Outlining a Posterior-Approximating HMCMC Algorithm

June 14, 2017

1 Brief Intro

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 (see Appendix), but any other *symplectic* numerical integrator will work.
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 $q_{n+1}, -p_{n+1}$. Then the candidate q_{n+1} gets plugged into the usual Metropolis accept/reject probability. Doing a Metropolis accept/reject would not be necessary if we could integrate (1) without error, but due to the error in (2) we would get biased estimates without the Metropolis step.

2 Testing Approximation Methods

Based on my criteria (see Appendix) and conversation with Ian Grooms (see update from 6/2) I identified two approximating algorithms to compare: (1) Local Quadratic Regression (LQR) and (2) radial basis functions (specifically Thin Plate Splines, TPS). This are

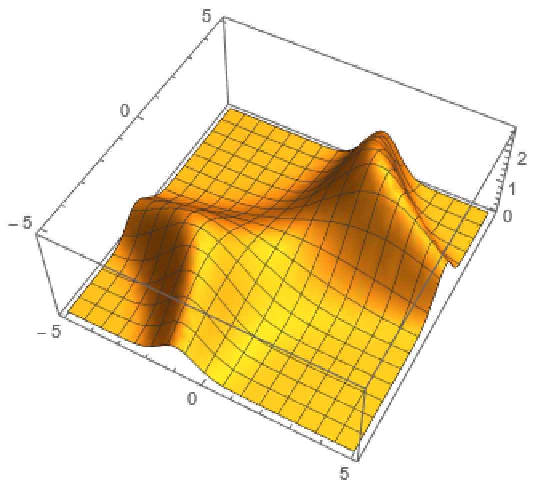


Figure 1: The test density function in (3) with $A = 1, B = 10, C = 1, P = 0.05$

both gridless methods, so are well-suited to ad hoc additions to the approximating set S. Here I implement both and compare their performance (the TPS was implemented with no low-rank approximation, as it reduced approximation quality substantially without providing a substantial computational speedup).

For comparison of performance I used the test density function:

$$\pi(x, y) = \mathrm{Exp}\left(-P * (A * x^2 * y^2 + x^2 + y^2 - B * x * y - C * x - C * y)\right) \tag{2}$$

With $A = C = 1$, $B = 10$ and $P = 0.05$ (Fig. 1). For gradient calculations I used the NumPy `gradient()` function, which computes the gradient by central differences. It should be noted that for both LQR and TPS the gradient can be computed by hand, but it was simpler in implementation to just use central differences for now. To compare the methods I fit them with 300 points chosen uniformly over the square $[-15, 15] \times [-15, 15]$ and computed the gradient at 1e5 gridpoints over the square $[-6, 6] \times [-6, 6]$. Letting $\hat{\pi}_i$ denote the approximation from method i, I computed the max gradient error as:

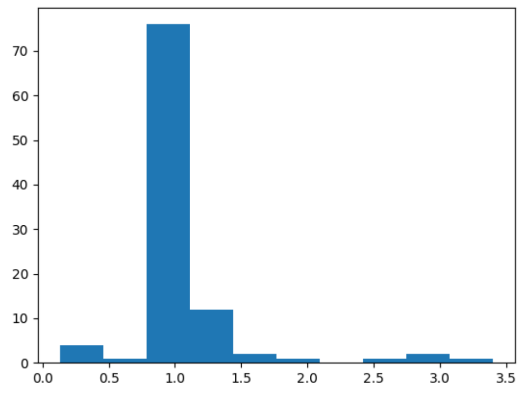
$$\epsilon_i = \frac{\max_j \left(\|\nabla\hat{\pi}_i(\vec{x}^{(j)}) - \nabla\pi(\vec{x}^{(j)})\|\right)}{\max_j \left(\|\nabla\pi(\vec{x}^{(j)})\|\right)}$$

Where the max's are taken over the grid points $\vec{x}^{(j)}$. I similarly computing a max function error:

$$e_i = \frac{\max_j \left(\|\hat{\pi}_i(\vec{x}^{(j)}) - \pi(\vec{x}^{(j)})\|\right)}{\max_j \left(\|\pi(\vec{x}^{(j)})\|\right)}$$

The thin plate splines fared substantially better, with $e_{\text{TPS}} = 0.319$ and $\epsilon_{\text{TPS}} = 0.826$ compared to $e_{\text{LQR}} = 0.596$ and $\epsilon_{LQR} = 57.9$. This was calculated with a TPS smoothness parameter $\lambda = 1e - 7$. To asses how varying this parameter changed the TPS fit I recomputed e_{TPS} and ϵ_{TPS} for each $\lambda \in [1e - 10, 1e - 7, 1e - 4, 1e - 1, 1, 10, 100]$. Virtually no effect of varying λ was detected on either measure of error, however, for all values of λ $e_{\text{TPS}} = 1.105$ and $\epsilon_{\text{TPS}} = 0.826$ with any differences being on the order of machine precision.

I also tested how the gradient error might effect Hamiltonian dynamics when numerically integrated with leapfrog. I chose a random initial, 2D position and momentum vector uniformly from the square $[-6, 6] \times [-6, 6]$, and then integrated forward for a random number of steps from 10 to 100, with fixed step size $h = 1e - 5$. I repeated


(a)

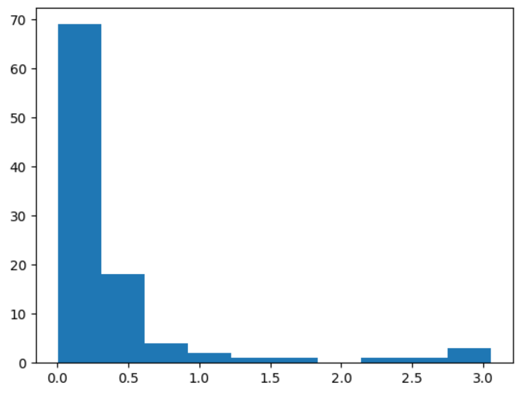

(b)

Figure 2: Histogram of path errors for (a) the LQR approximated potential and (b) the TPS approximated potential

this for 100 such initial conditions with the test density, as well as the LQR and TPS approximated densities. I calculated the path error for method i as $\rho_i = \frac{\sum_s \|q_i^{(s)} - q^{(s)}\|}{\sum_s \|q^{(s)} - q^{(s-1)}\|}$, $q_i^{(s)}$ is the s^{th} leapfrog step for $V(x)$ approximated by method i, and $q^{(s)}$ is same for the "true" $V(x)$. This is effectively the path integral of the leapfrog error normalized by the arc length of the true s-step forward integration. Normalizing by arc length doesn't give a sense of how the error might behave in the long-time, but intuitively one expects the error to behave fairly poorly in the long time anyways so I wasn't as interested in measuring it. My hope here is to instead get a sense of how badly the approximated potentials effect the Hamiltonian dynamics "on average"; if both methods behave similarly on average I may then dig into the long-time error to see if there's an important difference there.

Averaging over the random initial conditions and step sizes we get $E[\rho_{\text{LQR}}] = 1.09$ and $\text{Var}[\rho_{\text{LQR}}] = 0.193$, while for TPS we get $E[\rho_{\text{TPS}}] = 0.372$ and $\text{Var}[\rho_{\text{TPS}}] = 0.398$. The high variance of the TPS path error was surprising to me, although looking at the histograms of the observed error (Fig. 2) we see that the TPS approximated potential performed generally far better than the LQR approximation, however it appears that there were a small number of similarly bad initial conditions or step sizes, I'm guessing that these were a particularly sensistive IC for the Hamiltonians.

3 Goals for Next Week

- Diagnose and fix RBF interpolation scheme (may have to get back in touch with Ian for help with this)
- Add Greg's changes to the variance reduction paper, meet with him again on Friday to discuss them.

4 Appendix: Approximation Criteria

The key feature in the Shrinking Bullseye is the inclusion of the refinements to the set of samples S and $\pi(S)$ used for approximating the posterior density $\pi()$. Thinking about adapting their basic scheme (including the refinements) to HMCMC I identified the following criteria that an approximation method for $\pi()$ should satisfy to be a good candidate for the algorithm:

1. The approximation must be good for both π and $\nabla\pi$, and 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 $\nabla\pi$ 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 total that's $2s$ gradient evaluations per candidate proposal.

2. The approximation has to be amenable to refinements. Whatever interpolation scheme we use, it must allow us to test for refinements (ideally 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 the approximation 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.

5 Appendix: Leapfrog Method

One step in the integration of (1) using the leapfrog with step-size h takes the form:

$$\begin{aligned}\vec{p}_{t+h/2} &= \vec{p}_t - (\frac{h}{2})\nabla ln(\pi(\vec{q_t}))\\ \vec{q_h} &= \vec{q_t} + (h)M^{-1}\vec{p}_{t+h/2}\\ \vec{p_h} &= \vec{p}_{t+h/2} - (\frac{h}{2})\nabla ln(\pi(\vec{q_{t+h}}))\end{aligned} \tag{3}$$

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).