2009 International Symposium on Nonlinear Theory and its Applications
NOLTA'09, Sapporo, Japan, October 18-21, 2009

Generalization Error by Langevin Equation in Singular Learning Machines

Taruhi Iwagaki† and Sumio Watanabe‡

†Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology
4259 Nagatsuda, Midori-ku, Yokohama, 226-8503 Japan
‡Precision and Intelligence Laboratory, Tokyo Institute of Technology
4259 Nagatsuda, Midori-ku, Yokohama, 226-8503 Japan
Email: iwagaki@cs.pi.titech.ac.jp, swatanab@pi.titech.ac.jp

Abstract—The Langevin equation implies an algorithm that can generate samples from the stationary distribution of a biased random walk, which is equivalent to the posterior distribution of Bayesian learning. The Langevin algorithm uses gradient information of the target distribution; therefore it is expected to be more efficient than the Metropolis method especially in wide parameter space of singular learning machines e.g. neural networks. In this paper, we will discuss experimental results of generalization errors of both the Langevin algorithm and the Metropolis method for neural networks with practical dimension.

1. Introduction

The Langevin equation is a known stochastic differential equation as a mathematical model of Brownian motion. Its solution satisfies the Fokker-Planck equation as a probability density function, therefore random walking under the Langevin equation generates same samples from the stationary distribution of the Fokker-Planck equation. By adjustment of a potential term in the Langevin equation, the sampling algorithm from the Bayesian posterior distribution can be constructed and it indicates a steepest descent method with a stochastic term.

The big problem in Bayesian learning is computing the predictive distribution that contains high-dimensional integral, which can seldom be performed exactly and requires some approximations. Furthermore, The non-identifiable and non-regular models, e.g. neural networks and hidden Markov model, have a analytic set of parameters with singularities because they have hidden variables or hierarchical structures, which affects precision of sampling algorithm and estimation of generalization errors [3] [7].

Metropolis method, a basic sampling algorithm, causes slow convergence in the high-dimensional parameter space because a proposal candidate depends on only randomness. On the other hand, the Langevin algorithm uses gradient information of the target distribution in each iteration step, hence it is expected to be more efficient than the Metropolis method especially in wide parameter space of singular learning machines, but the problem of discretization of continuous-time stochastic process is left [2] [1].

This paper aims to compare the behavior of the generalization errors approximated by the Langevin algorithm and the Metropolis method for neural networks with practical dimension, especially on the view of iteration times and computational costs.

2. Bayesian Learning and Sampling Algorithms

2.1. Bayesian Posterior Distribution

Let $X^n = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ be a set of n-sample data, which are independently and identically generated by the true distribution $q(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})$. Assume that each $\boldsymbol{x}_i$ is in $\boldsymbol{R}^N$ and $\boldsymbol{y}_i$ is in $\boldsymbol{R}^M$.

Let $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ be a learning machine, and $p(\boldsymbol{w})$ a prior distribution on the set of parameters.

Then the Bayesian posterior distribution is defined by

$$p(\boldsymbol{w}|X^n) = \frac{1}{Z(X^n)} p(\boldsymbol{w}) \prod_{i=1}^{n} p(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{w}), \tag{1}$$

where

$$Z(X^n) = \int p(\boldsymbol{w}) \prod_{i=1}^{n} p(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{w}) dw. \tag{2}$$

The Bayesian predictive distribution is also given by

$$p(\boldsymbol{y}|\boldsymbol{x}, X^n) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w}|X^n) dw. \tag{3}$$

We need samples generated by $p(\boldsymbol{w}|X^n)$ to approximate $p(\boldsymbol{y}|\boldsymbol{x}, X^n)$.

2.2. Sampling Algorithm using Langevin Equation

Let $W_t \in \boldsymbol{R}^d$ be a sequence of random variables on a continuous-time stochastic process and $R_t \in \boldsymbol{R}^d$ a random variable of the Gaussian distribution $N(0, 2DtI)$ (D is a constant value). The following stochastic differential equation is known as Langevin equation.

$$\frac{dW_t}{dt} = -\nabla V(W_t) + \frac{dR_t}{dt}. \tag{4}$$

The equation leads the following difference method for computational simulation with a small $\alpha_{lan} \in \boldsymbol{R}$. Assume that R_k is independently and identically generated by $N(0, I)$.

$$W_{k+1} = W_k - \alpha_{lan}\nabla V(W_k) + \sqrt{2D\alpha_{lan}}R_k. \quad (5)$$

Let $p(\boldsymbol{w}, t)$ be a probability density function of the random variable W_t. It satisfies the Fokker-Planck equation below:

$$\frac{\partial}{\partial t}p(\boldsymbol{w}, t) - \nabla \cdot (\nabla V(\boldsymbol{w})p(\boldsymbol{w}, t)) = D\triangle p(\boldsymbol{w}, t). \quad (6)$$

Assume that the limiting distribution $q(w) = q(w, t)$ exists in $t \to \infty$ and $q(\boldsymbol{w}, t) = 0$ in $\|\boldsymbol{w}\| \to \infty$, then we get

$$p(w) \propto \exp\left(-\frac{V(\boldsymbol{w})}{D}\right). \quad (7)$$

This equation implies that $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k\}$ are from the limiting distribution,

$$\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \ldots, \boldsymbol{w}_k\} \sim \exp\left(-\frac{V(\boldsymbol{w})}{D}\right). \quad (8)$$

Let $D = 1$, $V(\boldsymbol{w}) = nL(\boldsymbol{w})$, and

$$L(\boldsymbol{w}) = -\frac{1}{n}\log p(X^n|\boldsymbol{w}) - \frac{1}{n}\log p(\boldsymbol{w}). \quad (9)$$

In this case, the limiting distribution is equivalent to the Bayesian posterior distribution. Therefore, the Langevin algorithm works to generate samples from the Bayesian posterior distribution:

$$\exp\left(-\frac{V(\boldsymbol{w})}{D}\right) = \exp\left(\log p(X^n|\boldsymbol{w}) + \log p(\boldsymbol{w})\right) \quad (10)$$

$$= p(X^n|\boldsymbol{w})p(\boldsymbol{w}) \propto p(\boldsymbol{w}|X^n). \quad (11)$$

Summary of the Langevin algorithm is as follows:

1. Initialize $\boldsymbol{w}_0$.
2. Calculate $\nabla V(\boldsymbol{w}_k)$ with the current $\boldsymbol{w}_k$.
3. Set $\boldsymbol{w}_{k+1}$ with a random value R from $N(0, I)$:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_{lan}\nabla V(\boldsymbol{w}_k) + \sqrt{2D\alpha_{lan}}R. \quad (12)$$

4. Go to (2).

2.3. Generalization Error

Let $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots \boldsymbol{w}_k\}$ be samples generated by the Bayesian posterior distribution $p(\boldsymbol{w}|X^n)$. Then the Bayesian conditional predictive distribution $p(\boldsymbol{y}|\boldsymbol{x}, X^n)$ can be approximated as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}, X^n) = \int_W p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})p(\boldsymbol{w}|X^n)dw \quad (13)$$

$$\simeq \frac{1}{k}\sum_{i=1}^{k} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}_i). \quad (14)$$

The generalization error between $p(\boldsymbol{y}|\boldsymbol{x}, X^n)$ and the true distribution $q(\boldsymbol{y}|\boldsymbol{x})$ can be approximated with test data $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}$ as follows:

$$G(X^n) = \int q(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})\log\frac{q(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{x}, X^n)}dxdy \quad (15)$$

$$\simeq \frac{1}{m}\sum_{i=1}^{m}\log\frac{q(\boldsymbol{y}_i|\boldsymbol{x}_i)}{p(\boldsymbol{y}_i|\boldsymbol{x}_i, X^n)}. \quad (16)$$

2.4. Metropolis Method

The Metropolis method generates samples $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k\}$ that converge in distribution to the target distribution

$$p(\boldsymbol{w}|X^n) = \frac{1}{Z}\exp\left(-nH(\boldsymbol{w})\right), \quad (17)$$

and its step-by-step instructions are the following:

1. Initialize $\boldsymbol{w}_0$.
2. Get the proposal sample $\boldsymbol{w}'$ by adding random value from $N(0, \sigma^2_{met}I)$ to the current sample $\boldsymbol{w}_k$.
3. Choose the new sample $\boldsymbol{w}_{k+1}$ according to the following rules:
   (a) In the case that $H(\boldsymbol{w}_k) > H(\boldsymbol{w}')$, let $\boldsymbol{w}_{k+1} = \boldsymbol{w}'$
   (b) In the case that $H(\boldsymbol{w}_k) \leq H(\boldsymbol{w}')$,
      i. let $\boldsymbol{w}_{k+1} = \boldsymbol{w}'$ with probability P
      ii. let $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k$ with probability $1 - P$

   where

$$P = \exp\left(nH(\boldsymbol{w}_k) - nH(\boldsymbol{w}')\right). \quad (18)$$

4. Go to (2).

3. Evaluation

3.1. Model and Setting

We use a 3-layer neural network as a learning machine

$$f(\boldsymbol{x}, \boldsymbol{w}) = \tanh(B\tanh(A\boldsymbol{x})), \quad (19)$$

where $\boldsymbol{x} \in \mathbf{R}^N, \boldsymbol{y} \in \mathbf{R}^M, A \in M(N, H; \mathbf{R}), B \in M(H, M; \mathbf{R})$, $\boldsymbol{w} = (A, B)$. There are N input units, M output units, and H hidden units, with tanh as an activation function.

The true model is another 3-layer neural network $g(\boldsymbol{x})$ with the same condition except for the number of hidden unit, $L(< H)$.

We consider that the set of input data $\{\boldsymbol{x}_i\}$ are generated by the normal distribution $N(0, I)$ and the set of observable data $\{\boldsymbol{y}_i\}$ contains Gaussian noise from $N(0, \sigma^2_{dat}I)$. Then the distribution of $\boldsymbol{x}$, the true model $q(\boldsymbol{y}|\boldsymbol{x})$, and the learning machine $p(\boldsymbol{y}|\boldsymbol{x})$ are characterized by the distribution density functions below:

$$q(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}^N}\exp\left(-\frac{1}{2}\|\boldsymbol{x}\|^2\right), \quad (20)$$

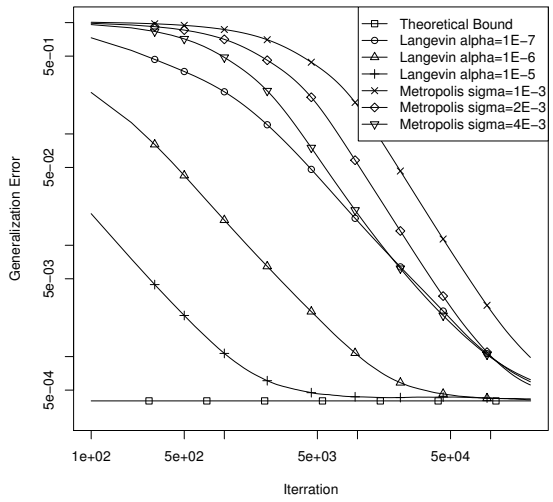

Figure 1: Generalization Errors by Iteration

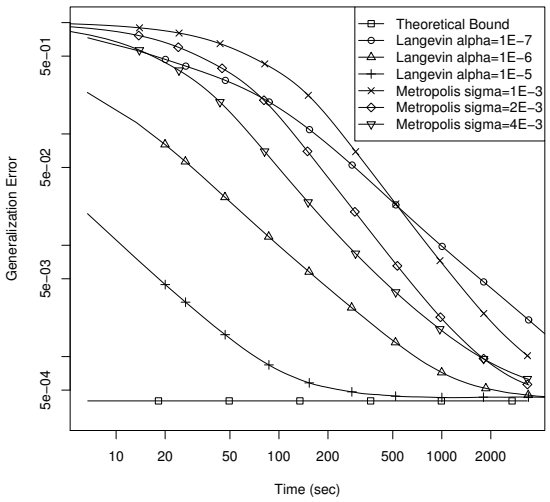

Figure 2: Generalization Errors by Execution Time

$$q(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{\sqrt{2\pi\sigma_{dat}^2}^M} \exp\left(-\frac{1}{2\sigma_{dat}^2}\|\boldsymbol{y} - g(\boldsymbol{x})\|^2\right), \quad (21)$$

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{\sqrt{2\pi\sigma_{dat}^2}^M} \exp\left(-\frac{1}{2\sigma_{dat}^2}\|\boldsymbol{y} - f(\boldsymbol{x}, \boldsymbol{w})\|^2\right). \quad (22)$$

The learning data $X^n = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ and the testing data are generated by $q(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})$ and $q(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})$ respectively.

Let $p(\boldsymbol{w})$ be $N(0, \sigma_{pri}^2 I)$ as the prior distribution. On the setting above, $V(\boldsymbol{w})$ is given by

$$V(\boldsymbol{w}) = \frac{1}{2\sigma_{dat}^2}\sum_{i=1}^{n}\|\boldsymbol{y}_i - f(\boldsymbol{x}_i, \boldsymbol{w})\|^2 + \frac{1}{2\sigma_{pri}^2}\|\boldsymbol{w}\|^2 + const. \quad (23)$$

Then, the gradient $\nabla V(\boldsymbol{w})$ for Langevin algorithm is given by

$$\frac{\partial}{\partial \boldsymbol{w}}V(\boldsymbol{w}) = \frac{1}{\sigma_{dat}^2}\sum_{i=1}^{n}\frac{\partial}{\partial \boldsymbol{w}}\left(\frac{1}{2}\|\boldsymbol{y}_i - f(\boldsymbol{x}_i, \boldsymbol{w})\|^2\right) + \frac{1}{\sigma_{pri}^2}\boldsymbol{w}, \quad (24)$$

which can be calculated as same as the back propagation method on neural networks.

The generalization error $G(X^n)$ also can be approximated on asymptotic analysis:

$$G(X^n) \simeq \frac{1}{2m}\sum_{i=1}^{m}\left(g(\boldsymbol{x}_i) - \frac{1}{k}\sum_{j=1}^{k}f(\boldsymbol{x}_i, \boldsymbol{w}_j)\right)^2. \quad (25)$$

Now we fixed the variance of the data noise $\sigma_{dat}^2 = 0.01$, and the variance of the prior distribution $\sigma_{pri}^2 = 10^6$, the number of learning data $n = 1000$, the number of testing data $m = 10000$. The parameters of the true model and the initial parameters of the learning machine are generated by the uniform distribution [0, 1.0].

The exact theoretical generalization error of neural networks is still unknown but the upper bound is given in [4]. To check the experimental results, we will refer the upper bound in graphs:

$$E_{X^n}[G(X^n)] \leq \sigma_{dat}^2\frac{2NH - (H - L)N}{2n}. \quad (26)$$

3.2. Experimental Results

3.2.1. Generalization Error by Iteration

We evaluated the generalization error by the Langevin algorithm and the Metropolis method. In a log-log plot Fig. 1, the vertical axis is $E_{X^n}[G(X^n)]$ and the horizontal axis is iteration times. We tried to calculate the generalization error with several parameters $\alpha_{lan} = 10^{-7}, 10^{-6}, 10^{-5}$ and $\sigma_{met} = 1.0 \times 10^{-3}, 2.0 \times 10^{-3}, 4.0 \times 10^{-3}$, which are the discretization coefficient and the variance of random value to be adjusted for each algorithm. The dimension of each layer was fixed as $N = M = 10$, $H = 5$, $L = 3$, and samples are chosen per 10 iterations in 2.0×10^5 iterations. Each average of acceptance ratio of the Metropolis method was 50.12% ($\sigma_{met} = 1.0 \times 10^{-3}$), 18.83% ($2.0 \times 10^{-3}$), 2.03% ($4.0 \times 10^{-3}$). The final generalization error of the Langevin algorithm with $\alpha_{lan} = 10^{-6}, 10^{-5}$ were $E_{X^n}[G(X^n)] = 4.09\times 10^{-4}, 4.17\times 10^{-4}$, and they were good approximations to the theoretical bound $G(X^n) \leq 4.000 \times 10^{-4}$. On the comparison between the generalization errors with the most appropriate parameters for each algorithm, the convergence of the Langevin algorithm was faster than one of Metropolis method.

	Execution Time (msec)
Langevin	66.84
Metropolis	17.31

Table 1: Execution Time per Iteration

3.2.2. Generalization Error by Execution Time

Fig. 2 is a re-scaled graph of Fig. 1 by the average of execution time as the horizontal axis.

The execution times by iteration step of each algorithm are different. Table 1 describes the average of execution time on our computer system, and the Langevin algorithm is 3.86 times slower than the Metropolis method. After scaling, the generalization error by the Langevin algorithm with $\alpha_{lan} = 10^{-6}, 10^{-5}$ still converges faster than one by the Metropolis method with $\sigma_{met} = 1.0 \times 10^{-3}, 2.0 \times 10^{-3}$.

4. Discussion

First, we discuss the convergence speed in initial transient phase. Fig. 1 and 2 describe that the convergence of the Langevin algorithm in the initial phase is faster than one of the Metropolis method. The generalization error by the Langevin algorithm declines immediately at the beginning of iterations but the generalization error by the Metropolis method is slow in converging until around 5×10^2 iteration. In the Metropolis method, a proposal sample is generated with a random value and a next sample is chosen probabilistically by difference of energy. In the case that $\sigma_{met} = 2.0 \times 10^{-3}$, the Metropolis algorithm approximates the generalization error most accurately, but the acceptance ratio is 18.83%. We use *tanh* as an activation function in this model, therefore the difference of energy of parameters cannot be observed when parameters are far away from the set of true parameters. This fact and high-dimension of parameter space seem to affect gain of rejection. On the other hand, the Langevin algorithm uses the gradient of the target distribution at the current sample, and therefore the Langevin algorithm seems to move efficiently using the gradient on the same condition. However, this graph describes only efficiency of initial transient phase, so we need to study efficiency of coverage of the target distribution after burn-in phase. Furthermore, the Langevin algorithm contains the problem of discretization of continuous-time stochastic process. Some discretization coefficient α_{lan} cannot reproduce the continuous-time stationary distribution [2] [1]. The Langevin Metropolis-Hasting method was also proposed [5][6].

Second, we discuss execution costs. The Metropolis method needs at least one forward calculation of neural network to compare energy of a proposal sample with one of a current sample. On the other hand, the Langevin algorithm does not need energy but needs gradient. We use the back propagation method to calculate the gradient in this experiment, and it requires both forward and backward calculation of neural network. The backward calculation of Langevin algorithm takes additional costs than Metropolis method, which affects execution time per iteration. Depending on the additional backward calculation costs, Langevin algorithm may lose its advantage of speed by efficient moving, especially in the other model which has high costs to calculate its gradient.

5. Conclusion

In this paper, we have evaluated the generalization errors with generated samples by the Langevin algorithm and the Metropolis method in a neural network with practical dimension, and compared their behavior in initial transient phase. Our future studies are the relation between discretization parameter and its effect for the target distribution of singular learning machines.

This research was partially supported by the Ministry of Education, Science, Sports and Culture in Japan, Grant-in-Aid for Scientific Research 18079007.

References

[1] G. O. Roberts and R. L. Tweedie, "Exponential Convergence of Langevin Diffusions and Their Discrete Approximations," 1995.

[2] G. O. Radford M. Neal, "Probabilistic Inference using Markov Chain Monte Carlo Methods," Technical report, University of Toronto, 1993.

[3] S. Watanabe, "Algebraic Analysis for Nonidentifiable Learning Machines," *Neural Computation*, vol.13(4), pp.899–933, 2001.

[4] S. Watanabe, "Learning Efficiency of Redundant Neural Networks in Bayesian Estimation," *IEEE Transactions on Neural Networks*, vol.12 (6), pp.1475–1486, 2001

[5] U. Grenander and M. Miller, "Representations of Knowledge in Complex Systems," *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol.56 (4), pp.549–603, 1994.

[6] D. Phillips and A.Smith, "Bayesian model comparison via jump diffusions," *Markov Chain Monte Carlo in Practice*, pp.215–239, 1996.

[7] M. Aoyagi, S. Watanabe, "Resolution of Singularities and the Generalization Error with Bayesian Estimation for Layered Neural Network," *IEICE Transactions on Information and Systems*, vol.J88-D-II (10), pp. 2112-2124, 2005.