Computational Statistics and Data Analysis xx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

Weighted particle tempering

Marcos Carzolio[*], Scotland Leman

Virginia Tech Department of Statistics, Hutcheson Hall, 250 Drillfield Drive, Blacksburg, VA 24061, USA

ARTICLE INFO

Article history:
Received 8 June 2016
Received in revised form 9 April 2017
Accepted 10 April 2017
Available online xxxx

Keywords:
Bayesian classification tree
Gaussian graphical model
Markov chain Monte Carlo
Multi-modality
Parallel algorithm

ABSTRACT

The application of Bayesian methods often requires Metropolis–Hastings or related algorithms to sample from an intractable posterior distribution. In especially challenging cases, such as with strongly correlated parameters or multimodal posteriors, exotic forms of Metropolis–Hastings are preferred for generating samples within a reasonable time. These algorithms require nontrivial and often prohibitive tuning, with little or no performance guarantees. In light of this difficulty, a new, parallelizable algorithm called weighted particle tempering is introduced. Weighted particle tempering is easily tuned and suitable for a broad range of applications. The algorithm works by running multiple random walk Metropolis chains directed at a tempered version of the target distribution, weighting the iterates and resampling. The algorithm's performance monotonically improves with more of these underlying chains, a feature that simplifies tuning. Through the use of simulation studies, weighted particle tempering is shown to outperform two similar methods: parallel tempering and parallel hierarchical sampling. In addition, two case studies are explored: breast cancer classification and graphical models for financial data.

1. Introduction

Since the inception of the Metropolis algorithm (Metropolis et al., 1953) and the Gibbs sampler (Geman and Geman, 1984), the scientific literature has seen an explosion of Markov chain Monte Carlo (MCMC) techniques for statistical inference. For details, Tierney (1994) is an excellent introduction to MCMC and many of its important features. It suffices to say that the underlying goal of statistical simulation is usually to characterize an unknown or complicated target distribution—usually a posterior. MCMC sets up a Markov chain whose stationary distribution is the distribution in question. Given enough iterations, a properly tuned MCMC will produce dependent random draws from its target distribution. We propose a new MCMC algorithm called weighted particle tempering, whose two major strengths over popular methods are simpler implementation (due to monotonicity in tuning) and scalability via parallelization. Weighted particle tempering performs well in a broad set of applications, and it is therefore an attractive alternative to other algorithms that are often time consuming to tune.

Several issues may arise that hinder the performance of an MCMC algorithm. Multimodal distributions, for example, provide a challenge for locally updated MCMC algorithms, since these algorithms get trapped in local modes. As a result, the researcher must choose between running the MCMC for an impractically long time to reduce Monte Carlo variance, or moving on to a more sophisticated algorithm altogether. Multimodal target distributions appear in many applications, such as in evolutionary or physical systems modeling, time series forecasting, or mixture models.

There are many methods that explore multimodal distributions well, starting with replica exchange Monte Carlo (Swendsen and Wang, 1986). Metropolis coupled MCMC (Geyer, 1991), more commonly known as parallel tempering,

[*] Correspondence to: 9 Christopher St., Apt. 9, New York, NY 10014, USA.
E-mail address: cmarcos8@vt.edu (M. Carzolio).

http://dx.doi.org/10.1016/j.csda.2017.04.005
0167-9473/© 2017 Published by Elsevier B.V.

was originally introduced for maximum likelihood inference in Ising models. Simulated tempering (Marinari and Parisi, 1992) shortly followed, and its connection to parallel tempering was developed in Geyer and Thompson (1995). A cousin of simulated tempering, the tempered transition algorithm, appeared in Neal (1996). Other important contributions to this field include multi-try Metropolis (Liu et al., 2000), evolutionary Monte Carlo (Liang and Wong, 2001), the equi-energy sampler (Kou et al., 2006), and more recently, the multiset sampler (Leman et al., 2009; Kim and MacEachern, 2015), and parallel hierarchical sampling (Rigat and Mira, 2012), among others. Usually these algorithms make sacrifices with respect to ease of implementation in order to improve performance. While increasing an algorithm's complexity might improve its ability to explore multimodal distributions, doing so typically adds an implementational burden on the researcher. In addition to a boost in performance, a significant advantage of weighted particle tempering is a substantial decrease in the required amount of tuning relative to its competitors.

Parallel tempering was one of the early solutions to the problem of sampling from multimodal distributions. The algorithm allows multiple particles to explore a ladder of tempered target distributions. Tempering flattens the landscape of a distribution so that bridges form between modes. Particles can traverse gaps between modes by visiting a rung on the ladder corresponding to a small tempering exponent (high temperature), then "cooling" back to the target distribution in a process analogous to metallurgic tempering. Mathematically, this is achieved by raising the target distribution to powers of the reciprocals of temperatures in a predefined sequence. The particles swap between adjacent steps on the ladder using Metropolis–Hastings acceptance rules.

One major complication with parallel tempering is the tuning of the temperature ladder, a nontrivial task that requires selecting the number of rungs on the ladder, the step size between rungs, and parameters for the proposal distributions at each of the ladder steps. Some guiding principles have been established to tune the temperature ladder (Kone and Kofke, 2005), while more recent methods have been proposed to adaptively optimize the ladder (Miasojedow et al., 2013).

Another issue with parallel tempering is that, despite its name, it is not a computationally parallel algorithm since its chains must communicate with each other at every step. The result is a substantial increase in computation and time, while only samples from the untempered chain are stored. Some efforts have been made to incorporate a full simulated or parallel tempering sample in approximating an expectation or functional. Of note in recent literature is a method known as importance tempering (Gramacy et al., 2010), where importance weights are computed for each sample from the target distribution in a simulated tempering run in order to increase the effective sample size used for the estimate. This method is similar to ours in that weights are computed for each sample, and rather than discarding the samples from tempered chains, they are allowed to (potentially) contribute to the final estimate. While importance tempering provides an overall improvement over naive simulated tempering and presumably over parallel tempering, its performance hinges on optimal tuning of the temperature ladder as noted by the authors.

Perhaps the most similar method to our own is parallel hierarchical sampling, which is a general way to utilize multiple Markov chains, taking advantage of their different mixing properties and reducing autocorrelation. Parallelization is complicated, however, by the communication among these chains at each step of the algorithm. We discuss how, in contrast, weighted particle tempering can be parallelized to take advantage of the increasingly affordable cost of computation.

In this work, we compare our proposed algorithm to parallel tempering and parallel hierarchical sampling because of their similarity to our method and their popularity. Both algorithms require tuning of several chains and their associated parameters, such as proposal standard deviations. Tuning weighted particle tempering is simplified as introducing more underlying chains always improves mixing regardless of choice of tuning parameters—of which there are only two. In Section 2, we define our notation and review parallel tempering and parallel hierarchical sampling. We define weighted particle tempering in Section 3, illustrate its performance through simulation studies in Section 4, and we provide a few case studies with Bayesian CART (Section 5.1) and Gaussian graphical models (Section 5.2). A proof of detailed balance is found in the Appendix. Finally, we wrap up with a brief discussion on limitations and what remains to be shown (Section 6).

2. Notation and existing algorithms

It will be useful to provide some general notation and definitions for the three algorithms we evaluate: parallel tempering, parallel hierarchical sampling, and weighted particle tempering. Each of these algorithms depends on *p underlying particles*, $u_{1,t}, \ldots, u_{p,t}$ that evolve over the iterations $t = 1, \ldots, N$ within their own respective *chains*. The final output of the algorithms is referred to as the *mother chain*, $x_1, \ldots, x_N$. In some cases it will be convenient to refer to the mother chain as the zeroth chain, $u_{0,1}, \ldots, u_{0,N}$. The $(p+1)$-tuple $\left\{x_t, u_{1,t}, \ldots, u_{p,t}\right\}$ is a Markov chain, but the individual particles are themselves non-Markovian.

We use π to denote the target distribution and π_ν with subscript $\nu \in [0, 1]$ to denote the normalized, *tempered* target. That is $\pi_\nu(x) = \frac{\pi^\nu(x)}{Z(\nu)}$, where

$$Z(\nu) = \int_\Omega \pi^\nu(x)dx < \infty \qquad (1)$$

is the normalizing constant and Ω is the sample space. In certain applications, tempering the target distribution gives a divergent integral in Eq. (1), so the requirement of a finite $Z(\nu)$ is necessary for π_ν to be a proper distribution and essential for parallel tempering and weighted particle tempering to work.