Posterior Sampling with Stochastic Gradient Langevin Dynamics

Umut Şimşekli

1 Introduction

So far in this course, we have discussed several different tasks that arise in statistical learning, and we expressed most of these tasks by certain optimization problems (such as the empirical risk minimization problem). The Maximum A-Posteriori (MAP) estimation problem being one of the examples of such optimization problems, arose naturally when we limited ourselves to the framework of probabilistic modeling. In this lecture, we will again consider the probabilistic modeling framework, but this time we will ask a different (and potentially more difficult) question than the MAP estimation problem.

Before defining the problem, let us remember what a Bayesian probabilistic model was. The main idea in Bayesian modeling is to treat the parameter of interest $\theta \in \mathbb{R}^d$ as a random vector that follows a *prior distribution* $p(\theta)$, and a single observation y is assumed to be a random variable drawn from the *conditional distribution* $p(y|\theta)$. We typically consider n observations $D_n = \{y_1, \dots, y_n\}$, which are i.i.d. distribution when *conditioned* on θ. We often denote a Bayesian probabilistic model as follows:

$$\theta \sim p(\theta) \tag{1}$$

$$y_i|\theta \sim p(y|\theta), \quad \text{for } i = 1, \dots, n. \tag{2}$$

Under this model, the goal in MAP estimation is to find the value $\hat{\theta}_n$ that maximizes the *posterior distribution* (in this lecture we will use the words distribution and density exchangeably):

$$p(\theta|D_n) = \frac{p(D_n|\theta)p(\theta)}{p(D_n)}. \tag{3}$$

Using the fact that the logarithm function is monotonic and $p(D_n)$ does not depend on θ, the MAP estimation problem can be written in the following form:

$$\hat{\theta}_n \in \arg\max_{\theta} \left\{ \sum_{i=1}^{n} \log p(y_i|\theta) + \log p(\theta) \right\}. \tag{4}$$

The MAP estimate can be very useful in practical applications and there is a broad literature for solving this optimization problem for a variety of choices of prior distributions and likelihood functions, with various theoretical guarantees. However, in a broad range of applications, a single point estimate can be insufficient, especially for risk-intolerant applications (such as autonomous driving). Hence, it would be of interest to extract more information about our estimate to have an understanding of the level of uncertainty in our estimation.

While there are many ways of doing this, here, we will consider the Bayesian approach and try to infer the uncertainty information by making a better use of the posterior distribution. For instance, as a measure of uncertainty, we can consider the posterior variance:

$$\sigma^2_{\text{post}} = \int \|\theta - \mu_{\text{post}}\|^2 \, p(\theta|D_n) \, \mathrm{d}\theta, \tag{5}$$

where $\mu_{\text{post}} = \int \theta p(\theta|D_n)\,\mathrm{d}\theta$ is the posterior expectation. In this context, if the value of σ^2_{post} is high, we might conclude that there is a high uncertainty in our estimation (since the posterior distribution is too spread), or if σ^2_{post} is low, we might be happy with our estimation.

In general, the goal in Bayesian statistics is to compute the expectation of a *test function* $f : \mathbb{R}^d \mapsto \mathbb{R}$ under the posterior distribution:

$$\bar{f} := \mathbb{E}_{\theta\sim p(\theta|D_n)}[f(\theta)] = \int f(\theta)p(\theta|D_n)\mathrm{d}\theta. \tag{6}$$

As opposed to MAP estimation, where the marginal likelihood $p(D_n)$ is irrelevant, hence can be safely discarded, unfortunately, in this task we need all the information about the posterior distribution, including $p(D_n) = \int p(D_n|\theta)p(\theta)\mathrm{d}\theta$. Therefore, computing these integrals can become very difficult very quickly, especially when d is large.

2 Markov Chain Monte Carlo

There are many ways to approximately compute integrals. In this lecture, we will consider an instance of a powerful family of tools, that is called Markov Chain Monte Carlo (MCMC) methods. In this section, we will cover the main idea in MCMC and then in the next section we will introduce the algorithm that is the main topic of this lecture.

Let us first consider the following "ideal" scenario and assume that we have access to an algorithm that can give us independent random samples drawn from the posterior distribution $p(\theta|D_n)$:

$$\theta_k \sim p(\theta|D_n), \quad k = 1, \dots, K. \tag{7}$$

Then, if this was the case, we could introduce an estimator to Equation 6 in an easy manner:

$$\hat{f}_K := \frac{1}{K}\sum_{k=1}^{K} f(\theta_k). \tag{8}$$

This estimator is called the Monte Carlo estimator and under mild integrability assumptions, we would know that $\hat{f}_K \to \bar{f}$ almost surely as $K \to \infty$ (why?). However, this approach introduces a new question: *"How can we draw samples from $p(\theta|D_n)$?"*

The goal in MCMC methods is to address this question through the theory of Markov processes. Essentially we will try to "design" a Markov process $\theta_t \in \mathbb{R}^d \in$ (which can be either continuous-time $t \in \mathbb{R}_+$ or discrete-time $t \in \{0, 1, 2, \dots\}$) in such a way that the probability distribution of θ_t, let us denote it by $q(\theta, t)$, converges to the posterior distribution, $q(\theta, t) \to p(\theta|D_n)$ as $t \to \infty$.

If we can manage to design such a Markov process, the above convergence property will imply that, as we keep simulating the Markov process, the probability distribution of the latest sample will get closer and closer to the posterior distribution. Hence, after simulating the Markov process sufficiently long, we can treat the simulated trajectory of the process as "approximate samples" drawn from the posterior distribution. Then finally, we can use them for approximating $\bar{f}$ by computing a sample average as in Equation 8. This idea will become more clear in the next section.

3 Stochastic Gradient Langevin Dynamics

The Langevin Equation. The Stochastic Gradient Langevin Dynamics (SGLD) algorithm was relatively recently proposed as an alternative to existing MCMC approaches [WT11], where its main

was to be able to scale up to large-scale, modern machine learning problems. The algorithm is based on the celebrated Langevin equation, which arose from statistical physics. But before defining the equation, let us introduce some terms, which will ease the overall notation clutter.

We will follow the classical Bayesian statistics notation and from now on, we will denote the posterior distribution by using π, such that $\pi(\theta) = p(\theta|D_n)$. Furthermore, we define the "potential energy function" (again the name coming from physics) as

$$U(\theta) := -\sum_{i=1}^{n} \log p(y_i|\theta) - \log p(\theta). \tag{9}$$

Note that we have access to the function U, since we assume that we know both the likelihood function and the prior distribution. With this notation, we have: (check yourself!)

$$\pi(\theta) = \frac{\exp(-U(\theta))}{\int \exp(-U(\theta'))\mathrm{d}\theta'}. \tag{10}$$

Even though we have the annoying marginal likelihood term $p(D_n)$ in the denominator (now in the form of $\exp(-U(\theta'))\mathrm{d}\theta'$ – check if you are not convinced), notice that if we want to compute the gradient of $\log \pi$ with respect to θ, this term no longer appears, i.e.,

$$\nabla \log \pi(\theta) = -\nabla U(\theta). \tag{11}$$

This shows that, even though we do not know what the marginal likelihood is, we can still compute the gradient of $\log \pi$ since we have access to U. This is an important property and will form one the bases of SGLD.

Now, we are ready to define the Langevin equation, which is a Stochastic Differential Equation (SDE):

$$\mathrm{d}\theta_t = -\nabla U(\theta_t)\mathrm{d}t + \sqrt{2}\mathrm{dB}_t, \tag{12}$$

where B_t denotes the standard Brownian motion in $\mathbb{R}^d$, and is defined as follows:

- $\mathrm{B}_0 = 0$ almost surely;
- For any $t_0 < t_1 < \cdots < t_N$, the increments $\mathrm{B}_{t_n} - \mathrm{B}_{t_{n-1}}$ are independent, $n = 1, 2, \ldots, N$;
- The difference $\mathrm{B}_t - \mathrm{B}_s$ and B_{t-s} have the same distribution: $\mathcal{N}(0, (t-s)\mathrm{I})$ for $s < t$;
- B_t is continuous almost surely.

The SDE in Equation 12 is in fact a short-hand notation for the *Markov process* that is defined as follows:

$$\theta_t = \theta_0 - \int_0^t \nabla U(\theta_s)\mathrm{d}s + \sqrt{2}\mathrm{B}_t, \tag{13}$$

where θ_0 is some initialization. This more explicit representation might be for intuitive for the ones who are not familiar with SDEs.

Let us illustrate why the Langevin equation is interesting for our purposes. Let us denote the trajectory $(\theta_t)_{t\geq 0}$ as defined in Equation 13 (this is called a "solution" of the SDE). We will try to understand how the probability distribution of θ_t evolves over "time" t. To do so, let us denote the distribution of θ_t by the density function $q(\theta, t)$. We will now illustrate that $q(\cdot, t) \to \pi$ as $t \to \infty$, which is going to be crucial for the reasons that we discussed in the previous section.

For simplicity, let us assume $d = 1$ (for general d the argument is pretty much the same), such that Equation 12 becomes

$$\mathrm{d}\theta_t = -\partial_\theta U(\theta_t) + \sqrt{2}\mathrm{dB}_t, \tag{14}$$

where ∂_θ denotes derivation with respect to θ. To understand how $q(\theta, t)$ evolves, we will use another celebrated equation, called the Fokker-Planck equation, which governs the evolution of $q(\theta, t)$ through the following partial differential equation:

$$\partial_t q(\theta, t) = \partial_\theta \left[\partial_\theta U(\theta) q(\theta, t)\right] + \partial_\theta^2 q(\theta, t). \tag{15}$$

This equation characterizes how the "change" in $q(\cdot, t)$ behaves, i.e. $\partial_t q(\theta, t)$. Now, the idea is, if $q(\cdot, t)$ converges to a distribution as $t \to \infty$, then whenever this limit is reached, there should not be any more changes in q. In other words, whenever $q(\cdot, t)$ hits its limit, $\partial_t q(\theta, t)$ has to be equal to 0.

Therefore, we can simply "check" if π is a limit of $q(\cdot, t)$ by replacing $q(\theta, t)$ with $\pi(\theta)$ in Equation 15 and observing whether the right-hand-side of Equation 15 is equal to 0 or not. Let us apply this procedure:

$$\begin{aligned}
\partial_\theta \left[\partial_\theta U(\theta) \pi(\theta)\right] + \partial_\theta^2 \pi(\theta) &= \partial_\theta \left[\partial_\theta U(\theta) \pi(\theta) + \partial_\theta \pi(\theta)\right] && (16)\\
&= \partial_\theta \left[\partial_\theta U(\theta) \pi(\theta) - \partial_\theta U(\theta) \pi(\theta)\right] && (17)\\
&= 0, && (18)
\end{aligned}$$

where we used the fact that

$$\begin{aligned}
\partial_\theta U(\theta) &= -\partial_\theta \log \pi(\theta) && (19)\\
&= -\frac{1}{\pi(\theta)} \partial_\theta \pi(\theta), && (20)
\end{aligned}$$

hence $\partial_\theta \pi(\theta) = -\pi(\theta) \partial_\theta U(\theta)$.

This nice property shows that, if we could simulate the process in Equation 13 and obtain $(\theta_t)_{t \geq 0}$, the distribution of θ_t will get closer and closer to π, which is our posterior distribution (in fact we need to prove a bit more to ensure this rigorously, but it is out of current scope). Hence, if we could simulate the process, we could use θ_t as an "approximate sample" from π, given that t is large enough. And given our ultimate goal of approximating expectations by finite averages, such θ_t can be used conveniently in the sample averages.

Numerical Simulation. Unfortunately, an exact simulation of $(\theta_t)_{t \geq 0}$ is not possible in general since t is continuous and Equation 13 cannot be analytically computed in general. However, we can develop approximate simulation algorithms with nice computational properties. In the stochastic analysis literature, such approximations are called "disretizations", since we would like to "discretize" the continuous-time process.

One of the most common and also simplest discretization schemes is called the Euler-Maruyama scheme, which is based on the following idea. Given θ_0, let us try to compute θ_η, for a very small η. Then, following Equation 13, we have:

$$\theta_\eta = \theta_0 - \int_0^\eta \nabla U(\theta_t) \mathrm{d}t + \sqrt{2} \mathrm{B}_\eta. \tag{21}$$

By using the definition of the Brownian motion, we know that $\sqrt{2}\mathrm{B}_\eta$ is equal in distribution to $\sqrt{2\eta} Z$, where $Z \sim \mathcal{N}(0, \mathrm{I})$ (here we used the fact that $\mathrm{B}_\eta \sim \mathcal{N}(0, \eta \mathrm{I})$). On the other hand, since we choose η very small, we can assume that $\nabla U(\theta_t)$ "does not change much" for $t \in [0, \eta]$. Hence, with this reasoning, we can replace $\nabla U(\theta_t)$ with $\nabla U(\theta_0)$ in the above equation. This will give us:

$$\begin{aligned}
\theta_\eta &\approx \theta_0 - \int_0^\eta \nabla U(\theta_0) \mathrm{d}t + \sqrt{2\eta} Z && (22)\\
&= \theta_0 - \left(\int_0^\eta \mathrm{d}t\right) \nabla U(\theta_0) + \sqrt{2\eta} Z && (23)\\
&= \theta_0 - \eta \nabla U(\theta_0) + \sqrt{2\eta} Z. && (24)
\end{aligned}$$

We can now iterate this approach k times, which gives us a recursion, which can be easily implementable on a computer:

$$\theta_{k\eta} \approx \theta_{(k-1)\eta} - \eta \nabla U(\theta_{(k-1)\eta}) + \sqrt{2\eta} Z_k, \tag{25}$$

where $Z_k \sim \mathcal{N}(0, \mathrm{I})$ for all k.

To simplify the notation, we can drop the dependence on η, which finally yields the following recursion:

$$\theta_k = \theta_{k-1} - \eta \nabla U(\theta_{k-1}) + \sqrt{2\eta} Z_k, \tag{26}$$

where Z_k is defined in the same way as before and η is now called the "step-size". This recursion is called Langevin Monte Carlo (or sometimes the Unadjusted Langevin Algorithm).

Notice that without the terms Z_k in the above equation, the recursion just reduces to the gradient descent algorithm! Hence, we can view this algorithm as a "noisy" version of gradient descent, such that when the "correct amount of noise" is added to the iterates, the algorithm starts providing approximate samples from the posterior distribution.

Having the recursion in Equation 26 at hand, we can now try to approximate the expectations in Equation 6. Assume that we have run the recursion in Equation 26 for a large number of iterations, say until $k = K_0$, such that we can assume that the distribution of θ_k is close to the posterior π. Then, we can generate K more samples to approximate the expectations:

$$\bar{f} = \int f(\theta) p(\theta | D_n) \mathrm{d}\theta \approx \frac{1}{K} \sum_{k=K_0+1}^{K_0+K} f(\theta_k) =: \hat{f}_K. \tag{27}$$

The initial number of iterations K_0 is called the "burn-in" period, where the Markov process is "getting ready to sample" from π.

Stochastic Gradient Langevin Dynamics. Even though the recursion in Equation 26 provides a practical algorithm, its computational complexity can be prohibitively high when the number data points n is very large, which is the typical case in modern machine learning applications. To see this, we need to remember the definition the function ∇U:

$$\nabla U(\theta) = -\sum_{i=1}^{n} \nabla \log p(y_i | \theta) - \nabla \log p(\theta). \tag{28}$$

Since we need to compute this gradient at every iteration, this means that we need to compute the gradient of the log-likelihood for n different data points at every iteration. Clearly, this can be problematic (and perhaps unnecessary) when n is very large.

The idea in SGLD is to address this issue by replacing ∇U with an unbiased estimator, which is much simpler to compute. More precisely, it replaces ∇U with the "stochastic gradient", that is the gradient of the following function:

$$\hat{U}_k(\theta) := -\frac{n}{b} \sum_{i \in \Omega_k} \log p(y_i | \theta) - \log p(\theta), \tag{29}$$

where $\Omega_k \subset \{1, 2, \ldots, n\}$ is a random data subsample that is drawn either with or without replacement, and $b = |\Omega_k|$ denotes the number of elements in Ω_k and typically b is much smaller than n. Notice that when conditioned on θ, $\nabla \hat{U}_k(\theta)$ is an unbiased estimator of $\nabla U(\theta)$, i.e. $\mathbb{E}[\nabla \hat{U}_k(\theta) | \theta] = \nabla U(\theta)$ for all k.

By replacing ∇U with $\nabla \hat{U}_k$ in Equation 26, we only need to compute the gradients of b terms instead of n terms at each iteration. This finally gives us the recursion of the SGLD algorithm:

$$\theta_k = \theta_{k-1} - \eta \nabla \hat{U}_k(\theta_{k-1}) + \sqrt{2\eta} Z_k. \tag{30}$$

By following a similar strategy, we can define the SGLD estimator as before, i.e, $\frac{1}{K}\sum_{k=K_0+1}^{K_0+K} f(\theta_k)$.

A final remark on this topic is that, since we are making a series of approximations, it is no longer guaranteed that the distribution of θ_k converges to π. Namely, we make the following approximations:

- Discretization of the process
- Replacing the true gradient with the stochastic gradient.

Clearly, at each step we introduce a certain amount of error. However, fortunately, it has been well-studied that this error can be controlled when η is small enough and b is large enough. For more information on the topic, you can refer to the survey paper [NF21].

References

[NF21] Christopher Nemeth and Paul Fearnhead. Stochastic gradient markov chain monte carlo. *Journal of the American Statistical Association*, 116(533):433–450, 2021.

[WT11] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.