Split Hamiltonian Monte Carlo

Babak Shahbaba*, Shiwei Lan*, Wesley O. Johnson*, and Radford M. Neal†

June 30, 2011

Abstract

We show how the Hamiltonian Monte Carlo algorithm can sometimes be speeded up by "splitting" the Hamiltonian in a way that allows much of the movement around the state space to be done at low computational cost. One context where this is possible is when the log density of the distribution of interest (the potential energy function) can be written as the log of a Gaussian density, which is a quadratic function, plus a slowly varying function. Hamiltonian dynamics for quadratic energy functions can be analytically solved. With the splitting technique, only the slowly-varying part of the energy needs to be handled numerically, and this can be done with a larger stepsize (and hence fewer steps) than would be necessary with a direct simulation of the dynamics. Another context where splitting helps is when the most important terms of the potential energy function can be evaluated quickly, with only a slowly-varying part requiring costly computations. With splitting, the quick portion can be handled with a small stepsize, while the costly portion uses a larger stepsize. We show that both of these splitting approaches can reduce the computational cost of sampling from the posterior distribution for a logistic regression model, using either a Gaussian approximation centred on the posterior mode, or a Hamiltonian split into a term that depends on only a small number of critical cases, and another term that involves the larger number of cases that have little influence on the posterior distribution.

Keywords: Markov chain Monte Carlo, Hamiltonian dynamics, Bayesian analysis

1 Introduction

The simple Metropolis algorithm (Metropolis et al., 1953) is often effective at exploring low-dimensional distributions, but it can be very inefficient for complex, high-dimensional distributions — successive states may exhibit high autocorrelation, due to the random walk nature of the movement. Faster exploration can be obtained using Hamiltonian Monte Carlo, which was first

*Department of Statistics, University of California, Irvine, USA.
†Department of Statistics and Department of Computer Science, University of Toronto, Canada.

introduced by Duane et al. (1987), who called it "hybrid Monte Carlo", and which has been recently reviewed by Neal (2010). Hamiltonian Monte Carlo (HMC) reduces the random walk behavior of Metropolis by proposing states that are distant from the current state, but nevertheless have a high probability of acceptance. These distant proposals are found by numerically simulating Hamiltonian dynamics for some specified amount of fictitious time.

For this simulation to be reasonably accurate (as required for a high acceptance probability), the stepsize used must be suitably small. This stepsize determines the number of steps needed to produce the proposed new state. Since each step of this simulation requires a costly evaluation of the gradient of the log density, the stepsize is the main determinant of computational cost.

In this paper, we show how the technique of "splitting" the Hamiltonian (Leimkuhler and Reich, 2004) can be used to reduce the computational cost of producing proposals for Hamiltonian Monte Carlo. In our approach, splitting separates the Hamiltonian, and consequently the simulation of the dynamics, into two parts. We discuss two contexts in which one of these parts can capture most of the rapid variation in the energy function, but is computationally cheap. Simulating the other, slowly-varying, part requires costly steps, but can use a large stepsize. The result is that fewer costly gradient evaluations are needed to produce a distant proposal. We illustrate these splitting methods using logistic regression models for classification problems. Computer programs for our methods are publicly available from `http://www.ics.uci.edu/~babaks/Homepage/Codes.html`.

Before discussing the splitting technique, we provide a brief overview of HMC. (See Neal, 2010, for an extended review of HMC.) To begin, we briefly discuss a physical interpretation of Hamiltonian dynamics. Consider a frictionless puck that slides on an uneven surface. The state space of this dynamical system consists of its *position*, denoted by the vector q, and its momentum (mass, m, times velocity, v), denoted by a vector p. Based on q and p, we define the *potential energy*, $U(q)$, and the *kinetic energy*, $K(p)$, of the puck. $U(q)$ is proportional to the height of the surface at position q. The kinetic energy is $m|v|^2/2$, so $K(p) = |p|^2/(2m)$. As the puck moves on an upward slope, its potential energy increases while its kinetic energy decreases, until it becomes zero. At that point, it slides back down, with its potential energy decreasing and its kinetic energy increasing.

The above dynamic system can be represented by a function of q and p known as the *Hamiltonian*, which for HMC is usually defined as the sum of a potential energy, U, depending only on

the position and a kinetic energy, K, depending only on the momentum:

$$H(q,p) \;=\; U(q) + K(p) \tag{1}$$

The partial derivatives of $H(q,p)$ determine how q and p change over time, according to *Hamilton's equations*:

$$\begin{aligned} \frac{dq_j}{dt} &= \frac{\partial H}{\partial p_j} = \frac{\partial U}{\partial p_j} \\ \frac{dp_j}{dt} &= -\frac{\partial H}{\partial q_j} = -\frac{\partial K}{\partial q_j} \end{aligned} \tag{2}$$

These equations define a mapping T_s from the state at time t to the state at time $t+s$.

We can use Hamiltonian dynamics to sample from some distribution of interest by defining the potential energy function to be minus the log of the density function of this distribution (plus any constant). The position variables, q, then correspond to the variables of interest. We introduce fictitious momentum variables, p, as well (of the same dimension as q), which will have a distribution defined by the kinetic energy function. The joint density of q and p is defined by the Hamiltonian function as

$$P(q,p) \;=\; \frac{1}{Z}\,\exp\Big(-H(q,p)\Big)$$

When $H(q,p) = U(q) + K(p)$, as we assume in this paper, we have

$$P(q,p) \;=\; \frac{1}{Z}\,\exp\Big(-U(q)\Big)\,\exp\Big(-K(p)\Big)$$

so q and p are independent.

In applications to Bayesian statistics, q represents model parameters, and our objective is to sample from the posterior distribution of q given the observed data D. To this end, we set

$$U(q) \;=\; -\log(P(q)L(q|D))$$

where $P(q)$ is our prior and $L(q|D)$ is the likelihood function given data D.

Having defined a Hamiltonian function corresponding to the distribution of interest (e.g., a posterior distribution of model parameters), we could in theory use Hamilton's equations, applied for some specified time period, to propose a new state in the Metropolis algorithm. Since Hamiltonian dynamics leaves invariant the value of H (and hence the probability density), and

preserves volume, this proposal would always be accepted. (For a more detailed explanation, see Neal (2010).)

In practice, however, solving Hamiltonian's equations exactly is too hard, so we need to approximate these equations by discretizing time, using some small step size ϵ. For this purpose, the *leapfrog* method is commonly used. It consists of iterating the following steps:

$$\begin{aligned} p_j(t+\epsilon/2) &= p_j(t) - (\epsilon/2)\frac{\partial U}{\partial q_j}(q(t)) \\ q_j(t+\epsilon) &= q_j(t) + \epsilon\frac{\partial K}{\partial p_j}(p(t+\epsilon/2)) \\ p_j(t+\epsilon) &= p_j(t+\epsilon/2) - (\epsilon/2)\frac{\partial U}{\partial q_j}(q(t+\epsilon)) \end{aligned} \tag{3}$$

Typically, $K(p) = p^T M^{-1} p$, with M usually being a diagonal matrix with elements $m_1, \ldots, m_d$, so that $K(p) = \sum_i p_i^2/2m_i$. The p_j are then independent and Gaussian with mean zero, and $\partial K/\partial p_j(p(t)) = p_j(t)/m_j$.

We can use some number, L, of these leapfrog steps, with some stepsize, ϵ, to propose a new state in the Metropolis algorithm. We apply these steps starting at the current state (q, p), with fictitious time set to $t = 0$. The final state, at time $t = L\epsilon$, it taken as the proposal, (q^*, p^*). (To make the proposal symmetric, we would need to negate the momentum at the end of the trajectory, but this turns out to be unnecessary when, as here, $K(p) = K(-p)$.) We either accept or reject this new proposal, with the acceptance probability being

$$\min[1, \exp(-H(q^*, p^*) + H(q, p))] = \min[1, \exp(-U(q^*) + U(q) - K(p^*) + K(p))]$$

These Metropolis updates will leave H approximately constant, and therefore do not explore the whole joint distribution of q and p. The HMC method therefore alternates these Metropolis updates with updates in which the momentum is sampled from its distribution (which is independent of q). When $K(p) = \sum_i p_i^2/2m_i$, each p_j is sampled independently from the Gaussian distribution with mean zero and variance m_j.

As an illustration, consider sampling from the following bivariate normal distribution

$$q \sim N(\mu, \Sigma), \quad \text{with } \mu = \begin{pmatrix} 3 \\ 3 \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} 1 & 0.95 \\ 0.95 & 1 \end{pmatrix}$$

For HMC, we set $L = 20$ and $\epsilon = 0.15$. The left panel in Figure 1 shows the first 30 states from an HMC run. The density contours of the bivariate normal distribution are shown as gray

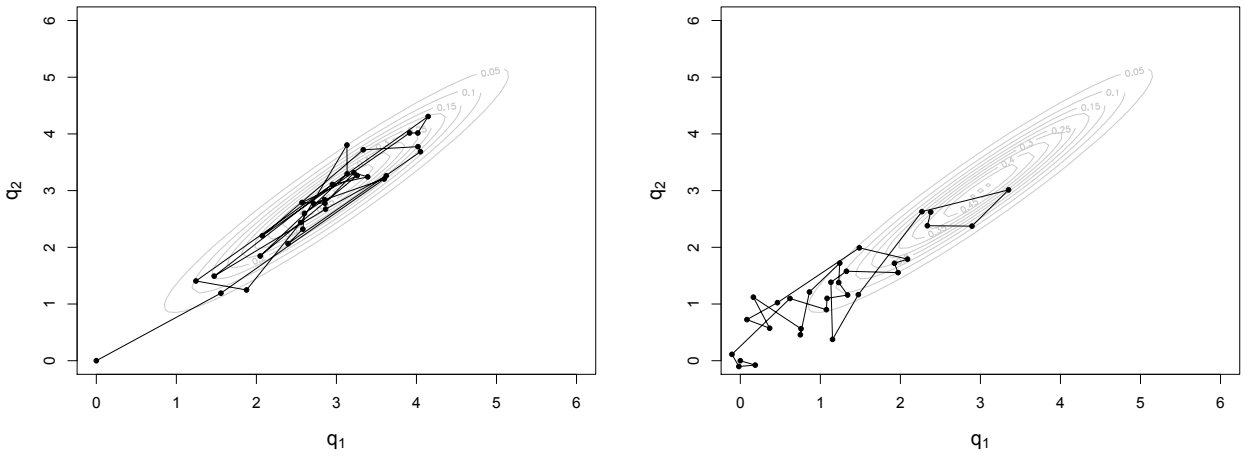

Figure 1: Comparison of Hamiltonian Monte Carlo (HMC) and Random Walk Metropolis (RWM) when applied to a bivariate normal distribution. Left Panel: The first 30 iterations of Split HMC with 20 leapfrog steps. Right Panel: The first 30 iterations of RWM with 20 updates per iterations.

ellipses. The left panel shows every 20^{th} state from the first 600 iterations of a run of a simple random walk Metropolis (RWM) algorithm. (This takes time comparable to that for the HMC run.) The proposal distribution for RWM is a bivariate normal with the current state as the mean, and $0.15^2 I_2$ as the covariance matrix. (The standard deviation of this proposal is the same as the stepsize of HMC.) Figure 1 shows that HMC explores the distribution more efficiently, with successive samples being further from each other, and autocorrelations being smaller. For an extended review of HMC, its properties, and its advantages over the simple random walk Metropolis algorithm, see Neal (2010).

In this example, we have assumed that one leapfrog step for HMC (which requires evaluating the gradient of the log density) takes approximately the same computation time as one Metropolis update (which requires evaluating the log density), and that both move approximately the same distance. The benefit of HMC comes from this movement being systematic, rather than in a random walk. We now propose a new approach called Split Hamiltonian Monte Carlo (Split HMC), which further improves the performance of HMC by modifying how steps are done, with the effect of reducing the time for one step or increasing the distance that one step moves.

2 Splitting the Hamiltonian

As discussed by Neal (2010), variations on HMC can be obtained by using discretizations of Hamiltonian dynamics derived by "splitting" the Hamiltonian, H, into several terms:

$$H(q,p) \;=\; H_1(q,p) + H_2(q,p) + \cdots + H_K(q,p)$$

We use $T_{i,t}$, for $i = 1, \ldots, k$ to denote the mapping defined by H_i for time t. Assuming that we can implement Hamiltonian dynamics H_k exactly, the composition $T_{1,\epsilon} \circ T_{2,\epsilon} \circ \ldots \circ T_{k,\epsilon}$ is a valid discretization of Hamiltonian dynamics based on H if H_i are twice differentiable (Leimkuhler and Reich, 2004). This discretization is symplectic and hence preserves volume. It will also be reversible if the sequence of H_i are symmetric: $H_i(q,p) = H_{K-i+1}(q,p)$.

Indeed, the leapfrog method (3) can be regarded as a symmetric splitting of the Hamiltonian $H(q,p) = U(q) + K(p)$ as

$$H(q,p) \;=\; U(q)/2 + K(p) + U(q)/2 \tag{4}$$

In this case, $H_1(q,p) = H_3(q,p) = U(q)/2$ and $H_2(q,p) = K(p)$. Hamiltonian dynamics for H_1 is

$$\begin{aligned} \frac{dq_j}{dt} &= \frac{\partial H_1}{\partial p_j} = 0 \\ \frac{dp_j}{dt} &= -\frac{\partial H_1}{\partial q_j} = -\frac{1}{2}\frac{\partial U}{\partial q_j} \end{aligned}$$

which for a duration of ϵ gives the first part of a leapfrog step For H_2, the dynamics is

$$\begin{aligned} \frac{dq_j}{dt} &= \frac{\partial H_2}{\partial p_j} = \frac{\partial K}{\partial p_j} \\ \frac{dp_j}{dt} &= -\frac{\partial H_2}{\partial q_j} = 0 \end{aligned}$$

For time ϵ, this gives the second part of the leapfrog step. Hamiltonian dynamics for H_3 is the same as that for H_1 since $H_1 = H_3$, giving the the third part of the leapfrog step.

2.1 Splitting the Hamiltonian when a partial analytic solution is available

Suppose the potential energy $U(q)$ can be written as $U_0(q) + U_1(q)$. Then, we can split H as follows:

$$H(q,p) \;=\; U_1(q)/2 + [U_0(q) + K(p)] + U_1(q)/2 \tag{5}$$

Here, $H_1(q,p) = H_3(q,p) = U_1(q)/2$ and $H_2(q,p) = U_0(p) + K(p)$. The first and the last terms in this case are similar to Eq. 4, where we use $U_1(q)$ instead of $U(q)$. Therefore, the first and the last part of a leapfrog step remain as before, but we use $U_1(q)$ as opposed to $U(q)$ to update p. Now suppose that the middle part of the leapfrog, which is based on the Hamiltonian $U_0(q) + K(p)$, can be handled analytically; that is, we can find the exact dynamics for time t. For this part, we should be able to take larger step sizes and fewer steps. This could lead to faster simulations from posterior probability distributions.

We are mainly interested in situations where $U_0(q)$ provides a reasonable approximation to $U(q)$. Recently, Beskos et al. (2010) proposed a similar splitting strategy for HMC on Hilbert spaces. In our approach, we approximate $U(\theta)$ by focusing on the region of highest posterior probability distribution. To this end, we focus on the region around the posterior mode, $\hat{q}$, of the distribution. To obtain $\hat{q}$ we can use methods such as the Newton-Raphson algorithm when analytical solutions are not available. We then approximate $U(q)$ with $U_0(q)$, the energy function for $N(\hat{q}, \mathcal{J}^{-1}(\hat{q}))$, where $\mathcal{J}(\hat{q}) = U''(\hat{q})$. We set $U_1(q) = U(q) - U_0(q)$.

Using the above normal approximation, $H_2(q,p) = U_0(q) + K(p)$ in Eq. (5) has a quadratic form and produces a first-order linear ODE system that can be handled analytically (Polyanin et al., 2002). In this case, we let $K(p) = \frac{1}{2}p^T p$, which is the energy for the standard normal distribution, and $U_0(q) = \frac{1}{2}(q - \hat{q})^T \mathcal{J}(\hat{q})(q - \hat{q})$ is the potential energy of the approximate normal distribution. Setting $q^* = q - \hat{q}$, the corresponding Hamiltonian dynamics can be written as follows:

$$\frac{d}{dt}\begin{bmatrix} q^*(t) \\ p(t) \end{bmatrix} = \begin{bmatrix} 0 & I \\ -\mathcal{J}(\hat{q}) & 0 \end{bmatrix}\begin{bmatrix} q^*(t) \\ p(t) \end{bmatrix}$$

where I is the identity matrix. We can rewrite the above ODE system as $\frac{d}{dt}X(t) = AX(t)$, where

$$A = \begin{bmatrix} 0 & I \\ -\mathcal{J}(\hat{q}) & 0 \end{bmatrix}$$

The fundamental solution matrix of the above system is $\Phi(t) = e^{At}$. We can diagonalize the coefficient matrix A

$$A \;=\; \Gamma D \Gamma^{-1}$$

where Γ is invertible and D is a diagonal matrix. Thus, we can rewrite the ODE system as

$$\frac{d}{dt}X(t) = \Gamma D \Gamma^{-1} X(t)$$

Algorithm 1 Leapfrog steps for split Hamiltonian Monte Carlo with a partial analytic solution.

$R \leftarrow \Gamma e^{D\epsilon}\Gamma^{-1}$

Sample initial values for p

for $\ell = 1$ to L **do**

$p \leftarrow p - (\epsilon/2)\frac{\partial U_1}{\partial q}$

$q^* \leftarrow q - \hat{q}$

$X_0 \leftarrow (q^*, p)$

$(q^*, p) \leftarrow RX_0$

$q \leftarrow q^* + \hat{q}$

$p \leftarrow p - (\epsilon/2)\frac{\partial U_1}{\partial q}$

end for

Now, let $Y(t) = \Gamma^{-1}X(t)$. Then,

$$\frac{d}{dt}Y(t) = DY(t)$$

The solution for the above equation is $Y(t) = e^{Dt}Y_0$, where $Y_0 = \Gamma^{-1}X_0$, and X_0 are the initial values. Therefore,

$$X(t) \quad = \quad \Gamma e^{Dt}\Gamma^{-1}X_0$$

The above analytical solution is of course for the middle part (denoted as H_2) of Eq. (5) only. We still need to approximate the overall Hamiltonian dynamics, H, using the leapfrog method. Algorithm 1 shows the corresponding leapfrog steps. As we can see, after an initial step of size $\epsilon/2$ based on $U_1(q)$, we obtain the exact solution at time ϵ based on $H_2(q,p) = U_0(q) + K(p)$. Then, we finish the iteration by taking another step of size $\epsilon/2$ based on $U_1(q)$.

2.2 Splitting the Hamiltonian by splitting the data

The method discussed in the previous section is based on the assumption that we are able to handle the Hamiltonian $H_2(q,p) = U_0(q) + K(p)$ analytically. If this is not possible, we might still benefit from splitting the Hamiltonian $H(q,p)$, if the computational cost associated with $U_0(q)$ is substantially lower than the computational cost of $U_1(q)$. In these situations, we can use the

Algorithm 2 Nested leapfrog steps for split Hamiltonian Monte Carlo by splitting the data.

Sample initial values for p

for $\ell = 1$ to L **do**

$p \leftarrow p - (\epsilon/2)\frac{\partial U_1}{\partial q}$

for $m = 1$ to M **do**

$p \leftarrow p - (\epsilon/2M)\frac{\partial U_0}{\partial q}$

$q \leftarrow q + (\epsilon/M)p$

$p \leftarrow p - (\epsilon/2M)\frac{\partial U_0}{\partial q}$

end for

$p \leftarrow p - (\epsilon/2)\frac{\partial U_1}{\partial q}$

end for

following split:

$$H(q,p) \;=\; U_1(q)/2 + \sum_{m=1}^{M}[U_0(p)/2M + K(p)/M + U_0(p)/2M] + U_1(q)/2 \tag{6}$$

for some $M > 1$. The above discretization can be considered as a nested leapfrog, where the outer part takes half steps to update p based on U_1 alone, and the inner part involves M leapfrog steps of size ϵ/M based on U_0.

For example, suppose our statistical analysis involves a large data set with many observations, but we believe that a small subset of data is sufficient to build a model that performs reasonably well (i.e., compared to the model that uses all the observations). In this case, we can construct $U_0(q)$ based on a small part of the observed data, and use the remaining observations to construct $U_1(q)$. That is, we divide the observed data, y, into two subsets: R_0, which is used to construct $U_0(q)$, and R_1, which is used to construct U_1:

$$\begin{aligned} U(\theta) &= U_0(\theta) + U_1(\theta) \\ U_0(\theta) &= -\log[P(\theta)] - \sum_{i \in R_0} \log[P(y_i|\theta)] \\ U_1(\theta) &= -\sum_{i' \in R_1} \log[P(y_{i'}|\theta)] \end{aligned}$$

Note that the prior appears in $U_0(\theta)$ only.

Neal (2010) discusses a related strategy for splitting the Hamiltonian by splitting the observed data into multiple subsets. However, instead of randomly splitting data, as proposed by Neal,

we split data by building an initial model based on the posterior mode, $\hat{q}$, and use this model to identify a small subset of data that sufficiently capture the overall patterns in the whole data set. Then, we use Eq. (5) to split the Hamiltonian dynamics. The corresponding leapfrog steps for this approach are presented in Algorithm 2.

3 Application of Split HMC to logistic regression models

In this section, we describe the application of Split HMC to Bayesian logistic regression models for binary classification problems. Consider the following logistic regression model:

$$P(y = 1|x, \alpha, \beta) \quad = \quad \frac{\exp(\alpha + x^T\beta)}{1 + \exp(\alpha + x^T\beta)} \tag{7}$$

For simplicity, we use θ to denote the set of all unknown parameters, (α, β). Let $P(\theta)$ be the prior distribution for θ. The posterior distribution of θ given x and y is proportional to $P(\theta)P(y|x, \theta)$. The corresponding potential energy function is

$$U(\theta) \quad = \quad -\log[P(\theta)] - \log[P(y|x, \theta)]$$

For logistic regression models, $U_0(\theta)$ based on the normal approximation, $N(\hat{\theta}, \mathcal{J}^{-1}(\hat{\theta}))$, usually provides a reasonable approximation to $U(\theta)$. Therefore, we expect the error term $U_1(\theta) = U(\theta) - U_0(\theta)$ to be small, and the splitting scheme of (5) to result in more efficient sampling by increasing the stepsize and reducing the number of leapfrog steps.

Alternatively, we could split the Hamiltonian dynamics by splitting the data into two subset and using the splitting scheme presented in (6). To illustrate the application of this approach, consider the binary classification problem presented in Figure 2, where the two classes are shown as white circles and black squares. The straight line represents the estimated classification boundary using the maximum likelihood estimate (MLE). Around this line, the estimated probabilities for the two groups are close to 0.5. Also, the points close to the boundary line have high *entropy*, which is defined as

$$e = -P(y = 0|x, \hat{\theta}) \log[P(y = 0|x, \hat{\theta})] - P(y = 1|x, \hat{\theta}) \log[P(y = 1|x, \hat{\theta})]$$

The shaded area in Figure 2 shows the region that includes the top 30% of the observations with the highest entropy values. In this example, the class probabilities for these points are between

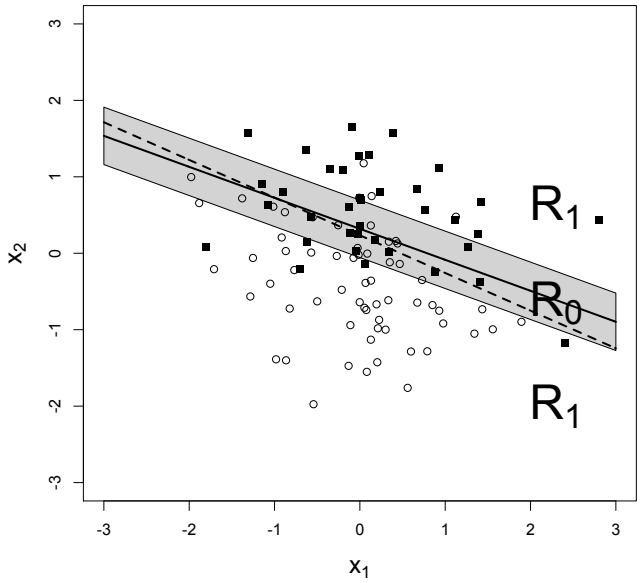

Figure 2: Splitting the observed data into two parts based on an initial logistic regression model represented by solid line. The dashed line is model using the data within R_0 region only. The energy function U is then divided into two parts: U_0 based on the data points in R_0, and U_1 based on the data points in R_1.

0.28 and 0.72. If we focus on this region and use the data points within the shaded area to re-estimate the MLE, we obtain a new classifier, which is represented by the dashed line. As we can see, the model based on a small subset of data provides a reasonable approximation for the model based on all data points. We can partition the data into two subsets. One subset includes data points with relatively high entropy points that fall within the shaded area, R_0. The remaining data points fall within the unshaded area, R_1.

Using the two subsets, we can split the energy function $U(\theta)$ into two terms: $U_0(\theta)$ based on

the data points that fall within R_0, and U_1 based on the data points that fall within R_1:

$$\begin{aligned} U(\theta) &= U_0(\theta) + U_1(\theta) \\ U_0(\theta) &= -\log[P(\theta)] - \sum_{i \in R_0} \log[P(y_i|x_i, \theta)] \\ U_1(\theta) &= -\sum_{i' \in R_1} \log[P(y_{i'}|x_{i'}, \theta)] \end{aligned}$$

To use this approach, first we obtain the posterior mode, $\hat{\theta}$. Then, we define U_0 based on the top 20% of the data points with the highest values of entropy.

In the following two sections, we use simulated and real data to compare our proposed methods to the standard HMC. For each problem, we set $L = 20$ for HMC, and find ϵ such that the acceptance rate is close to 0.65. We set the corresponding L and ϵ for Split HMC such that the trajectory length remains the same, but with longer stepsize and smaller number of steps. Among all possible options, we choose the values of L and ϵ for Split HMC such that its acceptance rate is not less than the acceptance rate of HMC. This is of course determined by how much we can increase ϵ before the acceptance rates of Split HMC methods falls bellow than what we we obtain from the standard HMC. Additionally, increasing the stepsizes beyond a certain point could lead to instability of trajectories. In such cases the error of the Hamiltonian grows without bound (Neal, 2010), and the proposals are rejected with very high probability. Therefore, in choosing the stepsizes for Split HMC methods, we are limited by the acceptance rate of the standard HMC and the point of instability of Hamiltonian dynamics.

To measure the efficiency of each sampling method, we use the autocorrelation time (ACT) by dividing the posterior samples into batches of size $B = 1000$, and estimating ACT as follows (Neal, 1993; Geyer, 1992):

$$\tau = B\frac{S_b^2}{S^2}$$

Here, S^2 is the sample variance and S_b^2 is the sample variance of batch means.

For the logistic regression models in the following two sections, we first find the autocorrelation times for each regression parameters separately, and then compare different methods using the maximum autocorrelation time over all regression parameters.

	HMC	Split HMC	
		Normal Appr.	Data Splitting
L	20	12	11
AP	0.69	0.83	0.78
τ	12.12	10.44	8.18
s	0.156	0.108	0.152
$\tau \times L$	242.4	125.28	-
$\tau \times s$	1.89	**1.13**	1.24

Table 1: Comparing Split HMC (with normal approximation and data splitting) to HMC using simulated data.

3.1 Simulated data

We simulate a data set with 100 covariates and 5000 observations. We start by sampling $x_{ij} \sim N(0, \sigma_j^2)$, for $i = 1, \ldots, 5000$ and $j = 1, \ldots, 100$. We set σ_j to 5 for the first five variables, to 1 for the next five variables, and to 0.2 for the remaining 90 variables. Next, we sample the corresponding class labels using the following model:

$$\begin{aligned} \alpha, \beta_1, \ldots, \beta_{100} &\sim N(0, 1) \\ \text{logit}(p_i) &= \alpha + x_i^T \beta \\ y_i &\sim \text{Bernoulli}(p_i) \end{aligned}$$

We run HMC, Split HMC with normal approximation, and Split HMC with data splitting (using the top 30% of the observations with the highest entropy values) for 50000 iterations. For HMC, we set $L = 20$ and $\epsilon = 0.02$ (i.e., trajectory length of $20 \times 0.02 = 0.4$). For Split HMC with normal approximation and Split HMC with data splitting, we reduce the number of leapfrog steps to 12 and 11 respectively, while increasing the stepsizes so that the trajectory length remains 0.4. For the data splitting methods, we set $M = 3$.

Table 1 shows the results for the three alternative methods. While these methods have comparable acceptance probabilities, AP, the CPU time (in seconds) per iteration, s, and $\tau \times s$ for Split HMC methods are substantially lower than that of HMC. Since the CPU time varies depending on the machine type, Table 1 also provides $\tau \times L$ for HMC and Split HMC with normal approximation.

		HMC	Split HMC	
			Normal Appr.	Data Splitting
StatLog	L	20	14	8
	AP	0.65	0.71	0.79
	τ	11.3	10.5	9.2
	s	0.046	0.034	0.039
	$\tau \times L$	226.0	147.8	-
	$\tau \times s$	0.52	0.37	**0.36**
Chess	L	20	13	9
	AP	0.78	0.89	0.85
	τ	109.12	89.43	93.61
	s	0.034	0.024	0.033
	$\tau \times L$	2182.4	1162.59	-
	$\tau \times s$	3.71	**2.18**	3.15
GISETTE	L	20	14	11
	AP	0.70	0.83	0.78
	τ	13.79	18.07	12.04
	s	0.151	0.114	0.147
	$\tau \times L$	275.8	253.0	-
	$\tau \times s$	2.08	2.06	**1.77**
CTG	L	20	14	9
	AP	0.67	0.82	0.86
	τ	220.26	167.8	179.45
	s	0.018	0.014	0.017
	$\tau \times L$	4405.2	2349.2	-
	$\tau \times s$	3.96	**2.28**	3.08

Table 2: Comparing Split HMC (with normal approximation and data splitting) to HMC using three real data sets.

3.2 Results for real data

In this section, we evaluate our proposed method using four real binary classification problems. We run each Markov chain for 200000 iterations. The autocorrelation times are obtained by dividing the posterior samples into 200 batches of size $B = 1000$.

The data for these four problems are available from the UCI Machine Learning Repository (`http://archive.ics.uci.edu/ml/index.html`). The first problem, StatLog, involves using multi-spectral values of pixels in a satellite image in order to classify the associated area into soil or cotton crop. (In the original data, different types of soil are identified.) The sample size for this data set is 4435, and the number of features is 37. For HMC, we set $L = 20$ and $\epsilon = 0.075$.

For the two Split HMC methods with normal approximation and data splitting, we reduce L to 14 and 8 respectively while increasing ϵ so $\epsilon \times L$ remains the same as that of HMC. For the data splitting method, we set $M = 3$. Table 2 compares the three sampling methods. As we can see, Split HMC methods substantially reduce $\tau \times s$ while keeping the acceptance probability above the acceptance probability of HMC.

The second problem, Chess, involves in chess endgame prediction: white can win vs. white cannot win. This data set includes 3196 instances, where each instance is a board-descriptions for the chess endgame. There are 36 attributes describing the board. For HMC, we set $L = 20$ and $\epsilon = 0.08$. We improved the computational cost (Table 2) by reducing the number of leapfrog steps to 13 and 9 for Split HMC with normal approximation and data splitting ($M = 3$) respectively.

The third problem, GISETTE, is a handwritten digit recognition problem (Guyon et al., 2004) constructed based on the MNIST data (`http://yann.lecun.com/exdb/mnist`). The objective is to separate the highly confusable digits "4" and "9". The data include $n = 6000$ observations and a set of 5000 highly sparse features. For our analysis, we use the projections of original covariates onto the first 100 principal components as predictors. We set $L = 20$ and $\epsilon = 0.11$ for HMC. For the two Split HMC methods with normal approximation and data splitting, we reduce L to 13 and 11 respectively. For the data splitting method, $\tau \times s$ is substantially reduced. For the Split HMC with normal approximation, however, reduction in s coincides with the increase in τ so the overall reduction in $\tau \times s$ is not substantial for this approach.

Finally, the fourth problem, CTG, involves analyzing 2126 fetal cardiotocograms along with their respective diagnostic features (de Campos et al., 2000). The objective is to determine whether the fetal state class is "pathologic" or not. The data include 2126 observations and 21 features. For HMC, we set $L = 20$ and $\epsilon = 0.075$. We reduced the number of leapfrog steps to 14 and 9 for Split HMC with normal approximation and data splitting ($M = 3$) respectively. As the result, the sampling efficiency improved substantially (Table 2).

4 Discussion

We have proposed the new methods for improving the efficiency of HMC. Our methods are based on splitting the Hamiltonian function, which allows much of the movement around the state space to be done at low computational cost. While we have focused on the application of our method to

logistic regression models for binary classification problems, it can easily be generalized to multinomial logistic (MNL) models for multiple classes. For these models, the regression parameters for p covariates and J classes form a matrix of $(p+1)$ rows and J columns. We can vectorize this matrix such that the model parameters, θ, becomes a vector of $(p+1) \times J$ elements. Then, for Split HMC with normal approximation, we define $U_0(\theta)$ using an approximate multivariate normal $N(\hat{\theta}, \mathcal{J}^{-1}(\hat{\theta}))$ as before. For Split HMC with data splitting, we can still construct $U_0(\theta)$ using a small subset of data with the highest entropy,

$$e = -\sum_{j=1}^{J} P(y = j|x, \hat{\theta}) \log[P(y = j|x, \hat{\theta})]$$

Our method can also be extended to other problems such as regression models. We can approximate the posterior distribution of model parameters by a Gaussian distribution. We can also split the the Hamiltonian by splitting the data. To this end, we can build a model using the posterior mode, and divide the data into two ports: 1) observations whose absolute residuals fall below a certain threshold, and 2) observations whose absolute residuals is above the threshold.

While simulated data and real classification problems presented in this paper have demonstrated the advantages of splitting the Hamiltonian dynamics in terms of improving the sampling efficiency, our proposed methods require preliminary analysis of data; mainly, we need to find the posterior mode. The performance of our approach obviously depends on how well the corresponding normal distribution approximates the posterior distribution, or how well the small subset of data (using the posterior mode and corresponding entropy measures) can capture the overall patterns in the whole data. In general, however, the computational cost associated with finding the posterior mode tends to be negligible compared to the potential improvement in sampling efficiency.

Future research in this area can involve finding a better tractable approximation (compared to the Gaussian distribution) for the posterior distribution. Also, one could investigate other methods for splitting the Hamiltonian dynamics by splitting the data. For example, one could fit a support vector machine (SVM) to binary classification data, and use the support vectors for constructing U_0. For regression models, the support vectors could be identified using a SVM model with an ϵ-insensitive error. Future research could also involve investigating the application of Split HMC to nonlinear regression and classification models, such as those based on Gaussian process priors. These models tend to be computationally intensive. In fact, the computational

cost is a major factor in limiting the application of these models. The method proposed in this paper could mitigate this by improving the efficiency of sampling from the posterior distribution of model parameters.

References

Beskos, A., Pinski, F. J., Sanz-Serna, J. M., and Stuart, A. M. (2010), "Hybrid Monte-Carlo on Hilbert spaces," Tech. rep., University College London.

de Campos, D. A., Bernardes, J., Garrido, A., de Sa, J. M., and Pereira-Leite, L. (2000), "SisPorto 2.0 A Program for Automated Analysis of Cardiotocograms," *Journal of Maternal-Fetal Medicine*, 9, 311–318.

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

Geyer, C. J. (1992), "Practical Markov Chain Monte Carlo," *Statistical Science*, 7, 473–483.

Girolami, M. and Calderhead, B. (2011), "Riemann manifold Langevin and Hamiltonian Monte Carlo methods," *Journal of the Royal Statistical Society, Series B*, 73, 1–37.

Guyon, I., Gunn, S. R., Ben-Hur, A., and Dror, G. (2004), "Result analysis of the NIPS 2004 feature selection challenge," in *NIPS*.

Leimkuhler, B. and Reich, S. (2004), *Simulating Hamiltonian Dynamics*, Cambridge University Press.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953), "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, 21, 1087–1092.

Neal, R. M. (1993), *Probabilistic Inference Using Markov Chain Monte Carlo Methods*, Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

Neal, R. M. (2010), "MCMC using Hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*, eds. S. Brooks, A. Gelman, G. Jones, and X. L. Meng, Chapman and Hall/CRC.

Polyanin, A. D., Zaitsev, V. F., and Moussiaux, A. (2002), *Handbook of First Order Partial Differential Equations*, Taylor & Francis, London.