Outlining a Posterior-Approximating HMCMC Algorithm

May 26, 2017

1 Brief Update

We would like to design an MCMC algorithm that combines Hamiltonian MCMC (HMCMC) (Neal 2012; M. J. Betancourt et al. 2014; M. Betancourt 2017) and the approximate posterior + refinement approach of the Shrinking Bullseye (Conrad et al. 2015).

The basic skeleton of an HMCMC algorithm procedes as follows:

1. Consider the sampler at point $\vec{q}_n$ in parameter space with posterior density $\pi(\vec{q}|\vec{y})$ where $\vec{y}$ is the observed data (supressed going forward) .

2. Draw a momentum vector $\vec{p}_n$ from the distribution $\pi(\vec{p}|\vec{q}) = \mathrm{N}(\vec{p}|0, M)$ where M is the mass matrix (in vanilla HMCMC this is usually chosen to be the covariance matrix of $\pi(\vec{q}|\vec{y})$ but for Riemanian HMCMC I believe one uses the Hessian of $\pi(\vec{q})$ with respect to $\vec{q}$ evaluated at $\vec{q}_n$, or something similar).

3. We then numerically integrate the following Hamiltonian with step size h for s steps (the integration time sh is a free parameter for the sampler, and needs to be chosen wisely based on the geometry of the posterior), with initial conditions $(\vec{q}_n, \vec{p}_n)$:

$$\begin{aligned} \dot{\vec{q}} &= M^{-1}\vec{p} \\ \dot{\vec{p}} &= -\nabla ln(\pi(\vec{q})) \end{aligned} \tag{1}$$

4. This integration is typically done by the leapfrog method, but any other *symplectic* numerical integrator will work. This takes the form:

$$\begin{aligned} \vec{p}_{t+h/2} &= \vec{p}_t - (\frac{h}{2})\nabla ln(\pi(\vec{q_t})) \\ \vec{q}_h &= \vec{q}_t + (h)M^{-1}\vec{p}_{t+h/2} \\ \vec{p}_h &= \vec{p}_{t+h/2} - (\frac{h}{2})\nabla ln(\pi(\vec{q_{t+h}})) \end{aligned} \tag{2}$$

5. After integrating to time sh the candidate points q_{n+1}, p_{n+1} (supressing vector notation) are set as q_{n+sh}, p_{n+sh}. To make the process reversible we need to now negate p_{n+1}, so our candidate points are $q_{n+1}, -p_{n+1}$. These are then plugged into the usual Metropolis-Hastings accept/reject step. This would not be necessary if we could integrate (1) without error, but due to the error in (2) the accept/reject step is necessary to prevent bias in the final Monte Carlo estimates.

If we were to try and implement the above algorithm using the local regressions in Conrad et. al. (Conrad et al. 2015) we would hit a problem when trying to compute $\nabla ln(\pi(q))$. Letting $\hat{\pi}(q)$ denote the approximated/interpolated posterior, we note that when $\hat{\pi}$ is found using local approximations it is not necessarily continuous on the boundaries where the set of nearest neighbors changes, hence the derivative or gradient at any point along these boundaries is not defined/infinite.

If we're going to perform Hamiltonian MCMC on $\hat{\pi}$ we need an approximating algorithm that satisfies the following (loose) criteria:

1. Our interpolant must be at least once differentiable anywhere in the support of π (twice if we'd like to do Riemanian HMCMC). Furthermore it must be straightforward or cheap to evaluate $\nabla\hat{\pi}$ since we have to do so twice during a single evaulation of (2), and we are performing s such evaluations per step of the sampler so that's $2s$ gradients evaluations per candidate proposal.

2. The algorithm has to be amenable to refinements. Whatever interpolation scheme we use, it must allow us to test for refinements (ie. through something like the cross-validation approach in the Shrinking Bullseye), as well as add new points to the interpolating data set on an ad hoc basis (ie. whenever and *wherever* the refinement criteria is triggered).

3. The Hamiltonian dynamics induced by $\hat{\pi}$ need to be similar enough to π that the candidates proposed are actually good. Following Betancourt's "Conceptual Introduction to Hamiltonian Monte Carlo", our interpolant needs to approximate the true posterior well on its *typical set*.

As far as I am aware, any interpolating scheme that requires gridded data would have a hard time incorporating refinements in a straightforward way; I am under the impression that one would have to increase the grid resolution everywhere every time you add a refinement point which would quickly become expensive computationally. Therefore, based on these criteria, I think we should be looking at Radial Basis Functions (RBFs) as our interpolating scheme. These produce smooth interpolants on scattered data (ie. not on a grid), which meets criteria (1) and (2), and generally they work pretty well to approximate surfaces and their derivatives (working from the text Buhmann 2003 for error bounds). The derivative can be computed analytically in terms of the interpolating coefficients and functional form of the RBFs, so one doesn't need to bother with a taking a numerical gradient or anything.

I did find one other paper (haven't read it closely yet) where they use a neural network to approximate the posterior (Zhang, Shahbaba, and Zhao 2015). Not sure if that would work better than RBFs or not, I'd like to implement their algorithm down the line and run some head-to-head comparisons.

Currently I think the best RBF for this task is the Thin Plate Spline (TPS). The TPS allows us to penalize the wiggliness of the interpolant and I think this will produce the least amount of noise in the gradient of the log posterior, whereas another interpolation scheme might produce a lot of sharp peaks and valleys in the potential energy. This could really spoil the behavior of the numerical Hamiltonian integrator, which doesn't work well when there is a high degree of curvature along the energy contours (M. J. Betancourt et al. 2014). However, fitting a TPS requires working with square matrices of the dimension of your interpolating dataset, which can rapidly become expensive, especially if we keep adding refinement points. Therefore I am following the low-rank TPS approximation of

Wood (Wood 2003) which claims to reduce the fitting cost. Using TPSs the algorithm would be basically the same (except now using the interpolated potential energy/log posterior), however we would need to add a refinement-checking step and find a cheap way to update the interpolant with the new refinement.

I haven't investigated it as carefully yet, but I think we could also use a local regression approach to approximate both π *and* $\nabla\pi$. It's not clear to me how well this would perform, but currently my plan is to implement both the TPS interpolation and the local regression approach and compare the two on some test problems. I think that any such local scheme will run into continuity issues, so my priors don't expect this approach to work, but maybe it will.

I got in touch with Ian Grooms last week about this project, I think he has a good background (both stats and numerics) to give me advice on this project, plus I think he might see value in an algorithm that can sidestep expensive posteriors (I'm thinking some gigantic PDE weather model). Unforunately he has been at a conference this week so I plan to meet with him next week and go over go over my criteria and my understanding of RBFs.

2 Goals for Next Week

I've been reading through both Buhmann and Wood on RBFs and TPSs (respectively), but I still don't fully understand how they work. I am in the process of implementing Wood's low-rank approximation to the TPS interpolant, and while this is almost finished it has been pretty slow going. I hope to have that implementation up and running by the end of Monday, and once that's done it's pretty straightforward to run HMCMC using it, which will allow me to start playing around with refinement schemes and see what works best. By the end of next week I hope to have a complete description of this algorithm (instead of just the outline presented above), and some comparisons to the alternative schemes I mentioned above. Once this is done I can get started on proving ergodicity, convergence, etc. as well as checking for failure modes. Based on the Shrinking Bullseye paper I think this scheme can be thought of as basically an adaptive MCMC, so I can hopefully use some of the Roberts and Rosenthal Roberts and Rosenthal 2007 results to get convergence.

References

Betancourt, M. J. et al. (2014). "The Geometric Foundations of Hamiltonian Monte Carlo". In: *arXiv:1410.5110 [stat]*. arXiv: 1410.5110. URL: http://arxiv.org/abs/1410.5110 (visited on 05/26/2017).

Betancourt, Michael (2017). "A Conceptual Introduction to Hamiltonian Monte Carlo". In: *arXiv:1701.02434 [stat]*. arXiv: 1701.02434. URL: http://arxiv.org/abs/1701.02434 (visited on 05/26/2017).

Buhmann, Martin D. (2003). *Radial Basis Functions: Theory and Implementations*. Google-Books-ID: TRMf53opzlsC. Cambridge University Press. 271 pp. ISBN: 978-1-139-43524-6.

Conrad, Patrick R. et al. (2015). "Accelerating Asymptotically Exact MCMC for Computationally Intensive Models via Local Approximations". In: *arXiv*. ISSN: 0162-1459. URL: http://dspace.mit.edu/handle/1721.1/99937 (visited on 05/26/2017).

Neal, Radford M. (2012). "MCMC using Hamiltonian dynamics". In: *arXiv:1206.1901 [physics, stat]*. arXiv: 1206.1901. URL: http://arxiv.org/abs/1206.1901 (visited on 05/26/2017).

Roberts, Gareth O. and Jeffrey S. Rosenthal (2007). "Coupling and Ergodicity of Adaptive Markov Chain Monte Carlo Algorithms". In: *Journal of Applied Probability* 44.2, pp. 458–475. ISSN: 0021-9002, 1475-6072. DOI: 10.1017/S0021900200117954. URL: https://www.cambridge.org/core/journals/journal-of-applied-probability/article/coupling-and-ergodicity-of-adaptive-markov-chain-monte-carlo-algorithms/91713CBCA261758088FCDFBF8DB43FE0 (visited on 05/26/2017).

Wood, Simon N. (2003). "Thin plate regression splines". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 65.1, pp. 95–114. ISSN: 1467-9868. DOI: 10.1111/1467-9868.00374. URL: http://onlinelibrary.wiley.com/doi/10.1111/1467-9868.00374/abstract (visited on 05/26/2017).

Zhang, Cheng, Babak Shahbaba, and Hongkai Zhao (2015). "Hamiltonian Monte Carlo Acceleration Using Surrogate Functions with Random Bases". In: *arXiv:1506.05555 [stat]*. arXiv: 1506.05555. URL: http://arxiv.org/abs/1506.05555 (visited on 05/26/2017).