Methods of Monte Carlo Simulation

Ulm University
Institute of Stochastics

Lecture Notes
Dr. Tim Brereton
Winter Term 2013/2014

Ulm, February 2014

Contents

Chapter 1

Introduction

Monte Carlo methods are methods that use random numbers to solve problems or gain insight into problems. These problems can be 'probabilistic' in nature or 'deterministic'. Probabilistic applications of Monte Carlo methods include:

- Estimating probabilities and expectations.
- Estimating the sensitivity of random objects to changes in parameters.
- Getting a sense of what random objects 'look like' and how they behave.

Deterministic problems which can be solved using Monte Carlo methods are, for example:

- Estimating solutions to difficult integration problems.
- Approximating or finding solutions to complicated optimization problems.
- Solving mathematical problems by transforming them into 'probabilistic' problems (an example is probabilistic methods for solving partial differential equations).

Monte Carlo techniques are not always the best tools, especially for simple problems. However, they are the best (or only) solutions for a lot of realistic problems.

1.1 Monte Carlo Integration

A simple problem that can be solved using Monte Carlo methods is to compute an integral of the form

$$I = \int_0^1 f(x)\,\mathrm{d}x,$$

where f is an arbitrary function. This can be written as

$$I = \int_0^1 f(x)dx = \int_0^1 \frac{1}{1} f(x)\,\mathrm{d}x.$$

Note that 1/1 is the probability density function (pdf) of the uniform distribution on $(0,1)$. So, we can write

$$I = \int_0^1 f(x)\,\mathrm{d}x = \mathbb{E}\, f(X),$$

where $X \sim \mathcal{U}(0,1)$. Now we can approximate $\mathbb{E}\, f(X)$ by

$$I \approx \hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i),$$

where $X_1, X_2, \ldots, X_N$ are independent and identically distributed (iid) copies of X. Then, under suitable technical conditions (e.g. $\mathbb{E}\, f(x)^2 < \infty$), the strong law of large numbers implies that

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} f(X) \to \mathbb{E}\, f(X_i) \qquad \text{almost surely (a.s.) and in } L^2$$

We can easily establish some important properties of the estimator $\hat{I}_N$.

1.1.1 Expectation, Variance and Central Limit Theorem

We have

$$\mathbb{E}\,\hat{I} = \mathbb{E}\left(\frac{1}{N}\sum_{i=1}^{N} f(X_i)\right) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\, f(X_i) = \frac{N}{N}\mathbb{E}\, f(x) = I.$$

Therefore, $\hat{I}_N$ is unbiased. Likewise, we have

$$\mathrm{Var}(\hat{I}_N) = \mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N} f(X_i)\right) = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}(f(X_i)) = \frac{1}{N}\mathrm{Var}(f(X)).$$

So the standard deviation is

$$\mathrm{Std}(\hat{I}_N) = \frac{\sqrt{\mathrm{Var}(f(X))}}{\sqrt{N}}.$$

Under suitable technical conditions (e.g. $\mathrm{Var}(f(X)) < \infty$), a central limit theorem holds and we additionally get that

$$\left(\hat{I}_N - I\right) \xrightarrow{D} \mathcal{N}\left(0, \frac{\mathrm{Var}(f(X))}{N}\right) \qquad \text{as } N \to \infty.$$

Example 1.1.1

$$\text{Estimate} \qquad \int_0^1 e^{-x^2}\,\mathrm{d}x$$

Listing 1.1: Matlab Code

```
N = 10^5;
X = rand(N,1);
f_X = exp(-X.^2);
I_hat = mean(f_X)
```

1.1.2 Higher Dimensional Integration Problems

Monte Carlo integration is especially useful for solving higher dimensional integration problems. For example, we can estimate integrals of the form

$$I = \int_0^1 \int_0^1 \cdots \int_0^1 f(x_1, \ldots, x_n)\, \mathrm{d}x_1\, \mathrm{d}x_2 \cdots \mathrm{d}x_n.$$

Similarly to the one-dimensional case, observe that $I = \mathbb{E}\, f(\mathbf{X})$, where $\mathbf{X}$ is a *vector* of iid $\mathcal{U}(0,1)$ random variables. This expected value can be approximated by

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^{N} f\left(\mathbf{X}_i\right),$$

where the $\{\mathbf{X}_i\}_{i=1}^N$ are iid n-dimensional vectors of iid $\mathcal{U}(0,1)$ random variables.

Example 1.1.2

Estimate $$\int_0^1 \int_0^1 e^{x_1} \cos(x_2)\, \mathrm{d}x_1\, \mathrm{d}x_2.$$

Listing 1.2: Matlab Code

```
N = 10^6;
f_X = zeros(N,1);
for i = 1:N
    X = rand(1,2);
    f_X(i) = exp(X(1))*cos(X(2));
end
I_hat = mean(f_X)
```

1.2 Further Reading

Very good reviews of Monte Carlo methods can be found in [1, 4, 7, 10, 15, 17, 18]. If you are interested in mathematical tools for studying Monte Carlo methods, a good book is [8].

Chapter 2

Pseudo Random Numbers

The important ingredient in everything discussed so far is the ability to generate iid $\mathcal{U}(0, 1)$ random variables. In fact, almost everything we do in Monte Carlo begins with the assumption that we can generate $\mathcal{U}(0, 1)$ random variables. So, how do we generate them?

Computers are not inherently random, so we have two choices:

1. Using a physical device that is 'truly random' (e.g. radioactive decay, coin flipping).
2. Using a sequence of numbers that are not truly random but have properties which make them seem/act like 'random numbers'.

Possible problems with the physical approach are:

1. The phenomenon may not be 'truly random' (e.g., coin flipping may involve bias).
2. Measurement errors.
3. These methods are slow.
4. By their nature, these methods are not reproducible.

A problem with the deterministic approach is, obviously, the numbers are not truly random.

The choice of a suitable random number generation method depends on the application. For example, the required properties of a random number generator used to generate pseudo random numbers for Monte Carlo methods are very different from those of a random number generator used to create pseudo random numbers for gambling or cryptography.

2.1 Requirements for Monte Carlo

In Monte Carlo applications there are many properties we might require of a random number generator (RNGs). The most import are:

1. The random numbers it produces should be uniformly distributed.
2. The random numbers it produces should be independent (or at least they should seem to be independent).
3. It should be fast.
4. It should have a small memory requirement.
5. The random numbers it produces should have a large period. This means that, if we use a deterministic sequence that will repeat, it should take a long time before it starts repeating.

Ideally, the numbers should be **reproducible** and the algorithm to generate them should be **portable** and should **not produce 0 or 1**.

2.2 Pseudo Random Numbers

2.2.1 Abstract Setting

S	finite set of states,
f	transition function $f : S \to S$,
S_0	a seed,
U	output space,
g	output function $g : S \to U$.

Algorithm 2.2.1 (General Algorithm)

1. Initialize: Set $X_1 = S_0$. Set $t = 2$.
2. Transition: Set $X_t = f(X_{t-1})$.
3. Output: Set $U_t = g(X_t)$.
4. Set $t = t + 1$. Repeat from 2.

2.2.2 Linear Congruential Generators

The simplest useful pseudo random number generator is a Linear Congruential Generator (LCG).

Algorithm 2.2.2 (Basic LCG)

1. Initialize: Set $X_1 = S_0$. Set $t = 2$.
2. Transition: Set $X_t = f(X_{t-1}) = (aX_{t-1} + c) \mod m$.

3. Output: Set $U_t = g(X_t) = \frac{X_t}{m}$.
4. Set $t = t + 1$ and repeat from step 2.

We call a the **multiplier** and c the **increment**.

Example 2.2.1
Take $a = 6$, $m = 11$, $c = 0$ and $S_0 = 1$.
Then

$$\begin{array}{ll} X_1 = 1 & U_1 = 1/11 \\ X_2 = (6 \cdot 1) \mod 11 = 6 & U_2 = 6/11 \\ X_3 = (6 \cdot 6) \mod 11 = 3 & U_3 = 3/11 \\ X_4 = (6 \cdot 3) \mod 11 = 7 & U_4 = 7/11 \end{array}$$

Sequence: $\underbrace{1, 6, 3, 7, 9, 10, 5, 8, 4, 2}_{\text{period } = 10 = (m-1)}, 1, 6, \ldots$

Listing 2.1: Matlab Code

```
N = 10;
a = 6; m = 11; c = 0;
S_0 = 1;
X = zeros(N,1); U = zeros(N,1);
X(1) = S_0;
for i = 2:N
    X(i) = mod(a*X(i-1)+c,m);
    U(i) = X(i)/m;
end
```

What happens if we take $m = 11$, $a = 3$, $c = 0$, $S_0 = 1$?

$$\begin{array}{l} X_1 = 1 \\ X_2 = (3 \cdot 1) \mod 11 = 3 \\ X_3 = (3 \cdot 3) \mod 11 = 9 \\ X_4 = (3 \cdot 9) \mod 11 = 5 \\ X_5 = (3 \cdot 5) \mod 11 = 4 \\ X_6 = (3 \cdot 4) \mod 11 = 1 \end{array}$$

Sequence: $\underbrace{1, 3, 9, 5, 4}_{\text{period } = 5}, 1, \ldots$

When using an LCG, it is important to have as long a period as possible. The following theorems give conditions for the maximum possible periods to be obtained.

Theorem 2.2.1 *An LCG with $c = 0$ has full period $(m - 1)$ if*

1. $S_0 \neq 0$.
2. $a^{m-1} - 1$ *is a multiple of* m.

3. *a^{j-1} is not a multiple of m for $j = 1, \ldots, m-2$.*

Theorem 2.2.2 *An LCG with $c \neq 0$ has full period (m) if and only if*

1. *c and m are relatively prime (Their only common divisor is 1).*
2. *Every prime number that divides m divides $a - 1$.*
3. *$a - 1$ is divisible by 4 if m is.*

The conditions are broken for the examples above (with $c \neq 0$) because 11 divides $m = 11$ but not $a = 3$ or $a = 6$. However, we know that the Theorem 2.2.2 is satisfied if c is odd, m is a power of 2 and $a = 4n + 1$.

Many LCGs have periods of $2^{31} - 1 \approx 2.1 \times 10^9$

Example 2.2.2 Minimal Standard LCG: $a = 7^5 = 16807$, $c = 0$, $m = 2^{31}$

In many modern Monte Carlo applications, samples sizes of $N = 10^{10}$ or bigger are necessary. A rough rule of thumb is that the period should be around N^2 or N^3. Thus, for $N \approx 10^{10}$ a period bigger than 10^{20} or even 10^{30} is required.

2.2.3 Extensions of LCGs

Multiple Recursive Generators

Algorithm 2.2.3 (Multiple Recursive Generator (MRG) of order k)

1. Initialize: Set $X_1 = S_0^1, \ldots, X_k = S_0^k$. Set $t = k + 1$.
2. Transition: $X_t = (a_1 X_{t-1} + \cdots + a_k X_{t-k}) \mod m$.
3. Output: $U_t = \frac{X_t}{m}$
4. Set $t = t + 1$. Repeat from step 2.

Usually, most of the $\{a_i\}_{i=1}^k$ are 0. Clearly, an LCG (with $c = 0$) is a special case of an MRG. Note that the state space is now $\{0, \ldots, m-1\}^k$ and the maximum period is now $m^k - 1$. It is obtained, e.g., if:

a) m is prime
b) The polynomial $p(z) = z^k - \sum_{i=1}^{k-1} a_i z^{k-i}$ is prime using modulo m arithmetic.

Obviously, $m^k - 1$ can be much bigger than m or $m - 1$.

Combined Generators

An example of a combined random number generator is the Wichmann-Hill pseudo random number generator.

Algorithm 2.2.4 (Wichmann-Hill)

1. Set $X_0 = S_0^X, Y_0 = S_0^Y, Z_0 = S_0^Z$. Set $t = 1$.
2. Set $X_t = a_1 X_{t-1} \mod m_1$
3. $Y_t = a_2 Y_{t-1} \mod m_2$
4. $Z_t = a_3 Z_{t-1} \mod m_3$
5. $U_t = \left(\frac{X_t}{m_1} + \frac{Y_t}{m_2} + \frac{Z_t}{m_3}\right) \mod 1$
6. Set $t = t + 1$ and repeat from step 2.

Pierre L'Ecuyer combines multiple recursive generators.

2.2.4 The Lattice Structure of Linear Congruential Generators

LCGs have what is known as a lattice structure.

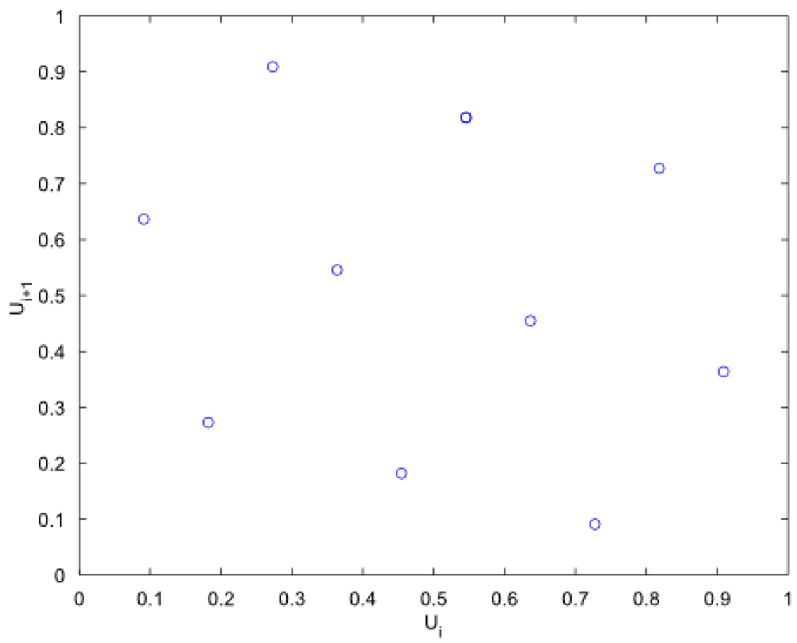


Figure 2.1: Points generated by LCG with $a = 6$, $c = 0$ and $m = 11$.

In higher dimensions, d-tuples – e.g., $(X_1, X_2), (X_2, X_3), (X_3, X_4) \ldots$ – lie on hyperplanes. It can be shown that points created by linear congruential generation with modulus m lie on at most $(d!m)^{\frac{1}{d}}$ hyperplanes in the d-dimensional unit cube. For $m = 2^{31} - 1$ and $d = 2$, they lie on at most ≈ 65536 hyperplanes. For $d = 10$ they lie on at most only ≈ 39 hyperplanes.

One way to asses the quality of a random number generator is to look at the "spectral gap" which is the maximal distance between two hyperplanes.

2.2.5 Linear Feedback Shift Register Type Generators

Each X_i can take values in $\{0, 1\}$. We update the X_i using

$$X_i = (a_1 X_{i-1} + \cdots + a_k X_{i-k}) \mod 2,$$

where the a_i take values in $\{0, 1\}$. We output

$$U_i = \sum_{j=1}^{L} X_{i\nu+j-1} 2^{-j},$$

where ν is the step size and $L \leq k$ is the word length (usually 32 or 64). Often $\nu = L$. Using this approach, we get a period of $2^k - 1$. The Mersenne Twister does a few additional things but is roughly of this form.

2.3 Testing Pseudo Random Number Generators

It is not enough for a random number generator to have a large period. It is even more important that the resulting numbers appear to be independent and identically distributed. Of course, we know that the numbers are not really random. However, a good random number generator should be able to fool as many statistical (and other) tests as possible into thinking that the numbers it produces are iid uniforms.

2.3.1 Testing the Sizes of the Gaps between Hyperplanes

This is the only non-statistical test we will discuss. Remember, that many random number generators have a lattice structure.

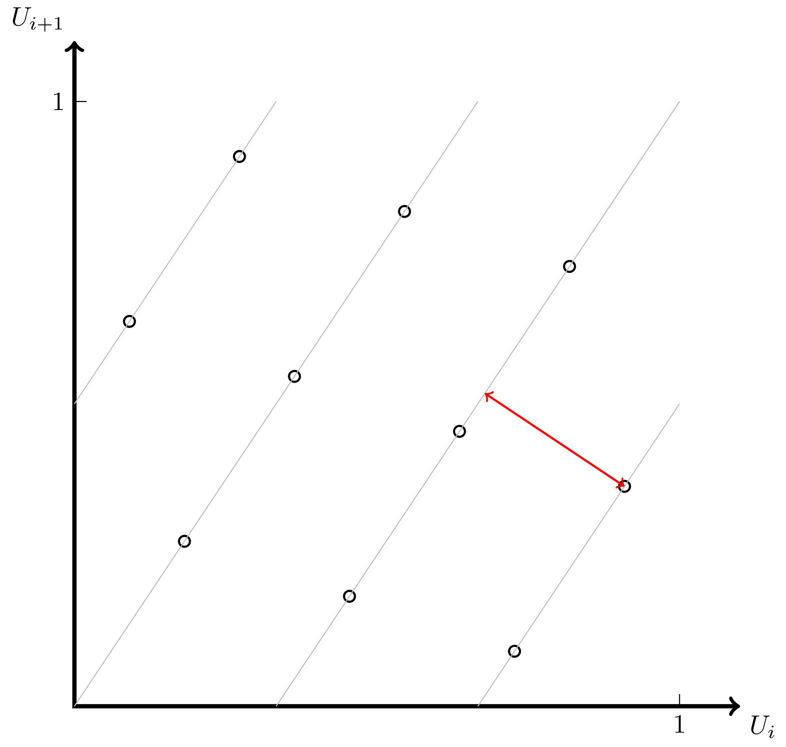


Figure 2.2: Lattice Structure of a Random Number Generator

In general, a lattice structure is bad, because it means numbers are not sufficiently independent of one another. We cannot avoid having a lattice structure, but we would like to have as many hyperplanes as possible (this means that patterns of association between random variables are less strong). 'Spectral' tests measure the gaps between hyperplanes. The mathematics involved in these tests is quite complicated.

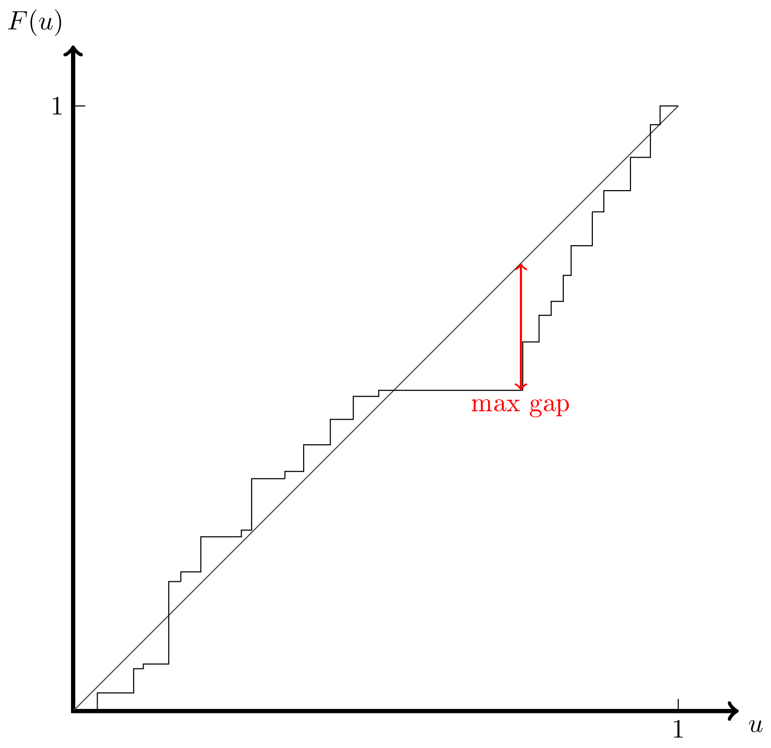
2.3.2 The Kolmogorov-Smirnov Test

This test checks if the output of a RNG is close enough to the uniform distribution. The idea of the Kolmogorov-Smirnov test is to compare the estimated cumulative distribution function (cdf) of the output of a random number generator against the cdf of the uniform (0,1) distribution. If the two cdfs are too different from one another, we say that the RNG does not produce uniform random variables. The test statistic is

$$D_n := \sup_{u \in (0,1)} |F_n(u) - F(u)|,$$

where F_n is an estimate of the cdf given the data (often called the empirical cdf). This statistic shouldn't be too big.

Example 2.3.1 (The Kolmogov-Smirnov statistic)



2.3.3 Chi-Squared Test

The Chi-Squared test is another test to check that the output of a random number generator has a uniform $(0, 1)$ distribution. We can test if a given sample $\{U_n\}_{n=1}^N$ is uniform by dividing it into equally spaced intervals.

Example 2.3.2 (An example subdivision)

$$\begin{aligned}
Q_1 &= \text{ number of points in } \{U_n\}_{n=1}^N \text{ between 0 and 0.1} \\
Q_2 &= \text{ number of points in } \{U_n\}_{n=1}^N \text{ between 0.1 and 0.2} \\
&\vdots \\
Q_{10} &= \text{ number of points in } \{U_n\}_{n=1}^N \text{ between 0.9 and 1}
\end{aligned}$$

If the given sample is uniformly distributed, the expected number of points in the ith interval (E_i) is the length times the number of points in the sample $(|Q_i|N)$. In the example, $E_1 = E_2 = \cdots = E_{10} = \frac{N}{10}$.

The Chi-Squared Test statistic is

$$\chi^2_{\text{stat}} = \sum_{i=1}^{L} \frac{(Q_i - E_i)^2}{E_i},$$

where L is the number of segments (in the example, $L = 10$). If χ^2_{stat} is too big, the random number generator does not appear to be producing uniform random variables.

2.3.4 Permutation Test

For iid random variables, all orderings should be equally likely.

For example, if X_1, X_2, X_3 are iid R.V.s, then

$$\mathbb{P}(X_1 \leq X_2 \leq X_3) = \mathbb{P}(X_2 \leq X_1 \leq X_3) = \mathbb{P}(X_3 \leq X_2 \leq X_1) = \dots$$

Let Π_d be the indices of an ordering of d iid uniformly distributed random variables. For example, if $X_1 \leq X_2 \leq X_3$, $\Pi_3 = \{1, 2, 3\}$.

It's not hard to see that

$$\mathbb{P}(\Pi = \pi) = \frac{1}{d!}. \tag{2.1}$$

This suggests we can test a random number generator by generating N runs of d random variables and then doing a Chi-Squared test on them to check whether their ordering indices are distributed according to (2.1).

2.4 Quasi Monte Carlo

So far we have considered too much structure in a random number generator as a negative thing (that is, we have wanted things to be as random as possible). However, another approach to Monte Carlo – called Quasi-Monte Carlo (QMC) – tries to use structure / non-independence to improve the performance of Monte Carlo methods.

2.4.1 Numerical Integration and Problems in High Dimensions

Monte Carlo integration is not very efficient in low dimensions. Instead of Monte Carlo integration, one can use Newton Cotes methods. These methods evaluate the function at equally spaced points. There are also fancier numerical methods like Gaussian quadrature.

Using Newton Cotes methods like the rectangle method or the trapezoidal method, one needs

$$\begin{array}{lcl} \text{in 2 D} & n \times n & \text{points} \\ \text{in 3 D} & n^3 & \text{points} \\ & \vdots & \\ \text{in } d \text{ D} & n^d & \text{points.} \end{array}$$

An exponential growth of points is required. This is an example of the "curse of dimensionality".

For most numerical integration methods, the error of the approximated integral is roughly proportional to $n^{-c/d}$ for some constant c. As d gets larger, the error decays more slowly. In comparison, the error of Monte Carlo integration (measured by standard deviation) is proportional to $n^{-1/2}$. In high enough dimensions, this will be smaller than the error of numerical integration.

2.4.2 The Basic Idea of QMC

Quasi Monte Carlo methods generate deterministic sequences that get rates of error decay close to n^{-1} (as compared to $n^{-1/2}$ for standard Monte Carlo methods). They perform best in reasonably low dimensions (say about 5 to 50). One disadvantage is that they need a fixed dimension (some Monte Carlo methods do not work in a fixed dimension). This means it is not always possible to use them.

A grid is a bad way to evaluate high dimensional integrals. The idea of QMC is that the points should be spread out more efficiently. A good spread means here a low 'discrepancy'.

Definition 2.4.1
Given a collection $\mathcal{A}$ of (Lebesgue measurable) subsets of $[0,1)^d$, the discrepancy relative to $\mathcal{A}$ is

$$D(x_1,\dots,x_n;\mathcal{A}) = \sup_{A\in\mathcal{A}} \left| \frac{\#\{x_i \in A\}}{n} - \text{vol}(A) \right|.$$

Basically, discrepancy measures the difference between the number of points that should be in each of the sets if the points are evenly spaced, and the number of points that are actually in these sets.

Example 2.4.1 'Ordinary discrepancyÂ´, is based on sets of the form

$$\prod_{j=1}^{d} [u_j, v_j) \qquad 0 \le u_j < v_j \le 1.$$

2.4.3 Van der Corput Sequences

A number of QMC methods are based on the Van der Corput sequences (which have low discrepancy). The idea is to write numbers $1, 2, \dots$ in base b and then 'flip them' and reflect them over decimal points.

Example 2.4.2 The van der Corput sequence with $b = 2$

$$\begin{aligned}
1 &= 001.0 \quad \Rightarrow 0.100 & \left(= \frac{1}{2}\right) \\
2 &= 010.0 \quad \Rightarrow 0.010 & \left(= \frac{1}{4}\right) \\
3 &= 011.0 \quad \Rightarrow 0.110 & \left(= \frac{3}{4}\right) \\
4 &= 100.0 \quad \Rightarrow 0.001 & \left(= \frac{1}{8}\right) \\
& \vdots &
\end{aligned}$$

Example 2.4.3 The van der Corput sequence with $b = 3$

$$\begin{aligned}
1 &= 001.0 \quad \Rightarrow 0.100 & \left(= \frac{1}{3}\right) \\
2 &= 002.0 \quad \Rightarrow 0.020 & \left(= \frac{2}{3}\right) \\
3 &= 010.0 \quad \Rightarrow 0.010 & \left(= \frac{1}{9}\right) \\
4 &= 011.0 \quad \Rightarrow 0.110 & \left(= \frac{4}{9}\right)
\end{aligned}$$

2.4.4 Halton Sequences

Probably the simplest QMC method is called the Halton sequence. It fills a d dimension cube with points whose coordinates follow Van Der Corput sequences. For example, we can sample points $(x_1, y_1), (x_2, y_2), \ldots$, where the x coordinates follow a Van der Corput sequence with base b_1 and the y coordinates follow a Van der Corput sequence with base b_2. The b_i are chosen to be relatively prime, which means that they have no common divisors.

2.5 Furter Reading

Important books on random number generation and QMC include [6, 11, 13]

Chapter 3

Non-Uniform Random Variables

3.1 The Inverse Transform Method

3.1.1 The Inverse Transform Method

The distribution of a random variable is determined by its cumulative distribution function (cdf), $F(x) = \mathbb{P}(X \leq x)$.

Theorem 3.1.1 *F is a cdf if and only if*

1. *$F(x)$ is non-decreasing,*
2. *$F(x)$ is right-continuous,*
3. *$\lim_{x \to -\infty} F(x) = 0$ and $\lim_{x \to \infty} F(x) = 1$.*

Definition 3.1.1 We define the inverse of $F(x)$ as,

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

F^{-1} is sometimes called the generalized inverse or quantile function.

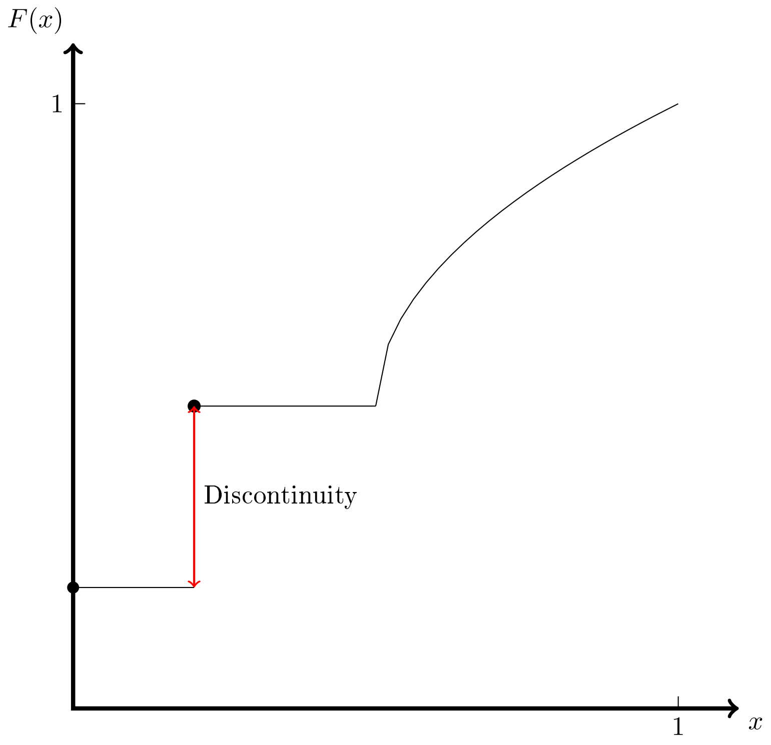


Figure 3.1: A cdf with a discontinuity at $x = 0.2$.

Theorem 3.1.2 *If $F(x)$ is a cdf with inverse $F^{-1}(y)$ and $U \sim \mathcal{U}(0,1)$ then, $F^{-1}(U) \sim F$.*

Proof We have $\mathbb{P}\left(F^{-1}(U) \leq x\right) = \mathbb{P}\left(U \leq F(x)\right)$ via the definition of the inverse. Now, $\mathbb{P}\left(U \leq u\right) = u$ for $u \in [0,1]$. So, $\mathbb{P}\left(U \leq F(x)\right) = F(x)$. □

This leads us to the inverse transform algorithm.

Algorithm 3.1.1 (Inverse Transform)

1. Generate $U \sim \mathcal{U}(0,1)$.

2. Return $X = F^{-1}(U)$.

Example 3.1.1 (Exponential distribution)
Let $X \sim \text{Exp}(\lambda),\ \lambda > 0$. The cdf of an exponential random variable is $F(x) = \left(1 - e^{-\lambda x}\right)\mathbb{I}(x \geq 0)$.

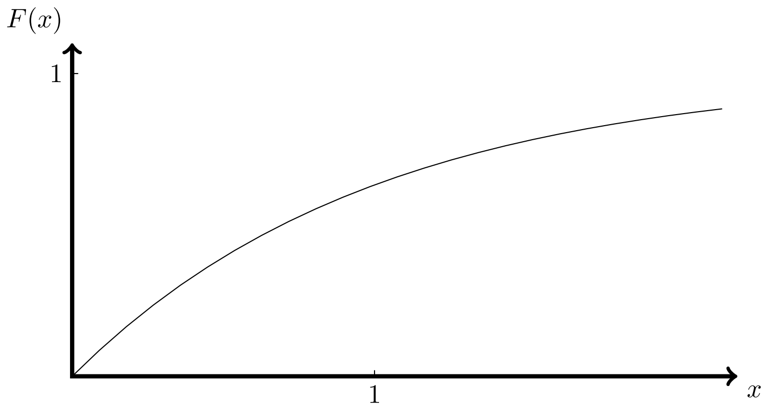


Figure 3.2: cdf of an exponential distributed random variable.

We have to calculate $F^{-1}(U)$.

$$U = 1 - e^{-\lambda X} \Rightarrow 1 - U = e^{-\lambda X}$$

$1 - U$ is also $\sim \mathcal{U}(0,1)$, so

$$U = e^{-\lambda X} \Rightarrow \log(U) = -\lambda X \Rightarrow X = -\frac{1}{\lambda}\log(U)$$

This is easily implemented in Matlab.

Listing 3.1: Matlab Code

```
lambda = 1;
X = -1/lambda*log(rand);
```

Example 3.1.2
Consider the discrete random variable X where

$$\begin{aligned} X &= 0 \quad \text{with probability } \tfrac{1}{2} \\ X &= 1 \quad \text{with probability } \tfrac{1}{3} \\ X &= 2 \quad \text{with probability } \tfrac{1}{6} \end{aligned}$$

The cdf of X is

$$F(x) = \begin{cases} 0 & \text{if } x < 0 \\ \frac{1}{2} & \text{if } 0 \le x < 1 \\ \frac{5}{6} & \text{if } 1 \le x < 2 \\ 1 & \text{if } x \ge 2 \end{cases} .$$

The inverse, F^{-1}, is given by

$$F^{-1}(u) = \begin{cases} 0 & \text{if } u \leq \frac{1}{2} \\ 1 & \text{if } \frac{1}{2} < u \leq \frac{5}{6} \\ 2 & \text{if } u > \frac{5}{6} \end{cases} .$$

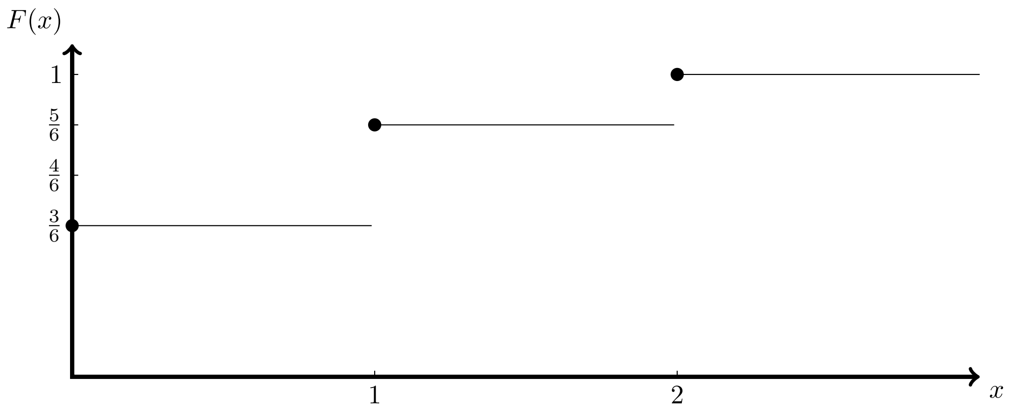


Figure 3.3: The cdf of X

This example can be implemented in Matlab.

Listing 3.2: Matlab Code

```
N = 10^5; X = zeros(N,1);

for i = 1:N
    U = rand;
    if U <= 1/2;
        X(i) = 0;
    end
    if U > 1/2 && U < 5/6
        X(i) = 1;
    end
    if U > 5/6
        X(i) = 2;
    end
end
```

3.1.2 Integration over Unbounded Domains

Now that we are able to generate non-uniform random variables, we can use Monte Carlo to integrate over unbounded domains.

In 1.1 we discussed how to integrate

$$\int_0^1 S(x)\,\mathrm{d}x \qquad \text{or} \qquad \int_a^b S(x)\,\mathrm{d}x.$$

In 1.1.2 we did the same for n-dimensional integrals. Now we we want to compute improper integrals of the form

$$\int_0^\infty S(x)\,\mathrm{d}x.$$

We need to write this as an expectation but that does not work with uniform distributed random variables. However we can use a random variable whose density f has the support $[0,\infty)$ (that is $f(x) > 0$ for all $x \in [0,\infty)$). An example would be the exponential density. Then we can write

$$\int_0^\infty S(x)\,\mathrm{d}x = \int_0^\infty \frac{S(x)}{f(x)} f(x)\,\mathrm{d}x = \mathbb{E}\,\frac{S(X)}{f(X)},$$

where $X \sim f$. Actually, we can relax the condition about the support so that we simply need a density such that $f(x) = 0 \Rightarrow S(x) = 0$.

Remark When calculating an integral in this fashion, we need to think carefully about our choice of f. A bad choice can give very high (possibly even infinite) variance.

Example 3.1.3 Estimate

$$\int_0^\infty x^2 e^{-x^3}\,\mathrm{d}x.$$

We can use the exponential distribution density with $\lambda = 1$ and write

$$\int_0^\infty \frac{x^2 e^{-x^3}}{e^{-x}} \cdot e^{-x}\,\mathrm{d}x = \int_0^\infty x^2 e^{-(x^3-x)} e^{-x}\,\mathrm{d}x = \mathbb{E}\left(X^2 e^{-(X^3-X)}\right).$$

Where $X \sim \mathrm{Exp}(1)$.

3.1.3 Truncated Distributions

Consider the conditional density $f(x \,|\, X \in [a,b])$. We can write this as

$$f(x \,|\, X \in [a,b]) = \frac{f(x)}{\mathbb{P}\left(X \in [a,b]\right)} \qquad a \le x \le b$$

Let $Z \sim f_Z(x) = f(x|X \in [a,b])$. Then

$$F_Z(x) = \frac{F(x) - F(a^-)}{F(b) - F(a^-)}.$$

Where

$$F(a^-) = \lim_{x \uparrow a} F(x).$$

We can use the inverse transform method on this cdf to generate replicates of Z.

3.1.4 The Table Lookup Method

As Example 3.1.2 makes clear, sampling from a finite discrete distribution using the inverse transform method is essentially just checking a lot of 'if' statements. One way to make this faster is to look values up in a table. The table lookup method uses a rule to randomly generate an index in the table so that the desired random variables are produced with the appropriate probabilities. It only works if the probabilities of the random variable are all rational.

Algorithm 3.1.2 (Table Lookup Method)

1. Draw $U \sim \mathcal{U}(0,1)$.
2. Set $I = \lceil nU \rceil$.
3. Return a_I.

Example 3.1.4 Let X be a random variable with the following distribution:

$$\begin{aligned}\mathbb{P}(X=0) &= \frac{1}{4} \quad (=\tfrac{5}{20})\\ \mathbb{P}(X=1) &= \frac{1}{5} \quad (=\tfrac{4}{20})\\ \mathbb{P}(X=2) &= \frac{11}{20}\end{aligned}$$

If we choose the following values, we will generate a random variable with the appropriate values.

$$\begin{aligned} n &= 20\\ a_1 = a_2 = \cdots = a_5 &\quad = 0\\ a_6 = a_7 = \cdots = a_9 &\quad = 1\\ a_{10} = a_{11} = \cdots = a_{20} &\quad = 2.\end{aligned}$$

3.1.5 Problems with the Inverse Transform Method

The inverse transformation method is not always a good method. For example, cdfs are often not known in an explicit form. An important example is the cdf of the standard normal distribution

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

It can sometimes be very computationally intensive to calculate inverse cdfs numerically. Also, in the discrete case, lookup tables do not always work (e.g. if the variable can take an infinite number of values).

3.2 Acceptance-Rejection

The other generic method of sampling a non-uniform random variable is the acceptance-rejection method. This relies on us knowing the density f of the variable we wish to sample. The basic idea is as follows, if we can sample points $(X_1, Y_1), (X_2, Y_2), \ldots$ uniformly from the set $\{(x, y) : x \in \mathbb{R}, 0 \leq y \leq f(x)\}$, which is known as the hypograph of f, then the $\{X_i\}$ will be samples from the density f. We do this using the following algorithm.

Algorithm 3.2.1 (Acceptance-Rejection Algorithm)

```
1. Generate $X \sim g(x)$.
2. Generate $U \sim \mathcal{U}(0,1)$ (independently of $X$).
3. If $U \leq \frac{f(X)}{Cg(X)}$ output $X$. Otherwise, return to step 1.
```

For this to work, we need $Cg(x) \geq f(x)\ \forall x \in \mathbb{R}$. The best possible choice of C is therefore $C = \max_{x \in \mathbb{R}} f(x)/g(x)$. If this doesn't exist, we need to find another choice of g.

The efficiency of the acceptance-rejection method is the percantage of proposals we accept. Roughly, this will be best when g is very close to f. We can calculate the acceptance probability exactly:

$$\mathbb{P}\left(\text{acceptance}\right) = \frac{\text{area under } f(x)}{\text{area under } Cg(x)} = \frac{1}{C}.$$

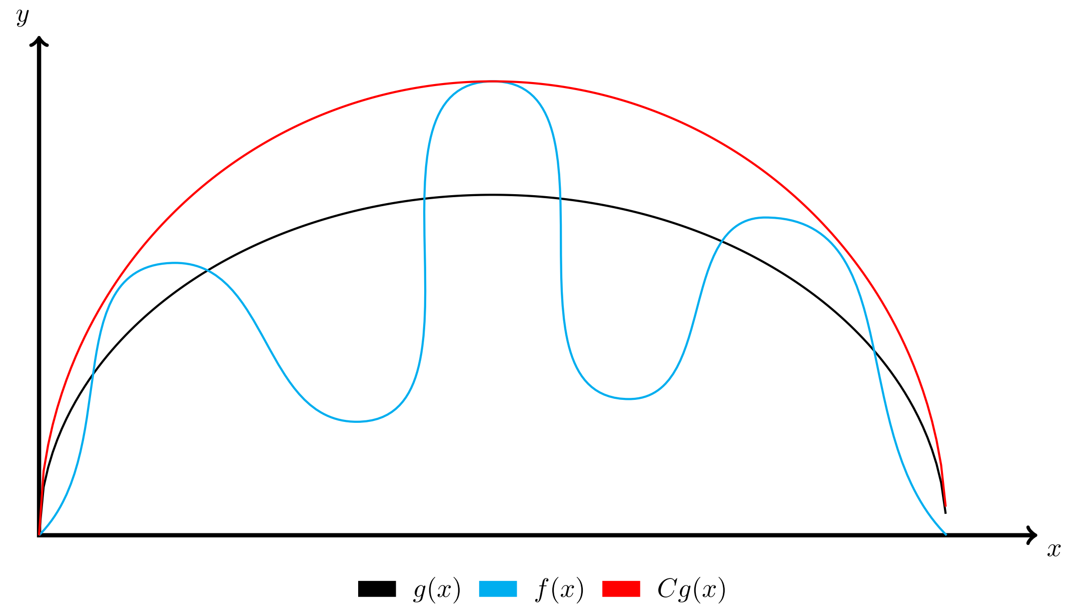


Figure 3.4: The acceptance-rejection method. We sample X from g then accept it with probability $f(X)/Cg(X)$.

Theorem 3.2.1 *The acceptance-rejection algorithm results in output with the desired distribution.*

Proof We are outputting $Y \sim g(x)$ conditioned on $U \leq \frac{f(Y)}{Cg(Y)}$

$$\mathbb{P}\left(Y \leq x \middle| U \leq \frac{f(Y)}{Cg(Y)}\right) = \frac{\mathbb{P}\left(Y \leq x, U \leq \frac{f(Y)}{Cg(Y)}\right)}{\mathbb{P}\left(U \leq \frac{f(Y)}{Cg(Y)}\right)}.$$

The numerator can be written as

$$\mathbb{P}\left(Y \leq x, U \leq \frac{f(Y)}{Cg(Y)}\right) = \int_{-\infty}^{x} \mathbb{P}\left(U \leq \frac{f(y)}{Cg(y)}\right) g(y)\,\mathrm{d}y$$
$$= \int_{-\infty}^{x} \frac{f(y)}{Cg(y)} g(y)\,\mathrm{d}y = \frac{1}{C}\int_{-\infty}^{x} f(y)\,\mathrm{d}y$$
$$= \frac{1}{C}F(x).$$

The denominator can be written as

$$\mathbb{P}\left(U \leq \frac{f(Y)}{Cg(Y)}\right) = \int_{-\infty}^{\infty} \mathbb{P}\left(U \leq \frac{f(y)}{Cg(y)}\right) g(y)dy$$
$$= \int_{-\infty}^{\infty} \frac{f(y)}{Cg(y)} g(y)\,\mathrm{d}y = \frac{1}{C}\int_{-\infty}^{\infty} f(y)\,\mathrm{d}y$$
$$= \frac{1}{C}.$$

So

$$\mathbb{P}\left(Y \leq x \middle| U \leq \frac{f(Y)}{Cg(Y)}\right) = \frac{1/C\,F(x)}{1/C} = F(x).$$

□

Example 3.2.1 (Positive normal distribution)
We wish to draw $X \sim \mathcal{N}(0,1)$ conditioned on $X \geq 0$.

$$f(x \,|\, X \geq 0) = \frac{\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}}{\mathbb{P}\left(X > 0\right)}\mathbb{I}(x \geq 0) = \frac{\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}}{\frac{1}{2}}\mathbb{I}(x \geq 0) = \sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}\mathbb{I}(x \geq 0).$$

We need to find C.

$$Cg(x) \geq f(x) \Rightarrow C\exp(-x) \geq \sqrt{\frac{2}{\pi}}\exp\left(-x^2\right) \Rightarrow C \geq \sqrt{\frac{2}{\pi}}\exp\left(\frac{-x^2}{2}+x\right).$$

The best choice of C is the maximum of the right-hand side. So we choose x such that

$$\frac{\partial}{\partial x}\left(\frac{-x^2}{2}+x\right) = -x+1 = 0 \Rightarrow x = 1.$$

We check second order conditions to make sure this is a maximum (though it is obvious here).

$$\frac{\partial^2}{\partial^2 x}\left(\frac{-x^2}{2}+x\right) = -1.$$

So

$$C = \sqrt{\frac{2}{\pi}}\exp\left(\frac{1}{2}\right) = \sqrt{\frac{2e}{\pi}}$$

Now

$$\frac{f(x)}{Cg(x)} = \frac{\overbrace{\sqrt{\frac{2}{\pi}}\exp\left(\frac{-x^2}{2}\right)}^{f(x)}}{\underbrace{\sqrt{\frac{2}{\pi}}\exp\left(\frac{1}{2}\right)}_{C}\underbrace{\exp(-x)}_{g(x)}} = \exp\left(-\frac{x^2}{2}+x-\frac{1}{2}\right).$$

Listing 3.3: Matlab Code

```
N = 10^5; X = zeros(N,1);
for i = 1:N
    Y = -log(rand);
    U = rand;
    while(U > exp(-Y^2/2+Y-1/2))
        Y = -log(rand);
        U = rand;
    end
    X(i) = Y;
end
```

We can use this approach to generate standard normals by exploiting the symmetry of the normal distribution.

1. Generate X as above.
2. Set $Z = \delta X$.

Where $\mathbb{P}(\delta = 1) = \mathbb{P}(\delta = -1) = 1/2$. We can generate δ using

$$\delta = 2\mathbb{1}\left(U \leq \frac{1}{2}\right) - 1$$

3.2.1 Drawing Uniformly from Regions of Space

The acceptance-rejection algorithm can also be used to sample uniformly from regions of space. Essentially, we draw uniformly from a box containing the object of interest, then only accept the points inside the object.

Example 3.2.2 (Drawing uniform random points from the unit ball)

1. Draw $X^* \sim \mathcal{U}(-1,1)$ (`X_star = 2*rand-1;`) and $Y^* \sim \mathcal{U}(-1,1)$ (`Y_star = 2*rand-1;`).
2. If $(X^*)^2 + (Y^*)^2 \leq 1$ return $X = X^*$ and $Y = Y^*$
   else, repeat from step 1.

The acceptance probability in this case is $\mathbb{P}\,(\text{acceptance}) = \frac{\pi}{4}$.

Example 3.2.3 (Drawing uniform random points from the unit sphere)

1. Draw $X^*, Y^*, Z^* \sim \mathcal{U}(-1,1)$
2. If $(X^*)^2 + (Y^*)^2 + (Z^*)^2 \leq 1$ return $X = X^*$, $Y = Y^*$, $Z = Z^*$
   else, repeat from step 1.

3.2.2 A Limitation of the Acceptance-Rejection Method

Consider the acceptance probability when drawing a point uniformly from a d-dimensional hypersphere. When $d = 3$, we have

$$\mathbb{P}\,(\text{acceptance}) = \frac{\text{vol sphere}}{\text{vol box}} = \frac{\frac{4}{3}\pi r^3}{2 \cdot 2 \cdot 2} = \frac{\frac{4}{3}\pi}{8} = \frac{\pi}{6} < \frac{\pi}{4}.$$

For general $d \in \mathbb{N}$:

$$\mathbb{P}\,(\text{acceptance}) = \frac{\text{vol hypersphere}}{\text{vol box}} = \frac{\frac{\pi^{\frac{d}{2}}}{\frac{d}{2}\Gamma(\frac{d}{2})}}{2^d} = \frac{\pi^{\frac{d}{2}}}{d 2^{d-1}\Gamma\left(\frac{d}{2}\right)} \longrightarrow 0 \text{ (quickly) as } d \to \infty.$$

What this example illustrates is that the acceptance-rejection method often works very badly in high dimensions.

3.3 Location-Scale Families

Definition 3.3.1 (Location-scale distributions) A family of continuous densities of the form

$$f(x;\mu,\sigma) = \frac{1}{\sigma}\mathring{f}\left(\frac{x-\mu}{\sigma}\right)$$

is called a location-scale family with base distribution $\mathring{f}$. It is shifted by μ and scaled by σ.

For a location scale family, if $X \sim \mathring{f} = f(x;0,1)$, then $\mu + \sigma X \sim f(x;\mu,\sigma)$.

Example 3.3.1 (The normal distribution)
If $Z \sim \mathcal{N}(0,1)$, then $X = \mu + \sigma Z \sim \mathcal{N}(\mu,\sigma^2)$.

There are a lot of location-scale distribution families: e.g., normal, Laplace, logistic, Cauchy. Some families are scale (but not location). For example, exponential, gamma, Pareto, Weibull.

Location-scale (and location) families are nice to work with, because we only have to worry about sampling from a base density (rather than from a density with arbitrary parameters).

3.4 Generating Normal Random Variables

To generate normal distributed random values using the inverse transform method either a good approximation of the inverse of the standard normal cdf is required or we need to do root-finding on the cdf (which requires a good approximation of the cdf). Nevertheless, many sophisticated modern algorithms do use the inverse transform method with a very accurate approximation of the inverse cdf. Another way to generate normal random variables is the Box-Muller method.

3.4.1 Box-Muller

The idea of the Box-Muller algorithm is based on the observation that the level sets of the joint probability density function (pdf) of two (independent) standard normal random variables $X, Y \sim \mathcal{N}(0,1)$ form circles. It is easy to generate points uniformly on a circle (just draw $\theta \sim \mathcal{U}[0, 2\pi]$), so we just need to find the right distribution for the radius of the circle.

Transforming to polar coordinates,

$$\begin{aligned} X &= R\cos\theta \\ Y &= R\sin\theta \\ \theta &\sim \mathcal{U}(0, 2\pi) \\ R &= \sqrt{X^2 + Y^2} \end{aligned}$$

We can calculate the cdf of R^2.

$$\mathbb{P}\left(R^2 \le x^2\right) = \mathbb{P}\left(R \le x\right) = \int_0^x \int_0^{2\pi} \frac{r}{2\pi} \exp\left(-\frac{r^2}{2}\right) d\theta dr = \int_0^x r\exp\left(-\frac{r^2}{2}\right) dr = 1 - \exp\left(-\frac{x^2}{2}\right)$$

It is clear from this that R^2 has an exponential distribution with parameter $\lambda = \frac{1}{2}$. This gives the following algorithm

Algorithm 3.4.1 (Box-Muller-Algorithm)

1. Generate $\theta \sim \mathcal{U}(0, 2\pi)$.
2. Generate $R^2 \sim \text{Exp}\left(\frac{1}{2}\right)$.
3. Return $Z_1 = \sqrt{R^2}\cos\theta$ and $Z_2 = \sqrt{R^2}\sin\theta$.

Listing 3.4: Matlab Code

```
N = 10^5; X = zeros(N,1);
for i = 1:N/2
    theta = 2*pi*rand;
    r_squared = -2*log(rand);
    Z1 = sqrt(r_squared)*cos(theta);
    Z2 = sqrt(r_squared)*sin(theta);
    X(2*i-1) = Z1;
    X(2*i) = Z2;
end
```

This algorithm is expensive to use because it involves the special functions sin, cos and log.

3.4.2 Generating Multivariate Normals

Consider now the more general problem of generating $Z \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We can use the location scale property of the normal density to generate these variables, but only if we are able to calculate the ‘squareÂ´ of the Variance-Covariance matrix. That is, we need to write $\Sigma = AA^\intercal$. Writing Σ in this way is called the Cholesky factorization of Σ (Matlab has a function to compute this). We can compute the Cholesky factorization of $\boldsymbol{\Sigma} = AA^T$ if $\boldsymbol{\Sigma}$ is positive definite. This leads to the following algorithm.

Algorithm 3.4.2

1. Generate $\mathbf{Z} \sim \mathcal{N}(0, 1)$.
2. Calculate $\boldsymbol{\Sigma} = AA^T$.
3. Output $X = \boldsymbol{\mu} + A\mathbf{Z}$.

This works because affine transformations are normal and the resulting mean and variance are correct (these uniquely describe a multivariate normal distribution). The mean is easy to check. For the variance,

$$\begin{aligned}\mathrm{Var}(X) &= \mathbb{E}\,(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^\intercal = \mathbb{E}\,(\boldsymbol{\mu} + A\mathbf{Z} - \boldsymbol{\mu})(\boldsymbol{\mu} + A\mathbf{Z} - \boldsymbol{\mu})^\intercal \\ &= \mathbb{E}\,[(A\mathbf{Z})(A\mathbf{Z})^\intercal] = \mathbb{E}\,[A\mathbf{Z}^\intercal\mathbf{Z}A^\intercal] = A\mathbb{E}\,[\mathbf{Z}^\intercal\mathbf{Z}]\,A^\intercal \\ &= AA^\intercal = \boldsymbol{\Sigma}\end{aligned}$$

3.5 Further Reading

The classic reference on non-uniform random numbers is [3].

Chapter 4

Markov Chains

So far, we have only considered sequences of independent random variables (with the exception of the multivariate normal distribution). Dependent random variables are much harder to model and to simulate. However, some dependency structures are easier to work with than others (but can still model lots of interesting phenomena). The most important models of dependent random variables from a simulation perspective are Markov chains.

There are many reasons why Markov chains are useful (from a Monte Carlo perspective).

- It is difficult to simulate non-Markovian stochastic processes.
- More complicated processes can sometimes be approximated by Markov Processes.
- Markov chains can be used to simulate from complex distributions and build complex random objects.
- Markov chains can be used to explore the parameter spaces of functions and look for optima (e.g., stochastic optimization/ genetic algorithms).

This chapter is just a brief description of Markov chains, which only focuses on things we need to know. It is not intended as a complete introduction. Most of the material in this chapter is based on the excellent book [14]. The chapter on discrete time chains is available on the internet (see the course website for a link). Other good books are [12, 2, 19].

4.1 Definitions

We will focus on Markov chains in discrete time with a countable state space. We will refer to these objects as Markov chains, without any qualifying terms. People argue about whether the term 'chain' means that time is countable or the state space is countable. Either way, the objects we are discussing are definitely 'Markov chains'. A Markov chain is a stochastic process. That is, a sequence of random variables $(X_i)_{i \in I}$, whose index set I often represents time (e.g., X_1 happens before X_2, which happens before X_3 and so on).

Definition 4.1.1 (Markov chain)
Let $(X_n)_{n\in\mathbb{N}}$ be a stochastic process taking values in a countable state space S (i.e., all the $(X_n)_{n\in\mathbb{N}}$ take values in S). We say $(X_n)_{n\in\mathbb{N}}$ is a Markov Chain if $\forall n > 0$ and $(i_0, \dots, i_n, j) \in S^{n+2}$

$$\mathbb{P}\left(X_{n+1} = j \,|\, X_0 = i_0, \dots, X_n = i_n\right) = p_{i_n,j}.$$

It's not hard to see that Definition 4.1.1 implies that

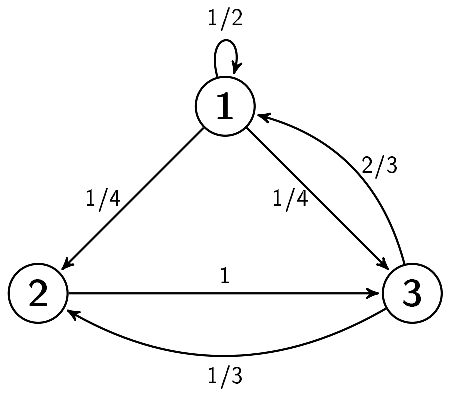
$$\mathbb{P}\left(X_{n+1} = j \,|\, X_0 = i_0, \dots, X_n = i_n\right) = \mathbb{P}\left(X_{n+1} = j \,|\, X_n = i_n\right).$$

That is, the probability of going from state i_n to state j does not depend on where the chain has been before (its history).

Example 4.1.1 (Simple random walk)
Let $S = \mathbb{Z}$. Set $X_0 = 0$ and $X_{n+1} = X_n + (2Z_n - 1)$ for $n \geq 0$, where $(Z_n)_{n\in\mathbb{N}}$ is a sequence of iid Bernoulli(1/2) random variables. In this case we have,

$$\mathbb{P}\left(X_{n+1} = j \,|\, X_0 = i_0, \dots, X_n = i_n\right) = \begin{cases} 1/2 & \text{if } i_n = j+1 \\ 1/2 & \text{if } i_n = j-1 \\ 0 & \text{otherwise} \end{cases}$$
$$= \mathbb{P}\left(X_{n+1} = j \,|\, X_n = i_n\right).$$

Example 4.1.2 (A graphical representation of a Markov chain)



The transition probabilities of a Markov chain can be thought of as a matrix $P = (p_{i,j})$, known as the **transition matrix**.

Example 4.1.3 (Graphical representation of a Markov chain, continued)
The chain in Example 4.1.2 has the transition matrix

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 1 \\ \frac{2}{3} & \frac{1}{3} & 0 \end{pmatrix} \end{array} \qquad \begin{array}{l} \sum_{i=1}^{3} p_{1,i} = 1 \\ \sum_{i=1}^{3} p_{2,i} = 1 \\ \sum_{i=1}^{3} p_{3,i} = 1 \end{array}$$

In the case of an infinite state space, this matrix will be infinite.

Definition 4.1.2 (Stochastic matrix)
A matrix with non-negative real values entries and rows that sum to 1 is called a (right) **stochastic matrix**.

4.2 Simulation

Algorithm 4.2.1 (Simulating a Markov Chain)

1. Draw X_0 from $\boldsymbol{\lambda}$ (the initial distribution of X_0. Set $i = 1$.
2. Set $X_{i+1} = j$ with probability $p_{X_i,j}$.
3. Set $i = i + 1$. If $i < N$ repeat from step 2.

Example 4.1.2 can be simulated in Matlab.

Listing 4.1: Matlab Code

```
N = 10^5; X = zeros(N,1);
X(1) = ceil(rand*3); i =1;
P = [1/2 1/4 1/4; 0 0 1; 2/3 1/3 0];
while i<N
   X(i+1) = min(find(rand<cumsum(P(X(i),:))));
   i = i+1;
end
```

4.3 Calculating Probabilities

When working with Markov Chains, we represent probability distributions as row vectors. A probability distribution will usually be denoted $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots)$, where $\lambda_i = \mathbb{P}(X = i)$. Two obvious requirements are $\lambda_i \geq 0$ for all i and $\Sigma_i \lambda_i = 1$.

We can define the probabilities of various paths of a Markov chain. If X_0 is drawn from some distribution $\boldsymbol{\lambda}$, then

$$\begin{aligned}
&\mathbb{P}(X_0 = i_0, X_1 = i_1, \cdots, X_N = i_N) \\
=&\mathbb{P}(X_0 = i_0)\mathbb{P}(X_1 = i_1 | X_0 = i_0)\cdots\mathbb{P}(X_N = i_N | X_0 = i_0, X_1 = i_1, \cdots, X_{N-1} = i_{N-1}) \\
=&\mathbb{P}(X_0 = i_0)\mathbb{P}(X_1 = i_1 | X_0 = i_0)\cdots\mathbb{P}(X_N = i_N | X_{N-1} = i_{N-1}) \\
=&\lambda_{i_0} p_{i_0,i_1} \cdots p_{i_{N-1},i_N}.
\end{aligned}$$

We say a Markov chain with initial distribution $\boldsymbol{\lambda}$ and transition matrix P is a $(\boldsymbol{\lambda}, P)$ Markov chain.

We can compute in what state we are likely to be after one step.

$$\begin{aligned}
\mathbb{P}(X_1 = 1) =&\mathbb{P}(X_0 = 1)\mathbb{P}(X_1 = 1 | X_0 = 1) + \mathbb{P}(X_0 = 2)\mathbb{P}(X_1 = 1 | X_0 = 2) + \cdots \\
=&\lambda_1 p_{1,1} + \lambda_2 p_{2,1} + \cdots = \sum_{i_0 \in S} \lambda_{i_0} p_{i_0,1}
\end{aligned}$$

If we do this for $\mathbb{P}(X_1 = 2)$, $\mathbb{P}(X_1 = 3)$, and so on, we see that

$$\left(\sum_{i_0 \in S} \lambda_{i_0} p_{i_0,1}, \sum_{i_0 \in S} \lambda_{i_0} p_{i_0,2}, \dots \right) = \boldsymbol{\lambda} P$$

If we take another step with the initial distribution $\boldsymbol{\lambda} P$ the next distribution is $\boldsymbol{\lambda} P^2$.

Recursively one can see that the distribution after n steps is

$$\boldsymbol{\lambda} P^n \text{ with } p_{i,j}^{(n)} = \mathbb{P}(X_n = j | X_0 = i).$$

4.3.1 Ways to calculate P^n

When S is finite:

1. Use a computer to find P^n (without thinking too hard about it).
2. Diagonalization
3. Recursion.

Diagonalization

If a square matrix has a complete basis of eigenvectors, one can write

$$P = QDQ^{-1},$$

where D is a matrix with the eigenvalues of P on the diagonal and zeros elsewhere and Q is a matrix with the eigenvectors of P as columns. A nice result is, that

$$P^n = QD^nQ^{-1}$$

Example 4.3.1 Let

$$P = \begin{pmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{2}{5} & \frac{3}{5} \end{pmatrix}$$

So, to find the eigenvalues, we need to solve

$$\left(\frac{1}{4} - \lambda\right)\left(\frac{3}{5} - \lambda\right) - \left(\frac{2}{5}\right)\left(\frac{3}{4}\right) = 0 \Rightarrow \lambda = 1, -\frac{3}{20}.$$

To get the eigenvectors, we solve $P\mathbf{u} = \lambda_i \mathbf{u}$, $i = 1, 2$

$$\begin{array}{llll} \lambda_1 : & \frac{1}{4}u_1 + \frac{3}{4}u_2 = u_1 \Rightarrow & u1 = u2 \Rightarrow & \mathbf{u} = \left(1, 1\right)^T \\ \lambda_2 : & \frac{1}{4}u_1 + \frac{3}{4}u_2 = -\frac{3}{20}u_1 \Rightarrow & u1 = -\frac{8}{15}u2 \Rightarrow & \mathbf{u} = \left(1, -\frac{8}{15}\right)^T. \end{array}$$

So

$$Q = \begin{pmatrix} 1 & 1 \\ 1 & -\frac{8}{15} \end{pmatrix}, \; D = \begin{pmatrix} 1 & 0 \\ 0 & -\frac{3}{20} \end{pmatrix}, \; P = QDQ^{-1}$$

$$P^n = Q \begin{pmatrix} (1)^n & 0 \\ 0 & (-\frac{3}{20})^n \end{pmatrix} Q^{-1}.$$

Note that, when the eigenvalues are distinct, the matrix P is of the form

$$p_{i,j}^{(n)} = a_1\lambda_1^n + \cdots + a_m\lambda_m^n.$$

This is slightly more complicated with repeating eigenvalues.

Recursion

In general $p_{i,j}^n = p_{i,1}^{(n-1)}p_{1,j} + \cdots + p_{i,k}^{(n-1)}p_{k,j} = \sum_{k\in S} p_{i,k}^{(n-1)}p_{k,j}$. It is sometimes possible to solve this recursion. This is mostly useful for calculating some quantities we will not focus on, such as absorption probabilities.

Example 4.3.2 (2 states)
We have

$$p_{1,2}^{(n)} = p_{1,2}^{(n-1)}p_{2,2} + p_{1,1}^{(n-1)}p_{1,2}.$$

Now

$$p_{1,1}^{(n-1)} + p_{1,2}^{(n-1)} = 1 \Rightarrow p_{1,1}^{(n-1)} = 1 - p_{1,2}^{(n-1)}.$$

So we can write

$$p_{1,2}^{(n)} = p_{1,2}^{(n-1)}p_{2,2} + \left(1 - p_{1,2}^{(n-1)}\right)p_{1,2} = (p_{2,2} - p_{1,2})p_{1,2}^{(n-1)} + p_{1,2}$$

This recursion can be solved.

4.4 Asymptotic behavior of Markov Chains

We are often interested in the behavior of a Markov chain as $n \to \infty$. We will introduce two important asymptotic quantities, the **stationary distribution** and the **limiting distribution**.

4.4.1 Class Structure

The class structure of a Markov chain gives us important information about its asymptotic behavior.

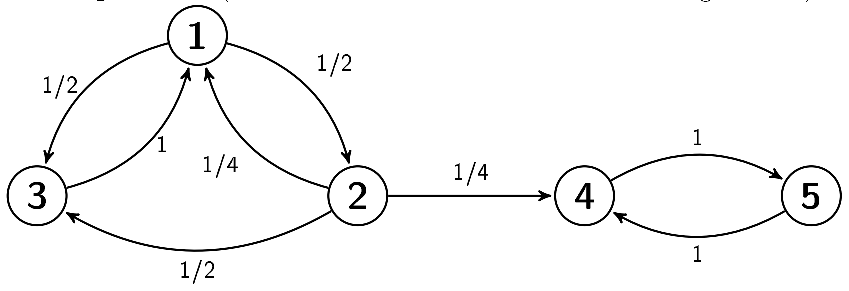
We can break a Markov Chain up into a number of separate components called **communicating classes**.

We say i **leads** to j ($i \to j$) if $\mathbb{P}(X_n = j \text{ some } n \geq 0 \,|\, X_0 = i) > 0$.

We say i **communicates** with j, and write $i \leftrightarrow j$, if both $i \to j$ and $j \to i$.

A state space S can be partitioned into communicating classes $C_1, C_2, \ldots$. A chain where S is a single class is called **irreducible**.

Example 4.4.1 (A Markov chain with two communicating classes.)



The state space of this chain can be separated in two communicating classes, $C_1 = \{1, 2, 3\}$ and $C_2 = \{4, 5\}$.

We say a communicating class C is **closed** if $i \in C, i \to j \Rightarrow j \in C$.

A state i is **recurrent** if $\mathbb{P}(X_n = i \text{ for infinitely many } n \,|\, X_0 = i) = 1$.

A state i is **transient** if $\mathbb{P}(X_n = i \text{ for infinitely many } n \,|\, X_0 = i) = 0$.

Theorem 4.4.1 *Let C be a communicating class, then all states in C are transient or all states are recurrent.*

4.4.2 Invariant Measures

Definition 4.4.1 (Invariant measure)
We say a measure, $\boldsymbol{\nu}$, by which we mean a non-trivial (that is, not $\mathbf{0}$) vector with non-negative components, is invariant if $\boldsymbol{\nu} P = \boldsymbol{\nu}$.

If $\boldsymbol{\nu}$ is a measure and $\sum_{i \in S} \nu_i = 1$ then $\boldsymbol{\nu}$ is a probability distribution (I will try to denote invariant probability distributions by $\boldsymbol{\pi}$).

Given a measure $\boldsymbol{\nu}$ with $\sum_{i \in S} \nu_i < \infty$, we can obtain an invariant probability distribution $\boldsymbol{\pi}$ by setting $\pi_i = \nu_i / \sum_{i \in S} \nu_i$.

Remember that the distribution of X_n for a $(\boldsymbol{\lambda}, P)$ Markov chain is $\boldsymbol{\lambda} P^n$. Because $\boldsymbol{\pi} P = \boldsymbol{\pi}$, the distribution of X_n if $X_0 \sim \boldsymbol{\pi}$ is still $\boldsymbol{\pi}$. The distribution does not change. For this reason, $\boldsymbol{\pi}$ is called a **stationary distribution** of the Markov chain.

Two obvious questions are under what circumstances $\boldsymbol{\pi}$ exists and whether or not $\boldsymbol{\pi}$ is unique.

For finite matrices one can always get a non-trivial vector $\mathbf{u}$ so that $\mathbf{u}P = \mathbf{u}$ (however, this vector is not guaranteed to be non-negative). We can see this by observing that

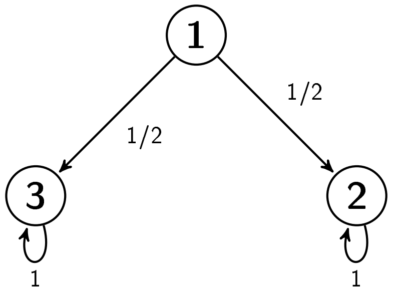
$$P \begin{pmatrix} 1 \\ 1 \\ \vdots \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & \cdots \\ p_{21} & p_{22} & \\ \vdots & & \ddots \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \end{pmatrix} = \begin{pmatrix} p_{11} + p_{12} + \cdots \\ p_{21} + p_{22} + \cdots \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \end{pmatrix}.$$

So we get a right eigenvector $(1, 1, \dots)^T$ of P with the right eigenvalue 1. A right eigenvalue is also a left eigenvalue, so there exists a vector $\mathbf{u}$ such that $\Rightarrow \mathbf{u}P = 1\mathbf{u}$.

We still have no guarantee that $\mathbf{u}$ is non-negative (and this argument doesn't work for infinite state spaces). Even if we can get a non-negative $\boldsymbol{\nu}$ such that $\sum_{i\in S} \nu_i = 1$, we don't necessarily get a unique $\boldsymbol{\pi}$.

Example 4.4.2 (A non-unique stationary distribution)

Consider the Markov chain



The transition matrix of this chain is

$$P = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Solving for the eigenvector corresponding to the eigenvalue 1, we get

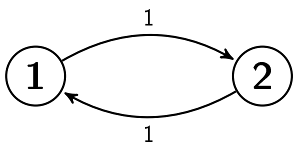
$$\begin{pmatrix} u_1 & u_2 & u_3 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & u_2 & u_3 \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & u_3 \end{pmatrix}.$$

We see that u_2 and u_3 are unrestricted. So, we have an infinite number of $\boldsymbol{\pi}$s, e.g., $\boldsymbol{\pi} = (0,1,0)$, $\boldsymbol{\pi} = (0,0,1)$ or $\boldsymbol{\pi} = (0, \frac{1}{2}, \frac{1}{2})$.

Even if $\boldsymbol{\pi}$ is unique, it is not necessarily a **limiting distribution** (by which, we mean $\lim_{n\to\infty} \boldsymbol{\mu} P^n = \boldsymbol{\pi}$ for all $\boldsymbol{\mu}$).

Example 4.4.3 (No limiting distribution)

Consider the Markov chain



which has transition matrix

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Solving for the eigenvector corresponding to the eigenvalue 1, we get

$$\begin{aligned} &\begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} u_2 & u_1 \end{pmatrix} = \begin{pmatrix} u_1 & u_2 \end{pmatrix}. \\ &\Rightarrow u_1 = u_2 \Rightarrow \boldsymbol{\pi} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \end{pmatrix}. \end{aligned}$$

However, no limiting distribution exists as

$$P^n = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ if } n \text{ is odd and } P^n = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ if } n \text{ is even .}$$

We can show the **existence** of an invariant and non-negative measure if P is irreducible and recurrent. With these conditions, we can also get a **uniqueness** result. But, for infinite state spaces, this doesn't guarantee a probability distribution (we need one more condition to get this).

We will make extensive use of the vector $\boldsymbol{\mu}^k$, which we define to be

$$\boldsymbol{\mu}_i^k = \mathbb{E}_k \sum_{n=0}^{T_k - 1} \mathbb{1}\{X_n = i\},$$

where $\mathbb{E}_k$ means the expected value whern $X_0 = k$ and $T_k = \inf\{n \geq 1 : X_n = k\}$. This is the average spent in i between trips to k

Theorem 4.4.2 (The 'existence' result)
Let P be irreducible and recurrent. Then

(i) $\mu_k^k = 1$

(ii) $\boldsymbol{\mu}^k P = \boldsymbol{\mu}^k$

(iii) $0 < \mu_i^k < \infty \ \forall i \in S$.

Proof

(i) We can only spend 1 step in k before we visit k again, so $\mu_k^k = 1$.

(ii) Note that $\mathbb{E}_k \sum_{n=0}^{T_{k-1}} \mathbb{1}\{X_n = i\} = \mathbb{E}_k \sum_{n=1}^{T_k} \mathbb{1}\{X_n = i\}$, so

$$\begin{aligned} \mu_j^k =& \mathbb{E}_k \sum_{n=1}^{T_k} \mathbb{1}\{X_n = j\} = \mathbb{E}_k \sum_{n=1}^{\infty} \mathbb{1}\{X_n = j, n \leq T_k\} = \sum_{i \in S} \sum_{n=1}^{\infty} \mathbb{P}(X_{n-1} = i, X_n = j, n \leq T_k) \\ =& \sum_{i \in S} \sum_{n=1}^{\infty} \mathbb{P}(X_{n-1} = i, n \leq T_k) \mathbb{P}(X_n = j | X_{n-1} = i) = \sum_{i \in S} p_{i,j} \sum_{n=1}^{\infty} \mathbb{P}(X_{n-1} = i, n \leq T_k) \\ =& \sum_{i \in S} p_{i,j} \mathbb{E}_k \sum_{n=0}^{\infty} \mathbb{1}\{X_n = i, n \leq T_{k-1}\} = \sum_{i \in S} p_{i,j} \mathbb{E}_k \sum_{n=0}^{T_k - 1} \mathbb{1}\{X_n = i\} \\ =& \sum_{i \in S} \mu_i^k p_{i,j}. \end{aligned}$$

(iii) P is irreducible. So, for each state i, there exist $n, m \geq 0$ so that $p_{i,k}^{(n)}$, $p_{k,i}^{(m)} > 0$. By (ii) we know

$$\mu_i^k = \sum_{j \in S} \mu_j^k p_{j,i}^{(m)} \geq \mu_k^k p_{k,i}^{(m)} > 0$$

and

$$1 = \mu_k^k = \sum_{j \in S} \mu_j^k p_{j,i}^{(n)} \geq \mu_i^k p_{i,k}^{(n)}.$$

So

$$0 < \mu_i^k \leq \frac{1}{p_{i,k}^{(n)}} < \infty.$$

□

Theorem 4.4.3 (The 'uniqueness' result)
Let P be irreducible and $\boldsymbol{\nu}$ an invariant measure for P with $\nu_k = 1$. Then $\boldsymbol{\nu} \geq \boldsymbol{\mu}^k$ (element wise). If P is also recurrent, $\boldsymbol{\nu} = \boldsymbol{\mu}^k$.

Proof
For each $j \in S$ we have

$$\begin{aligned}
\nu_j &= \sum_{i_1 \in S} \nu_{i_1} p_{i_1,j} = \sum_{i_1 \neq k} \nu_{i_1} p_{i_1,j} + p_{k,j} \\
&= \sum_{i_1 \neq k} \sum_{i_2 \neq k} \nu_{i_2} p_{i_2,i_1} p_{i_1,j} + \left(p_{k,j} + \sum_{i_1 \neq k} p_{k,i_1} p_{i_1,j} \right) \\
&\vdots \\
&= \sum_{i_1 \neq k} \cdots \sum_{i_n \neq k} \nu_{i_n} p_{i_n,i_{n-1}} \dots p_{i_1,j} + \left(p_{k,j} + \sum_{i_1 \neq k} p_{k,i_1} p_{i_1,j} + \dots + \sum_{i_1 \neq k} \cdots \sum_{i_{n-1} \neq k} p_{k,i_{n-1}} \cdots p_{i_2,i_1} p_{i_1,j} \right).
\end{aligned}$$

For $j \neq k$

$\nu_j \geq \mathbb{P}_k(X_1 = j \text{ and } T_k \geq 1) + \mathbb{P}_k(X_2 = j, T_k \geq 2) + \cdots + \mathbb{P}_k(X_n = j, T_k \geq n) \longrightarrow \mu_j^k$ as $n \to \infty$.

For $j = k$ we already have equality.

For P recurrent (and already irreducible) $\boldsymbol{\mu}^k$ is invariant. Define $\mathbf{w} = \boldsymbol{\nu} - \boldsymbol{\mu}^k$. $\mathbf{w}$ is invariant, as $\mathbf{w}P = (\boldsymbol{\nu} - \boldsymbol{\mu}^k)P = \boldsymbol{\nu}P - \boldsymbol{\mu}^k P = \boldsymbol{\nu} - \boldsymbol{\mu}^k = \mathbf{w}$. The result above implies that $\mathbf{w} \geq 0$.

As P is irreducible, for all i in S we can find an $n > 0$ such that $p_{i,k}^{(n)} > 0$ and

$$0 = w_k = \sum_{j \in S} w_j p_{i,j}^{(n)} \geq w_i p_{i,k}^{(n)}.$$

Now, $\mathbf{w} = 0 = \boldsymbol{\nu} - \boldsymbol{\mu}^k \Rightarrow \boldsymbol{\nu} = \boldsymbol{\mu}^k$ □

Remark For infinite state spaces, Theorem 4.4.3 doesn't guarantee that $\boldsymbol{\mu}^k$ can be turned into a probability measure as

$$0 < \mu_i^k < \infty \ \forall \ i \in S \not\Rightarrow \sum_{i \in S} \mu_i^k < \infty.$$

(Consider, for example, random walks).

Definition 4.4.2 Recurrence, which was defined as $\mathbb{P}_i(X_n = i$ for infinitely many $n) = 1$, is equivalent to $\mathbb{P}_i(T_i < \infty) = 1$. Define $m_i = \mathbb{E}_i T_i$. We say i is **positive recurrent** if $m_i < \infty$. Otherwise, i is **null recurrent**.

Theorem 4.4.4 *If P is irreducible, the following are equivalent.*

(i) *Every state is positive recurrent.*

(ii) *Some state k is positive recurrent.*

(iii) *P has an invariant distribution $\boldsymbol{\pi}$.*

Proof

- $(i) \to (ii)$ is obvious.
- $(ii) \to (iii)$: k is positive recurrent and therefore **recurrent**. $\Rightarrow P$ irreducible and recurrent. By Theorem 4.4.2 we know that an invariant measure $\boldsymbol{\mu}^k$ exists. Now

$$\sum_{j \in S} \mu_j^k = m_k < \infty,$$

So

$$\left(\frac{\mu_1^k}{\sum_{i \in S} \mu_i^k}, \frac{\mu_2^k}{\sum_{i \in S} \mu_i^k}, \dots \right) = \left(\frac{\mu_1^k}{m_k}, \frac{\mu_2^k}{m_k}, \dots \right) \Rightarrow \pi_i = \frac{\mu_i^k}{m_k},$$

- $(iii) \to (i)$: Take state k. P is irreducible and

$$\sum_{i \in S} \pi_i = 1 \Rightarrow \pi_k = \sum_{i \in S} \pi_i p_{i,k}^{(n)} > 0 \text{ for some } n.$$

Set $\lambda_i = \frac{\pi_i}{\pi_k}$. λ is invariant with $\lambda_k = 1$. By Theorem 4.4.3, we can say $\lambda \geq \mu^k$. So

$$m_k = \sum_{i \in S} \mu_i^k \leq \sum_{i \in S} \frac{\pi_i}{\pi_k} = \frac{1}{\pi_k} < \infty.$$

$\Rightarrow k$ is positive recurrent and P is recurrent. So, by Theorem 4.4.3,

$$\lambda = \mu^k \Rightarrow m_k = \sum_{i \in S} \mu_i^k = \sum_{i \in S} \frac{\pi_i}{\pi_k} = \frac{1}{\pi_k}.$$

□

Example 4.4.4 An infinite state space Markov chain that is positive recurrent for $p \in (0,1)$

$$P = \begin{pmatrix} 1-p & p & 0 & 0 & 0 & \cdots \\ 1-p & 0 & p & 0 & 0 & \cdots \\ 1-p & 0 & 0 & p & 0 & \\ 1-p & 0 & 0 & 0 & p & \\ \vdots & \vdots & & & \ddots & \ddots \end{pmatrix}$$

4.4.3 Limiting Distributions

In order to determine whether a chain has a limiting distribution or not (does it converge to the stationary distribution?), we need to determine the period of its states.

Definition 4.4.3
Define

$$\mathcal{T}(i) = \left\{ n \geq 0 : p_{ii}^{(n)} > 0 \right\}.$$

The **period** of state i is the greatest common divisor of $\mathcal{T}(i)$.

Lemma 4.4.1 *In an irreducible chain, all states have the same period.*

Proof Take $i, j \in S$. Irreducibility implies that there exist m, n so that $p_{ij}^{(m)} > 0$ and $p_{ji}^{(n)} > 0$. Let d be a common divisor of $\mathcal{T}(i)$. Then

$$\forall\ k \in \mathcal{T}(j) : p_{ii}^{(m+k+n)} \geq p_{ij}^{(m)} p_{jj}^{(k)} p_{ji}^{(n)} > 0 \Rightarrow m+k+n \in \mathcal{T}(i)$$

$\Rightarrow d$ divides $\{m+n+k : k \in \mathcal{T}(j)\}$, but $m+n \in \mathcal{T}(i)$ so d divides $m+n \Rightarrow d$ must divide $k \Rightarrow d$ must divide $\mathcal{T}(j) \Rightarrow d(i) \leq d(j)$, where $d(i)$ is the greatest common divisor of i. Do reverse and get $d(i) = d(j)$. □

Remark A chain is **aperiodic** if the period of all states is 1.

Now we need to define what we mean by "convergence to the stationary distribution".

Definition 4.4.4 Let $\mathbf{p}^1$ and $\mathbf{p}^2$ be probability distributions on S. The **total variation distance** between $\mathbf{p}^1$ and $\mathbf{p}^2$ is

$$\left\| \mathbf{p}^1 - \mathbf{p}^2 \right\|_{\text{TV}} = \max_{A \subset S} \left| \mathbf{p}^1(A) - \mathbf{p}^2(A) \right|.$$

Proposition 4.4.1

$$\left\| \boldsymbol{p}^1 - \boldsymbol{p}^2 \right\|_{TV} = \frac{1}{2} \sum_{i \in S} \left| p_i^1 - p_i^2 \right|.$$

Proof Let $B = \left\{ i : p_i^1 > p_i^2 \right\}$. $A \subset S$ any Event. Now,

$$p^1(A) - p^2(A) \leq p^1(A \cap B) - p^2(A \cap B) \leq p^1(B) - p^2(B).$$

Likewise,

$$p^2(A) - p^1(A) \leq p^2(B^c) - p^1(B^c).$$

And

$$\left[p^2(B^c) - p^1(B^c)\right] - \left[p^1(B) - p^2(B)\right] = 1 - p^2(B) - 1 + p^1(B) - p^1(B) + p^2(B) = 0.$$
$$\Rightarrow p^1(B) - p^2(B) = p^2(B^c) - p^1(B^c).$$

Now

$$\max_{A \subset S} \left|p^1(A) - p^2(A)\right| = \max_{A \subset S} \max\left\{p^1(A) - p^2(A), p^2(A) - p^1(A)\right\}$$
$$= \frac{1}{2}\left[p^1(B) - p^2(B) + p^2(B^c) - p^1(B^c)\right] = \frac{1}{2}\sum_{i \in S}\left|p_i^1 - p_i^2\right|$$

□

Theorem 4.4.5 (The main result) *Let P be a irreducible, aperiodic and positive recurrent. Then*

$$\lim_{n\to\infty} \|\boldsymbol{\mu} P^n - \pi\|_{TV} = 0$$

for all $\boldsymbol{\mu}$, where $\boldsymbol{\pi}$ is the unique stationary distribution.

4.4.4 Reversibility and Detailed Balance

Remark (Reversibility) A Markov chain run backwards is also a Markov chain. Of particular interest to us, is the behavior of such Markov chains at stationarity.

Theorem 4.4.6 *Let P be irreducible with invariant distribution $\boldsymbol{\pi}$. Suppose $(X_n)_{0\leq n\leq N}$ is Markov (π, P). Set $Y_n = X_{N-n}$. Then $(X_n)_{0\leq n\leq N}$ is Markov $(\pi, \widehat{P})$. Where $\widehat{P}$ is given by*

$$\pi_j \widehat{p}_{ji} = \pi_i p_{ij}.\ \forall\ i, j \in S.$$

$\widehat{P}$ is also irreducible with the invariant distribution $\boldsymbol{\pi}$.

Definition 4.4.5 (Detailed Balance) A matrix P and a measure $\boldsymbol{\nu}$ are in detailed balance if

$$\nu_i p_{ij} = \nu_j p_{ji}.\ \forall\ i, j \in S.$$

Definition 4.4.6 (Reversible) Let $(X_n)_{n\geq 0}$ be Markov $(\boldsymbol{\lambda}, P)$, with P irreducible. $(X_n)_{n\geq 0}$ is reversible if, for all $N \geq 1$, $(X_{N-n})_{0\leq n\leq N}$ is also Markov $(\boldsymbol{\lambda}, P)$.

Theorem 4.4.7 *Let $(X_n)_{n\geq 0}$ be Markov $(\boldsymbol{\lambda}, P)$. It is reversible $\Longleftrightarrow$ P and $\boldsymbol{\lambda}$ are in detailed balance.*

Theorem 4.4.8 *If $\boldsymbol{\lambda}$ and P are in detailed balance, then $\boldsymbol{\lambda}$ is invariant*

Proof

$$(\lambda P)_i = \sum_{j \in S} \lambda_j p_{ji} = \sum_{j \in S} \lambda_i p_{ij} = \lambda_i$$

□

Chapter 5

Markov Chain Monte Carlo

So far, we have tried to find the stationary distribution, $\boldsymbol{\pi}$, of a given Markov Chain. In Monte Carlo, we are usually interested in the inverse problem. Given a distribution $\boldsymbol{\pi}$, we wish to construct a Markov chain with stationary distribution $\boldsymbol{\pi}$. The methods for doing this are called Markov Chain Monte Carlo (MCMC) methods.

5.1 The Metropolis-Hastings Algorithm for Countable State Spaces

5.1.1 The Metropolis Algorithm

The basic idea of many MCMC methods is to use detailed balance to construct a Markov chain with the desired stationary distribution. That is, given $\boldsymbol{\pi}$, we need to find a transition matrix P, such that $\pi_i p_{ij} = \pi_j p_{ji}$. Nicholas Metropolis suggested the first method of doing this. His idea is the following.

1. Given the chain is in state i, a possible state to jump to, j, is proposed according to the transition matrix Q, where $q_{ij} = q_{ji}$ (that is, Q is symmetric). Q needs to be positive recurrent and irreducible on S for everything to work.
2. With probability $\alpha = \min\left\{1, \frac{\pi_j}{\pi_i}\right\}$ the chain jumps to j. Otherwise, it stays in i.

This gives a Markov chain with transition probabilities $p_{ij} = q_{ij} \min\left\{1, \frac{\pi_j}{\pi_i}\right\}$.

Theorem 5.1.1 *The transition matrix described above is in detailed balance with $\boldsymbol{\pi}$ (and thus $\boldsymbol{\pi}$ is its stationary distribution).*

Proof That is, we need to show $\pi_i p_{ij} = \pi_j p_{ji}$, i.e., $\pi_i q_{ij} \min\left\{1, \frac{\pi_j}{\pi_i}\right\} = \pi_j q_{ji} \min\left\{1, \frac{\pi_i}{\pi_j}\right\}$.

Cases

- $\pi_j > \pi_i$:

$$\pi_i q_{ij} \min\left\{1, \frac{\pi_j}{\pi_i}\right\} = \pi_i q_{ij} = \pi_i q_{ij} \frac{\pi_j}{\pi_j} = \pi_j q_{ji} \min\left\{\frac{\pi_i}{\pi_j}, 1\right\}.$$

- $\pi_i \geq \pi_j$:

$$\pi_i q_{ij} \min\left\{1, \frac{\pi_j}{\pi_i}\right\} = \pi_i q_{ij} \frac{\pi_j}{\pi_i} = \pi_j q_{ji} = \pi_j q_{ji} \min\left\{\frac{\pi_i}{\pi_j}, 1\right\}.$$

□

5.1.2 The Metropolis-Hastings Algorithm

Hastings modified the algoritm to include the case where $q_{ij} \neq q_{ji}$ (that is, Q is not symmetric). He did this by using a different acceptance probabiltity:

$$\alpha = \pi_i q_{ij} \min\left\{1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right\}$$

Theorem 5.1.2 *The transition matrix described above is in detailed balance with $\boldsymbol{\pi}$ (and thus $\boldsymbol{\pi}$ is its stationary distribution).*

Proof The proof is more or less the same as the proof for Theorem 5.1.1.

Cases

- $\pi_j q_{ji} > \pi_i q_{ij}$:

$$\pi_i q_{ij} \min\left\{1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right\} = \pi_i q_{ij} = \pi_i q_{ij} \frac{\pi_j q_{ji}}{\pi_j q_{ji}} = \pi_j q_{ji} \min\left\{\frac{\pi_i q_{ij}}{\pi_j q_{ji}}, 1\right\}.$$

- $\pi_i q_{ij} \geq \pi_j q_{ji}$:

$$\pi_i q_{ij} \min\left\{1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right\} = \pi_i q_{ij} \frac{\pi_j q_{ji}}{\pi_i q_{ij}} = \pi_j q_{ji} = \pi_j q_{ji} \min\left\{\frac{\pi_i q_{ij}}{\pi_j q_{ji}}, 1\right\}.$$

□

5.1.3 A Classical Setting for the Metropolis-Hastings Algorithm

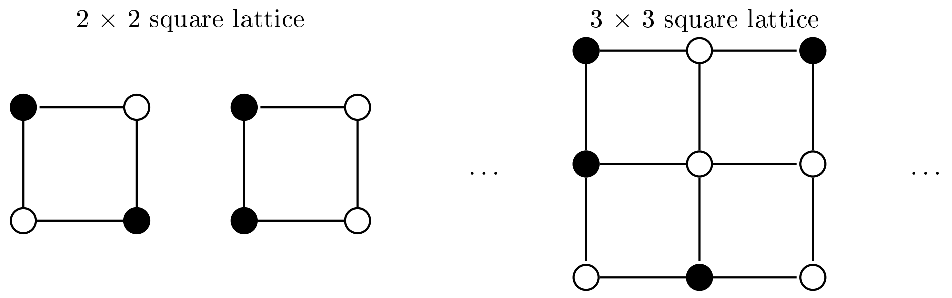
Consider a collection of objects $\{x_i\}_{i \in S}$, indexed by i and in some state space S. Let X be a random variable, taking on values according to

$$\pi_i = \mathbb{P}(X = x_i) = \frac{1}{Z_T} \exp\left\{-\frac{1}{T}\mathcal{E}(x_i)\right\}, \tag{5.1}$$

where Z_T is called the partition function and $\mathcal{E}$ is called the energy function. The partition function is the normalizing constant of this distribution. That is,

$$Z_T = \sum_{i \in S} \exp\left\{-\frac{1}{T}\mathcal{E}(x_i)\right\}$$

In this setting, we are often interested in quite complicated objects (e.g. random graphs, random fields etc). A reasonably simple example is fixed graphs with each vertex assigned -1 or 1. For example, the following.



Even in such simple settings $|S|$, the number of possible values of X, is very large. For an $N \times N$ square lattice we have $2^{N \times N}$ combinations. So, for example, when $N = 5$, there are $2^{25} = 33554432$ possible values. As you can see, for a more interesting object, a 100×100 lattice for example (which could model a 100×100 pixel black and white image), the size of the state space is enormous. This means

$$Z_T = \sum_{i \in S} \exp\left\{-\frac{1}{T}\mathcal{E}(x_i)\right\}$$

can be very difficult to calculate if $\mathcal{E}$ is complicated.

Markov Chains Monte Carlo methods such as the Metropolis-Hastings algorithm allow us to draw samples according to the distribution given in (5.1). Using these samples, and standard Monte Carlo estimators, we can estimate things like expected values of functionals (that we could not calculate otherwise in the absence of a normalizing constant).

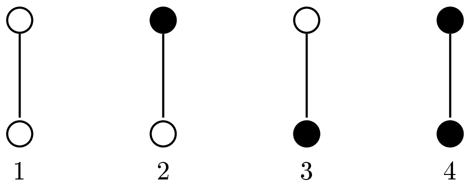
The reason for this is that we do not need to know the normalizing constant of a distribution in order to use MCMC. Let $\boldsymbol{\pi}$ be a distribution with $\pi_i \propto \lambda_i$, where $\sum_{i \in S} \lambda_i$ is unknown, then

$$\frac{\pi_j}{\pi_i} = \frac{\lambda_j / \sum\limits_{k \in S} \lambda_k}{\lambda_i / \sum\limits_{k \in S} \lambda_k} = \frac{\lambda_j}{\lambda_i}.$$

That is, we do not need to know the normalizing constant in order to calculate α (and thus generate a chain with the desired stationary distribution).

Example 5.1.1

Consider an extremely simple graph model, with only two vertices which can be marked 1 (white) or -1 (black). The possible states are:



We wish to sample from a distribution of the form (5.1) with $T = 1$ and $\mathcal{E}(x) = -\log(\#\text{ black in } x + 1)$. Because this is a very simple model, we can compute the distribution exactly.

$\pi_1 \propto \exp(-(-\log 1)) = 1, \pi_2 \propto 2, \pi_3 \propto 2, \pi_4 \propto 3$. This means that $Z_T = 8$, so $\pi_1 = 1/8, \pi_2 = 1/4, \pi_3 = 1/4$ and $\pi_4 = 3/8$.

However, we want to try to use MCMC methods to sample from this, pretending we do not know the normalizing constants (which would be true if the model had, say, 200 vertices).

We choose our Q matrix so that it flips the value of one circle at random.

$$Q = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix},$$

We can then work out the transition matrix of the Metropolis sampler (the Q matrix is symmetric). For example,

The transition probability from 1 to 2 is

$$p_{12} = q_{12} \times \min\left\{\frac{\pi_2}{\pi_1}, 1\right\} = \frac{1}{2} \times \min\left\{\frac{2}{1}, 1\right\} = \frac{1}{2}.$$

The transition probability from 2 to 1 is

$$p_{21} = q_{21} \times \min\left\{\frac{\pi_1}{\pi_2}, 1\right\} = \frac{1}{2} \times \min\left\{\frac{1}{2}, 1\right\} = \frac{1}{4}.$$

The transition probability from 2 to 4 is

$$p_{24} = q_{24} \times \min\left\{\frac{\pi_4}{\pi_2}, 1\right\} = \frac{1}{2} \times \min\left\{\frac{2}{2}, 1\right\} = \frac{1}{2}.$$

Continuing in this fashion, we get

$$P = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{2} \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

We can check this P matrix gives the desired stationary distribution.

$$\begin{aligned} 1 &= \frac{1}{4}\cdot 2 + \frac{1}{4}\cdot 2 = 1 \\ 2 &= \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{3}\cdot 3 = 2 \\ 2 &= \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{3}\cdot 3 = 2 \\ 3 &= \frac{1}{2}\cdot 2 + \frac{1}{2}\cdot 2 + \frac{1}{3}\cdot 3 = 3. \end{aligned}$$

The sampler is straightforward to implement.

Listing 5.1: Matlab Code

```matlab
N = 10^5; X = zeros(N,2);
X_0 = [0 0];
flips[0 1;1 0];
X(1,:) = xor(X_0,flips(ceil(2*rand),:));
for i = 1:n
    Y = xor(X(i-1),:),flips(ceil(2*rand),:));
    U = rand;
    alpha = (sum(Y)+1)/(sum(X(i-1,:))+1);
    if U < alpha
        X(i,:) = Y;
    else
        X(i,:) = X(i-1,:);
    end
end
```

5.1.4 Using the Metropolis-Hastings Sampler

An important result when using the MCMC sampler is the following.

Theorem 5.1.3 (Ergodic Theorem) *If P is irreducible, positive recurrent and $(X_n)_{n\geq 0}$ is Markov $(\boldsymbol{\mu}, P)$. Then, for any bounded function $\varphi : S \to \mathbb{R}$*

$$\mathbb{P}\left(\frac{1}{n}\sum_{k=0}^{n-1}\varphi(x_k) \longrightarrow \overline{\varphi} \text{ as } n \to \infty\right) = 1.$$

Where $\overline{\varphi} = \sum_{i\in S}\pi_i\varphi(x_i)$.

This tells us that we can estimate the expected values of functionals of random variables using MCMC methods. We still need to be careful though, as the $(X_n)_{n\geq 0}$ are not iid. This means that we might now have a central limit theorem (or any other results for calculating the error of our estimator).

As we have seen, MCMC gives us a way to sample from complicated probability distributions, even when we do not know their normalizing constants. Acceptance-rejection also doesn't need a normalizing constant. However, as we have discussed, acceptance-rejection does not work well in higher dimensions. In some sense, the Metropolis-Hastings algorithm can be thought of as a 'smarter' acceptance-rejection method.

5.1.5 Applications

Markov Chain Monte Carlo can be used in

- physics/statistical mechanics,
- drawing from complicated densities,
- drawing from conditional densities. E.g.

$$f\left(x|x \in A\right) = \frac{f(x)\mathbb{1}\left(x \in A\right)}{\int_A f(u)du}.$$

  where we don't know the normalizing constant.
- Bayesian statistics

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\int f(x|\theta)\pi(\theta)d\theta}.$$

  Usually the integral in the denominator is very hard to compute.

5.2 Markov Chains with General State Spaces

In order to extend the MCMC approach to work with possibly uncountable (general) state spaces, we need to generalize the concept of a transition matrix, $P = (p_{ij})_{i,j\in S}$.

Definition 5.2.1 (Transition Kernel) A transition kernel is a function $\mathcal{K}$ on $S \times \mathcal{B}(S)$ so that

1. $\forall\ x \in S,\ \mathcal{K}(x,\cdot)$ is a probability measure on $\mathcal{B}(S)$.
2. $\forall\ A \in \mathcal{B}(S),\ \mathcal{K}(\cdot, A)$ is measurable.

We say a sequence $(X_n)_{n\in\mathbb{N}}$ with transition kernel $\mathcal{K}$ is a Markov chain if

$$\mathbb{P}\left(X_n \in A|X_0 = x_0, \ldots, X_{n-1} = x_{n-1}\right) = \mathbb{P}\left(X_n \in A|X_{n-1} = x_{n-1}\right) = \int\limits_A \mathcal{K}(x_{n-1}, \mathrm{d}y).$$

The kernel for n transitions is given by

$$\mathcal{K}^n(x,A) = \int\limits_S K^{n-1}(y,A)\mathcal{K}(x,\mathrm{d}y).$$

Example 5.2.1 (AR(1) model) Let $X_n = \theta X_{n-1} + \varepsilon_n$, where the $(\varepsilon_n)_{n\in\mathbb{N}}$ and iid $\mathcal{N}(0,\sigma^2)$ and $\theta \in \mathbb{R}$. Here,

$$\mathcal{K}(x,A) = \frac{1}{\sqrt{2\pi\sigma^2}} \int\limits_A \exp\left(-\frac{(y-\theta x)^2}{2\sigma^2}\right) \mathrm{d}y.$$

We want to extend the concept of irreducibility to general state space models. Recall that, for countable state spaces, irreducibility means that there exists an $n > 0$ so that $\mathbb{P}(X_n = y | X_0 = x) > 0$. This no longer makes sense. Returning to the same point will often have probability 0. Instead, we use the concept of φ-irreducibility.

Definition 5.2.2 (φ-irreducibility) Given a measure φ, $(X_n)_{n\geq 0}$ is said to be **φ-irreducible** if, for every $A \in \mathcal{B}(S) > 0$ with $\varphi(A) > 0$, there exists an n so that $\mathcal{K}^n(x,A) > 0 \ \forall x \in S$.

As it turns out, the measure chosen does not really matter. 'Irreducibility' is an intrinsic property of the chain. We generalize recurrence in a similar way.

Definition 5.2.3 (Recurrence) A Markov Chain $(X_n)_{n\geq 0}$ is **recurrent** if

(i) there exists a measure ψ so that $(X_n)_{n\geq 0}$ is ψ-irreducible, and

(ii) $\forall A \in \mathcal{B}(S)$ so that $\psi(A) > 0$. $\mathbb{E}_X(\eta_A) = \infty$.
Where $\mathbb{E}_X(\eta_A)$ is the expected number of visits to A when starting in X.

Now that we have generalizations of recurrence and irreducibility, we can discuss stationary distributions.

Definition 5.2.4 (Stationarity) A σ-finite measure π is invariant for the transition kernel $\mathcal{K}(\cdot,\cdot)$ if

$$\pi(B) = \int\limits_S \mathcal{K}(x,B)\pi(\mathrm{d}x). \ \forall B \in \mathcal{B}(S).$$

Recurrence gives a unique (up to scaling) invariant measure. Finiteness comes from an additional technical condition. In a general state space, we also have the detailed balance equations (which we will use, again, to show that Metropolis-Hastings works).

Definition 5.2.5 (Detailed Balance) A Markov chain with transition kernel $\mathcal{K}$ satisfies the detailed balance equations if there exists a function f satisfying

$$\mathcal{K}(y,x)f(y) = \mathcal{K}(x,y)f(x). \ \forall (x,y) \in S \times S.$$

Lemma 5.2.1 *Suppose a Markov chain with transition kernel $\mathcal{K}$ satisfies the datailed balance conditions with π, a pdf, then*

1. *The density π is the invariant density of the chain.*
2. *The chain is reversible.*

Proof of 1

$$\begin{aligned}\int_S \mathcal{K}(y,B)\pi(y)\mathrm{d}y &= \iint\limits_{SB} \mathcal{K}(y,x)\pi(y)\mathrm{d}x\mathrm{d}y = \iint\limits_{SB} \mathcal{K}(x,y)\pi(x)\mathrm{d}x\mathrm{d}y \\ &= \iint\limits_{BS} \mathcal{K}(x,y)\pi(x)\mathrm{d}y\mathrm{d}x = \int\limits_B \pi(x)\mathrm{d}x.\end{aligned}$$

□

5.3 Metropolis-Hastings in General State Spaces

Using the machinery for general state spaces, the Metropolis-Hastings algorithm can now be used to sample from an arbitrary density f.

Algorithm 5.3.1 (Metropolis-Hastings in general state spaces) Given an objective (target) density f and a proposal density $q(y|x)$.

1. Given X_n, generate $Y_n \sim q(y|X_n)$.
2. Let
$$\alpha(x,y) = \min\left\{\frac{f(y)}{f(x)}\frac{q(x|y)}{q(y|x)}, 1\right\}.$$
Set $X_{n+1} = Y_n$ with probability $\alpha(X_n, Y_n)$ and $X_{n+1} = X_n$ otherwise.

The transition kernel of the Metropolis-Hastings sampler is given by

$$\mathcal{K}(x,y) = \alpha(x,y)q(y|x) + \mathbb{P}(\text{reject})\delta_x(y),$$

where the probability of not accepting the proposed new state is

$$\mathbb{P}(\text{reject}) = 1 - \int_S \alpha(x,y)q(y|x)\mathrm{d}y = 1 - \alpha^*(x,y).$$

Thus, $\mathcal{K}(x,y) = \alpha(x,y)q(y|x) + (1-\alpha^*(x,y))\delta_x(y)$.

Theorem 5.3.1 *The transition kernel described above is in detailed balance with f (and thus f is its stationary distribution).*

Proof We need to show that $\mathcal{K}(x,y)f(x) = \mathcal{K}(y,x)f(y)$. We break this into two parts

1. $(1-\alpha^*(x,y))\,\delta_x(y)f(x) = (1-\alpha^*(y,x))\,\delta_y(x)f(y)$.

2. $\alpha(x,y)q(y|x)f(x) = \alpha(y,x)q(x|y)f(y)$

Part 1 is easy. Both sides will be 0 except when $y = x$, in which case equality also clearly holds. For part 2, we consider cases.

- $f(y)q(x|y) > f(x)q(y|x)$:
$$\alpha(x,y)q(y|x)f(x) = q(y|x)f(x) = q(x|y)f(y)\frac{q(y|x)f(x)}{q(x|y)f(y)} = q(x|y)f(y)\alpha(y,x).$$
- $f(x)q(y|x) \geq f(y)q(x|y)$:
$$\alpha(x,y)q(y|x)f(x) = q(y|x)f(x)\frac{f(y)q(x|y)}{f(x)q(y|x)} = f(y)q(x|y) = f(y)q(x|y)\alpha(y,x).$$

$\square$

5.3.1 Types of Metropolis-Hastings Samplers

There is a lot of freedom in the choice of the proposal density $q(y|x)$ for the Metropolis-Hastings sampler. Different choices give quite different algorithms.

Independence Sampler

The idea of the independence sampler is that the choice of the proposed next state, Y, does not dependent on the current location. That is, $q(y|x) = g(y)$. This gives an acceptance probability of the form
$$\alpha(x,y) = \min\left\{\frac{f(y)g(x)}{f(x)g(y)}, 1\right\}.$$

In some ways, the independence sampler seems almost identical to the acceptance-rejection method. However, it turns out to be more efficient. Recall that the acceptance rate of the acceptance-rejection method is $1/C$.

Lemma 5.3.1 *If there exists a $C > 0$ so that $f(x) \leq Cg(x)$, then the acceptance rate of the independence sampler is at least $1/C$.*

Proof For $U \sim \mathcal{U}(0,1)$:
$$\begin{aligned}
\mathbb{P}\left(U \leq \alpha(X,Y)\right) &= \iint \min\left\{\frac{f(y)g(x)}{f(x)g(y)}, 1\right\} f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y \\
&= \iint \mathbb{1}\left\{f(y)g(x) \geq f(x)g(y)\right\} f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y + \iint \mathbb{1}\left\{f(x)g(y) \geq f(y)g(x)\right\} f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y \\
&= 2\iint \mathbb{1}\left\{f(y)g(x) \geq f(x)g(y)\right\} f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y \\
&\geq 2\iint \mathbb{1}\left\{f(y)g(x) \geq f(x)g(y)\right\} f(x)\frac{f(y)}{C}\,\mathrm{d}x\,\mathrm{d}y \\
&= \frac{2}{C}\mathbb{P}\left(f(Y)g(X) \geq f(X)g(Y)\right) = \frac{2}{C}\cdot\frac{1}{2} = \frac{1}{C}
\end{aligned}$$

□

Although the independence sampler seems superior to the acceptance-rejection method, it does have a major disadvantage, which is that the samples are not iid.

Random Walker Sampler

The idea of a random walk sampler is that it 'walks' around the support of the target pdf, with steps to high probability regions more likely. The form of the proposal density is

$$q(y|x) = g(|y - x|).$$

Where g is a symmetric distribution ($g(-x) = g(x)$). Note $q(x|y) = q(y|x)$ so $\alpha(x, y) = \min\left\{\frac{f(y)}{f(x)}, 1\right\}$. An example proposal density is the standard normal distribution with mean x, i.e.,

$$q(y|x) = \psi(y - x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y-x)^2}{2}\right)$$

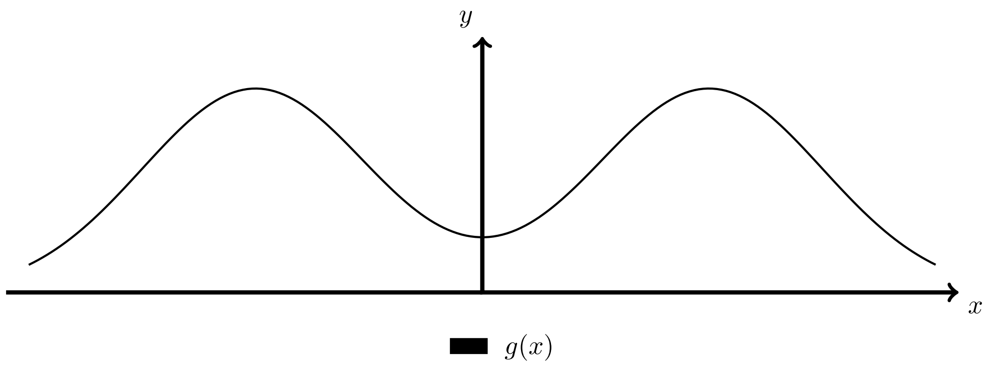
That is, we walk around the support of our target density with normally distributed step sizes (though these are not always accepted).

5.3.2 An example

We can compare the independence sampler and random walk sampler in the following example.

Example 5.3.1 (Sampling from a bimodal density) Consider the density

$$f(x) = \frac{1}{2}\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-2)^2}{2}\right) + \frac{1}{2}\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x+2)^2}{2}\right)$$



An example approach, using the independence sampler (with proposals having density $\mathcal{N}(0, 2)$), is the following.

Listing 5.2: Independence Sampler Matlab Code

```
N = 10^5; X = zeros(N,1);
x_0 = 0; X(1) = X_0;

for i = 2:N
    Y = sqrt(2)*randn; U = rand;
    f_Y = 1/2*normpdf(Y,2,1)+1/2*normpdf(Y,-2,1);
    f_X = 1/2*normpdf(X(i-1),2,1)+1/2*normpdf(X(i-1)-2,1);
    g_Y = normpdf(Y,0,sqrt(2));
    g_X = normpdf(X(i),0,sqrt(2));
    alpha = (f_Y*g_X)/(f_X*g_Y);
    if U < alpha
        X(i) = Y;
    else
        X(i) = X(i-1);
    end
end
```

An example approach, using the random walker sampler, with steps drawn from the $\mathcal{N}(0, \frac{1}{2})$ distribution, is

Listing 5.3: Matlab Code

```
N = 10^5; X = zeros(N,1);
x_0 = 0; X(1) = X_0;

for i = 2:N
    Y = X(i-1)+sqrt(1/2)*randn; U = rand;
    f_Y = 1/2*normpdf(Y,2,1)+1/2*normpdf(Y,-2,1);
    f_X = 1/2*normpdf(X(i-1),2,1)+1/2*normpdf(X(i-1)-2,1);
    alpha = f_Y/f_X;
    if U < alpha
        X(i) = Y;
    else
        X(i) = X(i-1);
    end
end
```

5.3.3 Burn-In

When using MCMC, we often do not know very much about f, the density we wish to sample from. This means that sometimes we start our Markov chain far from the stationary distribution. In this case, it can take a very long time to reach stationarity and the early values of our chain will not be "typical" of the desired distribution. One solution is to discard the early values. For example, we could throw away the first 1000 steps of our chain. The steps are then called the "Burn-In" period.

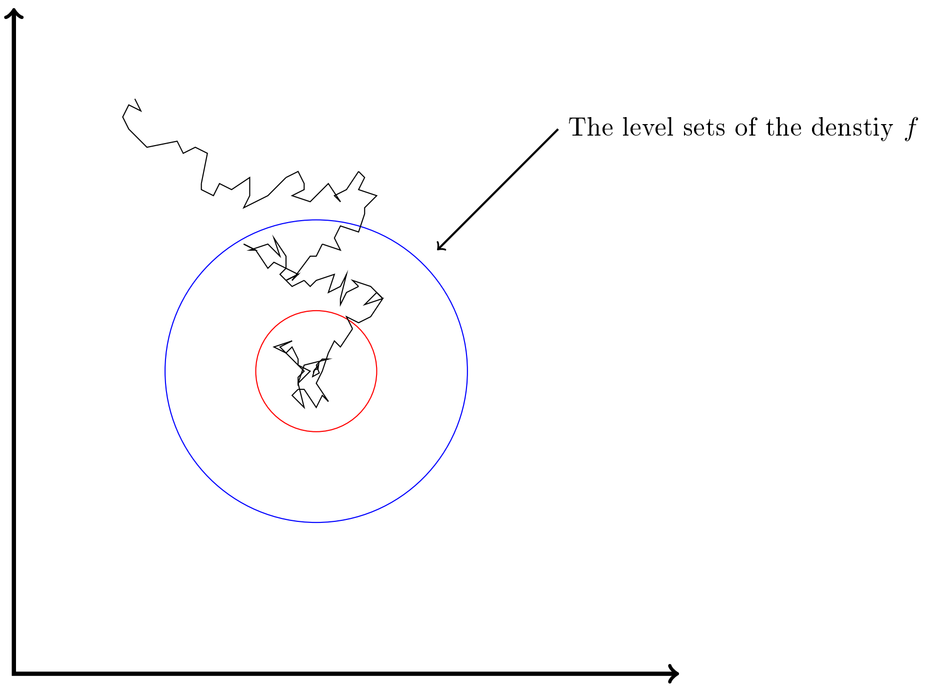


Figure 5.1: Here, the chain starts a long way from the high probability region of the density f. The first steps are not at all typical draws from the density.

5.3.4 The Ideal Acceptance Rate

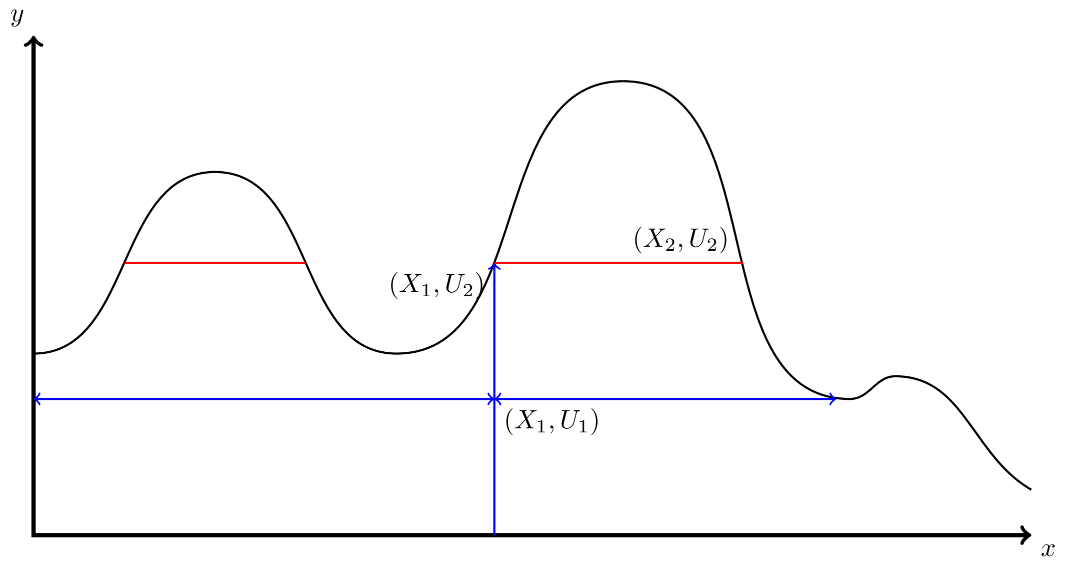
Look at the typical acceptance rate (that is, the typical value of α) is a good way to get information about the efficiency of an MCMC sampler and to 'tune' it. However, the optimal acceptance rate varies depending on the type of MCMC sampler you are using. For the independence sampler, the higher the acceptance rate, the better (as we are proposing points from anywhere in the support of f). However, for the random walk sampler, a too high acceptance rate can mean we are not exploring the state space quickly enough (after all, we will tend to accept points not too far from our current point, whereas it is often unlikely we will accept points far away).

5.4 The Slice Sampler

Another type of MCMC algorithm is the slice sampler. We wont say too much about this, other than to give the basic algorithm.

Algorithm 5.4.1 (Slice Sampler)

1. Draw $U_{n+1} \sim \mathcal{U}(0, f(X_n))$.
2. Draw $X_{n+1} \sim \mathcal{U}(A_{n+1})$, $A_{n+1} = \{x : f(x) \geq U_{n+1}\}$.



5.4.1 The Slice Sampler in Higher Dimensions

Algorithm 5.4.2 (Slice Sampler in higher dimensions)
Decompose Density in $f(\mathbf{x}) \propto \prod_{i=1}^{k} f_i(x)$.

1. $U_{n+1}^1 \sim \mathcal{U}(0, f_1(x_n))$.
2. $U_{n+1}^2 \sim \mathcal{U}(0, f_2(x_n))$.

$\vdots$

k. $U_{n+1}^k \sim \mathcal{U}(0, f_k(x_n))$.

$(k+1)$. $X_{n+1} \sim \mathcal{U}_{A_{n+1}}$, $A_{n+1} = \{x : f_i(x) \geq U_{n+1}\ i = 1, \ldots, k\}$.

Obviously, finding k can be very difficult.

5.5 The Gibbs Sampler

The other very important class of MCMC samplers are known as Gibbs samplers. These samplers can be used when we wish to sample a random vector $\mathbf{X} = (X, \ldots, X_D)$, where we know all the conditional densities of the form

$$X_i | X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_D \sim f_i\left(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d\right)$$

Algorithm 5.5.1 (Gibbs Sampler) Given $\mathbf{X}^n = (X_1^n, \ldots, X_D^n)$.

- 1: Generate $X_1^{n+1} \sim f_1\left(x_1 | X_2^n, \ldots, X_D^n\right)$.
- 2: Generate $X_2^{n+1} \sim f_2\left(x_2 \middle| X_1^{n+1}, X_3^n, \ldots, X_D^n\right)$.

$\vdots$

- D: Generate $X_D^{n+1} \sim f_D\left(x_D \middle| X_1^{n+1}, \ldots, X_{D-1}^{n+1}\right)$.

Example 5.5.1 (Drawing from a multivariate normal density)
Let $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where

$$\Sigma = \begin{pmatrix} 4 & .2 \times 2 \times 3 \\ .2 \times 2 \times 3 & 9 \end{pmatrix}. \quad \text{and } \boldsymbol{\mu} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Recall that a variance-covariance matrix has the general form

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

This joint density of the $\mathbf{X}$ is

$$f(\mathbf{x}) = \frac{1}{2\pi|\Sigma|} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right).$$

and the two conditional densities are given by

$$X_1 | X_2 = x_2 \sim \mathcal{N}\left(\mu_1 + \frac{\sigma_1}{\sigma_2}\rho(x_2 - \mu_2), \left(1-\rho^2\right)\sigma_1^2\right)$$

and

$$X_2 | X_1 = x_1 \sim \mathcal{N}\left(\mu_2 + \frac{\sigma_2}{\sigma_1}\rho(x_1 - \mu_1), \left(1-\rho^2\right)\sigma_2^2\right).$$

Implementing this in Matlab is straightforward.

Listing 5.4: Matlab Code

```
N = 10^5; X = zeros(N,2);
mu = [1 -1]; sigma = [2 3]; rho = .2;
for i = 2:N
    X1_mu = mu(1) + sigma(1)/sigma(2)*rho*(X(i-1,2)-mu(2));
    X1_sigma = sqrt(1-rho^2)*sigma(1);
    X(i,1) =X1_mu +X1_sigma * randn;
    X2_mu = mu(2) + sigma(2)/sigma(1)*rho*(X(i,1)-mu(1));
    X2_sigma = sqrt(1-rho^2)*sigma(2);
    X(i,2) = X2_mu +X2_sigma * randn;
end
```

5.5.1 Justification of the Gibbs Sampler

In order to show the Gibbs sampler works, we introduce the following notation / terminology. We define a transition from $\mathbf{x}$ to $\mathbf{y}$ (updating from 1 to n) as

$$K_{1\to n}(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{D} f(y_i|y_1,\ldots,y_{i-1},x_{i+1},\ldots x_D)$$

and a transition from $\mathbf{y}$ to $\mathbf{x}$ (updating from n to 1) as

$$K_{n\to 1}(\mathbf{x}|\mathbf{y}) = \prod_{i=1}^{D} f(x_i|y_1,\ldots,y_{i-1},x_{i+1},\ldots x_D)$$

We use the following theorem to show the Gibbs sampler works.

Theorem 5.5.1 (Hammersley-Clifford) *Let $f(x_i)$ be the ith marginal density of $f(\mathbf{x})$. Suppose $f(\mathbf{x})$ satisfies the* ***positivity condition***

$$\mathbf{y} \in \{\mathbf{x} : f(x_i) > 0,\ i = 1,\ldots,n\} \Rightarrow f(\mathbf{y}) > 0.$$

Then

$$f(\mathbf{y})K_{n\to 1}(\mathbf{x}|\mathbf{y}) = f(\mathbf{x})K_{1\to n}(\mathbf{y}|\mathbf{x}).$$

Note this is not the same as the detailed balance conditions. The Hammersley-Clifford theorem leads directly to the following.

Theorem 5.5.2 *Under the conditions given for Theorem 5.5.1 to hold, the Gibbs sampler produces a Markov chain with the desired stationary distribution, f.*

Proof

$$\int f(\mathbf{y})K_{n\to 1}(\mathbf{x}|\mathbf{y})\mathrm{d}\mathbf{x} = \int f(\mathbf{x})K_{1\to n}(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{x} \Rightarrow f(\mathbf{y}) = \int f(\mathbf{x})K_{1\to n}(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{x}.$$

This shows f is stationary with respect to the transition kernel of the Gibbs sampler. $\square$

5.5.2 Finding the Conditional Densities

The Gibbs sampler may appear a bit difficult to use in practice, as it requires knowledge of all the conditional densities. However, these conditional densities are often straightforward to determine. Observe that

$$f(x|y) = \frac{f(x,y)}{f(y)} \propto f(x,y).$$

So, we can determine the density (at least up to its constant proportionality) by looking only at the parts of $f(x,y)$ that involve x, and treating y as if it were constant.

Example 5.5.2
Consider the joint density of X and Y, which are both $\text{Exp}(\lambda)$, conditioned on $X + Y > a$.

$$f(x,y) \propto e^{-\lambda x} e^{-\lambda y} \mathbb{1}\{x + y > a\}.$$

So

$$f(x|y) \propto e^{-\lambda x} \mathbb{1}\{x > a - y\}.$$

This approach works very well for the complicated densities often found in Bayesian statistics.

Example 5.5.3
For example, consider the complicated density

$$f(\mathbf{x}, \mathbf{r}, \lambda, p) \propto \frac{b^a \lambda^{a-1} e^{-b\lambda}}{\Gamma(a)} e^{-\lambda \sum_{i=1}^n r_i} p^{\sum_{i=1}^n r_i} (1-p)^{n - \sum_{i=1}^n r_i} \lambda^{\sum_{i=1}^n x_i} \prod_{i=1}^n \frac{r_i^{x_i}}{(x_i)!}.$$

We can see that

$$f(p|\mathbf{x}, \mathbf{r}, \lambda) \propto p^{\sum_{i=1}^n r_i} (1-p)^{n - \sum_{i=1}^n r_i}$$

To find the normalizing constant, we try to find a pdf that looks like what we have above. A Beta pdf is of the form

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, \quad \text{where } B(\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$$

This looks identical when we set $\alpha = \sum\limits_{i=1}^n r_i + 1$ and $\beta = n - \sum\limits_{i=1}^n r_i + 1$. Thus, we have

$$f(p|\mathbf{x}, \mathbf{r}, \lambda) = \frac{p^{\sum_{i=1}^n r_i} (1-p)^{n - \sum_{i=1}^n r_i}}{\mathrm{B}\left(\sum_{i=1}^n r_i + 1, n - \sum_{i=1}^n r_i + 1\right)}$$

5.6 Further Reading

A very good book on Markov Chain Monte Carlo is [16]. A less theoretically orientated book is [5].

Chapter 6

Variance Reduction

The aim of the following chapter is to describe ways to develop Monte Carlo estimators that are very accurate, even when the sample size is small. The reason for doing this is pretty obvious. The more accurate a Monte Carlo estimator is, the less samples we need to get a good estimate and, thus, the less computer time we need.

It is often the case that we want to estimate a value of the form

$$\ell = \mathbb{E}\, H(\mathbf{X}),$$

where $\mathbf{X} \sim f$. Examples include $\mathbb{E}\,\mathbf{X}$, $\mathbb{E}\, f(\mathbf{X})$ and $\mathbb{E}\,(\mathbb{1}\,(\mathbf{X} \in A)) = \mathbb{P}\,(A)$. The standard Monte Carlo estimator (often called the Crude Monte Carlo (CMC) estimator) is

$$\widehat{\ell}_{\mathsf{CMC}} = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i),$$

where the $\mathbf{X}_i$ are iid draws from f. The variance of this estimator is

$$\mathrm{Var}\left(\widehat{\ell}_{\mathsf{CMC}}\right) = \mathrm{Var}\left(\frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i)\right) = \frac{1}{N^2} \cdot N \cdot \mathrm{Var}(H(\mathbf{X})) = \frac{1}{N}\mathrm{Var}(H(\mathbf{X})). \qquad (6.1)$$

We usually measure the error of a Monte Carlo estimator by its standard deviation. Here, this is

$$\frac{1}{\sqrt{N}} \sqrt{\mathrm{Var}(H(\mathbf{X}))}.$$

Unfortunately, we cannot improve upon the denominator, $\sqrt{N}$ (it would be nicer, for example, if it were N or N^2). However, we can often make an estimator with smaller variance than the Crude Monte Carlo estimator.

6.1 Antithetic Sampling

The idea of antithetic sampling (sometimes called antithetic variates) is that, instead of generating a stream of iid random variables, we generate pairs of correlated random variables $(Z_1, Z_2), (Z_3, Z_4), \ldots$.

The marginal densities of the Z_i are the same as the density of $H(\mathbf{X})$. So, $\mathbb{E}Z_1 = \mathbb{E}Z_2 = \mathbb{E}H(\mathbf{X})$, but $\mathrm{Cov}(Z_{2i}, Z_{2i-1}) \neq 0$. The antithetic variate estimator is

$$\widehat{\ell}_{\mathrm{A}} = \frac{1}{N/2}\sum_{i=1}^{N/2} \frac{Z_{2i-1} + Z_{2i}}{2}.$$

It is easy to check the estimator is unbiased, i.e., $\mathbb{E}\,\widehat{\ell}_{\mathrm{A}} = \mathbb{E}\,H(\mathbf{X})$. Now, $\mathrm{Cov}(Z_1, Z_2) = \rho\sigma_{Z1}\sigma_{Z2} = \rho\mathrm{Var}(H(X))$, so

$$\begin{aligned}
\mathrm{Var}\left(\widehat{\ell}_{\mathrm{A}}\right) &= \mathrm{Var}\left(\frac{1}{N/2}\sum_{i=1}^{N/2} \frac{Z_{2i-1} + Z_2}{2}\right) = \frac{4}{N^2}\frac{N}{2}\mathrm{Var}\left(\frac{Z_1 + Z_{2i}}{2}\right) \\
&= \frac{1}{2N}\left(\mathrm{Var}(Z_1) + \mathrm{Var}(Z_2) + 2\mathrm{Cov}(Z_1, Z_2)\right) \\
&= \frac{1}{2N}2(1+\rho)\mathrm{Var}(H(X)) \\
&= \frac{1}{N}(1+\rho)\mathrm{Var}(H(X)),
\end{aligned}$$

This will be smaller than the variance of the CMC estimator, given in (6.1), if $\rho < 0$ (that is, if Z_{2i-1} and Z_{2i} are negatively correlated).

In order to implement antithetic sampling effectively, we need to find a way to generate two variables X and Y which both have the correct marginal density but are also negatively correlated. If we use the inverse transform machine, this is quite straightforward. If $X = F^{-1}(U)$, where $U \sim \mathcal{U}(0,1)$ and F is the cdf of the desired distribution, then $Y = F^{-1}(1-U)$ will also have the marginal distribution F and will be negatively correlated with X.

Example 6.1.1 (Exponentially Distributed Random Variables)
If we wish to make $X \sim \mathrm{Exp}(1)$ and $Y \sim \mathrm{Exp}(1)$, with $\mathrm{Cov}(X,Y) < 0$, we can generate $X = -\log(U)$ and $Y = -\log(1-U)$, where $U \sim \mathcal{U}(0,1)$.

Example 6.1.2 (Bridge Network)
Consider the following bridge network.

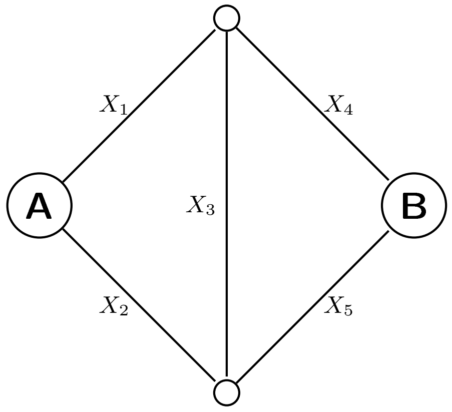


Figure 6.1: Bridge Network

where $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5)$, with the X_i iid $\mathcal{U}(0,1)$. We want to estimate the expected length of the shortest path from A to B, which we can write as $\ell = \mathbb{E}\, H(\mathbf{X})$, where

$$H(\mathbf{X}) = \min\{X_1 + X_4, X_1 + X_3 + X_5, X_2 + X_3 + X_4, X_2 + X_5\}.$$

We can estimate this using the CMC approach:

1. Generate iid vectors $\mathbf{X}^1, \ldots, \mathbf{X}^N$.
2. Calculate $\widehat{\ell}_{\mathsf{CMC}} = \frac{1}{N} \sum_{i=1}^{N} H\left(\mathbf{X}^i\right)$.

Using the antithetic variates approach, we note that H is monotone increasing in all its components, so $H(\mathbf{X})$ and $H(\mathbf{1} - \mathbf{X})$ will be negatively correlated.

1. Generate $\mathbf{X}^1, \ldots, \mathbf{X}^{N/2}$.
2. Set $Z_1 = H\left(\mathbf{X}^1\right), \ldots, Z_{N/2} = H\left(\mathbf{X}^{N/2}\right)$.
   Set $Z_1^* = H\left(\mathbf{1} - \mathbf{X}^1\right), \ldots, Z_{N/2}^* = H\left(\mathbf{1} - \mathbf{X}^{N/2}\right)$.
3. Compute $\widehat{\ell}_A = \frac{1}{N/2} \sum_{i=1}^{N/2} \frac{Z_i + Z_i^*}{2}$.

Note that, in order to estimate the variance of this estimator, we need to estimate $\mathrm{Cov}(Z_1, Z_1^*)$.

6.2 Conditional Monte Carlo

Conditional expectation can be a bit tricky (particularly if you are thinking about it rigorously). If you are unsure about anything, [9] has a good review of the essentials.

The only things we really need to know are the following.

$$\mathbb{E}\,(X|\mathbf{Y}) = G(\mathbf{Y})$$

can be though of as an (otherwise deterministic) function of the random variable $\mathbf{Y}$. A very important property of conditional expectation is the **tower property**:

$$\mathbb{E}\,(\mathbb{E}\,(X|\mathbf{Y})) = \mathbb{E}\,X$$

An important result about conditional expectation, that makes possible the variance reduction achieved by the conditional Monte Carlo method is the following.

Theorem 6.2.1 (Law of Total Variance) *Let X and Y be random variables on the same probability space.*

$$Var(X) = \mathbb{E}\,\left(Var(X|Y)\right) + Var\left(\mathbb{E}\,(X|Y)\right).$$

Proof

$$\begin{aligned}
\operatorname{Var}(X) &= \mathbb{E}\,X^2 - (\mathbb{E}\,X)^2 = \mathbb{E}\,\left(\mathbb{E}\,(X^2|Y)\right) - (\mathbb{E}\,(\mathbb{E}\,(X|Y)))^2 \\
&= \mathbb{E}\,(\mathbb{E}\,(X^2|Y)) - \mathbb{E}\,\left(\mathbb{E}\,(X|Y)^2\right) + \mathbb{E}\,\left(\mathbb{E}\,(X|Y)^2\right) - (\mathbb{E}\,(\mathbb{E}\,(X|Y)))^2 \\
&= \mathbb{E}\,\left(\mathbb{E}\,(X^2|Y)) - \mathbb{E}\,(X|Y)^2\right) + \operatorname{Var}(\mathbb{E}\,(X|Y)) \\
&= \mathbb{E}\,(\operatorname{Var}(X|Y)) + \operatorname{Var}(\mathbb{E}\,(X|Y)).
\end{aligned}$$

□

This is important because it implies that $\operatorname{Var}(\mathbb{E}\,(X|Y)) \leq \operatorname{Var}(X)$.

This leads to the following idea. Instead of estimating $\ell = H(\mathbf{X})$ directly, we can generate a related random vector $\mathbf{Y}$ and estimate $\mathbb{E}\,G(\mathbf{Y}) = \mathbb{E}\,[\mathbb{E}\,[H(\mathbf{X})|\mathbf{Y}]]$ (provided $G(\mathbf{Y})$ can be calculated exactly).

Algorithm 6.2.1 (Conditional Monte Carlo)

1. Generate $\mathbf{Y}_1, \ldots, \mathbf{Y}_N$.
2. Return the estimator

$$\widehat{\ell}_{\mathsf{Cond}} = \frac{1}{N}\sum_{i=1}^{N} G\left(\mathbf{Y}_i\right).$$

Example 6.2.1
Consider $\mathbb{P}\,(X_1 + X_2 > y)$, with $X_1 \sim \operatorname{Exp}(\lambda_1)$ and $X_2 \sim \operatorname{Exp}(\lambda_2)$. Here

$$\begin{aligned}
&l = \mathbb{P}\,(X_1 + X_2 > y) = \mathbb{E}\,\mathbb{1}\,\{X_1 + X_2 > y\}. \\
&\Rightarrow H(\mathbf{X}) = \mathbb{1}\,\{X_1 + X_2 > y\}.
\end{aligned}$$

If we condition on X_2, we get the following

$$\begin{aligned}
G(X_2) &= \mathbb{E}\,(\mathbb{1}\,\{X_1 + X_2 > y\}|X_2) \\
&= \mathbb{E}\,(\mathbb{1}\,\{X_1 > y - X_2\}|X_2) \\
&= \begin{cases} 1 & \text{if } y - X_2 < 0. \\ e^{-\lambda_1(y - X_2)} & \text{otherwise} \end{cases}
\end{aligned}$$

This gives the following algorithm:

1. Generate $X_2^1, \ldots, X_2^N \overset{iid}{\sim} \operatorname{Exp}(\lambda_2)$
2. Return

$$\frac{1}{N}\sum_{i=1}^{N}\left(e^{-\lambda_1(y - X_2^i)} \cdot \mathbb{1}\,\{y - X_2^i \geq 0\} + \mathbb{1}\,\{y - X_2^i < 0\}\right)$$

6.3 Control Variates

The idea of the control variate approach is to write $\mathbb{E}\,H(\mathbf{X})$ in the following form.

$$\begin{aligned}\mathbb{E}\,H(\mathbf{X}) &= \mathbb{E}\,(H(\mathbf{X}) - \alpha S(\mathbf{X}) + \alpha S(\mathbf{X})) = \mathbb{E}\,(H(\mathbf{X}) - \alpha S(\mathbf{X})) + \alpha \mathbb{E}\,(S(\mathbf{X}))\\ &= \mathbb{E}\,(H(\mathbf{X}) - \alpha S(\mathbf{X})) + \alpha s,\end{aligned}$$

where $s = \mathbb{E}\,S(\mathbf{X})$ is known in closed form. Hopefully, S and α can be chosen so that

$$\mathrm{Var}(H(\mathbf{X}) - \alpha S(\mathbf{X})) < \mathrm{Var}(H(\mathbf{X}))$$

This gives the following algorithm.

Algorithm 6.3.1 (Control Variates)

1. Draw $\mathbf{X}^1, \ldots, \mathbf{X}^N$.
2. Return $\frac{1}{N}\sum_{i=1}^{N}\left(H(\mathbf{X}^i) - \alpha S(\mathbf{X}^i)\right) + \alpha s$.

In order for this to work, we need to choose the α that minimizes the variance of

$$\widehat{\ell}_{\mathsf{Cont}} = \mathrm{Var}\,(H(\mathbf{X}) - \alpha S(\mathbf{X}) + \alpha s) = \mathrm{Var}\,(H(\mathbf{X})) + \alpha^2 \mathrm{Var}\,(S(\mathbf{X})) - 2\alpha \mathrm{Cov}\,(H(\mathbf{X}), S(\mathbf{X}))$$

Taking the derivative with respect to α and setting to 0, we get

$$2\alpha \mathrm{Var}\,(S(\mathbf{X})) = 2\mathrm{Cov}\,(H(\mathbf{X}), S(\mathbf{X})) \Rightarrow \alpha^* = \frac{\mathrm{Cov}(H(\mathbf{X}), S(\mathbf{X}))}{\mathrm{Var}\,(S(\mathbf{X}))}.$$

Checking the second order condition, we have

$$2\mathrm{Var}(S(\mathbf{X})) > 0,$$

so α^* is a minimizer. Although we generally do not know α^* exactly, we can easily estimate it using standard estimators for the variance and covariance (see the example code below).

Using α^*, the optimal choice of α, the estimator's variance is given by

$$\begin{aligned}&\mathrm{Var}(H(\mathbf{X})) + (\alpha^*)^2 \mathrm{Var}(S(\mathbf{X})) - 2\alpha^* \mathrm{Cov}(H(\mathbf{X}), S(\mathbf{X}))\\ =&\mathrm{Var}(H(\mathbf{X})) + \frac{\mathrm{Cov}(H(\mathbf{X}), S(\mathbf{X}))^2}{\mathrm{Var}(S(\mathbf{X}))^2}\mathrm{Var}(S(\mathbf{X})) - 2\frac{\mathrm{Cov}(H(\mathbf{X}), S(\mathbf{X}))}{\mathrm{Var}(S(\mathbf{X}))}\mathrm{Cov}(H(\mathbf{X}), S(\mathbf{X}))\\ =&\mathrm{Var}(H(\mathbf{X})) - \frac{\mathrm{Cov}(H(\mathbf{X}), S(\mathbf{X}))^2}{\mathrm{Var}(S(\mathbf{X}))} = \mathrm{Var}(H(\mathbf{X})) - \rho^2\frac{\mathrm{Var}(H(\mathbf{X}))\mathrm{Var}(S(\mathbf{X}))}{\mathrm{Var}(S(\mathbf{X}))}\\ =&(1-\rho^2)\mathrm{Var}(H(\mathbf{X})),\end{aligned}$$

which will give variance reduction so long as ρ, the correlation between $S(\mathbf{X})$ and $H(\mathbf{X})$, is greater than 0.

Example 6.3.1
We wish to estimate

$$\ell = \int_0^1 e^{-x^2} \mathrm{d}x \approx 0.7468.$$

We use the control variate $S(X) = e^{-X}$, where $X \sim \mathcal{U}(0,1)$.

$$s = \mathbb{E}\, S(X) = \int_0^1 e^{-x} \mathrm{d}x = -e^{-x}\Big|_0^1 = 1 - \frac{1}{e}.$$

Listing 6.1: Matlab Code

```
N = 10^4; X = rand(N,1);
H_X = exp(-X.^2); S_X = exp(-X);
cov_mat = cov(H_X,S_X);
alpha_hat = cov_mat(1,2)/cov_mat(2,2);

ell_hat_CMC = sum(H_X)/N
std_CMC = std(H_X)/sqrt(N)

ell_hat_CONT = sum(H_X-alpha_hat*S_X)/N+alpha_hat*(1-1/exp(1))
std_CONT = std(H_X-alpha_hat*S_X)/sqrt(N)
```

Listing 6.2: Example Output

```
ell_hat_CMC =
    0.7466
std_CMC =
    0.0020
ell_hat_CONT =
    0.7475
std_CONT =
  5.4539e-004
```

The standard deviation of the control variates estimator is roughly 1/4 of the standard deviation of the CMC estimator. In order to get an equivalent standard deviation using CMC, we would need about 16 times the current sample size.

6.4 Importance Sampling

We want to estimate $\ell = \mathbb{E}_f H(\mathbf{X})$, where $\mathbb{E}_f$ means the expectation when $\mathbf{X} \sim f$, where f is some density. We can write out the expectation of this estimator as

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

provided we choose g so that, $g(\mathbf{x}) = 0 \Rightarrow H(\mathbf{x}) f(\mathbf{x}) = 0$.

The motivation for doing this is that, often, some values of $\mathbf{X}$ are more important for calculating $\mathbb{E}\, H(\mathbf{X})$ than others. The idea is to choose a density, g, so that these important values occur more often.

Example 6.4.1
Suppose we wish to estimate $\mathbb{P}\,(X > 5)$, where $X \sim \mathcal{N}(0,1)$. The problem is then to estimate $\mathbb{E}\, H(X) = \mathbb{E}\,\mathbb{I}\{X > 5\}$. In general it is rare that a realization of X will be bigger than 5, so this quantity will be very difficult to estimate. However, if we choose g to be the pdf of a normal density with mean 5, then values of X greater than 5 will occur quite often. The importance sampling estimator is unbiased, so this estimator will give an unbiased estimate of $\mathbb{P}\,(X > 5)$ which (we can show) has a much lower variance than the CMC version.

Algorithm 6.4.1 (Importance Sampling)

1. Draw $X_1, \ldots, X_N \sim g$.
2. Return $\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N}\sum_{i=1}^N \frac{f(X_i)}{g(X_i)} H(X_i)$.

Recall the the variance of the CMC estimator is

$$\mathrm{Var}\left(\widehat{\ell}_{\mathsf{CMC}}\right) = \frac{1}{N}\mathrm{Var}_f\left(H(\mathbf{X})\right) = \frac{1}{N}\left[\mathbb{E}_f H(\mathbf{X})^2 - \left(\mathbb{E}_f H(\mathbf{X})\right)^2\right].$$

The variance of the importance sampling estimator is

$$\begin{aligned}\mathrm{Var}\left(\widehat{\ell}_{\mathsf{IS}}\right) &= \frac{1}{N}\mathrm{Var}_g\left(\frac{H(\mathbf{X})f(\mathbf{X})}{g(\mathbf{X})}\right)\\ &= \frac{1}{N}\mathbb{E}_g\left(\frac{H(\mathbf{X})f(\mathbf{X})}{g(\mathbf{X})}\right)^2 - \left(\mathbb{E}_g \frac{H(\mathbf{X})f(\mathbf{X})}{g(\mathbf{X})}\right)^2\\ &= \frac{1}{N}\left[\mathbb{E}_g\left(\frac{H(\mathbf{X})f(\mathbf{X})}{g(\mathbf{X})}\right)^2 - \left(\mathbb{E}_f H(\mathbf{X})\right)^2\right].\end{aligned}$$

Now,

$$\mathbb{E}_g\left(\frac{H(\mathbf{X})f(\mathbf{X})}{g(\mathbf{X})}\right)^2 = \int \frac{H(\mathbf{x})^2 f(\mathbf{x})^2}{g(\mathbf{x})^2} g(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int \frac{H(\mathbf{x})^2 f(\mathbf{x})}{g(\mathbf{x})} f(\mathbf{x})\,\mathrm{d}\mathbf{x} = \mathbb{E}_f\left(\frac{H(\mathbf{X})^2 f(\mathbf{X})}{g(\mathbf{X})}\right).$$

So, we get variance reduction if

$$\mathbb{E}_f\left(\frac{H(\mathbf{X})^2 f(\mathbf{X})}{g(\mathbf{X})}\right) \leq \mathbb{E}_f H(\mathbf{X}).$$

6.4.1 The Minimum Variance Density

Theorem 6.4.1 *The variance of the importance sampling estimator is minimized by the density g^* given by*

$$g^* = \frac{|H(\mathbf{x})|}{\mathbb{E}_f |H(\mathbf{X})|} f(\mathbf{x}).$$

Proof We need to show

$$\mathbb{E}_{g^*}\left(\frac{f(\mathbf{X})}{g^*(\mathbf{X})}H(\mathbf{X})\right)^2 \leq \mathbb{E}_{g}\left(\frac{f(\mathbf{X})}{g(\mathbf{X})}H(\mathbf{X})\right)^2.$$

for any g such that $H(\mathbf{x})f(\mathbf{x}) = 0 \Rightarrow g(\mathbf{x}) = 0$. Now

$$\begin{aligned}\mathbb{E}_{g^*}\left(\frac{f(\mathbf{X})}{g^*(\mathbf{X})}H(\mathbf{X})\right)^2 &= \mathbb{E}_{f}\frac{H(\mathbf{X})^2 f(\mathbf{X})}{g^*(\mathbf{X})} = \mathbb{E}_{f}\left(\frac{H(\mathbf{X})^2 f(\mathbf{X})}{|H(\mathbf{X})|f(\mathbf{X})}\right)E_f|H(\mathbf{X})| \\ &= (\mathbb{E}_f|H(\mathbf{X})|)^2 = \left(\mathbb{E}_g\frac{f(\mathbf{X})}{g(\mathbf{X})}|H(\mathbf{X})|\right)^2 \\ &\leq \mathbb{E}_g\left(\frac{f(\mathbf{X})}{g(\mathbf{X})}|H(\mathbf{X})|\right)^2 = \mathbb{E}_g\left(\frac{f(\mathbf{X})}{g(\mathbf{X})}H(\mathbf{X})\right)^2\end{aligned}$$

where the inequality is obtained, for example, using Jensen's inequality. □

If $H(\mathbf{x}) > 0\ \forall\mathbf{x}$ or $H(\mathbf{x}) < 0\ \forall\mathbf{x}$, we get an even stronger result, which is that the minimum variance density actually results in an estimator with zero variance (that is, an estimator that gives the exact answer every time!). Here, we have

$$g^* = \frac{H(\mathbf{x})}{\mathbb{E}\,H(\mathbf{x})}f(\mathbf{x}),$$

so

$$\operatorname{Var}\left(\widehat{\ell}_{\mathsf{IS}}\right) = \frac{1}{N}\operatorname{Var}_{g^*}\left(\frac{f(\mathbf{X})}{g^*(\mathbf{X})}H(\mathbf{X})\right) = \frac{1}{N}\operatorname{Var}_{g^*}\left(\frac{H(\mathbf{X})f(\mathbf{X})}{H(\mathbf{X})f(\mathbf{X})}\mathbb{E}\,H(\mathbf{X})\right) = \operatorname{Var}_{g^*}(\mathbb{E}\,H(\mathbf{X})) = 0.$$

Unfortunately we cannot really use the zero-variance / minimum variance densities because $\mathbb{E}\,H(X)$ and $\mathbb{E}\,|H(X)|$ are unknown (they involve knowledge of the very thing we are trying to estimate). However, the fact that it is often theoretically possible to find an estimator with zero-variance illustrates the power of importance sampling as a variance reduction technique.

In practice, we usually choose g from a parametric family of densities, $\{g(\mathbf{x};\boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}$, indexed by a parameter vector $\boldsymbol{\theta}$.

A standard way to come up with such a family of densities is to **exponentially twist** f. That is, we set

$$g(\mathbf{x};\boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta}^\intercal\mathbf{x} - k(\boldsymbol{\theta})\right)f(\mathbf{x}),$$

where $k(\boldsymbol{\theta}) = \log\mathbb{E}\,e^{\boldsymbol{\theta}^\intercal x}$. In order for this to work, we need $\boldsymbol{\theta} \in \Theta$, where $\Theta = \{\boldsymbol{\theta} : \mathbb{E}\,\exp\left(\boldsymbol{\theta}^\intercal\mathbf{x}\right) < \infty\}$. Note that $g(\mathbf{x};0) = f(\mathbf{x})$.

We then try to choose a good parameter vectors, $\boldsymbol{\theta}$. An obvious approach is called the **variance minimization** approach. We set

$$\boldsymbol{\theta}_{\mathsf{VM}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\ \mathbb{E}_f\frac{f(\mathbf{X})}{g(\mathbf{X};\boldsymbol{\theta})}H(\mathbf{X})^2.$$

Unfortunately, it is usually very difficult to solve this equation. An alternative method is the **Cross-Entropy Method**, where we choose $\boldsymbol{\theta}$ to minimize the 'distance' between $g(\mathbf{x};\boldsymbol{\theta})$ and $g^*(\mathbf{x})$, where as a measure of distance we use Kullback-Leibler divergence, which is defined as

$$\mathcal{D}(g,f) = \mathbb{E}_g \log \frac{g(\mathbf{x})}{f(\mathbf{x})}.$$

Note: This is not a metric as it is not symmetric.

Now we choose $\boldsymbol{\theta}$ to minimize $\mathcal{D}(g^*,g)$. That is,

$$\begin{aligned}\boldsymbol{\theta}_{\mathsf{CE}} &= \operatorname*{argmin}_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{g^*} \log \frac{g^*(\mathbf{X})}{g(\mathbf{X};\boldsymbol{\theta})} = \operatorname*{argmin}_{\boldsymbol{\theta}\in\Theta} \{\mathbb{E}_{g^*} g^*(\mathbf{X}) - \mathbb{E}_{g^*} \log g(\mathbf{X};\boldsymbol{\theta})\} \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{g^*} \log g(\mathbf{X};\boldsymbol{\theta})\end{aligned}$$

Chapter 7

Derivative Estimation

It is often the case that we wish to calculate the gradient of a function with respect to a vector of parameters $\boldsymbol{\theta}$. This is particularly difficult when the function itself needs to be estimated. We can write such functions, in a very general form, as

$$\ell(\boldsymbol{\theta}) = \mathbb{E}_{g(\boldsymbol{\theta}_2)} H(\mathbf{X}; \boldsymbol{\theta}_1) = \int H(\mathbf{X}; \boldsymbol{\theta}_1) g(\mathbf{x}; \boldsymbol{\theta}_2)\, \mathrm{d}\mathbf{x},$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$.

In derivative estimation, we wish to estimate

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{g(\boldsymbol{\theta}_2)} H(\mathbf{X}; \boldsymbol{\theta}_1).$$

There are a number of reasons why we might want to do this. Two of the most important are:

1. **Sensitivity Analysis** E.g., How does the value of $\ell(\boldsymbol{\theta})$ change when $\boldsymbol{\theta}$ changes?
2. **Optimization** E.g., for solving $\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = 0$.

7.1 Difference Estimators

Hopefully, most of you are familiar with the definition of a derivative:

$$f'(\theta) = \lim_{h \to 0} \frac{f(\theta + h) - f(\theta)}{h} = \lim_{h \to 0} \frac{f(\theta + h/2) - f(\theta - h/2)}{h}.$$

provided these limits exist at θ. There is an obvious extension to partial derivatives.

This definition leads directly to the first two derivative estimators. Let $\mathbf{e}_i$ be a vector that is composed entirely of 0s except for the ith element, which is 1. E.g., $e_1 = (1, 0, \ldots, 0)^\intercal$, $e_2 = (0, 1, 0, \ldots, 0)^\intercal$, etc.

- The forward difference estimator of the ith partial derivative:

$$\widehat{\mathrm{FD}}_i = \frac{\widehat{\ell}(\boldsymbol{\theta} + h\mathbf{e}_i) - \widehat{\ell}(\boldsymbol{\theta})}{h}.$$

- The central difference estimator of the ith partial derivative:

$$\widehat{\mathrm{CD}}_i = \frac{\widehat{\ell}(\boldsymbol{\theta} + \frac{h}{2}\mathbf{e}_i) - \widehat{\ell}(\boldsymbol{\theta} - \frac{h}{2}\mathbf{e}_i)}{h}.$$

One important question about these estimators is whether they are biased or not. We can check this in the one dimensional case. Taking Taylor expansions, the forward difference estimator can be written as

$$\frac{f(\theta+h) - f(\theta)}{h} = f'(\theta) + \underbrace{\frac{h}{2}f''(\theta) + O(h^2)}_{\text{bias}}$$

and the central difference estimator as

$$\frac{f(\theta+h/2) - f(\theta-h/2)}{h} = f'(\theta) + \underbrace{\frac{h^2}{24}f'''(\theta) + O(h^4)}_{\text{bias}}.$$

From this, we see that both estimators are biased. However, the bias of the central difference estimator is smaller than the bias of the forward difference estimator. From now on, we will just discuss the central difference estimator.

7.1.1 The Variance-Bias Tradeoff

A biased estimator is not necessarily a bad one. One way to evaluate how good a biased estimator is, is to consider its mean squared error (MSE).

$$\mathrm{MSE}(\widehat{\ell}) = \mathbb{E}\,(\widehat{\ell} - \ell)^2 = \mathrm{Var}(\widehat{\ell}) + \mathrm{Bias}(\widehat{\ell})^2.$$

Arguably, the estimator with the lowest MSE is the best.

A classic issue with biased estimators is the variance-bias tradeoff. Often a more biased estimator will have a lower variance, and a less biased estimator will have a higher variance. This is an issue for us.

The bias of the central difference estimator gets smaller as $h \to 0$. However,

$$\mathrm{Var}\left(\frac{\widehat{\ell}(\boldsymbol{\theta} + h/2\mathbf{e}) - \widehat{\ell}(\boldsymbol{\theta} - h/2\mathbf{e})}{h}\right) = \frac{1}{h^2}\mathrm{Var}\left(\widehat{\ell}(\boldsymbol{\theta} + h/2\mathbf{e}) - \widehat{\ell}(\boldsymbol{\theta} - h/2\mathbf{e})\right)$$

can get bigger as $h \to 0$.

The following result is an example of a rule for choosing the relationship between h and N to minimize the MSE of the central difference estimator.

Proposition 7.1.1 *Consider the simple case where*

$$l(\theta) = \mathbb{E}\, H(\mathbf{X}, \theta) = \underbrace{\mathbb{E}\, H(\theta)}_{suppressing\ \mathbf{X}} .$$

with θ a scalar. Let $\widehat{\ell}(\theta + h)$ and $\widehat{\ell}(\theta)$ be estimators using N samples each. Then the mean squared error of $\widehat{CD}$ is asymptotically minimized by setting.

$$h = \frac{1}{N^{1/6}} \frac{(576\, Var(H(\theta)))^{1/6}}{|l'''(\theta)|^{1/3}}.$$

7.2 Interchanging Differentiation and Integration

The next two methods of estimating derivatives work by interchanging differentiation and integration. This cannot always be done. However, in many situations, there is a technical result that justifies this operation. An example is the following.

Theorem 7.2.1 *Let $g(\mathbf{x}, \boldsymbol{\theta})$ be differentiable at $\boldsymbol{\theta}_0 \in \mathbb{R}^k$ with gradient $\nabla_{\boldsymbol{\theta}} g(\mathbf{x}, \boldsymbol{\theta}_0)$. Assume this gradient is integrable as function of $\mathbf{x}$. If there exists a neighborhood Θ of $\boldsymbol{\theta}_0$ and an integrable function $M(\mathbf{x}; \boldsymbol{\theta}_0)$ such that for all $\boldsymbol{\theta} \in \Theta$*

$$\frac{|g(\mathbf{x}; \boldsymbol{\theta}) - g(\mathbf{x}; \boldsymbol{\theta}_0)|}{\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|} \leq M(\mathbf{x}; \boldsymbol{\theta}_0).$$

Then

$$\nabla_{\boldsymbol{\theta}} \int g(\mathbf{x}; \boldsymbol{\theta}) \mathrm{d}\mathbf{x} \bigg|_{\boldsymbol{\theta} = \boldsymbol{\theta}_0} = \int \nabla_{\boldsymbol{\theta}} g(\mathbf{x}; \boldsymbol{\theta}_0) \mathrm{d}\mathbf{x}.$$

7.3 Infinitesimal Perturbation Analysis (IPA)

Consider the case where $\ell(\boldsymbol{\theta}) = \mathbb{E}_{g(\boldsymbol{\theta}_2)} H(\mathbf{X}; \boldsymbol{\theta}_1)$ can be written as $\mathbb{E}_f G(\mathbf{X}; \boldsymbol{\theta})$. That is, the parameters of interest only appear in the function G and not in the density f. If we can exchange expectation and differentiation, then

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_f G(\mathbf{X}; \boldsymbol{\theta}) = \mathbb{E}\, \nabla_{\boldsymbol{\theta}} G(\mathbf{X}; \boldsymbol{\theta}).$$

We can estimate this using the obvious estimator

$$\widehat{\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})} = \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} G(\mathbf{X}^i; \boldsymbol{\theta}),$$

where $\mathbf{X}^1, \ldots, \mathbf{X}^N$ are iid draws from f.

A significant advantage of this method over the finite difference estimators is that the estimator is unbiased.

Sometimes $\mathbb{E}_{g(\boldsymbol{\theta}_2)}H(\mathbf{X};\boldsymbol{\theta}_1)$ is is not already in the form $\mathbb{E}_f G(\mathbf{X};\boldsymbol{\theta})$. In this case, we need to find a way to rewrite $\mathbb{E}_{g(\boldsymbol{\theta}_2)}H(\mathbf{X};\boldsymbol{\theta}_1)$ in the appropriate form. A situation where this is easy to do is when we generate one dimensional draws from a density $g(x;\boldsymbol{\theta}_2)$ using the inverse transform method. In this case, $X = F^{-1}_{g(\boldsymbol{\theta}_2)}(U)$, where $F_{g(\boldsymbol{\theta}_2)}$ is the cdf of $g(x;\boldsymbol{\theta}_2)$ and $U \sim \mathcal{U}(0,1)$. Thus, we can write

$$\mathbb{E}_{g(\boldsymbol{\theta}_2)}H(\mathbf{X};\boldsymbol{\theta}_1) = \mathbb{E}_f\left[H\left(F^{-1}_{g(\boldsymbol{\theta}_2)}(U);\boldsymbol{\theta}_1\right)\right].$$

where f is the density of the uniform $(0,1)$ density. It is not hard to extend this approach to random vectors.

7.4 Score Function Method

Using the IPA approach, there are many situations where the exchange of expectation and differentiation is not allowed. An alternative approach, where this exchange is less problematic, is the score function method. In contrast to IPA, we now want to write $\ell(\boldsymbol{\theta}) = \mathbb{E}_{g(\boldsymbol{\theta}_2)}H(\mathbf{X};\boldsymbol{\theta}_1)$ as $\mathbb{E}_{f(\boldsymbol{\theta})}G(\mathbf{X})$, so that the parameters only appear in the density.

Exchanging integration and differentiation, we have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}}\int G(\mathbf{x})f(\mathbf{x};\boldsymbol{\theta})\mathrm{d}\mathbf{x} = \int G(\mathbf{x})\nabla_{\boldsymbol{\theta}}f(\mathbf{x};\boldsymbol{\theta})\mathrm{d}x \\ &= \int G(\mathbf{x})\frac{\nabla_{\boldsymbol{\theta}}f(\mathbf{x};\boldsymbol{\theta})}{f(\mathbf{x};\boldsymbol{\theta})}f(\mathbf{x};\boldsymbol{\theta})\mathrm{d}\mathbf{x} = \mathbb{E}_{f(\boldsymbol{\theta})}G(\mathbf{X})S(\boldsymbol{\theta};\mathbf{X}),\end{aligned}$$

where

$$S(\boldsymbol{\theta};\mathbf{x}) = \nabla_{\boldsymbol{\theta}}\log f(\mathbf{x};\boldsymbol{\theta}).$$

In statistics, $S(\boldsymbol{\theta};\mathbf{x})$ is sometimes known as the score function of f.

Like the IPA approach, this gives an unbiased estimator.

Example 7.4.1
Let $X \sim \mathcal{N}(\theta,1)$. We want to calculate $\frac{\partial}{\partial\theta}\mathbb{E}X^2$ at θ_0. In this simple example, we can calculate the right answer as $\mathbb{E}X^2 = 1+\theta^2$ so $\frac{\partial}{\partial\theta}\mathbb{E}X^2 = 2\theta$. In order to implement the score function method, we first calculate $S(\theta;x)$:

$$\begin{aligned}S(\theta;x) &= \frac{\partial}{\partial\theta}\log f(x;\theta) = \frac{\partial}{\partial\theta}\left(\log\left(\frac{1}{\sqrt{2\pi}}\right) - \frac{1}{2}(x-\theta)^2\right) \\ &= \frac{\partial}{\partial\theta}\left(-\frac{x^2}{2} + \frac{2x\theta}{2} - \frac{\theta^2}{2}\right) = x-\theta\end{aligned}$$

So, we wish to estimate

$$\mathbb{E}_{f(\theta)}X^2(X-\theta)$$

For $\theta_0 = 1$.

Listing 7.1: Matlab Code

```
N = 10^5; theta = 1;
X = theta + randn(N,1);
H_X = X.^2;
S_X = X-theta;
deriv_est = mean(S_X.*H_X)
error_est = std(S_X.*H_X) / sqrt(N)
```

Chapter 8

Optimization

Monte Carlo is a powerful tool for solving optimization problems. In this chapter, we consider problems of the general form

$$\min_{\boldsymbol{\theta} \in \Theta} S(\boldsymbol{\theta}),$$

where Θ is a feasible set.

Example 8.0.1 (Some Example Optimization Problems)

1. $\min \theta^2$, $\Theta = \{\theta \in \mathbb{R}\}$.
2. $\min \theta^2$, $\Theta = \{\theta \in \mathbb{R} : |\theta| \geq 2\}$.
3. $\max \mathbb{E}\, X^\theta = \min -\mathbb{E}\, X^\theta$, $\Theta = \{\theta \in \mathbb{R} : 1 \leq \theta \leq 4\}$.
4. $\max\{\theta_1 + \theta_2\} = \min\{-(\theta_1 + \theta_2)\}$,
   $\Theta = \{(\theta_1, \theta_2) : \theta_1, \theta_2 \in \mathbb{Z}, \theta_1, \theta_2 \geq 0, \theta_1 + 2\theta_2 \leq 6, \theta_2 + 2\theta_1 = 6\}$.

Optimization problems can be **noisy**, e.g., $S(\theta)$ or $\nabla_\theta S(\theta)$ may need to be estimated.

Optimization problems can involve **discrete** variables, **continuous** variables, or a **mixture** of both.

Optimization problems can involve **constraints** or be **unconstrained**.

8.1 Stochastic Approximation

In the stochastic optimization setting, $S(\boldsymbol{\theta})$ is noisy. That is, $S(\boldsymbol{\theta}) = \mathbb{E}_{g(\boldsymbol{\theta}_2)} H(\mathbf{X}; \boldsymbol{\theta}_1)$. In addition, $\boldsymbol{\theta}$ is assumed to be a vector of continuous variables. For example, $\boldsymbol{\theta} \in \mathbb{R}^d$.

Because $S(\boldsymbol{\theta})$ is noisy, it can be only estimated. We cannot just solve $\nabla_{\boldsymbol{\theta}} S(\boldsymbol{\theta}) = \mathbf{0}$.

The idea of the stochastic approximation algorithm is to set up a sequence $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots$ that converges to the optimal $\boldsymbol{\theta}^*$. The algorithm does this using the idea of gradient descent. That is, at each $\boldsymbol{\theta}_n$, it estimates $\nabla_{\boldsymbol{\theta}} S(\boldsymbol{\theta})$ (evaluated at $\boldsymbol{\theta}_n$) and moves in the direction of the negative gradient (which is the direction of steepest descent).

8.1.1 The Unconstrained Case

In the unconstrained case, the sequence is of the form

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \varepsilon_n \widehat{\nabla_{\boldsymbol{\theta}} S(\boldsymbol{\theta}_n)},$$

where $\widehat{\nabla_{\boldsymbol{\theta}} S(\boldsymbol{\theta}_n)}$ is a gradient estimator and $(\varepsilon_n)_{n \geq 1}$ is a sequence of step sizes.

Example 8.1.1 (A simple example)

Looking at a non-noisy problem, we can get an idea of how the algorithm behaves.

Let $S(\theta) = \theta^2$. We can see (because the problem is not noisy) that $\nabla_\theta S(\theta) = 2\theta$. As step sizes, we choose $\varepsilon_n = \frac{1}{n}$. We state at $\theta_1 = 2$.

$$\begin{aligned}
\theta_2 &= \theta_1 - \varepsilon_1 \nabla S(\theta_1) = 2 - 2(2) = -2 \\
\theta_3 &= \theta_2 - \varepsilon_2 \nabla S(\theta_2) = -2 - \frac{1}{2}(-4) = -2 + 2 = 0 \\
\theta_4 &= \theta_3 - \varepsilon_3 \nabla S(\theta_3) = 0 - \frac{1}{3}(0) = 0. \ldots
\end{aligned}$$

The algorithm jumps around for a few steps, but ultimately converges to the minimum (and stays there).

8.1.2 The Constrained Case

The extension to the constrained case is straightforward. We simply take the sequence

$$\boldsymbol{\theta}_{n+1} = \Pi_\Theta \left(\boldsymbol{\theta}_n - \varepsilon_n \widehat{\nabla S(\boldsymbol{\theta}_n)} \right),$$

where Π_Θ is a projection operator onto Θ. This might sound complicated, but usually it is not. $\Pi_\Theta(\cdot)$ just returns the closest point in Θ. Usually, we use the Euclidean norm, so

$$\Pi_\Theta(\boldsymbol{\theta}) = \underset{\boldsymbol{\xi} \in \Theta}{\operatorname{argmin}} \|\boldsymbol{\xi} - \boldsymbol{\theta}\|.$$

8.1.3 Choice of Parameters

The stochastic approximation algorithm has a lot of restrictions. It only really works for convex functions on convex sets. Otherwise, the algorithm tends to get trapped in local minima. In addition, the parameters of the algorithm have to be chosen carefully to ensure that it converges to the global minimum.

We need to be able to converge to $\boldsymbol{\theta}^*$, no matter how far away we start (which means we might need to take an infinite number of non-zero steps), so we require

$$\sum_{n=0}^{\infty} \varepsilon_n = \infty.$$

However, we also want $\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta} \to 0$, so we require that $\varepsilon_n \to 0$. Typical choices are $\varepsilon_n = n^{-\alpha}$, with $0 < \alpha \leq 1$, or $\varepsilon = c/n$ for some some constant c.

8.1.4 The Two Main Types of Stochastic Approximation Algorithms

There are two main approaches to stochastic approximation. These differ only in their choice of gradient estimator (though, this has a number of consequences for the mathematical analysis of these algorithms). The **Kiefer-Wolfowitz** algorithm uses finite differences to estimate $\widehat{\nabla_{\boldsymbol{\theta}} S(\boldsymbol{\theta})}$. The **Robbins-Monro** algorithm does not use finite differences. Instead, it usually uses IPA or the score function method.

For convergence results (which tend to have a lot of technical conditions) see Dirk Kroese's lecture notes.

8.2 Randomized Optimization

Two major disadvantages of stochastic approximation algorithms are

1. They need gradients to exist (so, for example, discrete problems cannot be solved).
2. They tend to get trapped in local minima.

An alternative is a large number of algorithms based on randomly (but intelligently) searching for solutions. These algorithms include:

- Simulated Annealing
- Cross Entropy
- Genetic algorithms

8.2.1 Simulated Annealing

We will focus on the simulated annealing algorithm, which is one of the most popular randomized optimization algorithms. The idea behind simulated annealing comes from materials science, where a metal is heated then slowly cooled in such a way that its molecules jump around until they find better positions. In randomized optimization, our guess of the optimal solution is a stochastic process that jumps around the state space. It jumps more when it is 'hotter' and jumps less as it cools.

NOTE!!! I've change notation in this section (to make some things clearer). Now, we write the problem as

$$\min_{\mathbf{x}\in\Theta} S(\mathbf{x}).$$

The basic idea is to choose random guesses $\mathbf{X}^1, \mathbf{X}^2, \ldots,$ of the optimal value of $\mathbf{x}$ from a sequence of distributions that converge to a 'degenerate' distribution that assigns probability 1 to the global minimum, $\mathbf{x}^*$, (for ease of explanation, let's assume it is unique).

To do this, we use an approach similar to MCMC. We create a random process that slowly converges to the right distribution, which is the one that assigns all of its probability mass to the global minimum. The distributions used by simulated annealing are the following.

- Discrete:
$$\mathbb{P}\left(\mathbf{X}(T)=\mathbf{x}\right)=\frac{e^{-S(\mathbf{x})/T}}{\sum_{\mathbf{y}\in\Theta}e^{-S(\mathbf{y})/T}}\propto e^{-S(\mathbf{x})/T}. \tag{8.1}$$

- Continuous:
$$f_T(x)=\frac{e^{-S(\mathbf{x})/T}}{\int_\Theta e^{-S(\mathbf{y})/T}\mathrm{d}\mathbf{y}}\propto e^{-S(\mathbf{x})/T}. \tag{8.2}$$

These distributions are indexed by T, the 'temperature' parameter. As $T\to 0$, these densities assign probability 1 to $\mathbf{x}^*$. This can be shown, for example, in the discrete case as follows.

Proposition 8.2.1 *Assume that there exists an $\mathbf{x}^*\in\Theta$, where Θ is discrete and finite, so that $a<b$, where $a=S(\mathbf{x}^*)$ and $b=\min\limits_{\mathbf{x}\neq\mathbf{x}^*}S(\mathbf{x})$. Then*

$$\mathbb{P}\left(\mathbf{X}(T)=\mathbf{x}^*\right)\longrightarrow 1 \text{ as } T\to 0$$

Proof

$$\begin{aligned}\mathbb{P}\left(\mathbf{X}(T)=\mathbf{x}\right)&=\frac{e^{-S(\mathbf{x})/T}}{\sum_{\mathbf{y}\in\Theta}e^{-S(\mathbf{y})/T}}\geq\frac{e^{-a/T}}{e^{-a/T}+(|\Theta|-1)e^{-b/T}}\\&=\frac{1}{1+(|\Theta|-1)e^{-(b-a)/T}}\to 1.\end{aligned}$$

□

The immediate limitation to using simulated annealing is that sampling directly from (8.1) or (8.2) is not usually possible (we discussed problems related to this in the MCMC chapter).

Note also, that to ensure that we draw the global minimum, $T\to 0$ is needed. For fixed T, we do not draw $\mathbf{x}^*$ with probability 1.

This suggests that we use the following approach. We use Markov Chain Monte Carlo (for example, the Metropolis Hastings random walk sampler) to sample from (8.1) or (8.2), while slowly reducing T.

Algorithm 8.2.1 (Simulated Annealing)
Given a cooling sequence $(T_n)_{n\geq 1}$.

1. Start at $\mathbf{X}_0$. Set $n=1$.
2. Generate a candidate state $\mathbf{Y}$ from the proposal density $q(\mathbf{Y}|\mathbf{X}_n)$, where $q(\mathbf{Y}|\mathbf{X}_n)=q(\mathbf{X}_n|\mathbf{Y})$.
3. Compute the acceptance probability
$$\alpha(\mathbf{X}_n,\mathbf{Y})=\min\left\{\exp\left(-\frac{S(\mathbf{Y})-S(\mathbf{X}_n)}{T_n}\right),1\right\}$$
Generate $U\sim\mathcal{U}(0,1)$ and set
$$\mathbf{X}_{n+1}=\left\{\begin{array}{ll}\mathbf{Y} & \text{, if } U\leq\alpha(\mathbf{X}_n,\mathbf{Y})\\ \mathbf{X}_n & \text{, if } U>\alpha(\mathbf{X}_n,\mathbf{Y})\end{array}\right. .$$
4. Set $n=n+1$ and repeat from step 2 onto stopping criterion.

8.2.2 Choosing $(T_n)_{n\geq 1}$

Usually we do not change T at each step, but rather let the Markov Chain Monte Carlo algorithm run for long enough to sample approximately from the desired density before we update T again. For example, we might choose $T_1, \ldots, T_{10^6} = 1$ and $T_{10^6}, \ldots, T_{2\cdot 10^6} = \frac{1}{2}$. In order to guarantee convergence, it is often necessary to let the intervals between updates of T get bigger and bigger as T gets smaller. For example,

$$T_n = T^{(k)}, n_k \leq n < n_{k+1}. \text{ With } n_{k+1} - n_k \to \infty.$$

If Θ is finite, it can be shown that the following choice of cooling schedule ensures convergence (in probability) to $\mathbf{x}^*$:

$$T_n \geq \frac{1}{\log n} |\Theta| \left(\max_{\mathbf{x}\in\Theta} S(\mathbf{x}) - \min_{\mathbf{x}\in\Theta} S(\mathbf{x}) \right).$$

The problem is, in practice, slow cooling is often much slower than necessary and, as a result, very inefficient (having to wait a practically infinite time in order to get an answer is not very useful).

8.2.3 Dealing with Constraints

We have not yet discussed how we build constraints into the simulated annealing algorithm. One way to deal with constraints is to sample in such a way that the constraints are not violated (i.e., include the constraints in our choice of $q(\mathbf{Y}|\mathbf{X}_n)$). An alternative is to change the cost function, S, so that it includes the constraints. Consider the following.

Example 8.2.1 $\min_{\mathbf{x}} x_1^2 + x_2^2$ subject to the constraint that $x_1 + x_2 \leq 1$. The cost function is

$$S(\mathbf{x}) = x_1^2 + x_2^2.$$

We can incorporate the constraints by changing the cost function to

$$S_c(\mathbf{x}) = x_1^2 + x_2^2 + 1000 \cdot \mathbb{1}\{x_1 + x_2 > 1\}.$$

Example 8.2.2 In this example, we want to minimize (without constraints)

$$x^2 + (3-x)^2.$$

We use a Metropolis-Hastings random walk sampler with a normal proposal density and geometric cooling. That is, $T_{i+1} = \beta T_i, \beta \in (0, 1)$.

Listing 8.1: Matlab Code

```
N = 10^5; beta = 0.99;
T = 10; X_0 = 2; X=X_0;
cost = X^2 +(3-X)^2;
for i = 1:N
    X_proposal = X+.5*randn;
    proposal_cost = X_proposal^2+(3-X_proposal)^2;
    alpha = exp(-1/T*(proposal_cost-cost));
    if rand < alpha
        X = X_proposal;
        cost = proposal_cost;
    end
    T = beta*T;
end
```

Note We don't have a stopping condition here, we just run the algorithm for a fixed amount of time.

Example 8.2.3 (The N-Queens Problems) We want to put N queens on a $N \times N$ chessboard so that no queen is on the same row, column or diagonal as another queen. E.g. on a 4×4 board one solution is

	Q		
			Q
Q			
		Q	

This problem can be solved by randomizing optimization.

- Step 1: Find a cost function.
  Here, I use the cost function = # rows with more than one queen.
  + # columns with more than one queen.
  + # diagonals with more than one queen.

- Step 2: Find a way to update a configuration. Our approach is to represent the chessboard as a matrix with ones for queen and zeros for empty spaces. We can update this by randomly swapping two cells. Note $p_{ij} = p_{ji}$ so if we use the Metropolis-Hastings Algorithm

$$\alpha(x,y) = \min\left\{\frac{\exp(-S(y))}{\exp(-S(x))}, 1\right\}.$$

Note Neither the cost function nor updating rule are very smart. If you think about it (and want to spend some time programming), you could come up with better ones. For example, we often swap empty cells with empty cells. A smarter way, that requires more work, would be to move only the queens. A much worse way would be to generate a whole random configuration each time.

Listing 8.2: Matlab Code

```
X_0 = [1 1 1 1 1;0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0];
T = 10; N = size(X_0,1);
X = X_0; current_cost = 10000;
while current_cost ~= 0
    new_X = X;
    randcoords_1 =[ceil(N*rand),ceil(N*rand)];
    randcoords_2 =[ceil(N*rand),ceil(N*rand)];
    new_X(randcoords_1(1),randcoords_1(2)) = X(randcoords_2(1),randcoords_2(2));
    new_X(randcoords_2(1),randcoords_2(2)) = X(randcoords_1(1),randcoords_1(2));
    col_cost = sum(sum(new_X)>1);
    row_cost = sum(sum(new_X')>1);
    diag_cost = 0;
    for i = 1:N
        diag_sum = 0;
        for j = 1:N-i+1
            diag_sum = diag_sum+new_X(i+j-1,j);
        end
        diag_cost = diag_cost + (diag_sum>1);
    end
    for j = 2:N
        diag_sum = 0;
        for i = 1:N-j+1
            diag_sum = diag_sum+new_X(i,i+j-1);
        end
        diag_cost = diag_cost + (diag_sum>1);
    end
    for i = 1:N
        diag_sum = 0;
        for j = 1:N-i+1
            diag_sum = diag_sum + new_X(i+j-1,N-j+1);
        end
        diag_cost = diag_cost + (diag_sum>1);
    end
    for j = 1:N-1
        diag_sum = 0;
        for i = 1:j
            diag_sum = diag_sum + new_X(i,j-i+1);
        end
        diag_cost = diag_cost + (diag_sum>1);
    end
    proposal_cost = row_cost+col_cost+diag_cost;
    T = T*.9999;
    alpha = exp(-(proposal_cost-current_cost)/T);
    if rand <= alpha
        X = new_X; current_cost = proposal_cost;
    end
end
```

Note This does not always converge (we could simply terminate it after enough steps, however).

Bibliography

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

[2] P. Bremaud. *Markov Chains, Gibbs Fields, Monte Carlo Simulations and Queues.* Springer-Verlag, New York, 1999.

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

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

[5] D. Gamerman and H. F. Lopes. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference.* Chapman and Hall / CRC, Boca Raton, 2006.

[6] J. E. Gentle. *Random Number Generation and Monte Carlo Methods.* Springer-Verlag, New York, second edition, 2003.

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

[8] C. Graham and D. Talay. *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation.* Springer, Heidelberg, 2013.

[9] J. Jacod and P. Protter. *Probability Essentials.* Springer-Verlag, Berlin, 2000.

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

[11] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling.* Springer, New York, 2009.

[12] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, Providence, 2009.

[13] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* SIAM, Philadelphia, 1992.

[14] J. Norris. *Markov Chains.* Cambridge University Press, Cambridge, 1997.

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

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

[17] S. Ross. *Simulation.* Academic Press, San Diego, fifth edition, 2013.

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

[19] D. W. Stroock. *An Introduction to Markov Processes.* Graduate Texts in Mathematics. Springer-Verlag, Berlin, 2005.