Chapter 15

Sampling Methods

15.1 Introduction

In Bayesian inference, **distributions** are the ultimate tool for representing knowledge about unknown quantities. This is the reason that we try to find $p(\boldsymbol{\theta}|\mathcal{D})$. If we have the distribution, we can find the **expected value** of various functions of the unknown quantity and in this way find point estimates or the probability of an event,

$$\hat{\boldsymbol{\theta}}_{Bayes} = \mathbb{E}[\boldsymbol{\Theta}|\mathcal{D}],$$
$$p(\boldsymbol{\theta} \in A|\mathcal{D}) = \mathbb{E}[\mathbb{1}(\boldsymbol{\Theta} \in A)|\mathcal{D}],$$

where A is an event and $\mathbb{1}$(condition) equals 1 if the condition holds and is 0 otherwise.

If we find the posterior distribution in closed form and it turns out to be one of the common distributions, e.g., Gaussian, Poisson, etc, then typically, we can easily compute expected values. However, this is not always the case, and we may face two difficulties:

1. Sometimes all we have is a function $q(\boldsymbol{\theta})$ that is proportional to $p(\boldsymbol{\theta}|\mathcal{D})$,

   $$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}) = q(\boldsymbol{\theta})$$

   and we are not even able to compute $p(\boldsymbol{\theta}|\mathcal{D})$ for a given $\boldsymbol{\theta}$ because the normalization factor is not known.

2. Even if we can compute $p(\boldsymbol{\theta}|\mathcal{D})$, computing expected values requires integration, which may be computationally infeasible.

In such cases, sampling from this distribution will be useful because sampling allows us to find expected values. For a function h, by the law of large numbers we have the following approximation

$$\mathbb{E}[h(X)] \simeq \sum_{i=1}^{N} h(x_i),$$

where x_i are independent samples drawn from the distribution p_X with respect to which the expected value is to be computed.

For example, recall that in Bayesian linear regression, a common likelihood is

$$\boldsymbol{y}|\boldsymbol{\theta}, \sigma^2 \sim \mathcal{N}(X\boldsymbol{\theta}, \sigma^2 I),$$

with prior

$$p(\boldsymbol{\theta}, \sigma^2) \propto 1/\sigma^2.$$

For this model, we found $p(\boldsymbol{\theta}|\mathcal{D}, \sigma^2)$ and $p(\sigma^2|\mathcal{D})$ and stated that while it is possible to obtain $p(\boldsymbol{\theta}|\mathcal{D})$ analytically, doing so is complicated. In practice, we proceed computationally by generating samples from $p(\sigma^2|\boldsymbol{y})$ and then $p(\boldsymbol{\theta}|\boldsymbol{y}, \sigma^2)$. With this sampling approach we can also perform prediction for a given input vector $\boldsymbol{x}_{n+1}$ of by producing samples from $p(y_{n+1}|\boldsymbol{\theta}, \sigma^2) \sim \mathcal{N}(\boldsymbol{x}_{n+1}^T\boldsymbol{\theta}, \sigma^2)$, and answer question such as finding $p(y_{n+1} > a|\boldsymbol{\theta}, \sigma^2)$ for a given constant a.

In this chapter, we will discuss methods for generating samples from a distribution $p(\boldsymbol{\theta})$ which we can only compute up to a multiplicative constant. The approach is identical for conditional distributions such as $p(\boldsymbol{\theta}|\mathcal{D})$. To emphasize the fact that the constant may not be known, we use p to refer to the true distribution and q to the "distribution" without the constant. We will use $\mathbb{E}_p$ to denote expectation with respect to distribution p. For a non-normalized distribution q we define $\mathbb{E}_q = \mathbb{E}_p$.

15.2 Basic Sampling Techniques

In this section, we will review some basic but useful sampling techniques.

15.2.1 Deterministic Integration

This method is not actually a sampling method but rather tries to approximate the expected value by approximating the corresponding integral over a grid,

$$\mathbb{E}_q[h(\boldsymbol{\theta})] = \int h(\boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta} \simeq \frac{\sum_{i=1}^N h(\boldsymbol{\theta}_i)q(\boldsymbol{\theta}_i)}{\sum_{i=1}^N q(\boldsymbol{\theta}_i)},$$

where $\boldsymbol{\theta}_i$ form a uniform grid covering the support of q. This method becomes computationally prohibitive if the number of dimensions of $\boldsymbol{\theta}$ is large.

15.2.1.1 The Inverse-CDF Method

Suppose θ is one dimensional and that we have the CDF $F(\theta)$. First, assume θ is continuous and $F(\theta)$ is invertible. Inverse-CDF sampling relies on sampling from the uniform distribution to generate samples for potentially more complex distributions. For $i = 1, \ldots, N$,

1. Generate $U_i \sim \text{Uni}[0, 1]$;
2. Let $\theta_i = F^{-1}(U_i)$.

Claim: If $U \sim \text{Uni}[0, 1]$, then $\theta = F^{-1}(U)$ has CDF F. To see this observe that:

$$p(\theta \leq c) = p(F^{-1}(u) \leq c) = p(U \leq F(c)) = F(c).$$

The algorithm is slightly modified if F has discontinuities or is not invertible. Specifically, we define $F^{-1}(u) = \min\{x : F(x) \geq u\}$.

15.2.2 Rejection Sampling

In rejection sampling, to produce samples for a distribution q, we first produce samples from another distribution g but then only keep some of the samples produced in a way that the resulting distribution is q. The distribution g needs to satisfy

$$\begin{aligned} &g(\boldsymbol{\theta}) > 0, && \text{if } q(\boldsymbol{\theta}) > 0, \\ &q(\boldsymbol{\theta}) \leq Mg(\boldsymbol{\theta}) && \text{for some known } M \text{ and for all } \boldsymbol{\theta}. \end{aligned}$$

We also need to sample $u \sim \text{Uni}(0, 1)$.

Rejection Sampling

1. Sample $\boldsymbol{\theta}' \sim g$.

2. Sample $U \sim \text{Uni}(0,1)$.
3. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$ if $U \leq \frac{q(\boldsymbol{\theta}')}{Mg(\boldsymbol{\theta}')}$. (Accept $\boldsymbol{\theta}'$ as a new sample if $U \leq \frac{q(\boldsymbol{\theta}')}{Mg(\boldsymbol{\theta}')}$; else reject the sample.)

We define the normalizing constants for the distribution,

$$Z_q = \int q(\boldsymbol{\theta}) d\boldsymbol{\theta}, \qquad Z_g = \int g(\boldsymbol{\theta}) d\boldsymbol{\theta}.$$

Note that the probability of a sample being accepted is

$$\begin{aligned} p(accepted) &= \int p(\boldsymbol{\theta}', accepted) d\boldsymbol{\theta}' = \int p(\boldsymbol{\theta}') p(accepted|\boldsymbol{\theta}') d\boldsymbol{\theta}' \\ &= \int \frac{g(\boldsymbol{\theta}')}{Z_g} \cdot \frac{q(\boldsymbol{\theta}')}{Mg(\boldsymbol{\theta}')} d\boldsymbol{\theta}' = \frac{Z_q}{MZ_g}. \end{aligned}$$

Let us now find the distribution for an accepted sample,

$$\begin{aligned} p(\boldsymbol{\theta}) &= p(\boldsymbol{\theta}'|accepted) \\ &= \frac{p(\boldsymbol{\theta}')p(accepted|\boldsymbol{\theta}')}{p(accepted)} \\ &= \frac{\frac{g(\boldsymbol{\theta}')}{Z_g} \cdot \frac{q(\boldsymbol{\theta}')}{Mg(\boldsymbol{\theta}')}}{\frac{Z_q}{MZ_g}} \\ &= \frac{q(\boldsymbol{\theta}')}{Z_q}, \end{aligned}$$

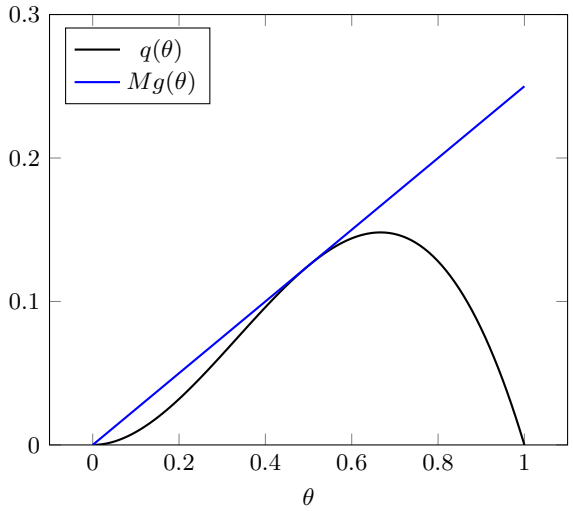
which is the desired distribution.

Rejection sampling does not take advantage of all the samples, unlike importance sampling that we will see next, so in some sense it is inefficient. In particular, if $Z_q = Z_g = 1$, then only a fraction of $\frac{1}{M}$ of the samples will be accepted. If M is large, i.e., g is not a good match for q, then we lose a lot of samples. But rejection sampling has a very important property: it is *self-evaluating*. If we are doing poorly, it is easy to find out by considering the number of samples that are rejected. This is a property that importance sampling lacks.

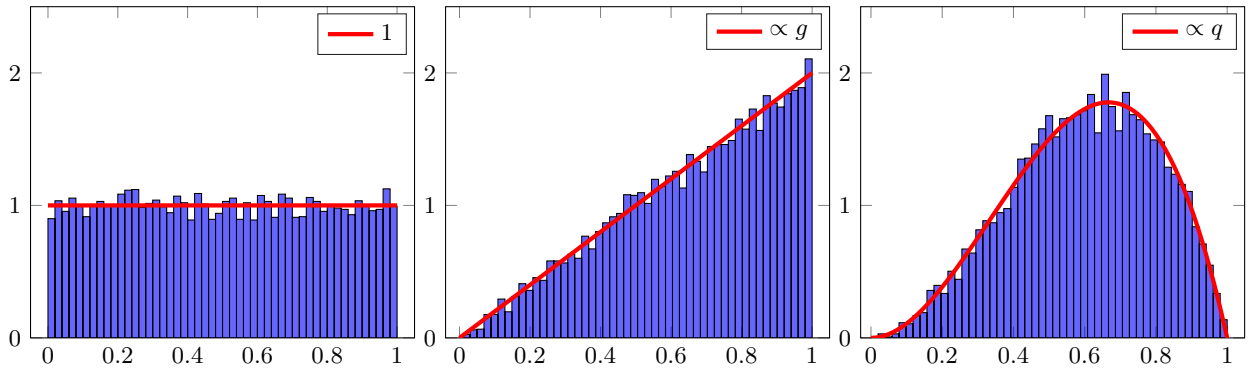
Example 15.1. Suppose we need to sample from Beta$(3,2)$ so we let $q(\theta) = \theta^2(1-\theta)$. We would like to do this by sampling from $g(\theta) = \theta$, which we can do using inverse-CDF sampling. First, let us find the required value for M. Observe that

$$Mg(\theta) \geq q(\theta) \iff M\theta \geq \theta^2(1-\theta) \iff M \geq \theta(1-\theta).$$

So the smallest valid value for M is $1/4$, which is what we will choose. Note that in practice, we don't need to find the smallest possible M. For example, here we could argue that the $\theta(1-\theta) \leq 1$ and so it would have been sufficient to let $M = 1$. The plots of q, g are shown below.



To generate samples, first we generate samples from Uni$(0, 1)$, obtaining $S_1 = \{x_1, \ldots, x_N\}$. To generate samples from g, we use the inverse CDF method. The CDF of g is θ^2 and its inverse is $\sqrt{\theta}$. So, our samples become $S_2 = \{\theta'_1, \ldots, \theta'_N\}$, where $\theta'_i = \sqrt{x_i}$. We then accept/reject these based on the rejection sampling rule to obtain S_3, which are samples with distribution q. Specifically, for a sample θ'_i, we accept it with probability $4\theta'_i(1 - \theta'_i)$. Note that this step again requires generating uniform samples, from Uni$(0, 1)$. The graphs below show histograms for x_i, θ'_i and θ_i, as well as the corresponding normalized pdfs. The histograms are normalized so that they are valid pdfs. In this experiment, out of the $N = 1000$ generated samples, 6692 were accepted.



△

15.2.3 Importance Sampling

Again, suppose we are interested in finding

$$\mathbb{E}_q[h(\boldsymbol{\Theta})],$$

where $\mathbb{E}_q$ denotes expectation with respect to distribution q. Now if q is a complicated distribution, we may have a hard time sampling from it. Even if we can sample from q, another issue may arise. The values of $\boldsymbol{\theta}$ such that $h(\boldsymbol{\theta})q(\boldsymbol{\theta})$ are large contribute to the expectation significantly. But $h(\boldsymbol{\theta})q(\boldsymbol{\theta})$ may be large in places where $q(\boldsymbol{\theta})$ is small. So unless we generate a lot of samples, we may not produce one for which $h(\boldsymbol{\theta})q(\boldsymbol{\theta})$ is large, and thus miss significant contributions to the expectation from such points.

Suppose we have a second (possibly unnormalized) distribution $g(\boldsymbol{\theta})$, which is simpler and from which we

can produce samples. Ideally, $g(\boldsymbol{\theta})$ is large if $q(\boldsymbol{\theta})h(\boldsymbol{\theta})$ is large. We have

$$\begin{aligned}\mathbb{E}_q[h(\boldsymbol{\theta})] &= \frac{\int h(\boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta}}{\int q(\boldsymbol{\theta})d\boldsymbol{\theta}} \\ &= \frac{\int h(\boldsymbol{\theta})[q(\boldsymbol{\theta})/g(\boldsymbol{\theta})]g(\boldsymbol{\theta})d\boldsymbol{\theta}}{\int [q(\boldsymbol{\theta})/g(\boldsymbol{\theta})]g(\boldsymbol{\theta})d\boldsymbol{\theta}} \\ &= \frac{\mathbb{E}_g[h(\boldsymbol{\theta})(q(\boldsymbol{\theta})/g(\boldsymbol{\theta}))]}{\mathbb{E}_g[q(\boldsymbol{\theta})/g(\boldsymbol{\theta})]}.\end{aligned}$$

So we have converted the problem into expectation with respect to g. Define $w(\boldsymbol{\theta}) = \frac{q(\boldsymbol{\theta})}{g(\boldsymbol{\theta})}$ as the importance weight or ratio at $\boldsymbol{\theta}$. Then we can estimate $\mathbb{E}_q[h(\boldsymbol{\theta})]$ as

$$\mathbb{E}_q[h(\boldsymbol{\theta})] = \frac{\mathbb{E}_g[h(\boldsymbol{\theta})w(\boldsymbol{\theta})]}{\mathbb{E}_g[w(\boldsymbol{\theta})]} \simeq \frac{\frac{1}{N}\sum_{i=1}^N h(\boldsymbol{\theta}_i)w(\boldsymbol{\theta}_i)}{\frac{1}{N}\sum_{i=1}^N w(\boldsymbol{\theta}_i)}, \qquad \text{with } \boldsymbol{\theta}_i \sim g(\boldsymbol{\theta}),$$

by producing samples from g rather than q.

Of course, if g is small where $h \times q$ is large, we may miss samples for which $h(\boldsymbol{\theta})g(\boldsymbol{\theta})$ makes significant contributions to the expectation; and this is a drawback of importance sampling.

Example 15.2. Let $h(x) = 1 - x$ and $q(x) = x$ for $0 \le x \le 1$. Then

$$\mathbb{E}_q[h(x)] = \int_0^1 (1-x)(2x)dx = \left(x^2 - 2x^3/3\right)_0^1 = 1/3.$$

To estimate this computationally, let $g(x) = 1$. The weights become $w(x) = x$. Generating $N = 100$ samples $x_i \sim \text{Uni}(0, 1)$ using MATLAB, we find

$$\mathbb{E}_q[h(x)] \simeq \frac{\sum_{i=1}^N (1-x_i)x_i}{\sum_{i=1}^N x_i} = 0.34623,$$

which is close to $0.33\cdots$. Of course, for such a simple q we wouldn't resort to importance sampling. △

15.3 Metropolis Monte Carlo

To generate samples from a distribution $p(\boldsymbol{\theta})$, one possible approach is to design a Markov chain whose state space includes all possible values for $\boldsymbol{\theta}$ and its stationary distribution $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\theta})$ is equal to the target distribution $p(\boldsymbol{\theta})$. *In the long term, the number of times that the MC spends in a given state is proportional to the probability of that state.* Hence, we can generate samples from the states of the Markov process by letting it run for a long time and record the states that are visited as samples. The distribution of these samples is approximately the same as $\boldsymbol{\sigma}$ and thus the same as $p(\boldsymbol{\theta})$. This is called Markov Chain Monte Carlo (MCMC).

In this section, we present elegant solutions to the challenging problem of finding a MC satisfying $\boldsymbol{\sigma}(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$. In fact, these methods only need $q \propto p$. While MCs can generate samples with the same distribution, we note that the samples are not independent.

We will first discuss the Metropolis algorithm. This algorithm requires a *jump distribution*, $J(\boldsymbol{\theta}'|\boldsymbol{\theta})$, which proposes a new state $\boldsymbol{\theta}'$ given that we are in state $\boldsymbol{\theta}$. We then either move to $\boldsymbol{\theta}'$ or stay at the current state. The jump distribution is chosen in a way that it guarantees $\boldsymbol{\sigma}(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$. We next describe the Metropolis algorithm more formally. We assume θ is one dimensional for simplicity of notation but this is not a requirement.

Metropolis Algorithm:

1. Choose θ^0 such that $q(\theta) > 0$.

2. For $t = 1, 2, 3, \cdots$, do
   (a) Generate a proposal θ' based on the jump distribution $J(\theta'|\theta^{t-1})$.
   (b) Move to θ' with probability
   $$r = \frac{q(\theta')}{q(\theta^{t-1})}.$$
   Specifically:
   i. Generate $u \sim \text{Uni}[0, 1]$.
   ii. The next state of the MC, θ^t, is given by
   $$\theta^t = \begin{cases} \theta', & u \leq r \\ \theta^{t-1}, & u > r \end{cases} \tag{15.1}$$

The transition probabilities. The rule (15.1) has interesting implications. Note that if $r > 1$, or equivalently if $q(\theta') > q(\theta^{t-1})$, then we will definitely move to θ'. Otherwise, we move to θ' with probability $r = \frac{q(\theta')}{q(\theta^{t-1})} = \frac{p(\theta')}{p(\theta^{t-1})}$. Define $D = \{\theta : p(\theta) > 0\}$ as the set of all possible values of θ based on target distribution p. If the transition probability of $\theta_a \to \theta_b$ in the MC is denoted by $\Pr(\theta_a \to \theta_b)$, we have

$$\Pr(\theta_a \to \theta_b) = J(\theta_b|\theta_a) \min\left(1, \frac{p(\theta_b)}{p(\theta_a)}\right).$$

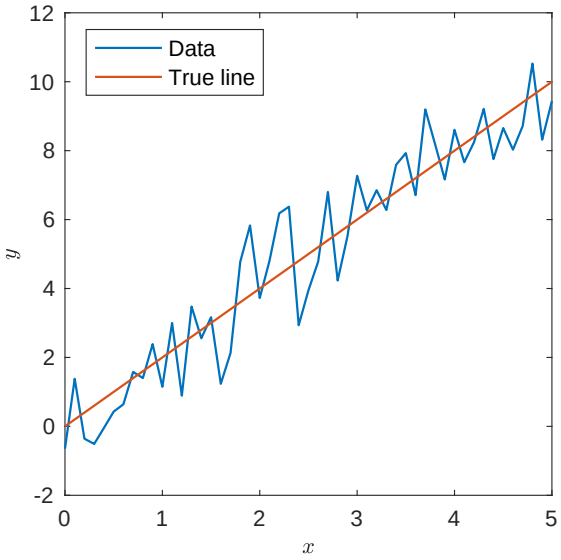
The jump distribution. In the Metropolis algorithm, it is not necessary for the jump distribution to have $p(\theta)$ as a stationary distribution. However, the jump distribution $J(\theta'|\theta^{t-1})$ should satisfy certain constraints, discussed below.

1. *Reachability.* To ensure that the MC is regular, we require that
   $$J(\theta'|\theta) > 0, \qquad \forall \theta, \theta' \in D. \tag{15.2}$$

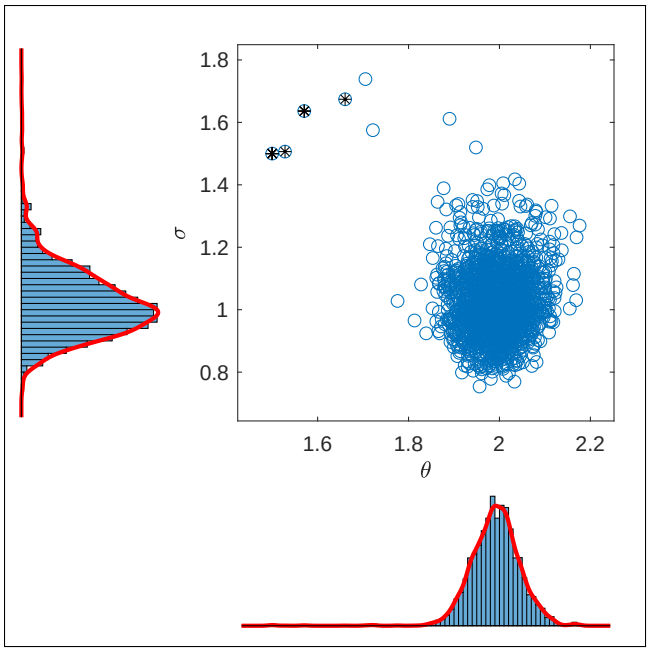
2. *Symmetry.* For $\theta_a, \theta_b \in D$, the detailed balance property with distribution $\pi(\theta) = p(\theta)$ can be written as
   $$p(\theta_a) \Pr(\theta_a \to \theta_b) = p(\theta_b) \Pr(\theta_b \to \theta_a).$$
   Assume without loss of generality that $p(\theta_a) < p(\theta_b)$. Then, the DBP can be written as
   $$p(\theta_a) J(\theta_b|\theta_a) = p(\theta_b) J(\theta_a|\theta_b) \frac{p(\theta_a)}{p(\theta_b)}.$$
   which is satisfied if the jump distribution is symmetric, i.e.,
   $$J(\theta'|\theta) = J(\theta|\theta'), \qquad \forall \theta, \theta' \in D. \tag{15.3}$$

If the jump distribution satisfies (15.2) and (15.3), then the MC is regular and $\boldsymbol{\sigma}(\theta) = p(\theta)$ satisfies the DBP. Hence, $p(\theta)$ is the unique stationary distribution of the Markov chain.

Example 15.3. Consider a Bayesian regression problem where the data as in Figure 15.1a. The data is generated using the distribution
$$y_i|\theta, \sigma \sim \mathcal{N}(\theta x_i, \sigma^2),$$
where the true values are $\theta = 2, \sigma = 1$. The figure provides a plot for the samples $\mathcal{D} = \{(x_i, y_i)_{i=1}^N\}$, where $x_i = 0, 0.1, 0.2, \ldots, 5$ as well as the line $y = 2x$.



(a) Data as well as the true (noiseless) line $y = \theta x$, with $\theta = 2$.



(b) Metropolis sampling for θ and σ. The first 10 samples are marked with $*$.

Figure 15.1: Metropolis sampling for 1-D Bayesian linear regression.

As we have seen, the Bayesian posteriors for this problem are rather complicated. But it is straightforward to obtain estimates using Metropolis sampling. Assuming the prior $p(\theta, \sigma) \propto 1/\sigma^2$, the posterior is

$$\ln p(\theta, \sigma|\mathcal{D}) \propto -(2+N)\ln\sigma - \frac{1}{2\sigma^2}(\boldsymbol{y} - \theta\boldsymbol{x})'(\boldsymbol{y} - \theta\boldsymbol{x}).$$

We use log-probability because probabilities may be very small and for numerical precision, it is better to work with logs. We can convert these to probabilities if we need to. But in this problem, since we are only interested in the samples, we can keep probabilities in log scale.

The samples produced by Metropolis are given in Figure 15.1b. As the jump proposal, we use a product of independent Gaussians:

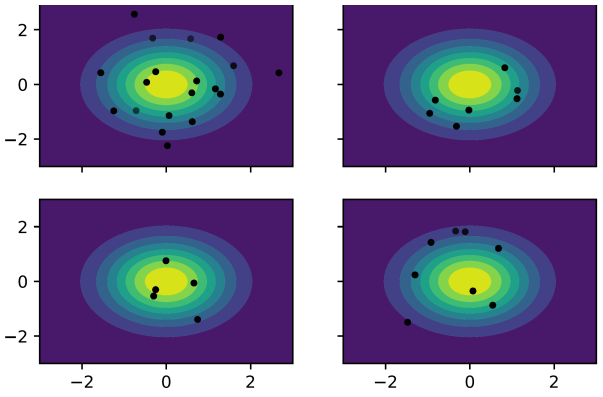
$$\begin{aligned} J(\theta', \sigma'|\theta, \sigma) &= J(\theta'|\theta)J(\sigma'|\sigma), \\ J(\theta'|\theta) &\sim \mathcal{N}(\theta, 0.01), \\ J(\sigma'|\sigma) &\sim \mathcal{N}(\sigma, 0.01). \end{aligned}$$

Based on these samples, the posterior mean for θ is 1.9911 with posterior std 0.055163. The posterior mean for σ is found to be 1.0198. It is also worth noting that the ML estimate for θ is 1.9896. In this example, the estimates are very accurate, which is probably the result of a combination of low noise in the data and chance. △

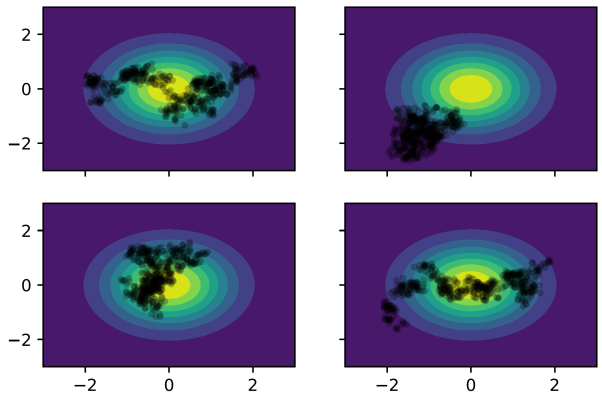
Metropolis-Hastings algorithm. We can eliminate the symmetry property of the jump distribution if we modify r in the Metropolis algorithm as

$$r = \frac{p(\theta')/J(\theta'|\theta^{t-1})}{p(\theta^{t-1})/J(\theta^{t-1}|\theta')}.$$

Exercise 15.4. Prove that with this definition for r, DBP holds even if J is not symmetric. △



(a) Large jumps cause many proposals to be rejected, making the chain stay in its current state.



(b) Small jumps exhibit a random walk behavior which does not necessarily explore the space efficiently.

Figure 15.2: Metropolis sampling with poorly designed jump distributions (4 runs for each case).

Sampling from a MC. Ideally, we should keep only one sample from every m samples for m "large enough" to ensure that the samples are nearly independent. However, there are two issues here:

- It is not easy to determine how large is "large enough."
- If m is too large, the process is inefficient.

However, as long as the empirical distribution (e.g., the histogram) is close to the target distribution, we are not too concerned about independence and the sampling algorithm does not need to throw away any samples, since the order of the samples is not considered. Because the samples at the start states don't satisfy the stationary distribution, it is a good idea to discard the samples produced by the chain at the beginning.

Strong dependence between samples that are close to each other in time could be problematic. For example, suppose we get N samples from a chain whose samples are strongly dependent during intervals of duration not much smaller than N. While each of the N samples may individually have the target distribution, due to strong dependence they all may be from the same area of the probability space and thus the empirical distribution may not look like the target distribution, necessitating obtaining a larger number of samples. This problem can be caused by choosing a poor jump distribution as discussed next.

The Jump distribution. The jumps should be neither too small nor too large!

- When the jumps are large, a large number of proposals will be rejected (we'll stay in the current state) because it is likely that with a large jump, we'll end up with a low probability proposal. In this case, strong dependence manifests as many samples being likely to be equal. An example is shown in Figure 15.2a, where most of the proposals are rejected, resulting in a small number of distinct samples.
- If the jumps are too small, the sampling process is similar to a random walk, because most proposals are accepted but we move only a small step. This means that the MC does not explore the probability space efficiently, again necessitating a large number of samples. An example is given in Figure 15.2b. To see why random walk behavior is not good, consider a random walk with step size ε. How far from the starting point will we be after T steps? For the random walk, let X_i be the movement in one step:

$$X_i = \begin{cases} \varepsilon, & \text{with } p = \frac{1}{2}; \\ -\varepsilon, & \text{with } p = \frac{1}{2}. \end{cases}$$

After T steps, the expected distance $L = \mathbb{E}\left[\left|\sum_{i=1}^{T} X_i\right|\right]$ is difficult to find. But we can approximate

the distance as

$$L^2 \simeq \mathbb{E}\left[(\sum_{i=1}^{T} X_i)^2\right] = T\varepsilon^2 \text{ (exercise).}$$

In conclusion, after T steps, we will be approximately at distance $\sqrt{T}\varepsilon$, which is a case of diminishing returns, and not very efficient. In other words, we need $\frac{L^2}{\varepsilon^2}$ steps to move distance L. In the context of MCMC, this means if the probability space has a dimension in which there is a high probability region with length L, we need to run the chain for *at least* $\frac{L^2}{\varepsilon^2}$ steps.

15.4 Gibbs Sampling

At each iteration of the Metropolis algorithm, all the components of $\boldsymbol{\theta}$ are updated at the same time. In Gibbs sampling, for $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_d)$, at each iteration, components are updated one-by-one as

$$\theta_j^t \sim p(\theta_j | \theta_1^t, \cdots, \theta_{(j-1)}^t, \theta_{(j+1)}^{(t-1)}, \cdots, \theta_d^{t-1}), \qquad \text{for } j = 1, \ldots, d.$$

Gibbs sampling may be simpler and more efficient than Metropolis sampling if the joint distribution is too complicated but we can easily sample from the conditional distributions. The components do not need to be one-dimensional necessarily; we can group several dimensions and update each the dimensions in each group simultaneously.

Example 15.5. Suppose $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and the observation $\boldsymbol{y} = (y_1, y_2)$ are related by the likelihood

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} | \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right),$$

with the prior $p(\boldsymbol{\theta}) \propto 1$. The posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$ is:

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} | \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right)$$

We can use Gibbs sampling to produce samples for $\boldsymbol{\theta}|\boldsymbol{y}$. The following fact is of use:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \right) \Rightarrow x_1 | x_2 \sim \mathcal{N}\left(\mu_1 + \frac{\rho\sigma_1}{\sigma_2}(x_2 - \mu_2), (1-\rho^2)\sigma_1^2 \right)$$

Then, in the t-th iteration, the θ_1^t is sampled by

$$\theta_1^t | \theta_2^{t-1} \sim \mathcal{N}\big(y_1 + \rho(\theta_2^{t-1} - y_2), (1-\rho^2)\big).$$

Similarly, the θ_2^t can be updated by

$$\theta_2^t | \theta_1^t \sim \mathcal{N}\big(y_2 + \rho(\theta_1^t - y_1), (1-\rho^2)\big).$$

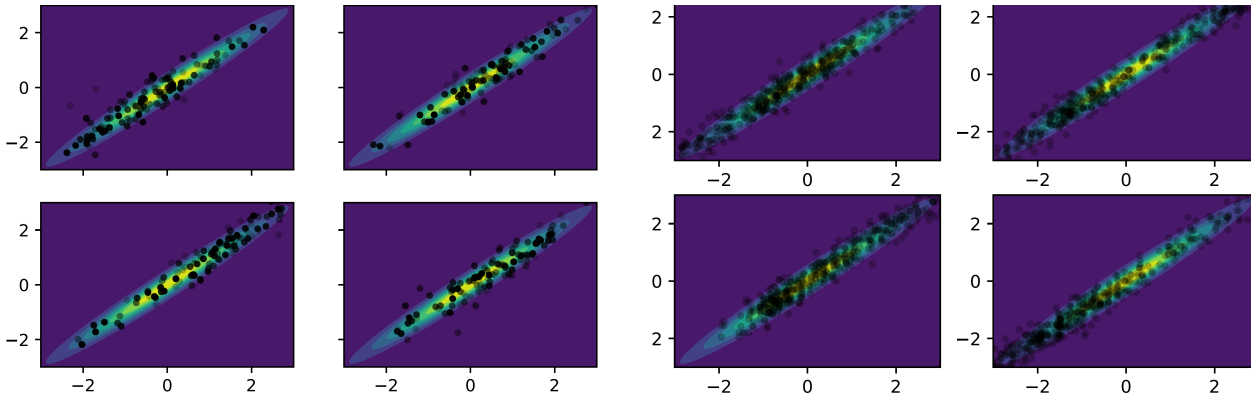
So we produce a new sample using 1-D distributions. △

Stationary distribution. We prove that Gibbs sampling (with a caveat) satisfies the DBP with distribution $p(\boldsymbol{\theta})$.

Suppose we are in state $\boldsymbol{\theta}$ and we update the jth component to get $\boldsymbol{\theta}'$. We have

$$\theta_j' \sim p(\theta_j' | \boldsymbol{\theta}_{-j}),$$

where $\boldsymbol{\theta}_{-j} = (\theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_d)$. Furthermore, the $\boldsymbol{\theta}'_{-j} = \boldsymbol{\theta}_{-j}$.



(a) Four runs of the Metropolis algorithm.

(b) Four runs of the Gibbs algorithm.

Figure 15.3: Metropolis and Gibbs sampling for highly correlated dimensions. Many proposals for Metropolis are rejected.

To prove DBP for this step, we need to prove $p(\boldsymbol{\theta}) \Pr(\boldsymbol{\theta} \to \boldsymbol{\theta}') = p(\boldsymbol{\theta}') \Pr(\boldsymbol{\theta}' \to \boldsymbol{\theta})$, which holds since

$$p(\boldsymbol{\theta}) \Pr(\boldsymbol{\theta} \to \boldsymbol{\theta}') = p(\boldsymbol{\theta}) p(\theta_j' | \boldsymbol{\theta}_{-j}) = p(\boldsymbol{\theta}_{-j}) p(\theta_j | \boldsymbol{\theta}_{-j}) p(\theta_j' | \boldsymbol{\theta}_{-j}),$$
$$p(\boldsymbol{\theta}') \Pr(\boldsymbol{\theta}' \to \boldsymbol{\theta}) = p(\boldsymbol{\theta}') p(\theta_j | \boldsymbol{\theta}'_{-j}) = p(\boldsymbol{\theta}_{-j}) p(\theta_j' | \boldsymbol{\theta}_{-j}) p(\theta_j | \boldsymbol{\theta}_{-j}).$$

Since the DBP holds for each sub-iteration. We can use this observation to prove that Gibbs sampling works if we choose a component to update at random to produce a new sample or if we update all components in a random order and after a full cycle produce a new sample. Updating the components in a predetermined order works in practice.

Gibbs sampling can be viewed as a special case of Metropolis-Hastings in which the proposal is always accepted and where we don't need to design a jump distribution. Gibbs can use the current state to provide better proposals. An example is shown in Figure 15.3. Here, the dimensions are highly correlated, with most of the probability concentrated in a narrow region. Because of this, many of the Metropolis proposals are rejected. Gibbs, which produces samples based on the conditional distribution given the current state, dose not suffer from this.

Note that θ_j may be independent from some dimensions of $\boldsymbol{\theta}_{-j}$ given others. In particular, if $\boldsymbol{\theta}$ denotes the nodes in a graphical model, given its Markov blanket, θ_j is independent of other elements of $\boldsymbol{\theta}_{-j}$.

15.5 Hamiltonian Monte Carlo **

One problem with the Metropolis algorithm is that, in certain situations, the proposed $\boldsymbol{\theta}'$ by the jump distribution may be rejected too often because $p(\boldsymbol{\theta}')$ is much smaller than $p(\boldsymbol{\theta}^{t-1})$, in which case we will let $\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1}$. While the stationary distribution is still $p(\boldsymbol{\theta})$, too many rejection means that it will take a long time to get a sample whose empirical distribution is close to the true distribution.

Let us write our target distribution $p(\boldsymbol{\theta})$ as

$$p(\boldsymbol{\theta}) \propto e^{-E(\boldsymbol{\theta})}$$

and suppose that we can also compute $\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$. Note that as $E(\boldsymbol{\theta})$ decreases, the probability increases.

Can we use the fact that we know the gradient to increase the chance of proposals being accepted? At first glance it may seem that we could let $\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} - \epsilon \nabla E(\boldsymbol{\theta})$, similar to gradient descent. But this is a deterministic path rather than a probabilistic MC.

A bit of (questionable) physics. Instead, we use an idea from Hamiltonian Mechanics. We can think of $\boldsymbol{\theta}$ as location and of $E(\boldsymbol{\theta})$ as potential energy. Note that lower potential has a higher probability (a river flows down the valley). Now let us also include momentum (speed) $\boldsymbol{\phi}$, which has the same number of dimensions as $\boldsymbol{\theta}$, in our formulation and define the total energy as

$$H(\boldsymbol{\theta}, \boldsymbol{\phi}) = E(\boldsymbol{\theta}) + K(\boldsymbol{\phi}),$$

where $K(\boldsymbol{\phi})$ is the Kinetic energy

$$K(\boldsymbol{\phi}) = \frac{1}{2}\boldsymbol{\phi}^T\boldsymbol{\phi}.$$

With this physical viewpoint, Hamilton's equations describing the motion of an object with position $\boldsymbol{\theta}$ and momentum $\boldsymbol{\phi}$ are

$$\dot{\boldsymbol{\theta}} = \frac{\partial \boldsymbol{\theta}}{\partial t} = \boldsymbol{\phi}$$
$$\dot{\boldsymbol{\phi}} = \frac{\partial \boldsymbol{\phi}}{\partial t} = -\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$$

(A more familiar form of these equations are obtained by representing position with $\boldsymbol{x}$ and speed with $\boldsymbol{v}$. Then, $\dot{\boldsymbol{x}} = \boldsymbol{v}, \dot{\boldsymbol{v}} = -\nabla_{\boldsymbol{x}} E(\boldsymbol{x})$.) It can then be shown that H, the total energy, stays constant in time.

Back to Sampling. Instead of sampling from $p(\boldsymbol{\theta})$, let us define and sample from

$$p(\boldsymbol{\theta}, \boldsymbol{\phi}) \propto e^{-H(\boldsymbol{\theta},\boldsymbol{\phi})} = e^{-E(\boldsymbol{\theta})} e^{-K(\boldsymbol{\phi})},$$

where $K(\boldsymbol{\phi}) = \frac{1}{2}\boldsymbol{\phi}^T\boldsymbol{\phi}$. We will then discard the $\boldsymbol{\phi}$ component of the samples.

The Hamiltonian Monte Carlo Algorithm is as follows:

1. Randomly choose $\boldsymbol{\theta}^0$ from the domain and choose $\boldsymbol{\phi}^0$ arbitrarily.
2. For $t = 1, 2, ...,$ do
   (a) Pick a random momentum $\boldsymbol{\phi}'$ according to the distribution $p(\boldsymbol{\phi}) \propto e^{-K(\boldsymbol{\phi})}$.
   (b) Starting from $(\boldsymbol{\theta}^{t-1}, \boldsymbol{\phi}')$, simulate the dynamic system for a certain amount of time according to

$$\dot{\boldsymbol{\theta}} = \boldsymbol{\phi},$$
$$\dot{\boldsymbol{\phi}} = -\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}).$$

   The final values of $(\boldsymbol{\theta}, \boldsymbol{\phi})$ are the new sample, $(\boldsymbol{\theta}^t, \boldsymbol{\phi}^t)$.

It can be shown this process leads to a Markov chain whose stationary distribution is $p(\boldsymbol{\theta}, \boldsymbol{\phi})$. This hinges on step (a) being reversible and step (b) keeping the Hamiltonian and thus the probability constant.

In practice however, we cannot have a perfect simulation. So instead of step (b) above, we perform the following:
(2.b)' For $i = 1, 2, \ldots, L$, perform the following steps, called leapfrog updates:

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \frac{1}{2}\epsilon \nabla E(\boldsymbol{\theta})$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \boldsymbol{\phi}$$
$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \frac{1}{2}\epsilon \nabla E(\boldsymbol{\theta})$$

Let the final $(\boldsymbol{\theta}, \boldsymbol{\phi})$ be denoted by $(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$. If our simulation is perfect, then this can be accepted as the new state. But because $\epsilon > 0$, we have to perform an accept/reject check similar to Metropolis. That is, we let

$$r = \frac{e^{-H(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*)}}{e^{-H(\boldsymbol{\theta}^{t-1}, \boldsymbol{\phi}^{t-1})}}.$$

If $r \geq 1$, we let $(\boldsymbol{\theta}^t, \boldsymbol{\phi}^t) = (\boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$. If $r \leq 1$, then we let $(\boldsymbol{\theta}^t, \boldsymbol{\phi}^t) = (\boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$ with probability r and with probability $1 - r$, we let $(\boldsymbol{\theta}^t, \boldsymbol{\phi}^t) = (\boldsymbol{\theta}^{t-1}, \boldsymbol{\phi}^{t-1})$.

If ϵ is too large, our simulation will be too rough, leading to many rejections. In this case, we decrease ϵ and increase L. On the other hand, if nearly all proposals are accepted, it may be a sign of being too conservative and not exploring the state space as fast as we can, in which case we can be more efficient by increasing ϵ and decreasing L.