Research Statement

Samuel Thomas

The essence of Bayesian data analysis is to ascertain posterior distributions. Posteriors generally do not have closed-form expressions for direct computation in practical application. Analysts therefore resort to Markov Chain Monte Carlo (MCMC) methods for generation of sample observations that approximate the desired posterior distribution. Standard MCMC methods such as Gibbs Sampling and Metropolis-Hastings are well developed and widely used in analytical practice. These algorithms have been implemented in publicly available computing platforms such as WinBUGS, OpenBUGS, and JAGS.

Standard MCMC methods simulate sample values from the desired posterior distribution via random proposals. As a result, the mechanism that one uses to generate the proposals inevitably determines the efficiency of the algorithm. For example, Metropolis-Hastings proposals are generated by random walks. While the proposals have good theoretical properties, an unguided random walk is known to be inefficient in covering the support of the target density function. Gibbs Sampling can be inefficient as well, particularly when the posterior density has regions with narrow support. In high dimensional parameter spaces such as those encountered in semiparametric regression analysis, the convergence of these standard MCMC methods can be prohibitively slow for practical use.

Semiparametric Regression

Semiparametric regression is a modern regression technique that is frequently used to explore the existence of nonlinear and multidimensional associations. Unlike the traditional parametric regression models with fully specified functional relationships, a semiparametric model uses spline functions to approximate the unknown associations. In comparison with the more familiar parametric regression, semiparametric regression tends to provide much-enhanced modeling flexibility and improved goodness-of-fit. The gains, however, are achieved at the expense of increased model complexity. Depending on the nature of the functional relationships, semiparametric models often have very large numbers of parameters, and thus presenting a significant challenge for parameter estimation and model fitting. Simultaneous estimation of parameters in complex semiparametric regression models, for example, is often beyond the capacity of maximum likelihood-based methods. Standard MCMC techniques also fall short in achieving reliable estimation in the absence of extraordinary computing resources.

Hamiltonian Monte Carlo (HMC) (MacKay 2003)(Neal et al. 2011) is a relatively new MCMC algorithm designed to improve the efficiency of traditional Random Walk Metropolis (RWM) algorithm while retaining its flexibility of application. HMC replaces the naive proposal in RWM with an informed proposal based in part on the gradient of the log posterior. This gradient works in cooperation with latent variables to form trajectories through the joint state space.

I am developing an **R** package called semiparMCMC to fit parametric and semiparametric models using Hamiltonian Monte Carlo simulation. The interface for this package is designed to be intuitive for statisticians and analysts with experience using **R** software for regression modeling. This package can be used as a general-purpose statistical modeling tool. However, the application of HMC is of greatest advantage for large datasets with many parameters, where traditional frequentist approaches can scale poorly (Evans and Swartz 2000).

Ongoing and Future Research

As the volume of available data continues to increase, I anticipate that statisticians will desire model-fitting software capable of meeting the computational demands of big data. Throughout most of the past few decades statisticians have, perhaps unknowingly, benefitted from hardware advances associated with Moore's law (Present 2000), meaning that processors have generally doubled in efficiency approximately every two years. For statisticians, this has meant that existing code also became substantially more efficient over time thanks solely to hardware advances.

However, the semiconductor industry has recently acknowledged that Moore's law will not hold indefinitely (Waldrop 2016). I anticipate that this development will force the statistical community to develop a greater understanding of computer hardware as well as programming techniques to most efficiently utilize that hardware. Parallel computation among many processors is one technique computer scientists use to design highly efficient applications. Use of parallel computation in statistical software today is often limited to high-level computation of embarrassingly parallel tasks. I believe that many opportunities exist to use parallel computation to improve the efficiency of statistical software.

High-Performance Computing

HMC is more computationally intensive than its traditional counterpart, Metropolis-Hastings. Both HMC and Metropolis-Hastings perform a log posterior computation for each new proposal. However, HMC also performs many gradient log posterior computations for each proposal. All of these computations essentially reduce to common linear algebra tasks including matrix multiplication and cholesky decompositions.

Typical MCMC software may offer embarrassingly parallel computations of multiple MCMC chains across available CPU's. However, current MCMC software does not take advantage of the parallel computational power of the many more available processors of GPU's. The computational power of GPU's is currently leveraged for many analytical applications, including deep learning of neural networks (Schmidhuber 2015). I believe that statistical model-fitting software can also benefit from the this innovative utilization of GPU's for large, high-dimensional datasets.

The design matrices in HMC can be large and are often sparsely populated. When these matrices become sufficiently large, linear algebra computations may be more efficiently distributed to the GPU rather than performed locally on the CPU. However, computations involving smaller matrices will still be more efficiently performed on the local CPU due to the communication cost of using the GPU.

Statistical software will need to be able to handle both small and large applications efficiently. As such, one challenge in developing statistical software will be assessing the appropriate hardware to use for the particular application. One software library that is currently available to handle such tasks is provided by NVIDIA. NVIDIA's NVBLAS library dynamically assigns linear algebra tasks to either the CPU or GPU depending on the computational intensiveness. Such software may be able to help improve the efficiency of computationally intensive statistical applications such as HMC.

Another potential application of high-performance computing to HMC is the use of mini-batches instead of the full dataset. To date, research on the use of mini-batches to approximate the gradient in HMC have been limited (Chen, Fox, and Guestrin 2014). The appropriateness of such applications depends on the quality of the approximation of the gradient on the full dataset. Using mini-batches for all general applications of HMC may not be appropriate. However, for certain common modeling applications such as models based on the exponential family of distributions mini-batches may provide an opportunity for substantial improvements in computational efficiency.

References

Chen, Tianqi, Emily B. Fox, and Carlos Guestrin. 2014. "Stochastic Gradient Hamiltonian Monte Carlo." In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, II–1683–II–1691. ICML'14. Beijing, China: JMLR.org. http://dl.acm.org/citation.cfm?id=3044805.3045080.

Evans, Michael, and Timothy Swartz. 2000. *Approximating Integrals via Monte Carlo and Deterministic Methods*. OUP Oxford.

MacKay, David J. C. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Neal, Radford M., Steve Brooks, Andrew Gelman, and Galin Jones. 2011. *Handbook of Markov Chain Monte Carlo*. CRC Press.

Present, I. 2000. "Cramming More Components onto Integrated Circuits." *Readings in Computer Architecture* 56.

Schmidhuber, Jürgen. 2015. "Deep Learning in Neural Networks: An Overview." *Neural Networks* 61. Elsevier: 85–117.

Waldrop, M Mitchell. 2016. "The Chips Are down for Moore's Law." *Nature News* 530 (7589): 144.