Uncertainty Analysis for Computationally Expensive Models

David Ruppert

Dept. of Statistical Science and School of Operations Research and Information Engineering, Cornell University

Mar 12, 2013

Collaborators

- Christine Shoemaker, Professor, Civil and Environmental Engineering, Cornell University (works in optimization)
- Nikolay Bliznyuk, Assistant Professor, Statistics, University of Florida
- Yingxing Li, Assistant Professor, Xiamen University
- Yilun Wang, Associate Professor, University of Electronic Science and Technology of China

- Calibration: estimate parameters in a model
- Uncertainty analysis: confidence or credible region, etc.
- Bayesian modeling and MCMC are particularly suitable for the calibration and uncertainty analysis
- A standard implementation requires the evaluation of a model (simulator) at each MCMC iteration
  - but often the model is computationally expensive
- A computationally feasible approach uses an emulator (interpolant) in place of the simulator

Outline, continued

- The emulator must be developed using a relatively small number of simulator evaluations
- These evaluations should be concentrated in the high posterior density region (HPDR)
  - The HPDR could be less than 1% of the parameter space
  - the location and shape of the HPDR is not known in advance
- Evaluations that are close to each other in the parameter space are wasteful
  - so are those outside the HPDR

Outline, continued

- Our algorithm iterates between
  - using the current emulator to select new points for running the model
  - updating the emulator using the new evaluations
- Except for a paper of Rasmussen, we are not aware of other work where the emulator is built on a small and a priori unknown set

SOARS = Statistical and Optimization Analysis using Response Surfaces

- SOARS has 4 steps and iterates between the final 3 steps
  1. locate the posterior mode using global optimization
  2. explore the region around the mode to learn the size and shape of the HPDR using GRIMA (Grow the (design) Region and IMprove the Approximation) (Bliznyuk et al., 2012)
  3. construct a Radial Basin Function (RBF) emulator (response surface) of the log posterior
  4. run MCMC using the emulator

- $\mathbf{Y}_i = (Y_{i,1} \ldots, Y_{i,d})^\mathsf{T}$, $i = 1, \ldots, n$, is a multivariate time series
- $\boldsymbol{f}_i(\boldsymbol{\beta}) = \Big(f_{i,1}(\boldsymbol{\beta}), \ldots, f_{i,d}(\boldsymbol{\beta})\Big)^\mathsf{T}$ is the simulator output for time i
- $\boldsymbol{\beta}$ is the vector of unknown parameters in the simulator
- In the absence of noise we expect that

$$\mathbf{Y}_i = \boldsymbol{f}_i(\boldsymbol{\beta})$$

- Noise can be modeled using standard techniques such as
  - transformations
  - variance functions
  - time series models

Example: Town Brook watershed

- Town Brook is in the Cannonsville watershed, part of the NYC water supply
- Town Brook is a small watershed so works well as a case study
  - MCMC using the exact posterior is feasible, although it takes over a week
  - therefore, SOARS can be compared with an exact implementation

Town Brook watershed: data and simulator

- $\mathbf{Y}_i = (Y_{i,1}, Y_{i,2})^{\mathsf{T}} =$ (flow, concentration of phosphorus) on ith day
- $\boldsymbol{f}_i(\boldsymbol{\beta})$ is output from SWAT2005 (Soil and Water Assessment Tool, 2005 version)
    - SWAT takes seconds to run on the Town Brook watershed
    - SWAT will take minutes or hours on larger watersheds
    - $\boldsymbol{\beta}$ is vector of eight parameters in the SWAT model

SOARS Step 1: Optimization

- $\boldsymbol{\theta}$ contains β (model parameters) and noise parameters
- $\pi(\boldsymbol{\theta}|\mathbf{Y})$ is the unnormalized posterior = likelihood $\times$ prior density
- The goal is to find the HPDR, characterize it, and construct the emulator on it
  - the HPDR is a $1-\alpha$ credible region for some small α
- The HPDR is located by using a global maximizer to find the posterior mode
  - high accuracy is not important
  - we only need to get into $C_R(\alpha)$, not find the mode

SOARS Step 2: GRIMA

- After optimization, but before GRIMA, evaluate the log-likelihood on a Latin hypercube centered at the (approximate) mode
- GRIMA produces a nested sequence $\mathcal{D}_0, \mathcal{D}_1, \ldots$ of sets of evaluation points
- $\mathcal{D}_0$ is the set of evaluation points from optimization plus those from the Latin hypercube
  - except "outliers" (outside the HPDR) are excluded

SOARS Step 2: GRIMA, continued

- Given the current set $\mathcal{D}_i$, let $\mathcal{C}$ be the set of parameter values whose distance from $\mathcal{D}_i$ is exactly r.
  - r is a tuning parameter that varies during GRIMA
- Let $\tilde{\ell}_i$ be the emulator of the log-posterior on $\mathcal{D}_i$.
- The candidate for the next evaluation point is the point in $\mathcal{C}$ where $\tilde{\ell}_i$ is maximized.
- Because this point is exactly at distance r from $\mathcal{D}_i$, it is neither
  - redundant (too close to the current evaluation points) nor
  - well outside the HPDR (too far from them)

SOARS Step 2: GRIMA, continued

- GRIMA allows r to increase initially so that the entire HPDR is covered quickly
- Eventually r decreases so that the set of evaluation points becomes dense

SOARS Step 3: RBF interpolation

- the RBF response surface is updated repeatedly
  - Bliynyuk et al. (2012) have an efficient algorithm for updating
- RBF interpolation is sensitive to the parametrization and is improved by sphering

Step 4: MCMC

- MCMC using the emulator is run after GRIMA terminates to estimate the posterior
- MCMC is also used during GRIMA to decide when to terminate
    - termination occurs when the total variation norms between successive estimates of the univariate log posterior densities are small
    - norms estimated by importance sampling

Summary

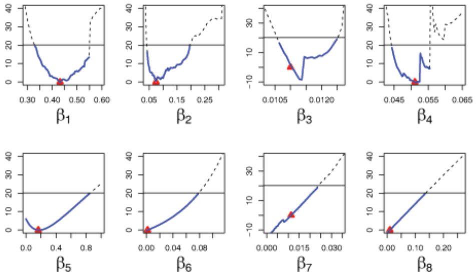
- In summary, SOARS has 4 steps and iterates between the final 3 steps
  1. locate the posterior mode
  2. explore the region around the mode to learn the size and shape of the HPDR
  3. construct a Radial Basin Function emulator of the log posterior of the HPDR
  4. run MCMC using the emulator

- $h(\mathbf{Y}_i, \boldsymbol{\lambda}) = h\{\boldsymbol{f}_i(\boldsymbol{\beta}), \boldsymbol{\lambda}\} + \boldsymbol{\varepsilon}_i$ (transform-both-sides)
    - $h(\mathbf{y}, \boldsymbol{\lambda}) = \{h(y_1, \lambda_1) \cdots h(y_d, \lambda_d)\}^\mathsf{T}$
    - $h(y, \lambda) = (1 - \Delta) h_{BC}(y, \lambda) + \Delta \log(y)$
    - $h_{BC}(y, \lambda)$ is the Box-Cox family
    - therefore, $\boldsymbol{\varepsilon}_i$ can be (multivariate) Gaussian
- $\boldsymbol{\varepsilon}_i = \boldsymbol{\Phi} \boldsymbol{\varepsilon}_{i-1} + \mathbf{u}_i$ (vector AR(1))
    - $\mathbf{u}_i$ is Gaussian white noise with covariance matrix $\boldsymbol{\Sigma}_u$
- noise parameters are $\boldsymbol{\lambda}$, $\boldsymbol{\phi}$, and $\boldsymbol{\Sigma}_u$

Town Brook Optimization

- Optimization was done with DSS, a global optimizator
- 1900 function evaluations were used
  - problem: SWAT output is nonsmooth with many local maxima and 8 parameters
- For a more computationally expensive simulator, one would need to parallelize the optimization

Town Brook Profile Plots



Plots of -2 log(posterior) using the exact unnormalized posterior

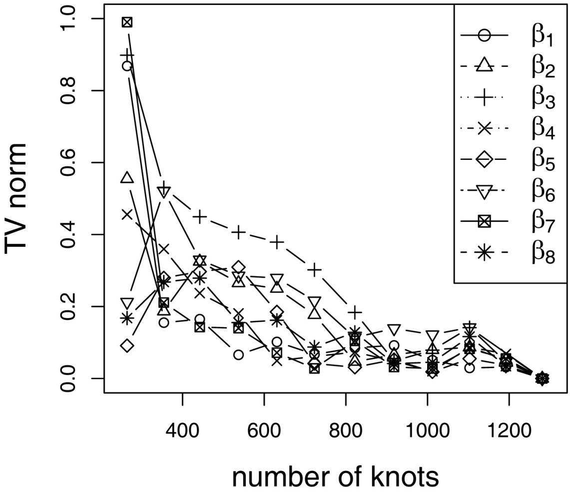
Parameters varied one at a time about DDS termination point at $\blacktriangle$

Horizontal line at $\chi^2_8(.99)$

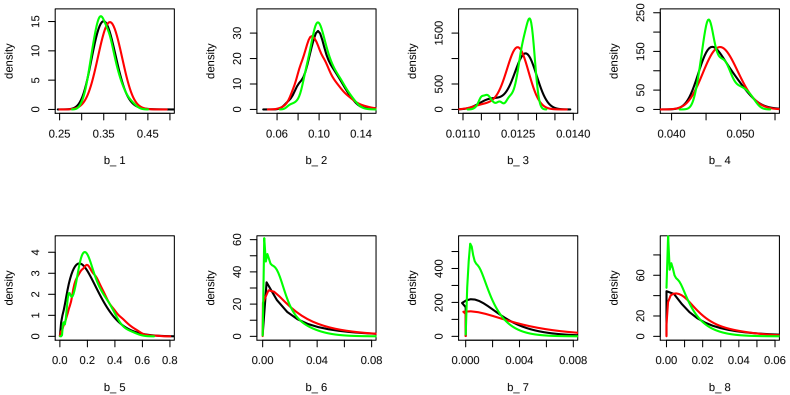
Town Brook: GRIMA

- We did 500 evaluations prior to GRIMA with a Latin hypercube design
- GRIMA used a total of 1017 function evaluations
- A total of 3,517 expensive evaluations were used for
  - optimization
  - the Latin hypercube sample, and
  - GRIMA

Town Brook: stopping GRIMA



Town Brook: Accuracy of the Emulator

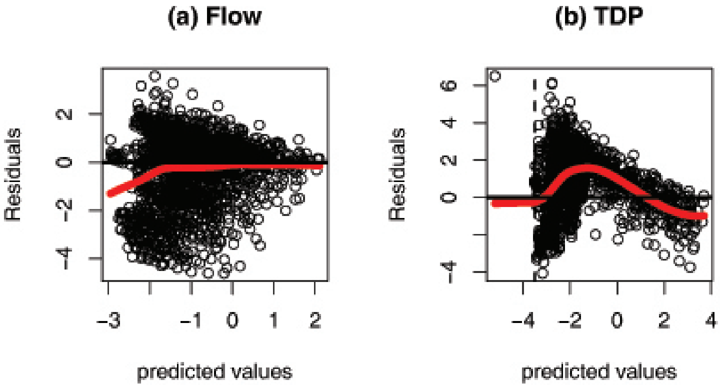


Black: 60,000 MCMC iterations with exact posterior

Red: SOARS (3517 exact plus 60,000 iterations with emulator)

Green: 3500 MCMC iterations with exact posterior

Town Brook: Model Adequacy



Conclusions: Good News

- SOARS, especially the GRIMA algorithm, can handle the nonsmoothness of SWAT output
- Given a budget for the expensive evaluaitons, SOARS outperforms standard MCMC
  - it is better to use the expensive evaluations to build the emulator rather than for MCMC itself
- Uncertainty analysis and calibration took about 3,500 evaluations
  - calibration alone took 1,900 evaluations and was not particular accurate
  - the calibration was improved during the uncertainty analysis

Conclusions: Not-so-good News

- RBF interpolation suffers from the curse of dimensionality
- The nonsmoothness of SWAT output makes optimization difficult
- Thousands of expensive function evaluations are necessary with an 8-parameter SWAT model
- Parallelization is necessary for larger problems (e.g., more parameters or larger Watershed)