Predictive Adaptation of Hybrid Monte Carlo with Bandits

by

Ziyu Wang

B.Math., University of Waterloo, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2012

Abstract

This thesis introduces a novel way of adapting the Hybrid Monte Carlo (HMC) algorithm using Gaussian process bandits. HMC is a powerful Markov chain Monte Carlo (MCMC) method, but it requires careful tuning of its hyper-parameters. We propose a Gaussian process bandit approach to carry out the adaptation of the hyper-parameters while the Markov chain progresses. We also introduce the use of cross-validation error measures for adaptation, which we believe are more pragmatic than many existing adaptation objectives. The new measures take the intended statistical use of the model, whose parameters are estimated by HMC, into consideration. We apply these two innovations to the adaptation of HMC for prediction and feature selection with multi-layer feed-forward neural networks. The experiments with synthetic and real data show that the proposed adaptive scheme is not only automatic, but also does better tuning than human experts.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to show my utmost gratitude to many people without whom this thesis would not have been possible. Above all, I would thank my advisor Nando de Freitas. He has not only devoted countless hours to myself and my work, but also he gave me invaluable guidance and support. His optimism and openness to new ideas motivated me to learn and explore this very interesting but sometimes intimidating field. Nothing more could have been asked from an advisor.

I also owe my thanks to many other people at UBC. Masrour Zoghi helped me to understand and navigate many difficult mathematical concepts and patiently put up with many silly questions of mine. I would like to thank Matthew Hoffman, Frank Hutter, Firas Hamze, Misha Denil, Gabriel Goh, and David Matheson, just to name a few, for sharing interesting discussions and ideas with me.

Last but not least, I would like to thank my family for their tireless support.

Dedication

This thesis is dedicated to my wife Lin and my son Ben for their love and support.

Chapter 1

Introduction

Markov chain Monte Carlo (MCMC) methods [2], such as the Metropolis-Hastings [37] algorithm, are widely used in statistics, physics and machine learning to sample from complex high-dimensional distributions and to solve combinatorial inference problems. Hybrid Monte Carlo (HMC), first introduced as a fast method for simulating molecular dynamics [11], is a particularly powerful MCMC algorithm. It was used to produce the winning entry of the *NIPS 2003 feature selection challenge* [16]. There is some evidence suggesting that HMC can perform better than traditional MCMC algorithms in high-dimensional, continuous, correlated spaces [9, 30]. Unfortunately, HMC has hyper-parameters that must be tuned every time it is deployed. It is often reported by HMC experts that tuning HMC is more difficult than tuning many other MCMC methods [20, 30].

Over the last decade, a few adaptive strategies were proposed to automatically tune the parameters of MCMC algorithms. For a comprehensive review of adaptive MCMC, we refer the reader to [5, 33]. Among the various adaptive MCMC algorithms, the ones based on stochastic approximation (SA) seem to be the most popular and successful. There are reasons for this. First, it can be shown theoretically that these algorithms are ergodic, despite the fact that that the Markov chains defined by these algorithms are inhomogeneous [3, 4, 35]. Secondly, these algorithms have been shown to produce impressive results in practice [17, 33]. However, these SA methods have important limitations too. In practice, they may be slow to converge. If one increases the learning rate to overcome this speed issue, then the algorithm is likely to get stuck in local optima and not fully explore the parameter space. We would like to have a method that is not as greedy and which allows one to have better control on the exploration-exploitation trade-off. In addition, often the objective measures that SA methods optimize (*e.g.* matching a particular acceptance rate) are based on restrictive asymptotic results [33].

Instead of using SA, in this thesis we propose to use bandits to adapt the parameters of HMC. Bandits are powerful optimization tools which balance the exploration-exploitation trade-off. They have been applied successfully in many complex stochastic domains, including web content optimization and advertising [22] and reinforcement learning [38]. They allow us to learn reward functions for actions. They also provide us with policies for choosing actions in an on-line manner. In our MCMC domain, the actions will correspond to the the HMC hyper-parameters and the rewards to the objective function of the adaptation scheme.

Bandits have two important advantages over the conventional SA approaches. First, they do not impose restrictions on the reward, such as differentiability. Second, they conduct global optimization as opposed to local optimization. In some earlier works [18, 24], Gaussian process bandits (GP bandits) were used to guide MCMC samplers. Although their techniques work well, they do not allow for infinite adaptation because of the unbounded growth in the GP bandit model. In some cases, as pointed out in [18], finite adaptation can cause serious mixing problems. For this reason we devise a technique that enables infinite adaptation while employing GP bandits in this work. More specifically, we use an annealing schedule to decrease the probability of a new set of parameters proposed by GP bandit to be accepted thus limiting the total number of unique points in our Gaussian process model. With fewer unique points, computation can be carried out more efficiently which then allows for infinite adaptation. Also note that, this annealing schedule also enforces diminishing adaptation which is essential to show ergodicity of the adapted sampler.

In [18, 24], the cumulative autocorrelation function is used as the objective for adaptation. Our bandit model could also use this objective function and, as a result, be applicable to any problem to which HMC is being applied. However, since we are mostly concerned with predictive tasks in machine learning, we introduce a new way to think about the objective function for doing adaptive MCMC in this thesis. Specifically, we use predictive losses, such as cross-validation error, to guide the adaptation. This approach, although never reported before to the best of our knowledge, makes perfect intuitive sense. Ultimately the models whose parameters we are estimating by running a Markov chain will be tested on predictive tasks. Hence, it is only natural to use predictive performance on such predictive tasks to improve the exploration of the posterior distribution. We expect this insight to have a profound impact on the development of adaptive MCMC algorithms for statistical prediction in the future. Of course, this will only be true in settings where enough data is available to obtain good predictive measures.

The fact that bandits do not require the objective function to be analytical is what endows us with so much flexibility in the choice of the objective function. The two improvements proposed in this thesis go hand-in-hand.

In our experiments, we use adaptive HMC to train Bayesian Neural Networks (BNNs) [29]. Since the end goal of BNNs is to predict well on test data, we use the cross-validation performance as the objective function for adapting the parameters of the HMC chain. When doing this, we can still have the right asymptotic distribution, under vanishing adaptation, provided the base samplers are uniformly ergodic or, at least, close to geometrically ergodic. We expand on these theoretical considerations in the analysis section. The experiments demonstrate, with both real and synthetic data, that the proposed adaptation scheme performs better than human experts in the task of tuning HMC for Bayesian neural networks.

1.1 Thesis Contribution

This thesis has two major contributions. First, it introduces an infinite adaptation scheme that we applied to HMC. Adapting MCMC chains by using GP bandits has been explored before [25]. The introduction of annealing schedules which enable infinite adaptation is however novel. This thesis also presents a new objective function that can be used to adapt Markov chains that are used for predictive tasks.

1.2 Thesis Organization

This thesis is organized as follows: chapter 2 briefly reviews HMC and motivates our work by demonstrating HMC's sensitivity to hyper-parameters changes. In chapter 3, we review the GP bandit model and introduces the proposed algorithm as well as our reward model. We also discuss in short the theoretical perspectives of our approach in chapter 3. In chapter 4, we evaluate the proposed approach through two sets of experiments. Finally, in chapter 5 we conclude the thesis and discuss potential future works.

Chapter 2

Hybrid Monte Carlo

In this chapter, we review the HMC method briefly. We also demonstrate HMC's sensitivity to parameter tuning with a simple set of experiments.

2.1 HMC

HMC is based on Hamiltonian mechanics. Let $\mathbf{x}$ be a d-dimensional position vector and $\mathbf{p}$ be a d-dimensional momentum vector. The total energy, also called the Hamiltonian, is given by: $\mathcal{H}(\mathbf{x}, \mathbf{p}) = \mathcal{U}(\mathbf{x}) + \mathcal{K}(\mathbf{p})$, where $\mathcal{U}(\mathbf{x})$ is the potential energy and $\mathcal{K}(\mathbf{p}) = \mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}/2$ is the kinetic energy for some mass matrix $\mathbf{M}$. In a closed system, the total energy $\mathcal{H}$ is conserved. Consequently, the dynamics of the system, according to Newton's law of motion, can be described by using Hamilton's equations:

$$\frac{\partial \mathcal{H}}{\partial \mathbf{x}} = -\dot{\mathbf{p}}, \qquad \frac{\partial \mathcal{H}}{\partial \mathbf{p}} = \dot{\mathbf{x}}. \tag{2.1}$$

Suppose we wish to draw samples from the distribution $\pi(\mathbf{x})$ which is known up to a normalization constant. That is $\pi(\mathbf{x}) \propto f(\mathbf{x})$. To make use of Hamiltonian Mechanics for sampling we can think of the vector $\mathbf{x}$ as the position vector in the Hamiltonian by defining the potential energy as $\mathcal{U}(\mathbf{x}) = -\log f(\mathbf{x})$. We introduce a kinetic energy term $\mathcal{K}(\mathbf{p}) = \mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}/2$ with $\mathbf{M}$ being positive definite and define the total energy to be $\mathcal{H}(\mathbf{x}, \mathbf{p}) = \mathcal{U}(\mathbf{x}) + \mathcal{K}(\mathbf{p})$.

Consider the distribution $\hat{\pi}(\mathbf{x}, \mathbf{p}) \propto \exp(-\mathcal{H}(\mathbf{x}, \mathbf{p}))$. Since $\hat{\pi}(\mathbf{x}, \mathbf{p}) \propto f(\mathbf{x}) \times \exp(-\mathcal{K}(\mathbf{p}))$, to sample from π, we can simply sample from $\hat{\pi}$ and discard the samples for $\mathbf{p}$. One way of sampling from $\hat{\pi}$ is to follow the Hamiltonian dynamics:

1. Sample $\hat{\mathbf{p}} \sim \mathcal{N}(\mathbf{0}, \mathbf{M})$,

2. Simulate the Hamiltonian dynamics described in Equation 2.1 for a certain time starting from the current state $(\mathbf{x}^t, \hat{\mathbf{p}})$ to generate the next state $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1})$,

The above procedure simulates a Markov chain that leaves the target distribution invariant. Suppose now $(\mathbf{x}^t, \mathbf{p}^t)$ follows the target distribution, then $(\mathbf{x}^t, \hat{\mathbf{p}})$ would also follow the target distribution since $\mathbf{x}$ and $\mathbf{p}$ are independent and $\mathbf{p}$ is indeed distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{M})$. Step 2 of the procedure employs the transformation defined in 2.1 which we shall name $\mathcal{T}$ for convenience. For any subset $\mathcal{A}$ of the joint space of $\mathbf{x}$ and $\mathbf{p}$, let $\mathcal{B} = \mathcal{T}(\mathcal{A})$. Due to the fact that $\mathcal{T}$ is time-reversible [30], we have that $\mathcal{A} = \mathcal{T}(\mathcal{B})$ which implies that $q(\mathcal{B}|\mathcal{A}) = q(\mathcal{A}|\mathcal{B}) = 1$ where q is the transition probability of $\mathcal{T}$. Because $\mathcal{T}$ also preserves volume and conserves total energy [30], it can be shown that $\int_{\mathcal{A}} \exp(-\mathcal{H}(\mathbf{x}, \mathbf{p})) d\mathbf{x} d\mathbf{p} = \int_{\mathcal{B}} \exp(-\mathcal{H}(\mathbf{x}, \mathbf{p})) d\mathbf{x} d\mathbf{p}$. Therefore, we have the detailed balance condition:

$$q(\mathcal{B}|\mathcal{A}) \int_{\mathcal{A}} \exp(-\mathcal{H}(\mathbf{x}, \mathbf{p})) d\mathbf{x} d\mathbf{p} = q(\mathcal{A}|\mathcal{B}) \int_{\mathcal{B}} \exp(-\mathcal{H}(\mathbf{x}, \mathbf{p})) d\mathbf{x} d\mathbf{p}, \tag{2.2}$$

which guarantees that the transformation $\mathcal{T}$ leaves the target distribution invariant. Since both steps of the procedure leaves the target distribution invariant and they only depend on the previous iteration of the algorithm, together they can be seen as a Markov transition kernel which leaves the target distribution invariant.

To simulate the Hamiltonian dynamics in practice, however, we must discretize the differential equations that describe the continuous motion. This can be accomplished by the Störmer-Verlet or leapfrog scheme [21]. It is composed of three steps as described below:

$$\mathbf{p}_{\tau+\frac{\epsilon}{2}} = \mathbf{p}_{\tau} - \frac{\epsilon}{2} \frac{\partial U}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{\tau}}, \ \mathbf{x}_{\tau+\epsilon} = \mathbf{x}_{\tau} + \epsilon M^{-1} \mathbf{p}_{\tau+\frac{\epsilon}{2}}, \ \mathbf{p}_{\tau+\epsilon} = \mathbf{p}_{\tau} - \frac{\epsilon}{2} \frac{\partial U}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{\tau+\epsilon}}$$

where τ is the current time and ϵ is the stepsize. To simulate the dynamics, we repeat the above steps L times with stepsize ϵ. Although the leapfrog scheme does preserve volume in the phase space and it is time-reversible, it no longer conserves the total energy. Because of this, the detailed balance condition no longer holds. HMC surmounts this equilibration problem with a Metropolis-Hastings re-weighting step [11]. The full HMC algorithm is summarized in Algorithm 1. It can be shown that Markov chain defined by HMC satisfies the detailed-balance condition [23, 30].

Algorithm 1 Hybrid Monte Carlo Algorithm

```
1: for $i = 1, 2, \ldots$ do
2:     Sample $\mathbf{p}^t \sim \mathcal{N}(\mathbf{0}, M)$
3:     Given $\epsilon$ and $L$, apply the leapfrog scheme $L$ times with stepsize $\epsilon$
       starting from the current state $(\mathbf{x}^t, \mathbf{p}^t)$ to generate a proposal state
       $(\mathbf{x}^*, \mathbf{p}^*)$
4:     Draw $\mathbf{u} \sim \mathcal{U}(0,1)$
5:     if $\mathbf{u} < \min[1, e^{H(\mathbf{x}^t,\mathbf{p}^t)-H(\mathbf{x}^*,\mathbf{p}^*)}]$ then
6:         Let $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1}) = (\mathbf{x}^*, \mathbf{p}^*)$
7:     else
8:         Let $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1}) = (\mathbf{x}^t, \mathbf{p}^t)$
9:     end if
10: end for
```

2.2 Sensitivity to Parameter Tuning

HMC has shown great promise [9, 30], however, it has not been extensively adopted in Bayesian statistics as a practical inference method [14]. One of the main reasons for this is HMC's sensitivity to changes in its hyper-parameters, such as ϵ and L. We demonstrate this with three experiments. We use HMC to sample from bivariate Gaussian distribution with covariance:

$$\Sigma = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 1 \end{bmatrix}. \tag{2.3}$$

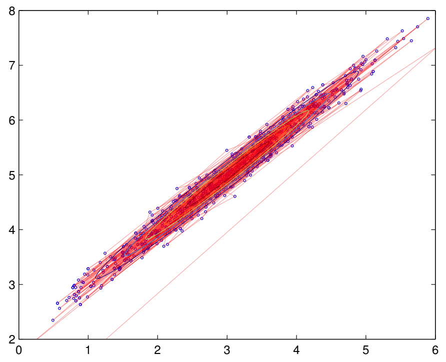
In the experiments, we vary the hyper-parameters of HMC. Specifically, we set $\epsilon = 0.16$ $L = 40$ in the first experiment, $\epsilon = 0.16$ and $L = 50$ in the second, and $\epsilon = 0.15$ and $L = 50$ in the third experiment. A total of 1000 samples were drawn in all three experiments. The results are summarized in Figure 2.1, 2.2 and 2.3.

In the first experiment, as evident from the trajectory plot and the trace plot in plot 2.1(**a**) and 2.2(**a**), the samples in the first experiment cover the space very well and the sampler seems to have converged to the target distribution very quickly. Also from the auto-correlation plot in Figure 2.3(**a**), we can see that the auto-correlation of this sampler is low (its absolute value is bounded by 0.1). By changing L by 10 in the second experiment, however, we observe different results. Although the samples still cover the space very well (as seen in the trajectory plot in Figure 2.1(**b**)), the sampler does not mix as well despite the increase in L. To see this, note that the auto-correlation (presented in Figure 2.3(**b**)) of this sampler is as high as 0.9 when lag is 1 and that the absolute value of the auto-correlation stays relatively high as the lag increases. One can also compare the trace plots 2.2(**a**) and 2.2(**b**) to draw the same conclusion. Finally, in the third experiment, we decrease the value of ϵ by 0.01. Despite the minimal change in ϵ, however, the behavior of the sampler changes dramatically. From Figure 2.1(**c**), we can see that after 1000 iterations, the sampler has not yet converged to the target distribution.

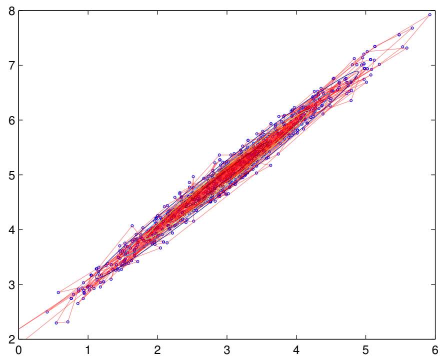
It is interesting to note that, theoretical results concerning the optimal acceptance rate of HMC exist. Neal and Beskos et al. both suggested it to be around 0.65 [8, 30]. Such results, however, would not help in choosing the best sampler out of the three since all of them have their acceptance rate to be around 0.7.

HMC's sensitivity to hyper-parameters would not be a problem if good hyper-parameters for one particular target distribution generalizes to many other target distributions. This is, however, not the case as we demonstrate by the following example. We again choose our target distribution to be bivariate Gaussian but this time with the covariance matrix

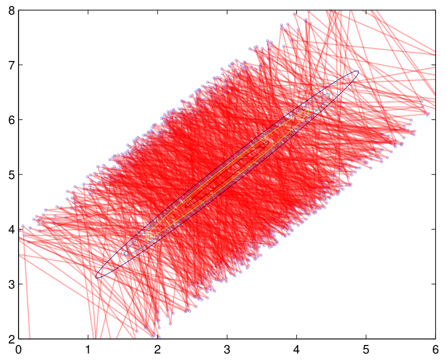
$$\Sigma = \begin{bmatrix} 2 & 0.99 \\ 0.99 & 2 \end{bmatrix}. \tag{2.4}$$



(**a**)



(**b**)



(**c**)

Figure 2.1: Sampling from a Gaussian distribution using HMC for 1000 iterations. **(a)** shows the trajectory of the samples when $\epsilon = 0.16$ and $L = 40$. **(b)** shows the trajectory of the samples when $\epsilon = 0.16$ and $L = 50$. **(c)** shows the trajectory of the samples when $\epsilon = 0.15$ and $L = 50$. It is evident from the plot that by changing the hyper-parameters of HMC by a small amount, we can drastically change the convergence behavior of the sampler.

As in the previous experiment, we set $\epsilon = 0.16$ $L = 40$ in the first experiment, $\epsilon = 0.16$ and $L = 50$ in the second, and $\epsilon = 0.15$ and $L = 50$ in the third. The auto-correlation of the samples are summarized in Figure 2.4. By comparing the result with that of the previous experiments, we can see that the hyper-parameters that led to the best result in the previous experiments perform poorly in this experiment. Also hyper-parameters with poor performance in the last experiments perform well here. This experiment illustrates that not only are HMC's hyper-parameters hard to tune, but also, they may have to be tuned for each new target distribution encountered.

These experiments demonstrate that even with a small change in hyper-parameters, the behavior of HMC can be altered substantially and the changes affect both convergence speed and mixing. This sensitivity to parameter changes hinders HMC's practical use. To overcome this problem, we introduce an automatic adaptation algorithm in the following section.

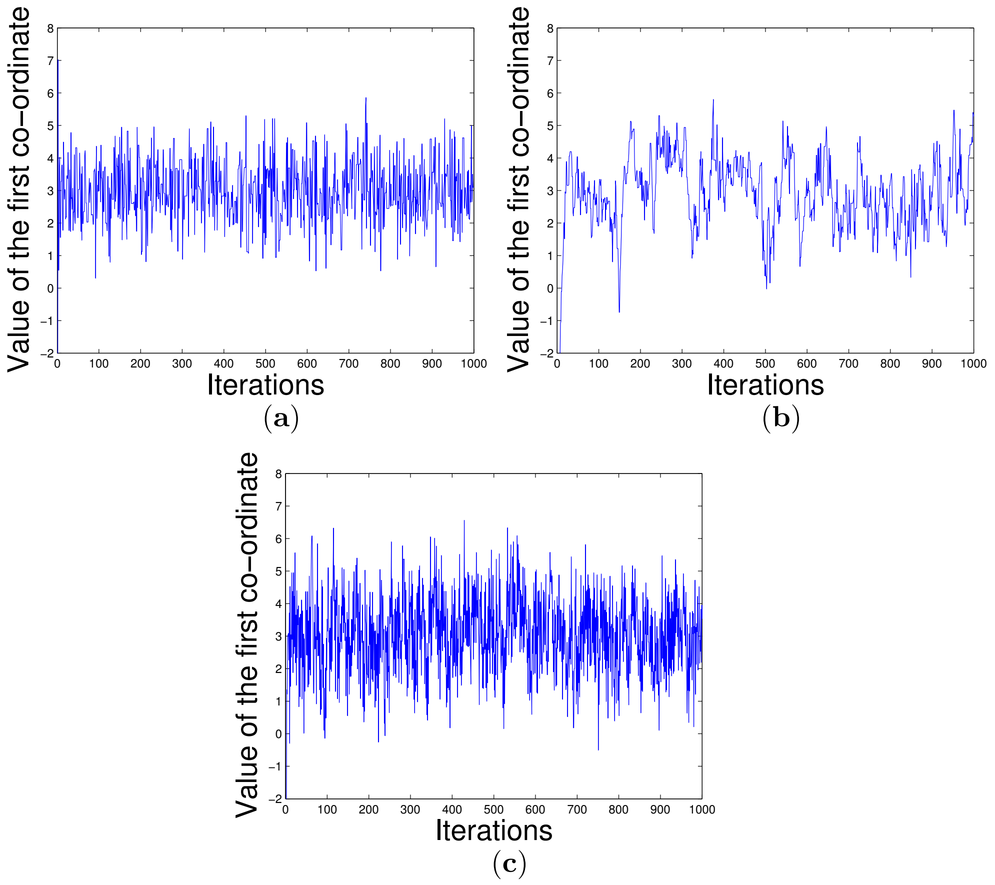
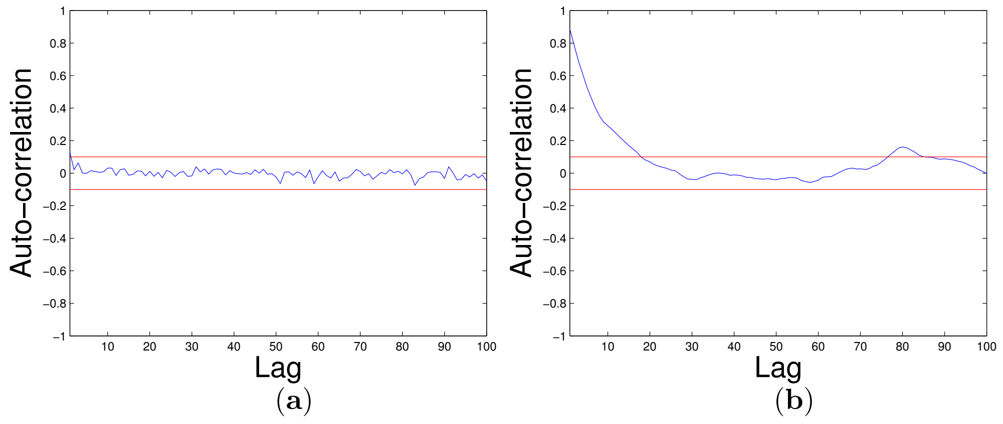


Figure 2.2: Sampling from a Gaussian distribution using HMC for 1000 iterations. **(a)** shows the trace of the first coordinate when $\epsilon = 0.16$ and $L = 40$. **(b)** shows the trace of the first coordinate when $\epsilon = 0.16$ and $L = 50$. **(c)** shows the trace of the first coordinate when $\epsilon = 0.15$ and $L = 50$.



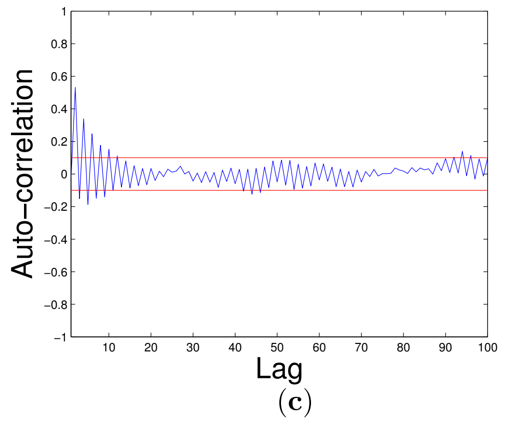
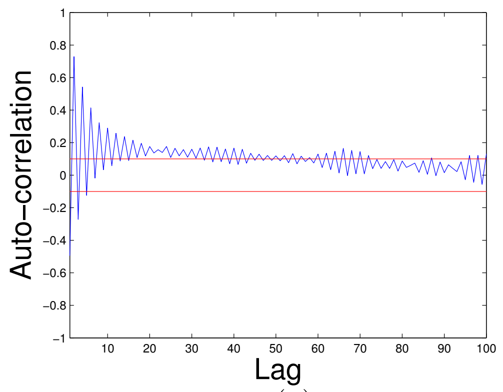
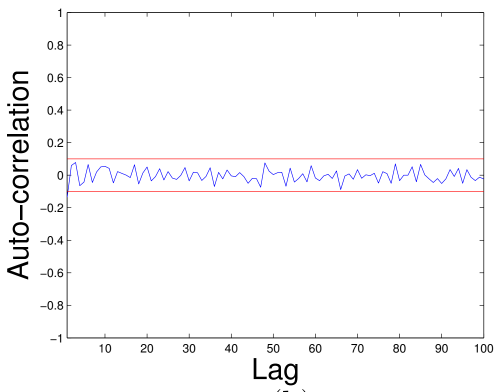


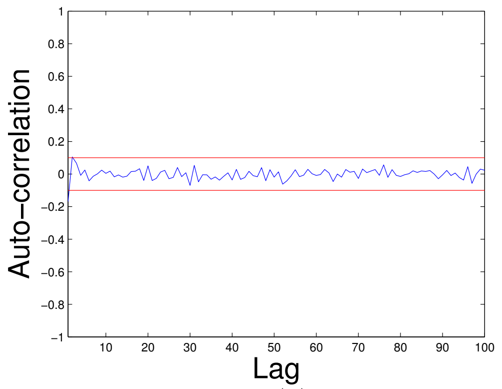
Figure 2.3: Sampling from a Gaussian distribution using HMC for 1000 iterations. **(a)**, **(b)** and **(c)** describes the auto-correlation of the samplers with $\epsilon = 0.16$ $L = 40$, $\epsilon = 0.16$ $L = 50$, and $\epsilon = 0.15$ $L = 50$ respectively. As we vary the hyper-parameters of HMC by a small amount, the speed of mixing of the chain can change drastically.



(**a**)



(**b**)



(**c**)

Figure 2.4: Sampling from a bivariate Gaussian distribution using HMC for 1000 iterations. **(a)**, **(b)** and **(c)** describes the auto-correlation of the samplers with $\epsilon = 0.16$ $L = 40$, $\epsilon = 0.16$ $L = 50$, and $\epsilon = 0.15$ $L = 50$ respectively. As we change the target distribution, the behavior of hyper-parameters changes with it.

Chapter 3

Adaptive Hybrid Monte Carlo

Our adaptive algorithm uses GP bandits to update L and ϵ on-line, as the HMC chain explores the parameter space of the model (*e.g.* a neural network). The rewards in our setting are given by the cross-validation accuracy of the predictive model. It is clear that our approach is more general and could in fact use any other reasonable objective functions to learn L and ϵ on-line. However, if the model over whose parameters we define the Markov chain is used for prediction, then it is reasonable to use prediction loss to update the hyper-parameters of the Markov chain sampler.

3.1 Gaussian Process Bandits

GP bandits (also known as Bayesian optimization, Kriging) is an efficient gradient-free optimization tool well suited for expensive black box functions such as our reward function which is based on cross-validation. It operates on a compact action space $\mathcal{A}$ [1] , where in our case an action $\mathbf{a} \in \mathcal{A}$ corresponds to a specific choice of the hyper-parameters L and ϵ. GP bandits assume a Gaussian process prior for the reward function which is defined over $\mathcal{A}$. That is

$$f(\cdot) \sim GP(m(\cdot), k(\cdot, \cdot))$$

[1] It is common in the GP bandit literature to assume $\mathcal{A}$ to be a bounding box such that each dimension is restricted to be within some interval $[b_l, b_h]$. In our experiments, we set $\mathcal{A}$ to be a box constraint such that

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

for some $b_l^\epsilon \leq b_u^\epsilon$ and $b_l^L \leq b_u^L$.

where $m(\cdot)$ is the mean function and $k(\cdot,\cdot)$ is the covariance function. Here we assume a Gaussian noise model such that an observation at an action $\mathbf{a}$ is assumed to be $\mathbf{r}(\mathbf{a}) = f(\mathbf{a}) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_\eta^2)$. Given noisy evaluations of the objective function $\{\mathbf{r}_k\}_{k=1}^i$ evaluated at points $\{\mathbf{a}_k\}_{k=1}^i$, we can compute the likelihood $\mathbb{P}(\mathcal{D}_i|f(\cdot))$ where $\mathcal{D}_i = \left(\{\mathbf{a}_k\}_{k=1}^i, \{\mathbf{r}_k\}_{k=1}^i\right)$. By combining the prior and the likelihood using Bayes rule, we acquire the posterior distribution of the reward model

$$\mathbb{P}\left[f(\cdot)|\mathcal{D}_i\right] \propto \mathbb{P}[\mathcal{D}_i|f(\cdot)]\mathbb{P}[f(\cdot)].$$

The posterior contains updated information about the reward model.

More specifically, by assuming $m(\cdot) = 0$, we arrive at the posterior predictive distribution:

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

where

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

$\mathbf{k} = [k(\mathbf{a},\mathbf{a}_1)\ \dots\ k(\mathbf{a},\mathbf{a}_i)]^T$, and $\mathbf{R}_i = [\mathbf{r}_1\ \dots\ \mathbf{r}_i]^T$. In this work, we used the Gaussian ARD kernel with $k(\mathbf{a}_i,\mathbf{a}_j) = \exp(-\frac{1}{2}\mathbf{a}_i^T\Sigma^{-1}\mathbf{a}_j)$ where Σ is a positive definite matrix[2]. For more details of the above model please refer to [31].

[2] In our experiments we set

$$\Sigma = \begin{bmatrix} [\alpha(b_u^\epsilon - b_l^\epsilon)]^2 & 0 \\ 0 & \left[\alpha(b_u^L - b_l^L)\right]^2 \end{bmatrix}$$

where $\alpha = 0.2$.

To sample efficiently, GP Bandits choose the next action $\mathbf{a} \in \mathcal{A}$ to be the optimizer of an acquisition function. The role of an acquisition function is to guide the GP bandit towards regions of potentially better objective values by sampling either a region known to have good values or a region with high uncertainty. That is to either exploit or explore. There are several popular choices of acquisition functions in the literature. For our purposes, we choose the Upper Confidence Bound (UCB) [36] which is defined as

$$u(\mathbf{a}|\mathcal{D}_i) = \mu_i(\mathbf{a}) + \beta_{i+1}^{\frac{1}{2}}\sigma_i(\mathbf{a}).$$

Here $\beta_{i+1} = 2\log\left(\frac{(i+1)^{\frac{d}{2}+2}\pi^2}{3\delta}\right)$ where d is the dimension of $\mathcal{A}$ and δ is set to 0.1. UCB is a standard acquisition function for which asymptotic rates of convergence have been proved [36] for the GP bandits. There are, however, a few other reasonable alternatives to UCB, such as Thompson sampling [26] and expected improvement (EI) [27]. A comparison among these options as well as portfolio strategies to combine them appeared recently in [19].

Algorithm 2 Gaussian Process Bandit

```
Given: function $f$, constraint $\mathcal{A}$, and $\mathbf{a}_1$.
Initialize $\mathcal{D}_0 = \emptyset$.
for $i = 1, 2, \ldots, I$ do
    Obtain a noisy evaluation of the reward function $\mathbf{r}_i = f(\mathbf{a}_i) + \epsilon$.
    Augment the data $\mathcal{D}_i = \{\mathcal{D}_{i-1}, (\mathbf{a}_i, \mathbf{r}_i)\}$.
    Update the bandit model.
    Optimize the acquisition function to acquire $\mathbf{a}_{i+1}$: $\mathbf{a}_{i+1} = \arg\max_{\mathbf{a}\in\mathcal{A}} u(\mathbf{a}|\mathcal{D}_i)$.
end for
```

There are several good ways of optimizing the acquisition function, including the method of DIvided RECTangles (DIRECT) of [13] and many versions of the projected Newton methods of [7]. We found DIRECT to provide a very efficient solution in our domain. Note that optimizing the acquisition function is much easier than optimizing the original objective function. This is because the acquisition functions can be easily evaluated and differentiated. For the full algorithm of GP bandit please refer to Algorithm 2.

3.2 Reward Model

To evaluate different parameter settings, we introduce super-transitions, which were first described by Neal in [29]. A super-transition consists of a number of HMC samples. This number can vary. A super-transition, however, defines the total number of leapfrog steps used. If each sample requires more leapfrog steps, then there will be fewer HMC samples generated in one super-transition. When running HMC with different parameters for one super-transition, we may have a different number of HMC iterations and a different L, but all runs will take approximately the same CPU time. Super-transitions thus enable us to evaluate the effectiveness of different parameter settings while preserving the total computational cost.

We use cross-validation to construct the reward signal. As in classical n-fold cross-validation, 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. Alternatively, we can think of the process as a single Markov chain exploring the state space which is a combination of n independent copies of the original state space. During each super-transition, we draw samples for each of the BNNs and calculate the mean cross-validation error by using only the samples drawn in this super-transition. The mean cross-validation error is then used to calculate the reward.

Here we remind the reader again that the samples of HMC are eventually used for prediction tasks. Therefore adapting the sampler to gain prediction accuracy is pragmatic.

3.3 The Algorithm

Our objective function cannot be evaluated analytically. However, noisy observations of the objective value can be obtained by running HMC with the specified parameter settings for one super-transition. These noisy observations together with the parameter settings can then be used to update the posterior distributions for the reward model. Finally, we use the acquisition function to propose a new set of parameters.

To ensure that the diminishing adaptation condition [32] is satisfied, we introduce an annealing schedule. Let p_i denote the probability of a new proposal generated by our bandit model being accepted at iteration i. In each iteration, p_i is calculated as follows, $p_i = \exp(-\lambda(i-1))$ where $\lambda > 0$. As p_i goes to 0, we change our parameter settings less frequently.[3] It is easy to check that the diminishing adaptation condition is satisfied.

As p_i decreases, it becomes increasingly difficult for the bandit to propose new hyper-parameters for HMC. Thus the sampler would often be using the same set of hyper-parameters for many iterations. In this case, we argue, it is more reasonable to exploit known good hyper-parameters rather than exploring for better ones. To this goal, we propose to adopt the following acquisition function:

$$\bar{u}(\mathbf{a}|\mathcal{D}_i) = \mu_i(\mathbf{a}) + p_i \beta_{i+1}^{\frac{1}{2}} \sigma_i(\mathbf{a}).$$

Some may argue that changing the acquisition function could lead to premature exploitation which may prevent the GP bandit from locating the true optimum of the reward function. This argument certainly holds. Our goal of adapting the Markov chain, however, is less about finding the absolute best hyper-parameters but more about finding sufficiently good hyper-parameters given the computational resources we have. Given enough time, we can slow the annealing schedule or simply delay annealing for a certain number of super-transitions thus allowing the bandit algorithm to explore fully the hyper-parameter space. During an aggressive annealing schedule though, we argue, it is important not to waste computational resources on hyper-parameters with little potential. As we demonstrate in out experiments, this form of aggressive exploitation still leads to promising outcomes. The full algorithm is presented in Algorithm 3.

Because of the use of GP priors, we could run into the situation where it becomes computationally prohibitive to update the bandit model or to

[3] λ is set to 0.01 in all our experiments.

Algorithm 3 Adaptive HMC with Bayesian GP Bandits

```
1: Given: constraint $\mathcal{A}$ and $\mathbf{a}_1$.
2: for $i = 1, 2, \ldots, I$ do
3:     Run HMC for 1 super-transition with hyper-parameters $\mathbf{a}_i = (\epsilon_i, L_i)$.
4:     Use the drawn samples to obtain a noisy evaluation of the reward
       function $\mathbf{r}_i$ (say, cross-validation accuracy).
5:     Augment the data $\mathcal{D}_i = \{\mathcal{D}_{i-1}, (\mathbf{a}_i, \mathbf{r}_i)\}$.
6:     Draw $\mathbf{u} \sim \mathcal{U}(0,1)$
7:     let $p_i = \exp(-\lambda(i-1))$, with $\lambda > 0$.
8:     if $\mathbf{u} < p_i$ then
9:         Optimize the acquisition function: $\mathbf{a}^\star = \arg\max_{\mathbf{a}\in\mathcal{A}} \bar{u}(\mathbf{a}|\mathcal{D}_i)$.
10:        Let $\mathbf{a}_{i+1} = \mathbf{a}^\star$
11:    else
12:        Let $\mathbf{a}_{i+1} = \mathbf{a}_i$
13:    end if
14: end for
```

optimize the acquisition function since GP bandits require the inversion of the covariance matrix, which has the complexity of $\mathcal{O}(i^3)$ where i is the number of iterations. However, observe that as i increases, p_i decreases exponentially. Thus the total expected cost of kernel matrix inversion is

$$\mathcal{O}\left(\sum_{i=1}^{\infty} \frac{i^3}{\exp(\lambda(i-1))}\right) = \mathcal{O}(1).$$

Please bear in mind that the proposed approach does not restrict itself from adopting other reasonable annealing schedules without losing its potential for infinite adaptation. For example, one could set p_i to be $p_i = i^{-\gamma}$, where $\gamma > 0$. This schedule would allow for more adaptation which may prove critical in recovering from bad hyper-parameters as hyper-parameters that perform well in the initial phase of adaptation do not necessarily work after the sampler converges to its target distribution. It would also enable infinite adaptation as the number of unique points in our Gaussian process model would only grow logarithmically in the number of iterations. This slow growth would in turn endow us the power to use kernel specification techniques like the one proposed in [12] to reduce the computational cost.

3.4 Analysis

In the proposed approach, the Markov chain is adapted so as to minimize prediction loss. In doing this, the question of ergodicity arises immediately. Fortunately, there exist theoretical results that establish the ergodicity of general adaptive MCMC schemes [6, 32]. Specifically two sets of conditions together guarantee ergodicity of an adaptive MCMC algorithm. First, the adaptation has to diminish eventually. This condition is usually ensured by the design of the adaptation scheme. The second set of conditions is usually placed on the underlying MCMC samplers. In [32], the samplers are required to be either uniformly ergodic or geometrically ergodic. Since the state space of HMC is unbounded, it is unlikely that HMC is uniformly ergodic. To the best of our knowledge, no theoretical results exist on the geometric ergodicity of HMC. However, Roberts et al. showed in [34] 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 [6], Atchadé et al. weakened the conditions required for a general adaptive MCMC algorithm to be ergodic. In their work, although the authors still require diminishing adaptation, 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 indeed satisfies them. Alternatively, if the HMC dynamics were defined on a compact space, then the proposed adaptive scheme would be ergodic [32].

Chapter 4

Application to Bayesian Neural Networks

We demonstrate in this section the proposed adaptive HMC strategy on two applications of BNNs. We choose this domain not only because it is very challenging, but also because in this case we can benchmark our adaptive algorithms against the results obtained by human experts.

4.1 Robot Arm Data

The robot arm data set is a classical nonlinear regression benchmark [10]. There are two real input variables x_1 and x_2 representing the joint angles and two real output variables y_1 and y_2 representing the resulting arm position in rectangular coordinates. The data is generated from the following model:

$$y_1 = 2.0\cos(x_1) + 1.3\cos(x_1 + x_2) + e_1 \tag{4.1}$$

$$y_2 = 2.0\sin(x_1) + 1.3\sin(x_1 + x_2) + e_2, \tag{4.2}$$

where e_1 and e_2 are independent Gaussian noise variables of mean 0 and standard deviation 0.05. This data set contains 600 input-target pairs. The first 200 cases of the data are used for training and the last 200 cases are used for testing. Neal in [29] applied BNNs to this problem. For inference, he used HMC. The hyper-parameters of HMC were hand tuned.

In the experiment that produced the best results, Neal used super-transitions of 32000 leapfrog steps to train a network of one hidden layer with 16 hidden units for 150 super-transitions. The dimension of the state space for HMC , formed by the weights, biases and, a few other parameters of the

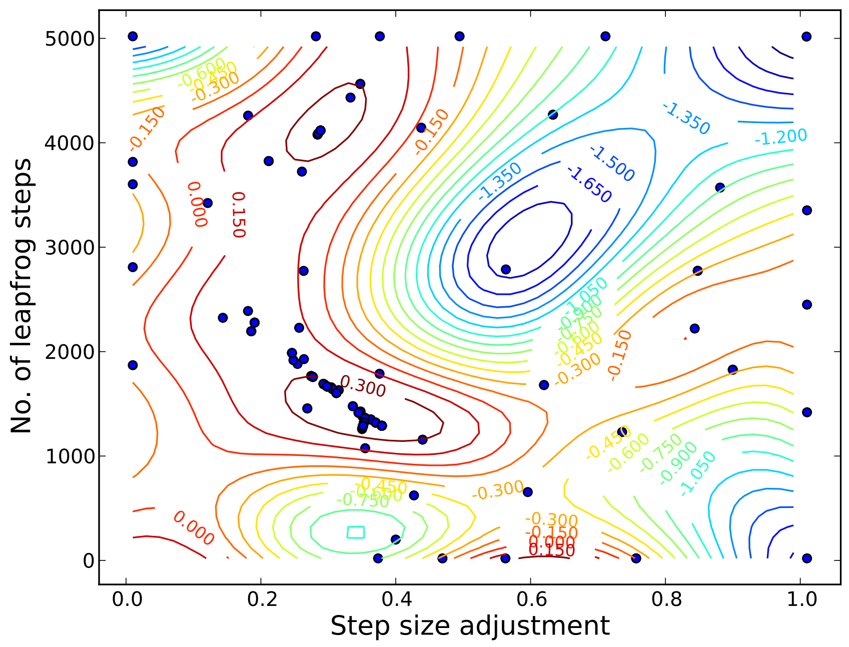


Figure 4.1: Learned mean reward surface for the Robot Arm data, with the tried parameters as dots. Notice that areas of the hyper-parameter space with higher reward value are tried more frequently.

network, is 82. We follow Neal on the structure of the network and train the network using super-transitions of 24000 leapfrog steps for 200 iterations. In doing this we preserve the total number of leapfrog steps. We use 8-fold cross-validation to obtain the rewards. That is, after each super-transition, the cross-validation error is calculated by averaging the mean squared error of each network. We would like to stress here that the cross-validation error is calculated on the training data set alone. In this experiment, we set $\mathcal{A}$ to be such that $b_l^\epsilon = 0.01$, $b_u^\epsilon = 1.01$, $b_l^L = 20$, and $b_u^L = 5020$.

The test set error (discarding the first 6000 samples) is summarized in Table 4.1. The table also includes results from other samplers [1]. The proposed adaptive scheme not only outperforms the human expert (Neal in this case), but also does better than state-of-the-art methods such as

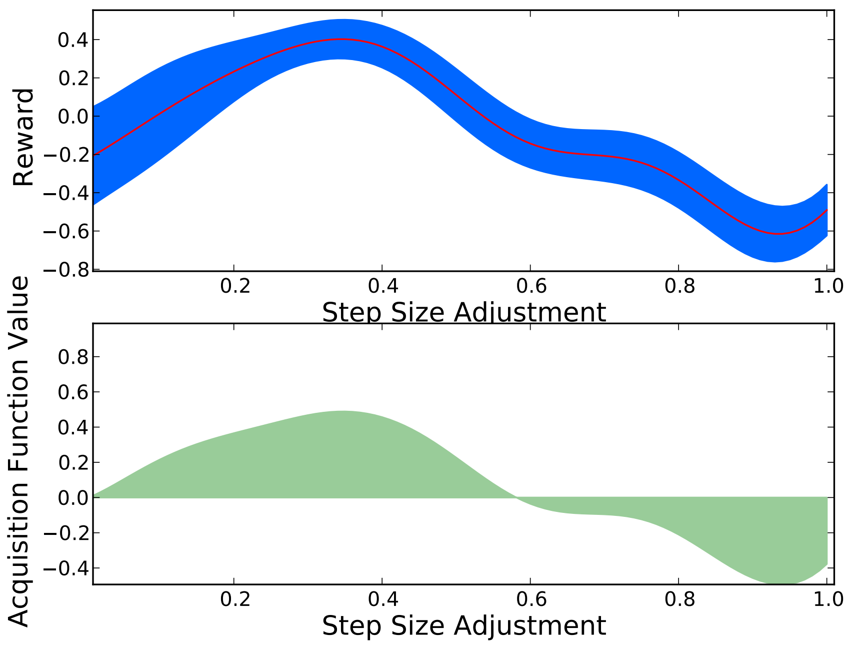


Figure 4.2: The upper plot shows the approximate reward function learned by the GP bandit along a slice with $L = 1500$. The blue-shaded region is formed by adding/subtracting one standard deviation to/from the mean (red line). The lower plot shows the value of the acquisition function along the same slice.

reversible jump MCMC. Figure 4.1 shows the reward surface learned by the GP bandit. Figure 4.2 shows the reward function along a slice $L = 1500$. The figure also shows the value of the acquisition function along this slice. Last but not least, Figure 4.3 shows the averaged mean squared error on the test set given the number of samples with the first 6000 samples discarded.

4.2 Dexter Data Set

For our second demonstration, we use the Dexter data set from the Neural Information Processing Systems (NIPS) feature selection challenge in

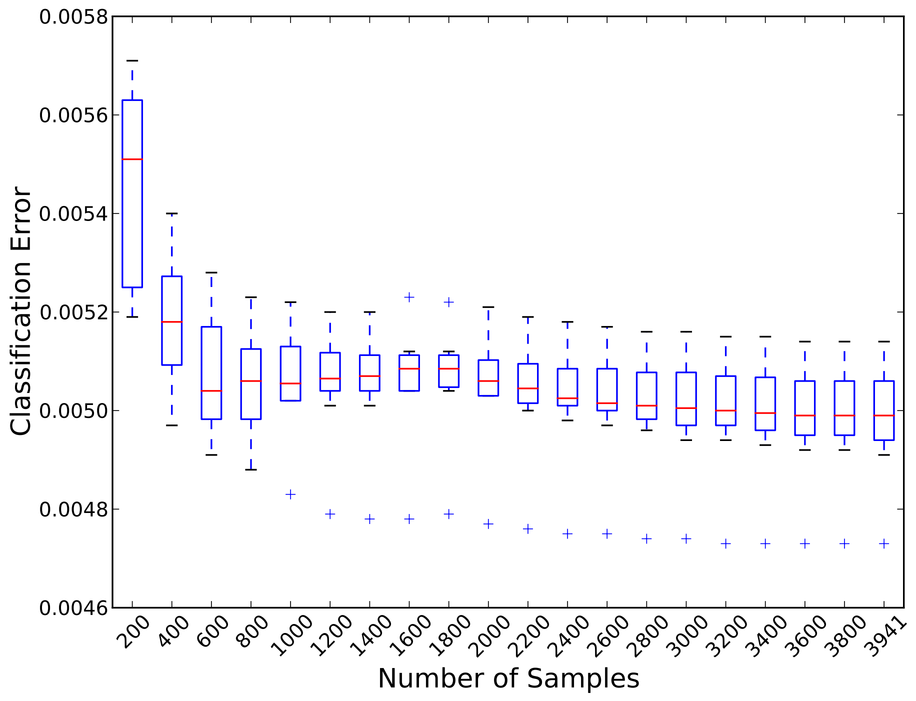


Figure 4.3: Mean squared error on the test set of the Robot Arm data set given the number of samples used (with the first 6000 samples discarded). As more samples are drawn, the error decreases steadily.

Table 4.1: Mean squared test error for the robot arm data set.

Method	**Mean Squared Error**
Rios Insua and Muller's (1998) MLP with reversible-jump MCMC	0.00620
Mackay's (1992) Gaussian approximation with highest evidence	0.00573
Neal's (1996) HMC	0.00554
Neal's (1996) HMC with ARD	0.00549
Reversible-jump MCMC with Bayesian model by Andrieu et al.	0.00502
Adaptive HMC (Median Error)	0.00499
Adaptive HMC (Mean Error)	0.00498 ± 0.00012

2003 [16]. The Dexter data set is a subset of the well-known Reuters text categorization benchmark. The data was originally collected and labeled by Carnegie Group Inc. and Reuters Ltd. in the course of developing the CONSTRUE text categorization system. The winning entries submitted by Neal and Zhang used a number of feature selection techniques followed by Bayesian Neural Networks and Dirichlet diffusion trees [28]. The entry that used only BNNs was placed second and achieved highly competitive results [16]. In this experiment, we apply our adaptation strategy to sample from this model.

Thanks to the public release of the code for the competition, we were able to use the same neural network model as Neal: "New-Bayes-nn-sel".

This model uses 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 a HMC sampler.

For a detailed description of this model and the feature selection steps please refer to [28]. In his entry, Neal used in total 3200000 leapfrog steps. In our experiments we used 160 super-transitions each with 20000 leapfrog

Table 4.2: Classification error on the validation set of the Dexter data set.

Method	**Classification Error**
New-Bayes-nn-sel	0.0800
Adaptive HMC (Mean error)	0.0730 ± 0.0096
Adaptive HMC (Median error)	0.0700
Adaptive HMC + Majority Voting	0.0667

Table 4.3: Classification error on the test set of the Dexter data set.

Method	**Classification Error**
New-Bayes-nn-sel	0.0510
Adaptive HMC (Mean error)	0.0498
Adaptive HMC (Median error)	0.0458
BNN + Dirichlet Diffusion Tree	0.0390
Adaptive HMC + Majority Voting	0.0355

steps. That is to say we used the same number of leapfrog steps that Neal used. We divided the 300 training cases into 10 equal parts and carried out 10-fold cross-validation to generate the reward. In this experiment, we set $b_l^\epsilon = 0.01$, $b_u^\epsilon = 0.61$, $b_l^L = 20$, and $b_u^L = 2000$.

In addition to the training set, the competition also provides a validation set which is used to assess the performance of one's method before formally submitting the result to the competition. The median and mean classification error rate of all 10 networks on the validation set as well as Neal's result is provided in Table 4.2 [15]. Figure 4.4 depicts the contour plot learned by the bandit approach, as well as, the hyper-parameters tried over time. Figure 4.5 presents the classification errors on the validation set. Figure 4.6 shows the mean reward function as well as the acquisition function learned along the slice $\epsilon = 0.22$.

The test set of the Dexter dataset is not publicly available. However, we were able to submit our result to the competition site to calculate the errors

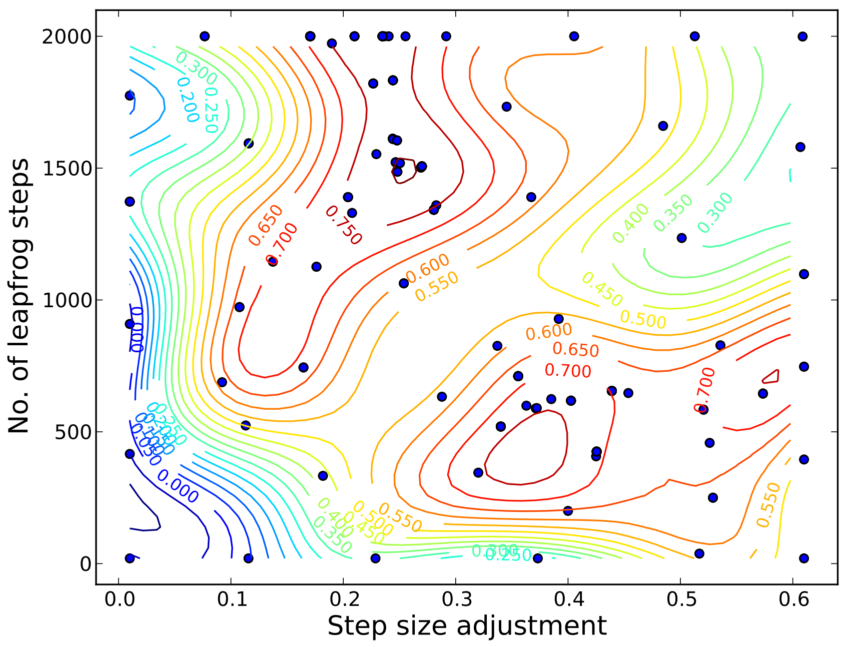


Figure 4.4: Reward surface as a contour plot learned for the Dexter data set from the NIPS 2003 feature selection challenge. The scattered dots are the hyper-parameters tried.

on the test set. The mean classification error of our 10 networks is 0.04975 whereas the median is 0.04575. In comparison, Neal and Zhang's entry with the same model had a 0.0510 error rate.

To assess whether we could combine the 10 neural networks trained to achieve better results, we used the 10 networks (from the 10-fold cross-validation procedure) to classify the test data set via majority voting. The majority voting procedure is implemented as such: We classify each input in the test data set with all 10 networks and obtain the resulting labels. The input is then assigned the majority label of the 10. Ties are broken arbitrarily. By using training data alone, our method attained on the test dataset a 0.0355 classification error rate. The best entry by Neal and Zhang for this data set, using Dirichlet diffusion trees together with BNNs trained on the training data set as well as the validation data set, achieved an error rate of 0.0390. The results on the test set is summarized in Table 4.3. The same majority voting procedure is also applied to the validation set and the result is shown in Table 4.2. The gains of the adaptive HMC strategy in this example from the NIPS competition are very clear and significant, demonstrating that good adaptation can sometimes be preferable to the introduction of more sophisticated models.

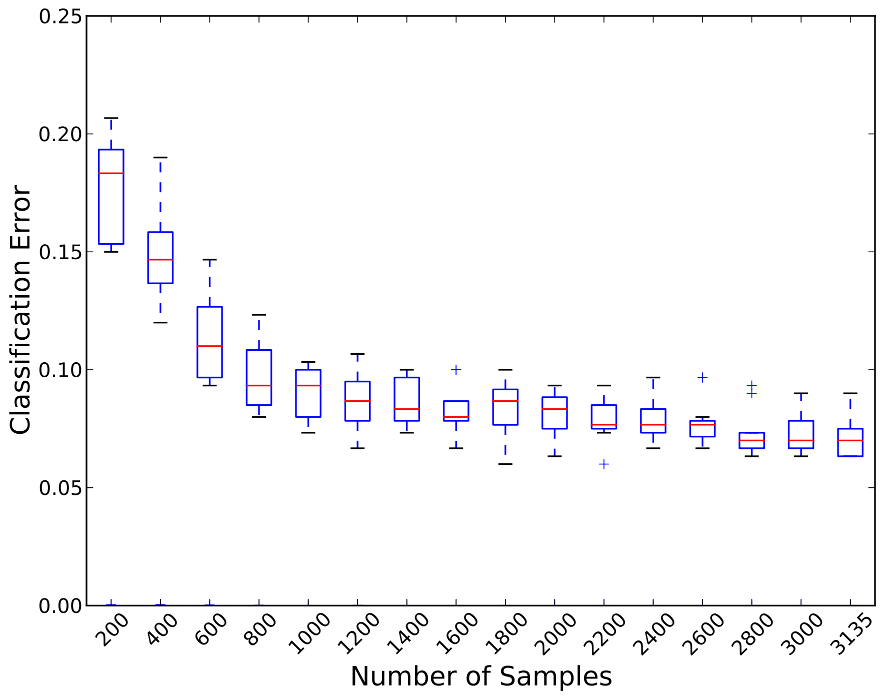


Figure 4.5: Classification errors on the validation set of the Dexter data set from NIPS 2003 feature selection challenge given the number of samples. The first 6000 samples were discarded.

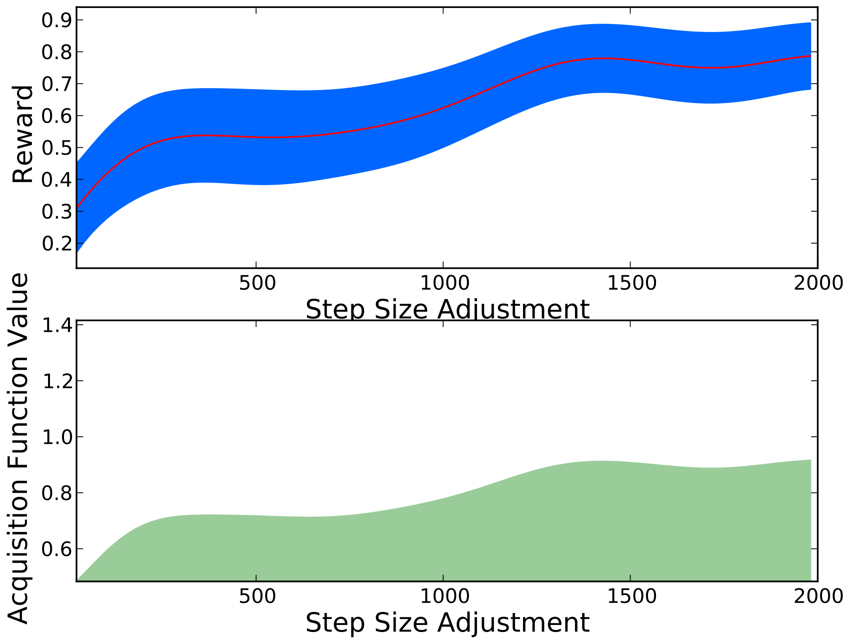


Figure 4.6: The upper plot shows the approximate reward function learned by the GP bandit along a slice with $\epsilon = 0.22$. The blue-shaded region is formed by adding/subtracting one standard deviation to/from the mean (red line). The lower plots shows the value of the acquisition function along the same slice.

Chapter 5

Conclusion

In this thesis we were able to show that GP bandits can be effectively used to adapt the hyper-parameters of hybrid Monte Carlo. This approach was shown not only for a simple, but high-dimensional dataset but also for a complex classification task, where Bayesian neural networks have proven their worth. The experiments showed that it is indeed possible to not only eliminate the tedious exercise of choosing the hyper-parameters, but that this can in fact lead to better results (as measured by NIPS competition standards).

This thesis also introduces a new data-driven objective function for adaptive MCMC. We believe this strategy increases the level of practicality of adaptive MCMC in statistical prediction tasks.

In proving ergodicity of the proposed approach, the lack of geometric ergodicity results for HMC poses a serious difficulty. Therefore a natural question to ask is whether HMC is indeed sub-geometrically ergodic or not. Let us keep in mind, however, the bandit strategy outlined here can be applied to other geometrically ergodic samplers. In such settings, the adaptive MCMC method would be provably ergodic.

Another reasonable next step is to carry out analysis that proves or disproves convergence of the GP bandit used in the proposed algorithm without the need of an annealing schedule. This result would be useful as convergence of the GP bandit given non-i.i.d. noise would not only be theoretically interesting but it would also ensure diminishing adaptation without the need of an annealing schedule. Moreover, since there are finite regret bounds for bandit methods as well as finite bounds for MCMC in discrete state spaces, we conjecture that these results together with our adaptive method should allow for the establishment of the first finite bounds on the convergence of adaptive MCMC schemes.

Bibliography

[1] C. Andrieu, N. Freitas, and A. Doucet. Robust full Bayesian learning for radial basis networks. *Neural Computation*, 13(10):2359–2407, 2001.

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

[3] Christophe Andrieu and Eric Moulines. On the ergodicity properties of some adaptive MCMC algorithms. *The Annals of Applied Probability*, 16(3):1462–1505, 2006.

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

[5] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and Computing*, 18(4):343–373, 2008.

[6] Y. Atchadé and G. Fort. Limit theorems for some adaptive MCMC algorithms with subgeometric kernels. *Bernoulli*, 16(1):116–154, 2010.

[7] Dimitri P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2):221–246, 1982.

[8] Alexandros Beskos, Natesh S. Pillai, Gareth O. Roberts, Jesus M. Sanz-Serna, and Andrew M. Stuart. Optimal tuning of the hybrid monte-carlo algorithm. *Arxiv preprint arXiv:1001.4460*, 2010.

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

[10] J.F.G. De Freitas. Bayesian methods for neural networks. *Unpublished doctoral dissertation, Cambridge University, Cambridge, UK*, 1999.

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

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

[13] Daniel E Finkel. *DIRECT Optimization Algorithm User Guide*. Center for Research in Scientific Computation, North Carolina State University, 2003.

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

[15] I. Guyon. *Feature extraction: foundations and applications*, volume 207. Springer Verlag, 2006.

[16] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, volume 17, pages 545–552, 2005.

[17] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

[18] Firas Hamze, Ziyu Wang, and Nando de Freitas. Self-avoiding random dynamics on integer complex systems. Technical Report arXiv:1111.5379v2, 2011.

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

[20] Hemant Ishwaran. 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.

[21] B. Leimkuhler and S. Reich. *Simulating Hamiltonian dynamics*, volume 14. Cambridge Univ Press, 2004.

[22] L. Li, W. Chu, J. Langford, and R.E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[23] Jun S. Liu. *Monte Carlo strategies in scientific computing.* Springer, 2001.

[24] Nimalan Mahendran, Ziyu Wang, Firas Hamze, and Nando de Freitas. Bayesian optimization for adaptive MCMC. Technical Report arXiv:1110.6497v1, 2011.

[25] Nimalan Mahendran, Ziyu Wang, Firas Hamze, and Nando de Freitas. Adaptive mcmc with bayesian optimization. *Articial Intelligence and Statistics*, 2012.

[26] Benedict C May, Nathan Korda, Anthony Lee, and David S Leslie. Optimistic Bayesian sampling in contextual bandit problems. 2011.

[27] Jonas Močkus. The Bayesian approach to global optimization. In *System Modeling and Optimization*, volume 38, pages 473–481. Springer Berlin / Heidelberg, 1982.

[28] R. Neal and J. Zhang. High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees. *Feature Extraction*, pages 265–296, 2006.

[29] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Verlag, 1996.

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

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

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

[33] Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, June 2009.

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

[35] Eero Saksman and Matti Vihola. On the ergodicity of the adaptive Metropolis algorithm on unbounded domains. *Annals of Applied Probability*, 20(6):2178 – 2203, 2010.

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

[37] W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[38] T.J. Walsh, I. Szita, C. Diuk, and M.L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 591–598. AUAI Press, 2009.