Adaptive Tuning Of Hamiltonian Monte Carlo Within Sequential Monte Carlo

Alexander Buchholz

MRC Biostatistics Unit

June 13, 2019

Credits

Acknowledgements

(a) Nicolas Chopin (ENSAE-CREST)

(b) Pierre E. Jacob (Harvard University, Dept. of Statistics)

Preprint available on Arxiv: 1808.07730

Monte Carlo sampling for Bayesian inference

- Observed data y
- **Interest**: Bayesian inference:

$$\pi(x) = \frac{l(y|x)p(x)}{Z}$$

- **Aim**: Sampling of $x \sim \pi$, calculation of Z
- **Problem**: Dimension of $x \in \mathbb{R}^d$ is large, normalizing constant difficult to calculate $Z = \int_{\mathbb{R}^d} l(y|x)p(x)dx$

Monte Carlo sampling for Bayesian inference

- **Our approach**: combine Sequential Monte Carlo (SMC) samplers [Del Moral et al., 2006] and Hamiltonian Monte Carlo (HMC) kernels [Neal, 2011]
- SMC samplers suitable for model choice as the normalizing constant is calculated on the fly
- HMC scales better with the dimension than other MCMC kernels
- **Problem**: HMC kernels are difficult to tune
- **Approach**: Use the information that is available through the cloud of particles for tuning the HMC kernels

A sequence of distributions

General idea of SMC samplers

- At $t = 1$ start with a simple distribution $\left\{x_t^i\right\}_{i \in 1:N} \sim p$ (e.g. the prior) and move the particles $\left\{x_t^i\right\}_{i \in 1:N}$ towards the distribution of interest π via a sequence of intermediate distributions

$$p = \pi_0, \cdots, \pi_t, \cdots, \pi_T = \pi$$

- In our Bayesian setting: tempering via a geometric bridge $\pi_t(x) \propto p(x) l(y|x)^{\lambda_t} = \gamma_t(x)$ and temperatures $0 = \lambda_0 < \cdots < \lambda_t < \cdots < \lambda_T = 1$

A sequence of distributions

Three steps in an SMC sampler: moving, weighting, resampling. Suppose $\left\{\tilde{x}_{t-1}^i\right\}_{i\in 1:N} \sim \pi_{t-1}$, then

1. Diversify particles: for each i,

$$x_t^i \sim \mathcal{K}_t(\tilde{x}_{t-1}^i, dx),$$

   where $\mathcal{K}_t$ is π_{t-1} invariant. This yields $\left\{x_t^i\right\}_{i\in 1:N} \sim \pi_{t-1}$.
2. Importance weight the particles: $w_t^i = \gamma_t(x_t^i)/\gamma_{t-1}(x_t^i)$. This yields a weighted set $\left\{x_t^i, w_t^i\right\}_{i\in 1:N}$ approximating π_t
3. Resample the particles according ot the weights: $\left\{\tilde{x}_t^i\right\}_{i\in 1:N} \sim \pi_t$

Note: $1/N \sum_{i=1}^N w_t^i \approx \frac{Z_t}{Z_{t-1}}$

A sequence of distributions

```
Algorithm 0: SMC sampler algorithm
Input: $\pi_0$ and $\pi_T$, Markov kernels $\mathcal{K}_t^h$, $\pi_{t-1}$ invariant.
Result: $\{x_t^i, w_t^i\}_{i\in 1:N}$ and $\widehat{Z_t/Z_{t-1}}$ for $t \in 1:T$.
Initialization: t = 1, $\lambda_0 = 0$;
1  foreach $i \in 1:N$ do
2      Sample $x_1^i \sim \pi_0$; Weight $w_1^i = \frac{\gamma_1(x_1^i)}{\pi_0(x_1^i)}$ ;
3  Calculate $\widehat{\frac{Z_1}{Z_0}} = N^{-1}\sum_{i=1}^N w_1^i$ ; resample $\{x_1^i, w_1^i\}_{i\in 1:N}$, get $\{\tilde{x}_1^i\}_{i\in 1:N}$;
4  Set $t = 2$;
   Iteration:
5  while $\lambda_t < 1$ do
6      foreach $i \in 1:N$ do
7          Move $x_t^i \sim \mathcal{K}_t^h(\tilde{x}_{t-1}^i, dx)$ ;
8          Weight particle $w_t^i = \frac{\gamma_t(x_t^i)}{\gamma_{t-1}(x_t^i)}$ ;
9      Calculate $\widehat{\frac{Z_t}{Z_{t-1}}} = N^{-1}\sum_{i=1}^N w_t^i$ ;
10     resample $\{x_t^i, w_t^i\}_{i\in 1:N}$ get $\{\tilde{x}_t^i\}_{i\in 1:N}$;
11     Set $t = t + 1$;
```

A sequence of distributions

Three important design choices to make

1. Choice of next temperature λ_t: based on effective sample size (ESS) [Kong et al., 1994]
2. Iteration of the number of move steps to assure proper mixing: based on autocorrelation of the kernel
3. Choice of tuning parameters of the kernel $\mathcal{K}_t$: based on [Fearnhead and Taylor, 2013] and another approach

Hamiltonian Monte Carlo

Exploiting gradient information for larger moves in the target space

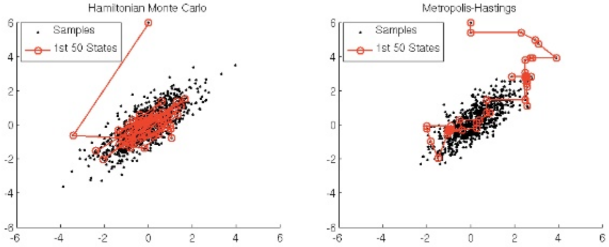


Figure: Exploration of a bivariate Normal: HMC and RWMH

- Based on the Hamiltonian

$$H(p,x) = -\log \mu(p,x) = -\underbrace{\mathcal{L}(x)}_{=\log \pi(x)} + \frac{1}{2} p^T \mathbf{M}^{-1} p.$$

Hamiltonian Monte Carlo

- Solving the equations of motions

$$\begin{cases} \frac{dx}{d\tau} = \frac{\partial H}{\partial p} = \mathbf{M}^{-1} p, \\ \frac{dp}{d\tau} = -\frac{\partial H}{\partial x} = \nabla_x \mathcal{L}(x), \end{cases}$$

- Via numerical leapfrog integrator over L steps

$$\begin{aligned} p_{\tau+\epsilon/2} &= p_\tau + \epsilon/2 \nabla_x \mathcal{L}(x_\tau), \\ x_{\tau+\epsilon} &= x_\tau + \epsilon \mathbf{M}^{-1} p_{\tau+\epsilon/2}, \\ p_{\tau+\epsilon} &= p_{\tau+\epsilon/2} + \epsilon/2 \nabla_x \mathcal{L}(x_{\tau+\epsilon}), \end{aligned}$$

- Tuning parameters: $\epsilon, L, \mathbf{M}$

Hamiltonian Monte Carlo

Algorithm 1: Hamiltonian Monte Carlo algorithm

```
Input: Gradient function $\nabla_x \mathcal{L}(\cdot)$, initial $x_s$, energy function
       $\Delta E = H(\cdot,\cdot) - H(p_s, x_s)$
Result: Next state of the chain $(p_{s+1}, x_{s+1})$
1 Sample $p_s \sim \mathcal{N}(0_d, \mathbf{M})$
2 Apply the leapfrog integration: $(\hat{p}_{s+1}, \hat{x}_{s+1}) \leftarrow \widehat{\Phi}_{\epsilon,L}(p_s, x_s)$
3 Sample $u \sim \mathcal{U}[0,1]$
4 if $\log(u) \leq \min(0, \Delta E_s)$ then
5     Set $(p_{s+1}, x_{s+1}) = (\hat{p}_{s+1}, \hat{x}_{s+1})$
6 else
7     Set $(p_{s+1}, x_{s+1}) = (p_s, x_s)$
```

Hamiltonian Monte Carlo

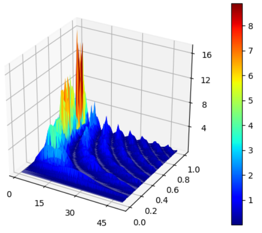
- How to choose ϵ, L?
- ϵ too large: trajectories become unstable
- L too large: waste of computation
- ϵ, L too short: bad exploration of target space
- Idea: maximize expected squared jumping distance [Pasarica and Gelman, 2010, Hoffman and Gelman, 2014]:

$$\mathrm{ESJD} = \mathbb{E}\left[\|x_s - x_{s-1}\|_2^2\right] = 2(1-\rho_1)\,\mathrm{Var}_\pi[x]$$

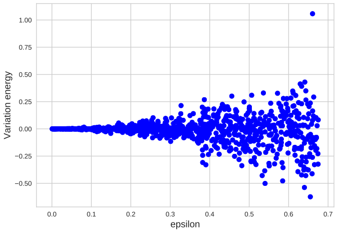
- A weighted version:

$$\tilde{\Lambda}(\hat{x}_s, x_{s-1}) = \frac{\|\hat{x}_s - x_{s-1}\|_M^2}{L} \times \min(1, \exp[\Delta E])$$

Hamiltonian Monte Carlo



(a)



(b)

Figure: Left: The normalized and weighted squared jumping distance (z-axis) as a function of ϵ (y-axis) and L (x-axis) for an isotropic Gaussian. Right: Variation of the difference in energy ΔE as a function of ϵ.

Tuning based on FT [Fearnhead and Taylor, 2013]

Tuning parameters as a weighted cloud of particles:

1. Assign different values of h_t^i according to their previous performance to the resampled particles $\tilde{x}_{t-1}^i$.
2. Propagate $x_t^i \sim \mathcal{K}_t^{h_t^i}(\tilde{x}_{t-1}^i, dx)$.
3. Evaluate the performance of h_t^i based on $x_t^i, \tilde{x}_{t-1}^i$.

Tuning based on FT [Fearnhead and Taylor, 2013]

Algorithm 2: (FT) Tuning of the HMC algorithm based on [Fearnhead and Taylor, 2013]

```
Input: Previous parameters $h^i_{t-1}$, estimator of associated
       utility $\tilde{\Lambda}(\tilde{x}^i_{t-2}, \hat{x}^i_{t-1})$, $i \in 1:N$, perturbation kernel $R$
Result: Sample of $h^i_t = (\epsilon^i_t, L^i_t)$, $i \in 1:N$
1 foreach $i \in 1:N$ do
2     Sample $h^i_t \sim \chi_t(h) \propto \sum_{i=1}^N \tilde{\Lambda}(\tilde{x}^i_{t-2}, \hat{x}^i_{t-1}) R(h; h^i_{t-1})$;
```

Tuning based on pretuning

1. Draw N test values $(\hat{\epsilon}, \hat{L})$ and assign these values to the different particles.
2. Apply the HMC flow with these parameters and randomly drawn momenta and record the performance corresponding to the assigned values.
3. Learn the dependence of the stability of the HMC flow on the values of ϵ.
4. Discard the initial HMC flows. Return to the starting point of the particle trajectories, redraw new momenta and assign the now weighted values (ϵ, L) to the particles.

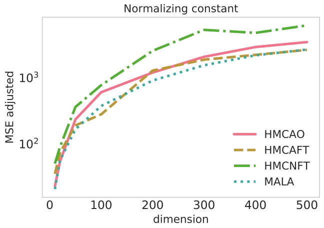
Tuning based on pretuning

Algorithm 3: (PR) Tuning of the HMC algorithm with pretuning

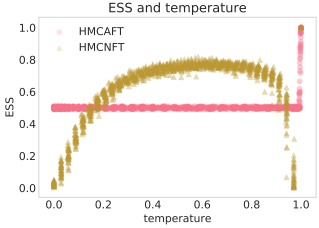
```
Input: Resampled particles $\tilde{x}_{t-1}^i$, $i \in 1:N$, HMC flow $\widehat{\Phi}_{\cdot,\cdot}$
       targeting $\pi_{t-1}$, $\epsilon_{t-1}^\star$
Result: Sample of $(\epsilon_t^i, L_t^i)$, $i \in 1:N$, upper bound $\epsilon_t^\star$
1 foreach $i \in 1:N$ do
2     Sample $\hat{\epsilon}_t^i \sim \mathcal{U}[0, \epsilon_{t-1}^\star]$ and $\hat{L}_t^i \sim \mathcal{U}\{1:L_{max}\}$;
3     Sample $p_t^i \sim \mathcal{N}(0_d, \mathbf{M}_{t-1})$;
4     Apply the leapfrog integration: $(\hat{p}_t^i, \hat{x}_t^i) \leftarrow \widehat{\Phi}_{\hat{\epsilon}_t^i, \hat{L}_t^i}(p_t^i, \tilde{x}_{t-1}^i)$;
5     Calculate $\Delta E_t^i$ and $\tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i)$
6 Calculate $\epsilon_t^\star$ based on a regression of $\Delta E_t^i$ on $\hat{\epsilon}_t^i$ $\forall i \in 1:N$;
7 Sample $(\epsilon_t^i, L_t^i) \sim \mathcal{C}at\left(w_t^i, \{\hat{\epsilon}_t^i, \hat{L}_t^i\}\right)$, where $w_t^i \propto \tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i)$
  $\forall i \in 1:N$;
```

Normal distribution

- Tempering from a normal to a shifted correlated normal
- Compare adaptation of temperature steps
- MALA outperforms HMC



(a)

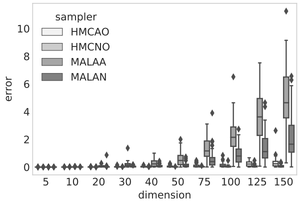


(b)

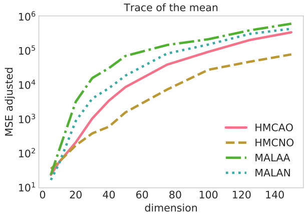
Figure: Adjusted mean squared error of the the normalization constant (Figure 4a) and ESS and temperature steps in dimension $d = 500$ (Figure 4b). Based on 40 repetitions of the sampler with $N = 1,024$ particles.

Student distribution

- Tempering from a t-student $\nu = 3$ to a shifted correlated t-student $\nu = 10$
- Compare adaptation of move steps



(a)

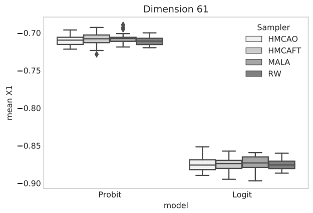


(b)

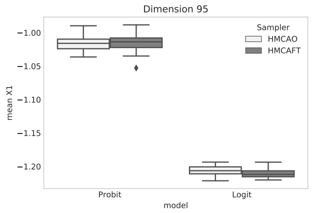
Figure: Figure 5a shows the squared error of the estimator of the normalizing constant. Figure 5b shows the squared error of the trace of the mean over different dimensions adjust for computation. The results are based on 100 runs of the samplers with $N = 1,024$ particles.

Bayesian binary Regression

- Model choice: Bayesian logit and probit in dimension 61 (sonar dataset) and 95 (Musk dataset)
- Compare to RW and MALA based samplers
- When adjusting for computation, HMC outperforms MALA and RW



(a)

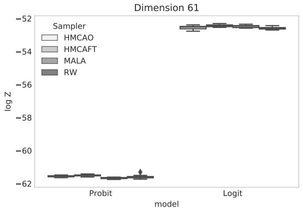


(b)

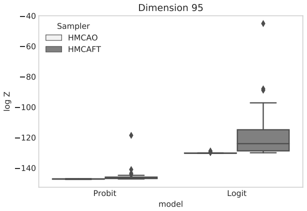
Figure: Estimated mean obtained for the probit and logit regression. Figure 6a corresponds to the sonar dataset. Figure 6b corresponds to the Musk dataset.

Bayesian binary Regression

- All samplers work reasonably well in dimension 61
- MALA and RW struggle in dimension 95 due to high correlation
- Pretuning important for estimation of the normalizing constant



(a)



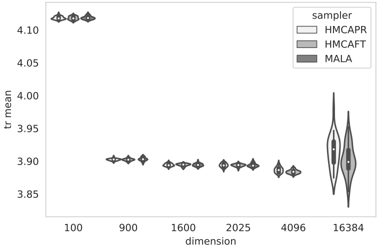
(b)

Figure: Normalization constants obtained for the probit and logit regression. Figure 7a corresponds to the normalization constants obtained for the sonar dataset. Figure 7b corresponds to the Musk dataset.

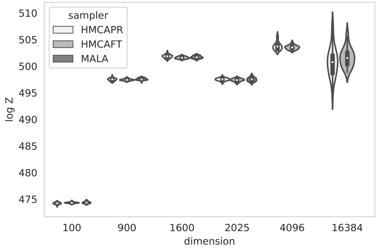
Log Gaussian Cox model

- Y is a Poisson process conditional on a Gaussian process X
- Aim: recover $X|Y$
- Applied to location of 126 pines (Finnish pines dataset)
- Dimension of target space depends on discretization of observed Y

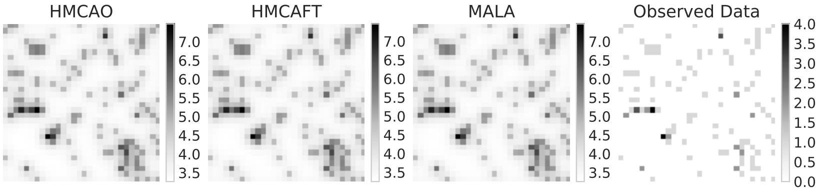
Log Gaussian Cox model



(a)



(b)



(c)

Conclusion

- Tuning of HMC samplers within SMC
- Two approaches: FT and pretuning
- Pretuning helpful in case of complicated distributions (high correlation)
- HMC outperforms MALA in high dimensions and when target is not to simple
- Cloud of particles helpful for adaptation of SMC samplers
- Model choice in high dimensions

Thanks for listening!

MRC Biostatistics Unit

UNIVERSITY OF CAMBRIDGE

alexander.buchholz@mrc-bsu.cam.ac.uk

www.mrc-bsu.cam.ac.uk

References (1)

Del Moral, P., Doucet, A., and Jasra, A. (2006).
Sequential Monte Carlo samplers.
Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68(3):411–436.

Fearnhead, P. and Taylor, B. M. (2013).
An adaptive sequential Monte Carlo sampler.
Bayesian Analysis, 8(2):411–438.

Hoffman, M. D. and Gelman, A. (2014).
The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.
Journal of Machine Learning Research, 15(1):1593–1623.

References (2)

Kong, A., Liu, J. S., and Wong, W. H. (1994).
Sequential imputation and Bayesian missing data problems.
Journal of the American statistical association, 89:278–288.

Neal, R. M. (2011).
MCMC using Hamiltonian dynamics.
Handbook of Markov Chain Monte Carlo, 2(11).

Pasarica, C. and Gelman, A. (2010).
Adaptively scaling the Metropolis algorithm using expected squared jumped distance.
Statistica Sinica, pages 343–364.