Predictive Adaptation of Hybrid Monte Carlo with Bayesian Parametric Bandits

Ziyu Wang & Nando de Freitas
University of British Columbia
Vancouver, Canada
{ziyuw, nando}@cs.ubc.ca

Abstract

This paper introduces a novel way of adapting the Hybrid Monte Carlo (HMC) algorithm using parametric bandits with nonlinear features. HMC is a powerful Markov chain Monte Carlo (MCMC) method, but it requires careful tuning of its hyper-parameters. We propose a Bayesian parametric 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.

1 Introduction

Markov chain Monte Carlo (MCMC) methods [2], such as the Metropolis-Hastings [36] 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 [12], is a particularly powerful MCMC algorithm. It was used to produce the winning entry of the *NIPS 2003 feature selection challenge* [14]. There is some evidence suggesting that HMC can perform better than traditional MCMC algorithms in high-dimensional, continuous, correlated spaces [10, 29]. 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 [29, 18].

Over the last decade, a few adaptive strategies were proposed to automatically tune the parameters of MCMC algorithms. 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 [4, 3, 33]. Secondly, these algorithms have been shown to produce impressive results in practice [15, 31]. 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 [31].

Instead of using SA, in this paper 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 [20] and reinforcement learning [37]. 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. There are two types of bandits that can be used to construct adaptive MCMC methods: nonparametric bandits (aka Bayesian optimization [8], Gaussian process bandits [35], EGO, etc.) and parametric bandits [6]. In our earlier work [22, 16], we adopted nonparametric bandits with Gaussian processes. Although nonparametric techniques work well, they do not allow for infinite adaptation because of the unbounded growth in the model. In some cases, as pointed out in [16], finite adaptation can cause serious mixing problems. For this reason we opt for parametric bandits in this work. Much work has been done in parametric bandits, but with a focus on linear models in the agnostic setting. Here, we propose Bayesian nonlinear parametric bandits with carefully specified priors. These behave similarly to the nonparametric counterparts, but allow for infinite adaptation.

In [22, 16], the cumulative autocorrelation function is used as the objective for adaptation. Our parametric bandits could also use this objective function and, as a result, be applicable to any problem to which HMC is being applied to. 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 paper. 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 the performance on such predictive tasks to improve the exploration of the posterior distribution. We expect this insight to have a profound impact in 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 paper go hand-in-hand.

In our experiments, we use adaptive HMC to train Bayesian Neural Networks (BNNs) [28]. 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.

2 Hybrid Monte Carlo

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 the system, the total energy $\mathcal{H}$ is conserved. Consequently, the dynamics of the system, according to Newton's law of motion, can be described using Hamilton's equations:

$$\frac{\partial \mathcal{H}}{\partial \mathbf{x}} = -\dot{\mathbf{p}}, \quad \frac{\partial \mathcal{H}}{\partial \mathbf{p}} = \dot{\mathbf{x}}. \tag{1}$$

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 (2) for a certain time starting from the current state $(\mathbf{x}^t, \hat{\mathbf{p}})$ to generate the next state $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1})$,

To simulate the Hamiltonian dynamics in practice, however, we must discretise the differential equation that describes the continuous motion. This can be accomplished by the Störmer-Verlet or leapfrog scheme [19]. It is composed of three steps as described below:

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

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. However, if we only do this, the detailed balance condition does not hold. HMC surmounts this equilibration problem with a Metropolis-Hastings re-weighting step [12]. The full HMC algorithm is summarized in Algorithm 1. It can be shown that Markov chain defined by HMC satisfies the detailed-balance condition [29, 21].

Algorithm 1 Hybrid Monte Carlo Algorithm

```
1: for $i = 1, 2, \dots$ 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
```

3 Adaptive Hybrid Monte Carlo

Our adaptive algorithm uses Bayesian parametric 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 function 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 Bayesian Parametric Bandits

For each action $\mathbf{a} \in \mathcal{A}$, where in our case an action corresponds to a choice of the hyper-parameters L and ϵ, we observe features $\mathbf{X}_a \in \mathbb{R}^K$. In our MCMC setting, we propose to use K Gaussian radial basis functions (RBFs) as the features. That is, for each action, there is a K-dimensional feature vector whose i-th coordinate is given by:

$$\mathbf{X}_a(i) = \exp\{-\left|\left|\mathbf{a} - \mu_i\right|\right|^2 / \beta\} \qquad \text{for } i \in \{1, 2, ..., K\}$$

where μ_i are the location parameters and β is the basis width. That is, given an input vector $\mathbf{a}$, we expand it using K Gaussian RBFs to form the features.

The locations for the RBFs are generated randomly using Latin hypercubes [24] to ensure we have a good covering of the space. This is fairly easy in our case as we only have two parameters. Instead of choosing the width explicitly, we place a discrete uniform prior on a finite set of potential values and carry out numerical quadrature to integrate it out.

We adopt the following linear Gaussian reward model:

$$p(\mathbf{r}_a|\mathbf{X}_a, \mathbf{w}, \xi, \sigma^2) = \mathcal{N}(\mathbf{r}_a|\xi + \mathbf{X}_a\mathbf{w}, \sigma^2\mathbf{I}_N), \tag{2}$$

where ξ is an offset term. We place an improper prior on ξ of the form $p(\xi) \propto 1$, and then integrate it out to get $p(\mathbf{r}_a|\mathbf{X}_a, \mathbf{w}, \sigma^2)$, with mean $\overline{r}\mathbf{1}_N + \mathbf{X}_a\mathbf{w}$, where $\overline{r} = \frac{1}{N}\sum_{i=1}^{N} r_i$ is the empirical mean of the output. For notational simplicity, we shall assume the output has been centered, and write $\mathbf{r}_a$ for $\mathbf{r}_a - \overline{r}\mathbf{1}_N$. The natural conjugate prior has the following form:

$$p(\mathbf{w}, \sigma^2) \quad = \quad \mathrm{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_0, \mathbf{V}_0, a_0, b_0). \tag{3}$$

Using Bayes rule, we obtain the posterior distribution:

$$\begin{aligned} p(\mathbf{w}, \sigma^2|\mathcal{D}) &= \mathrm{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_n, \mathbf{V}_n, a_n, b_n) \\ \mathbf{w}_n &= \mathbf{V}_n(\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{X}_a^T\mathbf{r}_a) \\ \mathbf{V}_n &= (\mathbf{V}_0^{-1} + \mathbf{X}_a^T\mathbf{X}_a)^{-1} \\ a_n &= a_0 + n/2 \\ b_n = b_0 &+ \frac{1}{2}\left(\mathbf{w}_0^T\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{r}_a^T\mathbf{r}_a - \mathbf{w}_n^T\mathbf{V}_n^{-1}\mathbf{w}_n\right). \end{aligned} \tag{4}$$

Given a new test point $\tilde{\mathbf{X}}$, the *predictive distribution* is the following Student-T distribution:

$$p(\tilde{r}_a|\tilde{\mathbf{X}}_a, \mathcal{D}) = \mathcal{T}(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}_a\mathbf{w}_n, \frac{b_n}{a_n}(\mathbf{I}_m + \tilde{\mathbf{X}}_a\mathbf{V}_n\tilde{\mathbf{X}}_a^T), 2a_n). \tag{5}$$

Note that the term $(b_n/a_n)\tilde{\mathbf{X}}_a\mathbf{V}_n\tilde{\mathbf{X}}_a^T$ varies depending on how close the test inputs are to the training data. For more details of the above model see [26]. In our experiments, we set $a_0 = 5$, $b_0 = 10$, $\mathbf{w}_0 = \mathbf{0}$, and $\mathbf{V}_0 = 5\mathbf{I}$.

Given the linear model that was learned through past data $\mathcal{D}$ and the basis functions, we would like to choose an action $\mathbf{a}$ such that some acquisition function $u(\mathbf{a}|\mathcal{D})$ is maximized. There are several popular choices of acquisition functions in the literature. Here, we adopt the expected improvement over the best candidate [25, 34, 8]. It is defined as follows:

$$\mathrm{EI}(\mathbf{a}|\mathcal{D}) = \mathbb{E}(\max(0, f(\mathbf{a}) - f(\mathbf{a}^+))|\mathcal{D}),$$

where f is the function we are trying to maximize and $\mathbf{a}^+$ is the best known action so far. This is a standard acquisition function for which asymptotic rates of convergence have been proved [9]. However, we point out that there are a few other reasonable alternatives, such as Thompson sampling [23] and upper confidence bounds (UCB) on regret [35]. A comparison among these options as well as portfolio strategies to combine them appeared recently in [17]. 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.

3.2 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 a few steps. 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 evaluate different parameter settings, we introduce super-transitions, which were first described by Neal in [28]. A super-transition defines the total number of leapfrog steps S taken in one run. 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 are needed because they enable us to evaluate the effectiveness of different parameter settings, with the total computational time fixed.

To ensure that the diminishing adaptation condition [30] is satisfied, we introduce p to denote the probability of a new proposal generated by our parametric bandit model being accepted. In each iteration, p is decreased by the following rule, $p = \lambda p$ where $0 < \lambda < 1$. As p goes to 0, we change our parameter settings less frequently. It is easy to check that the diminishing adaptation condition is satisfied. Because of the existence of vanishing regret bounds for bandit methods similar to ours, we conjecture that it is possible to side-step the need for introducing p as a mechanism to ensure vanishing adaptation. In the meantime, we include this mechanism for correctness. The full algorithm is presented in Algorithm 2.

Algorithm 2 Adaptive HMC with Bayesian Parametric Bandits

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

4 Application to Bayesian Neural Networks

We demonstrate 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.

In both experiments, we use cross-validation to construct the reward signal. As in classical n-fold cross-validation, we divide the data into n sets, train BNNs on $n - 1$ sets and test them in the remaining set. During each super-transition, we draw samples for each of the BNNs and calculate the average cross-validation error.

4.1 Robot Arm Data

The robot arm data set is a classical nonlinear regression benchmark [11]. The data is generated from the following model:

$$y_1 = 2.0\cos(x_1) + 1.3\cos(x_1 + x_2) + e_1 \tag{6}$$

$$y_2 = 2.0\sin(x_1) + 1.3\sin(x_1 + x_2) + e_2, \tag{7}$$

where e_1 and e_2 are independent Gaussian noise variables of mean 0 and standard deviation 0.05. The first 200 cases of the data are used for training and the last 200 cases are used for testing. Neal in [28] applied BNNs to this problem. For inference, he used HMC. The hyper-parameters of HMC were hand tuned.

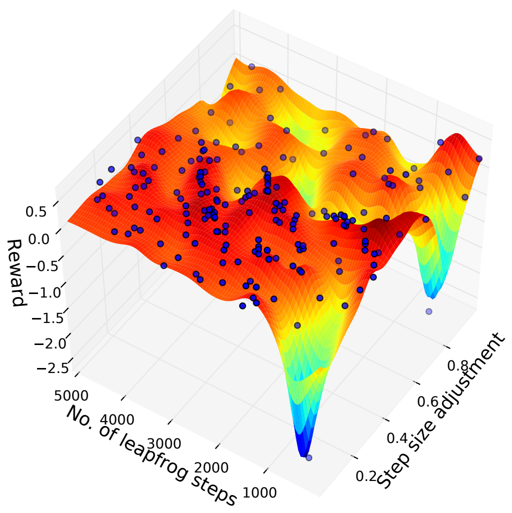
In the experiment that produced the best results, Neal used super-transitions of 32000 leapfrog steps to train a network of 16 hidden units for 150 super-transitions. We follow Neal on the structure of the network and train the network using super-transitions of 24000 leapfrog steps for 200 iterations. 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 tested on the validation part of the 200 training data. We

Table 1: Mean squared test error for the robot arm data set.

Method	**Mean Squared Error**
Mackay's (1992) Gaussian approximation with highest evidence	0.00573
Neal's (1996) HMC	0.00554
Reversible-jump MCMC with Bayesian model by Andrieu et al.	0.00502
Adaptive HMC	0.00494

normalize the averaged mean squared error to be within 0 and 1 and use it as the reward for the parametric bandit model.

The test set error (discarding the first 1500 samples) is summarized in Table 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 reversible jump MCMC. Figure 1 shows the mean of the 3D reward surface learned by the parametric bandit and the mean of the 2D reward function along a slice $L = 3520$. The figure also shows the value of the acquisition function along this slice.



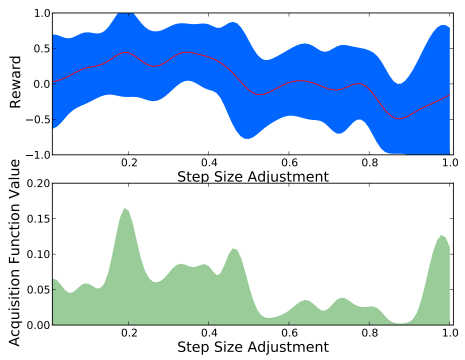


Figure 1: [Left] Learned mean reward surface for the Robot Arm data, with the tried parameters and corresponding rewards as dots. Areas of the hyper-parameter space with higher reward are explored more densely. [Right] The upper plot shows the mean reward function for a slice at $L = 3520$. The blue-shaded region indicates the one standard deviation confidence interval. The lower plot shows the corresponding acquisition function.

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 2003 [14]. 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 [27]. The entry that used only BNNs was placed second and achieved highly competitive results [14]. 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". For a detailed description of this model please refer to [27]. In his entry, Neal used in total 3200000 leapfrog steps. In our experiments we used 160

Table 2: Classification error on the validation set of the Dexter data set.

Method	**Classification Error**
New-Bayes-nn-sel	0.080
Adaptive HMC	0.074

super-transitions each with 20000 leapfrog 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.

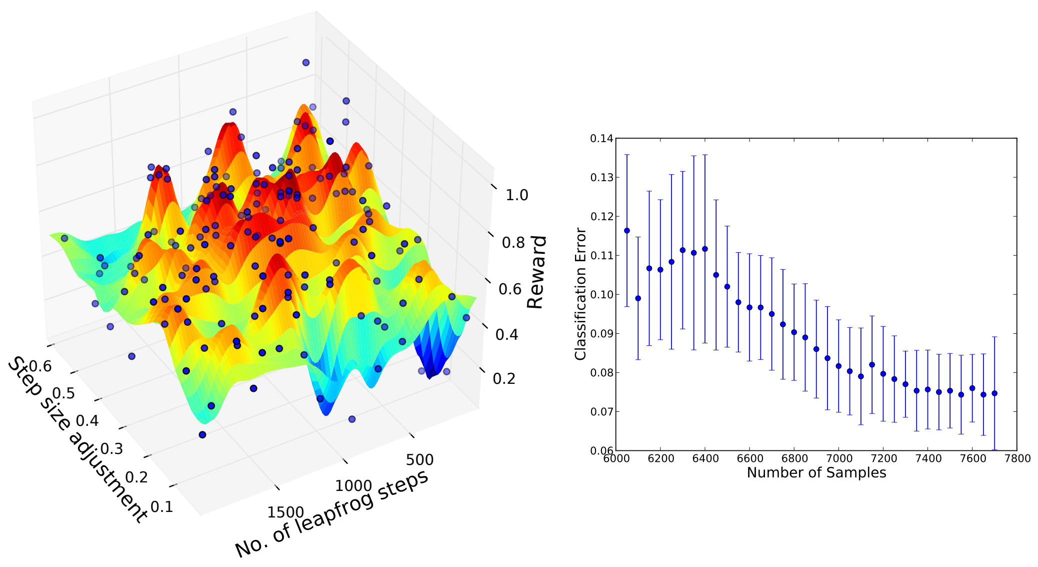


Figure 2: [Left] Mean reward function learned for the Dexter data set from the NIPS 2003 feature selection challenge. [Right] Average classification error on the validation set of the Dexter data set as a function of the number of MCMC iterations.

In addition to the training set, the competition also provides a validation set. The average classification error rate of all 10 networks on the validation set as well as Neal's result is provided in Table 2. Figure 2 depicts the 3D reward surface learned by the Bayesian parametric bandits approach, as well as, the classification errors on the validation set over time.

To assess whether we could combine the 10 neural networks 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. By using only training data alone, our method attained a 0.0385 classification error rate. Neal and Zhang's method with HMC had a 0.0510 error rate, while their best entry for this data set, using Dirichlet diffusion trees, achieved an error rate of 0.0390. 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.

5 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 general theoretical results that establish the ergodicity of adaptive MCMC schemes [30, 5]. In general, two sets of conditions are required for an adaptive MCMC algorithm to be ergodic. First, the adaptation has to vanish 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 [30], 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 [32] that Langevin diffusion, which is closely related to HMC, is geometrically ergodic. Thus one potential challenge would be to prove or disprove the geometric ergodicity of HMC. In [5], Atchadé et al. weakened the conditions required for a general adaptive MCMC algorithm to be ergodic. In their work, although the authors still require vanishing 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. As an alternative, if the HMC dynamics were defined on a compact space, then the proposed adaptive scheme would be ergodic.

6 Conclusion

In this paper we were able to show that parametric bandits, under a Bayesian treatment, can be effectively used to adapt the parameters of hybrid Monte Carlo. This 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 parameters, but that this can in fact lead to better results (as measured by NIPS competition standards).

This paper also introduced 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. 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. 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.

References

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

[6] Peter L. Bartlett, Varsha Dani, Thomas Hayes, Sham Kakade, Alexander Rakhlin, and Ambuj Tewari. High-probability regret bounds for bandit online linear optimization. In *Annual Conference on Learning Theory*, pages 335–342, 2008.

[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] Eric Brochu, Vlad M Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions. Technical Report TR-2009-023, University of British Columbia, Department of Computer Science, 2009.

[9] Adam D Bull. Convergence rates of efficient global optimization algorithms. Technical Report arXiv:1101.3501v2, 2011.

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

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

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

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

[14] 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.

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

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

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

[18] 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.

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

[20] 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.

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

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

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

[24] M.D. McKay, R.J. Beckman, and WJ Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

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

[26] K.P. Murphy. Machine learning: a probabilistic perspective. *Unpublished*.

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

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

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

[30] 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.

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

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

[33] 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.

[34] Matthias Schonlau, William J. Welch, and Donald R. Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, 34:11–25, 1998.

[35] 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.

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

[37] 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.