AMAGOLD: Amortized Metropolis Adjustment for Efficient Stochastic Gradient MCMC

Ruqi Zhang
Cornell University

A. Feder Cooper
Cornell University

Christopher De Sa
Cornell University

Abstract

Stochastic gradient Hamiltonian Monte Carlo (SGHMC) is an efficient method for sampling from continuous distributions. It is a faster alternative to HMC: instead of using the whole dataset at each iteration, SGHMC uses only a subsample. This improves performance, but introduces bias that can cause SGHMC to converge to the wrong distribution. One can prevent this using a step size that decays to zero, but such a step size schedule can drastically slow down convergence. To address this tension, we propose a novel second-order SG-MCMC algorithm—AMAGOLD—that *infrequently* uses Metropolis-Hastings (M-H) corrections to remove bias. The infrequency of corrections amortizes their cost. We prove AMAGOLD converges to the target distribution with a fixed, rather than a diminishing, step size, and that its convergence rate is at most a constant factor slower than a full-batch baseline. We empirically demonstrate AMAGOLD's effectiveness on synthetic distributions, Bayesian logistic regression, and Bayesian neural networks.

1 Introduction

Markov chain Monte Carlo (MCMC) methods play an important role in Bayesian inference. They work by constructing a Markov chain with the desired distribution as its equilibrium distribution; one samples from the chain and, as the algorithm converges to its equilibrium, the samples drawn reflect the desired distribution (Metropolis et al., 1953; Duane et al., 1987; Horowitz, 1991; Neal et al., 2011). Although MCMC is a powerful technique, when the sampled distribution depends on a very large dataset, its performance is often limited by the dataset's size—usually by the cost of computing sums over the entire dataset.

One approach for scaling MCMC is to decouple the algorithm from the size of the dataset, using stochastic gradients in lieu of full-batch gradients (Welling and Teh, 2011; Chen et al., 2014; Ding et al., 2014). This family of methods is called Stochastic gradient MCMC (SG-MCMC). For example, stochastic gradient Langevin dynamics (SGLD) replaces the gradient with a stochastic estimate in first order Langevin Monte Carlo (LMC) (Welling and Teh, 2011). The second-order analog of SGLD is stochastic gradient Hamiltonian Monte Carlo (SGHMC). SGHMC can be thought of as a stochastic version of the popular Hamiltonian Monte Carlo (HMC) algorithm and second-order Langevin dynamics (L2MC) (Chen et al., 2014). These algorithms have been particularly useful in Bayesian neural networks, and many variants have been proposed to increase their sampling efficiency and accuracy (Ding et al., 2014; Ahn et al., 2012; Ma et al., 2015; Zhang et al., 2017). Table 1 provides a clarifying summary of the algorithms considered in this paper.

The runtime efficiency benefits of such SG-MCMC methods also come with a drawback: stochastic gradients introduce bias. Bias can cause convergence to a stationary distribution that differs from the one we wanted to sample from, and usually comes from two sources: converting the continuous-time process into discrete gradient updates, and noise from stochastic gradient estimates. These sources of error are in a sense unavoidable because they are also the source of SG-MCMC's runtime efficiency. Discretization with a large step size ϵ (instead of diminishing $\epsilon \to 0$) allows SG-MCMC to quickly move around its state space, and using stochastic gradients is key for scalability.

The standard approach for removing bias from a Markov chain is to introduce a *Metropolis-Hastings (M-H) correction* (Metropolis et al., 1953). This involves rejecting some fraction of the chain's transitions

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

Algorithm	Exact?	Stochastic Gradient?
AMAGOLD	Yes	Yes
L2MC	Yes	No
HMC	Yes	No
SGHMC	No	Yes

Table 1: Comparing 2nd order MCMC methods.

to restore the correct stationary distribution. Naïvely applying M-H to SG-MCMC algorithms is often computationally prohibitive because the M-H step typically needs to sum over the entire dataset. Performing this expensive computation every iteration would defeat the purpose of using stochastic gradients to improve performance. Thus, more sophisticated techniques are needed to achieve efficient, unbiased sampling for SG-MCMC.

In this paper, we show that asymptotic exactness is possible *without being prohibitively expensive*. Specifically, we propose *Amortized Metropolis-Adjusted stochastic Gradient second-Order Langevin Dynamics (AMAGOLD)*. It achieves asymptotic exactness for SGHMC by using an M-H correction step and does so without obliterating the performance gains provided by stochasticity. Our key insight is to apply the M-H step *infrequently*. Rather than computing it every update, AMAGOLD performs it every T steps ($T > 0$). We prove this is sufficient to remove bias while also improving performance by amortizing the M-H correction cost over T steps. We develop both reversible and non-reversible AMAGOLD variants and prove both converge to the desired distribution. We also prove a convergence rate relative to using full-batch gradients, which cleanly captures the effect of using stochastic gradients. This result provides insight about the trade-off between minibatching speed-ups and the convergence rate. Our results also show the noise from stochastic gradients has a provably bounded effect on convergence. In summary, our contributions are as follows:

- We introduce AMAGOLD, an efficient, asymptotically-exact SGHMC variant that *infrequently* applies an M-H correction. We give reversible and non-reversible versions.
- We guarantee AMAGOLD converges to the target distribution, and does so without requiring step size $\epsilon \to 0$ or precise noise variance estimation.
- We prove a bound on AMAGOLD's convergence rate with mild assumptions, measured by the spectral gap. This bound is relative to how fast the algorithm *would have* converged if full-batch gradients were used. This is the first such relative convergence bound we are aware of for SG-MCMC.

Algorithm 1 SGHMC

given: Energy U, initial state $\theta \in \Theta$, step size ϵ, momentum variance σ^2, friction β
loop
  optionally, resample momentum:
  $r \sim \mathcal{N}(0, \sigma^2)$
  initialize position and momentum:
  $r_{\frac{1}{2}} \leftarrow r$, $\theta_0 \leftarrow \theta$
  for $t = 1$ **to** T **do**
    position update: $\theta_t \leftarrow \theta_{t-1} + \epsilon\sigma^{-2} r_{t-\frac{1}{2}}$
    sample noise $\eta_t \sim \mathcal{N}(0, 4\epsilon\beta\sigma^2)$
    sample random energy component $\tilde{U}_t$
    update momentum:

$$r_{t+\frac{1}{2}} \leftarrow r_{t-\frac{1}{2}} - \epsilon\nabla\tilde{U}_t(\theta_t) - 2\epsilon\beta r_{t-\frac{1}{2}} + \eta_t$$

  end for
  new values: $(\theta, r) \leftarrow (\theta_T, r_{T+\frac{1}{2}})$
  ▷ no M-H step
end loop

- We validate our convergence guarantees empirically. Comparing to SGHMC, AMAGOLD is more robust to hyperparameters. Regarding performance, AMAGOLD is competitive with full-batch baselines on synthetic and real-world datasets, and outperforms SGHMC on various tasks.

2 Related Work

Our work is situated within a rich literature of SG-MCMC variants that take advantage of stochastic gradient techniques. These methods have demonstrated success on deep neural networks (DNNs) for various tasks (Li et al., 2016c; Gan et al., 2016; Zhang et al., 2020). In particular, second-order SG-MCMC methods like SGHMC, which have a momentum term, have been shown to outperform first-order methods like SGLD on many applications (Chen et al., 2014, 2015). Gao et al. (2018) proves SGHMC's convergence can be faster than SGLD's on non-convex problem due to its momentum-based acceleration. SGHMC can also be thought of as a stochastic version of L2MC (Horowitz, 1991) or HMC; we therefore use both L2MC and HMC as experimental full-batch baselines.

Prior work has also studied SGHMC's convergence properties. Chen et al. (2014) examines its convergence for "asymptotically" small step sizes, in which a continuous-time system governs the dynamics (in contrast, our algorithm is asymptotically exact with a constant step size). Other work proves convergence with high-order integrators (Chen et al., 2015) and obtains non-asymptotic convergence bounds for SGHMC

on non-convex optimization tasks (Gao et al., 2018).

Additional work has studied the properties of first-order M-H adjusted Langevin methods, such as MALA (Grenander and Miller, 1994; Roberts et al., 1996; Roberts and Rosenthal, 1998; Roberts and Stramer, 2002; Stramer and Tweedie, 1999). Dwivedi et al. (2018) derives the mixing time of MALA for strongly log-concave densities, showing it has a better convergence rate than unadjusted Langevin (in comparison, AMAGOLD does not require the assumption of strongly log-concave densities). Korattikara et al. (2014) developed a minibatch M-H approach, which uses subsampling in the M-H correction step, and applied it to correct bias in SGLD. They show cases where SGLD diverges from the target distribution, while SGLD with a minibatched M-H correction performs well.

The work above involves first-order methods. To the best of our knowledge, we are the first to develop an unbiased, efficient second-order SG-MCMC algorithm. We are also the first in this space to use the spectral gap, a traditional metric to evaluate MCMC convergence (Hairer et al., 2014; Levin and Peres, 2017; De Sa et al., 2018). It requires milder assumptions than techniques in prior SG-MCMC work, such as 2-Wasserstein (Raginsky et al., 2017; Dalalyan and Karagulyan, 2019), mean squared error (Vollmer et al., 2016; Chen et al., 2015), and empirical risk (Gao et al., 2018).

3 Preliminaries

We start by briefly describing the standard setup of Bayesian inference. Given some dataset $\mathcal{D}$ and domain Θ, suppose we are interested in sampling from the posterior distribution $\pi(\theta) \propto \exp\left(-U(\theta)\right)$ where

$$U(\theta) = -\sum_{x\in\mathcal{D}} \log p(x|\theta) - \log p(\theta).$$

$U(\theta)$ is the *energy function*, θ ranges over Θ, and $\pi \propto \mu$ denotes π is the unique distribution with PDF proportional to μ. One way to compute this distribution is to construct a Markov chain with stationary distribution π and run it to produce a sequence of samples.

A second-order chain, such as HMC, SGHMC, or L2MC (Duane et al., 1987; Horowitz, 1991; Neal et al., 2011), does this by *augmenting* the state space with an additional momentum variable r, giving joint distribution

$$\pi(\theta, r) \propto \exp(-H(\theta, r)) = \exp\left(-U(\theta) - \frac{1}{2\sigma^2}\|r\|^2\right),$$

where H is the *Hamiltonian*, which measures the total energy of the system. Note we could replace the norm with any positive definite quadratic form on r. For simplicity, we only consider the case of isotropic momentum energy—where the *mass matrix* is $\sigma^2 I$. HMC then simulates Hamiltonian dynamics

$$d\theta = \sigma^{-2} r\, dt, \qquad dr = -\nabla U(\theta)\, dt. \tag{1}$$

The value of the Hamiltonian is preserved under these dynamics, so we must also include transitions that change the value of H to explore the whole state space. HMC does this by periodically resampling r from its conditional distribution. L2MC does so by continuously modifying r with a friction term and added Gaussian noise. Even though (1) preserves H, the discrete simulation of (1) run by HMC or L2MC *does not* necessarily do so. Therefore both algorithms need an M-H correction step to prevent bias due to discretization.

SGHMC (Algorithm 1) reduces the computational cost of these methods by using a stochastic gradient in lieu of the full-batch gradient ∇U. It estimates $U(\theta)$ using minibatch $\tilde{\mathcal{D}}$:

$$\tilde{U}(\theta) \approx -\frac{|\mathcal{D}|}{|\tilde{\mathcal{D}}|}\sum_{x\in\tilde{\mathcal{D}}} \log p(x|\theta) - \log p(\theta).$$

However, using a minibatch introduces noise; naïvely replacing U by $\tilde{U}$ leads to divergence from the target distribution. To offset this noise, SGHMC adds the friction term from L2MC (Appendix A). SGHMC uses the leapfrog algorithm to discretize the system (Neal et al., 2011). Notably, SGHMC does not include an M-H correction; to reduce the bias, it requires small ϵ.

4 Amortized Metropolis Adjustment

Reversible Markov chains are a particularly well-studied and well-behaved class of Markov chains. A Markov chain with transition probability operator G is reversible (also called satisfying the *detailed balance condition*) if for any pair of states x and y

$$\pi(x)G(x, y) = \pi(y)G(y, x). \tag{2}$$

It is well-known that a chain satisfying (2) has stationary distribution π. An M-H correction constructs a reversible chain G with stationary distribution π from any Markov chain P (called the *proposal distribution*) by doing the following at each iteration. First, starting from state x, sample y from the proposal distribution $P(x, y)$. Second, compute the *acceptance probability*

$$\tau = \min\left(1, \frac{\pi(y)P(y, x)}{\pi(x)P(x, y)}\right).$$

Finally, with probability τ transition to state y; otherwise, remain in state x. This correction results in a reversible chain with stationary distribution π; however, computing τ at every step can be costly.

The natural way to amortize the cost of running M-H is to replace the single proposal of baseline M-H with T proposal-chain steps. This divides its cost among T iterations of the underlying chain, effectively decreasing it by a factor of T. For stochastic MCMC, each proposal step can be written as $P(x, y; \zeta)$, which denotes the probability of going from state x to state y given stochastic sample ζ taken from some known distribution. (In minibatched MCMC, ζ captures information about which data we sample at that step.) Using this, we can run the following algorithm, starting at x. First, set $x_0 = x$, and run for t from 0 to $T-1$

$$\text{sample noise } \zeta_t, \text{ then sample } x_{t+1} \sim P(x_t, x_{t+1}; \zeta_t).$$

Next, set $y = x_T$: this is the proposal run an M-H correction on. Finally, compute the acceptance probability

$$\tau = \min\left(1, \frac{\pi(y)}{\pi(x)} \prod_{t=0}^{T-1} \frac{P(x_{t+1}, x_t; \zeta_t)}{P(x_t, x_{t+1}; \zeta_t)}\right) \tag{3}$$

$$= \min\left(1, \prod_{t=0}^{T-1} \frac{\pi(x_{t+1}) P(x_{t+1}, x_t; \zeta_t)}{\pi(x_t) P(x_t, x_{t+1}; \zeta_t)}\right). \tag{4}$$

and transition to state y with probability τ; otherwise, remain in state x.

It is straightforward to see this algorithm results in a reversible chain with stationary distribution π.[1] Additionally, this amortized M-H step (3) is easily computed as long as the probabilities $P(\cdot, \cdot; \zeta)$ are tractable.

We expect this approach will be effective when the M-H step does "not reject too often". This will certainly be the case when the terms inside the product in (4) all tend to be close to 1, which happens when the proposals $P(x, y; \eta)$ are "close" to being reversible with stationary distribution π. This is a good heuristic: our amortization approach should be effective when the proposals are close to being reversible.

Unfortunately, this heuristic does not apply to SGHMC's proposal step since SGHMC and other Hamiltonian-like steps are not close to satisfying the reversibility condition (2). Instead, the natural "reverse" trajectory for a Hamiltonian step reverses the order of the states and *negates the momentum*. The analog of reversibility for this sort of step is *skew-reversibility* (Turitsyn et al., 2011). Given some measure-preserving involution over the state space denoted $x^\perp$, a chain G is skew-reversible if $\pi(x) = \pi(x^\perp)$ and

$$\pi(x) G(x, y) = \pi(y^\perp) G(y^\perp, x^\perp). \tag{5}$$

Concretely, for Hamiltonian dynamics we use the involution that negates the momentum, i.e. $(\theta, r)^\perp = (\theta, -r)$.

[1] A detailed proof appears in Appendix B.

It is straightforward to show that a skew-reversible Markov chain also has π as its stationary distribution.[1] Such non-reversible chains have attracted a great deal of recent attention because they are more efficient than reversible ones in some situations (Turitsyn et al., 2011; Hukushima and Sakai, 2013; Ma et al., 2016).

A natural consequence of this setup is that we can amortize M-H in the same manner as before, using skew-reversibility in place of reversibility. This gives the same multi-step-proposal algorithm as before, except that the acceptance probability is replaced with

$$\tau = \min\left(1, \frac{\pi(y^\perp)}{\pi(x)} \prod_{t=0}^{T-1} \frac{P(x_{t+1}^\perp, x_t^\perp; \zeta_t)}{P(x_t, x_{t+1}; \zeta_t)}\right). \tag{6}$$

The resulting corrected chain will be skew-reversible with stationary distribution π.[1] Intuitively, this chain will "not reject too often" as long as the proposals P are "close" to being skew-reversible. Since SGHMC steps are close to being skew-reversible, this is the more natural approach for amortizing M-H, rather than using (3). If one wants to use the well-developed theoretical tools for a reversible chain, it is known that we can recover a reversible chain from a skew-reversible one by simply resampling the momentum at the beginning of the outer loop.[1] Note this reversible chain can be different from the one obtained by using condition (3).

5 AMAGOLD

We now apply the amortized Metropolis adjustment (AMA) method of Section 4 to second-order SG-MCMC. As a proposal, we use the stochastic leapfrog step that starts in (θ, r) and proposes (θ^*, r^*) by running

$$\theta_0 = \theta + \frac{1}{2}\epsilon\sigma^{-2} r$$
$$r^* = ((1 - \epsilon\beta) r - \epsilon \nabla \tilde{U}_t(\theta_0) + \mathcal{N}(0, 4\epsilon\beta\sigma^2 I))/(1 + \epsilon\beta)$$
$$\theta^* = \theta_0 + \frac{1}{2}\epsilon\sigma^{-2} r^*.$$

Applying our amortized M-H correction to this proposal step using the acceptance probability (6) results in AMAGOLD (Algorithm 2). AMAGOLD is, by construction, skew-reversible, and we have the option of making it reversible by resampling the momentum.

AMAGOLD has three key differences compared to SGHMC. First, motivated by the time-reversal-symmetric nature of conditions (2) and (5), we use a clearly time-symmetric update step in the inner loop (compare Line 12 of Algorithm 2, which can be written as $r_{t+\frac{1}{2}} \leftarrow r_{t-\frac{1}{2}} - \epsilon \nabla \tilde{U}_t(\theta_t) - \epsilon\beta(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}) + \eta_t$, with the less clearly symmetric Line 11 of Algorithm 1). Note this is just a different way of writing the algorithm: the update steps could be made equivalent by appropriately setting the hyperparameters. Second,

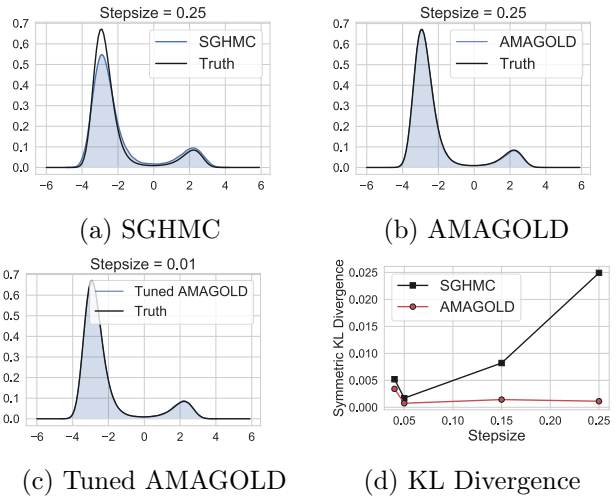
Algorithm 2 AMAGOLD

1: **given:** Energy U, initial state $\theta \in \Theta$, step size ϵ, momentum variance σ^2, friction β
2: **loop**
3: **optionally, resample momentum:** $r \sim \mathcal{N}(0, \sigma^2 I)$
4: **initialize momentum, energy acc:** $r_{-\frac{1}{2}} \leftarrow r$, $\rho_{-\frac{1}{2}} \leftarrow 0$
5: **half position update:** $\theta_0 \leftarrow \theta + \frac{1}{2}\epsilon\sigma^{-2} r_{-\frac{1}{2}}$
6: **for** $t = 0$ **to** $T-1$ **do**
7: **if** $t \neq 0$ **then**
8: **position update:** $\theta_t \leftarrow \theta_{t-1} + \epsilon\sigma^{-2} r_{t-\frac{1}{2}}$
9: **end if**
10: **sample noise** $\eta_t \sim \mathcal{N}(0, 4\epsilon\beta\sigma^2 I)$
11: **sample random energy component** $\tilde{U}_t$
12: **update momentum:** $r_{t+\frac{1}{2}} \leftarrow ((1-\epsilon\beta) r_{t-\frac{1}{2}} - \epsilon\nabla\tilde{U}_t(\theta_t) + \eta_t)/(1+\epsilon\beta)$
13: **update energy acc:** $\rho_{t+\frac{1}{2}} \leftarrow \rho_{t-\frac{1}{2}} + \frac{1}{2}\epsilon\sigma^{-2}\nabla\tilde{U}_t(\theta_t)^T \left(r_{t-\frac{1}{2}} + r_{t+\frac{1}{2}}\right)$
14: **end for**
15: **half position update:** $\theta_T \leftarrow \theta_{T-1} + \frac{1}{2}\epsilon\sigma^{-2} r_{T-\frac{1}{2}}$
16: **new values:** $\theta^* \leftarrow \theta_T$, $r^* \leftarrow r_{T-\frac{1}{2}}$
17: $a \leftarrow \exp\left(U(\theta) - U(\theta^*) + \rho_{T-\frac{1}{2}}\right)$
18: **with probability** $\min(1, a)$, **update** $\theta \leftarrow \theta^*$, $r \leftarrow r^*$ (as long as $\theta^* \in \Theta$)
19: **otherwise update** $r \leftarrow -r_{-\frac{1}{2}}$
20: **end loop**

we use a type of leapfrog integration that starts and ends the outer loop with a half-position-update (Lines 5 and 15). This too is done in the interest of time-reversal-symmetry. Third, there is an additional term ρ in AMAGOLD, which we call the *energy accumulator*, which accumulates the log of the product in (6). Computing ρ requires little extra cost since all its terms are already obtained in the standard update. AMAGOLD is thus unbiased without adding too much cost over SGHMC. The following theorem summarizes AMAGOLD's asymptotic accuracy. (This follows from the construction; an explicit proof is in Appendix C.)

Theorem 1. *Consider the Markov chain described by AMAGOLD (Algorithm 2). If the momentum is resampled (on line 3), then this Markov chain is reversible. Otherwise the Markov chain is skew-reversible. In either case, its stationary distribution is π.*

Connection to previous methods AMAGOLD is related to several previous MCMC methods. When using a full-batch gradient, AMAGOLD becomes L2MC with amortized M-H-adjustment. Using a full-batch, $\beta = 0$, and resampling, AMAGOLD becomes HMC (Appendix D). If we disable AMAGOLD's M-H step (and adjust hyperparameters), it becomes SGHMC.



(a) SGHMC (b) AMAGOLD (c) Tuned AMAGOLD (d) KL Divergence

Figure 1: Estimated densities of (a) SGHMC and (b) AMAGOLD for step size 0.25 compared to the ground truth and (c) step size 0.01 for tuned AMAGOLD (see Section 5.2); (d) Comparison of symmetric KL divergence, varying step sizes for SGHMC and AMAGOLD.

Illustrating AMAGOLD To illustrate AMAGOLD is able to achieve unbiased stochastic MCMC, we test our method on a double-well potential (Ding et al., 2014; Li et al., 2016b):

$$U(\theta) = (\theta+4)(\theta+1)(\theta-1)(\theta-3)/14 + 0.5.$$

The target distribution is proportional to $\exp(-U(\theta))$. To simulate stochastic gradients, we let $\nabla\tilde{U} = \nabla U + \mathcal{N}(0,1)$. We show the results of SGHMC and AMAGOLD when $\beta = 0.25$, $T = 10$ and $\epsilon = 0.25$. Results for $\epsilon = \{0.05, 0.15\}$ are in Appendix G.1.

In Figure 1 and Appendix G.1, the estimated densities of AMAGOLD are very close to the true density on varying step sizes. In contrast, SGHMC does not converge to the correct distribution asymptotically. This is especially the case when the step size is large: SGHMC diverges from the true distribution. These observations validate Theorem 1, as AMA guarantees convergence to the target distribution. To quantitatively measure divergence from the true distribution, we plot the symmetric KL divergence as a function of the step size in Figure 1d. We can see that SGHMC is very sensitive to step size, and may require careful tuning in practice, while AMAGOLD is more robust.

5.1 Convergence Rate Analysis

Using stochastic gradients in MCMC can reduce the cost of each iteration. However, this does not mean the overall cost of the algorithm will be less in comparison to its non-stochastic counterpart. Rather, it is possible that the stochastic chain's convergence rate becomes much slower than the non-stochastic one. To be confident in the effectiveness of an SG-MCMC method, we must rule this out: We must show that the convergence speed of the stochastic chain is not slowed down, or at least not too much, compared to the non-stochastic chain. We do this analysis for AMAGOLD as follows.

Since AMAGOLD can be regarded as stochastic L2MC, we study reversible AMAGOLD's convergence rate relative to L2MC with an amortized M-H correction. Prior work has used this type of bound to prove the convergence rate of subsampled MCMC methods (De Sa et al., 2018; Zhang and De Sa, 2019). Unlike work that uses 2-Wasserstein, MSE, or empirical risk minimization to evaluate the convergence rate of SG-MCMC, we are the first to use the spectral gap—a traditional metric for evaluating MCMC convergence (Hairer et al., 2014; Levin and Peres, 2017) that is directly related to another common measurement, the mixing time (Levin and Peres, 2017). Our bound only requires mild assumptions compared to prior work (Vollmer et al., 2016; Chen et al., 2015), and we measure convergence to the target distribution directly, rather than empirical risk minimization (Gao et al., 2018).

The spectral gap γ of a reversible Markov chain with transition probability operator G is defined as the smallest distance between any non-principal eigenvalue of G and 1, the principal eigenvalue of G (Levin and Peres, 2017). The spectral gap determines the convergence rate of a Markov chain: a chain with a smaller γ will take longer to converge. To ensure the existence of γ, we assume geometric ergodicity of the full-batch chain (Rudolf, 2011). To bound γ, we assume the covariance of the gradient samples of AMAGOLD is bounded isotropically with

$$\mathbf{E}\left[(\nabla\tilde{U}(\theta)-\nabla U(\theta))(\nabla\tilde{U}(\theta)-\nabla U(\theta))^T\right] \preceq \frac{V^2}{d}I$$

for some constant $V > 0$. This sort of bounded-variance assumption is standard in the analysis of stochastic gradient algorithms.

The following theorem shows that with appropriate hyperparameter settings the convergence rate of AMAGOLD will not be slowed down by more than a constant factor.

Theorem 2. *For some parameters $\epsilon > 0$, $\sigma > 0$, and $\beta > 0$, let $\bar{\gamma}$ denote the spectral gap of the L2MC chain running with parameters $(\epsilon, \sigma, \beta)$. Assume that these parameters are such that $\epsilon V^2 \leq 4\sigma^2\beta d$. Define a constant $c = 1 + \sqrt{\frac{\epsilon V^2}{16\sigma^2\beta T d^2}}$. Let γ denote the spectral gap of AMAGOLD running with parameters $(\epsilon, \sigma \cdot c^{-1/4}, \beta \cdot c^{-1/2})$. Then,*

$$\frac{\gamma}{\bar{\gamma}} \geq \exp\left(-\frac{\epsilon T V^2}{4\sigma^2\beta} - \sqrt{\frac{\epsilon T V^2}{\sigma^2\beta}}\right).$$

This requirement on parameters is easy to satisfy because d is generally large and ϵ is generally small in practice. For the same reason, c is usually close to 1, so the parameters used by the two chains are very close.

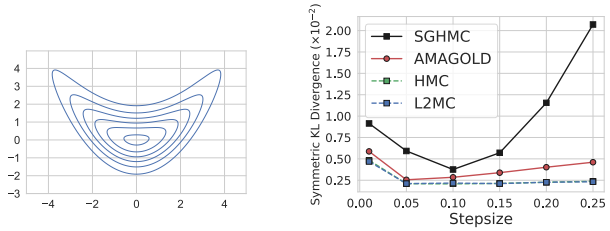
This theorem has three useful takeaways: First, AMAGOLD's convergence rate is essentially the same as L2MC up to a constant, which will approach 1 as the batch size increases (V decreases) or ϵ decreases. Second, it shows the effect of minibatching on convergence rate: if one reduces the minibatch size (i.e. V^2 increases), they can expect the convergence rate to decrease with a rate of $\exp(-O(V^2))$. Third, the theorem outlines a range of parameters (where $\epsilon T V^2 \ll \sigma^2\beta$) over which AMAGOLD converges at a similar rate to the full-batch algorithm.

5.2 AMAGOLD in Practice

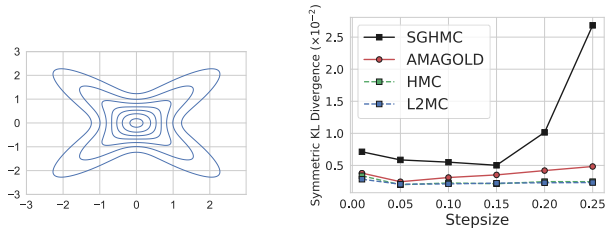
Here we describe some simple modifications that can further improve AMAGOLD's performance.

Minibatch M-H Amortizing the cost of an M-H correction over T steps is not always sufficient for achieving good performance on large datasets. This is because calculating the true energy U requires a scan over the whole dataset. We can further reduce the cost of a single correction by using minibatch M-H to compute the acceptance probability—using a minibatch at line 17 of Algorithm 2. Prior work has estimated the M-H correction using a subset of the data (Korattikara et al., 2014; Bardenet et al., 2014; Maclaurin and Adams, 2015; Seita et al., 2016). These methods are composable with, rather than exclusive with, AMAGOLD and could provide additional speed-ups.

Tuning the step size Our experiments on double well potential (Figure 1) show step size significantly influences SGHMC's performance. Besides being more robust to step size, AMAGOLD's step size can be more easily tuned. The M-H step's acceptance probability provides information about whether a step size is desirable. Based on this information, the step size can be tuned automatically to target some fixed acceptance probability during burn-in without affecting convergence. With a fixed step size $\epsilon = 0.01$, both AMAGOLD and SGHMC provide poor density estimates due to too small step size. However, when we



(a) Dist1 (b) KL comparison on Dist1



(c) Dist2 (d) KL comparison on Dist2

Figure 2: AMAGOLD's performance against baselines. In (b) and (d) the step size varies from 0.01 to 0.25; the symmetric KL divergence is a function of step size.

let AMAGOLD adjust ϵ such that the average M-H acceptance probability is 85%, it estimates the density accurately (Figure 1c, Appendix G.1).

6 Experiments

Here we validate our theory empirically and explore the performance of AMAGOLD on a variety of applications. We compare to full-batch baselines HMC and L2MC to show AMAGOLD is more efficient and we compare to SGHMC because, despite exhibiting bias, it is commonly used in the literature. Unless otherwise specified, we use reversible AMAGOLD, meaning we resample the momentum, $T = 10$ and $\beta = 0.25$. We set hyperparameters for AMAGOLD in a similar way as SGHMC (Chen et al., 2014). For simplicity, we do not use the techniques in Section 5.2. Additional details are in Appendix G. The code can be found at `https://github.com/ruqizhang/amagold`.

6.1 Synthetic Distributions

We conduct experiments on synthetic two-dimensional distributions (Figures 2a and 2c), which are adapted from (Yin and Zhou, 2018). The analytical expressions are in Appendix G.2.1. We compare our algorithm against three baselines: (1) HMC, (2) L2MC with amortized M-H correction, and (3) SGHMC. HMC and L2MC serve as non-stochastic, unbiased baselines. As in Chen et al. (2014), we replace ∇U by stochastic estimates $\nabla U = \nabla U + \mathcal{N}(0, I)$ for the stochastic methods. We draw 5×10^6 samples and use symmetric

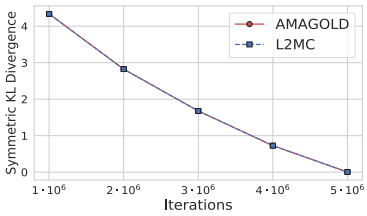


Figure 3: The convergence speed (symmetric KL divergence as a function of iterations) of AMAGOLD compared to L2MC on Dist1 with step size 0.15.

KL divergence as a function of step size to quantitatively evaluate the convergence of the Markov chain. On both distributions, AMAGOLD's symmetric KL divergence is close to full-batch methods and is much lower than SGHMC's, especially when the step size is large. This validates our theory that AMAGOLD is unbiased, while SGHMC's bias increases with step size. See Appendix G.2.3 for more details.

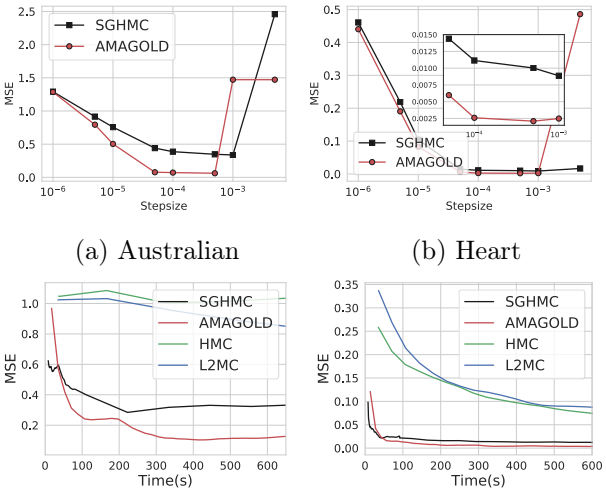
We then verify the theory that AMAGOLD has a comparable convergence rate to L2MC while using stochastic gradient estimates. Specifically, in Figure 3 AMAGOLD's convergence rate is the same as L2MC's (up to a constant factor slowdown of about 10^{-3}). We include runtime comparisons in Appendix G.2.2.

6.2 Bayesian Logistic Regression on Real-World Data

We evaluate our method on Bayesian logistic regression using two real-world datasets: *Australian* and *Heart* (Figure 4). We compute the MSE between the estimated and true parameters, obtained from 10^7 samples from HMC as in Li et al. (2016a). AMAGOLD exhibits smaller error than SGHMC on varying step sizes. We show runtime comparisons with step size 10^{-4}. Compared to full-batch HMC and L2MC, AMAGOLD is significantly faster due to minibatching. It is also not much slower than SGHMC, indicating AMA can reduce the cost of adding the M-H step. AMAGOLD's large error using a large step size is due to a drop in M-H acceptance probability (Appendix G.3). However, this drop can be easily avoided in practice. One can either set the step size such that it achieves a reasonable acceptance rate (usually 20–80%, depending on the application) or use the tuning technique in Section 5.2. With a reasonable acceptance rate, AMAGOLD achieves much lower error compared to SGHMC.

6.3 Bayesian Neural Networks

We apply AMAGOLD on Bayesian neural networks. The architecture is a MLP with two-layer with RELU non-linearities. The dataset size is 60000 and we use minibatch size 2000. We use irreversible AMAGOLD



(a) Australian (b) Heart

(c) Australian runtime (d) Heart runtime

Figure 4: We use two real-world datasets (a) *Australian* (15 covariates, 690 data points) and (b) *Heart* (14 covariates, 270 data points). The minibatch size is 32 and 16, respectively. We collect 5×10^6 samples and test step size varying from 10^{-6} to 5×10^{-3}.

Algorithm	b	$h = 0.0005$	$h = 0.001$
SGHMC	0.01	3.69±0.03	3.77±0.17
SGHMC	5e-6	89.95±0.29	89.70±0.91
AMAGOLD	0.01	3.63±0.04	3.65±0.08
AMAGOLD	5e-6	3.65±0.10	3.63±0.10

Table 2: Comparison between AMAGOLD and SGHMC of test error (%) ± standard error. We collect 20 samples in total.

since we find it gives better results. Similar to Zhang et al. (2020), to speed up the convergence of the sampling methods, we use SGD with momentum in the first 3 epochs as burn-in and then switch to either SGHMC or AMAGOLD.

Classification We evaluate the classification accuracy of AMAGOLD and SGHMC. As in Chen et al. (2014), we reparameterize our algorithm, setting $v = \epsilon\sigma^{-2}r, b = \epsilon\beta$ and $h = \epsilon^2\sigma^{-2}$ (Appendix F). This equivalent two-parameter reformulated update is similar to SGD with momentum and thus more easily tuned on DNNs. Table 2 shows the test error on various hyperparameter settings. AMAGOLD yields consistent test error, regardless of the hyperparameter values. In contrast, the performance of SGHMC is affected significantly by the hyperparameters. When b is small, SGHMC diverges. Similar performance of SGHMC has also been reported in Ding et al. (2014).

Uncertainty Evaluation We evaluate the sampling performance in terms of uncertainty evaluation, which is important in many ML applications (Lakshminarayanan et al., 2017; Blundell et al., 2015). We test predictive uncertainty estimation on out-of-distribution samples (Lakshminarayanan et al., 2017). The 8 models in Table 2 are tested on the notMNIST dataset (Bulatov, 2011). Since the models have never seen the samples from notMNIST, ideally the predictive distribution should be uniform, which gives the maximum entropy. We plot the empirical CDF for the entropy of the predictive distribution (Figure 5). AMAGOLD provides consistent uncertainty estimations on all settings, which aligns with the classification results. In contrast, when b is small or h is large, SGHMC performance suffers; it is overconfident about its prediction.

Both of these experiments indicate that SGHMC is very sensitive to hyperparameters. It needs to be carefully tuned to achieve desired performance on classification and uncertainty estimation. In contrast, AMAGOLD is robust to various hyperparameter settings because it is guaranteed to converge to the target distribution.

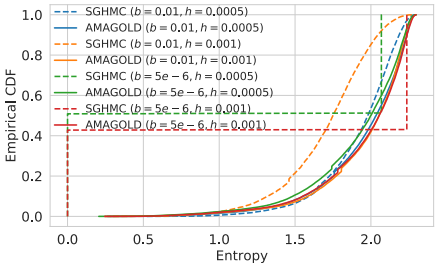


Figure 5: Empirical CDF on notMNIST dataset.

7 Conclusion

Our work represents a first step toward unbiased, efficient second-order SG-MCMC. We introduced AMAGOLD, which achieves these goals by infrequently applying the computationally-expensive Metropolis-Hasting adjustment step, amortizing the cost across multiple algorithm steps. We prove this is sufficient for convergence to the target distribution, and provide reversible and non-reversible versions for practical use. AMAGOLD's convergence rate is theoretically guaranteed: the bound captures the trade-off between the speed-up from minibatching and the convergence rate. Lastly, our work is complementary to, rather than exclusive with, other research in stochastic MCMC. In future work it would be interesting to explore combining AMA with other SG-MCMC variants (Ding et al., 2014; Zhang et al., 2017; Ma et al., 2015) and minibatch M-H methods (Korattikara et al., 2014; Bardenet et al., 2014; Maclaurin and Adams, 2015; Seita et al., 2016).

Acknowledgements

This work was supported in part by Huawei Technologies Co., Ltd.

References

Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

Shigeki Aida. Uniform positivity improving property, Sobolev inequalities, and spectral gaps. *Journal of functional analysis*, 158(1):152–185, 1998.

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *International Conference on Machine Learning (ICML)*, pages 405–413, 2014.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Yaroslav Bulatov. Not MNIST Dataset. 2011. http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html.

Changyou Chen, Nan Ding, and Lawrence Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pages 2278–2286, 2015.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International conference on machine learning*, pages 1683–1691, 2014.

Arnak S Dalalyan and Avetik Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 2019.

Christopher De Sa, Vincent Chen, and Wing Wong. Minibatch gibbs sampling on large graphical models. *arXiv preprint arXiv:1806.06086*, 2018.

Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In *Advances in neural information processing systems*, pages 3203–3211, 2014.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-hastings algorithms are fast! *arXiv preprint arXiv:1801.02309*, 2018.

Masatoshi Fukushima, Yoichi Oshima, and Masayoshi Takeda. *Dirichlet forms and symmetric Markov processes*, volume 19. Walter de Gruyter, 2010.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. Scalable bayesian learning of recurrent neural networks for language modeling. *arXiv preprint arXiv:1611.08034*, 2016.

Xuefeng Gao, Mert Gürbüzbalaban, and Lingjiong Zhu. Global convergence of stochastic gradient Hamiltonian Monte Carlo for non-convex stochastic optimization: Non-asymptotic performance bounds and momentum-based acceleration. *arXiv preprint arXiv:1809.04618*, 2018.

Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4): 549–581, 1994.

Martin Hairer, Stuart Andrew M., and Vollmer Sebastian J. Spectral gaps for a Metropolis–Hastings algorithm in infinite dimensions. *The Annals of Applied Probability 24, no. 6 (2014): 2455-2490*, 2014.

Alan M Horowitz. A generalized guided Monte Carlo algorithm. *Physics Letters B*, 268(2):247–252, 1991.

K Hukushima and Y Sakai. An irreversible Markov-chain Monte Carlo method with skew detailed balance conditions. In *Journal of Physics: Conference Series*, volume 473, page 012012. IOP Publishing, 2013.

Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *International Conference on Machine Learning*, pages 181–189, 2014.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016a.

Chunyuan Li, Changyou Chen, Kai Fan, and Lawrence Carin. High-order stochastic gradient thermostats for Bayesian learning of deep models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016b.

Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, and Lawrence Carin. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5666–5675, 2016c.

Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

Yi-An Ma, Tianqi Chen, Lei Wu, and Emily B Fox. A unifying framework for devising efficient and irreversible MCMC samplers. *arXiv preprint arXiv:1608.05973*, 2016.

Dougal Maclaurin and Ryan Prescott Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6): 1087–1092, 1953.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2 (11):2, 2011.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4): 1574–1609, January 2009. ISSN 1052-6234. doi: 10.1137/070704277. URL https://doi.org/10.1137/070704277.

Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. *arXiv preprint arXiv:1702.03849*, 2017.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

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

Gareth O Roberts, Richard L Tweedie, et al. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

Daniel Rudolf. Explicit error bounds for markov chain monte carlo. *arXiv preprint arXiv:1108.3201*, 2011.

Daniel Seita, Xinlei Pan, Haoyu Chen, and John Canny. An efficient minibatch acceptance test for Metropolis-Hastings. *arXiv preprint arXiv:1610.06848*, 2016.

O Stramer and RL Tweedie. Langevin-type models i: Diffusions with given stationary distributions and their discretizations. *Methodology and Computing in Applied Probability*, 1(3):283–306, 1999.

Konstantin S Turitsyn, Michael Chertkov, and Marija Vucelja. Irreversible Monte Carlo algorithms for efficient sampling. *Physica D: Nonlinear Phenomena*, 240(4-5):410–414, 2011.

Sebastian J Vollmer, Konstantinos C Zygalakis, and Yee Whye Teh. Exploration of the (non-) asymptotic bias and variance of stochastic gradient Langevin dynamics. *The Journal of Machine Learning Research*, 17(1):5504–5548, 2016.

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, 2011.

Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. *arXiv preprint arXiv:1805.11183*, 2018.

Ruqi Zhang and Christopher M De Sa. Poisson-minibatching for gibbs sampling with convergence rate guarantees. In *Advances in Neural Information Processing Systems*, pages 4923–4932, 2019.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *International Conference on Learning Representations*, 2020.

Yizhe Zhang, Changyou Chen, Zhe Gan, Ricardo Henao, and Lawrence Carin. Stochastic gradient monomial gamma sampler. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3996–4005. JMLR. org, 2017.