Report - Bayesian Learning via Stochastic Gradient Langevin Dynamics

Student: Zhengyang LAN
Professor: Rémi Bardenet

Abstract

The stochastic gradient algorithm is a classic algorithm in optimization, especially the mini-batch stochastic gradient, which can improve the efficiency of each iteration and also makes it applicable to online learning. However, the application of stochastic gradient method for MAP can just find the maximum of posterior, which can easily cause overfitting. This paper combines Stochastic Gradient and Langevin Dynamics, and uses Markov chain Monte Carlo to output samples that approximately follow the true full distribution of posterior. This report also show the result of apply the algorithm in some real data and some creativity and insightful addition.

1 Introduction

Stochastic gradient algorithm and Bayesian learning have been widely used in the optimization of machine learning.

Stochastic gradient algorithm can produce the approximation of parameters of the given data, and Bayesian learning is better at the uncertainty of data distribution.

This paper combines the stochastic gradient method with Langevin Dynamics, so it can not only learn the uncertainty, but also sample from the full distribution of the data, instead of just maximizing the posterior distribution based on the given data, by using the Markov chain Monte Carlo method.

Another advantage of this method is that as the number of iterations increases, it will automatically transition from stochastic gradient phase to Langevin dynamics phase. This article proposes a sampling threshold to detect whether the transition has occurred.

This report will briefly introduce the main theories and computation in the paper in the Section 2. In the Section 3, some applications of Stochastic Gradient and Langevin Dynamics algorithms in real data will be shown. The Section 4 will be some creativity and thinking based on this algorithm and Section 5 concludes.

2 Stochastic Gradient Langevin Dynamics

2.1 Stochastic Gradient and Langevin dynamics

To introduce Stochastic Gradient Langevin Dynamics, let us briefly review Stochastic Gradient and Langevin Dynamics.

Given a model which parameterized by θ. And choose the prior of θ as $p(\theta)$. According to Bayesian formula, the posterior: $p(\theta|X) \propto p(\theta)\prod_{i=1}^{N} p(x_i|\theta)$, where $X = \{x_i\}_{i=1}^{N}$ is data in the train set. The purpose of stochastic gradient is to find the parameter θ that maximizes the posterior. It's also called maximum a posteriori(MAP).

When the training set is too large, in order to reduce the computational complexity of each iteration. We use the n mini-batch stochastic gradient and the parameters will be updated by:

$$\Delta\theta_t = \frac{\epsilon_t}{2}(\nabla log(p(\theta_t))) + \frac{N}{n}\sum_{i=1}^{n}\nabla log(p(x_{ti}|\theta_t)) \tag{1}$$

where ϵ_t is called step sizes in the paper, but I prefer to call it learning rate.

There is one thing to remind. In non-Bayesian learning, we usually use stochastic gradient descent to minimize the loss function. But under our setting, we need to maximize the posterior function. So it will be stochastic gradient ascent and the update method is $\theta_{t+1} = \theta_t + \Delta\theta_t$.

The advantage of this algorithm is that even for very large training sets, the efficiency of this algorithm is still very high. But for each mini batch, the $\Delta\theta_t$ will contain some noise because it only uses part of the training set and does not reflect the gradient direction of the whole training set. But after enough iterations, these noises will be averaged out. Finally, the θ_t will converge towards the maximizing of posterior with respect of the whole training set. This is the disadvantage of this algorithm , because the MAP can't represent the true distribution of the posterior.
For example, there are some points with higher probability in the uniform distribution. If we sample from this point, with some prior like Gaussian or Laplace, the samples won't respect to the true posterior. It's why Stochastic Gradient can't learn the uncertainty of θ and may cause overfitting.

In order to ensure convergence, some restrictions must be added to the learning rate:

$$\prod_{i=1}^{n}\epsilon_t = \infty \qquad \prod_{i=1}^{n}\epsilon_t^2 < \infty \tag{2}$$

Typically, the learning rates $\epsilon_t = a(b+t)^{-\gamma}$ and $\gamma \in (0.5, 1]$.

For the Langevin dynamics, it will inject a Gaussian noise into the $\nabla\theta_t$ and sample at each iteration to get the uncertainty of θ. But it can only be used in the full stochastic gradient algorithm before the publication of this paper. $\Delta\theta_t$ will be written as:

$$\Delta\theta_t = \frac{\epsilon_t}{2}(\nabla log(p(\theta_t))) + \sum_{i=1}^{N}\nabla log(p(x_{ti}|\theta_t)) + \eta_t \qquad \eta_t \sim N(0, \epsilon_t) \tag{3}$$

As we all know, the disadvantage of this method is that whole training sets need to be considered in each iteration, which is very inefficient when the training set is too large.

Generally, this will cause some discretization errors, and we need to use the Metropolis-Hastings algorithm to decide whether to reject at each iteration. On the other hand, fortunately, ϵ_t will decrease as the number of iterations increases, and the rejection rate will approach zero. But ϵ_t keeps decreasing, the algorithm will slow down to almost stagnant. To solve this problem, we can fix ϵ_t when the rejection rate drops to almost negligible, then we can ignore the Metropolis-Hastings step.

2.2 Combination of Stochastic Gradient and Langevin dynamics

In order to design an algorithm which can process large training sets efficiently and learn the uncertainty and sample from the full distribution of the posterior, we merge Stochastic Gradient and Langevin dynamics, the new algorithm called Stochastic Gradient Langevin dynamics:

$$\Delta\theta_t = \frac{\epsilon_t}{2}(\nabla log(p(\theta_t))) + \frac{N}{n}\sum_{i=1}^{n}\nabla log(p(x_{ti}|\theta_t)) + \eta_t \qquad \eta_t \sim N(0, \epsilon_t) \tag{4}$$

It has been proven that samples from θ_t will converge to the posterior distribution in the paper. The stochastic gradient ascent will dominate at beginning then it will transition into Langevin dynamics phase smoothly.

There are two things we need to pay attention to.
One is about discretization errors. We need to do the same thing as in Langevin dynamics, that is to fix the learning rate when Metropolis-Hastings rejection can be ignored.
The second is that we can sample only after entering Langevin dynamics phase, because there has no meaning to sample under the stochastic gradient which is not a MCMC method. So we need to find the condition which indicate the Langevin dynamics phase is already dominant.

Let us first generalize this algorithm to a preconditioning version:

$$\Delta\theta_t = \frac{\epsilon_t}{2}M(\nabla log(p(\theta_t))) + \frac{N}{n}\sum_{i=1}^{n}\nabla log(p(x_{ti}|\theta_t)) + \eta_t \qquad \eta_t \sim N(0, M\epsilon_t) \tag{5}$$

where M is the symmetric preconditioning matrix used for transform all dimensions to the same scale to speed up the converge of the algorithm.

There are two ways to confirm whether the Langevin dynamics phase has been entered. One is to use a threshold of ϵ_t, and the other is to directly evaluate the level of the noise in the Stochastic gradient and Langevin dynamics to determine which one is dominant.

To compare the level of noise in the Stochastic gradient and Langevin dynamics to know whether it is a stochastic gradient or a Langevin dynamics phase dominate in a certain iteration, we can compare the variance of noise of the Stochastic gradient and the Langevin dynamics, which are:

$$\text{Stochastic Gradient Noise : } (\frac{\epsilon}{2})^2 V(\theta_t), \text{ Langevin Dynamics Noise : } \epsilon_t \tag{6}$$

The $V(\theta_t)$ can be estimated by:

$$V(\theta_t) \approx \frac{N^2}{n^2}\sum_{i=1}^{n}(s_{ti} - \bar{s}_t)(s_{ti} - \bar{s}_t)^T \tag{7}$$

where $s_{ti} = \frac{1}{N}\nabla log(p(\theta_t)) + \nabla log(p(x_{ti}|\theta_t))$ and $\bar{s}_t = \frac{1}{n}\sum_{i=1}^{n} s_{ti}$.

We can simply evaluate the variance of the two noises in each iteration. When the variance of the noise of Langevin dynamics is greater than the variance of the noise of Stochastic gradient, we can consider that Langevin dynamics is dominant.

Through (6) and (7), we can know that the variance of the stochastic gradient noise is $\frac{\epsilon^2 N^2}{4n} M V_s M$.
The paper also proposes an empirical condition that can indicate Langevin dynamics is dominant:

$$\frac{\epsilon^2 N^2}{4n}\lambda_{max}(M^{\frac{1}{2}} V_s M^{\frac{1}{2}}) = \alpha \ll 1 \tag{8}$$

where $\lambda_{max}(A)$ is the largest eigenvalue of A. It can only be used empirically, because we can't determine how much is much smaller. Let us associate it with ϵ_t to get a threshold.

The key point is to use Fisher information and the condition in (8), the process to get the threshold is as follows:

$$I_F \approx N V_s \Rightarrow \Sigma_\theta \approx I_F^{-1} \Rightarrow \epsilon_t \approx \frac{4\alpha n}{N}\lambda_{min}(\Sigma_\theta) \tag{9}$$

In my opinion, compared to directly comparing the noise level, this method intuitively gives a threshold to use, but too much approximation is used in the derivation process, which may reduce the reliability of this threshold.

Once the algorithm is in Langevin dynamics phase, we can start sampling to get samples that follow the full posterior distribution.

3 Application on real data

In this section, I will show my implementation this algorithm on real data. I implemented it on the 1D/2D normal distribution estimation similar to the article and linear regression. All the implementation can be find in https://github.com/LANZhengyang/Bayesian-Learning-via-SGLD.

3.1 1D and 2D normal distribution

At first, I demonstrate application of stochastic gradient Langevin algorithm for estimating 1D distribution. For simplify the code, the data is:

$$\theta \sim N(0,2), \qquad x_i \sim N(\theta, 2) \tag{10}$$

The train data size is N=5000. For the model, a=0.001, b=1, gamma=0.55 and batch size is 100. After 300 iterations, Langevin dynamics phase dominated, and then we fix the learning rate and start sampling up to 1000 iterations (700 samples in total). The result is in the figure 1 and figure 2. We can see that the distribution of sample is approaching the true posterior distribution.

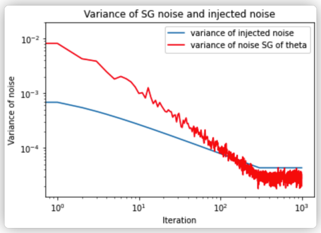


Figure 1: Variance of SG noise and injected noise

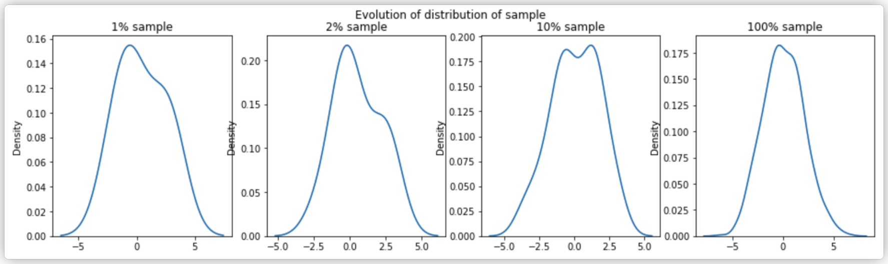


Figure 2: Evolution of distribution of sample

For the implementation of 2D normal distribution, the description of this implementation in the article is not very clear. For example, in the expression of x_i, it's 1D but in the figure 1 it's 2D. And there is no parameter value in $\epsilon_t = a(b+t)^{-\gamma}$, no indication of when to sample. So my research is not exactly the same as it, but it can also reflect the advantages of this algorithm. The data distribution is:

$$\theta_1 \sim N(4,1), \qquad \theta_2 \sim N(4,1), \qquad (x_i, y_i) \sim (N(\theta_1, 1), N(\theta_2, 1)) \tag{11}$$

The train data size is N=1000. For the model, a=0.0001, b=1, gamma=0.55 and batch size is 100. After 100 iterations, Langevin dynamics phase dominated, and then we start sampling up to 20000 iterations. Since in the figure of paper it does not fixed learning rate, so I just do mine like paper. In the figure 3 and 4, we can see that it also got very good results since the iteration is large enough.

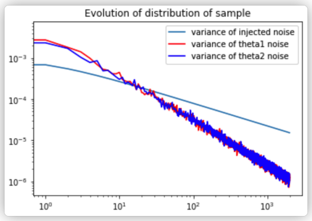
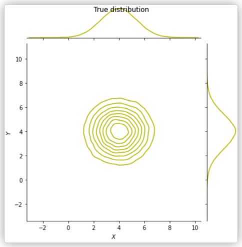
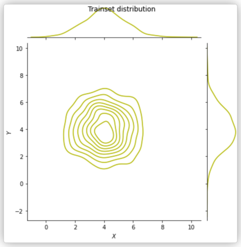


Figure 3: Variance of SG noise and injected noise

[a] 

[b] 

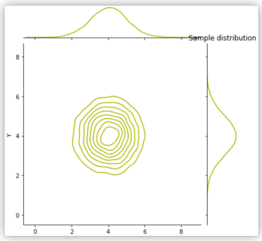
[c] 

Figure 4: (a) True distribution (b) Train distribution (c) Sample distribution

We can see that the sample data is follow the true distribution (not overfitting on trainset) and approach true distribution.

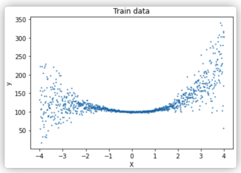
3.2 Linear regression

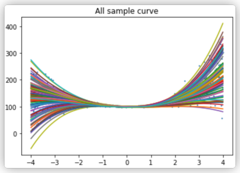
Generally, linear regression can easily cause overfitting. Through Stochastic Gradient Langevin Dynamics, the model can learn the uncertainty of the training set to show the true distribution better.

In the figure 5(a) this the distribution of data, the sampling process is:

$$\begin{aligned} &a \sim N(100,1), \qquad b \sim N(-2,1), \qquad c \sim N(5,1), \qquad d \sim N(1,1) \\ &y_i = a + bx_i + cx_i^2 + dx_i^3 \qquad x \in [-4,4] \end{aligned} \tag{12}$$

The figure 5(b), show the sample of during last 100 iteration. And figure 5(c) show the mean of sample curve with 1 and 2 standard deviation lines. It show the distribution of data well.

[a] 

[b] 

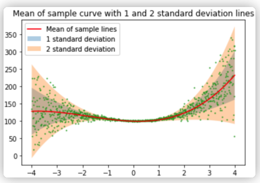
[c] 

Figure 5: (a) Train data distribution (b) All sample curve (c) Mean of sample curve with 1 and 2 standard deviation lines

4 Creativity and insightful addition

4.1 Meaningful thinking

In the process of implementing this algorithm, I found two shortcoming that the author did not specifically mention.

First, in order to make Langevin dynamics phase dominate we need to compare the variance of

Stochastic Gradient noise and the variance of injected noise. But in fact, $Vs \propto \frac{N^2}{n}$. So if we use a too large training set, although we can benefit from Stochastic Gradient with mini batch to reduce the amount of computation in each iteration, if N is too large or the batch size is too small, it will require more iterations make ϵ_t decay more but the algorithm will be slower with too small ϵ_t and more iteration will cost more. For example, in the Figure 2 of the article, Langevin dynamics phase dominate after 10^5 iteration. And if the learning rate is decayed too much, the update of parameters will be very small which can be almost ignored. So for a very large data set, we have to choose a large batch size to ensure the availability of the algorithm and get less benefit from Stochastic Gradient. Second, whether we choose to compare the variance of noise or the threshold, we need to compute Vs. We need to compute additional $batchsized \times \theta$ times gradients in each iteration. This will make the efficiency of the algorithm drop quickly, especially when iterating number is large and the dimension of parameters are large.

In addition, this article mentioned the use of preconditioning matrix M in section 4, but did not explain the details. But there is a paper[1] (published after this paper) proposes and explains it well.

4.2 Implementation optimization algorithm with other MCMC method

At the end of this article, the author mentioned that other MCMC algorithms can be used to solve the random walk problem, such as Hamiltonian Monte Carlo[2] is published after this paper. For other MCMC method, Stochastic Gradient Fisher Scoring[3] and Stochastic Gradient Nosé-Hoover Thermostat[4] are published a few years after this paper.

Stochastic Gradient Langevin Dynamics is based on Stochastic Gradient algorithm, but this paper was published about 10 years ago. The most popular optimization algorithm in deep learning today is the Adam algorithm, which has not been proposed when this article was published. After reading this article, I am thinking about whether it is possible to combine Adam with MCMC to achieve better performance of Bayesian learning. After some searches, it was found that this is feasible and has been used perfectly[5].

5 Conclusion

By using the Stochastic Gradient Langevin Dynamics optimization method, we can simultaneously obtain both the advantage of low cost of each iteration of the mini batch stochastic gradient and learning uncertainty in Bayesian learning and get the sample from the true posterior distribution. It is the main contribution of this article.

However, due to the limitations of the times and other factors, it didn't use better performance optimization algorithms and try other MCMC algorithms. But this article provides us with an idea of sampling from the true posterior distribution which is a combination of optimization algorithms and MCMC methods. Many articles published after this article which tried different combinations and achieved good performance.
For researcher in Bayesian learning, even if there are no new advances in the MCMC method, new advances in optimization may improve existing methods as well. This inspired me to let me know that it is important to keep up with the cutting edge in research.

References

[1] Li, Chunyuan, et al. "Preconditioned stochastic gradient Langevin dynamics for deep neural networks." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30. No. 1. 2016.

[2] Chen, T., Fox, E., Guestrin, C. (2014, January). Stochastic gradient hamiltonian monte carlo. In International conference on machine learning (pp. 1683-1691).

[3] Ahn, Sungjin, Anoop Korattikara, and Max Welling. "Bayesian posterior sampling via stochastic gradient Fisher scoring." arXiv preprint arXiv:1206.6380 (2012).

[4] Ding, Nan, et al. "Bayesian sampling using stochastic gradient thermostats." Advances in neural information processing systems 27 (2014): 3203-3211.

[5] Moss, Adam. "Accelerated Bayesian inference using deep learning." Monthly Notices of the Royal Astronomical Society 496.1 (2020): 328-338.

[6] Max Welling and Yee Whye Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. In Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11). Omnipress, Madison, WI, USA, 681–688.