Scaling up MCMC for Bayesian inference using adaptive data subsampling

Ayoub El hanchi

May 1, 2019

1 Introduction

Much of the success of modern machine learning and statistics can be attributed to the abundance of data and computational resources. Arguably the next most important factor is the development of reliable and scalable optimization algorithms. A large literature is available on the subject, dealing with a large class of problems encountered in practice, and providing non-asymptotic convergence guarantees. First order methods based on the gradient of the objective function have emerged as the clear winners among optimization methods due to their fast convergence rate, scalability, and ability to escape saddle points in the non-convex case. Crucially, the use of controlled stochasticity allowed many of these algorithms to be scaled for the needs of modern problems. Stochastic gradient descent (SGD) (and its variance reduced extensions) is perhaps the most popular such algorithm due to its fast convergence rate which is independent of the dimension of the problem and sublinear in the size of the data.

In the sampling literature, which is especially relevant to Bayesian inference, such development did not occur at the same rate. Until recently, little attention was given to non-asymptotic bounds on the convergence of sampling algorithms. Furthermore, all the major popular algorithms for sampling have a linear complexity dependence on the size of the data, and various conjectures suggest a polynomial dependence on the dimension of the problem, making them unsuitable for current large scale problems.

Perhaps one of the main causes of this difference is the focus on unbiased algorithms in the sampling literature. In optimization, first order methods are provably unbiased and converge to a stationary point of the objective function, and so are the stochastic versions of these methods. On the other hand, first order methods in the sampling literature are biased and require a Metropolis-Hastings correction step to preserve the right limiting distribution.

The correction step requires time linear in the size of the data, so a direct attempt to use stochasticity in this case results in a biased algorithm.

In light of this, a new framework for approximate sampling has emerged. The main idea is to allow for a user-specified bias-controlling parameter $\epsilon > 0$, and to construct a sampling algorithm that has a bias of at most ϵ in an appropriately chosen metric. The hope is that such algorithms would be more robust to the introduction of stochasticity allowing them to be used in large problems.

Focusing more specifically on problems of Bayesian inference, and interpreting MCMC algorithms as estimation procedures, this has perhaps a more appealing interpretation. An important idea in statistical estimation is that of the bias-variance trade-off. We can think of current MCMC algorithms as unbiased estimators that suffer from high variance (due to their high computational cost), whereas algorithms that fit in the above framework allow some non-zero bias while (hopefully) significantly reducing the variance.

It is important to note that while these two views on sampling algorithms (for MCMC or for pure sampling) are closely related, there is one important qualitative difference. While in a pure sampling problem we only care about reaching the target distribution as fast as possible, in MCMC we also need to have a chain with low autocorrelation, since asymptotically the variance of our estimator is the integrated autocorrelation time of the chain. Thus the fastest sampling algorithm might still perform very poorly if used for MCMC estimation if it suffers from high autocorrelation. To my knowledge, this has yet to be adressed from a theoretical perspective.

Two algorithms that fit in this framework are the stochastic Metropolis-Hastings (SMH) algorithm [15] and the stochastic gradient Langevin dynamics algorithm (SGLD) [24] (and its variants). Both are based on data subsampling and enjoy a sublinear per iteration cost in the size of the data. On the other hand, the relationship between the bias-controlling parameter ϵ and the bias of the algorithm is not clear, and our experiments (and others) show that these algorithms can be extremely biased or even fail altogether on fairly standard problems.

In this report we study these two algorithms and some of their extensions. We propose a new general-purpose importance sampling strategy, and use it to construct new versions of these algorithms. We use a logistic regression model to test our new algorithms and compare their performance with the original versions. We also test our algorithm against the standard MCMC algorithms and show that for a constrained time budget, our algorithms perform better.

2 Notation and general comments

We refer to the data by $x = \{x_i\}_{i=1}^N$. We write our posterior density function as $\pi(\theta|x) \propto e^{-U(\theta)}$, and refer to the (negative) log-posterior by $U(\theta)$ and its gradient by $\nabla_\theta U(\theta)$. We write $U_i(\theta)$ and $\nabla_\theta U_i(\theta)$ to refer to respectively the log-posterior and gradient corresponding to the i^{th} data point.

We let $\{\theta_t\}_{t\in\mathbb{N}}$ be our Markov chain. The discretized Langevin dynamics are given by, for any $t > 0$:

$$\theta_{t+1} = \theta_t + t\nabla_\theta U(\theta) + \mathcal{N}(0, 2t)$$

we refer to the algorithm that implements this Markov chain and uses the full gradient by LD (for Langevin dynamics).

All experiments, except when mentioned otherwise, were conducted on the MNIST dataset using a logistic regression model on the first two digits (0 and 1) with a normal prior $\mathcal{N}(0, I_{d\times d})$. We extract the 10 first principal components from the dataset using PCA. The final size of the dataset is $N = 14780 \times 10 = d$. All stochastic algorithms use a batch size $S = \lceil\sqrt{N}\rceil = 122$. All algorithms were run for 5000 iterations starting at the mode of the distribution (except when comparing the standard algorithms with the stochastic ones, I ran the stochastic one for many more iterations to increase their oracle use count so that their evolution is visible). We also run the Metropolis adjusted Hamiltonian Monte Carlo algorithm for 50000 iterations, and took its output as the ground truth against which we compare all other algorithms.

The 4×4 matrix plots show on the diagonal the kernel density estimates of the marginal distributions, and a scatter plot of the samples off the diagonals. The top two dimensions are in the direction of largest variance, while the bottom two dimensions are in the direction of smallest variance. The distribution plots for variance reduced algorithms (SAGA and modified SAGA) were made only after the algorithm has forget its initialization (after running it for 5000 iterations).

3 Stochastic Metropolis-Hastings

The Metropolis-Hastings (MH) algorithm is the backbone of almost all current MCMC algorithms. It uses an accept-reject scheme to enforce time reversibility of the Markov chain by computing an acceptance ratio, which directly depends on the value of the log-posterior for both the current iterate and the candidate. This requires a full data computation at each step, making it inefficient for large datasets.

Given that the MH algorithm entirely relies on this zeroth-order information, it is very sensitive to errors. Naive subsampling strategies are almost guaranteed to fail given that

a single extreme estimate of the log-posterior for some candidate can lead the chain to be stuck at that point for multiple iterations (or the rest of the simulation).

3.1 The stochastic MH step

The main idea in the stochastic Metropolis-Hastings (SMH) algorithm is to treat the MH accept-reject step as a statistical decision problem. Given the bias-controlling parameter $\epsilon > 0$, we build an approximate MH step which, with probability at least $(1 - \epsilon)$, makes the correct decision.

We start by rewriting the acceptance condition for $u \sim \mathcal{U}([0, 1])$:

$$
\begin{aligned}
& u \leq \frac{e^{-U(\theta)}}{e^{-U(\theta_t)}} \\
& \Leftrightarrow u \leq e^{U(\theta_t) - U(\theta)} \\
& \Leftrightarrow \log u \leq U(\theta_t) - U(\theta) \\
& \Leftrightarrow U(\theta) \leq U(\theta_t) - \log u
\end{aligned}
$$

Let $\mu := U(\theta_t) - \log u$. We assume that our estimate of the log-posterior $U(\theta_t)$ of the current iterate is exact and that the central limit approximation holds for our estimator $\hat{U}(\theta)$. Under these assumptions we can use the one-sample t-test to test the hypotheses: $H_1 : U(\theta) > \mu$ and $H_2 : U(\theta) < \mu$. We keep computing $\hat{U}(\theta)$ with larger and larger batches of data until the test is conclusive, at which point we make our decision.

The full procedure is described in Algorithm 1. ϕ_{n-1} refers to the cumulative distribution function of the t-distribution, $\hat{U}(\theta)$ is the sample mean, and $\hat{\sigma^2}$ is the sample variance multiplied by the correction term $(1 - \frac{n}{N})$ to account for sampling without replacement.

3.2 Experiments with SMH

To check the performance of the SMH algorithm, we first ran the full MH algorithm with proposal $\mathcal{N}(0, \sigma^2 I_{d \times d})$ for $\sigma^2 = 0.01$ and obtained an acceptance rate of ≈ 0.35. We then ran the SMH algorithm with the same proposal and $\epsilon = 0.01$. As figure 1 shows, the same problem that we were trying to avoid persists: the chain encounters an extreme estimate of the log posterior and gets stuck at the corresponding point. Figure 2 shows the distribution of the estimator $\hat{U}(\theta^*)$ at the mode of the distribution θ^*. We observe that the CLT assumption does not hold, and that the distribution is highly irregular and has a heavy tail.

Algorithm 1 Stochastic MH step

Input: $\theta, \epsilon, U(\theta_t), S, \{x_i\}_{i\in[N]}$

Output: *accept*

$\hat{U}(\theta) \leftarrow 0$, $\hat{\sigma}^2 \leftarrow 0$
$used \leftarrow \emptyset$, $n \leftarrow 0$
$\alpha \leftarrow 1$
Draw $u \sim \mathcal{U}([0,1])$
$\mu \leftarrow U(\theta_t) - \log u$
while $\alpha \geq \epsilon$ **do**
    Draw mini-batch M of size $\min\{S, N-n\}$ from $\{x_i\}_{i\in[N]} \setminus used$ without replacement
    $used \leftarrow used \cup M$, $n \leftarrow |used|$
    Compute $\hat{U}(\theta)$, $\hat{\sigma}^2$ using $used$
    $t \leftarrow \frac{\hat{U}(\theta)-\mu}{\hat{\sigma}}$
    $\alpha \leftarrow 1 - \phi_{n-1}(|t|)$
if $\hat{U}(\theta) \leq \mu$ **then**
    $accept \leftarrow true$
else
    $accept \leftarrow false$

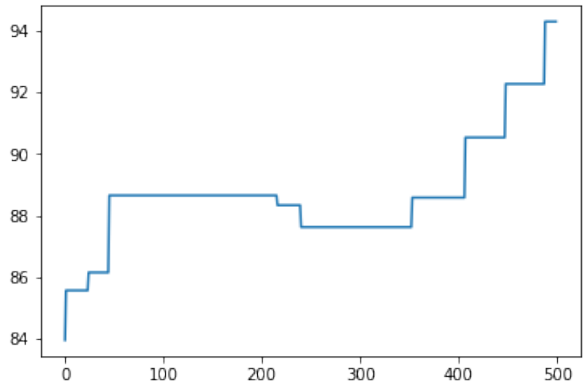

Figure 1: Log-posterior of iterates of chain

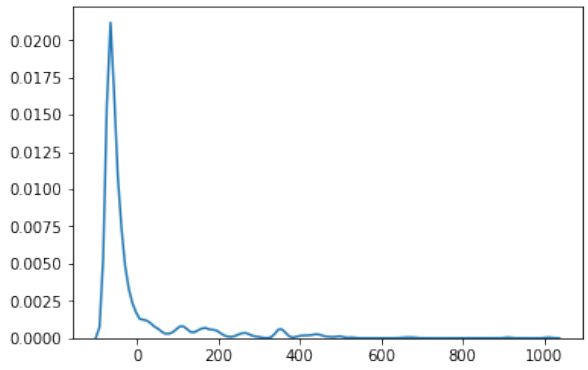

Figure 2: Distribution of $\widehat{U}(\theta^*) - U(\theta^*)$

4 Stochastic Gradient Langevin dynamics

4.1 The stochastic gradient LD step

Similar to stochastic gradient descent (SGD), stochastic gradient Langevin dynamics (SGLD) replaces the gradient of the log-posterior $\nabla_\theta U(\theta_t)$ in the Langevin dynamics equation with an unbiased estimate $\widehat{\nabla_\theta U(\theta_t)}$ computed using only a mini batch M corresponding to indices

I_M drawn uniformly at random from $[N]$ (with or without replacement) of size $S \ll N$:

$$\widehat{\nabla_\theta U(\theta_t)} = N \left(\frac{1}{S} \sum_{i \in I_M} \nabla_\theta U_i(\theta_t) \right)$$

Algorithm 2 Stochastic Gradient Langevin Dynamics step

Input: $\theta_t, \epsilon, S, \{x_i\}_{i \in [N]}$

Output: θ_{t+1}

1: Draw mini batch M of size S from $\{x_i\}_{i \in [N]}$ (with or without replacement)

2: Compute $\widehat{\nabla_\theta U(\theta_t)}$ using M

3: $\theta_{t+1} \leftarrow \theta_t - \epsilon \widehat{\nabla_\theta U(\theta_t)} + \sqrt{2\epsilon} w_t$ where $w_t \sim \mathcal{N}(0, 1)$

4.2 Experiments with SGLD

To check the performance of SGLD, we first ran LD with step size $\epsilon = 0.005$, and confirmed its convergence to the posterior (figure 3), before we ran SGLD with the same step size. On the diagonal elements of figure 4 we see the kernel density estimate of the stationary distribution of SGLD compared to the true posterior. Clearly, the stationary distribution of SGLD is far from the true posterior, even if we only look at the marginals. Looking at figure 5, we see that the distribution of the gradient estimator is very irregular, with heavy tails and a large variance.

Note that we could have used a smaller step size to improve performance, but a quick calculation comparing the variance along the largest component in the distribution of the estimator ($\sim 10^6$) and the variance along the largest component of the posterior distribution (~ 1) suggests reducing the step size by at least 2 order of magnitudes, whereas our use of stochastic gradients only reduces the computational cost by 2 orders of magnitude.

4.3 Variance reduced stochastic gradient LD

High variance and an irregular distribution of the estimator in SGLD is what causes its failure as illustrated in the previous section. Inspired by variance reduction techniques used for SGD, variance reduced versions of SGLD have been proposed. The most popular such variance reduction is known as SAGA [11]. The main idea of SAGA is to store the most recently evaluated gradient h_i for each data point x_i, keep updated their mean μ_h, and use the estimator:

$$\widehat{\nabla_\theta U(\theta_t)} = \frac{N}{S} \sum_{i \in I_M} (\nabla_\theta U_i(\theta_t) + \mu_h - h_i)$$

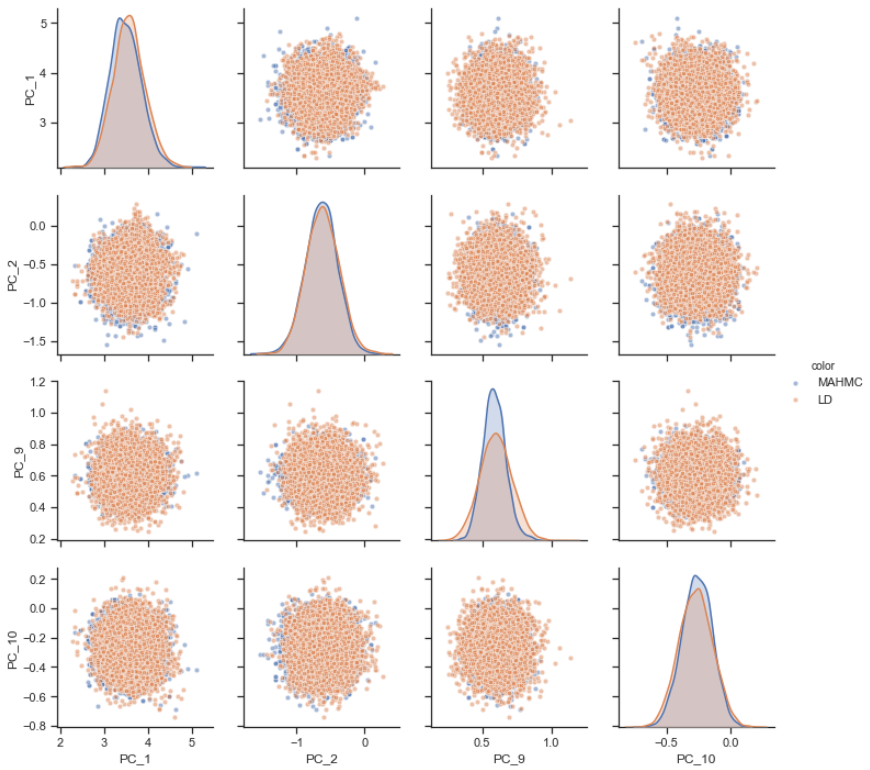

Figure 3: Stationary distribution of LD with $\epsilon = 0.005$

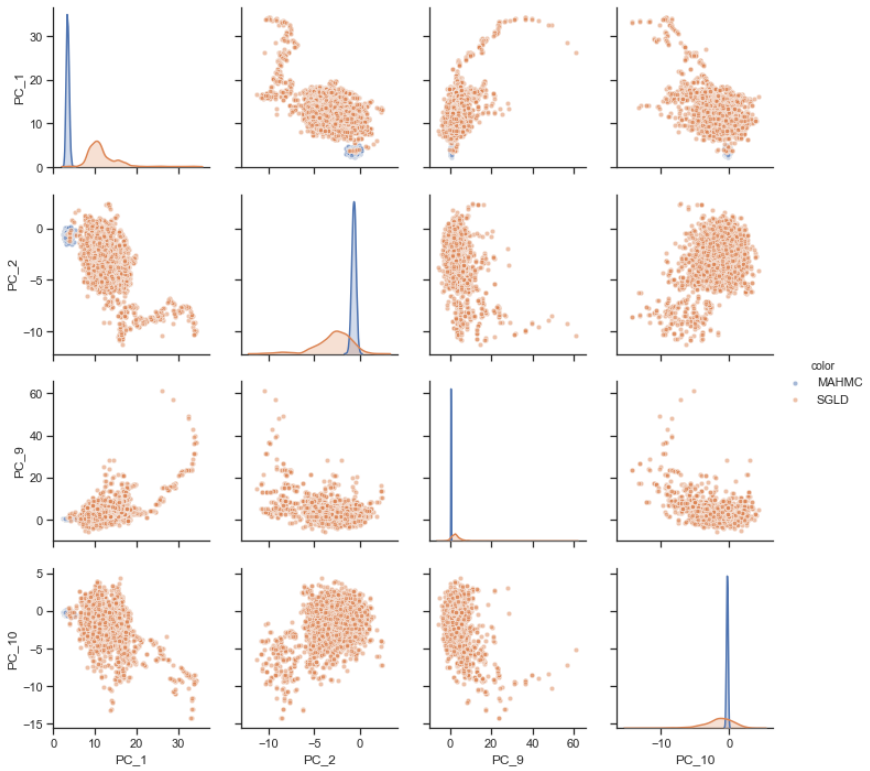

Figure 4: Stationary distribution of SGLD with $\epsilon = 0.005$

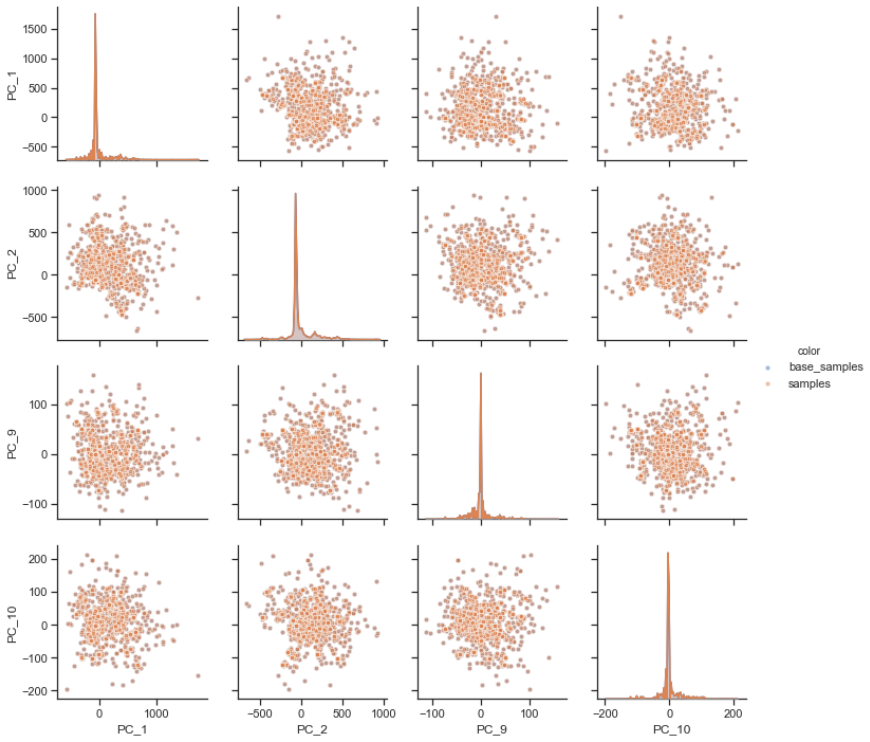

Figure 5: Distribution of $\widehat{\nabla_\theta U}(\theta) - \nabla_\theta U(\theta)$

This estimator has provably lower variance than the SGLD estimator. Yet another way to reduce the variance of this estimator is to consider importance sampling strategies. The idea is to target samples that might affect the gradient more than others. One such importance sampling strategy uses the Lipschitz constants L_i of the log-posterior corresponding to each data points x_i:

$$p_i = \frac{L_i}{\sum_{j=1}^{N} L_j}$$

for m-strongly log-concave distributions, the following distribution was proposed which currently achieves the best convergence rate in optimization when combined with SAGA [14]:

$$p_i = \frac{mN + L_i}{\sum_{j=1}^{N} mN + L_j}$$

The resulting estimator is given by:

$$\widehat{\nabla_\theta U(\theta_t)} = \frac{1}{S} \sum_{i \in I_M} \left(\frac{1}{p_i} \nabla_\theta U_i(\theta_t) + \mu_h - \frac{1}{p_i} h_i \right)$$

where $i \in I_M$ are now sampled with replacement according to $\{p_i\}_{i=1}^N$. The multiplication by $\frac{1}{p_i}$ preserves unbiasedness of the estimator.

4.4 Experiments with variance reduced stochastic gradient LD

Using the same step size $\epsilon = 0.005$, we ran SGLD with SAGA, importance sampling (IS), and both importance sampling and SAGA (IS-SAGA). Figure 6 and 7 show the evolution of the error on the mean and error on the variance in the 2-norm respectively for the estimates from each of the algorithms as a function of the number of gradient oracle uses. We see that the variance reduction algorithms do not perform any better than SGLD, at least at this step size.

Examining the variance of each of the estimators in Table 1, we see that it stays constant across all algorithms for $\epsilon = 0.005$, explaining the results in Figure 6 and 7. For smaller step sizes, the algorithms that use SAGA have a significantly lower variance in their estimators. This reduction in variance along with the decrease in the step size turns out to be enough to stabilize the algorithm and allow it to converge to the posterior as can be seen from figure 8. This comes at the cost of significantly higher autocorrelation as can be seen from figure 9. Thus from a pure sampling perspective SAGA-LD strictly outperforms LD, but from the point of view of MCMC it is less clear that this is the case.

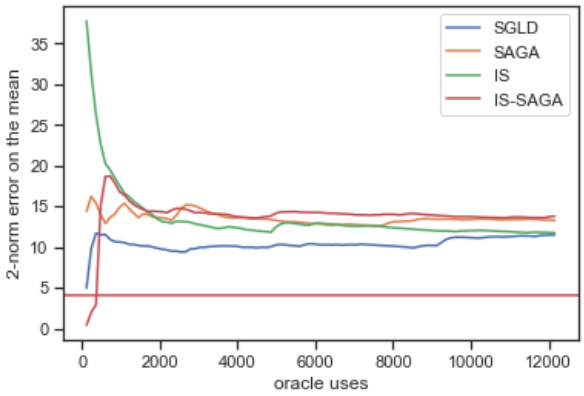

Figure 6: 2-norm error on the mean estimate of the posterior as a function of the number of gradient oracle uses. The red horizontal line is the norm of the mean itself.

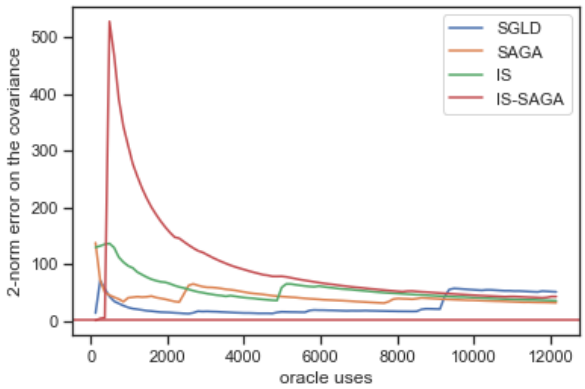

Figure 7: 2-norm error on the variance estimate of the posterior as a function of the number of gradient oracle uses. The red horizontal line is the norm of the variance itself.

ϵ	SGLD	SAGA	IS	IS-SAGA
5×10^{-3}	1000	1000	1000	1000
5×10^{-4}	1000	200	1000	300
5×10^{-5}	1000	100	1000	150

Table 1: Approximate values of the variance along the largest component of the covariance matrix of the estimator $\widehat{\nabla_\theta U(\theta_t)}$ for each algorithm.

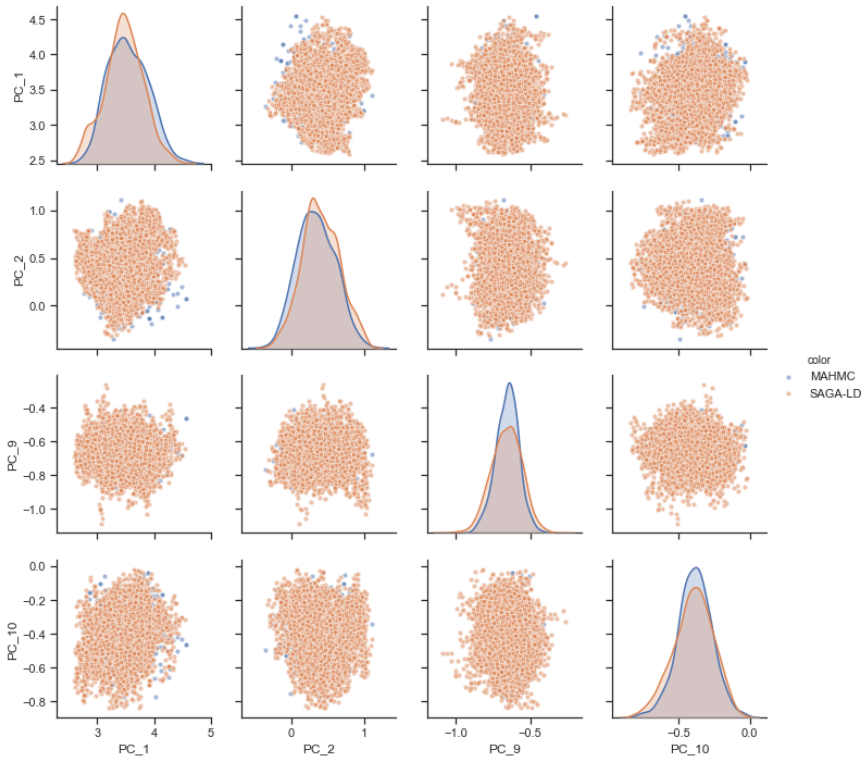

Figure 8: Stationary distribution of SAGA-LD with $\epsilon = 5 \times 10^{-4}$

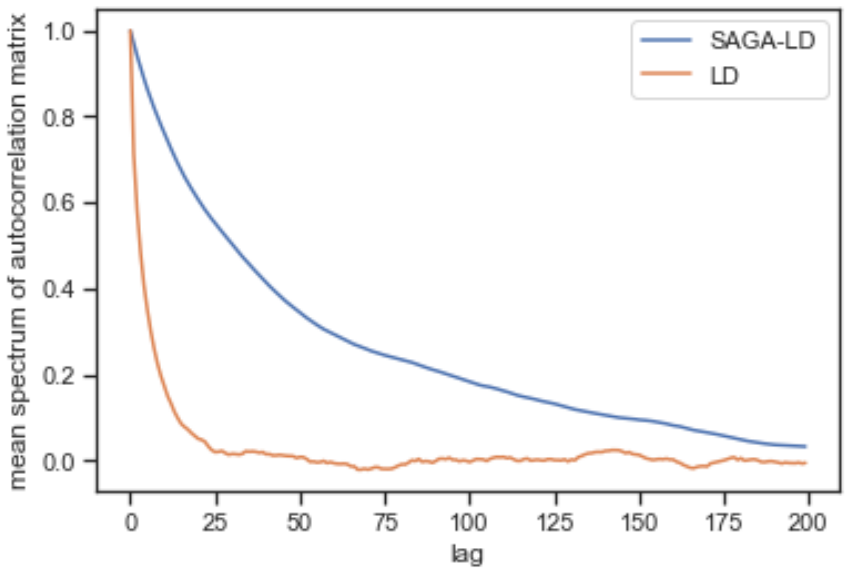

Figure 9: ACF of SAGA-LD compared to the ACF of LD.

5 Proposed subsampling strategy and Algorithms

The results of the previous two sections show the need for new variance reduction techniques that can, on the one hand, regularize the shape of the distribution of the estimators so as to eliminate the occurence of extreme values, and on the other, reduce the variance of the estimators so as to accelerate the algorithms by allowing faster conclusive tests for SMH, and the use of larger step sizes for SGLD.

In this section we propose a simple new importance sampling strategy based on SAGA and a minimization of a simple upper bound on the expected mean squared error of our estimators. We derive it here for the case of SGLD, but a very similar derivation works for many common transition kernels.

We then propose modifications to both SGLD and SMH, and present experimental results showing the performance of our proposed algorithms.

5.1 Derivation of the sampling distribution

Assume we are running SAGA, and at some step t, we have the gradients $\{h_i\}_{i=1}^{i=N}$ stored corresponding to the last evaluated gradients for each data points x_i. Let $\beta_i \geq 1$ be the number of iterations since h_i was last updated. Then $h_i = \nabla_\theta U_i(\theta_{t-\beta_i})$ and we have, letting $w_i \sim \mathcal{N}(0, 2t)$ be the normal noise generated (independently) at each step:

$$
\begin{aligned}
\|\nabla_\theta U_i(\theta_{t-\beta_i}) - h_i\|_2^2 &= \|\nabla_\theta U_i(\theta_t) - \nabla_\theta U_i(\theta_{t-\beta_i})\|_2^2 \\
&\leq L_i^2 \|\theta_t - \theta_{t-\beta_i}\|_2^2 \\
&= L_i^2 \left\| \theta_{t-\beta_i} + \sum_{i=t-\beta_i}^{t-1} \widehat{\nabla_\theta U(\theta_i)} + \sum_{i=t-\beta_i}^{t-1} w_i - \theta_{t-\beta_i} \right\|_2^2 \\
&= L_i^2 \left\| \mathcal{N}\left(\sum_{i=t-\beta_i}^{t-1} \widehat{\nabla_\theta U(\theta_i)}, 2t\beta_i \right) \right\|_2^2
\end{aligned}
$$

so that in expectation over the Gaussian noise we have:

$$\begin{aligned}
\mathrm{E}\left[\|\nabla_\theta U_i(\theta_{t-\beta_i}) - h_i\|_2^2\right] &\leq L_i^2 \,\mathrm{Tr}\left[2t\beta_i I_{d\times d}\right] + L_i^2 \left\| \sum_{i=t-\beta_i}^{t-1} \widehat{\nabla_\theta} U(\theta) \right\|_2^2 \\
&\leq 2td\beta_i d L_i^2 + L_i^2 \sum_{i=t-\beta_i}^{t-1} \left\| \widehat{\nabla_\theta} U(\theta_i) \right\|_2^2 \\
&=: L_i^2 (2td\beta_i d + K_{\beta_i})
\end{aligned}$$

Note that using the triangle inequality here is necessary to later avoid an $O(Nd)$ computational cost in computing the probabilities. Now the mean squared error of our estimator is given by:

$$\begin{aligned}
\mathrm{E}\left[\|\nabla_\theta U(\theta_t) - \widehat{\nabla_\theta U(\theta_t)}\|_2^2\right] &= \mathrm{Tr}\left[\mathrm{Var}\left[\widehat{\nabla_\theta U(\theta_t)}\right]\right] \text{ (unbiasedness)} \\
&= \frac{1}{S}\,\mathrm{Tr}\left[\underset{i\sim\{p_j\}_{j=1}^N}{\mathrm{Var}}\left[\frac{1}{p_j}\widehat{\nabla_\theta U_i(\theta_t)}\right]\right] \text{ (iid)} \\
&= \frac{1}{S}\left(\sum_{j=1}^N p_j \frac{1}{p_j^2}\|\nabla_\theta U_j(\theta_t)\|_2^2 - \|\nabla_\theta U(\theta_t)\|_2^2\right) \\
&= \frac{1}{S}\left(\sum_{j=1}^N \frac{1}{p_j}\|\nabla_\theta U_j(\theta_t)\|_2^2 - \|\nabla_\theta U(\theta_t)\|_2^2\right) \\
&= \frac{1}{S}\left(\sum_{j=1}^N \frac{1}{p_j}\|\nabla_\theta U_j(\theta_t) - h_j + h_j\|_2^2 - \|\nabla_\theta U(\theta_t)\|_2^2\right) \\
&\leq \frac{1}{S}\left(\sum_{j=1}^N \frac{1}{p_j}\left(\|\nabla_\theta U_j(\theta_t) - h_j\|_2^2 + \|h_j\|_2^2\right) - \|\nabla_\theta U(\theta_t)\|_2^2\right)
\end{aligned}$$

so that in expectation over the gaussian noise added at each step starting from step $t-\beta_i$ (recall that the inner expectation is with respect to the choice of mini batches), we have by the previous bound:

$$\mathrm{E}\left[\mathrm{E}\left[\|\nabla_\theta U(\theta_t) - \widehat{\nabla_\theta U(\theta_t)}\|_2^2\right]\right] \leq \frac{1}{S}\left(\sum_{j=1}^N \frac{1}{p_j}\left(L_i^2(2td\beta_i d + K_{\beta_i}) + \|h_j\|_2^2\right) - \|\nabla_\theta U(\theta_t)\|_2^2\right)$$

Only the first term depends on the choice of the sampling probability, hence minimizing this upper bound is equivalent to minimizing it. We therefore arrive at the following optimization problem:

$$\begin{aligned} \min_{p \in \mathbb{R}} \quad & \sum_{j=1}^{N} \frac{1}{p_j} \left(L_i^2 (2td\beta_i d + K_{\beta_i}) + \|h_j\|_2^2 \right) \\ \text{subject to} \quad & p_j > 0 \qquad \forall j \in [N] \\ & \sum_{j=1}^{N} p_j = 1 \end{aligned}$$

Relaxing the first constraint to $p_j \geq 0 \quad \forall j \in [N]$, and using the KKT conditions, we arrive at the solution:

$$p_i = \frac{L_i \sqrt{2td\beta_i d + K_{\beta_i}} + \|h_i\|_2}{\sum_{j=1}^{j=N} L_j \sqrt{2td\beta_j d + K_{\beta_j}} + \|h_j\|_2}$$

5.2 Modifications to SGLD and SMH

In this section we propose a modified version of SGLD and SMH which attempt to solve the problems we encountered in our experiments.

5.2.1 Modified SMH

For the case of SMH, we identified two major problems. Firstly, the high variance and irregular shape of the distribution of the estimator resulted sometimes in extreme estimates of the log-posterior. We propose using the importance sampling strategy outlined in the previous section to solve this problem. This will reduce the variance of our estimators and accelerate convergence to the CLT approximation, making the t-test more reliable.

Even with this correction, we still expect that our algorithm accepts a point with an extreme estimate of its log-posterior, although much less frequently. To allow our algorithm to recover from these extreme estimates, we propose eliminating the assumption that our current estimate is exact, and we suggest performing a two sample t-test at each iteration, updating both estimates whenever it is not conclusive (this is not strictly necessary, we can randomly pick one instead).

5.2.2 Modified SGLD

We have seen that the major problem that SGLD suffers from is the high variance of the gradient estimator. SAGA reduces this variance effectively, but requires a smaller step size, thus increasing the autocorrelation of the chain. We expect that our importance sampling strategy will reduce this variance even more, making it possible to reach the same step sizes as LD.

In addition, with this reduced variance, we expect our estimator to be normally distributed, and suggest estimating the variance of our estimator and using it to adjust the inherent noise of the process to recover the original discretized LD process.

5.2.3 Experiments modified SMH

We ran our modified SMH algorithm with the same proposal as the one initially used in section 2.2, and $\epsilon = 0.01$. As figure 10 shows, our modifications allowed the algorithm to avoid all the problems we encountered in section 2.2. Figure 11 shows the distribution of the estimator $\hat{U}(\theta^*)$ at the mode of the distribution θ^*, using our new importance sampling strategy. We observe that the distribution of our estimator is indistinguishable from a normal, with significantly reduced variance. Figure 12 shows that our algorithm indeed converges to a stationary distribution close to the true posterior.

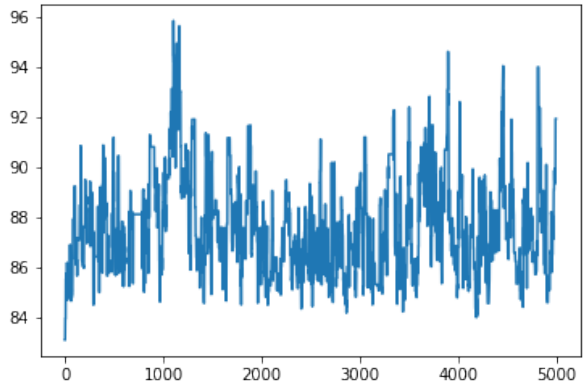

Figure 10: Log-posterior of iterates of chain

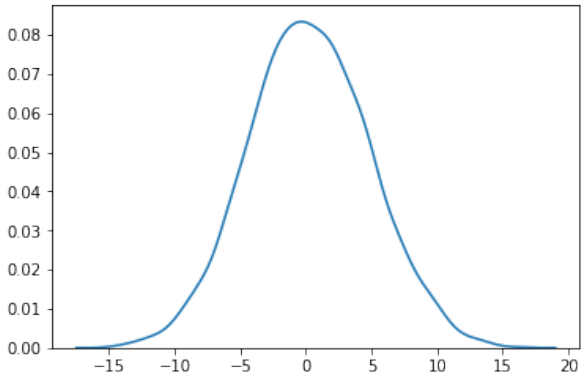

Figure 11: Distribution of $\widehat{U}(\theta^*) - U(\theta^*)$

5.2.4 Experiments modified SGLD

We ran our modified SGLD algorithm with the same step size $\epsilon = 0.005$, the same as with LD. Figure 13 shows that our modified algorithm converges to the true posterior even with this large step size. Figure 14 shows the distribution of our estimator. Looking at the variance along the principal component, we see that it decreased by 2 orders of magnitude

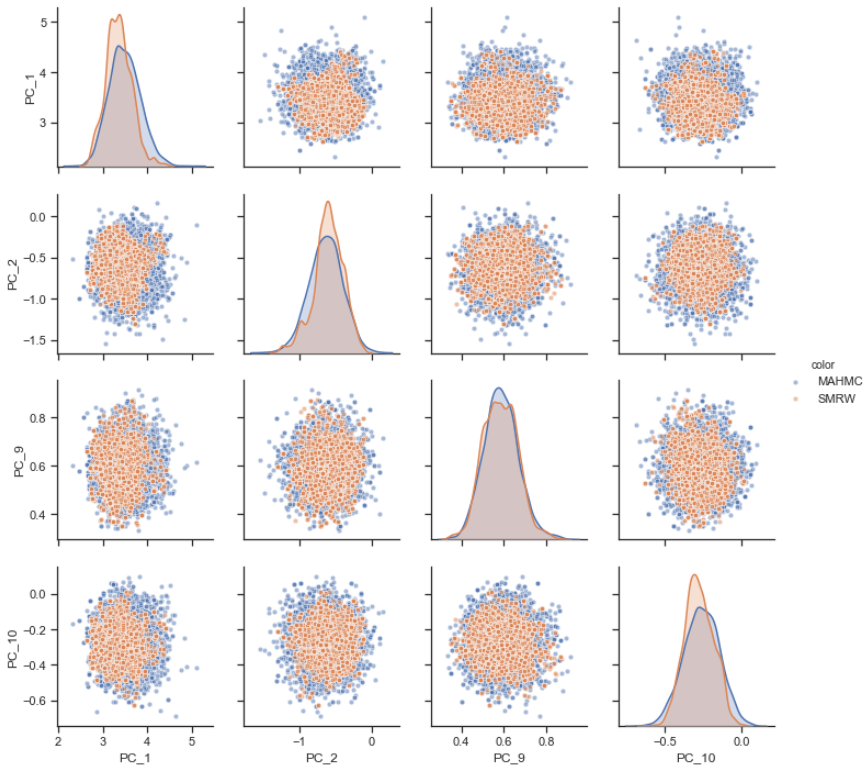

Figure 12: Stationary distribution of the modified SMH with $\epsilon = 0.01$

in comparison to all other algorithms at this step size. We also notice that the distribution of our estimator is very well approximated by a multivariate normal, allowing us to adjust for it and recover the correct transition kernel. Finally, looking at figure 15, we see that the modified SGLD enjoys the same autocorrelation as LD, in contrast with SAGA-LD. This is expected since the algorithm allows the use of a larger step size.

5.2.5 Comparison with standard MCMC algorithms

Finally we compare the performance of our algorithms with that of standard ones on the same dataset, but taking the first 100 principal components. We take the number of oracle uses as a measure of complexity and compare the convergence of the mean and variance estimates to their true value as a function of oracle calls. Figures 16 and 17 show a significant improvement on the time complexity of our modified SMH algorithm over the full MH algorithm. Similarly, figures 18 and 19 show an even more dramatic improvement when comparing modified SGLD with LD.

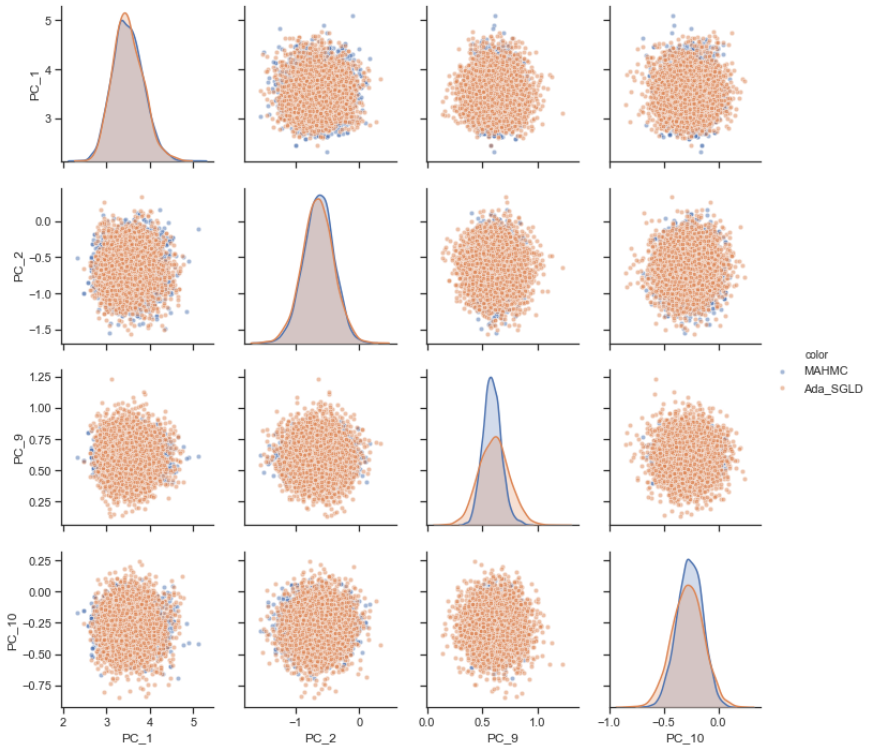

Figure 13: Stationary distribution of the modified SGLD with $\epsilon = 0.005$

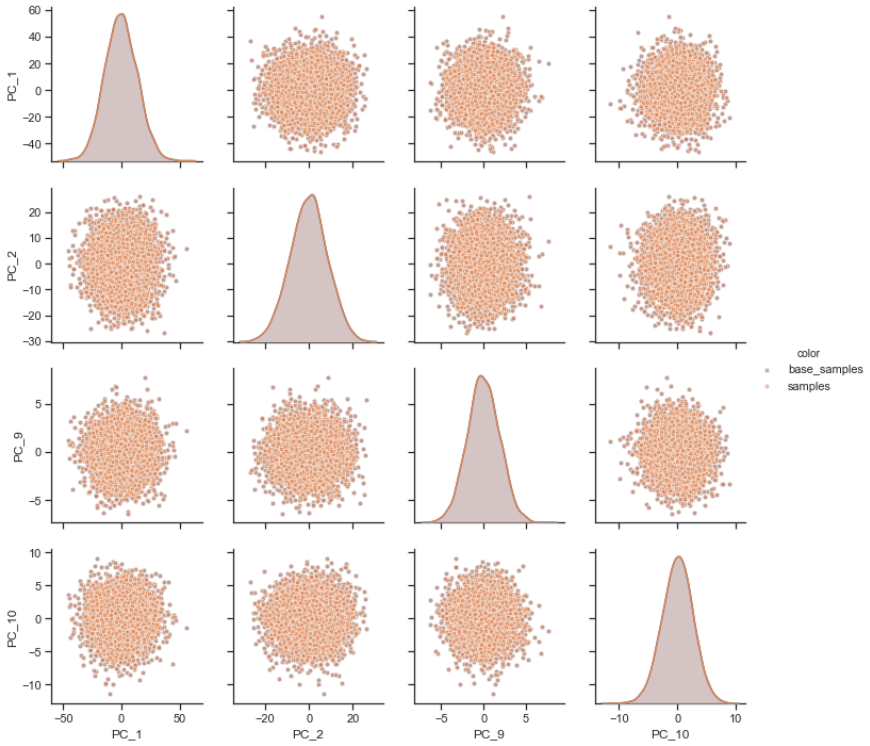

Figure 14: Distribution of $\widehat{\nabla_\theta U}(\theta) - \nabla_\theta U(\theta)$

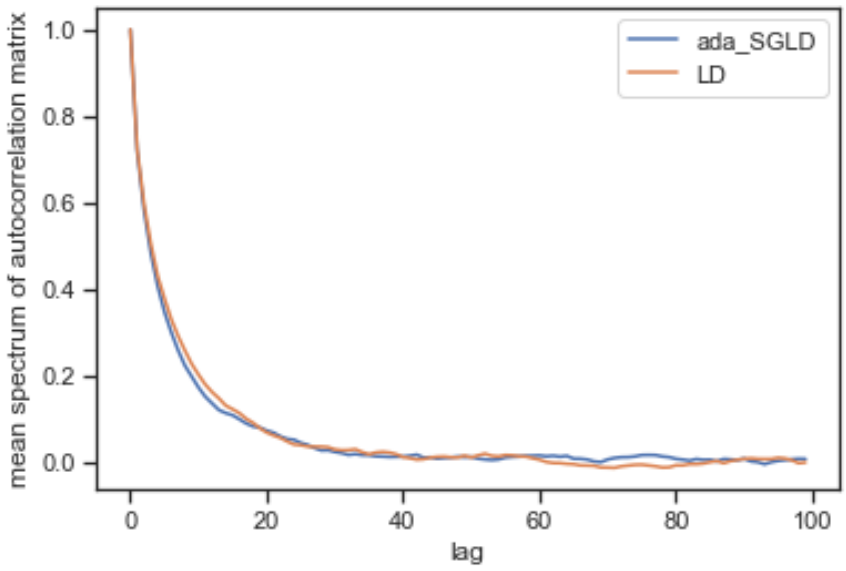

Figure 15: ACF of the modified SGLD compared to the ACF of LD

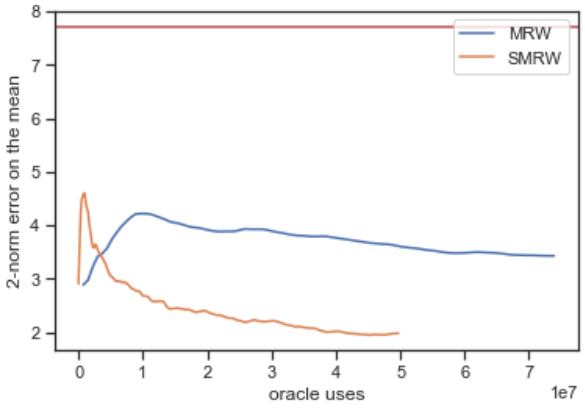

Figure 16: 2-norm of the error on the mean for modified SMH and MH as a function of oracle calls

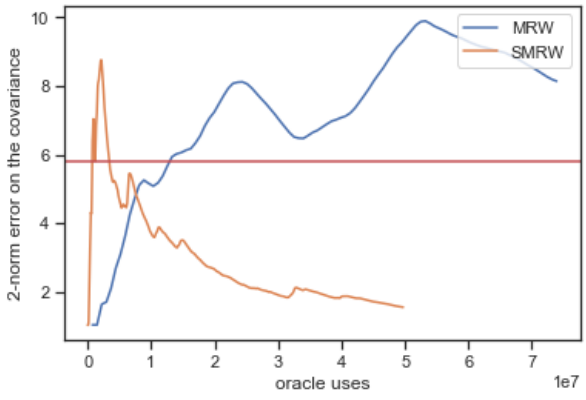

Figure 17: 2-norm of the error on the variance for modified SMH and MH as a function of oracle calls

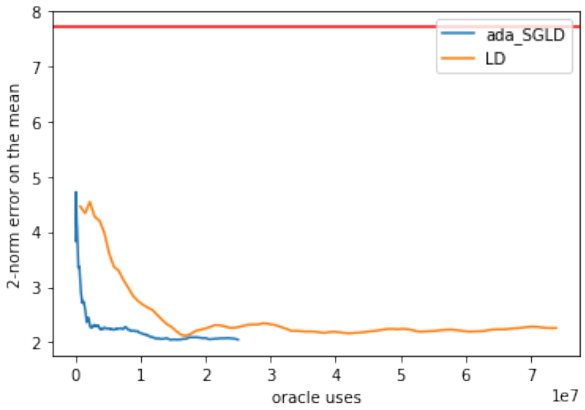

Figure 18: 2-norm of the error on the mean for modified SGLD and LD as a function of oracle calls.

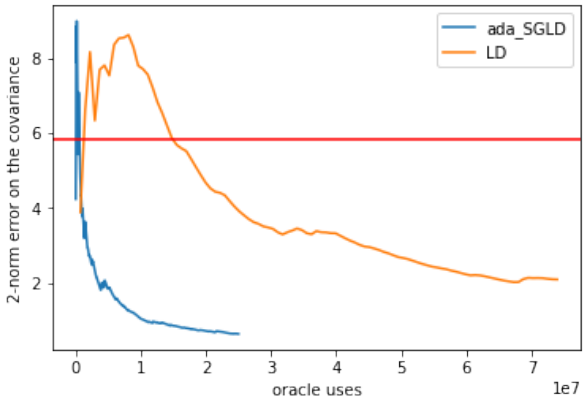

Figure 19: 2-norm of the error on the variance for modified SGLD and LD as a function of oracle calls.

6 Conclusion

In this report, we reviewed two popular approximate MCMC algorithms based on data subsampling. We analyzed their behavior through experiments and identified their weaknesses. High variance of the estimators was a common problem between the two algorithms. We proposed a novel adaptive subsampling strategy based on SAGA.

Using this novel importance sampling strategy, we showed that in the case of SGLD, our algorithm allows us to recover the dynamics of non-stochastic LD with the same large step size, and a fraction of the computational cost. In the case of SMH, the use of a two-sample t-test instead of a one-sample test, combined with our importance sampling strategy allows us to regularize the distribution of our estimator, and gave us a way to deal with extreme estimates, allowing us to recover convergence to a good approximation to the posterior. Our importance sampling strategy can also be used in an optimization context, although this is still to be tried.

An analysis of the effect of this importance sampling strategy on the convergence of both optimization and sampling algorithms is currently lacking, but experimental evidence suggests that it can greatly reduce the variance of estimators that are based on subsampling of the data. Such analysis should, among other things, allow us to estimate the optimal choice of batch size, which can crucially affect the performance of such algorithms.

References

[1] Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring. 2012.

[2] Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. Control Variates for Stochastic Gradient MCMC. pages 1–22, 2016.

[3] Alexandros Beskos, Natesh S. Pillai, Gareth O. Roberts, Jesus M. Sanz-Serna, and Andrew M. Stuart. Optimal tuning of the Hybrid Monte-Carlo Algorithm. 2010.

[4] Niladri S. Chatterji, Nicolas Flammarion, Yi-An Ma, Peter L. Bartlett, and Michael I. Jordan. On the Theory of Variance Reduction for Stochastic Gradient Monte Carlo. 2018.

[5] Tianqi Chen, Emily B Fox, Carlos Guestrin, Guestrin Cs, and Washington Edu. Stochastic Gradient Hamiltonian Monte Carlo. 32, 2014.

[6] Xiang Cheng and Peter Bartlett. Convergence of Langevin MCMC in KL-divergence. (1):1–27, 2017.

[7] Xiang Cheng, Niladri S. Chatterji, Yasin Abbasi-Yadkori, Peter L. Bartlett, and Michael I. Jordan. Sharp Convergence Rates for Langevin Dynamics in the Nonconvex Setting. 2018.

[8] Xiang Cheng, Niladri S. Chatterji, Peter L. Bartlett, and Michael I. Jordan. Underdamped Langevin MCMC: A non-asymptotic analysis. 2017.

[9] Arnak S. Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 79(3):651–676, 2017.

[10] Arnak S. Dalalyan and Avetik G. Karagulyan. User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. 0(1):1–29, 2017.

[11] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. pages 1–15, 2014.

[12] Nan Ding, Changyou Chen, Robert D Skeel, and Ryan Babbush. Bayesian Sampling Using Stochastic Gradient Thermostats. pages 1–14.

[13] Raaz Dwivedi, Yuansi Chen, Martin J. Wainwright, and Bin Yu. Log-concave sampling: Metropolis-Hastings algorithms are fast! pages 1–40, 2018.

[14] Robert M Gower, Peter Richtárik, Francis Bach, and Télécom ParisTech. Stochastic Quasi-Gradient Methods: Variance Reduction via Jacobian Sketching * session on Continuous Optimization for Machine Learning, EDF'Lab Paris-Saclay). pages 1–52, 2017.

[15] Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget. pages 1–23, 2013.

[16] Yi-An Ma, Niladri Chatterji, Xiang Cheng, Nicolas Flammarion, Peter Bartlett, and Michael I. Jordan. Is There an Analog of Nesterov Acceleration for MCMC? 2019.

[17] Yi-An Ma, Tianqi Chen, and Emily B. Fox. A Complete Recipe for Stochastic Gradient MCMC. pages 1–19, 2015.

[18] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic Gradient Descent as Approximate Bayesian Inference. 18:1–35, 2017.

[19] Oren Mangoubi and Aaron Smith. Rapid Mixing of Hamiltonian Monte Carlo on Strongly Log-Concave Distributions. 2017.

[20] Oren Mangoubi and Nisheeth K. Vishnoi. Dimensionally Tight Bounds for Second-Order Hamiltonian Monte Carlo. 2018.

[21] Tigran Nagapetyan, Andrew B Duncan, Leonard Hasenclever, Sebastian J Vollmer, and Konstantinos Zygalakis KZygalakis. The True Cost of SGLD. 2017.

[22] Gareth O. Roberts and Jeffrey S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1(0):20–71, 2005.

[23] Santosh S. Vempala and Andre Wibisono. Rapid Convergence of the Unadjusted Langevin Algorithm: Log-Sobolev Suffices. 2019.

[24] Max Welling and Yee-Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. *Icml'2011*, pages 681–688, 2011.

[25] Andre Wibisono. Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem. 75:1–35, 2018.