G8: Convergence Time of Langevin Algorithms

Rohan Deb, Siddhartha Laghuvarapu, Chandni Nagda, Mayank Shrivastava

Abstract

We present an empirical investigation into the convergence time of Langevin algorithms, specifically focusing on the validity of the recent claim of dimension-free convergence in Freund et al. [2021]. We first verify the theoretical results presented in the paper by testing the convergence of the algorithm on different instantiated distributions which meet the given assumptions. Additionally, we propose and test a Metropolis-adjusted variant or the dimension-free algorithm. Finally, we attempt to empirically verify whether dimension-free convergence holds under relaxed assumptions, and is thus applicable to neural network tasks. Our results suggest that the dimension-free convergence of Langevin algorithms holds under reasonable assumptions and that it is a promising direction for future experiments and research in sampling algorithms.

1 Introduction and Motivation

Langevin sampling algorithms are widely recognized for their success in generative tasks when used with deep learning techniques. An active area of research is to understand the convergence guarantees of these algorithms. While many convergence rates for Langevin and accelerated variants have been proposed based on different assumptions, their mixing time remains dependent on the problem's dimensionality. Gradient descent, on the other hand, has dimension-free convergence rates for convex problems in the optimization. Recently, a modified Langevin algorithm Freund et al. [2021] has been proposed that along with certain assumptions, claims to achieve dimension-independent convergence in the sampling domain. While this work produced remarkable results, it has not been empirically confirmed or subjected to broader applications. Our focus here is to validate the algorithm's dimensional independence, augment the existing work with Metropolis-Hastings adjustment, and explore the possibility of relaxing stringent assumptions.

A dimension-free convergence rate for Langevin algorithms has the potential for revolutionary impact. It would allow us to sample from high-dimensional spaces in a more efficient manner, which has implications for generative models and other machine learning tasks.

2 Related Work

The convergence rate of Langevin dynamics and its variants has been studied extensively in the literature. The seminal work of Roberts and Tweedie [1996] showed that the continuous Langevin dynamics converge exponentially fast to the target distribution under certain conditions. Based on this work, various algorithms have been proposed, including the Underdamped Langevin Algorithm,, the Riemannian Langevin Algorithm (RLA) Gatmiry and Vempala [2022], and stochastic gradient Langevin dynamics (SGLD) Welling and Teh [2011]. These algorithms have been successfully applied to various applications such as generative modeling.

In a noteworthy development, a novel framework for understanding the convergence properties of Langevin dynamics has been proposed in Freund et al. [2021]. The authors showed that, under certain conditions, the convergence rate of Langevin dynamics can be made independent of the dimension of the parameter space. This result is remarkable since, for existing algorithms, the convergence rate deteriorates as the dimensionality of the problem increases.

More recently, Chen and Gatmiry used a similar analysis to show that the mixing time of the Metropolis-Adjusted Langevin algorithm (MALA) Dwivedi et al. [2018] can be made to depend on the trace of the Hessian of the target density, rather than the dimension itself, when the target distribution satisfies an isoperimetry.

3 Problem Formulation

The authors of Freund et al. [2021] propose a composite optimization approach to obtain a bound on the mixing time of Langevin dynamics that is independent of the dimension of the parameter space. They consider the problem of sampling from a posterior distribution over a parameter w given a dataset $\mathbf{z}$, where $p(w|\mathbf{z}) \propto \exp(-U(w))$. The potential function U decomposes into two parts, $U(w) = \beta^{-1}(f(w) + g(w))$. In the context of machine learning, $f(w)$ corresponds to the negative log-likelihood and $g(w)$ corresponds to the negative log-prior. The authors assume that g is m-strongly convex and can be explicitly integrated. They then consider two cases regarding the regularity of the function f. Either f is G-Lipschitz continuous or f is L-Lipschitz smooth and the result has a ridge separable structure.

The proposed Unadjusted Langevin Algorithm with Prior Diffusion (ULA-PD) algorithm samples from a posterior distribution over parameter $w \in \mathbb{R}^d$, given an initial distribution p_0 on $\mathbb{R}^d$, and a stepsize η_t. At each iteration t, ULA-PD samples a perturbed point $\widetilde{w}_t \sim \mathcal{N}(e^{-m\eta_t} w_{t-1}, \frac{1-e^{-2m\eta_t}}{m}\beta\mathbf{I})$, and performs a gradient descent step on f evaluated at the perturbed point, resulting in a final estimate $\widetilde{w}_T$. The full algorithm is provided in Appendix 1.

4 Methodology and Experiments

4.1 Dimension dependence

Based on the experiments in Dwivedi et al. [2018], we perform an empirical analysis of the mixing time of the ULA-PD algorithm for two choices of the target distribution $\pi(w)$: Multivariate Gaussian distribution and Mixture of Gaussians (MoG). We compare dimensional dependence using approximate mixing time, trace plots and autocorrelation as defined below.

4.1.1 Multivariate Gaussian via mixing time

The objective of this simulation is to illustrate how the dimensionality affects the performance of ULA, MALA, and MRW algorithms in terms of mixing time and compare it with the dimensional free version of ULA, ULA-PD when sampling from a non-isotropic multivariate Gaussian distribution defined as $\pi(x) \propto e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$. where μ and Σ are the mean and variance. Recall that for ULA-PD we need to decompose $U(x) = -\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)$ to $f(x) + g(x)$ where $g(x) = \frac{m}{2}\|x\|^2$. Therefore we need to compute f such that $e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \propto e^{-f(x)-\frac{m}{2}\|x\|^2}$ This leads to $f(x) = (x-\mu_f)^T\Sigma_f^{-1}(x-\mu_f)$ where, $\mu_f = x - (\Sigma^{-1} - mI)^{-1}\Sigma^{-1}\mu$ and $\Sigma_f = (\Sigma^{-1} - mI)^{-1}$.

To compute an estimate of the mixing time we calculate the error in the 75th percentile of the sample distribution compared to the true distribution along the least favorable direction, which corresponds to the eigenvector of Σ associated with the largest eigenvalue. Then we choose a tolerance δ and denote the approximate mixing time, $\hat{k}_{mix}(\delta)$ to be the q the smallest iteration at which this error falls below a given threshold δ.

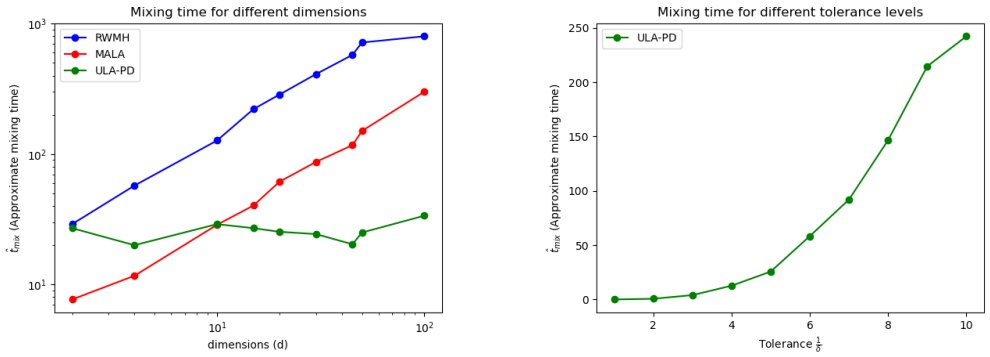
Figure 1 reveals that the dimension dependency of MALA, ULA, and RWMH are all close to order d. On the other hand, ULA-PD clearly has no dependence on d. This verifies the claims in Freund et al. [2021]. Figure 1a shows mixing time with respect to error tolerance. Mixing time increases with decreasing error tolerance with near quadratic dependence. The details regarding the μ and Σ can be found in the appendix.

4.1.2 Mixture of Gaussians via trace and autocorrelation plots

Next we examine the problem of sampling from a Gaussian mixture distribution with two components where the target density of the mixture distribution is given by

$$\pi(x) = \frac{1}{2\sqrt{2\pi}}\left(e^{-\frac{\|x-a\|^2}{2}} + e^{-\frac{\|x+a\|^2}{2}}\right),$$

where $a \in \mathbb{R}^d$ is a fixed vector. Here the corresponding U is given by: $U(x) = \frac{1}{2}\|x-a\|_2^2 - \log(1+e^{-2x^Ta})$. As before we need to compute f such that $U(x) = f(x) + g(x)$ for $g(x) = \frac{m}{2}\|x\|^2$. Here $e^{-f(x)}$ would again



(A) (B)

FIGURE 1: Scaling of the approximate mixing time $\hat{k}_{mix}$ (refer to the Section 3 for definition) on multivariate Gaussian density. Note that RWMH denotes the standard metropolized random-walk algorithm. (A) Dimension dependency of mixing time for $\delta = 0.2$. (B) Error-tolerance dependency on mixing time.

be give by a mixture of Gaussian with

$$e^{-f(x)} \propto e^{-\frac{(1-m)}{2}\left\|x-\frac{a}{1-m}\right\|^2} + e^{-\frac{(1-m)}{2}\left\|x+\frac{a}{1-m}\right\|^2}.$$

The corresponding f and gradient are then given by:

$$f(x) = \frac{(1-m)}{2}\left\|x-\frac{a}{1-m}\right\|^2 + 2a\log(1+e^{-2x^Ta}), \quad \nabla f(x) = (1-m)\left(x-\frac{a}{(1-m)}\right) + 2a(1+e^{2x^Ta})$$

Instead of computing approximate mixing time we study the trace plot and the autocorrelation function.

1. **Trace Plot:** The trace plot shows the sampled values along a specific dimension. In the trace plots, it is desirable to minimize prolonged periods of inactivity (where the chain remains in the same state for an extended duration) or excessive consecutive steps in a single direction.

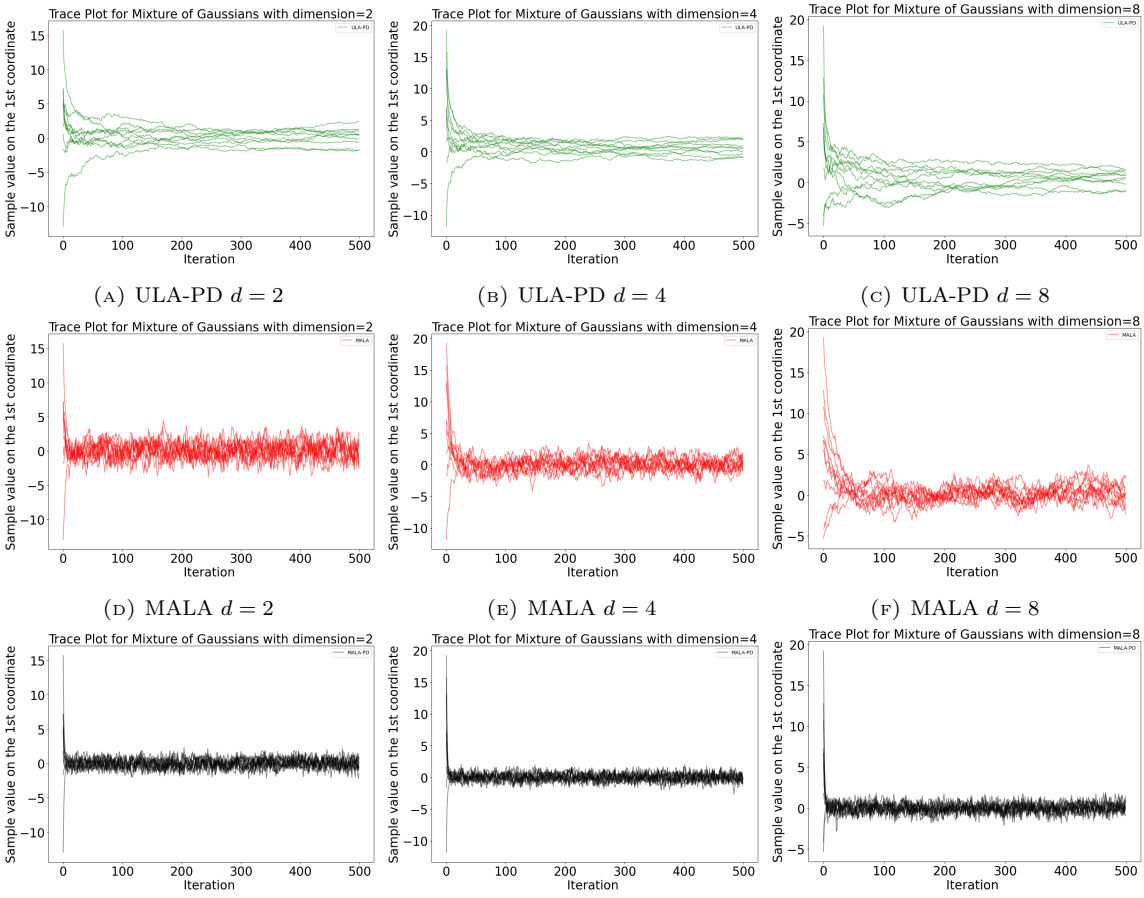
2. **Autocorrelation:** An additional method for assessing convergence is by examining the autocorrelations among the samples generated by the MCMC. Autocorrelation measures the correlation between a sample and the sample obtained k steps prior. Ideally, as the lag-k increases, the autocorrelation should decrease, indicating that the samples are becoming more independent. However, if the autocorrelation remains high for larger values of k, it suggests a strong correlation among the samples and slow mixing.

We compare the trace plots of MALA, ULA-PD, and MALA-PD in 2. With MALA, we observe that mixing time increases with dimension. In the case of ULA-PD, the chain does not mix well due a step size which decreases with time. However, MALA-PD allows us to set fixed step size, leading to improved mixing times.

Finally, we compare the algorithms using the autocorrelation plot shown in Figure 8 in Appendix. It is worth noting that ULA-PD shows high autocorrelation values for various lags, indicating that the samples are not uncorrelated. This indicates that that it takes longer for the chain to reach the stationary distribution. Again, we believe this is due to the decreasing step size. We thus propose a variant that allows us to use a constant step size, resulting in faster convergence. The details regarding the choice of a and m for the experiment can be found in the appendix.

4.2 Metropolis Adjusted Langevin Algorithm with Prior Diffusion (MALA-PD)

We extend the Langevin algorithm and use Metropolis Hastings accept-reject step. The idea is once $\tilde{w}_t$ is generated in step 3 of Algorithm-1, we perform an accept reject step by computing α_t as in step and if



(A) ULA-PD $d = 2$ (B) ULA-PD $d = 4$ (C) ULA-PD $d = 8$

(D) MALA $d = 2$ (E) MALA $d = 4$ (F) MALA $d = 8$

(G) MALA-PD $d = 2$ (H) MALA-PD $d = 4$ (I) MALA-PD $d = 8$

FIGURE 2: Trace-plots of the first coordinate on a two component Gaussian mixture.

accepted, take a gradient step from $\tilde{w}_t$ along $\nabla f(\tilde{w}_t)$, else stay at w_{t-1}. We call this version MALA-PD and provide the full algorithm in Appendix 2.

As indicated by the results presented in Figure 2, the MALA-PD algorithm exhibits faster convergence for all dimensions compared to other methods. However, we also observe that for large step sizes, the algorithm converges to an incorrect distribution, where the mean is correct but the variance is different. This issue has been previously reported in the literature Zhang et al. [2022]. To address this problem, we conducted experiments with different step sizes and found that reducing the step size to 0.01 leads to convergence to the correct multivariate distribution, as illustrated in Figure 5. Nevertheless, selecting an appropriate step size in general remains a challenging problem, and further work is needed to develop methods for controlling the variance of the MALA-PD algorithm, as suggested in Zhang et al. [2022].

4.3 Relaxing the convexity assumption on negative log-likelihood

We explore the possibility of relaxing the convexity assumption on f. Towards this we assume that f satisfies the weaker PL condition:

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f^*)$$

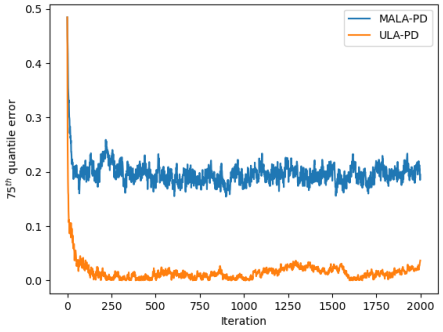


FIGURE 3: Decrease in error over iterations

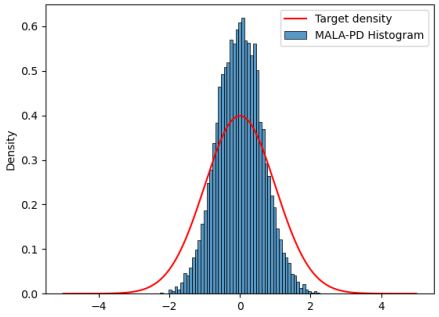


FIGURE 4: Samples drawn from MALA-PD with step size = 0.1

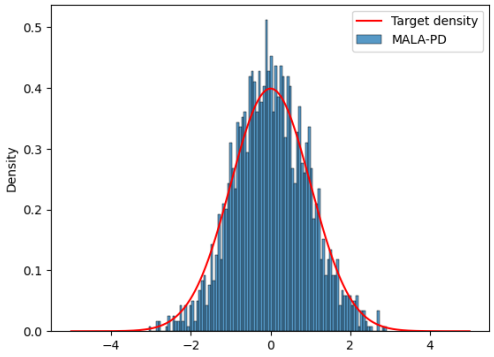


FIGURE 5: Samples drawn from MALA-PD with step size = 0.01

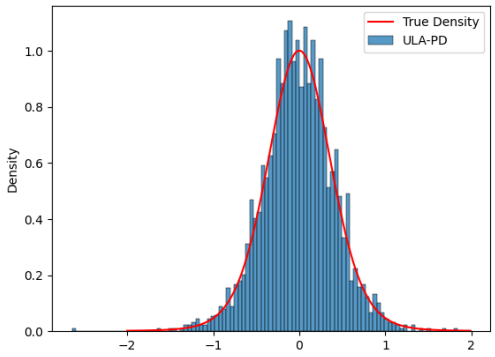


FIGURE 6: Comparing samples drawn by ULA-PD with the True Distribution for a non-convex f that satisfies PL condition

The PL condition has gained significant interest in recent times because neural losses for over-parameterized models satisfy the PL condition. Here we run ULA-PD on $f(x) = x^2 + 3\sin^2 x$ which satisfies the PL condition with $\mu = 1/8$. Figure 6 plots the true posterior distribution along with histogram of samples drawn by ULA-PD. We observe that empirically ULA-PD seems to converge for this choice of f.

4.4 Application to Bayesian neural networks

We further investigate the practical implications of relaxing the convexity assumption on f to satisfying the PL condition with a simple experiment using Bayesian Neural Networks. Bayesian deep learning is a compelling application of sampling methods. At their core, such methods aim to learn a probability distribution over the weights of a model, rather than a point esimate. Unlike convential learning, Bayesian methods are able to capture the uncertainty of learned parameters.

Let us consider the prior distribution over the weights of the neural network, represented by $p(w)$. Additionally, let $\mathcal{D}_x$ be the input data, and $\mathcal{D}_y$ be the labels. The likelihood function, which defines the conditional probability of the data given the weights, is denoted by $p(\mathcal{D}_y|\mathcal{D}_x, w)$. The goal is to compute the posterior distribution over the weights, incorporating the data and any prior knowledge: $p(w|\mathcal{D}_x, \mathcal{D}_y) = \frac{p(\mathcal{D}_y|\mathcal{D}_x, w)p(w)}{p(\mathcal{D}_y|\mathcal{D}_x)}$. The marginal likelihood, $p(\mathcal{D}_y|\mathcal{D}_x)$, can be expanded to $p(\mathcal{D}_y|\mathcal{D}_x) = \int p(\mathcal{D}_y|\mathcal{D}_x, w)p(w)dw$, which is computationally challenging. As a result, Monte Carlo methods like stochastic Langevin gradient descent (SGLD) are

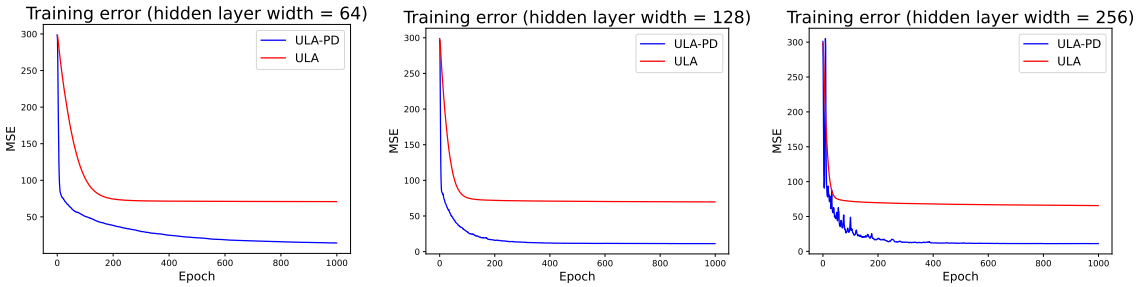


FIGURE 7: Training error of Bayesian Neural Networks (BNNs) with 1 hidden layer of width 64, 128, or 256.

utilized to optimize the neural networks. The objective function during training is the negative log-posterior, $-\log p(w|\mathcal{D}_x, \mathcal{D}_y) \propto -\log p(\mathcal{D}_y|\mathcal{D}_x, w) - \log p(w)$. The update rule for SGLD is given by:

$$w_{t+1} = w_t + \frac{\eta}{2}\left(\nabla \log p(\mathcal{D}_y|\mathcal{D}_x, w_t) + \nabla \log p(w_t)\right) + \boldsymbol{\epsilon}_t \tag{0.1}$$

We adopt the framework for use with the SGLD-PD algorithm given in Freund et al. [2021]. Recall that the prior diffusion step relies on decomposing the potential function $U(w)$ into two parts, $f(w)$ and $g(w)$. In the Bayesian setting, we let $f(w)$ correspond to the negative log-likelihood and $g(w)$ correspond to the negative log-prior. Since we assume a Gaussian prior distribution $p(w)$, $\log p(w)$ becomes the regularization term $\frac{m}{2}||w||^2$. As $f(w)$ relies on the output of a neural network, it does not satisfy the previously stated convexity assumptions. However, it would satisfy the PL condition for overparametrized neural networks. Then, rather than the single SGLD update rule, the algorithm is broken into two steps: diffusion along the integrable prior distribution, and gradient descent on the log-likelihood.

In our experiments, we generate a synthetic regression dataset as follows. We sample $n = 400$ input data points $x \sim U(-3, 3)$. To generate labels, we transform the data points using a RBF kernel and perturb with noise $C(x) = \exp(-0.5x^2) + \sigma^2$, where $\sigma = 0.3$. Labels are sampled as $y \sim \mathcal{N}(0, C(x))$. We then train Bayesian networks with one hidden layer, of varying width, which use either the SGLD or SGLD-PD algorithm. 7 shows the training error over 1000 epochs. Networks trained with SGLD-PD converge to a lower mean squared error (MSE) than networks trained with SGLD. We also note that as the width of the hidden layer, and thus dimension, increases, training with SGLD-PD become more noisy and unstable. It is unclear why this occurs. Further work is required to determine whether it is indeed possible to train truly overparametrized networks using such an algorithm, and if the claimed dimension-free convergence provides practical benefits.

5 Conclusion

In this work, we investigated the dimension-free convergence properties of Unadjusted Langevin Algorithm with Prior Diffusion (ULA-PD), proposed in Freund et al. [2021], and verified that the theoretical results hold for simple distributions which meet the given assumptions. We further proposed augmenting the algorithm with a Metropolis-Hasting accept-reject step and attempted to empirically investigate whether the convexity assumption on negative log-likelihood may be relaxed to the PL condition. Based on our results, there are several opportunities for future work. It may be possible to provide a theoretical analysis for ULA-PD and for the relaxation of the convexity assumption to the PL condition. Additionally, further work is required to determine whether prior diffusion algorithms will be advantageous in tasks such as Bayesian deep learning. Our early results hint that this will not be a straightforward application. Lastly, we believe the the dimension-free convergence bound may be extended to accelerated methods such as Hamiltonian Monte Carlo and Higher-order Langevin Dynamics.

References

Y. Chen and K. Gatmiry. A simple proof of the mixing of metropolis-adjusted langevin algorithm under smoothness and isoperimetry. URL http://arxiv.org/abs/2304.04095.

R. Dwivedi, Y. Chen, M. J. Wainwright, and B. Yu. Log-concave sampling: Metropolis-hastings algorithms are fast. 2018. doi: 10.48550/ARXIV.1801.02309. URL https://arxiv.org/abs/1801.02309.

Y. Freund, Y.-A. Ma, and T. Zhang. When is the convergence time of langevin algorithms dimension independent? a composite optimization viewpoint, 2021. URL https://arxiv.org/abs/2110.01827.

K. Gatmiry and S. S. Vempala. Convergence of the riemannian langevin algorithm, 2022. URL https://arxiv.org/abs/2204.10818.

G. O. Roberts and R. L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2:341–363, 1996.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. ICML'11, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

R. Zhang, A. G. Wilson, and C. D. Sa. Low-precision stochastic gradient langevin dynamics, 2022.

6 Appendix

Algorithm 1: Unadjusted Langevin Algorithm with Prior Diffusion (ULA-PD)

Input: Initial distribution p_0 on $\mathbb{R}^d$, stepsize η_t, $\beta = 1$

Draw w_0 from p_0
for $t = 1, 2, \ldots, T$ **do**
  Sample $\widetilde{w}_t \sim \mathcal{N}\left(e^{-m\eta_t} w_{t-1}, \frac{1-e^{-2m\eta_t}}{m}\beta \mathrm{I}\right)$, where dB_s is the standard Brownian motion on $\mathbb{R}^d$.
  Let $w_t = \widetilde{w}_t - \widetilde{\eta}_t \nabla f(\widetilde{w}_t)$
end
return $\widetilde{w}_T$

Algorithm 2: Metropolis Adjusted Langevin Algorithm with Prior Diffusion (MALA-PD)

Input: Initial distribution p_0 on $\mathbb{R}^d$, stepsize η_t, $\beta = 1$

Draw w_0 from p_0
for $t = 1, 2, \ldots, T$ **do**
  Sample $\widetilde{w}_t \sim \mathcal{N}\left(e^{-m\eta_t} w_{t-1}, \frac{1-e^{-2m\eta_t}}{m}\beta \mathrm{I}\right)$, where dB_s is the standard Brownian motion on $\mathbb{R}^d$.
  $\alpha_t = \min\{1, \frac{\exp -g(w_t) - \|\tilde{w}_t - w_t + \eta \nabla g(\tilde{w}_t)\|/4\eta}{\exp -g(w_t) - \|w_t - \tilde{w}_t + \eta \nabla g(\tilde{w}_t)\|/4\eta}\}$
  Generate u_t from Uniform[0,1]
  if $u_t \leq \alpha_t$ **then**
    $w_t = \tilde{w}_t - \eta \nabla f(\tilde{w}_t)$
  end
  else
    Go to step 2.
  end
end
return $\widetilde{w}_T$

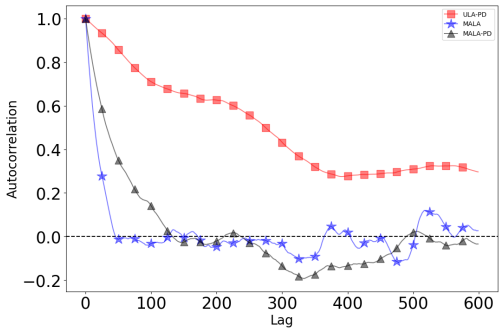


FIGURE 8: Markov chain autocorrelation function plot. The burn-in time for the plot is set to 300 iterations and d=8.

7 Choice of parameters for experiments

For all experiments we set the strong-convexity parameter $m = \frac{1}{8}$. For multivariate gaussian, we set Σ as a diagonal matrix with the largest eigenvalue 4.0 and the smallest eigenvalue 1.0 so that the $\kappa = 4$ is fixed across different settings. For a fixed dimension d, we simulate 3 independent runs of the three chains each with $N = 10,000$ samples to determine the approximate mixing time. We set $\mu = 0$. For mixture of gaussians, we set $a = \frac{1}{\sqrt{2d}}[1, 1, \cdots, 1]^T$ and $\sigma = 1$.