT + 3(H+c)K']$$

Total runtime is more difficult to estimate but its dependence on parameters can be inferred by inspection; runtime is dominated by the MCMC loop and in particular by the sweep calculation; initialization and postprocessing runtimes have not been a significant consideration.

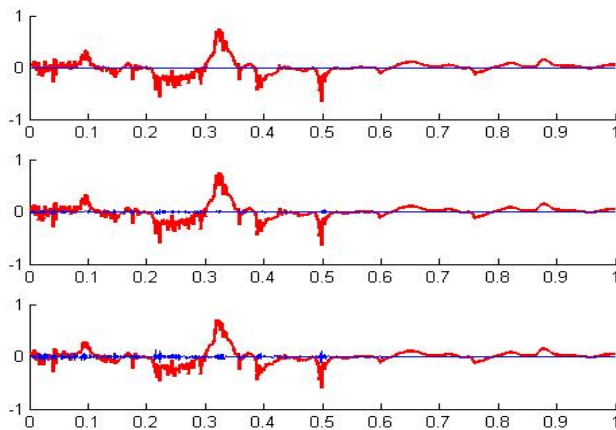
Runtime depends linearly on number of iterations = $B^*(\text{thin}) + \text{burnin}$ and the number of nonthresholded wavelet coefficients K' . Since the MCMC loop is dominated by the sweep calculation, we can estimate runtime dependence by estimating floating point operations as a function of the swept matrix size and number of sweeps (m).

$$\text{Sweep Runtime} \sim K'm(p + 2m + 1)^2, \text{ or } O(K'm^3) \text{ for } m \gg p \text{ and } O(K'p^2) \text{ for } p \gg m.$$

Actual Runtime and Memory Usage

Table 1. Measured runtimes, RAM, and hard disk usage for seven functional data sets demonstrating effect of parameters. Runtimes are based on 11,000 iterations (burnin=1000, thin=5, B=2000) and $K' = K \sim T$ for all cases except 2D Gel, where wavelet compression was used to reduce K' to 9529. Note strong dependence of runtime and RAM on m demonstrated by TriTrac weekday set and effect of large K demonstrated by proteomics and 2D Gel data sets.

	N	T	m	p	Runtime (hr/min)			RAM(MB)			HD(MB)
					Init	MCMC	Post	Init	MCMC	Post	
Rat colon carcinogenesis	256	256	30	13	0/1	1/8	0/1	42	97	148	138
TriTrac weekday	237	660	148	4	0/38	125/2	0/1	157	271	92	136
TriTrac weekend	52	540	45	4	0/2	1/1	0/1	28	103	144	112
Pancreatic Cancer MALDI	256	12096	0	5	0/11	3/7	0/9	132	840	930	2532
Organ-by-Cell-Line SELDI	32	7985	16	5	0/15	10/50	0/6	78	1093	600	1629
Organ-by-Cell-Line MALDI	184	6518	0	14	0/12	3/15	0/13	62	901	1211	3079
2D Gel data	57	490448	21	3	0/55	15/39	4/15	807	803	1194	24034



- Wavelet compression speeds up runtime by 5 to 20 fold without qualitatively altering results.
- Subsequent peak detection showed no data loss with the 5-fold reduction. 20-fold reduction was lossy and therefore not appropriate for this application.

Figure 1. Beta curve showing cancer effect for mass spect dataset.

a.) Uncompressed b.) 5-fold reduction c.) 20-fold reduction. Blue curve shows difference between compressed and uncompressed data

CONCLUSIONS

- It is practical to analyze even very large data sets with readily available computing resources using WFMM methodology.
- This application is available for free download from:
http://odin.mdacc.tmc.edu/~jeffmo/papers_files/wfmm_supplement.html

REFERENCES

1. Morris JS and Carroll RJ: Wavelet-Based Functional Mixed Models. *Journal of the Royal Statistical Society, Series B*, 68(2): 179-199, 2006.
2. Wolfinger, R., Tobias, R. and Sall, J. (1994) Computing Gaussian likelihoods and their derivatives for general linear mixed models. *SIAM J. Scient. Comput.*, **15**, 1294–1310.j
3. Laird N. and Ware J.H. (1982). Random effects models for longitudinal data. *Biometrics*, 38, 963-974.
4. Dawid, A.P. (1981). Some matrix-variate distribution theory: Notational considerations and a Bayesian application. *Biometrika*, 68, 265-274.