MA578 Project Report: **Scalable Bayesian Learning Using Stochastic Gradient Langevin Dynamics**

Xuelei Chen (xuelei@bu.edu)

December, 2022

Abstract

Bayesian learning treats parameters as random variables and can model the uncertainty of these parameters. Markov chain Monte Carlo (MCMC) algorithms are widely used in Bayesian learning. MCMC methods are essentially sampling methods to estimate posterior distributions. Every iteration of MCMC requires computation over the whole dataset, thus being computationally inefficient. Recent advances have mitigated this issue. This project investigates two recently proposed sampling methods for bayesian inference: Stochastic Gradient Langevin Dynamics (SGLD) [1] and preconditioned Stochastic Gradient Langevin Dynamics (pSGLD) [2]. Experiments were conducted on synthetic data and imbalanced MNIST data to show the advantages of SGLD/pSGLD over traditional sampling method MALA [3] and traditional stochastic optimization method SGD [4].

1 Introduction

Machine learning aims to learn a model from finite sample data. Then this model can be used to finish some tasks like classification or prediction. In parametric machine learning, the model is parameterized. Many methods have been proposed to estimate the parameters of the model. But most of them only give a point estimation, for example, MAP and MLE. In contrast, Bayesian learning gives a distribution estimation of the parameters. Distribution estimation contains more information and can model the uncertainty of parameters.

Gradient Descent (GD) can be used to find MAP or MLE. When there exists a large amount of data, Stochastic Gradient Descent (SGD) is more computationally efficient than GD. Based on SGD, many variants like RMSProp[5], Adagrad [6] and Adam [7] have been proposed.

Metropolis-adjusted Langevin algorithm (MALA) is a sampling method motivated by Langevin dynamics. It also faces the difficulty of calculating the posterior and gradient on the whole large-scale dataset. Combining the idea of SGD with MALA, we can get a scalable bayesian learning method SGLD. Similarly, combining the idea of RMSProp with MALA, we can get another scalable bayesian learning method pSGLD. Different from MALA, both SGLD and pSGLD remove the Accept/Reject step by annealing the stepsize.

The remaining part of this report is organized as follows: Section 2 describes the idea and formulation of investigated methods SGLD and pSGLD. Section 3 gives a nonrigorous analysis of the convergence property. Section 4 presents experiments conducted and compares results from SGLD/pSGLD with results from MALA or SGD. Section 5 concludes the report with some discussion.

2 Method

2.1 MALA and ULA

The Langevin equation describes the random movement of a particle in the fluid. When the particle inertia is negligible compared to the damping force, the dynamics of the particle can be described using the overdamped Langevin equation [8] as

$$\lambda dX_t = -\nabla V(X_t)dt + \sigma(X_t)dB_t \tag{1}$$

where X_t is the position of the particle, λ is the velocity-friction coefficient, V is the potential of the particle, and $\sigma(X_t)dB_t$ is the noise term representing molecule collision.

In the case of MCMC, Equation 1 can be rewritten as

$$dX_t = \nabla \log \pi(X_t)dt + \sqrt{2}dB_t \tag{2}$$

where X_t is the random variable of interest, π is the unnormalized density of X_t, and B_t is the Gaussian noise.

We can approximate Equation 2 using Euler-Maruyama discretization with positive fixed time step δ. Then we get the following update rule:

$$X_{k+1} = X_k + \delta\nabla \log \pi(X_k) + \sqrt{2\delta}B_k, \quad B_k \sim N(0, I_d) \tag{3}$$

Then the **Metropolis-Adjusted Langevin algorithm (MALA)** uses Equation 3 as the proposal distribution, which can be denoted as $q(y|x)$. The accept/reject step after the proposal step uses the Metropolis-Hasting algorithm to accept the proposal or reject it and remain unchanged. This can be formulated as:

$$X_{k+1} = \begin{cases} \tilde{X}_{k+1}, & \text{if} \quad U_k \leq \min(1, \frac{\pi(\tilde{X}_{k+1})q(X_k|\tilde{X}_{k+1})}{\pi(X_k)q(\tilde{X}_{k+1}|X_k)}), \quad U_k \sim U[0,1] \\ X_k \end{cases} \tag{4}$$

The **Unadjusted Langevin Algorithm (ULA)** removes the accept/reject step compared to MALA. This will cause the stationary distribution biased from π. But the bias converges to 0 as $\delta \to 0$ [9].

2.2 SGLD and pSGLD

In Bayesian learning, we want to estimate the distribution of model parameters given sample data. Then Equation 3 can be rewritten as

$$\theta_{\mathbf{k+1}} = \theta_{\mathbf{k}} + \delta\nabla \log \pi(\theta_{\mathbf{k}}|\mathcal{D}) + \sqrt{\mathbf{2}\delta}\mathbf{B_k}, \quad \mathbf{B_k} \sim \mathbf{N}(\mathbf{0}, \mathbf{I_d}) \tag{5}$$

where θ is the model parameter vector, δ is the step size, $\mathcal{D} = \{x_i\}_{i=1}^N$ is the sample data, and $\pi(\theta_\mathbf{k}|\mathcal{D})$ is the posterior density function of model parameter vector. By Bayes' Theorem,

$$\log \pi(\theta_\mathbf{k}|\mathcal{D}) = \log \mathbf{p}(\theta_\mathbf{k}) + \sum_{\mathcal{D}} \log \mathbf{p}(\mathbf{x_i}|\theta_\mathbf{k}). \tag{6}$$

Then substitute Equation 6 into Equation 5 and we get the following proposal function

$$\theta_\mathbf{k+1} = \theta_\mathbf{k} + \delta \left(\nabla \log \mathbf{p}(\theta_\mathbf{k}) + \sum_{\mathcal{D}} \nabla \log \mathbf{p}(\mathbf{x_i}|\theta_\mathbf{k}) \right) + \sqrt{\mathbf{2}\delta}\mathbf{B_k}, \quad \mathbf{B_k} \sim \mathbf{N}(\mathbf{0}, \mathbf{I_d}) \tag{7}$$

The essence of **SGLD** is to use batch data to estimate the gradient of whole data:

$$\sum_{\mathcal{D}} \nabla \log p(x_i|\theta) \approx \frac{\mathbf{N}}{\mathbf{n}} \sum_{\mathcal{B}} \nabla \log \mathbf{p}(\mathbf{x_j}|\theta) \tag{8}$$

where $\mathcal{B} = \{x_j\}_{i=1}^n$ is the set of data points in a batch.

pSGLD borrows the idea of rescaling different coordinates of a gradient vector from RMSProp. The rescaling is realized by constructing a preconditioning matrix $G(\theta_\mathbf{k})$ at each iteration. The value of the preconditioning matrix depends on exponentially smoothed squared-gradient $V(\theta_\mathbf{k})$. The motivation behind this is that a larger squared gradient usually brings about fluctuations in the loss surface. The fluctuations will impede the algorithms from converging. The detailed formulation of $V(\theta_\mathbf{k})$ and $G(\theta_\mathbf{k})$, and proposal function are as follows:

$$V(\theta_\mathbf{k+1}) = \beta \mathbf{V}(\theta_\mathbf{k}) + (\mathbf{1} - \beta) \nabla \log \pi(\theta_\mathbf{k}|\mathcal{D}) \odot \nabla \log \pi(\theta_\mathbf{k}|\mathcal{D}) \tag{9}$$

$$G(\theta_\mathbf{k} + \mathbf{1}) = \mathbf{diag}(\mathbf{1} \oslash (\lambda \mathbf{1} + \sqrt{\mathbf{V}(\theta_\mathbf{k+1})})) \tag{10}$$

$$\theta_\mathbf{k+1} = \theta_\mathbf{k} + \delta \mathbf{G}(\theta_\mathbf{k}) \left(\nabla \log \mathbf{p}(\theta_\mathbf{k}) + \sum_{\mathcal{D}} \nabla \log \mathbf{p}(\mathbf{x_i}|\theta_\mathbf{k}) \right) + \mathbf{G}(\theta_\mathbf{k})^{\frac{1}{2}} \sqrt{\mathbf{2}\delta}\mathbf{B_k}, \quad \mathbf{B_k} \sim \mathbf{N}(\mathbf{0}, \mathbf{I_d}) \tag{11}$$

Note that both SGLD and pSGLD do not have an accept/reject step. They must have a decaying learning rate to theoretically guarantee that the bias converges to zero.

3 Analysis

In this section, I will present my understanding of some assumptions and convergence properties. They are not rigorous proofs.

In stochastic optimization, a common requirement for step size is $(1)\sum^\infty \delta_t = \infty$; $(2)\sum^\infty(\delta_t^2) < \infty$. The first condition ensures that the step size is large enough to lead the point to the high-probability region no matter where the initial point is. The second condition ensures the point converges instead of fluctuating.

The reason for removing accept/reject step in SGLD is that the variance from injected noise is $2\delta_t$ while the variance from stochastic gradient is $\delta_t^2 V(\theta_t)$. So the injected noise dominates because $\delta_t \to 0$. Then the stochasticity error from batch gradient estimation can be neglected. We get an approximate Langevin dynamics.

An issue in the previous analysis is that infinitesimal δ_t when t is large can cause bad mixing rate and high correlation in the sampled sequence. A trade-off is to keep the step size at a low constant value after the SGLD algorithm enter the Langevin dynamics phase.

In the paper of SGLD and pSGLD, there are still many math-heavy proofs for different error bounds. Understanding those proofs requirs knowledge of stochastic differential equations. I tried to understand them all but time does not permit.

4 Experiments

In this section, I conducted two experiments to investigate the performance of SGLD and pSGLD using synthetic data and MNIST data respectively.

4.1 Sythetic Data

I sample 10000 points from a **RingMixture** distribution with $r_1 = 1$ and $r_2 = 2$, as visualized in Figure 1. The unnormalized log density function of **RingMixture** is

$$\log f(x) = \log(\frac{1}{2\sigma\sqrt{2\pi}} \exp(-\frac{(\|x\| - r_1)^2}{2\sigma^2}) + \frac{1}{2\sigma\sqrt{2\pi}} \exp(-\frac{(\|x\| - r_2)^2}{2\sigma^2})) \tag{12}$$

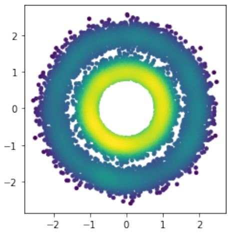


Figure 1: Sampled points

The goal of this task is to estimate parameters r_1 and r_2 from these sampled points. Prior distributions for r_1 and r_2 are both Gaussian $N(0, 100)$.

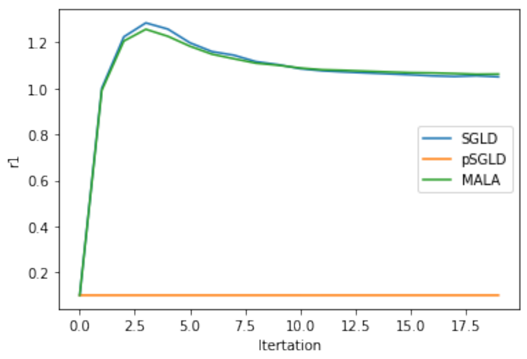
Three methods MALA, SGLD and pSGLD are implemented. They all use the same annealing step sizes that satisfy this function: $\delta_k = 10^{-5} * (10^{-4} + k)^{-0.55}$. All these three algorithms are trained for 10000 steps.

Firstly, the execution time is compared in Table 1. We can see that MALA takes longer time to finish. This is because MALA use all 10000 sample points to calculate the gradient at each iteration. However, both SGLD and pSGLD only use 100 points to calculate the gradient.

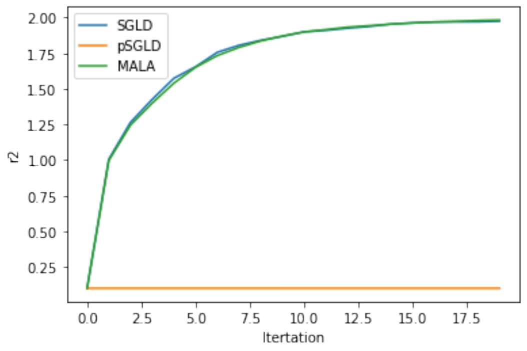
	SGLD	pSGLD	MALA
Time(Sec)	232.26	230.80	280.06

Table 1: Execution Time of different algorithms.

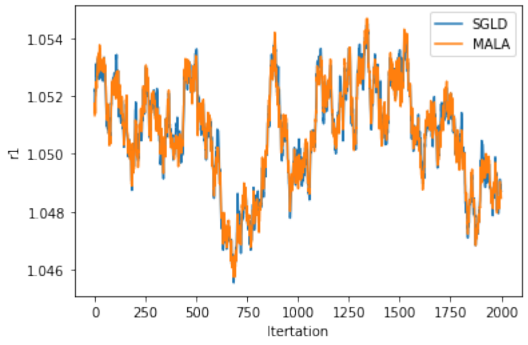
Secondly, the points sampled along the training process are visualized in Figure 2. We can see SGLD and MALA can quickly move to the high-probability region $r_1 = 1$



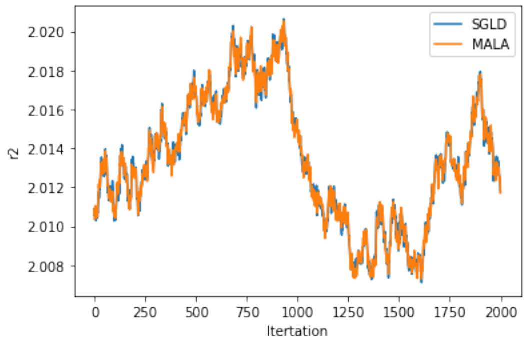
(a) r1 first 20 points



(b) r2 first 20 points



(c) r1 last 2000 points



(d) r2 last 2000 points

Figure 2: Estimated parameters

and $r_2 = 2$ within the first 20 iterations. And in the last 2000iterations, we can see the mixing is not perfect. This corresponds to our analysis in Section 3 that low step size can lead to bad mixing. And we can also see that SGLD moves very close to MALA. This also corresponds to our analysis that SGLD will enter the Langevin dynamics phase when the learning rate is low. One interesting phenomenon is that pSGLD does not work at all. This is also reasonable. pSGLD has a preconditioning maatrix. This can reduce the effective step size in the beginning. So the low effective step size make pSGLD never reach high-probability region.

4.2 Bayesian Neural Network on MNIST Data

I tried to implement SGLD and want to compare it with SGD on a specially designed task: learning a classification model from imbalanced data. But some bugs still exist and I havent got result to present here.

5 Conclusion

In this project, I investigated SGLD and pSGLD for Bayesian learning. The experimental results show the computational efficiency and validate some theoretical results.

References

[1] M. Welling and Y. W. Teh, “Bayesian learning via stochastic gradient langevin dynamics,” in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

[2] C. Li, C. Chen, D. Carlson, and L. Carin, “Preconditioned stochastic gradient langevin dynamics for deep neural networks,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[3] G. O. Roberts and J. S. Rosenthal, “Optimal scaling of discrete approximations to langevin diffusions,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 60, no. 1, pp. 255–268, 1998.

[4] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

[5] G. Hinton, N. Srivastava, and K. Swersky, “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent,” *Cited on*, vol. 14, no. 8, p. 2, 2012.

[6] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.,” *Journal of machine learning research*, vol. 12, no. 7, 2011.

[7] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[8] N. Sandrić, “Stability of the overdamped langevin equation in double-well potential,” *Journal of Mathematical Analysis and Applications*, vol. 467, no. 1, pp. 734–750, 2018.

[9] A. Durmus and E. Moulines, “High-dimensional bayesian inference via the unadjusted langevin algorithm,” *Bernoulli*, vol. 25, no. 4A, pp. 2854–2882, 2019.