Chapter 1

Monte Carlo

These notes provide a very basic introduction to Monte Carlo methods. For a brief history on how John von Neumann and Stanisław Ulam came up with Monte Carlo methods including "inverse transform" and "rejection" sampling see Roger [1987].

1.1 Motivation

Here Monte Carlo methods refer to algorithms that can be employed to approximate probability distributions. We have to start somewhere, so let's start with a random variable X, that is a measurable function from a set Ω, of elements written ω, to a measurable space $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$. In these notes $\mathbb{X}$ is often going to be countable, or to some subsets of $\mathbb{R}^d$ for some dimension d. There are many ways of describing a random variable X.

- We can describe how to compute X based on other random variables, for instance we could say that X is equal to $-\log U$ where U is a uniform variable in the interval $[0, 1]$.
- We can specify the probability density function of X, for instance $x \mapsto \exp(-x)\mathbb{1}(x \geq 0)$ on $\mathbb{R}$.
- We can provide the cumulative distribution function of X, for instance $x \mapsto (1-\exp(-x))\mathbb{1}(x \geq 0)$ on $\mathbb{R}$, or its moment-generating or characteristic function.
- We can give a set of properties that only the variable X satisfies. For instance: the support of X is $\mathbb{R}_+$ and $\mathbb{E}[X] = 1$ and $\mathbb{P}(X > t+s|X > s) = \mathbb{P}(X > t)$ for all $t, s \in \mathbb{R}_+$.
- We can read out loud the Wikipedia page associated with the variable X.

Above the function $\mathbb{1}(\cdot)$ equals 1 when the argument holds and 0 otherwise. Also $\mathbb{R}_+$ refers to the non-negative reals, $\mathbb{E}$ and $\mathbb{P}$ are symbols for expectation and probability. The "support" of a random variable X is the smallest closed set $R_X \subset \mathbb{X}$ such that $\mathbb{P}(X \in R_X) = 1$. The examples mentioned above all refer to the same Exponential(1) random variable. The first description tells us how to generate realizations of these variables on a computer... that is, if we have access to uniform random variables. Let's get that out of the way: we assume throughout that we have access to realizations of uniform random variables on $[0, 1]$ [e.g. L'Ecuyer, 2012].

The capacity to generate (arbitrarily many) realizations of a random variable allows the empirical study of many of its properties. For instance, a kernel density estimator provides an approximation of a probability density function based on samples. With empirical averages we can approximate the cumulative distribution function or any marginal, or any other probability or vector of probabilities associated with X and transformations of X. Instead of having to specify exactly which aspect of the variable X is of interest, we will often generically introduce some "test" function $h : \mathbb{X} \to \mathbb{R}$, and consider the expectation $\mathbb{E}[h(X)]$ as the quantity of interest. For the most part, in these notes we will view the goal of Monte Carlo methods to be the approximation of such expectations.

For some random variables, such as Binomials, Exponentials, Normals, different ways of generating realizations are already available. Tautologically, if we describe X by providing a recipe for the generation of X, as in the first

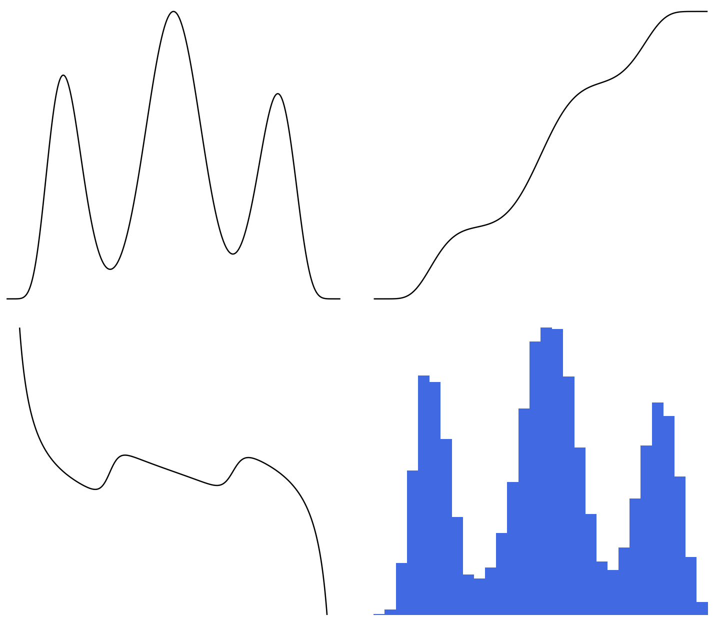

Figure 1.1: Representations of the same probability: density, cumulative distribution function, derivative of log-density, and a way of producing samples.

point in the above list, then... we know how to generate X. For instance, if we introduce a Geometric(0.5) variable K, and then $G_1, \ldots, G_K$ independent Gamma($K, K/2$) variables, and then define Z as $\min(K, \max_k G_k)$, and then we define a variable $X = Z \times K$, then we know exactly how to sample that variable X, but it does not have a name nor a wikipedia page (yet).

For most random variables that we would encounter in our scientific adventures, we would have some information about them: random variables do not appear out of thin air. But it might not instantly clear how to generate or sample them. This is where Monte Carlo methods come in: they provide recipes, taking as input some knowlege about a random variable X, and providing implementable procedures to obtain samples. Monte Carlo methods can be categorized according to the type of knowledge available about X. Here are some examples of information that we might have about random variables.

- We might know how to evaluate the density of X point-wise "up to a multiplicative constant". This means that, denoting the density by $f_X : \mathbb{X} \to \mathbb{R}_+$, we might be able to numerically evaluate a function $\tilde{f}_X$ such that for all $x \in \mathbb{X}$, $f_X(x) = \tilde{f}_X(x)/Z_X$ where $Z_X = \int \tilde{f}_X(x)dx$ is called the normalizing constant. We write $f_X(x) \propto \tilde{f}_X(x)$ to mean "equality up to a multiplicative constant". For instance, consider a uniform random variable over some set A. Then $f_X(x) = \mathbb{1}(x \in A)/\text{Vol}(A)$, so we can define $\tilde{f}_X(x) = \mathbb{1}(x \in A)$. We might be able to determine whether any

$x \in \mathbb{X}$ belongs to a certain set A, and thus how to evaluate $\tilde{f}_X$ point-wise, without knowing the volume of A. Posterior distributions in Bayesian inference provide another common example where point-wise evaluations of density functions are often available.

- If the space is continuous and the density f_X differentiable, we might further know how to compute its derivatives at any $x \in \mathbb{X}$, either numerically or analytically. We can also imagine a case where we would know how to evaluate the derivatives $\nabla \log f_X(x)$ for all x, but we could not evaluate f_X itself, even up to a constant. Although, no such cases come to my mind at the moment.

- We might not know how to generate X on its own, but perhaps we know how to generate X given Y and Y given X, for some other variable Y.

- We might know that the density function f_X decomposes in a certain way. For instance into a product of terms with properties to be exploited.

- We might know of a certain $x_0 \in \mathbb{X}$ at which f_X attains its maximal value, and we might know what that maximal value is.

- We might not know how to evaluate f_X or $\tilde{f}_X$ at any $x \in \mathbb{X}$ but, given any x, we might know how to generate an almost surely non-negative random variable with expectation equal to $f_X(x)$.

Monte Carlo methods refer to algorithms that exploit some information such as in the above list, and use it to obtain realizations of X. The wide variety of Monte Carlo algorithms comes in part from the variety of the types of available information. Appreciating the diversity of settings in which Monte Carlo methods are employed comes with time, and a lot of reading, but is highly recommended as it helps understanding why there is not a single algorithm that dominates all others.

As a motivating example, two realizations of a random variable are shown in Table 1.1. After reading these notes, you will be able to produce such tables too!

1.23	0.75

Table 1.1: Two realizations of a random variable (to two decimal places).

1.2 Sampling exactly

The most pleasing class of sampling method generates independent values that satisfy exactly the specification of the random variable X of interest. In some circumstances “inverse transform sampling” and “rejection sampling”, described below, achieve that goal. We recall these methods briefly. A very impressive book describing how these techniques and many others can be combined to sample from a variety of probability distributions is Devroye [2006]. Later on

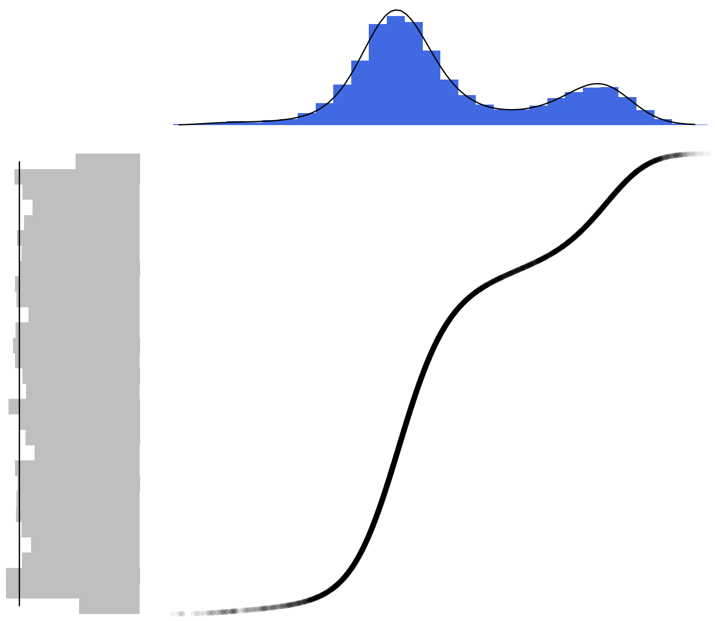

Figure 1.2: Gogo gadget inverse transform!

in these notes, we will describe a wonderful algorithm, called “coupling from the past”, which also produces exact samples from some non-trivial probability distributions. Methods of that type are sometimes called “perfect samplers”, “exact samplers”, “exact simulation”. Mention perfect sampling to any veteran of the Monte Carlo world, and you will see glittering eyes and drooling mouths, and you will hear tales of hopes and despair.

1.2.1 Inverse transform sampling

The inversion method [Chapter 2, Section 2, Devroye, 2006] allows to sample a variable X defined on $\mathbb{R}$, using draws from a uniform distribution and the inverse cdf function of X. Recall that the cumulative distribution function is $F : x \mapsto \mathbb{P}(X \leq x)$. Define the generalized inverse

$$F^{-}(u) = \inf\{x \in \mathbb{R}; F(x) \geq u\}, \tag{1.2.1}$$

also known as the “quantile function”. Then draw U uniformly on $[0, 1]$, and compute $X = F^{-}(U)$. Tadam! The generated X has a distribution with cumulative distribution function F. See Figure 1.2?

A limitation of the inversion method is that it works only for univariate distributions, with a quantile function that can be evaluated point-wise. If a distribution is specified via its probability density function, then one needs to perform numerical integration to obtain the cumulative distribution function,

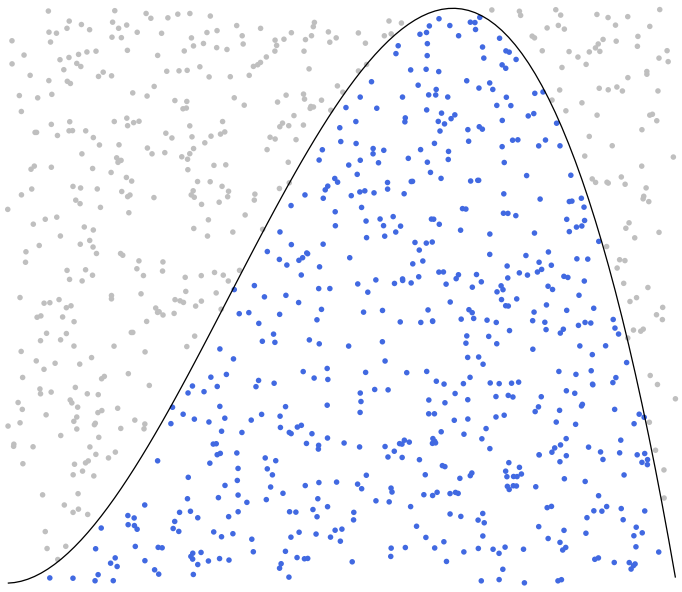

Figure 1.3: Some points are accepted, some points are rejected.

and numerical inversion, in order to obtain $F^{-}(u)$ for a given $u \in (0,1)$. In the literature, there are various other sampling methods that involves sampling from a simple distribution, and then applying a not-so-simple transformation to it; some of them are related to transport maps and will be mentioned in the next chapter.

1.2.2 Rejection sampling

The basic idea of rejection sampling [Chapter 2, Section 3, Devroye, 2006] is to sample from a proposal distribution q, different from the target π, and then to correct for the discrepancy by accepting some draws and rejecting others. Assume that $\pi\left(x\right)/q(x) \leq M$ for all $x \in \mathbb{X}$ and for some $M < \infty$... and assume that we know the value of M. The algorithm proceeds as follows.

1. Draw $X \sim q$.
2. With probability
$$\alpha = \frac{\pi\left(X\right)}{Mq\left(X\right)}, \tag{1.2.2}$$
return X, otherwise go back to step 1.

The distribution of the returned ("accepted") samples is exactly π. The algorithm also works if $\pi(x)$ and $q(x)$ are only density evaluations up to an unknown normalizing constant, but it must hold that $\pi(x)/q(x) \leq M$ for all $x \in \mathbb{X}$, i.e. M must be an upper bound on the ratio π/q being evaluated in the algorithm.

The algorithm takes a random time to complete. For such algorithms, the compute time becomes an important aspect to consider. Each step involves sampling from q, and evaluating π and q at a realization. These might be cheap or expensive operations, depending on the setting. We can work out that the number of steps before completion is a Geometric variable, with parameter $1/M$. In particular its expectation is equal to M. Thus it is better to choose a small value for M, but M must be at least $\sup_x \pi(x)/q(x)$ for the algorithm to return draws from π. We can see that if q is close to π, and if M is close to one, then draws from q are then likely to be accepted and the algorithm completes rapidly. On the other hand if not much is known about π, it might be difficult to choose q and M so that the algorithm completes in a reasonable amount of time. Note also that the number of trials before acceptance is independent of the accepted draw.

Figure 1.3 is an illustration of the rejection sampling algorithm. There are many interesting cases and variations of the rejection sampler, for instance see Wild and Gilks [1993], Devroye [2012] for log-concave distributions, Görür and Teh [2011] for a wide class of univariate distributions, or Deligiannidis et al. [2020] in the context of state space models.

1.3 Sampling and weighting

When applicable the above methods produce realizations of a random variable X of interest, or equivalently of a target distribution π. Typically users would then execute these algorithms many times independently, so as to obtain an independent and identically distributed "i.i.d." sample $X_1, \ldots, X_N$. In cases where sampling exactly appears to be difficult, Monte Carlo methods do not simply give up. They pull themselves together and find alternative solutions, and so should we. We next consider ways of obtaining approximations of π of different kinds, starting with importance sampling, that gives weighted samples approximating a target distribution. Good references include Robert and Casella [Chapter 3, 1999] and Owen [Chapter 9, 2013].

Consider again the setting of rejection sampling, where one knows how to sample from some distribution q on $\mathbb{X}$, and how to evaluate π and q at least up to a multiplicative constant. This time, we won't assume that $w : x \mapsto \pi(x)/q(x)$ is upper-bounded and that we know the bound. We will assign a weight $w(X_n) = \pi(X_n)/q(X_n)$, called importance weight, to each one of N independent draws $X_1, \ldots, X_N$ from q. Figure 1.4 represents a proposed sample, log-weights and the weighted sample obtained with importance sampling. Then the weighted samples approximate π, in the sense that for sufficiently regular functions $h : \mathbb{X} \mapsto \mathbb{R}$,

$$\mathrm{IS}(N, q, h) := \frac{N^{-1}\sum_{n=1}^{N} w(X_n)h(X_n)}{N^{-1}\sum_{n=1}^{N} w(X_n)} \xrightarrow[N\to\infty]{\mathbb{P}} \int h(x)\pi(x)dx. \tag{1.3.1}$$

Indeed one can apply the law of large numbers to both averages on the left-hand

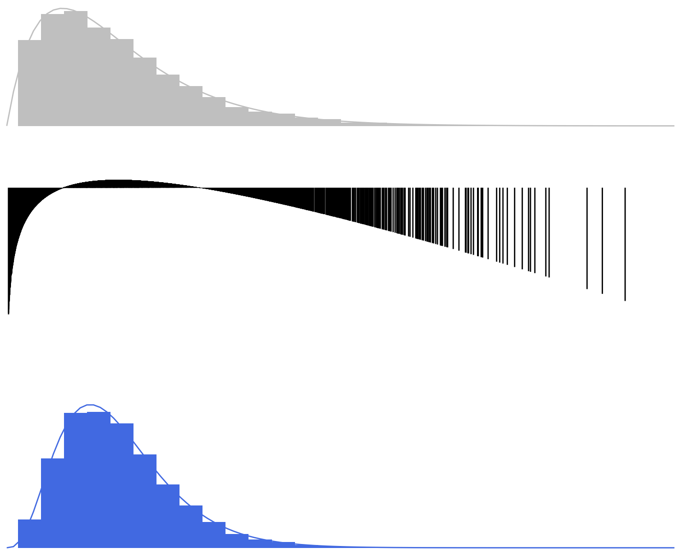

Figure 1.4: Proposed samples, weights, weighted samples.

side, and the continuous mapping theorem. The conditions on h, π and q are the finiteness of $\mathbb{E}_q[w(X)|h(X)|]$ and of $\mathbb{E}_q[w(X)]$. In particular the weight function w does not need to be upper-bounded. We can further obtain a central limit theorem, of the sort $\sqrt{N}(\mathrm{IS}(N,q,h) - \mathbb{E}[h(X)]) \to \mathcal{N}(0, v(q,h))$ for some $v(q,h) > 0$, under further moment conditions. By estimating the variance $v(q,h)$ using the same weighted samples, it is possible to construct asymptotically correct confidence intervals for $\mathbb{E}[h(X)]$, as $N \to \infty$.

To sum up, importance sampling works under weaker conditions than rejection sampling (no need to know M!) but only provides approximations in the form of weighted samples. Robert and Casella [Section 3.3.3, 1999] provides a more detailed comparison. Note that the compute time of importance sampling is deterministic in the number of calls to a sampler for q and evaluators for $q(x)$ and $\pi(x)$.

It might be worth noting that the following does not yield a sample from π: compute $W_n = w(X_n)/\sum_{m=1}^{N} w(X_m)$, sample $A \in \{1,\ldots,N\}$ such that $\mathbb{P}(A = n) = W_n$, and return X_A. The marginal distribution of the returned X_A will get closer to π as $N \to \infty$ but it is not exactly π for any fixed N; see Deligiannidis et al. [2020].

Rejection and importance sampling both require the specification of a distribution q on the full space $\mathbb{X}$, that is meant to be a close approximation to π. This prompts the question: what does it mean for a distribution q to be a good approximation of π? Is there a notion of "distance" between distributions, which would meaningfully quantify how well a proposal distribution q operates in a rejection sampler, or in an importance sampler? Some answers can be found in Agapiou et al. [2017], Chatterjee and Diaconis [2018], where the roles of the χ^2-divergence, and of the Kullback–Leibler divergence respectively, is made very explicit. These divergences are also related to empirical diagnostics of the performance of importance sampling, such as the "effective sample size" or the "entropy" computed from the weights. The effective sample size, for example, is related to the χ^2-divergence

$$\frac{1}{N}\mathrm{ESS}_N := \frac{1}{N}\frac{(\sum_{n=1}^N w(X_n))^2}{\sum_{n=1}^N w(X_n)^2} \xrightarrow[N\to\infty]{\mathbb{P}} \frac{1}{1+\chi^2(\pi,q)}, \tag{1.3.2}$$
$$\text{where} \quad \chi^2(\pi,q) = \int \frac{(\pi(x) - q(x))^2}{q(x)} dx.$$

The quantity ESS_N is between 1 and N and is informally used as a number of "equivalent" exact samples [Kong, 1992]. It is a measure of the relative variability among the weights. In many cases where importance sampling fails to provide accurate results, the effective sample size will be close to its minimal value, allowing the user to see that the method does not work.

We will describe various distances between probability distances in the next chapter, as this is a topic closely related to couplings.

1.4 Markov chain Monte Carlo

In many challenging settings, coming up with a global approximation q of π, for which we can evaluate q point-wise, is infeasible. In such cases the preferred methods operate without global information, and instead perform iterative modifications of a running state denoted by X_t at step t. These methods are broadly called Markov chain Monte Carlo.

1.4.1 Markov chains to approximate distributions

A state X_0 is initialized from some arbitrary distribution π_0 at time $t = 0$. Here π_0 is really meant to be describing arbitrary starting points, and does not have to be a good approximation of π, unlike q in importance sampling. An MCMC algorithm generates X_t given $X_0, \ldots, X_{t-1}$ by modifying the previous state X_{t-1}, possibly using new independent random numbers, but not using the states $X_0, \ldots, X_{t-2}$. This makes this sequence $(X_t)_{t\geq 0}$ a Markov chain. The distribution of X_t given X_{t-1} will be assumed to depend on X_{t-1} but not on the time index, and is denoted by $K(X_{t-1} \to \cdot)$ and called "transition kernel". Textbooks on Markov chains include Meyn and Tweedie [1993], Douc et al. [2018]. A short review on some of the relevant Markov chain theory is Nummelin [2002].

MCMC algorithms produce a sequence $(X_0, \ldots, X_T)$ that approximates π in the sense that, under appropriate conditions and for sufficiently nice functions

$h : \mathbb{X} \mapsto \mathbb{R}$,

$$\mathrm{MC}(T, K, h) := (T+1)^{-1} \sum_{t=0}^{T} h(X_t) \xrightarrow[T\to\infty]{\mathbb{P}} \mathbb{E}[h(X)]. \qquad (1.4.1)$$

This is a law of large numbers, or "ergodic theorem", for Markov chains. The variable $\mathrm{MC}(T, K, h)$ is the proposed estimator for $\mathbb{E}[h(X)]$. Compared to the previously mentioned limiting results, the difference is in the correlations between successive samples in $(X_t)_{t\geq 0}$. Thus it is common, when running MCMC algorithms, to investigate the correlations between successive states, for example using autocorrelograms as in Figure 1.7 (right) for instance. Because of the convergence in (1.4.1), which we expect to hold in the MCMC setting, the distribution π is interchangeably called the "target", the "stationary" distribution, or the "limiting" distribution.

In the estimator $\mathrm{MC}(T, K, h)$ there is a choice of Markov kernel K, and some examples will be given below. For any target distribution π there will usually be many potential choices of Markov kernels K, and a vast literature investigates the performance and design of Markov kernels. But in any case, there is also a choice of the number of iterations T. This is a crucial choice: there is no use of MCMC without a choice of the number of iterations. The couplings described in the later chapters will provide insights regarding the number of iterations necessary for the Markov chain (X_t) to be "close" to its stationary and limiting distribution π, in some precise sense. The basic way of choosing that number of iterations is to plot various components of the chain along the iteration t, as in Figure 1.5, obtained from multiple independent chains started at different states, and to choose a number of iterations "b" that appears to be large enough for X_b to be approximately from π. For non-trivial Markov chains, this is an arduous job. This b is called the "burn-in", or "warm-up" part of the chain. Upon choosing it, one would discard the $b-1$ first iterations and compute an estimator from the later iterations, such as $(T-b+1)^{-1}\sum_{t=b}^{T} h(X_t)$. Asymptotically in T, this makes no difference compared to $\mathrm{MC}(T, K, h)$ in (1.4.1), but non-asymptotically and thus in practice, the choice of burn-in is important.

Beyond the above law of large numbers, central limit theorems can be obtained under further conditions, and by estimating the associated asymptotic variance, one can obtain asymptotically valid confidence intervals for $\mathbb{E}[h(X)]$, as $T \to \infty$. Non-asymptotic confidence intervals for MCMC algorithms are much harder to compute and hardly ever used, to the best of my knowledge. We next review some classical MCMC methods, to provide concrete examples of transition kernels $K(X_{t-1} \to \cdot)$.

1.4.2 Metropolis–Hastings

In this algorithm (see e.g. Hastings [1970] or Hitchcock [2003] for a history), assuming the chain reached state X_{t-1} at step $t-1$, a new state $X^\star$ is sampled from a probability distribution $q(X_{t-1} \to \cdot)$, that can depend on X_{t-1}. It is called the proposal distribution or proposal kernel. A typical choice when $\mathbb{X} = \mathbb{R}^d$

is to draw $X^\star$ from $\mathcal{N}(X_{t-1}, \Sigma)$, a Normal distribution centered at X_{t-1} and with user-chosen covariance matrix Σ. Then the state X_t is set to either the proposal $X^\star$ or the previous state X_{t-1}, according to a coin flip with probability

$$\alpha_{\mathrm{MH}}\left(X_{t-1}, X^\star\right) = \min\left(1, \frac{\pi\left(X^\star\right) q\left(X^\star \to X_{t-1}\right)}{\pi\left(X_{t-1}\right) q\left(X_{t-1} \to X^\star\right)}\right). \tag{1.4.2}$$

The above is called the Metropolis–Hastings (MH) acceptance probability. Compared to a random walk where each proposed state would be accepted as the next state, the algorithm here accepts or not according to the ratio of target density evaluations between current and proposed states. Billera and Diaconis [2001] interprets this modification as a certain projection of the proposal kernel to the space of π-invariant Markov kernels. The generated Markov chain has a transition kernel K given by

$$\begin{aligned}\forall x \in \mathbb{X} \quad K\left(x \to A\right) = & \int_A \alpha_{\mathrm{MH}}\left(x, x'\right) q\left(x \to dx'\right) \\ & + \left(1 - \int \alpha_{\mathrm{MH}}\left(x, x''\right) q(x \to dx'')\right) \mathbb{1}\left(x \in A\right). \end{aligned} \tag{1.4.3}$$

The transition kernel K is π-invariant or "leaves π invariant". That is, if we start with $X_{t-1} \sim \pi$, and sample $X_t \sim K(X_{t-1} \to \cdot)$ then marginally, $X_t \sim \pi$. In other words π is a fixed point of the operator that applies one step of this Markov transition kernel to a probability measure. In practice we start X_0 from π_0 and repeatedly apply Markov transitions using the kernel K. The convergence of the resulting (X_t) to π is thus not obvious [e.g. Jarner and Hansen, 2000], but perhaps unsurprising for readers familiar with fixed-point iterations. We will go back to convergence of Markov chains in a later chapter, as this is a topic where couplings have been extensively used.

The choice $q(X_{t-1} \to \cdot) \equiv \mathcal{N}(X_{t-1}, \Sigma)$ leads to the "random walk Metropolis–Hastings" algorithm. A common variant, applicable when the target is defined on a subset of $\mathbb{R}^d$ and $\nabla \log \pi(x)$ can be evaluated for all x, is called "Metropolis-adjusted Langevin algorithm" (MALA). It has been around for a long time; Besag [2001] attributes it to a comment by the same author published in the discussion of Grenander and Miller [1994]. The algorithm is a discretization of the continuous-time Langevin process, which converges marginally to π, combined with an MH acceptance step; see e.g. Robert and Casella [Section 6.5.2, 1999]. It corresponds to defining $q(X_{t-1} \to \cdot) \equiv \mathcal{N}(X_{t-1} + (h\Sigma/2)\nabla \log \pi(X_{t-1}), h\Sigma)$. Here h is a positive scalar representing a step size, stemming from the discretization of the Langevin process, and Σ is a covariance matrix. We see that the proposal is now centered around $X_{t-1} + (h\Sigma/2)\nabla \log \pi(X_{t-1})$ instead of around X_{t-1} as in random walk proposals. In other words, the proposal "follows" the gradient of the target density, which is expected to be a good idea wherever following the gradient is a good idea; this includes notably some large-dimensional, smooth and mostly unimodal probability distributions. The analysis of this algorithm, and various variants thereof, has been the topic of numerous articles, for instance Dalalyan [2017], Durmus and Moulines [2015], Dwivedi et al. [2019] in recent years. The sensitivity of the choice of step size is discussed in Livingstone and Zanella [2019]. Later chapters will discuss some associated coupling strategies.

1.4.3 Gibbs sampling

Another essential idea in MCMC is referred to as Gibbs sampling. Let us assume that X has d components, denoted by $X[1],\ldots,X[d]$. The target distribution π of X has "full conditional" distributions of π, denoted by $\pi(dx[i]|x[\backslash i])$ for all $x\in\mathbb{X}$ and all $i\in[d]=\{1,\ldots,d\}$. Here $x[\backslash i]$ denotes "all the components of x except the i-th one". The idea of a Gibbs sampler is to repeatedly update some component of the state given the other current components, by sampling $X[i]$ from a Markov kernel leaving full conditional distribution $\pi(dx[i]|X[\backslash i])$ invariant. In some cases we can sample $X[i]$ from $\pi(dx[i]|X[\backslash i])$ exactly, otherwise we could employ a Metropolis–Hastings mechanism with target $\pi(dx[i]|X[\backslash i])$. Note that the idea of Gibbs sampling is already present in the early works on MCMC, for instance it is discussed in Hastings [Section 2.4, 1970]; it is likely as old as the idea of MCMC itself, and is called Glauber dynamics in parts of physics following e.g. Glauber [1963]. In fact Hastings [1970] credits earlier articles such as Ehrman et al. [1960]; early works include also Besag [1974], Geman and Geman [1984] who gave the idea the name of "Gibbs sampling", and Gelfand and Smith [1990]; see Robert and Casella [2011] for a history.

We can the following procedure as an iteration of a Gibbs sampler, sampling state X_t given a state X_{t-1} by updating one component. As $t\to\infty$ all the components should get updated infinitely many times.

- Sample I uniformly in $\{1,\ldots,d\}$,
  or set I deterministically as a function of t by d.
- Sample $X_t[i]\sim\pi(dx[i]|X_{t-1}[\backslash i])$ and set $X_t[\backslash i]=X_{t-1}[\backslash i]$.

It turns out that the vast majority of MCMC algorithms can be seen as composition of Gibbs and Metropolis–Hastings steps with various choices of proposal distributions, on appropriately defined state spaces. We can think for instance of slice samplers [Neal, 2003], or of Hamiltonian Monte Carlo algorithms [Duane et al., 1987, Neal, 2011] and their variants. Considerable attention has been given to Hamiltonian Monte Carlo algorithm lately, with some coupling constructions that will be described in later chapters, thus we devote the next section to that method.

1.4.4 Hamiltonian Monte Carlo

Assume the target π is defined on $\mathbb{R}^d$ and that its density is differentiable, the same setting as for the Metropolis adjusted Langevin algorithm. We define an extended target distribution $\bar{\pi}(x,p)=\pi(x)\mathcal{N}(p;0,M)$ on $\mathbb{R}^d\times\mathbb{R}^d$. The matrix M is called the mass matrix. Given a state X_{t-1} at step $t-1$, we can sample a variable P_{t-1} from its conditional distribution under $\bar{\pi}$ by simply sampling from $\mathcal{N}(0,M)$. This can be seen as an update in a Gibbs sampler leaving $\bar{\pi}$

invariant. Now that we have a pair (X_{t-1}, P_{t-1}), we imagine the P component to be the momentum of a particle with position given by the X component, and with mass M (nevermind that the mass is a matrix and not a scalar). Minus the logarithm of $\bar{\pi}$ is interpreted as an energy, defined as the sum of potential energy $-\log \pi$ and kinetic energy $p^T M^{-1} p/2$. Putting a physicist's hat on (gloves are optional), we can solve the equations of motion to obtain the trajectory of the particle, evolving through the energy landscape, over time. We define the initial position $q(0) = X_{t-1}$ and initial momentum $p(0) = P_{t-1}$, the Hamiltonian function $H(q,p) = -\log \pi(q) + p^T M^{-1} p/2$, then we can solve the Hamiltonian equation numerically to obtain $(q(t), p(t))$ for times $t \geq 0$. This is often done by performing L steps of a leap-frog or "Verlet" integrator with stepsize ε. This means that for $\ell = 1, \ldots, L$ we recursively define

$$\begin{aligned} p(\ell + 1/2) &= p(\ell) + \frac{\varepsilon}{2} \nabla \log \pi(q(\ell)), \\ q(\ell + 1) &= q(\ell) + \varepsilon M^{-1} p(\ell + 1/2), \\ p(\ell + 1) &= p(\ell + 1/2) + \frac{\varepsilon}{2} \nabla \log \pi(q(\ell + 1)). \end{aligned} \tag{1.4.4}$$

After L steps we set $X^\star = q(L)$ and $P^\star = p(L)$ to be the proposed state to be accepted or not as replacement for (X_{t-1}, P_{t-1}) according to some probability. Note that $(X^\star, P^\star)$ is a deterministic function of (X_{t-1}, P_{t-1}), that we can write $\Phi(X_{t-1}, P_{t-1})$. The function Φ depends on ε and L. It remains to see whether we can define a valid acceptance probability so that, at the end of the day, $(X_t)_{t \geq 0}$ will constitute a Markov chain converging to π as requested. To that effect, define $S : (x,p) \mapsto (x, -p)$, another deterministic function that simply flips the sign of the momentum. Define the acceptance probability

$$\alpha_{\text{HMC}}(X_{t-1}, P_{t-1}) = \min\left(1, \frac{\bar{\pi}(X^\star, P^\star)}{\bar{\pi}(X_{t-1}, P_{t-1})}\right), \tag{1.4.5}$$

and the Markov kernel

$$\begin{aligned} K_{\text{HMC}}((X_{t-1}, P_{t-1}), A) = {} & \alpha_{\text{HMC}}(X_{t-1}, P_{t-1}) \mathbb{1}(\Phi(X_{t-1}, P_{t-1}) \in A) \\ & + (1 - \alpha_{\text{HMC}}(X_{t-1}, P_{t-1})) \mathbb{1}(S(X_{t-1}, P_{t-1}) \in A). \end{aligned} \tag{1.4.6}$$

It can be shown that K_{HMC} is $\bar{\pi}$-invariant, using some properties of Φ: namely, the fact that $\Phi^{-1} = S \circ \Phi \circ S$, and the fact that Φ preserves the volumes of subsets of $\mathbb{R}^d \times \mathbb{R}^d$. These properties would not be shared by all ways of numerically solving Hamiltonian equations, and the interestedreader can consult Bou-Rabee and Sanz-Serna [2018] on the topic of which solver or "integrator" to use in Hamiltonian Monte Carlo. This entire business above is a succession of two Markov steps, both leaving $\bar{\pi}$ invariant, which implies that the composition of these steps is a Markov kernel leaving $\bar{\pi}$ invariant. Since the P-component of the state is refreshed independently at each iteration, we can also view this as a Markov move from X_{t-1} to X_t, leaving πinvariant as π is the marginal distribution of $\bar{\pi}$ on the X-component. To summarize, the Hamiltonian Monte Carlo algorithm can be written as follows, at step $t \geq 1$.

- Sample $P_{t-1} \sim \mathcal{N}(0, M)$.
- Perform L steps of leap-frog integrator with stepsize ε as in (1.4.4) to obtain $(X^\star, P^\star) = \Phi(X_{t-1}, P_{t-1})$.
- With probability $\alpha_{\text{HMC}}(X_{t-1}, P_{t-1})$ set $X_t = X^\star$, otherwise set $X_t = X_{t-1}$.

It can be checked that setting $L = 1$, Hamiltonian Monte Carlo with a stepsize ε and mass matrix M is equivalent to the Metropolis-adjusted Langevin algorithm described in Section 1.4.2, with stepsize $h = \varepsilon^2$ and covariance matrix $\Sigma = M^{-1}$, unless I got it wrong. Another general feature is that, if the stepsize ε is close to zero, the acceptance probability $\alpha_{\text{HMC}}(X_{t-1}, P_{t-1})$ is close to one. If the user further chooses a large number of steps L, then the proposed state $(X^\star, P^\star)$ might be far away from the current state (X_{t-1}, P_{t-1}), but can still have a large chance of being accepted. Informally, this is the motivation for using this algorithm compared to e.g. random walk Metropolis–Hastings, whereby large moves can be proposed if the proposal covariance matrix Σ is "large" but they are likely to be rejected.

The Hamiltonian Monte Carlo is an appealing algorithm, also because the analogies with physics provide numerous opportunities to write exciting words such as "momentum", "energy" or even "symplectic" and "flow". These opportunities should not be missed! Furthermore the algorithm performs very well in some (interesting!) settings, often when the target distribution is close to being Normal [e.g. Beskos et al., 2013, Bou-Rabee et al., 2018]. Impressive numerical results were for instance obtained in Caron and Fox [2017] where Hamiltonian Monte Carlo is used as part of a larger Gibbs sampler. Hamiltonian Monte Carlo also has some apparent limitations. First, it shines mostly in continuous state spaces and for smooth targets where gradients are helpful guide for proposed moves. There has been many proposed extensions to discrete state spaces, see Zhang et al. [2012] for instance, but little consensus yet; see Zanella [2019] for some analogies between gradient-based methods in continuous spaces and certain local moves in discrete state spaces. The Hamiltonian Monte Carlo algorithm does not feature any particular mechanism to escape local modes [e.g. Mangoubi et al., 2018]. The distribution of the momentum needs to be tuned according to the tails of the target [e.g. Livingstone et al., 2019], with the version described above expected to work only for tails similar to that of the Normal distribution. Tuning the step-size is potentially difficult [e.g. Livingstone and Zanella, 2019]. Various articles have been proposed to remedy each of these difficulties, and research on the topic is active.

1.5 MCMC examples

In this section we provide concrete examples of MCMC algorithms, which might appear again in later chapters.

1.5.1 Bayesian logistic regression and titanic data set

Some of the passengers of the Titanic survived, such as Rose, and some did not, such as Jack. A data set gathers information about 714 passengers, out of whom 290 survived. The information includes whether the passenger was traveling in first, second and third class, whether they were male or female, their age, and how many siblings or spouses were aboard. We can fit a regression model to measure the association between the probability of survival and these covariates. This example follows the blog post written by Arthur Charpentier (http://freakonometrics.hypotheses.org/60575). Denoting by n the number of individuals, $y_i \in \{0,1\}$ indicates whether the i-th individual survived ($y_i = 1$), x_i is a column vector of information about the individual, and the logistic regression model specifies

$$\mathbb{P}(Y_i = 1|\beta) = \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)},$$

and the likelihood is $\mathcal{L}(\beta) = \prod_{i=1}^n \mathbb{P}(Y_i = y_i|\beta)$. We will put a prior distribution on β, with all components being independent and marginally $\mathcal{N}(0,\sigma^2)$. Let's set $\sigma^2 = 5^2$. The target distribution is the posterior distribution of $\beta \in \mathbb{R}^p$ given the data set $(y_i, x_i)_{i=1}^n$, ad the number of covariates will be $p = 6$, namely

- the first entry is equal to one, which leads to the inclusion of an intercept,
- a binary variable indicating whether the passenger travelled in second class,
- a binary variable indicating whether the passenger travelled in third class,
- a binary variable indicating whether the passenger was male,
- the age of the passenger,
- the number of siblings/spouses aboard.

Logistic regressions are often used to illustrate MCMC algorithms, because the posterior distribution is not conjugate to any prior distribution and thus Monte Carlo algorithms are somewhat necessary. However, they are not necessarily sensible as a benchmark to compare methods [Chopin and Ridgway, 2017].

For the target at hand, it turns out that a Laplace approximation of the posterior distribution is very accurate. Such approximation is defined as $\mathcal{N}(\hat\beta, \hat\Sigma)$, where $\hat\beta$ is defined as the maximum likelihood estimator (the maximizer of the likelihood, obtained by numerical optimization), and $\hat\Sigma$ is defined as the inverse of the negative Hessian of the log-likelihood evaluated at $\hat\beta$. From classical asymptotics [Chapter 1, Ghosh and Ramamoorthi, 2003], $\mathcal{N}(\hat\beta, \hat\Sigma)$ would consistently approximate the posterior as $n \to \infty$.

By drawing samples from $\mathcal{N}(\hat\beta, \hat\Sigma)$ and computing importance weights as in Section 1.3, we obtain an effective sample size (1.3.2) close to its maximal value.

We next consider a Metropolis–Hastings with Normal random walk proposals. We use $\hat\Sigma$ as the variance of the proposed states, and initialize the chains from

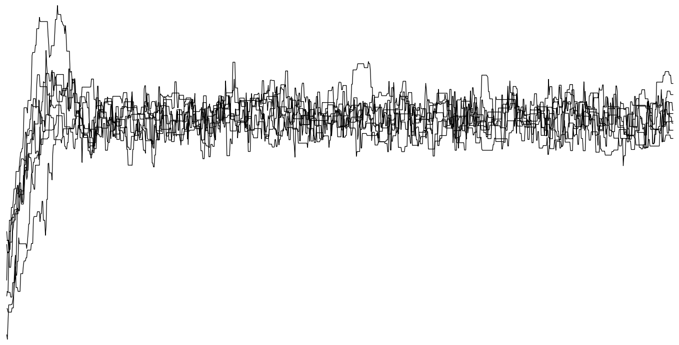

Figure 1.5: Trace plot.

$\mathcal{N}(0, I)$; it would be more sensible to start from $\mathcal{N}(\hat{\beta}, \hat{\Sigma})$ but then the trace plot (of the first component of multiple independent chains) in Figure 1.5 would be boring.

After discarding some burn-in and running multiple independent chains for longer, one can start approximating the 6-dimensional posterior distribution. Figure 1.6 shows a possible way of processing the samples into a visualization of the posterior distribution. From this, we can guess that Rose was much more likely to survive than Jack.

We continue with an implementation of Hamiltonian Monte Carlo. Here we set $M = \hat{\Sigma}^{-1}$, $L = 10$ and $\varepsilon = 0.1$. Some leapfrog trajectories are shown in Figure 1.7 (left) and a comparison of autocorrelograms with random walk MH is on the right. This is one way of comparing algorithms: one would often prefer samples with little autocorrelations. But correlations are defined for a choice of test function h. Some chains can exhibit large autocorrelations for some test functions but not for others.

1.5.2 Gibbs sampler and Ising model

This example is very classical in statistical physics, and poses different kinds of challenges. The state space is that of $n \times n$ grids, such as in Figure 1.8. At each site, there is a variable $X_{(i,j)}$, called a spin, taking values in $\{-1, +1\}$. Thus in total the grid X takes values in a finite set with $2^{n \times n}$ elements. The probability π is defined to encourage nearby spins to be of the same sign. It is defined as

$$\pi(x) \propto \exp\left(\beta \sum_{(i',j')\sim(i,j)} X_{(i,j)} X_{(i',j')}\right). \tag{1.5.1}$$

The $\sum_{(i',j')\sim(i,j)}$ is over the neighbours of (i, j). A site (i, j) has four neighbours, located up, down, left or right, and we assume periodic boundary conditions, e.g. the site $(1, 1)$ is a neighbour of $(1, 2)$, $(2, 1)$, $(1, n)$ and $(n, 1)$. The parameter

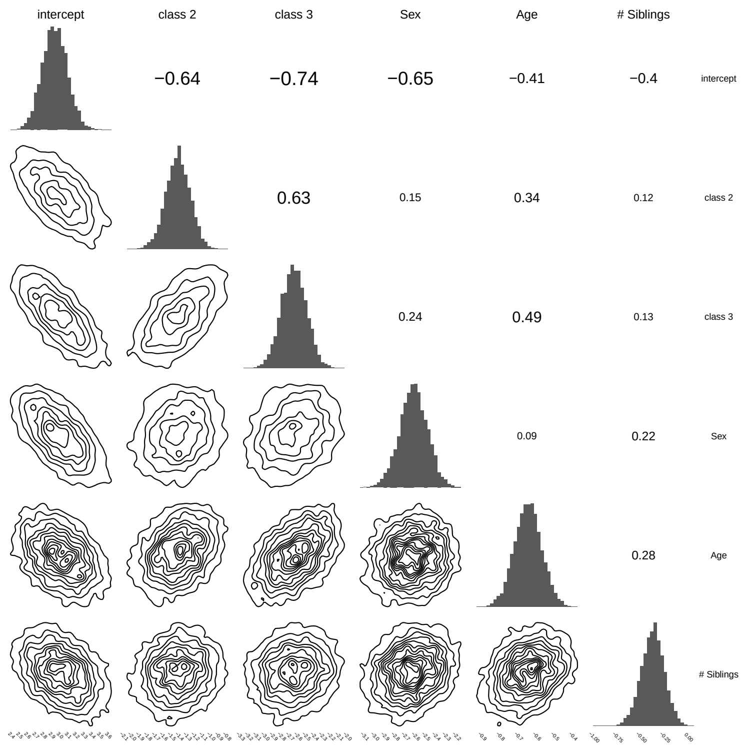

Figure 1.6: Pairs of variables and marginals.

β is referred to as an "inverse temperature" and is a non-negative real value. When it is larger, neighbouring sites are more likely be of the same sign under π. Figure 1.8 shows realizations of X obtained for different values of β.

The classical MCMC strategy to target π is a Gibbs sampler. With current state $X_{t-1} = x$, the sites (i,j) are "sweeped" randomly or deterministically, and a new state $x_{(i,j)}$ is drawn from its conditional distribution given the neighbouring sites. That is a Bernoulli distribution on $\{-1,+1\}$ with a probability that depends on β and on the sum of the neighbouring spins. That Gibbs sampler is more often called Glauber dynamics, and its performance as a function of n and β has been the topic of a rich applied probability literature, e.g. Levin et al. [2010].

Here the state space is large and discrete, and visual diagnostics tend to be less convenient and less useful than in continuous state spaces. However we still hope that an MCMC estimator as in (1.4.1) will be consistent for expectations with respect to π, and the question of how many iterations are required for the chains to reach stationarity remains equally important as in continuous spaces. Note that there are continuous space variants of the Ising model, such as the one considered for image denoising in Gibbs [2004].

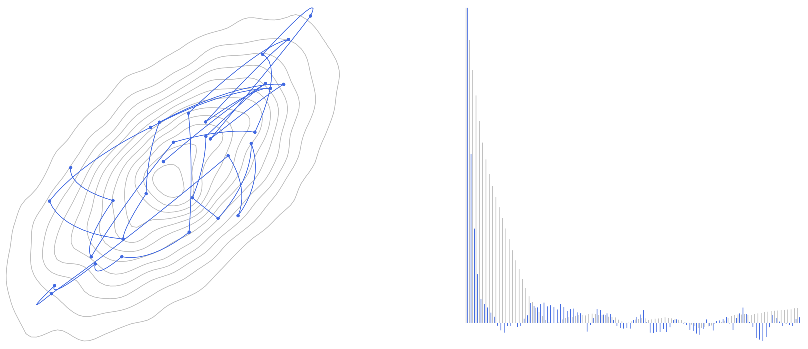
Figure 1.7: Hamiltonian Monte Carlo and autocorrelograms.

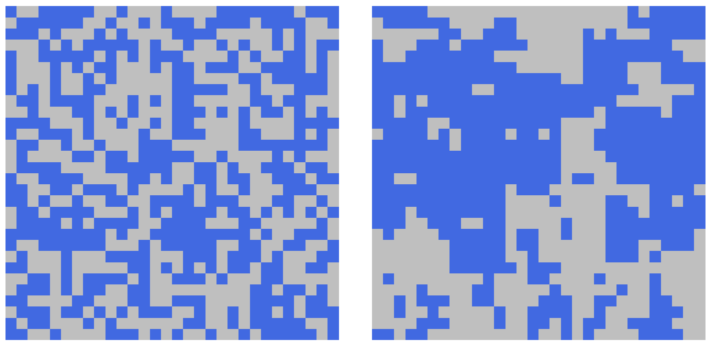
Figure 1.8: States in the Ising model.

1.5.3 Graphs colourings

[This section will present the example of graph colouring taken from Jerrum [1998]].

1.6 Some implementation tips and tricks

Most topics of these notes are very concrete in the sense that they can be directly illustrated and apprehended with a computer. Without undermining all the work that has been put into various software packages for Monte Carlo methods, it is worth stressing the importance of implementing Monte Carlo methods oneself. Doing so is an essential part of understanding these methods and their limitations. Any programming language will do, but it is useful to have ready-made functions for the simulation of standard random variables, for basic optimization tasks, for univariate numerical integration, and a basic linear algebra toolbox. Irrespective of the languagem, there are generic tricks that are helpful in implementing Monte Carlo methods, some of which are recalled briefly.

When manipulating probability density functions, likelihood functions and such, it is often very helpful to work on log-scale. For instance, suppose that we define a function $w : x \mapsto f(x)/g(x)$ where f and g are two unnormalized probability density functions. Suppose that we draw a sample $X_1, \ldots, X_N$ i.i.d. Normal(0,1) and that we want to compute $N^{-1}\sum_{n=1}^{N} w(X_n)$ or its logarithm. Some code to do that in R could look as follows.

```r
N <- 1e4
x <- rnorm(N)
logf <- function(x) dcauchy(x, log = TRUE)
logg <- function(x) dnorm(x, log = TRUE)
logw <- logf(x) - logg(x)
mlw <- max(logw)
logZ <- mlw + log(mean(exp(logw - mlw)))
```

In a randomized algorithm, there might be instructions that should be executed with some probability $\alpha \in [0,1]$. In that case, the trick is to generate a uniform random variable U in $[0,1]$, and to compare it to α: indeed U is less than α with probability equal to α. For instance, suppose that we want to assign the value $-\infty$ to a variable Z with probability $\min(1, f(X)/g(X))$ for some functions f, g and some variable X_1, otherwise Z is set to X. An R code implementing this might look as follows.

```r
if (log(runif(1)) < (logf(x) - logg(x))){
  z <- -Inf
} else {
  z <- x
}
```

In general, when creating large objects in R or Python, such as a large matrix, it is useful to create the structure entirely from the beginning, rather than e.g. adding rows one by one. So for instance, if generating a Markov chain of size d forward in time for T steps and storing it in a matrix with T rows and d columns, it is probably a good idea to create the $T \times d$ matrix initially and filling it row by row, rather than increasing the size of the matrix by one row T times. Some code instantiating the large matrix first might look like the following.

```r
T <- 50
x <- matrix(0, nrow = T, ncol = 2)
xcurrent <- c(1,-1)
for (iteration in 1:T){
  increment <- rnorm(1)
  xcurrent <- 0.9 * xcurrent + increment
  x[iteration,] <- xcurrent
}
```

In general when implementing Monte Carlo methods, start with a simple target distribution, for which you have ways of checking that your code produces correct results. For example, with an Ising model, or with images, you could start with small grids or low resolutions. In the case of a Gibbs sampler for a Bayesian hierarchical model, you could start by setting most of the parameters to constants, so that there would only remain a small-dimensional problem left to tackle. If you are implementing gradient-based MCMC algorithms, you could start with the closest Gaussian target distribution to your target, and you can check manually computed gradients with numerical differentiation tools or with auto-differentiation.

1.7 Summary and references

Random variables can be defined in various ways, some of which make sampling these variables an interesting task. Monte Carlo methods refer to algorithms that generate samples that approximate a distribution of interest, called the "target distribution". One way of formalizing the validity of a Monte Carlo method is that some sample averages computed on the generated samples, converges to some expectation of interest as the number of samples goes to infinity. Research on Markov chain Monte Carlo methods includes the design of new algorithms, as well as advancing our common understanding of the convergence of iterative methods, and determining in practice how many iterations to perform.

Ripley [1987] is a classic and insightful book on Monte Carlo methods, and similarly to Devroye [2006], describes countless clever ideas. Robert and Casella [1999] contains perhaps more details and more unifying framework, including a self-contained treatment of Markov chain theory for MCMC purposes. Besag [2001] provides a brief account, with both pedagogical descriptions of Monte Carlo methods, and interesting uses of Monte Carlo in statistics but outside of Bayesian statistics, for instance in hypothesis testing. Liu [2008] has a broader range of application areas, covers sequential Monte Carlo methods, and contains a selection of more theoretical topics. Brooks et al. [2011] provides a recent overview, representative of the state of MCMC with applications to statistics. Lelièvre et al. [2010] has more emphasis on applications in statistical physics, and thus treats more extensively the problems of multimodality and normalizing constant estimation.

In the next chapters we will see how coupling techniques can be used to study various properties of random variables in general and of Monte Carlo methods. Couplings are related to various meaningful distances between probability distributions, which can be used for instance to compare the marginal distribution of a Markov chain to its limiting distribution. Concretely, we will see examples where couplings have been used in proof techniques, e.g. to establish asymptotic approximations or to establish convergence rates of Markov chains, examples where they provide practical diagnostics of convergence for MCMC (how many iterations to perform?), and examples where they have been used in the design of Monte Carlo algorithms.

Bibliography

S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32 (3):405–431, 2017. 9

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2): 192–225, 1974. 12

J. Besag. Markov chain Monte Carlo for statistical inference. *Center for Statistics and the Social Sciences*, 9:24–25, 2001. 11, 20

A. Beskos, N. Pillai, G. Roberts, J.-M. Sanz-Serna, and A. Stuart. Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501–1534, 2013. 14

L. J. Billera and P. Diaconis. A geometric interpretation of the Metropolis–Hastings algorithm. *Statistical Science*, pages 335–339, 2001. 11

N. Bou-Rabee and J. M. Sanz-Serna. Geometric integrators and the Hamiltonian Monte Carlo method. *Acta Numerica*, 27:113–206, 2018. 13

N. Bou-Rabee, A. Eberle, and R. Zimmer. Coupling and convergence for Hamiltonian Monte Carlo. *arXiv preprint arXiv:1805.00452*, 2018. 14

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov chain Monte Carlo*. CRC press, 2011. 20

F. Caron and E. B. Fox. Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79 (5):1295–1366, 2017. 14

S. Chatterjee and P. Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018. 9

N. Chopin and J. Ridgway. Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. *Statistical Science*, 32(1):64–87, 2017. 15

A. S. Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):651–676, 2017. 11

G. Deligiannidis, A. Doucet, and S. Rubenthaler. Ensemble rejection sampling. *arXiv preprint arXiv:2001.09188*, 2020. 7, 8

L. Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006. 4, 5, 6, 20

L. Devroye. A note on generating random variables with log-concave densities. *Statistics & Probability Letters*, 82(5):1035–1039, 2012. 7

R. Douc, E. Moulines, P. Priouret, and P. Soulier. *Markov chains*. Springer, 2018. 9

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987. 12

A. Durmus and É. Moulines. Quantitative bounds of convergence for geometrically ergodic Markov chain in the Wasserstein distance with application to the Metropolis Adjusted Langevin Algorithm. *Statistics and Computing*, 25 (1):5–19, 2015. 11

R. Dwivedi, Y. Chen, M. J. Wainwright, and B. Yu. Log-concave sampling: Metropolis-hastings algorithms are fast. *Journal of Machine Learning Research*, 20(183):1–42, 2019. 11

J. Ehrman, L. Fosdick, and D. Handscomb. Computation of order parameters in an Ising lattice by the Monte Carlo method. *Journal of Mathematical Physics*, 1(6):547–558, 1960. 12

A. E. Gelfand and A. F. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410): 398–409, 1990. 12

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984. 12

J. K. Ghosh and R. V. Ramamoorthi. *Bayesian nonparametrics*. Springer Science & Business Media, 2003. 15

A. L. Gibbs. Convergence in the Wasserstein metric for Markov chain Monte Carlo algorithms with applications to image restoration. *Stochastic Models*, 20(4):473–492, 2004. doi: 10.1081/STM-200033117. URL `https://doi.org/10.1081/STM-200033117`. 17

R. J. Glauber. Time-dependent statistics of the Ising model. *Journal of Mathematical Physics*, 4(2):294–307, 1963. doi: 10.1063/1.1703954. URL https://doi.org/10.1063/1.1703954. 12

D. Görür and Y. W. Teh. Concave-convex adaptive rejection sampling. *Journal of Computational and Graphical Statistics*, 20(3):670–691, 2011. 7

U. Grenander and M. I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994. 11

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970. 10, 12

D. B. Hitchcock. A history of the Metropolis–Hastings algorithm. *The American Statistician*, 57(4):254–257, 2003. 10

S. F. Jarner and E. Hansen. Geometric ergodicity of Metropolis algorithms. *Stochastic processes and their applications*, 85(2):341–361, 2000. 11

M. Jerrum. Mathematical foundations of the Markov chain Monte Carlo method. In *Probabilistic methods for algorithmic discrete mathematics*, pages 116–165. Springer, 1998. 18

A. Kong. A note on importance sampling using standardized weights. *University of Chicago, Dept. of Statistics, Tech. Rep*, 348, 1992. 9

P. L'Ecuyer. Random number generation. *Handbook of computational statistics*, pages 35–71, 2012. 2

T. Lelièvre, G. Stoltz, and M. Rousset. *Free energy computations: A mathematical perspective*. World Scientific, 2010. 20

D. A. Levin, M. J. Luczak, and Y. Peres. Glauber dynamics for the mean-field Ising model: cut-off, critical power law, and metastability. *Probability Theory and Related Fields*, 146(1-2):223, 2010. 17

J. S. Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008. 20

S. Livingstone and G. Zanella. On the robustness of gradient-based MCMC algorithms. *arXiv preprint arXiv:1908.11812*, 2019. 12, 14

S. Livingstone, M. F. Faulkner, and G. O. Roberts. Kinetic energy choice in Hamiltonian/hybrid Monte Carlo. *Biometrika*, 106(2):303–319, 2019. 14

O. Mangoubi, N. S. Pillai, and A. Smith. Does Hamiltonian Monte Carlo mix faster than a random walk on multimodal densities? *arXiv preprint arXiv:1808.03230*, 2018. 14

S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability.* Springer Science & Business Media, 1993. 9

R. M. Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003. 12

R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011. 12

E. Nummelin. MC's for MCMC'ists. *International Statistical Review*, 70(2): 215–240, 2002. 9

A. B. Owen. *Monte Carlo theory, methods and examples.* 2013. 7

B. D. Ripley. *Stochastic simulation.* John Wiley & Sons, 1987. 20

C. Robert and G. Casella. *Monte Carlo statistical methods.* Springer Science & Business Media, 1999. 7, 8, 11, 20

C. Robert and G. Casella. A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data. *Statistical Science*, pages 102–115, 2011. 12

E. Roger. Stan Ulam, John Von Neumann and the Monte Carlo method. *Argonne, USA*, 1987. 1

P. Wild and W. Gilks. Algorithm as 287: Adaptive rejection sampling from log-concave density functions. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 42(4):701–709, 1993. 7

G. Zanella. Informed proposals for local MCMC in discrete spaces. *Journal of the American Statistical Association*, pages 1–27, 2019. 14

Y. Zhang, Z. Ghahramani, A. J. Storkey, and C. A. Sutton. Continuous relaxations for discrete Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 3194–3202, 2012. 14