Bayesian Learning via Stochastic Gradient Langevin Dynamics

December 2025

1 Introduction

In recent years, the field of machine learning has become increasingly interested in handling large datasets. We are therefore motivated to find fast and memory-efficient optimization methods. In addition, as more complex models become popular, some research has shifted towards Bayesian methods or probabilistic machine learning. Assessing uncertainty in learned parameters is believed to avoid overfitting and help quantify model trustworthiness. However, Bayesian methods are still less common in large-scale machine learning.

Welling and Teh's 2011 paper *Bayesian Learning via Stochastic Gradient Langevin Dynamics* combines ideas from stochastic optimization with Bayesian Monte Carlo methods. By injecting specified noise into stochastic gradient updates and annealing the step size, SGLD prevents collapse to the maximum a posteriori solution and transitions to a sampling via a discretization of an SDE whose equilibrium distribution is the true posterior. In this expository paper, I give an overview of Langevin dynamics, a subclass of Hamiltonian dynamics and MCMC. I show the connection between Langevin dynamics and the continuous-time Langevin equation, which is crucial to the importance of posterior sampling via SGLD.

2 Stochastic Optimization

We will begin with a quick overview of the standard approach to optimization in machine learning, **stochastic gradient descent**.

2.1 Gradient Descent

Suppose we wish to minimize a multivariable function $f(x)$. If f is differentiable in a neighborhood of x, then f decreases fastest in the direction of $-\nabla f(x)$. Gradient descent is a first order iterative optimization technique which seeks to find a minimum of $f(x)$ by the update:

$$x_{n+1} := x_n - \alpha \nabla f(x_n),$$

for each time step n and α a hyperparameter called the **learning rate** or **step size**.

Gradient descent is commonly used in machine learning. Suppose we are modeling data (X, Y) by some function $f(\theta)$, where θ is a vector of parameters. In this case, the **cost function**, $J(\theta)$, is minimized by updating θ. Standard gradient descent would iteratively repeat

$$\theta_{j,n+1} := \theta_{j,n} - \alpha \frac{\partial}{\partial \theta_j} J(\theta_{\cdot,n})$$

for each j at time step n until **convergence**.

In modern machine learning, however, datasets are large and have high dimensionality. Calculating the gradient of the entire dataset at each step can be computationally expensive and subject to memory limitations. A popular solution is to use **stochastic optimization**.

2.2 Stochastic Optimization (Robbins-Monro)

In stochastic approximation, we wish to approximate the extrema of a function f by observing random samples. **Robbins-Monro algorithms** are an popular class of such methods.

Theorem 1 (Robbins-Monro)**.** *Suppose there is some constant c such that $f(\theta) = c$ has a unique root at θ^*. Instead of directly observing $f(\theta)$, we have samples of the random variable $g(\theta)$ where $\mathbb{E}(g(\theta)) = f(\theta)$.*

Suppose f is nondecreasing and $f'(\theta^*) > 0$. Then for some sequence of positive step sizes a_n such that

$$\sum_{n=0}^{\infty} a_n = \infty, \sum_{n=0}^{\infty} a_n^2 < \infty,$$

iterates of

$$\theta_{n+1} := \theta_n - a_n(g(\theta_n) - c)$$

converge to θ^* in $\mathcal{L}^2$.

2.3 Stochastic Gradient Descent

In traditional stochastic gradient descent, at each iteration t, a subset of n data items $X_t = \{x_{t_1}, ..., x_{t_n}\}$ is given, and the parameters are updated as follows:

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

where ϵ_t is an appropriately selected sequence of step sizes.

SGD parameter estimates fluctuate around their maximum a posteriori values. Most approaches shrink step sizes to zero to reach a fixed point. The authors argue against doing this, which optimizes beyond the scale of the posterior distribution:

"The posterior represents the intrinsic statistical scale of precision and trying to determine parameter values with more precision runs the risk of overfitting at additional computational cost."

This motivates finding a Bayesian approach to parameter estimation in which one seeks the posterior distribution of the parameters rather than pointwise solutions.

3 Markov Chain Monte Carlo

The authors examine a class of Monte Carlo sampling algorithms called **Langevin dynamics**. Monte Carlo methods are a popular way to generate random samples according to desired distributions. Generally, they:

1. Define a domain of possible inputs
2. Generate inputs randomly from a desired probability distribution over the domain
3. Perform a deterministic computation of the outputs, such as taking the mean to approximate the expected value
4. Aggregate the results

Note this requires the distribution to be something we can actively sample from. **Markov Chain Monte Carlo (MCMC)** methods, in particular, use Markov chains to simulate draws from a probability distribution. This is valid because of the **ergodic theorem** for Markov chains, which roughly states:

Theorem 2 (Ergodic Theorem for Markov Chains)**.** *Let $\mathcal{P}$ be an irreducible probability transition*

rule having a stationary distribution π. Then

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} I\{X_t = j\} = \pi(j)$$

for all states j, regardless of π_0.

So, suppose $X_0, X_1, \ldots$ is an irreducible Markov Chain with a stationary distribution π. Then by the above, for n sufficiently large, i.e., we let the Markov chain run sufficiently "long," then $X_1, X_2..., X_n$ is like a sample from π.

Langevin dynamics can be considered a subclass of **Hamiltonian dynamics**, so we will first build up to a discussion of **Hamiltonian Monte Carlo.**

3.1 Metropolis Proposals

The **Metropolis-Hastings** algorithm is an MCMC method which probabilistically adds draws to the Markov chain.

Roughly, given a state space S and a probability mass function π on S, we want a probability transition matrix P such that $\pi P = \pi$.

Start with matrix Q being a Markov chain probability transition matrix. Q must be irreducible but is otherwise any convenient way of proposing candidates. For $i \neq j$ define

$$\mathbb{P}(i,j) = Q(i,j) \min\left\{1, \frac{\pi(j)Q(i,j)}{\pi(i)Q(i,j)}\right\}.$$

Define $\mathbb{P}(i,i) = 1 - \sum_{j\neq i} \mathbb{P}(i,j)$.

Hence the Metropolis method adds draws based on the ratio of how often state i occurs vs. state j in the target distribution.

3.2 Hamiltonian Dynamics

Hamiltonian Monte Carlo (HMC) simulates draws from a distribution by combining the physics-based model of **Hamiltonian dynamics** with a Metropolis-type proposal.

In two dimensions, we can visualize Hamiltonian dynamics by a frictionless puck sliding over a surface of varying height. The state of the system consists of the position of the puck, given as vector q, and the momentum, represented by vector p.

The **potential energy** $U(q)$ is proportional to the height of the surface at its current position; the **kinetic energy** $K(p)$ is equal to $\frac{|p|^2}{2m}$, where m is the mass of the puck. On a level surface, the puck travels with constant velocity equal to $\frac{p}{m}$.

In non-physical MCMC applications of Hamiltonian dynamics, the position corresponds to the variables of interest (e.g., the parameters of the model).

Hamiltonian dynamics operate on a d dimensional position vector q and a d dimensional momentum vector p; therefore, the full state space has dimension $2d$. The **Hamiltonian** $H(q, p)$ and its relationship to q, p over time t is determined by its partial derivatives:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i,}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

for $i = 1, \ldots, d$.

In HMC, we usually use Hamiltonian functions that can be written

$$H(q, p) = U(q) + K(p).$$

Momentum variables, one for each position variable, will be introduced artificially.

$K(p)$, the kinetic energy, is usually defined as

$$K(p) = \frac{p^T M^{-1} p}{2}.$$

Here, M is a symmetric PSD mass matrix, which is typically diagonal, and is often a scalar multiple of the identity matrix. This form for the kinetic energy corresponds to minus log probability density (plus a constant) of the zero-mean Gaussian distribution with covariance matrix M.

3.2.1 Probabilities and the Hamiltonian

Above, we have viewed Hamiltonian dynamics through the lens of energies. However, in a non-physical application, we must understand how to define energies based on the probability distributions we wish to sample from.

The distribution we wish to sample can be related to a potential energy function via a **canonical distribution**. Given some energy function $E(x)$ for the state x of some physical system, the canonical distribution over states has probability density function:

$$P(x) = \frac{1}{Z} \exp\left(-\frac{E(x)}{T}\right),$$

where T is the temperature of the system and Z is the appropriate normalizing constant such that the $P(x)$ integrates to 1.

Therefore, letting $T = 1$, if we are interested in some distribution with PDF P, we can set $E(x) = -\log P(x) - \log(Z)$.

The Hamiltonian is an energy function for the joint state of position q and momentum p. Therefore,

$$P(q,p) = \frac{1}{Z}\exp\left(-\frac{H(q,p)}{T}\right).$$

If $H(q,p) = U(q) + K(p)$, the joint density is

$$P(q,p) = \frac{1}{Z}\exp\left(-\frac{U(q)}{T}\right)\exp\left(-\frac{K(p)}{T}\right)$$

and q, p are independent, each with canonical distributions determined by energy functions $U(q), K(p)$.

Since we are interested in approximating the posterior distribution for model parameters in Bayesian methods, we will let these parameters be the position q. Let $\pi(q)$ be the prior density and $L(q|D)$ the likelihood function given data D. Express the posterior distribution as a canonical distribution using potential energy function

$$U(q) = -\log[\pi(q)L(q|D)].$$

3.3 Hamiltonian Monte Carlo

Assume the Hamiltonian has form $U(q) + K(p)$, that K has the form $p^T M^{-1} p$, M diagonal with elements $m_1, m_2, \ldots, m_d$ such that

$$K(p) = \sum_{i=1}^{d} \frac{p_i^2}{2m_i}.$$

Let

$$U(q) = -\log[\pi(q)L(q|D)].$$

Define the **leapfrog method** for proposing new values as:

$$p_i\left(t + \frac{\epsilon}{2}\right) = p_i(t) - (\frac{\epsilon}{2})\frac{\partial U}{\partial q_i}(q(t))$$

$$q_i(t+\epsilon) = q_i(t) + \epsilon\frac{p_i\left(t+\frac{\epsilon}{2}\right)}{m_i}$$

$$p_i(t+\epsilon) = p_i\left(t+\frac{\epsilon}{2}\right) - \left(\frac{\epsilon}{2}\right)\frac{\partial U}{\partial q_i}(q(t+\epsilon))$$

This starts with a half step for the momentum variables, then a full step for the position variables that uses the new momentum values. Then, we do another half step for the momentum variables, using the new values for the position variables.

Then we can run the **Hamiltonian Monte Carlo** algorithm as follows:

Each iteration has two steps. The first changes momentum and the second may change both position and momentum. Both leave the canonical joint distribution invariant; therefore, their combination also leaves the distribution invariant.

1. New values for momentum variables are randomly drawn from their Gaussian distribution, independently of the current values of the position variables. For the KE

$$K(p) = \sum_{i=1}^{d} \frac{p_i^2}{2m_i},$$

   the d momentum variables are independent, with p_i having mean zero and variance m_i.

2. A Metropolis update is performed, using Hamiltonian dynamics to propose a new state. Starting with current state q, p, Hamiltonian dynamics are simulated for L steps using **leapfrog method** with stepsize ϵ. The momentum variables at the end of the L step trajectory are negated, giving a proposed state (q^*, p^*). This proposed state is accepted as the next state of the chain with probability

$$\min\left[1, \exp(-H(q^*, p^{*)} + H(q,p))\right] = \min\left[1, \exp\left(-U(q^{*)} + U(q) - K(p^{*)} + K(p)\right)\right].$$

Thus, HMC uses Hamiltonian dynamics and a Metropolis update from a leapfrog proposal to sample from a desired distribution.

4 Langevin Dynamics

Now that we have laid the groundwork for Hamiltonian Monte Carlo, we can understand Langevin Dynamics both as a discretization of the continuous-time Langevin equation and as a 1-step Hamiltonian leapfrog proposal.

One HMC iteration for q with leapfrog can be expressed as follows:

1. Sample values for the momentum variables, ζ, which are iid $N(0,1)$.

2. Propose new value θ^*:

$$\theta_i^* = \theta_i - \frac{\epsilon^2}{2}\frac{\partial U}{\partial \theta_i}(\theta) + \epsilon\zeta_i,$$

and accept θ^* as the new state with some probability.

This Monte Carlo update process is equivalent to a one-step Euler-Maruyama discrete time approximation of the Langevin diffusion.

4.1 Langevin Equation

The Langevin equation is a stochastic differential equation (SDE), which describes how a system evolves when subjected to a combination of deterministic and stochastic forces. The original formulation of the Langevin equation describes Brownian motion:

$$m\frac{dv}{dt} = -\lambda v + \eta(t),$$

where v is the velocity, λ is the damping coefficient, and m is mass. η is a noise term with Gaussian probability distribution such that:

$$\left\langle \eta_i(t)\eta_j(t')\right\rangle = 2\lambda k_B T\delta_{i,j}\delta(t-t')$$

for k_B the Boltzmann constant, T temperature, $\eta_i(t)$ the i-th component of vector $\eta(t)$.

Like all SDEs, the above form is just a notational abbreviation for

$$mv = \int^t (-\lambda v + \eta(t)).$$

4.1.1 Discrete Approximation to Langevin Equation

Because the Langevin equation describes a random continuous time process, we are often interested in approximating sample paths of the Langevin diffusion with various discrete time methods. The most fundamental of these approximations arises from the **Euler-Maruyama method**, which is an extension of Euler's method for solving deterministic ordinary differential equations.

The Euler-Maruyama approximation from time τ_n to τ_{n+1} is derived as follows:

Consider the SDE

$$dX_t = a(X_t, t)dt + b(X_t, t)dW_t$$

with initial condition $X_0 = x_0$ and W_t the Wiener process. We can write

$$X_{\tau_{n+1}} = X_{\tau_n} + \int_{\tau_n}^{\tau_{n+1}} a(X_s, s)ds + \int_{\tau_n}^{\tau_{n+1}} b(X_s, s)dW_s.$$

integral form of SDE. For a sufficiently small time interval $[\tau_n, \tau_{n+1}]$, approximate the functions a, b as $a(X_s, s) \approx a(X_n, \tau_n)$ and $b(X_s, s) \approx b(X_n, \tau_n)$.

Then we have

$$X_{\tau_{n+1}} \approx X_{\tau_n} + \int_{\tau_n}^{\tau_{n+1}} a(X_n, \tau_n) ds + \int_{\tau_n}^{\tau_{n+1}} b(X_n, \tau_n) dW_s$$

so

$$X_{\tau_{n+1}} \approx X_{\tau_n} + a(X_n, \tau_n)(\Delta\tau) + b(X_n, \tau_n)(W_{\tau_{n+1}} - W_{\tau_n}).$$

Since W is the Wiener process, we know that

$$W_{\tau_{n+1}} - W_{\tau_n} \sim N(\Delta\tau)$$

Therefore, for stepsize $\Delta\tau$,

$$X_{\tau_{n+1}} \approx X_{\tau_n} + a(X_n, \tau_n)(\Delta\tau) + b(X_n, \tau_n)\zeta_k\sqrt{\Delta\tau}$$

where ζ_k are independently and identically distributed as $\sim N(0, 1)$.

4.2 Connection Between Euler-Maruyama and Leapfrog

In many physics and machine learning applications, the stochastic differential equation for Langevin dynamics is overdamped or written with a $\sqrt{2}$ term such that

$$dX_t = -\nabla U(X_t)dt + \sqrt{2}dW_t.$$

Recall the potential energy formulation from the canonical distribution. Then the Euler-Maruyama expression for step size τ and iid $\zeta_k \sim N(0, 1)$ can be written:

$$X_{t+\tau} = X_t + \tau\nabla \log p(X_t) + \sqrt{2\tau}\zeta_t.$$

Compare this to a Hamiltonian leapfrog proposal for θ^* :

$$\theta^*_{t+\tau} = \theta_t + \frac{\epsilon^2}{2}\frac{\partial U}{\partial \theta_i}(\theta) + \epsilon\zeta_i$$

with step size $\tau = \frac{\epsilon^2}{2}$ and potential energy $U(\theta) = -\log[p(\theta)L(\theta|D)]$.

Rewriting,

$$\theta^*_{t+\Delta t} = \theta_t + \frac{\epsilon^2}{2}\nabla \log[\pi(\theta_t)L(\theta_t|D)] + \epsilon\zeta_i$$

$$\implies \Delta\theta_t = \epsilon^2/2\left(\nabla \log(p(\theta_t) + \sum_{i=1}^{N} p(x_i|\theta_t))\right) + \epsilon\zeta_i.$$

Let $\epsilon' = \epsilon^2$. Then

$$\implies \Delta\theta_t = \frac{\epsilon'}{2}\left(\nabla \log(p(\theta_t) + \sum_{i=1}^{N} p(x_i|\theta_t))\right) + \sqrt{\epsilon'}\zeta_t$$

and we recover

$$\Delta\theta_t = \frac{\epsilon'}{2}\left(\nabla \log p(\theta_t) + \sum_{i=1}^{N} \nabla \log p(x_i|\theta_t)\right) + \eta_t, \eta_t \sim N(0, \epsilon').$$

Therefore, we can understand Langevin dynamics both as an SDE and as a Hamiltonian leapfrog proposal. However, these updates still require computation over the entire dataset and are still limited. This is addressed by **stochastic gradient Langevin dynamics**.

5 Stochastic Gradient Langevin Dynamics

The core of SGLD lies in recognizing that the Langevin dynamics update

$$\Delta\theta_t = \frac{\epsilon'}{2}\left(\nabla \log p(\theta_t) + \sum_{i=1}^{N} \nabla \log p(x_i|\theta_t)\right) + \eta_t, \eta_t \sim N(0, \epsilon').$$

looks very similar to SGD, which we saw before:

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

The authors combine the two ideas, proposing a new update scheme:

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

where $\eta_t \sim N(0, \epsilon_t)$ and the step size decreases towards 0 at the suitable rates determined from Robbins-Monro.

Let $g(\theta) = \nabla \log p(\theta) + \sum_{i=1}^{N} \nabla \log p(x_i|\theta)$ be the true gradient of the log probability at θ and

$$h_t(\theta) = \nabla \log p(\theta) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{ti}|\theta) - g(\theta).$$

Then the stochastic gradient is $g(\theta) + h_t(\theta)$ where $h_t(\theta)$ is a zero mean random variable with finite variance $V(\theta)$.

And the proposed update

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

becomes

$$\Delta\theta_t = \frac{\epsilon_t}{2}(g(\theta_t) + h_t(\theta_t)) + \eta_t.$$

5.1 Transition to Langevin Dynamics

There are two sources of randomness in the update

$$\Delta\theta_t = \frac{\epsilon_t}{2}(g(\theta_t) + h_t(\theta_t)) + \eta_t.$$

There is the injected Gaussian noise with variance ϵ_t, as well as the noise in the stochastic gradient, which has variance $\left(\frac{\epsilon_t}{2}\right)^2 V(\theta_t)$.

For large t, $\epsilon_t \to 0$. Then the injected noise will dominate the stochastic gradient noise. So

$$\Delta\theta_t = \frac{\epsilon_t}{2}(g(\theta_t) + h_t(\theta_t)) + \eta_t$$

essentially becomes Langevin dynamics, where differences in the parameters are determined by the random Gaussian variables η. So, run sufficiently long, SGLD will mimic the Bayesian sampling method of Langevin dynamics.

Notice also as $\epsilon_t \to 0$, the discretization error of the Langevin dynamics will be negligible. The Metropolis rejection probability will approach 0, and so we can ignore the proposal accept/reject step in subsequent updates.

It remains to show that the sequence of parameters $\theta_1, \theta_2, ...$ converges to the posterior distribution. Following the authors, we will show that a subsequence $\theta_{t_1}, \theta_{t_2}, ...$ will converge to the posterior, implying the same of the whole sequence.

Let ϵ_0 be fixed and small $\ll 1$. Since $\{\epsilon_t\}$ satisfies the Robbins-Monro step size property, there exists a subsequence $t_1 < t_2 < ...$ such that

$$\sum_{t=t_s+1}^{t_{s+1}} \epsilon_t \to \epsilon_0 \text{ as } s \to \infty.$$

Since the injected noise is independent, for large enough s the total injected noise

$$|| \sum_{t=t_s+1}^{t_{s+1}} \eta_t ||_2$$

between steps t_s and t_{s+1} is $O(\sqrt{\epsilon_0})$. Because $\epsilon_0 \ll 1$, we can say

$$||\theta_t - \theta_{t_s} \ll 1$$

for t between the same steps. Assuming the gradient of $g(\cdot)$ varies smoothly, the total stochastic gradient can be written as

$$\sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2}(g(\theta_t) + h_t(\theta_t)) = \frac{\epsilon_0}{2} g(\theta_{t_s}) + O(\epsilon_0) + \sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} h_t(\theta_t).$$

Then the randomness in $h_t(\theta_t)$ is dominated by the randomness in choice of mini-batch. If they are chosen randomly and independently, $h_t(\theta_t)$ for each t will be essentially *iid*. Then the variance of the $\sum_{t=t_s+1}^{t_{s+1}} \frac{\epsilon_t}{2} h_t(\theta_t)$ term is $O(\sum_t \frac{\epsilon_t^2}{4})$ and

$$= \frac{\epsilon_0}{2} g(\theta_{t_s}) + O(\epsilon_0).$$

So the total stochastic gradient step is approximately the exact gradient step at θ_{t_s} with step size ϵ_0. This deviation is dominated by $O(\epsilon_0)$, which is then dominated by the injected noise $O(\sqrt{\epsilon_0})$. Then the subsequence approaches a sequence generated by Langevin dynamics with fixed step size ϵ_0. This implies convergence of the full sequence to the posterior distribution with infinite effective sample size.

6 Discussion

With the above analysis, we know that SGLD will initially imitate stochastic gradient ascent. It will then mimic a Langevin dynamics Metropolis-Hastings algorithm, with a smooth transition between the two phases.

This is an incredibly interesting result! Injecting specified noise to stochastic gradient updates and annealing the step size properly leads to efficient Bayesian sampling of the true posterior, somewhat like early stopping.