STOCHASTIC GRADIENT BARKER DESCENT

Pramodh Gopalan
IITK
Indian Institute of Technology Kanpur
{pramodh}@iitk.ac.in

Dootika Vats
IITK
Department of Mathematics and Statistics
{dootika}@iitk.ac.in

1 Introduction

Markov chain Monte Carlo (MCMC) is a large family of sampling algorithms, which allow you to efficiently sample from high dimensional densities in a reasonable amount of time. These algorithms have played a significant role in statistics, computer science and physics over the last two decades. MCMC provides a strategy to explore state space $\mathcal{X}$ using a Markov chain mechanism. The mechanism is so constructed that it spends time in regions of high density, from which we can reliably estimate statistics about the distribution. Here, we focus on MCMC algorithms from a bayesian perspective, wherein we are able to obtain the posterior distributions in an unnormalized form. That is, for a parameter θ, prior data y, and likelihood $f(y|\theta)$ we know $\pi(\theta|y)$ given by

$$\pi(\theta|y) \propto \pi(\theta) f(y|\theta)$$

Several recent MCMC such as Metropolis Adjusted Langevin Algorithm(MALA) or Hamiltonian Monte Carlo(HMC) make use of gradient of the log posterior. In cases with large amount of data, calculating the gradients and acceptance rates might prove to be computationally expensive. [1] propose an approximate MCMC algorithm called Stochastic Gradient Langevin Dynamics(SGLD), which bypasses the need for computation of an acceptance ratio and uses stochastic gradients in place of full gradients. They also provide a weak convergence proof that the above sampling does converge to the actual posterior. Their paper lead to the proposal of several other Stochastic Gradient MCMC(SG-MCMC) methods such as SG-HMC(Stochastic Gradient Hamiltonian Monte Carlo), SG-RLD(Stochastic Gradient Reimannian Langevin Dynamics). The field of SGMCMC algorithms has a wide range of applications, especially in modern day Bayesian Deep Learning, thus proving to be an active area of research.

Recent work by [2] introduces Barker's proposal, a new MCMC algorithm which proves to be effective as gradient based methods like MALA, while being robust to tuning parameters. We investigate the Stochastic Gradient version of this, (dubbed as SGBD) in this project. We note that it is easier to tune than SGLD, while still providing low bias approximations of the posteriors. We first showcase existing MCMC and SGMCMC methods, then introduce SGBD. We show it's advantages over existing methods through a series of experiments on different datasets.

2 Markov chain Monte Carlo

A series of random variables $X_1, X_2 \dots X_n \in \mathbb{R}^d$ is a first order Markov chain if the following condition holds:

$$p(X_{n+1}|X_n, X_{n-1}, \dots X_1) = p(X_{n+1}|X_n)$$

We can specify the possible choices of X_{n+1} using transfer probability, $p(X_{n+1}|X_n)$.

Markov chain Monte Carlo combines the theory of Markov chains and Monte Carlo. We use a Markov chain to sample from our distribution of interest, and the Monte Carlo method to approximate an expectation that is computationally hard to exactly determine. In MCMC, we require the Markov chain so generated to be *reversible* (it has an invariant distribution) and *ergodic* (it converges to the invariant distribution regardless of the choice of the initial distribution $p(X_1)$). The samples generated by an MCMC algorithm are not independent of each other, since each state depends on the previous state. We require a *good* MCMC sampler to have low correlation in the samples generated by it, thus leading to lesser variance of the Monte Carlo estimate. One way to judge this is through the usage of ACF

Algorithm 1: The Metropolis-Hastings algorithm

Data: The proposal distribution $q(X|X_t)$
Set $X_1 \sim p(X_1)$
for $t = 2$ **to** n **do**
  Draw $X' \sim q(X|X_t)$
  Compute acceptance ratio α
  Draw $u \sim U[0,]$
  if $u < \alpha$ **then**
    $X_{t+1} = X$
  else
    $X_{t+1} = X_t$

(Auto Correlation Function). Another way is to monitor the number of samples taken to come "close" to the sampling distribution, referred to as mixing time. We say a MCMC algorithm mixes better if the samples generated have a lower mixing time. A lower mixing time also leads to lower correlation between the samples. For more analysis of the output generated by MCMC algorithms, refer to [3].

Let us say we have a Markov chain with a kernel $Q(x, A) = \int_A q(x, y)dy$ and a distribution $\pi(x)$ with support $\mathcal{X} \in \mathbb{R}^d$. $\pi(x)$ is invariant to the kernel Q iff it follows the detailed balanced condition given by

$$\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)} := t(x, y) = 1 \tag{1}$$

holds for all $x, y \in \mathcal{X}$ such that $\pi(x)q(x, y) > 0$ and $\pi(y)q(y, x) = 0$ elsewhere. Now consider the scenario when $\pi(x)$ is not the invariant distribution for $Q(x, A)$; Equation 1 does not hold here. We can create a new kernel $p(x, y)$ by setting

$$p(x, y) := g(t(x, y))q(x, y)$$

where $g(t)$ satisfies

$$g(t) = tg(1/t) \tag{2}$$

It is trivial to see that $p(x, y)$ does satisfy Equation 1. The function $g(t)$ is referred to as the balacing function in this paper. Popular MCMC algorithms like the Metropolis-Hastings algorithm use a particular balancing function to enforce a Markov chain to have specific invariant distribution. We now present a few MCMC and SGMCMC algorithms relevant to this paper.

2.1 Metropolis-Hastings algorithm

The MH algorithm is one of the most widely applicable MCMC algorithm. Given the current sample X_t, we obtain the next sample X from the proposal distribution $q(X_{t+1}|X_t)$ of the user's choice. We pass the proposal through an accept-reject step, where the acceptance is given by:

$$\alpha = \textbf{min}\left(1, \frac{\pi(X)q(X_t|X)}{\pi(X_t)q(X|X_t)}\right)$$

where $\pi(X)$ is the unnormalized target distribution. If accepted, we update the chain as $X_{t+1} = X$ or maintain it as $X_{t+1} = X_t$ if rejected. If the proposal distribution is a gaussian, the resulting MH algorithm is known as a RWM (Random Walk Metropolis). In RWM, we regulate the *step size* of the proposal through controlling the variance of the gaussian. Having a small step size leads to more accepts but with poor mixing, whereas having a large step size might lead to lesser acceptances. In practice, we aim at 44% acceptance rate if the target distribution is 1-D, and 22% if it is greater than 5-D. The MH algorithm proves to be sufficiently robust to tuning parameters; it's spectral gap decays in a polynomial fashion, as shown by [2]. The pseudo-code is described in Algorithm 1.

However, MH algorithm does not give preference to directions of high acceptance probability which can be found through the gradient of the target distribution. This leads to lower acceptance rates when compared to gradient based MCMC methods. The efficiency of this method decays as d^{-1}, where d is the dimension of X_ts, which is worse than gradient based methods.

2.2 Metropolis Adjusted Langevin Algorithm

The MALA uses gradient information of the target proposal to introduce better proposals distribution. Widely considered as the state of the art in MCMC methods, it uses Langevin Dynamics to simulate a stochastic process with the target distribution as it's stationary distribution. The dynamics are modeled by the following SDE.

$$dX_t = \frac{1}{2}\nabla_x \log(\pi(X))dt + dB_t$$

Where B_t denotes a Brownian motion of the appropriate dimension. This SDE is solved using a first order Euler approximation given by:

$$X_{t+1} = X_t + \frac{h}{2}\nabla_x \log(\pi(X)) + \sqrt{h}Z$$

Where Z is a standard gaussian random variable of the appropriate dimension. The Euler discretization is not exact; the error of approximation is controlled through h. As h decreases, the approximation error decreases too. A large value of h leads to mismatch of the actual stationary distribution π and the equilibrium distribution of the discretization. To mitigate this error, we add a Metropolis Accept-Reject Step with the proposal distribution as:

$$q(X|X_t) = \mathcal{N}(X|X_t + \frac{h}{2}\nabla_{X_t} \log(\pi(X)) + \sqrt{h}Z)$$

Gradient based methods such as MALA, are considered to be difficult to tune when compared to Algorithms like MH. In fact, [2] states that the spectral gap of MALA reduces exponentially. When the tails of the proposal decay too quickly, the MALA algorithm generating unreasonable proposals. The efficiency of this method decays as $d^{\frac{-1}{3}}$, where d is the dimension of X_ts, which is better than MH.

2.3 Barker's Proposal

To have an invariant distribution, any Markov chain needs to follow the detailed balance condition. The MH algorithm implements this balance condition through it's acceptance ratio and the following if-else conditions. However, there also exists another approach through the use of continuous time Markov jump process, where jumps occur with the intensity

$$\lambda(x) = \int_{-\infty}^{\infty} g(t(x,y))q(x,y)dy$$

Note that the above integral is not tractable in general. To resolve this, we choose a symmetric q, and $g(t) = \frac{t}{1+t}$. Along with this, we do a first order approximation of $t(x,y) \approx \exp(y-x)\nabla \log \pi(x)$. Then, the complicated integral reduces to a constant. The jump kernel so generated is called the barker's proposal introduced in [2], [4].

The barker's proposal maintains the dimensional scaling $d^{\frac{-1}{3}}$ of gradient based methods, shares the robustness to tuning as MH, and generates sensible proposals in light tailed targets. While MALA adds a symmetric gaussian noise to the proposal, Barker's proposal skews the normal in the direction of the gradient. This maintains robustness to tuning. The spectral gap of Barker's Proposal decreases polynomially, which is the same as the MH algorithm. In fact, [2] tells us that the balancing function $g(t) = \frac{t}{1+t}$ used in Barker's proposal is the only choice of $g(t)$ that gives us skew-symmetric kernels. One specific way is as follows. Given a current point X_t, we draw a proposal Z from a Gaussian Kernel. We calculate $b(X,Z)$ given as:

$$b(X,Z) = \frac{1}{1 + e^{-Z^T \nabla_X \log(\pi(x))}}$$

From this, we generate the next point X_{t+1} as:

$$X_{t+1} = \begin{cases} X_t + Z & \textbf{with prob. } b(X_t, Z) \\ X_t - Z & \textbf{with prob. } 1 - b(X_t, Z) \end{cases}$$

Now, it is not necessary that we always sample Z from a gaussian kernel. In fact, [2] does not provide any other reason other than convenience for using a gaussian kernel. [5] however, derives the best noise for the balancing function discussed above to be a Rademacher Distribution, which is extremely surprising. Since the Rademacher kernel does not always lead to π-reducibe kernels, the authors settle for a bimodal gaussian with means at $\pm\sqrt{1-\sigma^2}$, and with variance σ^2. They also show that they have almost similar performance.

3 Stochastic Gradient MCMC

3.1 Drawbacks of exact MCMC methods

The main problem of MCMC methods in bayesian settings is realised when we use very large datasets. For example, in algorithms like MH, we require the computation of posterior at every iteration for calculating the acceptance ratio, which becomes prohibitive in cases with large amounts of data. In gradient based methods, we also require the calculation of gradients over the whole datasets, which is costly too. Other methods like Gibbs sampling are free from the calculation of accept-reject ratio, but they are limited in general applicability.

One way to curb this is through the usage of stochastic gradients. We use a minibatch (set of randomly picked samples without replacement from dataset) in place of the whole dataset to compute gradients. In order to remove the computational cost of calculating acceptance ratios, we don't calculate them. This induces some error in the approximation of the target distribution. These set of methods, referred to as SGMCMC, can also be viewed from the lens of bias-variance tradeoff. Normal MCMC methods are unbiased samples, but contain variance that decrease with the number of sampling points. SGMCMC methods however, produce biased samples. Due to the reduced computational cost, they are able to produce more samples which reduce the variance of the Markov chain, thereby leading to a lesser mean squared error.

3.2 Stochastic Gradient Langevin Dynamics

As discussed previously, a bayesian inference problem with large dataset(of size N) is made more tractable through the usage of SG-MCMC methods. SGLD or Stochastic Gradient Langevin Descent is one of them. It is closely related to the MALA sampling method. While MALA is an asymptotically unbiased MCMC algorithm at the cost of increased computation, SGLD makes two major changes discussed previously:

- Exact gradients in MALA is replaced with stochastic gradients with batch size of $n << N$, sampled at random without replacement at every iteration of the algorithm.
- Omit the accept-reject rule. The SGLD algorithm accepts all proposals. The original paper proved that SGLD samples from the target distribution asymptotically, given that the step sizes decrease.

With these modifications, the equation for the dynamics in the bayesian setting now becomes:

$$X_{t+1} = X_t + \frac{h_t}{2}\left(\nabla_{X_t}\log(p(X_t)) + \frac{N}{n}\sum_{x\in MB}\nabla_{X_t}\log(p(x|X_t))\right) + \sqrt{h_t}Z$$

where Z is a standard gaussian variable of appropriate dimension.

Even though in theory, SGLD samples from posterior as $h_t \to 0$, such step sizes make mixing poor. In practice, the annealing is stopped at a finite value so that mixing can still occur, with lesser error. It also reduces the amount of burn-in samples, since it efficiently moves towards regions of higher density. It finds wide use in fields with large data, such as probabilistic neural networks and Large Scale Recommender systems. As a result, several improvements have been proposed to this algorithm.

SGLD also faces some disadvantages. Like with MALA, a large value of h_t might lead to poor accuracy of the algorithm, due to approximation errors. It also suffers from poor mixing, since we need to reduce the step size to get reliable samples. A corollary is that in spaces where the log density is highly curved, SGLD is not able to adapt well. A variety of methods [6] [7] have been proposed to alleviate this partially. In standard MCMC algorithms with Accept-Reject steps, if an example is proposed out of support, that is immediately rejected. However, with SGLD, this is not possible, if a sample is proposed out of support, it is accepted which might lead to loss of efficiency and huge numerical errors (since $\nabla_x \log(p(x))$ is not defined out of support). Hence, SGLD requires an unconstrained state space. Sometimes, the discretization of SDE might not be stable, thus leading to bad performance of SDE. Note that inorder to get robust estimates of gradients, n needs to scale linearly with N.

3.3 Stochastic Gradient Barker Dynamics

Algorithm 2: The Stochastic Barker's Algorithm

Set $X_1 \sim p(X_1)$
for $t = 2$ **to** n **do**
  Draw $X' \sim \mathcal{N}(X|X_t)$
  Compute $b(X', X_t) = \frac{1}{1+\exp^{X'^T \nabla_{X_t} \log(\pi(x))}}$
  Draw $u \sim U[0, 1]$
  if $u < b(X', X_t)$ **then**
    $X_{t+1} = X_t + Z$
  else
    $X_{t+1} = X_t - Z$

In this section, we propose the Stochastic Gradient Barker's Algorithm. It does not use the accept reject step, and uses stochastic gradients to approximate the full gradient. A pseudo code of SGBD is illustrated in Algorithm 2. It provides good approximations of the posterior, even when the step size is not optimal. As a consequence, it allows for better mixing rates when compared to SGLD. Note that in SGBD, only the direction of the move is decided by the magnitude of the gradient, not the jump size itself. An advantage of this is that it allows for defined behavior when the proposed point is out of support. We now showcase it's effectiveness through experiments on various datasets. We also note that the SGBD algorithms is just a baseline, and can be used as drop in replacements in other improvements made to SGLD, such as [8]

4 Experiments

In this section, we compare SGBD against methods like SGLD. To compare against SGMCMC methods, we use two metrics, Kernel Stein Density and Effective Sample size. Both on their own prove to be insufficient to compare SGMCMC methods. When used togather, they give a more comprehensive comparison between the methods.

4.1 Comparison Metrics

1. **Kernel Stein Discrepancy** Kernel Stein Discrepancy(KSD)[9] measures how well a set of samples $X_1, X_2 \dots X_n$ with an empirical distribution $\tilde{\pi}$ approximates π, the actual distribution. Mathematically put,

$$d_{\mathcal{H}}(\tilde{\pi}, \pi) = sup_{h \in \mathcal{H}} |E_{\pi}[h(X)] - E_{\tilde{\pi}}[h(X)]|$$

   for a class of functions $\mathcal{H}$. Note that in SGMCMC, we don't know π. [9] chooses a class of functions $\mathcal{H}$ such that $E_{\pi}[h(X)] = 0$ for all $h \in \mathcal{H}$. Kernel Stein Discrepancy also detects non convergence of SGMCMC. However, calculating KSD scales quadratically as the number of samples n, which becomes prohibitive quickly.

2. **Effective Sample Size**
   Effective Sample Size(ESS) defines an exchange rate in between dependant samples generated from Markov chain and independant draws. Given samples $X_1, X_2, \dots X_n$, it is calculated as

$$\textbf{ESS} = \frac{n}{1 + 2\sum_{k=0}^{\infty} \rho(k)}$$

   Where $\rho(k) = corr(X_i, X_{i+k})$, is the k lag autocorrelation which is independant of i given a stationary chain. Calculation of ESS scales linearly with the number of samples n, and hence is less expensive when compared to KSD. however, it is not an appropriate method to measure effectiveness of SGMCMC methods. A higher ESS(which is better) corresponds to a higher step value. In the case of SGMCMC, higher step size is not always better due to the bias in approximation.

We show three experiments and compare SGBD with SGLD on diagnostics such as Kernel Stein Discrepancy(KSD) and Effective Sample Size(ESS). We first present case studies on the univariate Normal distribution $\mathcal{N}(0, 1)$ where both the models perform well. Next, we showcase the effectiveness of SGBD in a constrained support system like the Chi

square distribution. Finally, we show SGBD performs better than SGLD on the Arrythmia Dataset, which has skewed posteriors and imbalanced data.

4.2 Normal Distribution

We try and draw samples from the standard normal distribution $\mathcal{N}(0, 1)$. We compare SGBD and SGLD on a variety of step sizes, and plot the appropriate posteriors in figure 1. We also plot metrics like KSD and ESS for these step sizes, shown in figure 2

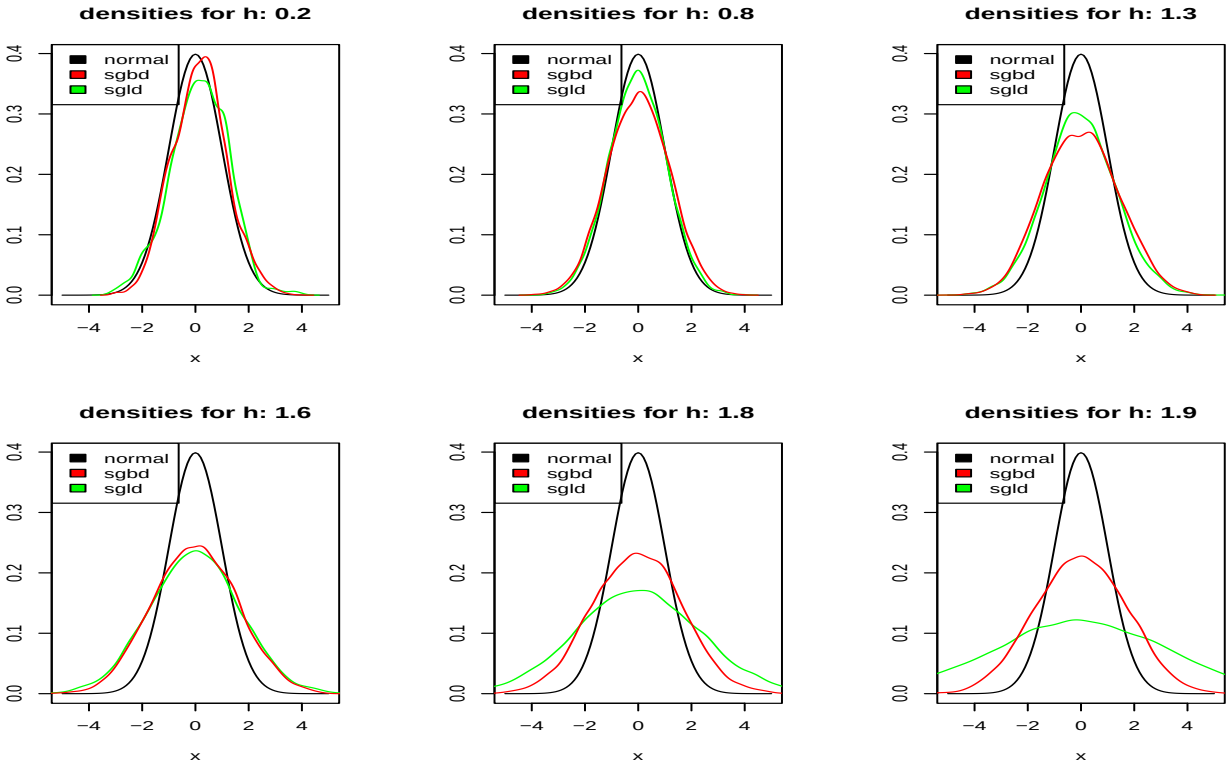


Figure 1: Density plots for various step sizes as labelled. SGLD and SGBD perform equally well until a certain threshold. Then, we clearly see SGBD produce better approximations at larger step sizes when compared to SGLD

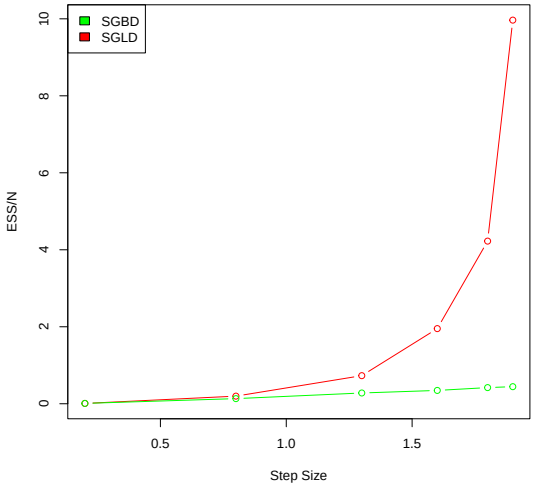
4.3 Chi Square Distribution

We try and draw samples from the Chi square distribution with 2 degrees of freedom. We compare SGBD and SGLD on a variety of step sizes, and plot the appropriate posteriors. Since Chi square distribution has positive support, we define the gradient of log density at points < 0 to be a large poisitive number. To showcase the effectiveness, we plot the density plots across a range of step sizes in Figure 3, and the fractions of out of bound samples(i.e, samples less than 0) proposed by both SGBD and SGLD, shown in Figure 4

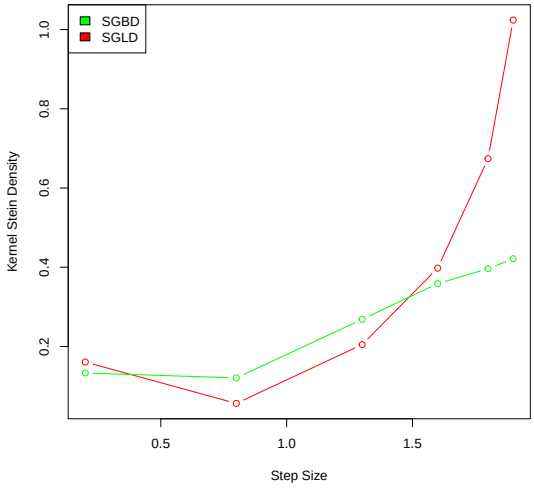
The results shown in Figure 4a are abnormal. We see that SGLD proposes very less samples that are out of bounds(it actually only proposes one sample that is OOB), but still the density plots are extremely off. This is because once a samples < 0 is attained, the next sample proposed would be large and positive due to the definition of the gradient above. This proposal is in a low density area of the chi-sq distribution, and remains oscillating there. Hence, it generates poor density plots. This is seen clearly in the time series plot of the SGLD chain at $h = 1$, in Figure 4b.

4.4 Arrythmia Dataset

We procure arrhythmia dataset from the UCI machine learning repository available here. We choose the same experimental settings from [4], and run SGBD and SGLD algorithms with diagonal preconditioners calculated from



(a) Effective Sample Size(higher is better) for different step sizes



(b) Kernel Stein Discrepancy(lower is better) for different step sizes

Figure 2: We can see that at non optimal step sizes, SGBD performs better than SGLD, with lesser KSD

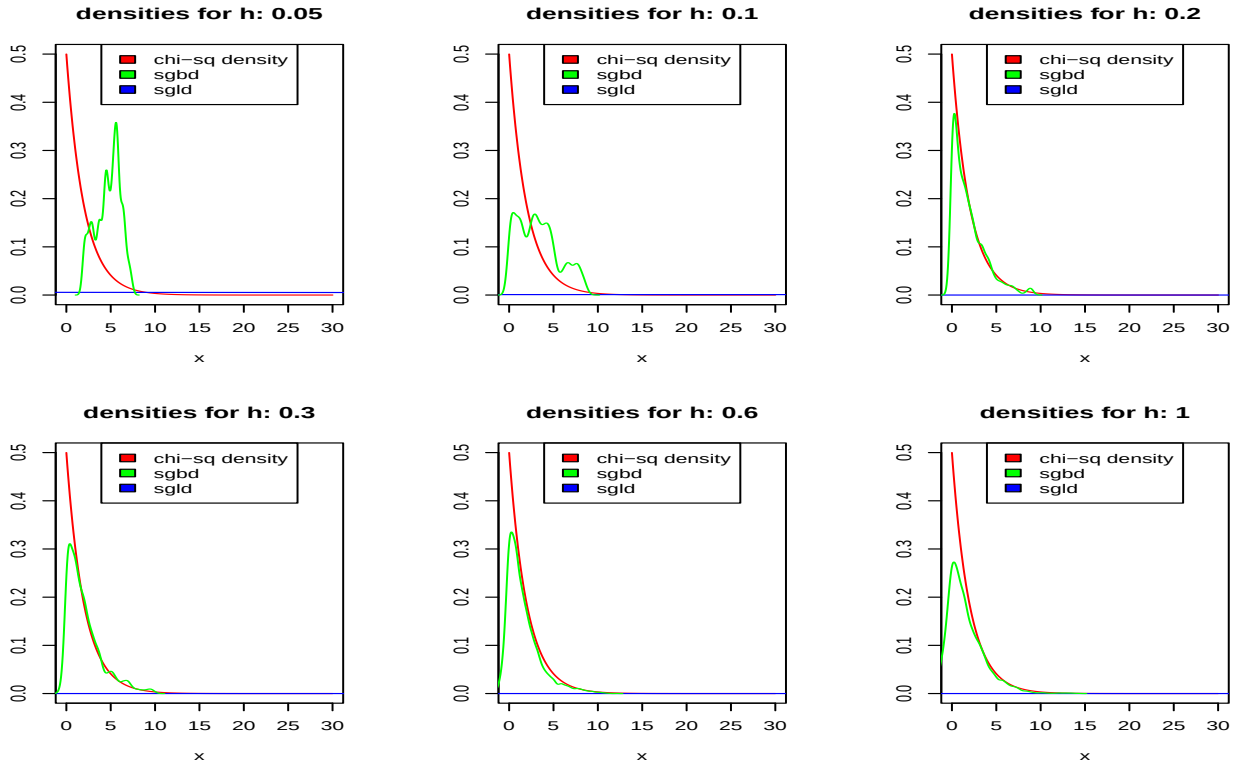
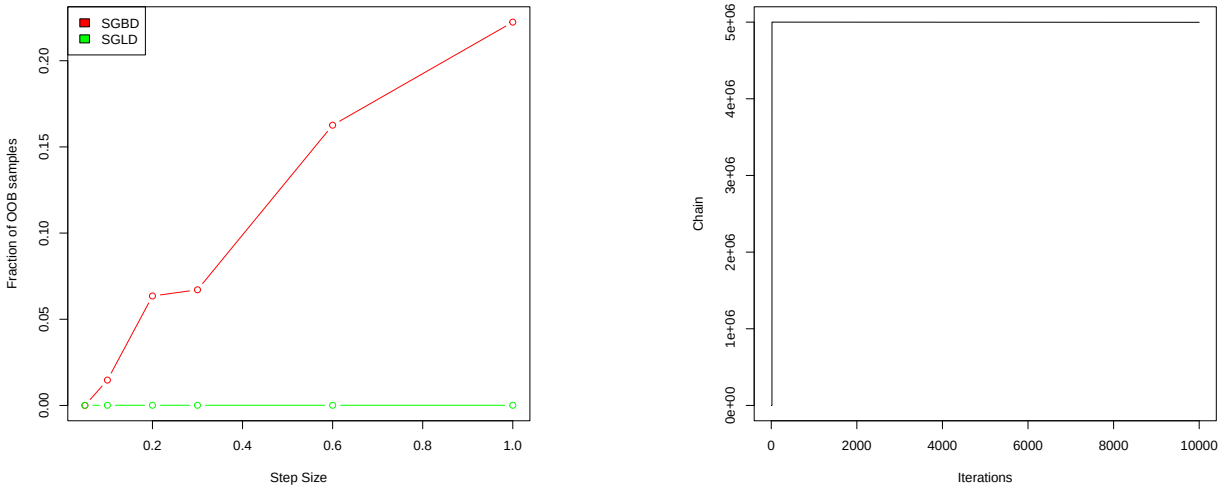


Figure 3: Density plots for various step sizes as labelled. In all step sizes, SGBD performs better than SGLD

running pilot chains. For the baseline posteriors, we run an Adaptive MCMC algorithm with the Barker's proposal to compare the posteriors generated by SGBD and SGLD. We show the density plots of different components across various step sizes in Figure 5. Clearly, SGBD is more robust to step size changes than SGLD.



(a) Fraction of out of bounds for different step sizes

(b) A time series plot of SGLD when $h = 1$

Figure 4: A few plots for the Chi square distribution with 2 degrees of freedom

5 Conclusion and Future Work

In this paper, we introduce Stochastic Gradient Barker Dynamics, a new SGMCMC routine based on the barker's proposal. We highlight it's advantages over traditional gradient based SGMCMC methods such as SGLD, and showcase it's efficacy through experiments. There is a lot of future work possible in this area. A proof of convergence of SGBD could be extremely helpful to establish a theoretical foundation. One more line of work to pursue is the comparison of two SGMCMC methods. Kernel Stein Discrepancy is one method of judging samples from the SGMCMC algorithm, but do not detect non convergence. This is an active field of work, and recent work such as [10] seem to be a promising step in the right direction.

References

[1] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[2] Samuel Livingstone and Giacomo Zanella. The barker proposal: combining robustness and efficiency in gradient-based mcmc, 2020.

[3] Dootika Vats, James M. Flegal, and Galin L. Jones. Multivariate output analysis for markov chain monte carlo, 2017.

[4] Max Hird, Samuel Livingstone, and Giacomo Zanella. A fresh take on 'barker dynamics' for mcmc, 2021.

[5] Jure Vogrinc, Samuel Livingstone, and Giacomo Zanella. Optimal design of the barker proposal and other locally-balanced metropolis-hastings algorithms, 2022.

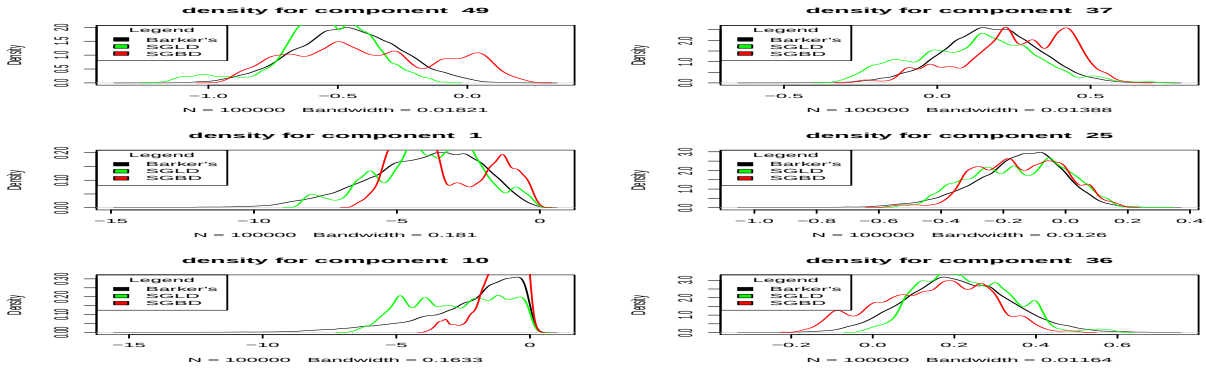
[6] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks, 2015.

[7] Nicolas Brosse, Alain Durmus, and Eric Moulines. The promises and pitfalls of stochastic gradient langevin dynamics, 2018.

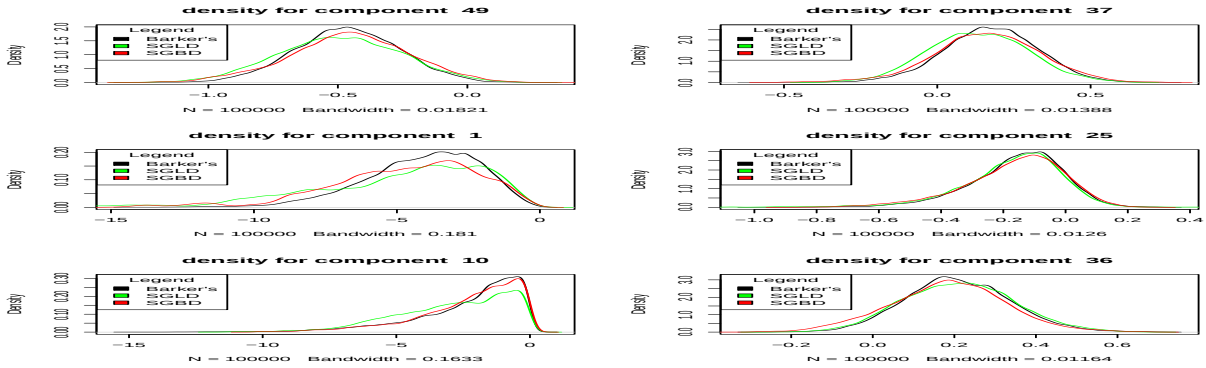
[8] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning, 2019.

[9] Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. March 2017.

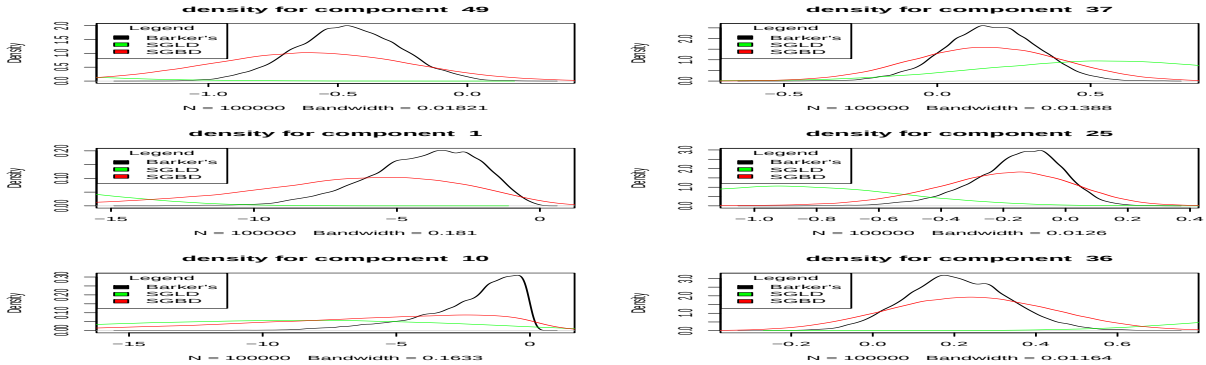
[10] Bokgyeong Kang, John Hughes, and Murali Haran. Diagnostics for monte carlo algorithms for models with intractable normalizing functions, 2021.



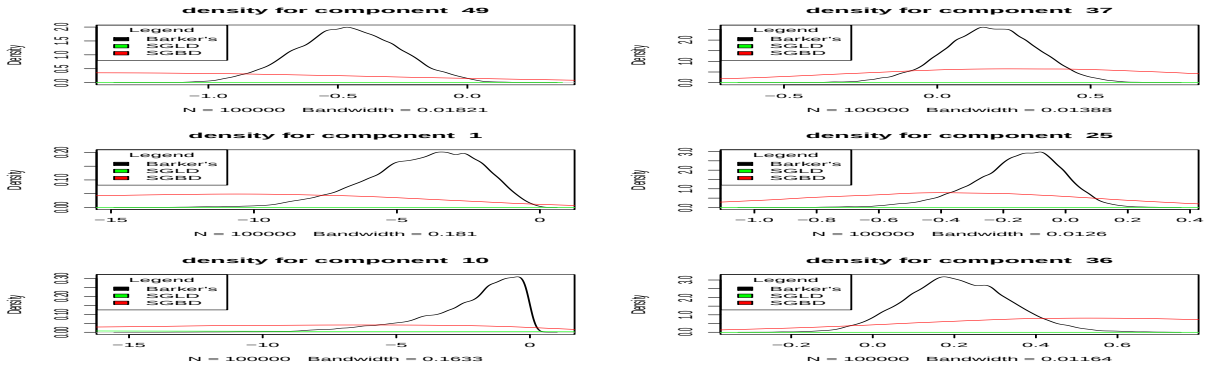
(a) Densities for different components at $h = 0.02$



(b) Densities for different components at $h = 0.1$



(c) Densities for different components at $h = 1$



(d) Densities for different components at $h = 3$

Figure 5: Densities for different components at different step sizes.