Monte Carlo *and* Simulation

nicolas.chopin@ensae.fr

Section 1

Introduction

Monte Carlo: principle

$$\mathbb{E}[\varphi(X)] \approx \frac{1}{N}\sum_{n=1}^{N}\varphi(X_n)$$

with $\varphi : \mathcal{X} \to \mathbb{R}$.

Rationale: MSE, Law of large number, central limit theorems.

Also: **confidence intervals**!

Need for *simulation* methods. Note that simulation has other uses beyond Monte Carlo.

Bibliography

- Stats, MCMC: *Monte Carlo statistical methods*, C.P Robert, Springer
- IID simulation: *non-uniform random variate generation*, L. Devroye, Springer
- in French: *La simulation de Monte Carlo*, B. Tuffin, Lavoisier
- QMC: *Monte Carlo and quasi-Monte Carlo sampling*, C. Lemieux, Springer
- finance: *Monte Carlo methods in financial engineering*, P. Glasserman, Springer

Section 2

Pseudo-random number generators

Famous quotes, outline

Anyone who uses software to produce random numbers is in a "state of sin''. John von Neumann

One should not use a random method to generate random numbers. Donald Knuth

A PRNG is a *convenient fiction*. Ideally, it should:

- be *fast*,
- be *reproducible*,
- look *random* (at least according to statistical tests, e.g. "die-hard").

PRNGs: a few facts

- The general structure of a PRNG: $x_t = f(x_{t-1})$, where $x_t \in \{0, \ldots, 2^k - 1\}$; by construction, x_t is **periodic**.
- LCG (linear congruential generators):

$$x_{t+1} = (ax_t + c) \pmod{m}$$

  and take $u_t = x_t/m$ so that the u_t's are in $[0, 1]$.
- Take $c = 0$ for simplicity (then seed 0 is forbidden; and 0 is never generated, provided m is prime, and $a < m$).
- Assuming m is prime, the period is $m - 1$ iff $a^k - 1$ is a multiple of m for $k = m - 1$, but not $k \leq m - 2$.

Lattice structure

- Vectors of dim d lie on at most $(d!m)^{1/d}$ hyperplanes in the d-dimensional unit cube; e.g. for $m = 2^{31} - 1$, 108 for $d = 3$ and 39 for $d = 10$.
- RANDU, the most ill-conceived random number generators ever designed... has $a = 65539 = 2^{16} + 3$, $c = 2^{31}$, and is such that $x_t = 6x_{t-1} - 9x_{t-2}$.
- See Table 2.1 p 44 of Glasserman for better choices of (a,c).
- note that if a is not small, then computing $a * x$ is not easy even using floating point operations. We could take $a = 2^k$, but then generators typically have bad properties (see RANDU).

More modern PRNGs

- basic LCGs (even with *good* values of a and c) are now considered obsolete.
- Combine several generators to (a) increase period; and (b) reduce lattice structure: e.g. take the sum of K generators modulo one (Wichmann-Hill).
- Mersenne twister: very popular 32-bit PRNG (Python, R, Matlab, etc), has period $2^{19937} - 1$.
- Also push for 64-bit PRNG.

Main conclusion

- **DO NOT** use C standard implementation `rand()`.
- **DO NOT** implement your own PRNG.
- **DO** resort to some **modern** implementation of a **modern** generator, such as Mersenne twister; see e.g. GSL in C.

Section 3

Non-uniform simulation

Outline

A few general recipes:

- inversion
- rejection
- chain rule

plus several specialised ones (e.g. Box-Muller).

inversion

inversion algorithm

If X has CDF F, take

$$X = F^{-1}(U), \quad U \sim \mathcal{U}[0,1].$$

Applications: exponential, Laplace, Gaussian?

Box-Muller

Box-Muller

$$\begin{cases} X & = \sqrt{-2\log(U)} * \cos(2\pi V) \\ Y & = \sqrt{-2\log(U)} * \sin(2\pi V) \end{cases}$$

Then $X,\ Y \sim N(0,1)$, independently.

A sneaky introduction to rejection

To understand the coming slides, note that the following algorithm

Rejection

```
Repeat $X \sim \mathcal{U}(\mathcal{A})$
Until $X \in \mathcal{B}$.
```

draws from $\mathcal{U}(\mathcal{B})$ (provided $\mathcal{B} \subset \mathcal{A}$).

Modified Box-Muller

Box-Muller with rejection

```
Repeat
    $U,\ V \sim \mathcal{U}[-1,1]$
Until $S := U^2 + V^2 \leq 1$.
Return
```

$$\begin{cases} X & = U\sqrt{-2\log(S)/S} \\ Y & = V\sqrt{-2\log(S)/S} \end{cases}$$

Then $X,\ Y \sim N(0,1)$, independently.

Note: avoid computing sin and cos.

Rejection

Let f, g PDFs such that $f \leq Mg$ (with $M \geq 1$).

Accept-reject

```
Repeat
                    $X \sim g, \quad U \sim \mathcal{U}[0,1]$,
Until $U \leq f(X)/Mg(X)$.
```

Properties: $X \sim f$, number of draws until acceptance is Geometric$(1/M)$.

Justification: uniform sampling under the graph, see next slide.

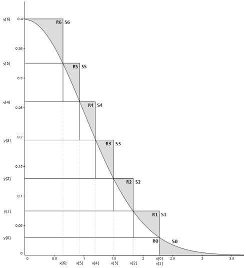
Uniform sampling under the graph

For a function f, let $\mathcal{G} = \{(x, y) \in \mathbb{R}^2 :\, 0 \le y \le Mf(x)\}$, then

$$(X, Y) \sim \mathcal{U}(\mathcal{G}) \Leftrightarrow \begin{cases} X & \sim f \\ Y|X = x & \sim \mathcal{U}[0, Mf(x)] \end{cases}$$

Note: this construction is in fact not restricted to real-valued random variables.

Ziggurat algorithm for $N(0,1)$ (Marsaglia, 60?)



K Slices $S_k = [-x_k, x_k] \times [y_k, y_{k+1}]$ constructed to have the same area.

1. Choose slice k (uniformly).
2. Sample (X, Y) within slice k.
3. If $X \leq x_{k+1}$, return X, else, if $Y \leq \varphi(X)$, return X, else go to 1.

Note: If slice 0 is selected, extra steps required (truncated Gaussian distribution).

Multivariate simulation: chain-rule decomposition

The inverse transform method is restricted to real-valued random variables, the inverse transform *is not*.

General recipe to generate jointly (X, Y, Z), with PDF $f(x, y, z)$:

1. `Generate` $X \sim f_X(x)$ (marginal). Call x the output.
2. `Generate` $Y|X = x \sim f_{Y|X}(y|x)$ (conditional given $X = x$). Call y the output.
3. `Generate` $Z|X = x, Y = y \sim f_{Z|Y,X}(z|x, y)$ (full conditional). Call z the output.

Gaussian vectors

The standard method to generate $X \sim N_d(\mu, \Sigma)$ is:

- `Generate` $Z_1, \ldots, Z_d \sim N(0,1)$.
- `Compute` $C = \mathrm{Choleksy}(\Sigma)$. (i.e. $\Sigma = CC^T$, and C is lower triangular)
- `Return` $X = \mu + CZ$.

The Cholesky decomposition costs $\mathcal{O}(d^3)$.

Section 4

Non-uniform simulation in spaces other than $\mathbb{R}^d$

Outline

Some recipes to sample specific cases of

- distributions over constrained sets
- discrete distributions

How to sample N sorted uniforms

Naive method: `sample` $U_n \sim \mathcal{U}[0,1]$ for $n = 1, \ldots, N$, `return` sort$(U_{1:N})$.
Cost is $\mathcal{O}(N \log N)$ (not bad).

How to sample N sorted uniforms

Naive method: `sample` $U_n \sim \mathcal{U}[0,1]$ for $n = 1, \ldots, N$, `return` sort$(U_{1:N})$. Cost is $\mathcal{O}(N \log N)$ (not bad).

Smart $\mathcal{O}(N)$ method:

- `Sample` $E_1, \ldots, E_{N+1} \sim \mathrm{Exp}(1)$.
- `Compute` $V_{1:(N+1)} = \mathrm{cumsum}(E_{1:(N+1)})$.
- `Return` $(V_1/V_{N+1}, \ldots, V_N/V_{N+1})$.

How to sample uniformly on the sphere

- `Sample` $X \sim N_d(0, I_d)$.
- `Return` $X/\|X\|$.

How to sample from a discrete distribution over $\mathbb{N}$

The inverse methods extends to the discrete case. Simply define:

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}$$

In practice:

- `Sample` $U \sim \mathcal{U}[0,1]$
- `If` $U \leq p_0$, `return` 0
- `If` $p_0 < U \leq p_0 + p_1$, `return` 1
- etc

What if N and K are large

Suppose we want to sample N times from a distribution over $\{0, \ldots, K-1\}$. If we run the algorithm of the previous slide N times, we do $\mathcal{O}(NK)$ operations (on average). Can we do better?

What if N and K are large

Suppose we want to sample N times from a distribution over $\{0, \ldots, K-1\}$. If we run the algorithm of the previous slide N times, we do $\mathcal{O}(NK)$ operations (on average). Can we do better?

Solution: use as input N sorted uniforms. Then cost is $\mathcal{O}(N+K)$.

Application: (weighted) bootstrap.

Inverse CDF algorithm

```
def inversecdf(su,W):
    """ Input:  su[0:N] sorted uniforms
                W[0:K]  normalised weights (sum to one)
        Output: A[0:N]  indexes (in {0,...,K-1})
    """
    j=0; s=W[0]; N = su.shape[0]
    A = empty(N,'int')
    for n in xrange(N):
        while su[n]>s:
            j += 1
            s += W[j]
        A[n] = j
    return A
```

How to sample a permutation

"Naive" $\mathcal{O}(N \log N)$ method: $\sigma = \mathrm{argsort}(U_{1:N})$.

How to sample a permutation

"Naive" $\mathcal{O}(N \log N)$ method: $\sigma = \mathrm{argsort}(U_{1:N})$.

Smart $\mathcal{O}(N)$ method:

- Let $\sigma = (1, 2, \ldots, N)$.
- $I \sim \mathcal{U}(1, \ldots, N)$, swap $\sigma(1)$ and $\sigma(I)$.
- $I \sim \mathcal{U}(2, \ldots, N)$, swap $\sigma(2)$ and $\sigma(I)$.
- etc.

Section 5

Variance reduction

Objectives, outline

Given a certain quantity

$$I = \mathbb{E}[\varphi(X)] = \int_{\mathcal{X}} f(x)\varphi(x)\,dx$$

find a Monte Carlo estimator with smaller variance than the standard estimator

$$\frac{1}{N}\sum_{n=1}^{N}\varphi(X_n).$$

Objectives, outline

Given a certain quantity

$$I = \mathbb{E}[\varphi(X)] = \int_{\mathcal{X}} f(x)\varphi(x)\,dx$$

find a Monte Carlo estimator with smaller variance than the standard estimator

$$\frac{1}{N}\sum_{n=1}^{N}\varphi(X_n).$$

Recipes:

- antithetic variables
- control variates

Antithetic variables

In cases where $\varphi(-X)$ has the same distribution as $\varphi(X)$, use:

$$\hat{I}_{\mathrm{anti}} = \frac{1}{2N} \sum_{n=1}^{N} \{\varphi(X_n) + \varphi(-X_n)\}$$

Antithetic variables

In cases where $\varphi(-X)$ has the same distribution as $\varphi(X)$, use:

$$\hat{I}_{\mathrm{anti}} = \frac{1}{2N}\sum_{n=1}^{N}\{\varphi(X_n)+\varphi(-X_n)\}$$

Lemma:

$$\mathrm{Var}(\hat{I}_{\mathrm{anti}}) \leq \mathrm{Var}(\hat{I})$$

Antithetic variables

In cases where $\varphi(-X)$ has the same distribution as $\varphi(X)$, use:

$$\hat{I}_{\mathrm{anti}} = \frac{1}{2N}\sum_{n=1}^{N}\{\varphi(X_n)+\varphi(-X_n)\}$$

Lemma:

$$\mathrm{Var}(\hat{I}_{\mathrm{anti}}) \leq \mathrm{Var}(\hat{I})$$

Note: we have less variance, but twice as many evaluations of $\varphi\ldots$

Control variates (univariate case)

Let Z a real-valued r.v. such that $\mathbb{E}(Z) = 0$. For any β,

$$\hat{I}_{\mathrm{cv}} = \frac{1}{N}\sum_{n=1}^{N}\{\varphi(X_n) - \beta Z_n\}$$

is an unbaised estimator of $I = \mathbb{E}[\varphi(X)]$.

Control variates (univariate case)

Let Z a real-valued r.v. such that $\mathbb{E}(Z) = 0$. For any β,

$$\hat{I}_{\mathrm{cv}} = \frac{1}{N}\sum_{n=1}^{N}\{\varphi(X_n) - \beta Z_n\}$$

is an unbaised estimator of $I = \mathbb{E}[\varphi(X)]$.

The smallest variance is obtained by taking

$$\beta_{\mathrm{opt}} = \frac{\mathrm{Cov}(\varphi(X), Z)}{\mathrm{Var}(Z)}.$$

Control variates (multivariate case)

$Z^1, \ldots, Z^K$ are mean-zero real-valued r.v. Take

$$\hat{I}_{\mathrm{cv}} = \frac{1}{N}\sum_{n=1}^{N}\{\varphi(X_n) - \sum_{k=1}^{K}\beta_k Z_n^k\}.$$

Control variates (multivariate case)

Z^1, ...,Z^K are mean-zero real-valued r.v. Take

$$\hat{I}_{\mathrm{cv}} = \frac{1}{N}\sum_{n=1}^{N}\{\varphi(X_n) - \sum_{k=1}^{K}\beta_k Z_n^k\}.$$

In practice, replace β_k by $\hat{\beta}_k$, the OLS estimate for regression:

$$\varphi(X_n) = \alpha + \sum_{k=1}^{K}\beta_k Z_n^k + \varepsilon_n.$$

Variance reduction and Rao-Blackwellisation

Often variance reduction techniques may be cast as particular **Rao-Blackwellisation** schemes, i.e. the idea that

$$\mathrm{Var}\left[\mathbb{E}[\varphi(X)|Z]\right] \leq \mathrm{Var}[\varphi(X)].$$

Section 6

Importance sampling

A simple identity

$$\begin{aligned}\mathbb{E}[\varphi(X)] &= \int_{\mathcal{X}} \varphi(x) f(x)\,dx \\ &= \int_{\mathcal{X}} \varphi(x) \frac{f(x)}{g(x)} g(x)\,dx = \mathbb{E}_g\left[\frac{f(X)}{g(X)}\varphi(X)\right]\end{aligned}$$

assuming $\mathrm{Supp}(f) \subset \mathrm{Supp}(g)$.

Any expectation w.r.t. PDF f may be rewritten thusly as an expectation w.r.t. PDF g (which may be easier to simulate from):

$$\hat{I}_{IS} = \frac{1}{N}\sum_{n=1}^{N} \frac{f(X_n)}{g(X_n)}\varphi(X_n).$$

How to choose proposal g?

1. Check that variance exists, $\Leftrightarrow \mathbb{E}_g\left[\varphi(X)^2\frac{f(X)^2}{g(X)2}\right] < \infty$. (Sufficient condition: $f/g \leq M$, and $\mathbb{E}_f[\varphi^2] < \infty$.)

How to choose proposal g?

1. Check that variance exists, $\Leftrightarrow \mathbb{E}_g\left[\varphi(X)^2 \frac{f(X)^2}{g(X)2}\right] < \infty$. (Sufficient condition: $f/g \leq M$, and $\mathbb{E}_f[\varphi^2] < \infty$.)
2. Optimal g (in terms of minimizing variance) is

$$g_{\mathrm{opt}}(x) \propto f(x)|\varphi(x)|.$$

It is often not possible to simulate from g_{opt}, so more generally, it is recommended to take $g \approx f$.

Auto-normalised IS

Sometimes either f or g are known only up to a constant: $f = f_u/Z_f$, $g = g_u/Z_g$, and Z_f, Z_g are intractable. In that case, we use the auto-normalised IS estimator:

$$\hat{I}_{\mathrm{AIS}} = \frac{\sum_{n=1}^{N} w_n \varphi(X_n)}{\sum_{n=1}^{N} w_n}, \quad w_n = \frac{f_u(X_n)}{g_u(X_n)}.$$

This estimator is biased, and asymptotically Gaussian:

$$\sqrt{N}\left(\hat{I}_{\mathrm{AIS}} - I\right) \Rightarrow N(0, v_{f/g})$$

with $v_{f/g} = \mathbb{E}_g[\left(\frac{f}{g}\right)^2 (\varphi - I)^2]$ (assuming this quantity is $< \infty$).

How to choose g (bis repetita)

Same points as for standard IS:

1. Check that at least $v_{f/g} < \infty$; sufficient condition is (a) $f/g < \ M$ and (b) $\mathrm{Var}_f(\varphi) < \infty$.
2. Optimal g is
$$g_{\mathrm{opt}}(x) \propto f(x)|\varphi(x) - I|$$
which depends on I... In practice, take $g \approx f$.

Estimating the Z's, effective sample size

Note that AIS also provides an estimate of Z_f/Z_g:

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N} w_n\right] = \frac{Z_f}{Z_g}$$

and of $v_{f/g}$:

$$\frac{N\sum_{n=1}^{N} w_n^2 \left\{\varphi(X_n) - \hat{I}\right\}^2}{\left(\sum_{n=1}^{N} w_n\right)^2}.$$

Estimating the Z's, effective sample size

Note that AIS also provides an estimate of Z_f/Z_g:

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N} w_n\right] = \frac{Z_f}{Z_g}$$

and of $v_{f/g}$:

$$\frac{N\sum_{n=1}^{N} w_n^2 \left\{\varphi(X_n) - \hat{I}\right\}^2}{\left(\sum_{n=1}^{N} w_n\right)^2}.$$

Similarly, the **effective sample size**

$$\frac{\left(\sum_{n=1}^{N} w_n\right)^2}{\sum_{n=1}^{N} \left(w_n\right)^2} \in [1, N]$$

is a good indicator of AIS efficiency.

Curse of dimensionality

For $\mathcal{X} = \mathbb{R}^d$, $f(x) = \prod_{i=1}^d f_1(x_i)$, $g(x) = \prod_{i=1}^d g_1(x_i)$, one has:

$$\mathbb{E}_g[f^2/g^2] = C^d, \quad C \geq 1.$$

We expect the variance of IS to grow exponentially with the dimension.

Resampling

How to transform weighted sample (w_n, X_n) into an **unweighted** sample?

Resampling

How to transform weighted sample (w_n, X_n) into an **unweighted** sample?

Simply draw randomly $\tilde{X}_n$ from

$$\sum_{n=1}^{N} W_n \delta_{X_n}, \quad W_n = \frac{w_n}{\sum_{m=1}^{N} w_m}$$

(as in the bootstrap).

Resampling

How to transform weighted sample (w_n, X_n) into an **unweighted** sample?

Simply draw randomly $\tilde{X}_n$ from

$$\sum_{n=1}^{N} W_n \delta_{X_n}, \quad W_n = \frac{w_n}{\sum_{m=1}^{N} w_m}$$

(as in the bootstrap).

See previous chapter on multinomial sampling.

Section 7

Quasi-Monte Carlo

Principle

Often one may rewrite quantity of interest as:

$$I = \mathbb{E}[\varphi(U)], \quad U \sim \mathcal{U}[0,1]^d$$

and then use

$$\hat{I} = \frac{1}{N} \sum_{n=1}^{N} \varphi(U_n).$$

Principle

Often one may rewrite quantity of interest as:

$$I = \mathbb{E}[\varphi(U)], \quad U \sim \mathcal{U}[0,1]^d$$

and then use

$$\hat{I} = \frac{1}{N} \sum_{n=1}^{N} \varphi(U_n).$$

Can we construct (deterministic or random) points $U_1, \ldots, U_N$ in $[0,1]^d$ so that the approximation error is smaller than with standard Monte Carlo (i.e. U_n are IID uniforms)?

Stratification ($d = 1$)

- Generate N/K uniforms in each interval $[(k-1)/K, k/K]$, $k = 1, \ldots, K$. (Note the connection with antithetic variables.)

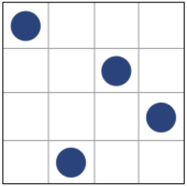
Stratification ($d = 1$)

- Generate N/K uniforms in each interval $[(k-1)/K, k/K]$, $k = 1, \ldots, K$. (Note the connection with antithetic variables.)
- Or even take $K = N$, i.e. generate $U_n \sim \mathcal{U}[(n-1)/N, n/N]$.

Stratification ($d = 1$)

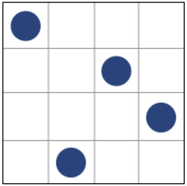
- Generate N/K uniforms in each interval $[(k-1)/K, k/K]$, $k = 1, \dots, K$. (Note the connection with antithetic variables.)
- Or even take $K = N$, i.e. generate $U_n \sim \mathcal{U}[(n-1)/N, n/N]$.
- or even take $u_n = (2n-1)/2N$, the (deterministic) centre of interval $[(n-1)/N, n/N]$).

Stratification for $d > 1$: Latin hypercube sampling



Generate the U_n's so that exactly one point falls in each horizontal or vertical strip (of area $1/N$).

Stratification for $d > 1$: Latin hypercube sampling



Generate the U_n's so that exactly one point falls in each horizontal or vertical strip (of area $1/N$).

Tip: use random permutations.

Koksma–Hlawka inequality

$$|\hat{I} - I| \leq V(\varphi) D^\star(u_{1:N})$$

where $V(\varphi)$ is the variation of φ (in the sense of Hardy and Krause), and

$$D^\star(u_{1:N}) = \sup_{[0,b]\subset[0,1]^d} \left| N^{-1} \sum_{n=1}^{N} \mathbb{I}_{[0,b]}(u_n) - \prod_{i=1}^{d} b_i \right|$$

is the **star discrepancy**.

Proof for $d = 1$

$$N^{-1}\sum_{n=1}^{N}\varphi(u_n) - \int_0^1 \varphi(u)\,du = \int_0^1 \delta(u)\varphi'(u)\,du$$

where $\delta(u) = u - N^{-1}\sum_{n=1}^{N}\mathbf{1}(u_n \leq u)$.

Why is the previous inequality so important?

Because we know how to construct:

- point-sets such that $D^\star(u_{1:N}) = \mathcal{O}\left(\frac{(\log N)^{d-1}}{N}\right)$
- sequences such that $D^\star(u_{1:N}) = \mathcal{O}\left(\frac{(\log N)^{d}}{N}\right)$

hence we can do **better** than Monte Carlo, i.e. $\mathcal{O}_P(\frac{1}{\sqrt{N}})$.

Why is the previous inequality so important?

Because we know how to construct:

- point-sets such that $D^{\star}(u_{1:N}) = \mathcal{O}\left(\frac{(\log N)^{d-1}}{N}\right)$
- sequences such that $D^{\star}(u_{1:N}) = \mathcal{O}\left(\frac{(\log N)^{d}}{N}\right)$

hence we can do **better** than Monte Carlo, i.e. $\mathcal{O}_P(\frac{1}{\sqrt{N}})$.

Side note: there are good reasons to believe that these rates are optimal.

$d = 1$

Take $u_n = (2n-1)/2N$, $n = 1, \ldots, N$. Then

$$D^{\star}(u_{1:N}) = \frac{1}{2N}.$$

Van der Corput (sequence for $d = 1$)

In base b, for $n = \sum_{j=0}^{k} a_j(n) b^j$, take

$$u_n = \sum_{j=0}^{k} a_j(n) b^{-1-j}.$$

Van der Corput (sequence for $d = 1$)

In base b, for $n = \sum_{j=0}^{k} a_j(n) b^j$, take

$$u_n = \sum_{j=0}^{k} a_j(n) b^{-1-j}.$$

e.g. for $b = 2$: $1/2$, $1/4$, $3/4$, $1/8$, $\dots$

For $b = 3$: $1/3$, $2/3$, $1/9$, $\dots$

Van der Corput (sequence for $d = 1$)

In base b, for $n = \sum_{j=0}^{k} a_j(n) b^j$, take

$$u_n = \sum_{j=0}^{k} a_j(n) b^{-1-j}.$$

e.g. for $b = 2$: $1/2$, $1/4$, $3/4$, $1/8$, $\ldots$

For $b = 3$: $1/3$, $2/3$, $1/9$, $\ldots$

Then $D^\star(u_{1:N}) = \mathcal{O}(\log N/N)$.

$d > 1$: Halton & Hammersley

Halton sequence: component j is a van der Corput sequence in base b_j, where the b_j are the first d prime numbers. Discrepancy is $\mathcal{O}((\log N)^d/N)$.

$d > 1$: Halton & Hammersley

Halton sequence: component j is a van der Corput sequence in base b_j, where the b_j are the first d prime numbers. Discrepancy is $\mathcal{O}((\log N)^d/N)$.

Hammersley point set (of size N): take N first elements of Halton sequence of dimension d, replace last component by n/N.
Discrepancy is $\mathcal{O}((\log N)^{d-1}/N)$.

$d > 1$: Halton & Hammersley

Halton sequence: component j is a van der Corput sequence in base b_j, where the b_j are the first d prime numbers. Discrepancy is $\mathcal{O}((\log N)^d/N)$.

Hammersley point set (of size N): take N first elements of Halton sequence of dimension d, replace last component by n/N.
Discrepancy is $\mathcal{O}((\log N)^{d-1}/N)$.

Note however that for large d, both Halton and Hammersley require many points to cover the space...

Other low-discrepancy sequences and point sets

- Niederreiter
- Faure
- Sobol'
- . . .

RQMC (randomised QMC)

QMC is purely deterministic. It lacks a simple way to evaluate the numerical error. Imagine we are able to randomise $U_{1:N}$ so that

1. $U_n \sim \mathcal{U}[0,1]^d$ (marginally).
2. $U_{1:N}$ is still a low-discrepancy point-set (or sequence).

Then

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N}\varphi(U_n)\right] = \mathbb{E}[\varphi(U)]$$

and we can evaluate the numerical error through the empirical variance (over repeated runs).

RQMC: random shift

The simplest RQMC strategy is to generate a low-discrepancy point set $v_{1:N}$, $W \sim \mathcal{U}[0,1]^d$, then take:

$$U_n = v_n + W \pmod 1 \qquad \text{(componentwise)}$$

RQMC: a surprising result

Owen (1998) showed that for smooth functions φ

$$\mathrm{Var}[\hat{I}] = \mathcal{O}\left(\frac{(\log N)^{(d-1)/2}}{N^3}\right)$$

when scrambling (a particular RQMC technique) is used.

conclusion: QMC vs MC

- QMC has a better convergence rate.
- But for large d, QMC might need a very large N to beat MC.
- With MC, the (square) error is simple to estimate, whereas for QMC, we have only a deterministic bound, which is hard to evaluate, and is often pessimistic. See RQMC however.
- Variance reduction: may be used in conjunction with (R)QMC. (Recommendation is to do variance reduction, *then* replace MC with QMC).
- Practical recommendation: scrambled Sobol' seems like a good default choice (or Latin Hypercube sampling for very high dimensions).

Section 8

Markov chain Monte Carlo

Outline

In some settings, simulating *independently* $X \sim \pi(dx)$ is difficult, but it is possible to simulate a Markov chain (X_n) that leaves $\pi(dx)$ **invariant**. Then, we still have

$$\frac{1}{N}\sum_{n=1}^{N}\varphi(X_n) \approx \mathbb{E}_\pi[\varphi(X)]$$

in some sense.

Outline

In some settings, simulating *independently* $X \sim \pi(dx)$ is difficult, but it is possible to simulate a Markov chain (X_n) that leaves $\pi(dx)$ **invariant**. Then, we still have

$$\frac{1}{N}\sum_{n=1}^{N}\varphi(X_n) \approx \mathbb{E}_\pi[\varphi(X)]$$

in some sense.

This is the case in particular when density π is known only up to a constant.

Definitions

- A **Markov kernel** $K(x, dy)$ is an application $\mathcal{X} \to \mathcal{P}(\mathcal{X})$.

Definitions

- A **Markov kernel** $K(x, dy)$ is an application $\mathcal{X} \to \mathcal{P}(\mathcal{X})$.
- A Markov kernel K leaves distribution π **invariant** iff

$$\int_{\mathcal{X}} \pi(dx) K(x, dy) = \pi(dy).$$

Definitions

- A **Markov kernel** $K(x, dy)$ is an application $\mathcal{X} \to \mathcal{P}(\mathcal{X})$.
- A Markov kernel K leaves distribution π **invariant** iff

$$\int_{\mathcal{X}} \pi(dx)K(x, dy) = \pi(dy).$$

- A Markov kernel is **reversible** w.r.t. π iff

$$\pi(dx)K(x, dy) = \pi(dy)K(y, dx).$$

This implies that π is invariant.

Metropolis-Hastings

Let $Q(x, dy)$ a Markov kernel, such that $Q(x, dy) = q(x, y)dy$.

Hastings-Metropolis step

Input: X_{n-1}

1. Generate $Y \sim Q(X_{n-1}, dy)$
2. With probability $1 \wedge r(X_{n-1}, Y)$, where
$$r(x, y) = \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}$$
accept Y, i.e. $X_n = Y$; otherwise $X_n = X_{n-1}$.

Property: This kernel is reversible (w.r.t. π).

An important practical point

Note that Hastings-Metropolis may be implemented even if π is known only up to a constant: $\pi(x) = \pi_u(x)/Z$, Z is intractable. Then

$$r(x, y) = \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)} = \frac{\pi_u(y)q(y, x)}{\pi_u(x)q(x, y)}$$

Examples of MH algorithms

1. $q(x,y) = q(y,x)$, for instance $q(x,y) = N(y; x, \Sigma)$ (Gaussian **random walk**); then

$$r = \frac{\pi(y)}{\pi(x)}$$

2. $q(x,y) = q(y)$: **independent Metropolis**; then

$$r = \frac{\pi(y)q(x)}{\pi(x)q(y)}$$

3. Langevin proposal:

$$Y \sim N(x + \frac{1}{2}\Sigma\nabla \log \pi(x), \Sigma)$$

(two-block) Gibbs sampling

Assume $X = (X_1, X_2)$, $\pi(x) = \pi(x_1, x_2)$, with conditional distributions $\pi_{1|2}(x_1|x_2)$, $\pi_{2|1}(x_2|x_1)$.

Gibbs sampling step

Input: $X_{n-1} = (X_{n-1,1}, X_{n-1,2})$

1. `Generate` $X_{n,1} \sim \pi_{1|2}(\bullet|X_{n-1,2})$.
2. `Generate` $X_{n,2} \sim \pi_{2|1}(\bullet|X_{n,1})$.

Again, this leaves invariant π. Gibbs can be generalised to $k > 2$ blocks.

Combining chains

1. If K_1, K_2 leave π invariant, then so does $K_1 K_2$.
2. **Within** Gibbs, we can replace the exact simulation of $X_1|X_2$ (say) by a Metropolis step that leaves invariant $\pi_{1|2}$.

MCMC in practice

1. Assess how long it takes for the chain to reach stationarity;
2. When chain seems stationary, check for intra-correlations, i.e. look at ACF (Auto-Correlation Function).

Then we compute averages

$$\frac{1}{N - N_0} \sum_{n=N_0+1}^{N} \varphi(X_n)$$

where N_0 is burn-in time, and $N - N_0$ is sufficiently large relative to the **auto-correlation time** (i.e. time k so that X_n and X_{n+k} are nearly uncorelated).

scaling random walk Metropolis

One big hurdle with random walk Metropolis is the choice of Σ, in the proposal $N(x, \Sigma)$. If too small, chain moves slowly, if too large, proposals always get rejected.

scaling random walk Metropolis

One big hurdle with random walk Metropolis is the choice of Σ, in the proposal $N(x, \Sigma)$. If too small, chain moves slowly, if too large, proposals always get rejected.

Theory (e.g. Roberts and Rosenthal, 2004) indicates that one should take

$$\Sigma = c\Sigma_\pi$$

where Σ_π is the covariance matrix of target π, and c calibrated so that acceptance rate is ≈ 0.25.

a tiny bit of MCMC theory

1. From an arbitrary starting point $X_0 = x_0$, and any $\varepsilon > 0$, we have

   $$\|K^n(x_0, dx_n) - \pi(dx_n)\|_{\mathrm{TV}} \leq \varepsilon$$

   for n large enough.

2. CLT:

   $$\sqrt{N}\left(\frac{1}{N}\sum_{n=1}^{N}\varphi(X_n) - I\right) \Rightarrow N(0, V(\varphi))$$

   with

   $$V(\varphi) = \mathrm{Var}_\pi(\varphi) + 2\sum_{k=1}^{\infty}\gamma_k(\varphi)$$

   and $\gamma_k(\varphi) = \mathrm{Cov}[\varphi(X_n), \varphi(X_{n+k})]$.

Adaptive MCMC?

Can we use past samples to automatically calibrate Metropolis-Hastings? e.g. at time t, do a random walk Metropolis step, of size $\Sigma = c\hat{\Sigma}_t$, where $\hat{\Sigma}_t$ is the empirical covariance matrix computed from $X_0, \ldots, X_{t-1}$.

Adaptive MCMC?

Can we use past samples to automatically calibrate Metropolis-Hastings? e.g. at time t, do a random walk Metropolis step, of size $\Sigma = c\hat{\Sigma}_t$, where $\hat{\Sigma}_t$ is the empirical covariance matrix computed from $X_0, \ldots, X_{t-1}$.

Big theoretical problem: we are not simulating a Markov chain any more (X_t depends on the whole past). Convergence is more difficult to establish.

Section 9

Interlude: Bayesian classification

Outline

Consider model with responses $y_i \in \{-1, 1\}$, covariates $x_i \in \mathbb{R}^p$, likelihood

$$L(x, y; \beta) = \prod_{i=1}^{n_d} F(y_i \beta^T x_i)$$

with $F = \Phi$ (probit), or $F = L$ (logit), and prior

$$\pi(\beta) = 1$$

Outline

Consider model with responses $y_i \in \{-1, 1\}$, covariates $x_i \in \mathbb{R}^p$, likelihood

$$L(x, y; \beta) = \prod_{i=1}^{n_d} F(y_i \beta^T x_i)$$

with $F = \Phi$ (probit), or $F = L$ (logit), and prior

$$\pi(\beta) = 1$$

The posterior is

$$\pi(\beta | x, y) \propto \prod_{i=1}^{n_d} F(y_i \beta^T x_i)$$

We will use this example to discuss many of the approaches seen so far.

Laplace approximation

Taylor expansion of log posterior density around its mode, $\widehat{\beta} = \arg\max \pi(\beta|x,y)$ (MLE):

$$\log \pi(\beta|x,y) \approx \log \pi(\widehat{\beta}|x,y) - \frac{1}{2}(\beta - \widehat{\beta})^T H(\beta - \widehat{\beta})$$

implies a Gaussian approximation of the posterior: $\approx N(\widehat{\beta}, H^{-1})$.

Laplace approximation

Taylor expansion of log posterior density around its mode, $\widehat{\beta} = \arg\max \pi(\beta|x, y)$ (MLE):

$$\log \pi(\beta|x, y) \approx \log \pi(\widehat{\beta}|x, y) - \frac{1}{2}(\beta - \widehat{\beta})^T H(\beta - \widehat{\beta})$$

implies a Gaussian approximation of the posterior: $\approx N(\widehat{\beta}, H^{-1})$.

In practice, this Gaussian approximation may be:

1. obtained numerically (Newton);
2. used as a proposal in various approaches (e.g. importance sampling).

Section 10

Monte Carlo optimisation

Objectives

Numerical maximisation:

$$\max_{\theta \in \Theta} \psi(\theta)$$

when

1. ψ can be evaluated point-wise, but is difficult to maximise by standard methods: **Exploration**
2. ψ is an (intractable) expectation:

$$\psi(\theta) = \mathbb{E}_\theta[h(X,\theta)]$$

Stochastic approximation

Statistical applications: MLE

Exploration

When ψ can be evaluated point-wise, one may sample N times from some distribution $\pi(d\theta)$, and return $\max_{n=1,\ldots,N}\psi(\theta_n)$; for instance if Θ is compact, take $\pi(d\theta)$ to be the Uniform dist. over Θ.

Exploration

When ψ can be evaluated point-wise, one may sample N times from some distribution $\pi(d\theta)$, and return $\max_{n=1,\dots,N} \psi(\theta_n)$; for instance if Θ is compact, take $\pi(d\theta)$ to be the Uniform dist. over Θ.

In particular, consider

$$\pi_\lambda(\theta) \propto \exp\{\lambda\psi(\theta)\}.$$

When λ (inverse temperature) increases, support of π_λ gets more concentrated around modal regions, but in return it may be more difficult to sample from π_λ.

Simulated annealing

Simulate a (inhomogeneous) Markov chain as follows: at iteration t, do a Metropolis step w.r.t. π_{λ_t}, and make λ_t increase at a logarithmic rate.

The Cross-Entropy method

For some parametric family $\{f_\xi,\, \xi \in \Xi\}$, choose initial ξ_0, then iteratively:

1. Sample $\theta_1, \ldots, \theta_n \sim f_{\xi_t}$.
2. Estimate (using e.g. MLE) ξ_{t+1} from the 10% best of the θ_i (in terms of $\psi(\theta_i)$).

Other heuristic optimisation procedures

- genetic algorithms
- tabu search
- ant colony algorithm

and also more specialised ones.

Stochastic approximation

One has: $\psi(\theta) = \mathbb{E}_\theta[h(X, \theta)]$ (double dependence on θ). Possible approaches:

1. If Expectation is w.r.t. a fixed dist' f, $\psi(\theta) = \mathbb{E}[h(X, \theta)]$, generate $X_1, \ldots, X_n \sim f$, maximise $\theta \to N^{-1} \sum_{n=1}^{N} h(X_n, \theta)$.
2. Gradient-based approach, e.g.

$$\theta_{t+1} = \theta_t + \alpha_t \hat{\nabla} \psi(\theta_t)$$

   where $\hat{\nabla}\psi(\theta_t)$ is some MC estimate of the gradient of ψ.

Robins-Monroe

Take α_t such that $\alpha_t \to 0$, and $\sum_t \alpha_t = \infty$; e.g. $\alpha_t = Ct^{-b}$, $1/2 < b \leq 1$.

Robins-Monroe

Take α_t such that $\alpha_t \to 0$, and $\sum_t \alpha_t = \infty$; e.g. $\alpha_t = Ct^{-b}$, $1/2 < b \leq 1$.

To estimate the gradient, if $\psi(\theta) = \mathbb{E}_\theta[h(X)]$, one has

$$\nabla\psi(\theta) = \mathbb{E}_\theta[h(X)s_\theta(X)], \quad s_\theta(x) = \frac{\partial}{\partial\theta}\log f_\theta(x)$$

and thus a possible choice is:

$$\hat{\nabla}\psi(\theta) = \frac{1}{N}\sum_{n=1}^{N} h(X_n)s_\theta(X_n)$$

Section 11

Selected applications of Monte Carlo

Outline

1. Derivative pricing
2. Statistical applications: MCEM, Bayesian inference, ABC
3. Enumeration
4. Go playing. . .

Derivative pricing: statement

There, X is continuous-time process on $[0, T]$, and φ could be:

- $\varphi(X) = (K - X_T)^+$
- $\varphi(X) = (K - \frac{1}{k}\sum_i X_{t_i})^+$
- $\varphi(X) = (K - \int X_t dt)^+$
- $\varphi(X) = \mathbb{I}\{\tau_b > T\}(X_T - K)^+$, with $\tau_b = \inf\{t : X_t \leq b\}$
- etc.

Simulating Brownian paths

For a Brownian motion $\{W_t\}$, several ways to simulate *exactly* vector $(W_{t_1}, \ldots, W_{t_k})$:

- random walk: $W_{t_i} | W_{t_{i-1}} \sim N(W_{t_{i-1}}, t_i - t_{i-1})$
- Brownian bridge: $W_{t_i} | W_{t_{i-1}}, W_{t_{i+1}} \sim$

$$N\left(\frac{(t_{i+1} - t_i)W_{t_{i-1}} + (t_i - t_{i-1})W_{t_{i+1}}}{t_{i+1} - t_{i-1}}, \frac{(t_{i+1} - t_i)(t_i - t_{i-1})}{t_{i+1} - t_{i-1}}\right)$$

  and order times according to a van der Corput sequence: first t_k, then $t_{k/2}$, $t_{k/4}$, and so on.
- principal components

Simulating Brownian paths

For a Brownian motion $\{W_t\}$, several ways to simulate *exactly* vector $(W_{t_1}, \ldots, W_{t_k})$:

- random walk: $W_{t_i} | W_{t_{i-1}} \sim N(W_{t_{i-1}}, t_i - t_{i-1})$
- Brownian bridge: $W_{t_i} | W_{t_{i-1}}, W_{t_{i+1}} \sim$

  $$N\left(\frac{(t_{i+1} - t_i) W_{t_{i-1}} + (t_i - t_{i-1}) W_{t_{i+1}}}{t_{i+1} - t_{i-1}}, \frac{(t_{i+1} - t_i)(t_i - t_{i-1})}{t_{i+1} - t_{i-1}} \right)$$

  and order times according to a van der Corput sequence: first t_k, then $t_{k/2}$, $t_{k/4}$, and so on.
- principal components

Try to think about the implications for QMC. . .

QMC and Brownian paths

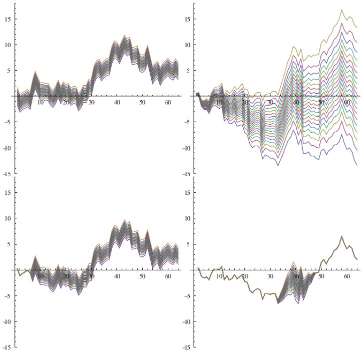


Fig. 8.6 Paths of Brownian motion obtained by the forward construction (*left*) and the Brownian bridge construction (*right*). All but one parameters are fixed

Top: all but first component fixed; bottom, all but seventh component fixed

Source: Chap. 8 of Leobacher and Pillichshammer (2014).

Euler discretization

In general, diffusion processes need to be discretized:

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW_t$$

becomes

$$X_{t+1} - X_t = \delta\mu(X_t) + \sigma(X_t)\epsilon_t, \quad \epsilon_t \sim N(0, \delta)$$

where δ is the discretization step.

Euler discretization

In general, diffusion processes need to be discretized:

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW_t$$

becomes

$$X_{t+1} - X_t = \delta\mu(X_t) + \sigma(X_t)\epsilon_t, \quad \epsilon_t \sim N(0, \delta)$$

where δ is the discretization step.

Choice of δ: trade-off between discretization bias and CPU time.

Multi-level Monte Carlo

Consider a sequence of decreasing steps: $\delta_0 > \ldots > \delta_L$; say $\delta_l = 2^{-l}$.

$$\mathbb{E}_{\delta_L}(\varphi) = \mathbb{E}_{\delta_0}(\varphi) + \sum_{l=1}^{L} \left\{ \mathbb{E}_{\delta_l}(\varphi) - \mathbb{E}_{\delta_{l-1}}(\varphi) \right\}$$

To get a low-variance estimate for each level, use **coupling**: e.g. use Brownian bridge construction to obtain the finer level from the coarser level.

Multi-level Monte Carlo

Consider a sequence of decreasing steps: $\delta_0 > \ldots > \delta_L$; say $\delta_l = 2^{-l}$.

$$\mathbb{E}_{\delta_L}(\varphi) = \mathbb{E}_{\delta_0}(\varphi) + \sum_{l=1}^{L} \left\{ \mathbb{E}_{\delta_l}(\varphi) - \mathbb{E}_{\delta_{l-1}}(\varphi) \right\}$$

To get a low-variance estimate for each level, use **coupling**: e.g. use Brownian bridge construction to obtain the finer level from the coarser level.

To minimise variance, choose N_l (number of samples for level l) to be:

$$N_l \propto \sqrt{V_l / C_l}$$

where V_l (resp. C_l) is variance (resp. CPU cost per sample) of estimate for level l.

Other worthy points

- control variates: simulation involves many Gaussian variables, with known mean and variance
- antithetic variables (Gaussians variables are symmetric)
- QMC very popular nowadays in option pricing

Statistical applications

- Bayesian estimation: already covered, see MCMC
- Frequentist estimation: MC for the E part of EM
- Likelihood-free inference

MCEM = Monte Carlo EM

For a statistical model involving a latent X and an observed y, maximise iteratively:

$$\theta_{t+1} = \arg\max_{\theta} \mathbb{E}[\log L(X, y; \theta)]$$

where the expectation is w.r.t. the distribution of X given $Y = y$ and $\theta = \theta_t$.

MCEM = Monte Carlo EM

For a statistical model involving a latent X and an observed y, maximise iteratively:

$$\theta_{t+1} = \arg\max_{\theta} \mathbb{E}[\log L(X, y; \theta)]$$

where the expectation is w.r.t. the distribution of X given $Y = y$ and $\theta = \theta_t$.

When the expectation is not tractable: use Monte Carlo. (To get convergence, use a larger and larger Monte Carlo sample)

MCEM = Monte Carlo EM

For a statistical model involving a latent X and an observed y, maximise iteratively:

$$\theta_{t+1} = \arg\max_{\theta} \mathbb{E}[\log L(X, y; \theta)]$$

where the expectation is w.r.t. the distribution of X given $Y = y$ and $\theta = \theta_t$.

When the expectation is not tractable: use Monte Carlo. (To get convergence, use a larger and larger Monte Carlo sample)

See also SAEM.

ABC (likelihood-free inference)

Data $y^\star$, model $p(y|\theta)$ such that (a) one can simulate from $p(y|\theta)$; (b) one cannot compute the likelihood $p(y|\theta)$. (Many scientific models fall in this category.)

ABC (likelihood-free inference)

Data $y^\star$, model $p(y|\theta)$ such that (a) one can simulate from $p(y|\theta)$; (b) one cannot compute the likelihood $p(y|\theta)$. (Many scientific models fall in this category.)

ABC (Approximate Bayesian inference) samples from:

$$p_\varepsilon(\theta, y|y^\star) \propto p(\theta)p(y|\theta)\mathbb{I}(\|s(y) - s(y^\star)\| \leq \varepsilon).$$