Adaptive Hamiltonian and Riemann Manifold Monte Carlo Samplers

Ziyu Wang ZIYUW@CS.UBC.CA
Shakir Mohamed SHAKIRM@CS.UBC.CA
Nando de Freitas NANDO@CS.UBC.CA
Department of Computer Science, University of British Columbia, Vancouver, BC, Canada

Abstract

In this paper we address the widely-experienced difficulty in tuning Monte Carlo sampler based on simulating Hamiltonian dynamics. We develop an algorithm that allows for the adaptation of Hamiltonian and Riemann manifold Hamiltonian Monte Carlo samplers using Bayesian optimization that allows for infinite adaptation of the parameters of these samplers. We show that the resulting samplers are ergodic, and that the use of our adaptive algorithms makes it easy to obtain more efficient samplers, in some cases precluding the need for more complex solutions. Hamiltonian-based Monte Carlo samplers are widely known to be an excellent choice of MCMC method, and we aim with this paper to remove a key obstacle towards the more widespread use of these samplers in practice.

1. Introduction

Hamiltonian Monte Carlo (HMC) (Duane et al., 1987) is widely-known as a powerful and efficient sampling algorithm, having been demonstrated to outperform many existing MCMC algorithms, especially in problems with high-dimensional, continuous, and correlated distributions (Chen et al., 2001; Neal, 2010). Despite this flexibility, HMC has not been widely adopted in practice, due principally to the sensitivity and difficulty of tuning its hyperparameters. Tuning HMC has been reported by many experts to be more difficult than tuning other MCMC methods (Ishwaran, 1999; Neal, 2010). In this paper we aim to remove this obstacle in the use of HMC by providing an automated method for determining these tunable parameters, paving the way for a more widespread application of HMC in statistics and machine learning.

There are few existing works dealing with the automated tuning of HMC. Two notable approaches are: the No U-turn sampler (NUTS) (Hoffman & Gelman, 2011), is an adaptive algorithm that aims to find the best parameter settings by tracking the sample path and preventing HMC from retracing its steps in this path; and Riemann manifold HMC (RMHMC) (Girolami & Calderhead, 2011), which provides adaptations using Riemannian geometry.

In this paper, we follow the approach of *adapting* Markov chains in order to improve the convergence of both HMC and RMHMC. Our adaptive strategy is based on Bayesian optimization; see for example Brochu et al. (2009) and Snoek et al. (2012) for a comprehensive introduction to Bayesian optimization. Bayesian optimization has been proposed previously for the adaptation of general MCMC samplers by Mahendran et al. (2012) and Hamze et al. (2013). To guarantee convergence, these works were limited to a finite adaptation of the Markov chain. This finite adaptation can result in the sampler being trapped in suboptimal parameter settings, leading to inefficient sampling, and we address this limitation in this paper.

We describe Hamiltonian-based Monte Carlo samplers (sect. 2), and then make the following contributions:

- We present an algorithm for adaptive HMC in which we allow for *infinite adaptation* of the Markov chain, thus avoiding parameter traps due to finite adaptation (section 3).
- Importantly, we prove that the adaptive MCMC samplers we present are *ergodic* in this infinite adaptation setting (section 4).
- We provide a comprehensive set of experiments demonstrating that the adaptive schemes perform better across a diverse set of problems (section 5).
- We use a version of the *expected squared jumping distance* (Pasarica & Gelman, 2010) as the objective function for adaptation. In section 5, we also introduce a new approach based on predictive measures, for use in settings where it is possible to perform cross-validation.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

2. Hamiltonian-based Monte Carlo Sampling

Hamiltonian (or Hybrid) Monte Carlo (Duane et al., 1987; Neal, 2010), has become established as a powerful, general purpose Markov chain Monte Carlo (MCMC) algorithm for sampling from general, continuous distributions. Its efficiency is due to the fact that it makes use of gradient information from the target density to allow for an ergodic Markov chain capable of large transitions that are accepted with high probability. This efficiency and flexibility is demonstrated by the wide range of applications to which HMC has been applied, including: Bayesian generalized linear models (Ishwaran, 1999), Bayesian neural networks (Neal & Zhang, 2006), Gaussian process regression and classification (Rasmussen & Williams, 2006), exponential family PCA and factor analysis (Mohamed et al., 2008), and restricted Boltzmann machines (Ranzato & Hinton, 2010), amongst others.

For HMC, we are required to specify a potential energy function, which is the log of the joint distribution we wish to sample from, $U(\mathbf{x}) = -\log p(\mathbf{x})$ and a kinetic energy function, most typically, $K(\mathbf{p}) = \mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}/2$, with momentum vector $\mathbf{p}$ and a positive definite mass matrix $\mathbf{M}$. For standard HMC, the mass matrix is set to the identity. We defer the technical details of HMC to existing work (Neal, 2010), and present only the algorithm here (Alg. 1).

HMC requires the selection of two free parameters: a step-size ϵ and a number leapfrog steps L. The accepted guidance is to choose the step-size to ensure that the sampler's rejection rate is between 25%-35%. It is also preferable to have a large L, since this reduces the random walk behavior of the sampler (Neal, 2010), but too large an L results in unnecessary computation. In this paper, we consider a slight variation of the HMC algorithm: instead of performing L leapfrog steps at each iteration, we perform a random number of leapfrog steps, chosen from the discrete uniform distribution over $\{1, \cdots, L\}$, i.e. $L_r \sim \mathcal{U}(1, L)$ steps. This approach amounts to using a mixture of L different HMC transition kernels, thus preserving detailed balance (Andrieu et al., 2003).

HMC is known to be highly sensitive to the choice of ϵ and L. We demonstrate HMC's sensitivity to these parameters by sampling from a bivariate Gaussian with correlation coefficient 0.99. We consider three settings $(\epsilon, L) = \{(0.16, 40), (0.16, 50), (0.15, 50)\}$ and show the behavior of the sampler as well as the autocorrelation plot in figure 1. While the first setting exhibits good behavior and low auto-correlation, small changes to these settings results in poor mixing and high autocorrelation, as seen in the other graphs. Theoretical results concerning the optimal acceptance rate for HMC exist, having been described by Beskos et al. (2010) and Neal (2010), with both concluding a rate around 0.65 as optimal. Such results, however, would not help in choosing the best sampler out of the three in Figure 1, since all three samplers in this demonstration have an acceptance rate around 0.7, leaving little guidance for finding the most efficient sampler.

Algorithm 1 Hamiltonian Monte Carlo Algorithm

Given: M, L, ϵ, and $\mathbf{x}^1$.
for $t = 1, 2, \cdots$ **do**
  Sample $\mathbf{p}^t \sim \mathcal{N}(\mathbf{0}, M)$ and $L_r \sim \mathcal{U}(1, L)$
  Let $\mathbf{x}_0 = \mathbf{x}^t$ and $\mathbf{p}_0 = \mathbf{p}^t + \frac{\epsilon}{2} \frac{\partial U}{\partial \mathbf{x}}\big|_{\mathbf{x}_0}$
  for $l = 1, 2, \cdots, L_r$ **do**
    $\mathbf{x}_l = \mathbf{x}_{l-1} + \epsilon M^{-1}\mathbf{p}_{l-1}$
    $\mathbf{p}_l = \mathbf{p}_{l-1} + \epsilon \frac{\partial U}{\partial \mathbf{x}}\big|_{\mathbf{x}_l}$
  end for
  $\mathbf{p}_l = \mathbf{p}_{l-1} - \frac{\epsilon}{2} \frac{\partial U}{\partial \mathbf{x}}\big|_{\mathbf{x}_l}$
  Draw $\mathbf{u} \sim \mathcal{U}(0, 1)$
  if $\mathbf{u} < \min[1, e^{U(\mathbf{x}^t) + K(\mathbf{p}^t) - U(\mathbf{x}_l) - K(\mathbf{p}_l)}]$ **then**
    Let $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1}) = (\mathbf{x}_l, \mathbf{p}_l)$
  else
    Let $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1}) = (\mathbf{x}^t, \mathbf{p}^t)$
  end if
end for

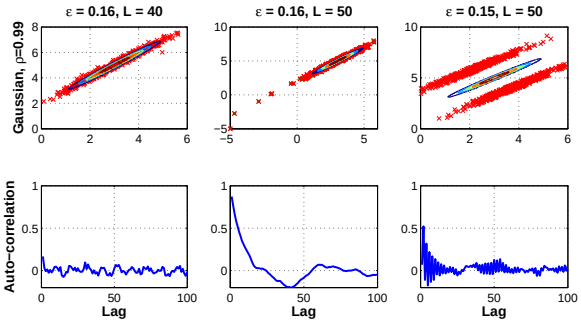

Figure 1. 1000 samples from a bivariate Gaussian distribution generated using HMC. We show the trajectory and auto-correlation of the samples for 3 parameter settings.

To address the problem of choosing these parameters, we will introduce a method for automatically and adaptively tuning the parameters of HMC, reducing the need for time-consuming, expert tuning. An existing approach for automatic tuning of HMC was introduced by Hoffman & Gelman (2011), referred to as the No U-turn sampler (NUTS). NUTS allows for automatic tuning of both HMC's parameters by tuning the stepsize ϵ during the burn-in phase, after which it is fixed and the number of leapfrog steps is adjusted thereafter for every iteration. ϵ is chosen using a stochastic approximation method referred to as dual averaging, and L is chosen for every sample using a recursive algorithm in which the number of leapfrog steps is allowed to increase until the proposal trajec-

tory taken by the sampler begins to move back towards its initial point, thus preventing U-turns and allowing for the good mixing of the chain.

Riemann manifold HMC (RMHMC) (Girolami & Calderhead, 2011) is a sampling method derived from HMC, and provides an adaptation mechanism by exploiting the Riemannian geometry of the parameter space. Rather than adapting ϵ and L, RMHMC accounts for the local structure of the joint density by adapting the mass matrix $\mathbf{M}$ used in HMC. Since RMHMC automatically adapts its mass matrix, the stepsize ϵ is usually fixed and the number of leapfrog steps L, which is a single scalar, can be chosen using the rejection rate. While the sensitivity to these parameters is greatly reduced, they must still be set and there is no general guidance on how these parameters should be chosen, making it desirable to have a fully automatic method for RMHMC as well.

3. Adaptive Hamiltonian Monte Carlo

In order to adapt the MCMC parameters L and ϵ for HMC, we need to (i) define an objective function and (ii) choose a suitable optimization method.

As pointed out in Pasarica & Gelman (2010), a natural objective function for adaptation is the asymptotic efficiency of an MCMC sampler, $(1 + 2\sum_{k=1}^{\infty} \rho_k)^{-1}$, where ρ_k is the auto-correlation of the sampler with lag k. Despite its appeal, this measure is problematic because the higher order auto-correlations are hard to estimate. To circumvent this problem, Pasarica & Gelman (2010) introduced an objective measure called the expected squared jumping distance (ESJD):

$$\text{ESJD}(\boldsymbol{\gamma}) = \mathbb{E}_{\boldsymbol{\gamma}} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2,$$

where $\boldsymbol{\gamma} = (L, \epsilon)$ denotes the set of parameters for HMC. Maximizing the above objective is equivalent to minimizing the first-order auto-correlation ρ_1. In practice, the above intractable expectation with respect to the Markov chain is approximated by an empirical estimator, as outlined in Pasarica & Gelman (2010).

The ESJD measure is efficient in situations where the higher order auto-correlations increase monotonically with respect to ρ_1. However, it is not suitable for tuning HMC samplers since by increasing the number of leapfrog steps one can almost always generate better samples. What we need is a measure that also takes computing time into consideration. With this goal in mind, we introduce the following objective function:

$$f(\gamma) = \frac{\text{ESJD}(\boldsymbol{\gamma})}{\sqrt{L}} = \frac{\mathbb{E}_{\boldsymbol{\gamma}} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2}{\sqrt{L}}.$$

This function simply normalizes the ESJD by the number of leapfrog steps L, thus taking both statistical efficiency and computation into consideration. Most of our experiments will use this measure as we have found it to work very well in practice. Many works in the adaptive MCMC literature have considered matching empirical and theoretical acceptance rates in order to adapt MCMC samplers; see for example Andrieu & Robert (2001) or Vihola (2010). We have found this strategy to perform poorly in the case of HMC, where samplers with the same acceptance rate can exhibit different mixing behavior (figure 1). When discussing Bayesian neural networks in our experiments (section 5.4), we will introduce an alternative objective function based on predictive performance. Such a measure does however only apply in predictive domains and is, consequently, less general than the normalized ESJD objective.

Now that we are armed with an objective function, we need to address the issue of optimization. Since the objective is only available point-wise (that is, it can be evaluated but its exact form is intractable), researchers typically use stochastic approximation. We use Bayesian optimization to optimize the objective. A discussion contrasting these two alternatives is presented in Hamze et al. (2013).

Bayesian optimization is an efficient gradient-free optimization tool well suited for expensive black box functions. Our objective function (normalized ESJD) is of this nature. As mentioned earlier, normalized ESJD involves an intractable expectation that can be approximated by sample averages, where the samples are produced by running HMC for a few iterations. Each set of HMC samples for a specific set of hyper-parameters $\boldsymbol{\gamma} \in \Gamma$ results in a noisy evaluation of the normalized ESJD: $r(\boldsymbol{\gamma}) = f(\boldsymbol{\gamma}) + \varepsilon$, where we assume that the measurement noise is Gaussian $\varepsilon \sim \mathcal{N}(0, \sigma_\eta^2)$.

Following the standard Bayesian optimization methodology, we set Γ to be a box constraint such that

$$\Gamma = \{(\epsilon, L) : \epsilon \in [b_l^\epsilon, b_u^\epsilon], L \in [b_l^L, b_u^L]\}$$

for some interval boundaries $b_l^\epsilon \leq b_u^\epsilon$ and $b_l^L \leq b_u^L$. The parameter L is discrete. The parameter ϵ is continuous, but since it is one-dimensional, we can discretize it using a very fine grid.

Since the true objective function is unknown, we specify a zero-mean Gaussian prior over it: $f(\cdot) \sim GP(0, k(\cdot, \cdot))$, where $k(\cdot, \cdot)$ is the covariance function, which forms the covariance matrix:

$$\mathbf{K} = \begin{bmatrix} k(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_1) & \dots & k(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_i) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_1) & \dots & k(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i) \end{bmatrix},$$

and $\mathbf{k} = [k(\boldsymbol{\gamma}, \boldsymbol{\gamma}_1) \quad \dots \quad k(\boldsymbol{\gamma}, \boldsymbol{\gamma}_i)]^T$. In this

Algorithm 2 Adaptive HMC.

Given: Γ, m, k, α, and $\boldsymbol{\gamma}_1$.
for $i = 1, 2, \ldots,$ **do**
  Run HMC for m iterations with $\boldsymbol{\gamma}_i = (\epsilon_i, L_i)$.
  Obtain the objective function value r_i using the drawn samples.
  Augment the data $\mathcal{D}_i = \{\mathcal{D}_{i-1}, (\boldsymbol{\gamma}_i, r_i)\}$.
  if $r_i > \sup_{j \in \{1,\cdots,i-1\}} r_j$ **then**
    $s = \frac{\alpha}{r_i}$
  end if
  Draw $u \sim \mathcal{U}([0, 1])$
  let $p_i = (\max\{i - k + 1, 1\})^{-0.5}$, with $k \in \mathbb{N}^+$.
  if $u < p_i$ **then**
    $\boldsymbol{\gamma}_{i+1} := \arg\max_{\boldsymbol{\gamma} \in \Gamma} u(\boldsymbol{\gamma}, s|\mathcal{D}_i)$.
  else
    $\boldsymbol{\gamma}_{i+1} := \boldsymbol{\gamma}_i$
  end if
end for

work, we adopt a Gaussian ARD covariance function with $k(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_j) = \exp(-\frac{1}{2}\boldsymbol{\gamma}_i^T \Sigma^{-1} \boldsymbol{\gamma}_j)$ where Σ is a positive definite matrix. We set $\Sigma = \text{diag}\left(\left[\kappa(b_u^\epsilon - b_l^\epsilon)\right]^2; \left[\kappa(b_u^L - b_l^L)\right]^2\right)$, where $\kappa = 0.2$.

Given noisy evaluations of the objective function $\{r_k\}_{k=1}^i$ where evaluated at points $\{\boldsymbol{\gamma}_k\}_{k=1}^i$, we form the dataset $\mathcal{D}_i = \left(\{\boldsymbol{\gamma}_k\}_{k=1}^i, \{\mathbf{r}_k\}_{k=1}^i\right)$. Using Bayes rule, we arrive at the posterior predictive distribution over the unknown objective function:

$$
\begin{aligned}
f|\mathcal{D}_i, \boldsymbol{\gamma} &\sim \mathcal{N}(\mu_i(\boldsymbol{\gamma}), \sigma_i^2(\boldsymbol{\gamma})) \\
\mu_i(\boldsymbol{\gamma}) &= \mathbf{k}^T (\mathbf{K} + \sigma_\eta^2 \mathbf{I})^{-1} \mathbf{r}_i \\
\sigma_i^2(\boldsymbol{\gamma}) &= k(\boldsymbol{\gamma}, \boldsymbol{\gamma}) - \mathbf{k}^T (\mathbf{K} + \sigma_\eta^2 \mathbf{I})^{-1} \mathbf{k}
\end{aligned}
$$

For more details on Gaussian processes, please refer to Rasmussen & Williams (2006).

The Gaussian process simply provides a surrogate model for the true objective. The surrogate can be used to efficiently search for the maximum of the objective function. In particular, it enables us to construct an acquisition function $u(\cdot)$ that tells us which parameters $\boldsymbol{\gamma}$ to try next. The acquisition function uses the Gaussian process posterior mean to predict regions of potentially higher objective values (exploitation). It also uses the posterior variance to detect regions of high uncertainty (exploration). Moreover, it effectively trades-off exploration and exploitation. Different acquisition functions have been proposed in the literature (Močkus, 1982; Srinivas et al., 2010; Hoffman et al., 2011). We adopt a variant of the Upper Confidence Bound (UCB) (Srinivas et al., 2010), modified to suit our application:

$$
u(\boldsymbol{\gamma}, s|\mathcal{D}_i) = \mu_i(\boldsymbol{\gamma}, s) + p_i \beta_{i+1}^{\frac{1}{2}} \sigma_i(\boldsymbol{\gamma}).
$$

As in standard UCB, we set $\beta_{i+1} = 2\log\left(\frac{(i+1)^{\frac{d}{2}+2}\pi^2}{3\delta}\right)$, where d is the dimension of Γ and δ is set to 0.1. The parameter p_i ensures that the diminishing adaptation condition for adaptive MCMC (Roberts & Rosenthal, 2007) is satisfied. Specifically, we set $p_i = (\max\{i - k + 1, 1\})^{-0.5}$ for some $k \in \mathbb{N}^+$. As p_i goes to 0, the probability of Bayesian optimization adapting $\boldsymbol{\gamma}$ vanishes as shown in Algorithm 2.

It could be argued that this acquisition function could lead to premature exploitation, which may prevent Bayesian optimization from locating the true optimum of the objective function. There is some truth to this argument. Our goal when adapting the Markov chain, however, is less about finding the absolute best hyper-parameters but more about finding sufficiently good hyper-parameters given finite computational resources. Given enough time, we could slow the annealing schedule thus allowing Bayesian optimization to explore the hyper-parameter space fully. However, under time constraints we must use faster annealing schedules. As p_i decreases, it becomes increasingly difficult for Bayesian optimization to propose new hyper-parameters for HMC. Consequently, the sampler ends up using the same set of hyper-parameters for many iterations. With this in mind, we argue, it is more reasonable to exploit known good hyper-parameters rather than exploring for better ones. This intuition matches our experience when conducting experiments.

The acquisition function also includes a scalar scale-invariance parameter s, such that $\mu_i(\boldsymbol{\gamma}, s) = \mathbf{k}^T (\mathbf{K} + \sigma_\eta^2 \mathbf{I})^{-1} \mathbf{r}_i s$. This parameter is estimated automatically so as to rescale the rewards to the same range each time we encounter a new maximal reward.

Gaussian processes require the inversion of the covariance matrix and, hence, have complexity $\mathcal{O}(i^3)$, where i is the number of iterations. Fortunately, thanks to our annealing schedule, the number of unique points in our Gaussian process grows sub-linearly with the number of iterations. This slow growth makes it possible to adopt kernel specification techniques, as proposed by Engel (2005), to drastically reduce the computational cost without suffering any loss in accuracy.

Even our adaptive strategy has hyperparameters that must be set. Of all these, the length scale of the kernel is the most important. In all our experiments, we set $\alpha = 4$, $k = 100$, $m = \frac{B}{k}$, where B is the number of burn-in samples. In our experience, our algorithm is highly robust to these settings and we *use the same parameter settings for all of our experiments*, with the exception of Γ. Γ is easy to set, since one can choose the bound to be large enough to contain all reasonable ϵ and L, while allowing the adaptive algorithm

enough time to explore. Alternatively, one could gauge the hardness of the sampling problem at hand and set more reasonable bounds. In general, harder sampling problems require a smaller ϵ and a larger L. We follow this second strategy throughout our experiments and found that most sensible bounds led to performance similar to the ones reported.

4. Analysis of Convergence

The proof of ergodicity of the adaptive HMC algorithm capitalizes on existing results for Langevin diffusions and adaptive MCMC on compact state spaces. The method of proof is based on the standard Lyapunov stability functions, also known as drift or potential functions.

We assume that our target distribution is compactly supported on $\mathcal{M}$. In practice, for target distributions that are not compactly supported, we could set $\mathcal{M}$ large enough to contain most of the mass of our target distribution. The sampler is restricted to $\mathcal{M}$ by following this standard approach of rejecting all proposals that fall outside $\mathcal{M}$.

Let $\{P_\gamma\}_{\gamma\in\Gamma}$ be a collection of Markov chain kernels, each admitting π as the stationary distribution. That is, for each value of $\gamma = (\epsilon, L)$, we have one such kernel. Moreover, let P_γ^n denote the n-step Markov kernel. Our proof requires the following classical definitions:

Definition 1. (**Small set**) *A subset of the state space $C \subseteq \mathcal{X}$ is small if there exists $n_0 \in \mathbb{N}^+$, $\xi > 0$ and a probability measure $\nu(.)$ such that $P^{n_0}(x,\cdot) \geq \xi\nu(\cdot)$ $\forall x \in C$.*

Definition 2. (**Drift condition**) *A Markov chain satisfies the drift condition if for a small set C, there exist constants $0 < \lambda < 1$ and $b < \infty$, and a function $V : \mathcal{X} \to [1, \infty]$ such that $\forall x \in \mathcal{X}$*

$$\int_{\mathcal{X}} P(x, dy)V(y) \leq \lambda V(x) + b\mathbf{1}_C(x).$$

Having defined the necessary concepts, we now move on to show the ergodicity of our adapted approach.

Proposition 3. *Suppose that P_γ, when restricted to a compact set $\mathcal{M}$, admits the stationary distribution π for all $\gamma \in \Gamma$. If π is continuous, positive and bounded on $\mathcal{M}$, and $|\Gamma|$ is finite, then the adaptive HMC sampler is ergodic.*

Proof. To show that adaptive HMC converges on a compact set, we first show that $\mathcal{M}$ is a small set.

The transition kernel of the random time HMC algorithm can be written as $P_\gamma(x,.) = \sum_{l=1}^{L} \frac{1}{L} Q_{l,\epsilon}(x,.)$ where $Q_{l,\epsilon}(x,.)$ is the transition kernel of an HMC sampler that takes l leapfrog steps with parameter ϵ. In particular $Q_{1,\epsilon}(x,.)$ is the transition kernel of Metropolis adjusted Langevin algorithm (MALA). As π is bounded, and the proposal distribution of MALA is positive every where, we have that $Q_{1,\epsilon}$ is μ^{Leb}-irreducible. By a slight modification of Theorem 2-2 in Roberts & Tweedie (1996), for Markov chains defined by MALA, and any compact set C with positive Lebesgue measure (i.e. $\mu^{Leb}(C) > 0$) there exists $\xi > 0$ and a probability measure $\nu(\cdot)$ such that $\forall x \in C$ $Q_{1,\epsilon}^1(x,.) \geq \xi\nu(.)$. Hence, $\mathcal{M}$ is a small set since

$$P_\gamma^1(x,.) \geq \frac{1}{L} Q_{1,\epsilon}^1(x,.) \geq \frac{1}{L}\xi\nu(.)$$

for any compact set C where $\mu^{Leb}(C) > 0$. The drift condition is trivially satisfied by each HMC sampler when we choose C to be $\mathcal{M}$, and V to be such that $V(x) = 1$ for all x.

Having proved these conditions, we can now appeal to Theorem 15.0.1 of Meyn & Tweedie (1993) to conclude that $\|P_\gamma^n(x,\cdot) - \pi(\cdot)\| < R_\gamma V(x)\rho_\gamma^n$ for all n and for $0 < \rho_\gamma < 1$. Since $V(X) = 1$ $\forall x$, we have

$$\|P_\gamma^n(x,\cdot) - \pi(\cdot)\| < R_\gamma \rho_\gamma^n.$$

Define $R_{max} = \sup_{\gamma\in\Gamma} R_\gamma$ and $\rho_{max} = \sup_{\gamma\in\Gamma} \rho_\gamma$, then $\forall x \in \mathcal{M}$ and $\forall \gamma \in \Gamma$ we have

$$\|P_\gamma^n(x,\cdot) - \pi(\cdot)\| < R_{max}\rho_{max}^n.$$

We have shown that the kernels $\{P_\gamma(x,\cdot)\}_{\gamma\in\Gamma}$ are simultaneously uniformly ergodic. Also, the adaptive HMC sampler has diminishing adaptation by design. By Theorem 5 of Roberts & Rosenthal (2007), these two conditions imply the claim of our proposition. □

In general two sets of conditions together guarantee ergodicity of an adaptive MCMC algorithm (Roberts & Rosenthal, 2007; Atchadé & Fort, 2010). First, the adaptation has to diminish eventually. The second set of conditions is usually placed on the underlying MCMC samplers. In Roberts & Rosenthal (2007), the samplers are required to be either simultaneously uniformly or geometrically ergodic. Without restricting the state space to be compact, it is unlikely that HMC is uniformly ergodic. Also, to the best of our knowledge, no theoretical results exist on the geometric ergodicity of HMC when the state space is not compact. However, Roberts & Stramer (2002) showed that Langevin diffusion, which is closely related to HMC, is geometrically ergodic. Thus one potential challenge would be to prove or disprove geometric ergodicity of HMC in general state spaces. Atchadé & Fort (2010) weakened the conditions required, still requiring diminishing adaptation, but the requirements

on the underlying MCMC samplers were reduced to sub-geometric ergodicity. Although these conditions are weaker, it remains hard to check whether HMC satisfies them.

5. Results

We show the performance of our adaptive algorithm on four widely-used models. We evaluate the performance of the samplers using the effective sample size (ESS) using: $ESS = R\,(1 + 2\sum_k \rho_k)$, where R is the number of posterior samples, and $\sum_k \rho_k$ is the sum of K monotone sample auto-correlations computed using the monotone sequence estimator (Girolami & Calderhead, 2011). We adopt the total number of leapfrog steps used in producing the set of samples as a proxy for computational demand, since the computation is dominated by the gradient evaluation required for each leapfrog step. An efficient sampler will result in the highest ESS for the least computation, and we will thus report the effective sample size per leapfrog step used (ESS/L), similarly to Hoffman & Gelman (2011), since this takes into account computational requirements. We compute the ESS/L over all dimensions of the target distribution and report the minimum, median and maximum ESS obtained. While we report all three summary statistics, we focus on the *minimum ESS/L* as the most useful measure, since this allows us to evaluate the efficiency of the most confined coordinate, and is more indicative of ESS jointly over all coordinates rather than, as computed, over every coordinate independently (Neal, 2010; Girolami & Calderhead, 2011).

We compare our adaptive HMC to NUTS, and extend our approach and compare an adaptive version of RMHMC to the standard RMHMC. For NUTS, we tuned the free parameters of its dual averaging algorithm to obtain the best performance, and for RMHMC we use the experimental protocol and code used by Girolami & Calderhead (2011). We do this for all experiments in this section. Code to reproduce these results will be available online[1].

5.1. Bayesian Logistic Regression

We consider a data set $\mathbf{X}$ consisting of N observations and D features or covariates, and a binary label $\mathbf{y}$. Using regression coefficients $\boldsymbol{\beta}$ and bias β_0 the joint distribution for the logistic regression model is:

$$\log p(\mathbf{X},\mathbf{y},\boldsymbol{\beta},\beta_0) \propto \log p(\mathbf{y}|\mathbf{X},\boldsymbol{\beta},\beta_0) + \log p(\boldsymbol{\beta},\beta_0)$$
$$= -\sum_i \log\left(1+\exp\left(-y_i(\beta_0+\mathbf{x}_i^\top\boldsymbol{\beta})\right)\right) - \frac{\beta_0^2}{2\sigma^2} - \frac{\boldsymbol{\beta}^\top\boldsymbol{\beta}}{2\sigma^2}, \quad (1)$$

[1] http://www.cs.ubc.ca/~ziyuw/ahmc/index.html

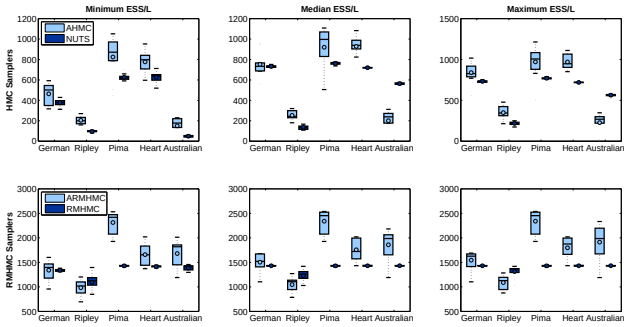

Figure 2. Box plots comparing ESS/L for Bayesian logistic regression. Top row: AHMC vs NUTS. Bottom row: ARMHMC vs RMHMC.

where $y_i \in \{-1, 1\}$, and σ^2 is the prior variance of the regression coefficients. We present results on five data sets from the UCI repository. The data sets have varying characteristics with features D ranging from 2 to 24, and the number of observations N from 250 to 1000. For each data set, we generate 5000 samples after a burnin phase of 1000 samples, and repeat this process 10 times using differing starting points. The top row of figure 2 compares the performance of our adaptive HMC (AHMC) to NUTS, while the bottom row compares our adaptive RMHMC (ARMHMC) to RMHMC. For this experiment, for AHMC, we set Γ such that $\epsilon \in [0.01, 0.2]$ and $L \in \{0, \cdots, 100\}$, and for ARMHMC, we use $\epsilon \in [0.1, 1]$ and $L \in \{1, \cdots, 12\}$.

The columns of figure 2 show box plots of the minimum, median and maximum ESS/L values obtained. We see that the adaptive methods (AHMC and ARMHMC) exhibit good performance. For the minimum ESS/L, AHMC has better (higher) values that NUTS for all the data sets, and this behavior is consistent across most other data sets for the other summary statistics. Thus AHMC typically provides better performance and a higher effective number of samples per unit of computation used than NUTS. We also see that the ARMHMC can improve RMHMC and provide better ESS/L on what is already a highly efficient sampler.

5.2. Log-Gaussian Cox Point Process

We model a data set $\mathbf{Y} = \{y_{ij}\}$ that consists of counts at locations $(i, j), i, j = 1 \ldots, d$ in a regular spatial grid using a log-Gaussian Cox point process (LGC) (Christensen et al., 2005; Girolami & Calderhead, 2011). Observations y_{ij} are Poisson distributed and conditionally independent given a latent intensity process $\boldsymbol{\Lambda} = \{\lambda_{ij}\}$ with means $s\lambda_{ij} = s\exp(x_{ij})$, where $s = \frac{1}{d^2}$. The rates $\mathbf{X} = \{x_{ij}\}$ are obtained from a Gaussian process with mean function $m(x_{ij}) = \mu\mathbf{1}$ and covari-

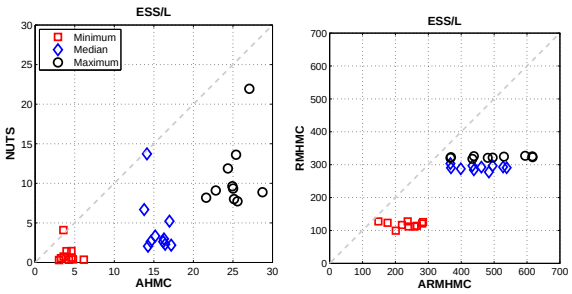

Figure 3. Comparing minimum (red), median (blue) and maximum (black) ESS/L for the Log-Gaussian Cox model. Each of the colored glyphs represents one of the 10 chains generated.

ance function $\Sigma(x_{ij}, x_{i'j'}) = \sigma^2 \exp\left(-\delta(i,i',j,j')/\beta d\right)$, where $\delta(i,i',j,j') = \sqrt{(i-i')^2 + (j-j')^2}$. The joint probability $\log p(\mathbf{y}, \mathbf{x}|\mu, \sigma, \beta)$ is proportional to:

$$\sum_{i,j} y_{ij}x_{ij} - d\exp(x_{ij}) - \frac{1}{2}(\mathbf{x}-\mu\mathbf{1})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu\mathbf{1}). \quad (2)$$

We generate samples jointly for $\mathbf{x}, \sigma, \mu, \beta$ using a grid of size $d = 64$, using a synthetic data set obtained by drawing from the generative process for this model. We generate 5000 samples after a burnin of 1000 samples. For this model, we use $L \in \{1, \cdots, 500\}$, $\epsilon \in [0.001, 0.1]$ for AHMC, and use $L \in \{1, \cdots, 60\}$, $\epsilon \in [0.01, 1]$ for ARMHMC. We compare the performance of the adaptive method we presented in terms of ESS per leapfrog step in figure 3. We compare AHMC versus NUTS and ARMHMC versus RMHMC, showing the minimum, median and maximum ESS per leapfrog step obtained for 10 chains with dispersed starting points. We see that almost all points lie below the diagonal line, which indicates that the AHMC and ARMHMC have better ESS/L compared to NUTS and RMHMC, respectively. Thus even for high-dimensional models with strong correlations our adaptive method allows for automatic tuning of the sampler and consequently the ability to obtain higher quality samples than with competing methods.

We examine the quality of the posterior distribution obtained for AHMC and NUTS in figure 4, by visualizing the latent field and its variance, and comparing to the true data (which is known for this data set). The top row shows the true latent fields. The true data observations are shown in top right corner and we see that there are few data points in this region, and thus we expect to have a high variance in this region. The average of the samples obtained using AHMC shows that we can accurately obtain samples from the latent field $\mathbf{x}$, and that the samples have a variance matching our expectations. While NUTS is able to also produce

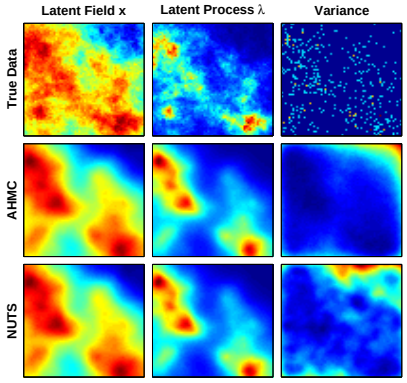

Figure 4. Comparing quality of posterior distributions from samples obtained using AHMC and NUTS for the log-Gaussian Cox model. The top-right image shows the locations of the true data.

good samples of the latent field, the variance of the field is not well captured (bottom right image).

5.3. Stochastic Volatility

We consider a stochastic volatility model described by Kim et al. (1998) and Girolami & Calderhead (2011), in which we consider observations y_t, regularly spaced in time for $t = 1, \ldots, T$. Each y_t is specified using a latent variable x_t, which represents the log-volatility following auto-regressive AR(1) dynamics. The model is specified as:

$$y_t = \epsilon_t \beta \exp\left(0.5 x_t\right), \qquad \epsilon_t \sim \mathcal{N}(0,1) \quad (3)$$

$$x_{t+1} = \phi x_t + \eta_{t+1}, \qquad \eta_{t+1} \sim \mathcal{N}(0, \sigma^2) \quad (4)$$

$$x_1 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1-\phi^2}\right), \qquad p(\beta) \propto \frac{1}{\beta}. \quad (5)$$

$|\phi| < 1$ to ensure stationarity of the log-volatility, and the standard deviation $\sigma > 0$, whose priors we set to $\frac{\phi+1}{2} \sim Beta(20, 1.5)$ and $\sigma^2 \sim \text{inv-}\chi^2(10, 0.05)$, respectively. The parameters to be sampled by HMC are thus $\mathbf{\Theta} = \{\mathbf{x}, \beta, \phi, \sigma^2\}$, and the joint probability is:

$$p(\mathbf{y}, \mathbf{\Theta}) = \prod_{t=1}^{T} p(y_t|x_t, \beta) p(x_t|x_{t-1}, \phi, \sigma^2) p(\beta) p(\sigma^2) p(\phi). \quad (6)$$

We make use of the transformations $\sigma = \exp(\gamma)$ and $\phi = \tanh(\alpha)$ to ensure that we sample using unconstrained variables; the use of this transformation requires the addition of the Jacobian of the transformation of variables. We generate samples jointly using AHMC, using training data with $T = 2000$. For our experiments, we use a burnin period of $10,000$ samples and thereafter generate $20,000$ posterior samples. We restrict our box constraint such that $L \in \{1, \cdots, 300\}$, $\epsilon \in [10^{-4}, 10^{-2}]$. We show the results comparing ESS

Table 1. Comparative results for stochastic volatility.

	ESS per Leapfrog		
Sampler	minimum	median	maximum
AHMC	**1.3 ± 0.1**	6.9 ± 0.7	14.9 ±1.4
NUTS	0.7 ± 0.3	3.5 ±1.6	9 ±2.8

for the two methods in table 1. These results again show higher values for ESS per leapfrog step, demonstrating that a better performing sampler can be obtained using AHMC – further demonstrating the advantages of AHMC methods for sampling from complex hierarchical models.

5.4. Bayesian Neural Networks

We demonstrate the application of our adaptive approach using Bayesian neural networks (BNNs) to show that AHMC allows for more effective sampling of posterior parameters even when compared to samplers finely tuned by an expert. We make use of the Dexter data set from the NIPS 2003 feature selection challenge, which is a subset of the well-known Reuters text categorization benchmark. The winning entries submitted by Neal & Zhang (2006) used a number of feature selection techniques followed by a combination of Bayesian Neural Networks and Dirichlet diffusion trees. The entry that used only BNNs was placed second and achieved highly competitive results (Guyon et al., 2005).

The BNN model consists of 295 input features and 2 hidden layers with 20 and 8 hidden units respectively. The input features are selected from the full set of features through univariate feature selection. The weights and bias as well as a few other parameters of this particular network adds up to form a 6097 dimensional state space for the HMC sampler.

For this model, we use cross-validation to construct the reward signal. We divide the data into n sets, and train n BNNs each on $n-1$ sets and test them on the remaining set like in the case of normal cross-validation. The cross-validation error is then used to calculate the reward. To take computation into account, we always evaluate the reward over the same number of leapfrog steps, i.e. for each evaluation of the reward we use a different number of samples and a different number of leapfrog steps for each sample, but the product of the two remains constant.

We compare the results in table 2, where the performance measure is the prediction error on a test set (unknown to us) and was obtained after submission to the competition system. The improved results obtained using the AHMC strategy are clear from the table, also demonstrating that good adaptation can be preferable to the introduction of more sophisticated models.

Table 2. Classification error on the Dexter test data set. We show the mean and the median prediction errors of the 8 BNNs trained by cross-validation. The majority vote of these networks achieves slightly lower error than a more sophisticated model using Dirichlet diffusion trees.

Method	**Error**
Expert-tuned HMC for BNN	0.0510
AHMC for BNN (Mean error)	0.0498
AHMC for BNN (Median error)	0.0458
Winning entry (using Dirichlet Diffusion Trees)	0.0390
AHMC for BNN + Majority Voting	0.0355

6. Discussion and Conclusion

We described the use of the expected squared jumping distance, which is a general objective suitable for models used both for inferential and predictive tasks, e.g., generalised regression. We also presented predictive measures as an alternative when prediction is the key task, e.g. neural networks, and where sufficient data is available for cross-validation. Several other objectives, such as the mean update distance, cross-validation error and the cumulative auto-correlation, are also suitable, and their use depends on the particular modelling problem. In many machine learning tasks, researchers design MCMC algorithms to estimate model parameters and, subsequently, evaluate these models using cross-validation, such as the competition task in section 5.4. In this paper, we demonstrate the use of predictive losses, such as cross-validation error, to guide the adaptation. Moreover, researchers often modify their samplers so as to reduce test set error. Hence, it is natural to use predictive performance on such predictive tasks to improve the exploration of the posterior distribution. This approach, although never reported before to the best of our knowledge, simply makes the tuning process followed by many researchers explicit.

We addressed the widely-experienced difficulty in tuning Hamiltonian-based Monte Carlo samplers by developing algorithms for infinite adaptation of these Markov chains using Bayesian optimization. The adaptive Hamiltonian Monte Carlo and adaptive Riemann manifold HMC we developed automate the process of finding the best parameters that control the performance of the sampler, removing the need for time-consuming and expert-driven tuning of these samplers. Our experiments show conclusively that over a wide range of models and data sets, the use of adaptive algorithms makes it easy to obtain more efficient samplers, in some cases precluding the need for more complex approaches. Hamiltonian-based Monte Carlo samplers are widely known to be an excellent choice of MCMC method, and we hope that this paper removes a key obstacle towards the more widespread use of these samplers in practice.

References

Andrieu, Christophe and Robert, Christian. Controlled MCMC for optimal sampling. Technical Report 0125, Cahiers de Mathematiques du Ceremade, Universite Paris-Dauphine, 2001.

Andrieu, Christophe, de Freitas, Nando, Doucet, Arnaud, and Jordan, Michael I. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43, 2003.

Atchadé, Yves and Fort, Gersende. Limit theorems for some adaptive MCMC algorithms with subgeometric kernels. *Bernoulli*, 16(1):116–154, 2010.

Beskos, Alexandros, Pillai, Natesh S., Roberts, Gareth O., Sanz-Serna, Jesus M., and Stuart, Andrew M. Optimal tuning of the hybrid Monte-Carlo algorithm. *Preprint arXiv:1001.4460*, 2010.

Brochu, Eric, Cora, Vlad M, and de Freitas, Nando. A tutorial on Bayesian optimization of expensive cost functions. *Preprint arXiv:1012.2599*, 2009.

Chen, Lingyu, Qin, Zhaohui, and Liu, Jun S. Exploring Hybrid Monte Carlo in Bayesian Computation. *Sigma*, 2:2–5, 2001.

Christensen, Ole F., Roberts, Gareth O., and Rosenthal, Jeffrey S. Scaling limits for the transient phase of local Metropolis–Hastings algorithms. *Journal of the Royal Statistical Society: Series B*, 67(2):253–268, 2005.

Duane, S, Kennedy, A D, Pendleton, B J, and Roweth, D. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Engel, Yaakov. Algorithms and representations for reinforcement learning. *Doktorarbeit, The Hebrew University of Jerusalem*, 2005.

Girolami, Mark and Calderhead, Ben. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 73(2):123–214, 2011.

Guyon, Isabelle, Gunn, Steve, Ben-Hur, Asa, and Dror, Gideon. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, volume 17, pp. 545–552, 2005.

Hamze, Firas, Wang, Ziyu, and de Freitas, Nando. Self-avoiding random dynamics on integer complex systems. *ACM Transactions on Modeling and Computer Simulation*, 23(1):9:1–9:25, 2013.

Hoffman, Matthew, Brochu, Eric, and de Freitas, Nando. Portfolio allocation for Bayesian optimization. In *Uncertainty in Artificial Intelligence*, pp. 327–336, 2011.

Hoffman, Matthew D and Gelman, Andrew. The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Preprint arXiv:1111.4246*, 2011.

Ishwaran, Hemant. Applications of hybrid Monte Carlo to Bayesian generalized linear models: Quasicomplete separation and neural networks. *Journal of Computational and Graphical Statistics*, 8(4):779–799, 1999.

Kim, Sangjoon, Shephard, Neil, and Chib, Siddhartha. Stochastic volatility: likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, 65(3):361–393, 1998.

Mahendran, Nimalan, Wang, Ziyu, Hamze, Firas, and de Freitas, Nando. Adaptive MCMC with Bayesian optimization. *Articial Intelligence and Statistics*, 2012.

Meyn, Sean P. and Tweedie, Richard L. *Markov chains and stochastic stability*. Springer-Verlag, 1993.

Močkus, Jonas. The Bayesian approach to global optimization. In *System Modeling and Optimization*, volume 38, pp. 473–481. Springer, 1982.

Mohamed, Shakir, Heller, Katherine, and Ghahramani, Zoubin. Bayesian exponential family PCA. In *Advances in Neural Information Processing Systems*, pp. 1089–1096. 2008.

Neal, R. and Zhang, J. High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees. In *Feature Extraction*, pp. 265–296. Springer, 2006.

Neal, Radford M. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

Pasarica, Cristian and Gelman, Andrew. Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, 20(1):343, 2010.

Ranzato, Marc'Aurelio and Hinton, Geoffrey. Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *IEEE Computer Vision and Pattern Recognition*, pp. 2551–2558, 2010.

Rasmussen, Carl Edward and Williams, Christopher K I. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts, 2006.

Roberts, Gareth O. and Rosenthal, Jeffrey S. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of applied probability*, 44(2):458–475, 2007.

Roberts, Gareth O. and Stramer, Osnat. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

Roberts, Gareth O and Tweedie, Richard L. Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83(1):95–110, 1996.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan Prescott. Practical Bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, 2012.

Srinivas, Niranjan, Krause, Andreas, Kakade, Sham M., and Seeger, Matthias. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010.

Vihola, Matti. Grapham: Graphical models with adaptive random walk Metropolis algorithms. *Computational Statistics and Data Analysis*, 54(1):49 – 54, 2010.