Article type: Overview

Monte Carlo Methods

Article ID

Dirk P. Kroese

The University of Queensland

Reuven Y. Rubinstein

Technion, Israel Institute of Technology

Keywords

Monte Carlo, simulation, MCMC, estimation, optimization

Abstract

Many quantitative problems in science, engineering, and economics are nowadays solved via statistical sampling on a computer. Such *Monte Carlo methods* can be used in three different ways: (1) to generate random objects and processes in order to observe their behavior, (2) to estimate numerical quantities by repeated sampling, and (3) to solve complicated optimization problems through randomized algorithms.

The idea of using computers to carry out statistical sampling dates back to the very beginning of electronic computing. Stanislav Ulam and John Von Neumann pioneered this approach with the aim to study the behavior of neutron chain reactions. Nicholas Metropolis suggested the name *Monte Carlo* for this methodology, in reference to Ulam's fondness of games of chance [18].

This article gives an overview of modern Monte Carlo methods. Starting with random number and process generation, we show how Monte Carlo can be useful for both estimation and optimization purposes. We discuss a range of established Monte Carlo methods as well as some of the latest adaptive techniques, such as the cross-entropy method.

Generating Random Variables and Processes

At the heart of any Monte Carlo method is a **uniform random number generator**: a procedure that produces an infinite stream $U_1, U_2, \ldots$ of *random*[1] numbers on the interval (0,1).

[1]Since such numbers are usually produced via deterministic algorithms, they are not truly random. However, for most applications all that is required is that such *pseudo*-random numbers are statistically indistinguishable from genuine random numbers, which are uniformly distributed on the interval (0,1) and are independent of each other.

Random Number Generators Based on Linear Recurrences

The most common methods for generating uniform random numbers use simple linear recurrence relations. For a **linear congruential generator** (LCG) the output stream $U_1, U_2, \ldots$ is of the form $U_t = X_t/m$, where the *state* X_t satisfies the linear recursion

$$X_t = (aX_{t-1} + c) \bmod m\,, \quad t = 1, 2, \ldots\ . \tag{1}$$

Here, the integers m, a and c are called the *modulus*, the *multiplier*, and the *increment*, respectively. Applying the modulo-m operator in (1) means that $aX_{t-1} + c$ is divided by m, and the remainder is taken as the value for X_t.

An often-cited LCG is that of Lewis, Goodman, and Miller [15], who proposed the choice $a = 7^5 = 16807$, $c = 0$, and $m = 2^{31} - 1 = 2147483647$. Although this LCG passes many of the standard statistical tests for randomness and has been successfully used in many applications, its statistical properties no longer meet the requirements of modern Monte Carlo applications; see, for example, [14].

A **multiple-recursive generator** (MRG) of *order* k is a random number generator defined by a k-dimensional vector $\mathbf{X}_t = (X_{t-k+1}, \ldots, X_t)^\top$, whose components satisfy the linear recurrence

$$X_t = (a_1 X_{t-1} + \cdots + a_k X_{t-k}) \bmod m, \quad t = k, k+1, \ldots \tag{2}$$

for some modulus m and multipliers $\{a_i, i = 1, \ldots, k\}$. To yield fast algorithms, all but a few of the multipliers should be 0. When m is a large integer the output stream of random numbers is, similar to the LCG case, obtained via $U_t = X_t/m$. It is also possible to take a small modulus, in particular $m = 2$, so that the state of the generator is represented by a binary vector of length k. The output function for such **modulo 2 generators** is then typically of the form

$$U_t = \sum_{i=1}^{w} X_{tw+i-1} 2^{-i}$$

for some $w \leqslant k$, e.g., $w = 32$ or 64. Examples of modulo 2 generators are the **feedback shift register** generators, the most popular of which are the **Mersenne twisters**; see, for example, [16] and [17].

MRGs with excellent statistical properties can be implemented efficiently by combining several simpler MRGs. One of the most successful is L'Ecuyer's `MRG32k3a` generator; see [13].

Generating Random Variables

Generating a random variable X from an arbitrary (that is, not necessarily uniform) distribution invariably involves the following two steps:

1. Generate uniform random numbers $U_1, \ldots, U_k$ on $(0, 1)$ for some $k = 1, 2, \ldots$.
2. Return $X = g(U_1, \ldots, U_k)$, where g is some real-valued function.

Two of the most useful general procedures for generating random variables are the *inverse-transform* method and the *acceptance–rejection* method.

Inverse-Transform Method

Let X be a random variable with cumulative distribution function (cdf) F. Let F^{-1} denote the inverse of F and U be a uniform random number on (0,1) — we write this as $U \sim \mathsf{U}(0,1)$. Then,

$$\mathbb{P}(F^{-1}(U) \leqslant x) = \mathbb{P}(U \leqslant F(x)) = F(x) \ . \tag{3}$$

This leads to the inverse-transform method: to generate a random variable X with cdf F, draw $U \sim \mathsf{U}(0,1)$ and return $X = F^{-1}(U)$.

Acceptance–Rejection Method

The acceptance–rejection method is used to sample from a "difficult" probability density function (pdf) $f(x)$ by generating instead from an "easy" pdf $g(x)$ satisfying $f(x) \leqslant C\, g(x)$ for some constant $C \geqslant 1$ (for example, via the inverse-transform method), and then accepting or rejecting the drawn sample with a certain probability. More precisely, the algorithm is as follows.

Algorithm 1 (Acceptance–Rejection)

1. *Generate $X \sim g$; that is, draw X from pdf g.*
2. *Generate $U \sim \mathsf{U}(0,1)$, independently of X.*
3. *If $U \leqslant f(X)/(C\, g(X))$ output X; otherwise, return to Step 1.*

The **efficiency** of the acceptance–rejection method is defined as the probability of acceptance, which is $1/C$. The acceptance–rejection method can also be used to generate random vectors in $\mathbf{X} \in \mathbb{R}^d$ according to some pdf $f(\mathbf{x})$, although its efficiency is typically very small for dimensions $d \geqslant 10$; see, for example [22, Remark 2.5.1].

Generating Normal (or Gaussian) Random Variables

The **polar method** is based on the polar coordinate transformation $X = R\cos\Theta$, $Y = R\sin\Theta$, where $\Theta \sim \mathsf{U}(0, 2\pi)$ and $R \sim f_R$ are independent. Using standard transformation rules it follows that the joint pdf of X and Y satisfies

$$f_{X,Y}(x,y) = \frac{f_R(r)}{2\pi r} \ , \quad \text{with } r = \sqrt{x^2+y^2} \ ,$$

so that $f_X(x) = \int_0^\infty f_R(r)/(\pi\, r)\, \mathrm{d}y$. When $f_R(r) = r\, \mathrm{e}^{-r^2/2}$, then $f_X(x) = \mathrm{e}^{-x^2/2}/\sqrt{2\pi}$, which corresponds to the pdf of the standard normal distribution $\mathsf{N}(0,1)$. In this case R has the same distribution as $\sqrt{-2\ln U}$ with $U \sim \mathsf{U}(0,1)$. These observations lead to the *Box–Muller* method for generating standard normal random variables:

Algorithm 2 ($\mathsf{N}(0,1)$ Generator, Box–Muller Approach)

1. *Generate $U_1, U_2 \overset{\text{iid}}{\sim} \mathsf{U}(0,1)$.*
2. *Return two independent standard normal variables, X and Y, via*

$$\begin{aligned} X &= \sqrt{-2\ln U_1}\ \cos(2\pi U_2) \ , \\ Y &= \sqrt{-2\ln U_1}\ \sin(2\pi U_2) \ . \end{aligned} \tag{4}$$

Many other generation method may be found, for example, in [11].

Generating Random Vectors and Processes

A vector $\mathbf{X} = (X_1, \ldots, X_n)$ of random variables is called a **random vector**. More generally, a **random process** is a collection of random variables $\{X_t\}$. The techniques for generating such processes are as diverse as the random processes themselves. We mention a few important examples; see, also [11].

When $X_1, \ldots, X_n$ are *independent* random variables, with pdfs f_i, $i = 1, \ldots, n$, so that the joint pdf is $f(\mathbf{x}) = f_1(x_1) \cdots f_n(x_n)$, the random vector $\mathbf{X} = (X_1, \ldots, X_n)$ can be simply generated by drawing each component $X_i \sim f_i$ individually — for example, via the inverse-transform method or acceptance–rejection. For *dependent* components $X_1, \ldots, X_n$, we can represent the joint pdf $f(\mathbf{x})$ as

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = f_1(x_1)\, f_2(x_2 \,|\, x_1) \cdots f_n(x_n \,|\, x_1, \ldots, x_{n-1}) \,, \tag{5}$$

where $f_1(x_1)$ is the marginal pdf of X_1 and $f_k(x_k \,|\, x_1, \ldots, x_{k-1})$ is the conditional pdf of X_k given $X_1 = x_1, X_2 = x_2, \ldots, X_{k-1} = x_{k-1}$. Provided the conditional pdfs are known, one can generate $\mathbf{X}$ by first first generating X_1, then, given $X_1 = x_1$, generate X_2 from $f_2(x_2 \,|\, x_1)$, and so on, until generating X_n from $f_n(x_n \,|\, x_1, \ldots, x_{n-1})$.

The latter method is particularly applicable for generating Markov chains. A **Markov chain** is a stochastic process $\{X_t, t = 0, 1, 2, \ldots\}$ which satisfies the *Markov property*; meaning that for all t and s the conditional distribution of X_{t+s} given $X_u, u \leqslant t$ is the same as that of X_{t+s} given only X_t. A direct consequence of the Markov property is that Markov chains can be generated *sequentially*: $X_0, X_1, \ldots$, as expressed in the following generic recipe.

Algorithm 3 (Generating a Markov Chain)

1. *Draw X_0 from its distribution. Set $t = 0$.*
2. *Draw X_{t+1} from the conditional distribution of X_{t+1} given X_t.*
3. *Set $t = t + 1$ and repeat from Step 2.*

In many cases of interest the chain is **time-homogeneous**, meaning that the conditional distribution of $(X_{t+s} \,|\, X_t)$ only depends on s.

Diffusion processes are random processes that satisfy the Markov property and have continuous paths and continuous time parameters. The principal example is the **Wiener process** (Brownian motion). In addition to being a time-homogeneous Markov process, the Wiener process is *Gaussian*; that is, all its finite-dimensional distributions are multi-variate normal. In particular, $W_t \sim \mathsf{N}(0, t)$ for all $t \geqslant 0$.

The Wiener process can be viewed as a continuous version of a random walk process. The basic generation algorithm below uses the Markovian and Gaussian properties of the Wiener process.

Algorithm 4 (Generating a Wiener Process) *Let $0 = t_0 < t_1 < t_2 < \cdots < t_n$ be the set of distinct times for which the outcomes $\{W_{t_k}, k = 0, 1 \ldots, n\}$ of the Wiener process is required.*

Draw $Z_1, \ldots, Z_n \overset{\text{iid}}{\sim} \mathsf{N}(0, 1)$ and output $W_{t_k} = \sum_{i=1}^{k} \sqrt{t_k - t_{k-1}}\, Z_i, \quad k = 1, \ldots, n \,.$

The Wiener process plays a central role in probability and forms the basis of other diffusion processes. These are often formulated via a **stochastic differential equation** (SDE), which is an expression of the form

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}W_t\ , \tag{6}$$

where $\{W_t, t \geqslant 0\}$ is a Wiener process and $a(x,t)$ and $b(x,t)$ are deterministic functions. The coefficient (function) a is called the **drift** and b^2 is called the **diffusion** coefficient.

A simple technique for approximately simulating such diffusion processes is **Euler's method**; see, for example, [9]. The idea is to replace the SDE with the stochastic difference equation

$$Y_{k+1} = Y_k + a(Y_k, kh)\,h + b(Y_k, kh)\sqrt{h}\,Z_k\ , \tag{7}$$

where $Z_1, Z_2, \ldots \sim_{\text{iid}} \mathsf{N}(0,1)$. For a small step size h the process $\{Y_k, k = 0,1,2,\ldots\}$ approximates the process $\{X_t, t \geqslant 0\}$ in the sense that $Y_k \approx X_{kh}$, $k = 0,1,2,\ldots$.

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a generic method for *approximate* sampling from an arbitrary distribution. The main idea is to generate a Markov chain whose limiting distribution is equal to the desired distribution.

The MCMC method originates from Metropolis et al. [19] and applies to the following setting. Suppose that we wish to generate samples from an arbitrary multidimensional pdf

$$f(\mathbf{x}) = \frac{p(\mathbf{x})}{\mathcal{Z}}, \quad \mathbf{x} \in \mathscr{X}\ ,$$

where $p(\mathbf{x})$ is a known positive function and $\mathcal{Z}$ is a known or unknown normalizing constant. Let $q(\mathbf{y}\,|\,\mathbf{x})$ be a **proposal** or **instrumental** density: a Markov transition density describing how to go from state $\mathbf{x}$ to $\mathbf{y}$. Similar to the acceptance–rejection method, the Metropolis–Hastings algorithm is based on the following "trial-and-error" strategy.

Algorithm 5 (Metropolis–Hastings Algorithm) *To sample from a density $f(\mathbf{x})$ known up to a normalizing constant, initialize with some $\mathbf{X}_0$ for which $f(\mathbf{X}_0) > 0$. Then, for each $t = 0,1,2,\ldots,T-1$ execute the following steps:*

1. *Given the current state $\mathbf{X}_t$, generate $\mathbf{Y} \sim q(\mathbf{y}\,|\,\mathbf{X}_t)$.*

2. *Generate $U \sim \mathsf{U}(0,1)$ and deliver*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y} & \textit{if } U \leqslant \alpha(\mathbf{X}_t, \mathbf{Y}) \\ \mathbf{X}_t & \textit{otherwise}\ , \end{cases} \tag{8}$$

where

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{f(\mathbf{y})\,q(\mathbf{x}\,|\,\mathbf{y})}{f(\mathbf{x})\,q(\mathbf{y}\,|\,\mathbf{x})}, 1\right\}\ . \tag{9}$$

The probability $\alpha(\mathbf{x},\mathbf{y})$ is called the **acceptance probability**. Depending on the choice of the proposal density the algorithm can overcome the limitations of the acceptance–rejection method for sampling from high-dimensional densities. However, unlike the acceptance–rejection method, the algorithms produces *dependent* samples. Moreover, the algorithm may require a substantial *burn-in* period to reach stationarity of the Markov chain. Note that in (9) we may replace f by p.

If the proposal function $q(\mathbf{y} \,|\, \mathbf{x})$ does not depend on $\mathbf{x}$, that is, $q(\mathbf{y} \,|\, \mathbf{x}) = g(\mathbf{y})$ for some pdf $g(\mathbf{y})$, then the acceptance probability is

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

and Algorithm 5 is referred to as the **independence sampler**.

If the proposal is symmetric, that is, $q(\mathbf{y} \,|\, \mathbf{x}) = q(\mathbf{x} \,|\, \mathbf{y})$, then the acceptance probability (9) is

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{f(\mathbf{y})}{f(\mathbf{x})},\ 1\right\}, \tag{10}$$

and Algorithm 5 is referred to as the **random walk sampler**. An example of a random walk sampler is when $\mathbf{Y} = \mathbf{X}_t + \sigma\mathbf{Z}$ in Step 1 of Algorithm 5, where $\mathbf{Z}$ is typically generated from some spherically symmetrical distribution (in the continuous case), such as $\mathsf{N}(\mathbf{0}, I)$.

The **Gibbs sampler** can be viewed as a particular instance of the Metropolis–Hastings algorithm for generating n-dimensional random vectors [6]. The distinguishing feature of the Gibbs sampler is that the underlying Markov chain is constructed from a sequence of conditional distributions, in either a deterministic or random fashion.

Suppose that we wish to sample a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ according to a target pdf $f(\mathbf{x})$. Let $f(x_i \,|\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represent the conditional pdf of the i-th component, X_i, given the other components $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$.

Algorithm 6 (Gibbs Sampler) *Given an initial state $\mathbf{X}_0$, iterate the following steps for $t = 0, 1, \ldots$.*

1. *For a given $\mathbf{X}_t$, generate $\mathbf{Y} = (Y_1, \ldots, Y_n)$ as follows:*
   - *(a) Draw Y_1 from the conditional pdf $f(x_1 \,|\, X_{t,2}, \ldots, X_{t,n})$.*
   - *(b) Draw Y_i from $f(x_i \,|\, Y_1, \ldots, Y_{i-1}, X_{t,i+1}, \ldots, X_{t,n})$, $\quad i = 2, \ldots, n-1$.*
   - *(c) Draw Y_n from $f(x_n \,|\, Y_1, \ldots, Y_{n-1})$.*
2. *Let $\mathbf{X}_{t+1} = \mathbf{Y}$.*

Algorithm 6 presents a **systematic** (coordinatewise) Gibbs sampler. That is, the components of vector $\mathbf{X}$ are updated in the coordinatewise order $1 \to 2 \to \cdots \to n$. The completion of all the conditional sampling steps in the specified order is called a **cycle**. Alternative updating of the components of vector $\mathbf{X}$ are possible. In the **reversible Gibbs sampler** a single cycle consists of the coordinatewise updating

$$1 \to 2 \to \cdots \to n-1 \to n \to n-1 \to \cdots \to 2 \to 1\ .$$

In the **random sweep/scan Gibbs sampler** a single cycle can either consist of one or several coordinates selected uniformly from the integers $1, \ldots, n$, or a random permutation $\pi_1 \to \pi_2 \to \cdots \to \pi_n$ of all coordinates.

Monte Carlo for Estimation

Estimation of Expectations

Suppose the data $Y_1, \ldots, Y_N$ from a simulation experiment are independent and identically distributed according to some known or unknown discrete or continuous pdf f. Often such data are obtained by executing N independent runs of the simulation, producing output Y_i for the i-th run. Suppose the aim of the simulation is to estimate the performance measure $\ell = \mathbb{E}Y$, with $Y \sim f$. Assuming $|\ell| < \infty$, an unbiased estimator for ℓ is the *sample mean* of the $\{Y_i\}$; that is,

$$\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i \ . \tag{11}$$

Provided that the variance of Y, say σ^2, is finite, $\bar{Y}$ approximately has a $\mathsf{N}(\ell, \sigma^2/N)$ distribution for large N (an immediate consequence of the central limit theorem). If σ^2 is unknown, it can be estimated without bias via the *sample variance* of the $\{Y_i\}$:

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Y_i - \bar{Y})^2 \ , \tag{12}$$

which (by the law of large numbers) tends to σ^2 as $N \to \infty$. This leads to an approximate $1-\alpha$ *confidence interval* for ℓ:

$$\left(\bar{Y} - z_{1-\alpha/2}\frac{S}{\sqrt{N}}, \quad \bar{Y} + z_{1-\alpha/2}\frac{S}{\sqrt{N}}\right) \ , \tag{13}$$

where z_γ denotes the γ-quantile of the $\mathsf{N}(0,1)$ distribution.

Instead of specifying the confidence interval, one often reports only the *estimated standard error*: $S/\sqrt{N}$, or the *estimated relative error* $S/(\bar{Y}\sqrt{N})$.

The basic estimation procedure for independent data is summarized below. The procedure is sometimes referred to as **crude Monte Carlo** (CMC).

Algorithm 7 (Crude Monte Carlo for Independent Data)

1. *Generate* $Y_1, \ldots, Y_N \overset{\text{iid}}{\sim} f$ *(for example, from independent simulation runs).*
2. *Calculate the point estimate* $\bar{Y}$ *and confidence interval* (13) *of* $\ell = \mathbb{E}Y$.

It is often the case that the output Y is a function of some underlying random vector or stochastic process; that is, $Y = H(\mathbf{X})$, where H is a real-valued performance function and $\mathbf{X}$ is a random vector or process. The beauty of Monte Carlo for estimation is that (13) holds regardless of the dimension of $\mathbf{X}$.

Example 1 *In* **Monte Carlo integration**, *simulation is used to evaluate complicated integrals. Consider, for example, the integral*

$$\ell = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sqrt{|x_1 + x_2 + x_3|}\ \mathrm{e}^{-(x_1^2+x_2^2+x_3^2)/2}\,\mathrm{d}x_1\,\mathrm{d}x_2\,\mathrm{d}x_3 \ .$$

Defining $Y = |X_1 + X_2 + X_3|^{1/2}(2\pi)^{3/2}$, *with* $X_1, X_2, X_3 \overset{\text{iid}}{\sim} \mathsf{N}(0,1)$, *we can write* $\ell = \mathbb{E}Y$. *Using the following* MATLAB *program, with a sample size of* $N = 10^6$, *we obtained an estimate* $\bar{Y} = 17.04$ *with a* 95% *confidence interval* $(17.026, 17.054)$.

```
c = (2*pi)^(3/2);
H = @(x) c*sqrt(abs(sum(x,2)));
N = 10^6; z = 1.96;
x = randn(N,3); y = H(x);
mY = mean(y); sY = std(y); RE = sY/mY/sqrt(N);
fprintf('Estimate = %g, CI = (%g, %g)\n', ...
                     mY, mY*(1- z*RE), mY*(1 + z*RE))
```

Variance Reduction

The estimation of performance measures in Monte Carlo simulation can be made more efficient by utilizing known information about the simulation model. Variance reduction techniques include antithetic variables, control variables, importance sampling, conditional Monte Carlo, and stratified sampling; see, for example, [11, Chapter 9]. We shall only deal with the first three techniques.

One of the simplest approaches is to estimate $\ell = \mathbb{E}Y$ is to generate an independent sequence of **antithetic pairs** $(Y_1, Y_1^*), \ldots, (Y_{N/2}, Y_{N/2}^*)$, where each Y_i and Y_i^* are distributed as Y but are *negatively correlated*. The **antithetic estimator**

$$\widehat{\ell}^{(a)} = \frac{1}{N} \sum_{k=1}^{N/2} \{Y_k + Y_k^*\} \,, \tag{14}$$

is an unbiased estimator of $\ell = \mathbb{E}Y$ with a variance that is a factor $(1 + \varrho_{Y,Y^*})$ of that of the CMC estimator, where ϱ_{Y,Y^*} is the correlation between Y and Y^*.

In general, the output of a simulation run is of the form $Y = h(\mathbf{U})$, where h is a real-valued function and $\mathbf{U} = (U_1, U_2, \ldots)$ is a random vector of iid $\mathsf{U}(0,1)$ random variables. Suppose that $\mathbf{U}^*$ is another vector of iid $\mathsf{U}(0,1)$ random variables which is dependent on $\mathbf{U}$ and for which Y and $Y^* = h(\mathbf{U}^*)$ are negatively correlated. Then (Y, Y^*) is an antithetic pair. In particular, if h is a monotone function in each of its components, then the choice $\mathbf{U}^* = \mathbf{1} - \mathbf{U}$, where $\mathbf{1}$ is the vector of 1s, yields an antithetic pair.

Another useful variance reduction approach is to use **control variables**. Let Y be the output of a simulation run. A random variable $\widetilde{Y}$, obtained from the same simulation run, is called a control variable for Y if Y and $\widetilde{Y}$ are correlated (negatively or positively) and the expectation of $\widetilde{Y}$ is known. The use of control variables for variance reduction is based on the following observation.

Theorem 1 (Control Variable Estimation) *Let $Y_1, \ldots, Y_N$ be the output of N independent simulation runs, and let $\widetilde{Y}_1, \ldots, \widetilde{Y}_N$ be the corresponding control variables, with $\mathbb{E}\widetilde{Y}_k = \widetilde{\ell}$ known. Let $\varrho_{Y,\widetilde{Y}}$ be the correlation coefficient between each Y_k and $\widetilde{Y}_k$. For each $\alpha \in \mathbb{R}$ the (linear) estimator*

$$\widehat{\ell}^{(c)} = \frac{1}{N} \sum_{k=1}^{N} \left[Y_k - \alpha \left(\widetilde{Y}_k - \widetilde{\ell} \right) \right] \tag{15}$$

is an unbiased estimator for $\ell = \mathbb{E}Y$. The minimal variance of $\widehat{\ell}^{(c)}$ is

$$\mathrm{Var}(\widehat{\ell}^{(c)}) = \frac{1}{N} (1 - \varrho^2_{Y,\widetilde{Y}}) \mathrm{Var}(Y) \tag{16}$$

which is obtained for $\alpha = \varrho_{Y,\widetilde{Y}} \sqrt{\mathrm{Var}(Y)/\mathrm{Var}(\widetilde{Y})}$.

Usually the optimal α in (16) is unknown, but it can be easily estimated from the sample covariance matrix of the $\{(Y_k, \widetilde{Y}_k)\}$.

One of the most important variance reduction techniques is **importance sampling**. This technique is especially useful for the estimation of very small probabilities. The standard setting is the estimation of a quantity

$$\ell = \mathbb{E}_f H(\mathbf{X}) = \int H(\mathbf{x}) \, f(\mathbf{x}) \, \mathrm{d}\mathbf{x} \,, \tag{17}$$

where H is a real-valued function and f the probability density of a random vector $\mathbf{X}$, called the **nominal pdf**. The subscript f is added to the expectation operator to indicate that it is taken with respect to the density f.

Let g be another probability density such that $H\,f$ is **dominated** by g. That is, $g(\mathbf{x}) = 0 \;\Rightarrow\; H(\mathbf{x})\,f(\mathbf{x}) = 0$. Using the density g we can represent ℓ as

$$\ell = \int H(\mathbf{x})\,\frac{f(\mathbf{x})}{g(\mathbf{x})}\,g(\mathbf{x})\,\mathrm{d}\mathbf{x} = \mathbb{E}_g H(\mathbf{X})\,\frac{f(\mathbf{X})}{g(\mathbf{X})}\ . \tag{18}$$

Consequently, if $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\text{iid}} g$, then

$$\widehat{\ell} = \frac{1}{N}\sum_{k=1}^{N} H(\mathbf{X}_k)\,\frac{f(\mathbf{X}_k)}{g(\mathbf{X}_k)} \tag{19}$$

is an unbiased estimator of ℓ. This estimator is called the **importance sampling estimator** and g is called the importance sampling density. The ratio of densities, $W(\mathbf{x}) = f(\mathbf{x})/g(\mathbf{x})$, is called the **likelihood ratio**. The importance sampling algorithm is as follows.

Algorithm 8 (Importance Sampling Estimation)

1. *Select an importance sampling density g that dominates Hf.*
2. *Generate $\mathbf{X}_1, \ldots, \mathbf{X}_N \overset{\text{iid}}{\sim} g$ and let $Y_i = H(\mathbf{X}_i)f(\mathbf{X}_i)/g(\mathbf{X}_i),\ i = 1, \ldots, N$.*
3. *Estimate ℓ via $\widehat{\ell} = \bar{Y}$ and determine an approximate $1-\alpha$ confidence interval as*
$$\left(\widehat{\ell} - z_{1-\alpha/2}\frac{S}{\sqrt{N}},\ \widehat{\ell} + z_{1-\alpha/2}\frac{S}{\sqrt{N}}\right),$$
where z_γ denotes the γ-quantile of the $\mathsf{N}(0,1)$ distribution and S is the sample standard deviation of $Y_1, \ldots, Y_N$.

The main difficulty in importance sampling is how to choose the importance sampling distribution. A poor choice of g may seriously affect the accuracy of the estimate and the confidence intervals. The theoretically optimal choice g^* for the importance sampling density minimizes the variance of $\widehat{\ell}$, and is therefore the solution to the functional minimization program

$$\min_g \mathrm{Var}_g\left(H(\mathbf{X})\,\frac{f(\mathbf{X})}{g(\mathbf{X})}\right)\ . \tag{20}$$

It is not difficult to show that if either $H(\mathbf{x}) \geqslant 0$ or $H(\mathbf{x}) \leqslant 0$ for all $\mathbf{x}$, then

$$g^*(\mathbf{x}) = \frac{H(\mathbf{x})\,f(\mathbf{x})}{\ell}\ , \tag{21}$$

in which case $\mathrm{Var}_{g^*}(\widehat{\ell}) = \mathrm{Var}_{g^*}(H(\mathbf{X})W(\mathbf{X})) = \mathrm{Var}_{g^*}(\ell) = 0$, so that the estimator $\widehat{\ell}$ is *constant* under g^*. An obvious difficulty is that the evaluation of the optimal importance sampling density g^* is usually not possible, since $g^*(\mathbf{x})$ in (21) depends on the unknown quantity ℓ. Nevertheless, one can typically choose a good importance sampling density g "close" to the minimum variance density g^*.

One of the main considerations for choosing a good importance sampling pdf is that the estimator (19) should have finite variance. This is equivalent to the requirement that

$$\mathbb{E}_g H^2(\mathbf{X})\frac{f^2(\mathbf{X})}{g^2(\mathbf{X})} = \mathbb{E}_f H^2(\mathbf{X})\frac{f(\mathbf{X})}{g(\mathbf{X})} < \infty\ . \tag{22}$$

This suggests that g should not have lighter tails than f, and that, preferably, the likelihood ratio, f/g, should be bounded.

Variance Minimization Method

When the pdf f belongs to some parametric family of distributions, it is often convenient to choose the importance sampling distribution from the *same* family. In particular, suppose that $f(\cdot;\boldsymbol{\theta})$ belongs to the family

$$\{f(\cdot;\boldsymbol{\eta}),\ \boldsymbol{\eta}\in\Theta\}\ .$$

Then, the problem of finding an optimal importance sampling density in this class reduces to the following *parametric* minimization problem

$$\min_{\boldsymbol{\eta}\in\Theta} \mathrm{Var}_{\boldsymbol{\eta}}\left(H(\mathbf{X})\,W(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})\right)\ , \tag{23}$$

where $W(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta}) = f(\mathbf{X};\boldsymbol{\theta})/f(\mathbf{X};\boldsymbol{\eta})$. We call $\boldsymbol{\theta}$ the **nominal parameter** and $\boldsymbol{\eta}$ the **reference parameter vector** or **tilting vector**. Since under any $f(\cdot;\boldsymbol{\eta})$ the expectation of $H(\mathbf{X})\,W(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})$ is ℓ, the optimal solution of (23) coincides with that of

$$\min_{\boldsymbol{\eta}\in\Theta} V(\boldsymbol{\eta})\ , \tag{24}$$

where

$$V(\boldsymbol{\eta}) = \mathbb{E}_{\boldsymbol{\eta}} H^2(\mathbf{X})\,W^2(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta}) = \mathbb{E}_{\boldsymbol{\theta}} H^2(\mathbf{X})\,W(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})\ . \tag{25}$$

We call either of the equivalent problems (23) and (24) the **variance minimization** (VM) problem; and we call the parameter vector ${}_*\boldsymbol{\eta}$ that minimizes the programs (23) – (24) the **VM-optimal reference parameter vector**. The VM problem can be viewed as a stochastic optimization problem, and can be approximately solved via Monte Carlo simulation by considering the stochastic counterpart of (24) – (25):

$$\min_{\boldsymbol{\eta}\in\Theta} \widehat{V}(\boldsymbol{\eta})\ , \tag{26}$$

where

$$\widehat{V}(\boldsymbol{\eta}) = \frac{1}{N}\sum_{k=1}^{N} H^2(\mathbf{X}_k)\,W(\mathbf{X}_k;\boldsymbol{\theta},\boldsymbol{\eta})\ , \tag{27}$$

and $\mathbf{X}_1,\ldots,\mathbf{X}_N \sim_{\mathrm{iid}} f(\cdot;\boldsymbol{\theta})$. This problem can be solved via standard numerical optimization techniques, such as the Newton–Raphson method. This gives the following modification of Algorithm 8.

Algorithm 9 (Variance Minimization Method)

1. *Select a parameterized family of importance sampling densities* $\{f(\cdot;\boldsymbol{\eta})\}$*.*
2. *Generate a pilot sample* $\mathbf{X}_1,\ldots,\mathbf{X}_N \overset{\text{iid}}{\sim} f(\cdot;\boldsymbol{\theta})$*, and determine the solution* ${}_*\widehat{\boldsymbol{\eta}}$ *to the variance minimization problem* (26)*.*
3. *Generate* $\mathbf{X}_1,\ldots,\mathbf{X}_{N_1} \overset{\text{iid}}{\sim} f(\cdot;{}_*\widehat{\boldsymbol{\eta}})$ *and let* $Y_i = H(\mathbf{X}_i)f(\mathbf{X}_i;\boldsymbol{\theta})/f(\mathbf{X}_i;{}_*\widehat{\boldsymbol{\eta}}), i = 1,\ldots,N_1$*.*
4. *Estimate* ℓ *via* $\widehat{\ell} = \bar{Y}$ *and determine an approximate* $1-\alpha$ *confidence interval as*
$$\left(\widehat{\ell} - z_{1-\alpha/2}\frac{S}{\sqrt{N_1}},\ \widehat{\ell} + z_{1-\alpha/2}\frac{S}{\sqrt{N_1}}\right),$$
where z_γ *denotes the* γ*-quantile of the* $\mathsf{N}(0,1)$ *distribution and* S *is the sample standard deviation of* $Y_1,\ldots,Y_{N_1}$*.*

Cross-Entropy Method

An alternative approach to the VM method for choosing an "optimal" importance sampling distribution is based on the Kullback–Leibler cross-entropy distance, or simply **cross-entropy** (CE) distance. The CE distance between two continuous pdfs g and h is given by

$$\begin{aligned}\mathcal{D}(g,h) &= \mathbb{E}_g \ln\frac{g(\mathbf{X})}{h(\mathbf{X})} = \int g(\mathbf{x})\ln\frac{g(\mathbf{x})}{h(\mathbf{x})}\,\mathrm{d}\mathbf{x}\\ &= \int g(\mathbf{x})\ln g(\mathbf{x})\,\mathrm{d}\mathbf{x} - \int g(\mathbf{x})\ln h(\mathbf{x})\,\mathrm{d}\mathbf{x}\ .\end{aligned} \tag{28}$$

For discrete pdfs replace the integrals with the corresponding sums. Observe that, by Jensen's inequality, $\mathcal{D}(g,h) \geqslant 0$, with equality if and only if $g = h$. The CE distance is sometimes called the Kullback–Leibler *divergence*, because it is not symmetric, that is, $\mathcal{D}(g,h) \neq \mathcal{D}(h,g)$ for $g \not\equiv h$.

The idea of the CE method is to choose the importance sampling density, h say, such that the CE distance between the optimal importance sampling density g^* in (21) and h, is minimal. We call this the **CE-optimal pdf**. This pdf solves the *functional* optimization program $\min_h \mathcal{D}(g^*,h)$. If we optimize over all densities h, then it is immediate that the CE-optimal pdf coincides with the VM-optimal pdf g^*.

As with the VM approach in (23) and (24), we shall restrict ourselves to a parametric family of densities $\{f(\cdot;\boldsymbol{\eta}), \boldsymbol{\eta}\in\Theta\}$ that contains the nominal density $f(\cdot;\boldsymbol{\theta})$. Moreover, without any loss of generality, we only consider positive functions H. The CE method now aims to solve the *parametric* optimization problem

$$\min_{\boldsymbol{\eta}\in\Theta} \mathcal{D}\left(g^*, f(\cdot;\boldsymbol{\eta})\right)\ . \tag{29}$$

The optimal solution coincides with that of

$$\max_{\boldsymbol{\eta}\in\Theta} D(\boldsymbol{\eta})\ , \tag{30}$$

where

$$D(\boldsymbol{\eta}) = \mathbb{E}_{\boldsymbol{\theta}} H(\mathbf{X})\ \ln f(\mathbf{X};\boldsymbol{\eta})\ . \tag{31}$$

Similar to the VM program (24), we call either of the equivalent programs (29) and (30) the **CE program**; and we call the parameter vector $\boldsymbol{\eta}^*$ that minimizes the program (29) and (30) the **CE-optimal reference parameter**.

Similar to (26) we can estimate $\boldsymbol{\eta}^*$ via the stochastic counterpart method as the solution of the stochastic program

$$\max_{\boldsymbol{\eta}} \widehat{D}(\boldsymbol{\eta}) = \max_{\boldsymbol{\eta}} \frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k) \, \ln f(\mathbf{X}_k; \boldsymbol{\eta}) \,, \tag{32}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\text{iid}} f(\cdot; \boldsymbol{\theta})$.

In typical applications the function $\widehat{D}$ in (32) is convex and differentiable with respect to $\boldsymbol{\eta}$. In such cases the solution of (32) may be obtained by solving (with respect to $\boldsymbol{\eta}$) the following system of equations:

$$\frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k) \, \nabla \ln f(\mathbf{X}_k; \boldsymbol{\eta}) = \mathbf{0} \,, \tag{33}$$

where the gradient is with respect to $\boldsymbol{\eta}$. Various numerical and theoretical studies have shown that the solutions to the VM and CE programs are qualitatively similar. The main advantage of the CE approach over the VM approach is that the solution to (32) (or (33)) can often be found *analytically*; see, for example, [3].

Monte Carlo for Optimization

In this section we discuss several Monte Carlo optimization methods. Such randomized algorithms can be useful for solving optimization problems with many local optima and complicated constraints, possibly involving a mix of continuous and discrete variables. Randomized algorithms are also used to solve *noisy* optimization problems, in which the objective function is unknown and has to be obtained via Monte Carlo simulation.

Suppose we have a minimization problem on $\mathscr{X} \subseteq \mathbb{R}^n$ of the form

$$\min_{\mathbf{x} \in \mathscr{X}} S(\mathbf{x}) \,, \tag{34}$$

where S is an unknown function of the form $\mathbb{E}\widetilde{S}(\mathbf{x}, \boldsymbol{\xi})$, with $\boldsymbol{\xi}$ a random vector and $\widetilde{S}$ a known function. A typical example is where $S(\mathbf{x})$ is the (usually unknown) expected performance measure from a Monte Carlo simulation. Such a problem is said to be a **noisy** optimization problem, as typically only realizations of $\widetilde{S}(\mathbf{x}, \boldsymbol{\xi})$ can be observed.

Because the gradient ∇S is unknown, one cannot directly apply classical optimization methods. The **stochastic approximation method** mimics the classical gradient descent method by replacing a deterministic gradient with a random approximation. More generally, one can approximate a subgradient instead of the gradient. It is assumed that an estimate of the gradient of S is available at any point $\mathbf{x} \in \mathscr{X}$. We denote such an estimate by $\widehat{\nabla S}(\mathbf{x})$. There are several established ways of obtaining $\widehat{\nabla S}(\mathbf{x})$. These include the finite difference method, infinitesimal perturbation analysis, the score function method, and the method of weak derivatives; see, for example, [7, Chapter 7].

In direct analogy to gradient descent methods, the stochastic approximation method produces a sequence of iterates, starting with some $\mathbf{x}_1 \in \mathscr{X}$, via

$$\mathbf{x}_{t+1} = \Pi_{\mathscr{X}} \left(\mathbf{x}_t - \beta_t \, \widehat{\nabla S}(\mathbf{x}_t) \right) \,, \tag{35}$$

where $\beta_1, \beta_2, \ldots$ is a sequence of strictly positive step sizes and $\Pi_{\mathscr{X}}$ is a projection operator that takes a point in $\mathbb{R}^n$ and returns a closest (typically in Euclidean distance) point in $\mathscr{X}$, ensuring that iterates remain feasible. That is, for any $\mathbf{y} \in \mathbb{R}^n$, $\Pi_{\mathscr{X}}(\mathbf{y}) \in \operatorname{argmin}_{\mathbf{z} \in \mathscr{X}} \|\mathbf{z} - \mathbf{y}\|$. Naturally, if $\mathscr{X} = \mathbb{R}^n$, then $\Pi_{\mathscr{X}}(\mathbf{y}) = \mathbf{y}$. A generic stochastic approximation algorithm is as follows.

Algorithm 10 (Stochastic Approximation)

1. *Initialize* $\mathbf{x}_1 \in \mathscr{X}$. *Set* $t = 1$.
2. *Obtain an estimated gradient* $\widehat{\nabla S}(\mathbf{x}_t)$ *of* S *at* $\mathbf{x}_t$.
3. *Determine a step size* β_t.
4. *Set* $\mathbf{x}_{t+1} = \Pi_{\mathscr{X}}\left(\mathbf{x}_t - \beta_t \, \widehat{\nabla S}(\mathbf{x}_t)\right)$.
5. *If a stopping criterion is met, stop; otherwise, set* $t = t + 1$ *and repeat from Step 2.*

There are many theorems on the convergence of stochastic approximation algorithms. In particular, for an arbitrary deterministic positive sequence $\beta_1, \beta_2, \ldots$ such that

$$\sum_{t=1}^{\infty} \beta_t = \infty, \quad \sum_{t=1}^{\infty} \beta_t^2 < \infty \ ,$$

the random sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots$ converges in the mean square sense to the minimizer $\mathbf{x}^*$ of $S(\mathbf{x})$ under certain regularity conditions. See, for example, [12].

When $\widehat{\nabla S}(\mathbf{x}_t)$ is an *unbiased* estimator of $\nabla S(\mathbf{x}_t)$ in (35) the stochastic approximation Algorithm 10 is referred to as the **Robbins–Monro** algorithm. When finite differences are used to estimate $\widehat{\nabla S}(\mathbf{x}_t)$ the resulting algorithm is known as the **Kiefer–Wolfowitz** algorithm.

An alternative approach to stochastic approximation is the **stochastic counterpart method** (also called **sample average approximation**). The idea is to replace the noisy optimization problem (34) with

$$\min_{\mathbf{x} \in \mathbb{R}^n} \widehat{S}(\mathbf{x}) \,, \tag{36}$$

where

$$\widehat{S}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \widetilde{S}(\mathbf{x}, \boldsymbol{\xi}_i)$$

is a sample average estimator of $S(\mathbf{x}) = \mathbb{E}\widetilde{S}(\mathbf{x}, \boldsymbol{\xi})$ on the basis of N iid samples $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N$.

A solution $\widehat{\mathbf{x}}^*$ to this sample average version is taken to be an estimator of a solution $\mathbf{x}^*$ to the original problem (34). Note that (36) is a *deterministic* optimization problem to which any of the standard deterministic optimization methods could apply.

Simulated annealing is a Markov chain Monte Carlo technique for approximately locating a global maximum of a given density $f(\mathbf{x})$. The idea is to create a sequence of points $\mathbf{X}_1, \mathbf{X}_2, \ldots$ that are approximately distributed according to pdfs $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$ with $f_t(\mathbf{x}) \propto f(\mathbf{x})^{1/T_t}$, where $T_1, T_2, \ldots$ is a sequence of **temperatures** (known as the **annealing schedule**) that decreases to 0. If each $\mathbf{X}_t$ were sampled *exactly* from $f(\mathbf{x})^{1/T_t}$, then X_t would converge to a global maximum of $f(\mathbf{x})$ as $T_t \to 0$. However, in practice sampling is *approximate* and convergence to a global maximum is not assured. A high-level simulated annealing algorithm is as follows.

Algorithm 11 (Simulated Annealing)

1. *Choose a starting state* $\mathbf{X}_0$ *and an initial temperature* T_0. *Set* $t = 1$.
2. *Select a temperature* $T_t \leqslant T_{t-1}$ *according to the annealing schedule.*
3. *Approximately generate a new state* $\mathbf{X}_t$ *from* $f_t(\mathbf{x}) \propto (f(\mathbf{x}))^{1/T_t}$.
4. *Set* $t = t + 1$ *and repeat from Step 2 until stopping.*

The most common application for simulated annealing is in optimization. In particular, consider the minimization problem (34) for some deterministic real-valued function $S(\mathbf{x})$. Define the **Boltzmann pdf** as

$$f(\mathbf{x}) \propto \mathrm{e}^{-S(\mathbf{x})}, \quad \mathbf{x} \in \mathscr{X} \; .$$

For $T > 0$ close to 0 the global maximum of $f(\mathbf{x})^{1/T} \propto \exp(-S(\mathbf{x})/T)$ is close to the global minimum of $S(\mathbf{x})$. Hence, by applying simulated annealing to the Boltzmann pdf, one can also minimize $S(\mathbf{x})$. Maximization problems can be handled in a similar way, by using a Boltzmann pdf $f(\mathbf{x}) \propto \exp(S(\mathbf{x}))$. Note that this may not define a valid pdf if the exponential terms are not normalizable.

There are many different ways to implement simulated annealing algorithms, depending on (1) the choice of Markov chain Monte Carlo sampling algorithm, (2) the length of the Markov chain between temperature updates, and (3) the annealing schedule. A popular annealing schedule is **geometric cooling**, where $T_t = \beta T_{t-1}$, $t = 1, 2, \ldots$, for a given initial temperature T_0 and a **cooling factor** $\beta \in (0, 1)$. Appropriate values for T_0 and β are problem-dependent, and this has traditionally required tuning on the part of the user. Theoretical results on adaptive tuning schemes may be found, for example, in [23].

The following algorithm describes a popular simulated annealing framework for minimization, which uses a random walk sampler; that is, a Metropolis–Hastings sampler with a symmetric proposal distribution. Note that the temperature is updated after a *single* step of the Markov chain.

Algorithm 12 (Simulated Annealing for Minimization)

1. *Initialize the starting state* $\mathbf{X}_0$ *and temperature* T_0. *Set* $t = 1$.
2. *Select a temperature* $T_t \leqslant T_{t-1}$ *from the annealing schedule.*
3. *Generate a candidate state* $\mathbf{Y}$ *from the symmetric proposal density* $q(\mathbf{Y} \,|\, \mathbf{X}_t) = q(\mathbf{X}_t \,|\, \mathbf{Y})$.
4. *Compute the acceptance probability*

   $$\alpha(\mathbf{X}_t, \mathbf{Y}) = \min\left\{\mathrm{e}^{-\frac{(S(\mathbf{Y}) - S(\mathbf{X}_t))}{T_t}},\, 1\right\} \; .$$

   Generate $U \sim \mathsf{U}(0,1)$ *and set*

   $$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y} & \textit{if } U \leqslant \alpha(\mathbf{X}_t, \mathbf{Y})\,, \\ \mathbf{X}_t & \textit{if } U > \alpha(\mathbf{X}_t, \mathbf{Y})\,. \end{cases}$$

5. *Set* $t = t + 1$ *and repeat from Step 2 until a stopping criterion is met.*

The **cross-entropy** (CE) method can be used for both deterministic and noisy optimization. Consider first the deterministic minimization problem (34), where S is some real-valued performance function on a set $\mathscr{X}$. The basic idea of the CE method for optimization is to define a parametric family of probability densities $\{f(\cdot;\mathbf{v}), \mathbf{v} \in \mathscr{V}\}$ on the state space $\mathscr{X}$, and to iteratively update the parameter $\mathbf{v}$ so that $f(\cdot;\mathbf{v})$ places more mass closer to solutions than on the previous iteration.

In practice, this results in an algorithm with two basic phases:

- *Sampling*: Samples $\mathbf{X}_1, \ldots, \mathbf{X}_N$ are drawn independently according to $f(\cdot;\mathbf{v})$. The objective function S is evaluated at these points.
- *Updating*: A new parameter $\widehat{\mathbf{v}}$ is selected on the basis of those $\mathbf{X}_i$ for which $S(\mathbf{X}_i) \leqslant \widehat{\gamma}$ for some level $\widehat{\gamma}$. These $\{\mathbf{X}_i\}$ form the **elite sample** set, $\mathscr{E}$.

At each iteration the level parameter $\widehat{\gamma}$ is chosen as the worst performance (for minimization: the largest) of the best performing N^{e} samples, and the parameter $\mathbf{v}$ is updated as

$$\widehat{\mathbf{v}} = \underset{\mathbf{v}\in\mathscr{V}}{\operatorname{argmax}} \sum_{\mathbf{X}\in\mathscr{E}} \ln f(\mathbf{X};\mathbf{v}) \ . \tag{37}$$

This updating formula is the result of minimizing the Kullback–Leibler or CE distance between the conditional density of $\mathbf{X} \sim f(\mathbf{x};\mathbf{v})$ given $S(\mathbf{X}) \leqslant \widehat{\gamma}$, and $f(\mathbf{x};\widehat{\mathbf{v}})$; see [21]. Note that (37) yields the maximum likelihood estimator of $\mathbf{v}$ based on the elite samples. Hence, for many specific families of distributions, including exponential families, explicit solutions can be found. An important example is where $\mathbf{X} \sim \mathsf{N}(\boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}^2))$, where the mean vector $\boldsymbol{\mu}$ and the vector of variances $\boldsymbol{\sigma}^2$ are updated via the sample mean and sample variance of the elite samples. This is known as **normal updating**. A generic CE algorithm for minimization is as follows.

Algorithm 13 (CE Algorithm for Minimization)

1. *Choose an initial parameter vector* $\widehat{\mathbf{v}}_0$. *Let* $N^{\mathrm{e}} = \lceil \varrho N \rceil$ *be the number of elite samples. Set* $t = 1$.
2. *Generate* $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\mathrm{iid}} f(\cdot;\widehat{\mathbf{v}}_{t-1})$. *Calculate the performances* $S(\mathbf{X}_i)$ *for all* i *and order them from smallest to largest:* $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. *Let* $\widehat{\gamma}_t$ *be the sample* ϱ*-quantile of performances; that is,* $\widehat{\gamma}_t = S_{(N^{\mathrm{e}})}$.
3. *Use the* same *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *and set*

$$\widehat{\mathbf{v}}_t = \underset{\mathbf{v}}{\operatorname{argmax}} \sum_{k=1}^{N} \mathrm{I}_{\{S(\mathbf{X}_k) \leqslant \widehat{\gamma}_t\}} \ln f(\mathbf{X}_k;\mathbf{v}) \ . \tag{38}$$

4. *If some stopping criterion is met, stop; otherwise, set* $t = t + 1$ *and return to Step 2.*

Algorithm 13 can easily be modified to minimize *noisy* functions $S(\mathbf{x}) = \mathbb{E}\widetilde{S}(\mathbf{x}, \boldsymbol{\xi})$, as defined in (34). The only change required in the algorithm is that every function value $S(\mathbf{x})$ be replaced by its estimate $\widehat{S}(\mathbf{x})$.

In practice various modifications of the basic algorithm are used; see, for example, [2, 10]. In particular, it is often useful to include **smoothing** of the parameter vectors: given a vector of *smoothing parameters* $\boldsymbol{\alpha}$, replace Step 3 of Algorithm 13 by

3'. Use the same *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *and set*

$$\widetilde{\mathbf{v}}_t = \underset{\mathbf{v}}{\operatorname{argmax}} \sum_{k=1}^{N} \mathrm{I}_{\{S(\mathbf{X}_k) \leqslant \widehat{\gamma}_t\}} \ln f(\mathbf{X}_k; \mathbf{v}) \,, \tag{39}$$

updating $\widehat{\mathbf{v}}_t$ *as*

$$\widehat{\mathbf{v}}_t = \operatorname{diag}(\boldsymbol{\alpha})\, \widetilde{\mathbf{v}}_t + \operatorname{diag}(\mathbf{1} - \boldsymbol{\alpha})\, \widehat{\mathbf{v}}_{t-1} \,.$$

Conclusion

Monte Carlo simulation provides one of the most useful generic approaches to statistical computing. During the last few years significant advances have taken place in the application and theory of Monte Carlo. Recent insights in adaptive estimation methods provide novel approaches to rare-event simulation and randomized optimization, while Markov chain Monte Carlo techniques have significantly increased the scope and applicability of Bayesian statistical techniques.

References

[1] S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Springer-Verlag, New York, 2007.

[2] 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.

[3] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[4] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[5] G. S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, New York, 1996.

[6] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[7] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York, 2004.

[8] M. H. Kalos and P. A. Whitlock. *Monte Carlo Methods, Volume I: Basics*. John Wiley & Sons, New York, 1986.

[9] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, New York, 1992. corrected third printing.

[10] D. P. Kroese, S. Porotsky, and R. Y. Rubinstein. The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability*, 8(3):383–407, 2006.

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

[12] H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, New York, second edition, 2003.

[13] P. L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159 – 164, 1999.

[14] 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.

[15] P. A. Lewis, A. S. Goodman, and J. M. Miller. A pseudo-random number generator for the system/360. *IBM Systems Journal*, 8(2):136–146, 1969.

[16] T. G. Lewis and W. H. Payne. Generalized feedback shift register pseudorandom number algorithm. *Journal of the ACM*, 20(3):456–468, 1973.

[17] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[18] N. Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, 15:125–130, 1987.

[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

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

[21] 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.

[22] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, New York, second edition, 2007.

[23] Y. Shen, S. Kiatsupaibul, Z. B. Zabinsky, and R. L. Smith. An analytically derived cooling schedule for simulated annealing. *Journal of Global Optimization*, 38(2):333–365, 2007.

Further Reading

In addition to the references in the text, further information on Monte Carlo methods may be found, for example, in [1, 4, 5, 8, 20].

Cross-References

Markov chain Monte Carlo, Simulation methods, Importance sampling

Supplementary Information

The homepage `www.montecarlohandbook.org` for the *Handbook of Monte Carlo Methods* contains an extensive library of MATLAB code for Monte Carlo simulation.

The homepage for the *Cross-Entropy Method*, featuring many articles on the CE method, is `www.cemethod.org`.