Monte Carlo Methods

Thomas Taimre[1] , *Dirk P. Kroese*[1] , *and Zdravko I. Botev*[2]

Keywords: *simulation, random variables, numerical integration, MCMC, variance reduction, randomized optimization*

Abstract

Abstract: Monte Carlo methods are algorithms that employ pseudo-random numbers to produce numerical approximations to a range of difficult computational problems. This article describes the key ideas behind such algorithms and their use in simulation, estimation, and optimization problems.

Monte Carlo methods rely on carrying out random experiments using a computer [5]. Such methods are often used to *sample* from a stochastic model or distribution (*see* **simulation**) or *estimate* particular quantities associated with a simulation model. They can also be used to *optimize* complicated or noisy functions. In this article, we highlight some of the fundamental Monte Carlo methods for sampling, estimation, and optimization.

[1]The University of Queensland, Australia
[2]The University of New South Wales, Australia

1 Sampling

At their core, Monte Carlo methods rely on simulating realizations of random objects: **random variables**, random vectors, stochastic processes, etc.

When designing techniques to generate these random objects, a theoretical assumption is that one has access to an infinite sequence $U_1, U_2, \ldots$ of *independent and identically distributed* (iid) random variables from the uniform distribution on $(0,1)$, denoted here by $\mathsf{Unif}(0,1)$. In practice, however, one uses a computer to produce (pseudo-) **uniform random numbers** that have good distributional qualities (that is, their output is difficult to distinguish statistically from the ideal) and can be generated fast [7]. An additional advantage is that such sequences are *repeatable*, when using the same initial states (seeds) of the (pseudo-) **random number generators**.

The simulation of a random variable X typically involves the following two steps:

1. Simulate $U_1, \ldots, U_k \sim \mathsf{Unif}(0,1)$ independently, for some $k = 1, 2, \ldots$.
2. Return $X = g(U_1, \ldots, U_k)$ for an appropriate real-valued function g.

Two of the most fundamental approaches for sampling from a desired distribution are the *inverse-transform method* and the *rejection sampling* (*see also* **Simulation of Stochastic Processes**).

1.1 Inverse-Transform Method

To generate a single random variable X with cdf $F(x) = \mathbb{P}[X \le x]$, the most direct approach is via the inverse-transform method: $X = F^{-1}(U)$, where $U \sim \mathsf{Unif}(0,1)$, and where F^{-1} is defined as $F^{-1}(u) = \inf\{x \in \mathbb{R} \mid F(x) \ge u\}$, which coincides with the usual definition of the inverse of F, if F is bijective.

Throughout, we will use the notation $X \sim \mathsf{Dist}$, $X \sim F$, or $X \sim f$ to denote that the random variable X has distribution Dist, or **cumulative distribution function** (cdf) F, or **probability density function** (pdf) f.

Example 1.1 (Exponential Random Variables)**.** The **exponential distribution** with rate $\lambda > 0$, which we denote by $\mathsf{Exp}(\lambda)$, has cdf $F(x) = 1 - \exp(-\lambda x)$ for $x > 0$ and 0 otherwise. To simulate a random variable $X \sim \mathsf{Exp}(\lambda)$, we use the fact that $F^{-1}(u) = -\ln(1-u)/\lambda$, so that the inverse-transform generator is simply to return $X = -\ln(1-U)/\lambda$ where $U \sim \mathsf{Unif}(0,1)$. A small computational saving can be made by instead using $X = -\ln(U)/\lambda$, which follows from noting that $1 - U$ and U are both distributed according to $\mathsf{Unif}(0,1)$. □

1.2 Rejection Sampling

Suppose that the objective is to sample from $X \sim f$, where the pdf f is difficult to sample from using the inverse-transform method. Suppose further that we can easily sample from a pdf g with the property that $f(x) \leq Cg(x)$ for all $x \in \mathbb{R}$ for some constant $C \geq 1$. The idea of *rejection sampling* is to propose a Y according to g (that is, simulate $Y \sim g$) and accept $X = Y$ as a sample from f with probability $f(Y)/(Cg(Y))$; otherwise, rejecting and proposing once again. This results in the following algorithm:

Algorithm 1 : Rejection Sampling

```
repeat
    Generate $X \sim g$ and $U \sim \mathsf{Unif}(0,1)$, independently.
until $U \leq f(X)/(Cg(X))$
return $X$
```

This procedure draws samples according to f, with each proposed $Y \sim g$ accepted with probability $1/C$, which quantifies the *efficiency* of the procedure. Although described for the case of a one-dimensional random variable, in principle this approach applies to random vectors taking values in $\mathbb{R}^d$, although its efficiency tends to decrease as the dimension d increases.

Example 1.2 (Positive Normal Distribution)**.** Suppose we wish to draw from the *positive* *normal distribution*, with pdf $f(x) = \sqrt{\frac{2}{\pi}} \exp(-x^2/2)$ for $x > 0$ using an $\mathsf{Exp}(1)$ proposal with pdf $g(x) = \exp(-x)$ for $x > 0$ (*see also* **Half-Normal Distribution**). The smallest constant C such that $f(x) \leq Cg(x)$ for all $x > 0$ is $C = \sqrt{\frac{2\mathrm{e}}{\pi}}$. Thus, the rejection sampling **algorithm** reads as follows.

Algorithm 2 : Positive Normal Sampler

```
repeat
    Simulate $X \sim \mathsf{Exp}(1)$ and $U \sim \mathsf{Unif}(0,1)$, independently.
until $U \leq \exp\left(-\frac{1}{2}(X-1)^2\right)$
return $X$
```

The efficiency of the method is $1/C = \sqrt{\frac{\pi}{2\mathrm{e}}} \approx 0.76$. □

We have just described two approaches for *exact* sampling of a single random variable from a desired distribution. Next, we outline a well-known general idea for sampling *approximately* from a desired distribution.

2 Markov Chain Monte Carlo

It is often the case (particularly in Bayesian statistics — *see* **Bayesian Inference**) that one wishes to simulate a random vector $\boldsymbol{X} \sim f$ taking values in some set $\mathscr{X} \subseteq \mathbb{R}^d$, where the pdf f is of the form $f(\boldsymbol{x}) = p(\boldsymbol{x})/\mathscr{Z}$ such that $p : \mathscr{X} \to \mathbb{R}_+$ is a known function and $\mathscr{Z}$ is a finite normalization constant which may be known or unknown. The fact that $\mathscr{Z}$ does not need to be known explicitly is particularly useful in practice.

The idea of **Markov chain Monte Carlo** (MCMC) is to construct an ergodic Markov chain whose limiting distribution has pdf f (*see* **Markov Chains and Markov Processes**). Then, by running the Markov chain for a sufficiently long time (a so-called *burn-in* period) so that it is approximately stationary, one has access to (dependent) samples approximately distributed according to f (*see also* **Stationary Distribution**).

2.1 Metropolis–Hastings

The template for many MCMC samplers is the **Metropolis–Hastings algorithm**, which takes as its ingredients the target pdf $f(\boldsymbol{x}) = p(\boldsymbol{x})/\mathcal{Z}$, and a proposal transition function $q(\boldsymbol{y} \,|\, \boldsymbol{x})$, according to which transitions from the current state $\boldsymbol{x}$ to a new state $\boldsymbol{y}$ are simulated. Each proposed transition is accepted with the *acceptance probability*

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = \min\left\{\frac{f(\boldsymbol{y})q(\boldsymbol{x} \,|\, \boldsymbol{y})}{f(\boldsymbol{x})q(\boldsymbol{y} \,|\, \boldsymbol{x})}, 1\right\} , \tag{1}$$

otherwise the chain remains in the current state. Note that in (1), f can be replaced by p as the normalization constants $\mathcal{Z}$ in the numerator and denominator cancel.

Provided that the proposal transition function $q(\boldsymbol{y} \,|\, \boldsymbol{x})$ is chosen such that the entire state space $\mathcal{X}$ is *accessible* from any point $\boldsymbol{x} \in \mathcal{X}$ — that is, the Markov chain never gets "trapped" and can always traverse the entire state space — this procedure describes an ergodic Markov chain on $\mathcal{X}$ whose stationary pdf is f [9]. The Metropolis–Hastings algorithm is thus as follows:

Algorithm 3 : Metropolis–Hastings Sampler

```
Require: Initialize with some state $\boldsymbol{X}_0$ for which $f(\boldsymbol{X}_0) > 0$ and set $t \leftarrow 0$
  repeat
      Propose $\boldsymbol{Y} \sim q(\boldsymbol{y} \,|\, \boldsymbol{X}_t)$ and generate $U \sim \mathsf{Unif}(0,1)$ independently.
      $\boldsymbol{X}_{t+1} \leftarrow \begin{cases} \boldsymbol{Y}, & \text{if } U \le \alpha(\boldsymbol{X}_t, \boldsymbol{Y}) \\ \boldsymbol{X}_t, & \text{otherwise} \end{cases}$
      $t \leftarrow t+1$
  until a stopping criterion is met
  return $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$
```

There are many flavors of MCMC samplers that can be viewed as particular cases of the Metropolis–Hastings framework. If $q(\boldsymbol{y} \,|\, \boldsymbol{x}) = g(\boldsymbol{y})$ does not depend on $\boldsymbol{x}$ then we have the *independence sampler*, and the acceptance probability (1) simplifies to

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = \min\left\{\frac{f(\boldsymbol{y})g(\boldsymbol{x})}{f(\boldsymbol{x})g(\boldsymbol{y})}, 1\right\} .$$

If $q(\boldsymbol{y} \,|\, \boldsymbol{x}) = q(\boldsymbol{x} \,|\, \boldsymbol{y})$, so that the proposal is symmetric in its arguments, the algorithm is known as the *random walk sampler* (*see* **Random Walk Metropolis Sampler**) and the acceptance probability becomes

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = \min\left\{\frac{f(\boldsymbol{y})}{f(\boldsymbol{x})}, 1\right\} .$$

Example 2.1 (Langevin Metropolis–Hastings Algorithms)**.** An example of a special flavor of the Metropolis–Hastings algorithm is the *Metropolis-Adjusted Langevin Algorithm* (MALA), which is motivated by the *Langevin diffusion*:

$$\mathrm{d}\boldsymbol{X}_t = \frac{\sigma^2}{2}\nabla \ln f(\boldsymbol{X}_t)\,\mathrm{d}t + \sigma\,\mathrm{d}\boldsymbol{W}_t,$$

where $(\boldsymbol{W}_t, t \geq 0)$ is the d-dimensional *Wiener process* (*see* **Brownian Motion**). The Langevin diffusion has stationary and limiting density f. Note that $\nabla \ln f(\boldsymbol{X}_t)$ does not depend on $\mathcal{Z}$, so we can freely replace f by p. The coefficient $\sigma > 0$ is frequently taken equal to unity. The idea with MALA is to take the one-step *Euler–Maruyama* approximation of the Langevin diffusion as a proposal (see also **Simulation Methods for Stochastic Differential Equations**). That is, propose

$$\boldsymbol{Y}_{n+1} = \boldsymbol{X}_n + \frac{\sigma^2}{2}h_n\nabla \ln p(\boldsymbol{X}_n) + \sigma\sqrt{h_n}\boldsymbol{Z}_n\,,$$

where $\boldsymbol{Z}_0, \boldsymbol{Z}_1, \ldots$ are iid d-dimensional standard normal vectors and $h_0, h_1, \ldots$ is a sequence of step sizes (*see also* **Multivariate Normal Distributions: Theory**). For a given $\boldsymbol{x}$, the proposal at step $n+1$ (that is, $\boldsymbol{Y}_{n+1}$) is then multivariate normal with mean vector $\boldsymbol{x} + \frac{\sigma^2}{2} h_n \nabla \ln p(\boldsymbol{x})$ and covariance matrix $\sigma^2 h_n \mathbf{I}_d$. □

2.2 Gibbs Sampling

The Gibbs sampler (*see* **Gibbs sampling**) can be viewed as a specialization of the Metropolis–Hastings algorithm to sample a d-dimensional random vector $\boldsymbol{X} \sim f$, where the pdf f is such that it is easy to sample from each (one-dimensional) conditional pdf $f(x_i \,|\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ for $i = 1, 2, \ldots, d$. Essentially, the Gibbs sampler is a Metropolis–Hastings algorithm with coordinate-wise updates which are always accepted. The coordinates can be updated in any deterministic order. For example, the *systematic* Gibbs sampler updates the coordinates in the order $1 \to 2 \to \cdots \to d$, as follows:

Algorithm 4 : Systematic Gibbs Sampler

```
Require: Initialize with some state $\boldsymbol{X}_0$ for which $f(\boldsymbol{X}_0) > 0$ and set $t \leftarrow 0$.
  repeat
      Draw $Y_1 \sim f(x_1 \,|\, X_{t,2}, \ldots, X_{t,d})$
      for $i = 2, \ldots, d-1$ do
          Draw $Y_i \sim f(x_i \,|\, Y_1, \ldots, Y_{i-1}, X_{t,i+1}, \ldots, X_{t,d})$
      Draw $Y_d \sim f(x_d \,|\, Y_1, \ldots, Y_{d-1})$
      $\boldsymbol{X}_{t+1} \leftarrow \boldsymbol{Y}$ and then $t \leftarrow t+1$
  until a stopping criterion is met
  return $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$
```

We shall write $\boldsymbol{X} \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$ to denote a normal random vector with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. When $\boldsymbol{\mu} = \boldsymbol{0}$ and $\Sigma = \mathbf{I}_d$ (the $d \times d$ identity matrix), then we refer to $\boldsymbol{X}$ as a (d-dimensional) standard normal vector.

If instead the coordinates are updated in the order $1 \to 2 \to \cdots \to d-1 \to d \to d-1 \to \cdots \to 2 \to 1$ one obtains the *reversible Gibbs sampler*, and updating one or multiple coordinates drawn randomly from a distribution on the coordinate indices — for instance, updating all coordinates in the order of a random permutation or selecting a single coordinate according to the uniform distribution — is known as the *random scan Gibbs sampler*.

In all of the MCMC algorithms discussed so far, the Markov chain takes values in the same state space as the target distribution. A number of specialized MCMC algorithms augment the state space by introducing an *auxiliary variable* $\boldsymbol{u}$ in the Markov chain, in such a way that $\int f(\boldsymbol{x}, \boldsymbol{u})\, \mathrm{d}\boldsymbol{u} = f(\boldsymbol{x})$; that is, the marginal pdf with respect to $\boldsymbol{x}$ of the limiting pdf is equal to the target pdf $f(\boldsymbol{x})$.

Example 2.2 (Hamiltonian Monte Carlo)**.** The idea of Hamiltonian Monte Carlo (HMC) is to sample from a target pdf $f(\boldsymbol{x})$ with state space $\mathcal{X} \subseteq \mathbb{R}^d$ by augmenting each *position* vector $\boldsymbol{x} \in \mathcal{X}$ by a d-dimensional *momentum* vector $\boldsymbol{u} \in \mathcal{M} \subseteq \mathbb{R}^d$, so that the joint position–momentum density is of the form $f(\boldsymbol{x}, \boldsymbol{u}) = f(\boldsymbol{x}) f(\boldsymbol{u} \,|\, \boldsymbol{x})$, which ensures that marginalizing out $\boldsymbol{u}$ yields the desired target pdf $f(\boldsymbol{x})$. The corresponding *Hamiltonian* or *total energy* function $H : \mathcal{X} \times \mathcal{M} \to \mathbb{R}$ is defined by

$$H(\boldsymbol{x}, \boldsymbol{u}) = -\ln f(\boldsymbol{x}, \boldsymbol{u}) = -\ln f(\boldsymbol{u} \,|\, \boldsymbol{x}) - \ln f(\boldsymbol{x}) = K(\boldsymbol{u} \,|\, \boldsymbol{x}) + V(\boldsymbol{x}) \,,$$

where K is referred to as the *kinetic energy* and V as the *potential energy*, by analogy to a physical dynamical system.

Given an initial position–momentum pair $(\boldsymbol{x}_0, \boldsymbol{u}_0)$ with energy $H(\boldsymbol{x}_0, \boldsymbol{u}_0) = E$, Hamilton's equations (see e.g. [8]), given below, describe the deterministic evolution in continuous time $t > 0$ of this position–momentum pair $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ which maintains energy — that is

$H(\boldsymbol{x}_t, \boldsymbol{u}_t) = E$ for all $t > 0$. Hamilton's equations are:

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} &= \frac{\partial K}{\partial \boldsymbol{u}} \,, \\
\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} &= -\frac{\partial K}{\partial \boldsymbol{x}} - \frac{\partial V}{\partial \boldsymbol{x}} \,.
\end{aligned}$$

Then, using the chain rule, we can verify that the rate of change of the energy with time is zero:

$$\frac{\mathrm{d}H}{\mathrm{d}t} = \frac{\partial H}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + \frac{\partial H}{\partial \boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial H}{\partial \boldsymbol{u}}\left(-\frac{\partial H}{\partial \boldsymbol{x}}\right) + \frac{\partial H}{\partial \boldsymbol{x}}\frac{\partial H}{\partial \boldsymbol{u}} = 0 \,.$$

The idea of HMC is to (a) start with an initial position vector; (b) augment the position vector by a sampled momentum vector; (c) deterministically evolve the position–momentum vector pair according to Hamilton's equations; (d) "forget" the momentum vector, and repeat from (a). This gives the following HMC algorithm:

```
Algorithm 5 : Hamiltonian Monte Carlo Sampler
Require: Initialize with $\boldsymbol{X}_0$ such that $f(\boldsymbol{X}_0) > 0$ and set $n \leftarrow 0$.
  repeat
      Augment position vector $\boldsymbol{X}_n$ with sampled momentum vector $\boldsymbol{U}_n \sim f(\boldsymbol{u} \,|\, \boldsymbol{X}_n)$.
      Evolve Hamilton's equations for some fixed time $t > 0$ with initial condition
      $(\boldsymbol{X}_n(0), \boldsymbol{U}_n(0)) \leftarrow (\boldsymbol{X}_n, \boldsymbol{U}_n)$ to obtain $(\boldsymbol{X}_n(t), \boldsymbol{U}_n(t))$.
      Set $\boldsymbol{X}_{n+1} \leftarrow \boldsymbol{X}_n(t)$ and then set $n \leftarrow n + 1$.
  until a stopping criterion is met
  return $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$
```

In this idealized presentation of HMC, one has freedom to choose the kinetic energy K and the integration time. Note that $\exp(-K(\boldsymbol{u} \,|\, \boldsymbol{x})) \propto f(\boldsymbol{u} \,|\, \boldsymbol{x})$ describes how momentum vectors are sampled given position vectors. A common choice for $f(\boldsymbol{u} \,|\, \boldsymbol{x})$ is the pdf of a $\mathsf{N}(\boldsymbol{0}, \Sigma(\boldsymbol{x}))$ distribution, known as the Riemannian–Gaussian kinetic energy (or the Euclidean–Gaussian kinetic energy in the case that $\Sigma(\boldsymbol{x}) = \Sigma$ does not depend on the position $\boldsymbol{x}$). A well-known heuristic for determining an appropriate integration time t in a dynamic/on-line fashion is the No-U-Turn criterion [4]. In practice Hamilton's equations are usually solved only approximately, not exactly. $\square$

3 Estimation

A simple example of an estimation problem is to determine the quantity $\ell = \mathbb{E}H(\boldsymbol{X})$, where $\boldsymbol{X}$ is a random object taking value in some set $\mathcal{X}$, $H : \mathcal{X} \to \mathbb{R}$ is a real-valued *performance function*, and $\mathbb{E}$ denotes the expectation operator. If $\boldsymbol{X}$ is a d-dimensional random vector taking values in $\mathbb{R}^d$ with a probability density function f, then $\ell = \int_{\mathbb{R}^d} H(\boldsymbol{x})f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$. Thus determining ℓ can be viewed as a **numerical integration** problem.

3.1 Crude Monte Carlo

If $\boldsymbol{X}$ is easy to simulate on a computer and the performance function is computationally inexpensive to evaluate, then the easiest way to estimate ℓ is by *Crude Monte Carlo*: generate N iid vectors $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N$ distributed according to f and, setting $Y_i = H(\boldsymbol{X}_i)$ for $i = 1, \ldots, N$, approximate ℓ by the **unbiased** estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} Y_i \equiv \overline{Y}\,.$$

By the **law of large numbers**, $\widehat{\ell}$ converges almost surely to ℓ, provided the **expectation** exists (*see* **Convergence of Sequences of Random Variables**). Moreover, if $\mathbb{E}H(\boldsymbol{X})^2 < \infty$, the rate of convergence is of the order $N^{-1/2}$, independent of the dimensionality of the random object $\boldsymbol{X}$ (a consequence of the **Central Limit Theorem** — *see also* **Convergence in Distribution and in Probability**). An approximate $1 - \alpha$ **confidence interval** for ℓ

can then be constructed as

$$\hat{\ell} \pm z_{1-\alpha/2} \frac{S}{\sqrt{N}} \, ,$$

where z_γ denotes the γ-**quantile** (*see also* **Quantiles**) of the $\mathsf{N}(0,1)$ distribution, and

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} \left(Y_i - \overline{Y} \right)^2 \tag{2}$$

is the **sample variance** of $Y_1, \ldots, Y_N$.

Example 3.1 (Hypothesis Testing)**.** In the context of **hypothesis testing**, simulation of a test **statistic** T under the **null hypothesis** can be used to estimate quantiles or P-values, for example when it is simple to sample T under the null hypothesis but it is difficult to analytically determine the **critical region** or P**-value** associated with T.

For the sake of illustration, suppose that we have a one-sided test (*see* **alternative hypothesis**) and the critical region is of the form $[t_{1-\alpha}, \infty)$, where $t_{1-\alpha}$ is the $(1-\alpha)$-quantile of the test statistic T; hence, the null hypothesis is rejected if $T \geq t_{1-\alpha}$. If $t_{1-\alpha}$ is unknown, a simple application of crude Monte Carlo is to sample $T_1, \ldots, T_N \sim \mathsf{Dist}_{H_0}$ independently and to estimate $t_{1-\alpha}$ by the sample quantile $\widehat{t}_{1-\alpha} = \min\{T_i \mid N^{-1} \sum_{k=1}^{N} \mathrm{I}\{T_k \leq T_i\} \geq 1-\alpha\}$, where $\mathrm{I}\{A\} = 1$ if event A occurs and 0 otherwise. Similarly, given an observed test statistic t, the crude Monte Carlo estimate of the P-value is $N^{-1} \sum_{k=1}^{N} \mathrm{I}\{T_k \geq t\}$. □

Example 3.2 (Bootstrap)**.** In the previous example, we relied explicitly on the ability to generate realizations of a random object $\boldsymbol{X}$. However, when $\boldsymbol{X}$ represents the outcome of a natural or physical system, in practice one may not have a sensible generative model, preferring instead for observed data to "speak for itself". Herein lies the idea of the **bootstrap** (*see also* **Bootstrap: Theory**): use the **empirical distribution** of an observed sample $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ as the model for $\boldsymbol{X}$, implicitly assuming that the sample are iid from some unknown distribution. Then sampling from the empirical distribution (in other words, resampling from the observed data — *see* **Sampling with and without replacement**) gives us a means to simulate approximately from the "true" generative distribution and to estimate quantities of interest (statistics) associated with it [3]. $\square$

Despite the guarantee of convergence, the crude Monte Carlo approach can require a large number of samples N to achieve a desired level of accuracy (as measured by the **standard error** $\sqrt{\mathrm{Var}(\hat{\ell})}$ or relative error $\sqrt{\mathrm{Var}(\hat{\ell})}/\ell$ of the estimator $\widehat{\ell}$ — estimated via $S/\sqrt{N}$ and $S/(\hat{\ell}\sqrt{N})$, respectively). This is particularly evident in **rare-event** estimation problems (*see* **Stochastic Simulation of Rare Events**), for instance when the performance function is of the form $H(\boldsymbol{X}) = \mathrm{I}\{S(\boldsymbol{X}) > \gamma\}$, for some real-valued function $S : \mathscr{X} \to \mathbb{R}$ and (large) constant γ. Here, the relative error of $\widehat{\ell}$ is given by

$$\frac{\sqrt{\ell(1-\ell)/N}}{\ell} = \frac{1}{\sqrt{N}}\sqrt{\frac{1}{\ell} - 1} \approx \sqrt{\frac{1}{N\ell}}\,,$$

for small ℓ. If $\ell = 10^{-6}$ then to achieve a relative error of 1% would require approximately $N = 10^{10}$ samples!

As this simple example indicates, techniques to construct Monte Carlo estimators with smaller **variance** can sometimes be essential in practice.

3.2 Variance Reduction Techniques

Variance reduction techniques such as **importance sampling**, **conditional Monte Carlo**, as well as the use of **control variates**, *splitting methods*, and **quasi Monte Carlo** (*see also* **Quasi-Random Sequences**) are commonly used to increase the efficiency over the crude Monte Carlo approach.

We highlight the use of importance sampling and control variates below.

Example 3.3 (Importance Sampling). Importance sampling (IS) relies on determining a pdf g which is easy to sample from and for which, for every $\boldsymbol{x} \in \mathcal{X}$, $g(\boldsymbol{x}) = 0$ implies $H(\boldsymbol{x})f(\boldsymbol{x}) = 0$. For such a pdf g, it is clear that

$$\ell = \mathbb{E}_f H(\boldsymbol{X}) = \int H(\boldsymbol{x}) f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = \int H(\boldsymbol{x}) \frac{f(\boldsymbol{x})}{g(\boldsymbol{x})} g(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = \mathbb{E}_g \left[H(\boldsymbol{X}) \frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} \right] ,$$

so that one can estimate ℓ unbiasedly via the IS estimator

$$\hat{\ell}_{\mathrm{IS}} = \frac{1}{N} \sum_{i=1}^{N} H(\boldsymbol{X}_i) \frac{f(\boldsymbol{X}_i)}{g(\boldsymbol{X}_i)} ,$$

where $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \sim g$ independently. Setting $Y_i = H(\boldsymbol{X}_i) f(\boldsymbol{X}_i)/g(\boldsymbol{X}_i)$, approximate confidence intervals can be constructed in the same way as for crude Monte Carlo.

The quality of an IS pdf g is typically measured by the common variance of the $\{Y_i\}$. One can show that the minimum variance IS pdf is given by $g^* \propto |H| f$. In particular, if $H > 0$ or $H < 0$ then $g^* = Hf/\ell$, and the common variance is equal to zero. This theoretical ideal gives one a clear idea of the desired structure of a "good" IS pdf — note that if one actually had access to g^* in the case that $H < 0$ or $H > 0$, then the problem of estimation would be rendered moot since the desired quantity ℓ would already be known.

The *variance minimization* approach and the *cross-entropy method* are two common techniques used to approximately determine the optimal reference parameter $\boldsymbol{\eta}^*$ for a given parametric family $g(\cdot; \boldsymbol{\eta})$ of IS densities [10, 11]. $\square$

Example 3.4 (Control Variates)**.** When estimating an unknown quantity $\ell = \mathbb{E} Y$, a *control variate* is another variable $\widetilde{Y}$ whose expectation $\widetilde{\ell} = \mathbb{E}\widetilde{Y}$ is known and for which Y and $\widetilde{Y}$ are correlated.

Given simulated values $Y_1, \ldots, Y_N$ and control variates $\widetilde{Y}_1, \ldots, \widetilde{Y}_N$ computed from the *same* simulation run, any estimator of the form

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N}\Big(Y_i - \beta(\widetilde{Y}_i - \widetilde{\ell})\Big)$$

for $\beta \in \mathbb{R}$ is unbiased, and the constant β which minimizes the variance of this estimator can be determined as $\beta^* = \mathrm{Cov}(Y, \widetilde{Y})/\mathrm{Var}(\widetilde{Y})$. Usually β^* is unknown but can be straightforwardly estimated using the sample covariance matrix of the $\{(Y_i, \widetilde{Y}_i)\}$. $\square$

4 Optimization

Monte Carlo methods are also found in **stochastic optimization** algorithms [12]. Here we highlight two well-known approaches, *stochastic approximation* and **simulated annealing**. Other useful techniques include the *cross-entropy* [1] and *splitting* [2] methods.

Consider first the *noisy optimization* setting, for which we wish to solve the minimization problem $\min_{\boldsymbol{x}\in\mathscr{X}} S(\boldsymbol{x})$ for some $\mathscr{X} \subseteq \mathbb{R}^d$ where $S : \mathscr{X} \to \mathbb{R}$ is an (unknown) objective function of the form $S(\boldsymbol{x}) = \mathbb{E}\widetilde{S}(\boldsymbol{x}, \boldsymbol{\xi})$ for some random vector $\boldsymbol{\xi}$ and $\widetilde{S}$ is a known real-valued function. In this setting, S cannot be directly observed, and instead only "noisy" observations $\widetilde{S}$ are available. Classical techniques such as gradient descent are therefore not directly applicable since ∇S is not available. However, one can still apply the same idea by constructing an estimator $\widehat{\nabla S}$ of the gradient. Such methods include *finite difference* methods, *infinitesimal perturbation analysis*, and the *score function* method; see e.g., [11].

Given a gradient estimator, the *stochastic approximation* method is the direct analogue of classical gradient descent to the noisy optimization setting. For each iteration $t = 0, 1, \ldots,$ steps are taken according to a strictly positive step size sequence $\beta_0, \beta_1, \ldots$ in the direction opposite to the estimated gradient, with the resulting point projected back into the feasible set $\mathscr{X}$ via a projection operator $\Pi_{\mathscr{X}} : \mathbb{R}^d \to \mathscr{X}$ — typically, $\boldsymbol{y} = \Pi_{\mathscr{X}}(\boldsymbol{x})$ is the closest point in $\mathcal{X}$ to $\boldsymbol{x}$ according to the Euclidean distance (i.e., $\boldsymbol{y} \in \operatorname{argmin}_{\boldsymbol{z} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{z}\|_2$). Thus, the updating step is captured in the following expression:

$$\boldsymbol{x}_{t+1} = \Pi_{\mathcal{X}} \left(\boldsymbol{x}_t - \beta_t \widehat{\nabla S}(\boldsymbol{x}_t) \right) .$$

If the step size sequence satisfies $\sum_{k=1}^{\infty} \beta_k = \infty$ and $\sum_{k=1}^{\infty} \beta_k^2 < \infty$, then the stochastic approximation algorithm is guaranteed to converge to a minimizer $\boldsymbol{x}^*$ of S in the mean-squared sense provided certain (mild) regularity conditions hold [6].

For stochastic approximation, the objective function S is not directly observable, and randomness is inherent to the problem. Even when S is directly observable, it can be beneficial to view the optimization problem as a problem of sampling from a distribution that is degenerate at the set of minimizers if S. Thinking along these lines, it is natural to try to construct a sequence of pdfs $f_1, f_2, \ldots$ which degenerates to the uniform distribution on the set of minimizers of S as $t \to \infty$. If one could sample a corresponding sequence of points $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$ *exactly* from $f_1, f_2, \ldots$ then the sequence would converge to a solution to the original problem! This ideal is not realistic, and instead one settles for sampling approximately from such a sequence, for instance via MCMC.

The well-known example is to take $f_t(\boldsymbol{x}) \propto \exp(-S(\boldsymbol{x})/T_t)$, where $T_1, T_2, \ldots$ is a sequence of positive numbers — "temperatures" — decreasing to zero. This is the basis of *simulated annealing*. The most common choice for the sequence T_t is to apply a so-called *geometric cooling schedule*, for which $T_{t+1} = \beta T_t$ for some cooling factor $\beta \in (0, 1)$ and initial temperature $T_0 > 0$.

The simulated annealing approach can be viewed as a special case of sampling from the set global maxima of the *Boltzmann pdf* $f(\boldsymbol{x}) \propto \exp(-S(\boldsymbol{x}))$. Taking $f_t \propto f^{1/T_t}$ generalizes the idea to produce a method for sampling from the set of global maxima of an arbitrary pdf f.

5 Related Articles

See also **Stochastic Simulation**; **Monte Carlo simulation**; **Monte Carlo Methods**; **Monte Carlo Methods, Sequential**; **Markov Chain Monte Carlo Algorithms**; **Variance reduction**; **Importance sampling including the bootstrap**; **Bootstrap Inference**; **Numerical integration**; **Monte Carlo Goodness of Fit Tests**; **Generation of Random Variables, Computer**; **Adaptive Monte Carlo Integration**.

References

[1] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer. The cross-entropy method for optimization. In V. Govindaraju and C.R. Rao, editors, *Handbook of Statistics*, volume 31:Machine Learning. North Holland, 2011.

[2] Q. Duan and D. P Kroese. Splitting for optimization. *Computers & Operations Research*, 73:119–131, 2016.

[3] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1994.

[4] M. D. Hoffman and A. Gelman. The No-U-Turn Sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[5] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. John Wiley & Sons, New York, 2011.

[6] H. J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag, New York, 1978.

[7] P. L'Ecuyer and R. Simard. TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33(4), 2007. Article 22.

[8] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics.* Cambridge University Press, Cambridge, 2004.

[9] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer-Verlag, New York, second edition, 2004.

[10] R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning.* Springer-Verlag, New York, 2004.

[11] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method.* John Wiley & Sons, New York, third edition, 2017.

[12] J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control.* John Wiley & Sons, New York, 2003.