Methods of Monte Carlo Simulation

Ulm University
Institute of Stochastics

Lecture Notes
Dr. Tim Brereton
Winter Term 2015 – 2016

Ulm

Contents

Chapter 1

Introduction

Speaking very broadly, Monte Carlo methods are tools for solving problems using random numbers. Although this might sound somewhat specific and not very promising, Monte Carlo methods are fundamental tools in many areas of modern science (ranging all the way from theoretical physics to political science). There are a number of reasons why Monte Carlo methods are so useful.

(i) They can be easy. Sometimes it is very straightforward to find an (approximate) answer to a problem using Monte Carlo methods. In these best cases, little thought or time is required to obtain such an answer.

(ii) Sometimes they are the only obvious solution to a problem. Most realistic problems need to be solved numerically. Sometimes it is unclear how to do this using any kind of deterministic method. However, it is often straightforward to develop a crude Monte Carlo algorithm to solve the problem.

(iii) They outperform other methods in high dimensions. These days many interesting problems are very high-dimensional. It can be shown that Monte Carlo methods are often a very good choice (or, even, the best choice) for high dimensional problems.

1.1 Some simple examples

1.1.1 Example 1

The best way to introduce Monte Carlo methods is to start with some simple examples. Suppose we want to solve the integral

$$I = \int_0^1 h(u)\,\mathrm{d}u,$$

for some function h. In some cases (e.g., $h(u) = u^2$) we can do this easily with a pen and paper. However, in other cases, this is not possible. For example, one cannot solve

$$\int_0^\infty e^{-u^2}\,\mathrm{d}u \tag{1.1}$$

by hand. One way to solve such an integral is to use numerical integration (for example, the trapezoidal rule). However, as we will see later in this course, if the integration is in high dimensions such an approach does not work so well. Alternatively, we can use Monte Carlo. In order to do this, we need to rewrite 1.1 into something involving random numbers (which are the necessary ingredient in the Monte Carlo method).

Recall that if U is uniformly distributed on the interval $(0,1)$, i.e., $U \sim \mathcal{U}(0,1)$, then the *probability density function* (pdf) of U, $f(u)$, is given by

$$f(u) = \mathbb{I}\{u \in (0,1)\},$$

where

$$\mathbb{I}\{A\} = \begin{cases} 1 & \text{if } A \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

is the *indicator function*. Now, observe that

$$I = \int_0^1 e^{-u^2}\,\mathrm{d}u = I = \int_0^1 e^{-u^2}\,1\,\mathrm{d}u = \int_{-\infty}^{\infty} e^{-u^2}\mathbb{I}\{u \in (0,1)\}\,\mathrm{d}u = \mathbb{E}e^{-U^2},$$

where $U \sim \mathcal{U}(0,1)$. That means we can solve I by calculating the expected value of the random value e^{-U^2}. Thus, we have turned our problem into one involving random numbers. The next step is to solve this problem. The critical tool we need in order to do this is the *Strong Law of Large Numbers* (SLLN).

Theorem 1.1.1 (Strong Law of Large Numbers)**.** Let $\{Y_n\}_{n=1}^{\infty}$ be a sequence of independent and identically distributed (i.i.d.) random variables with $\mathbb{E}|Y_1| < \infty$. Then,

$$\frac{S_N}{N} \to \mathbb{E}Y_1 \quad \text{almost surely}$$

as $N \to \infty$, where $S_N = Y_1 + \cdots + Y_N$.

This result tells us that we can estimate the expected value of a sequence of i.i.d. random variables by generating a sample and taking the *sample mean*. In the case of our example, the variable of interest is $Y = e^{-U^2}$. Thus, if we can generate an i.i.d. sample of replicates of Y, we can estimate I. We can do this as follows.

Algorithm 1.1.1. Estimating $I = \int_0^1 e^{-u^2}\,\mathrm{d}u$.

(i) Simulate N independent $\mathcal{U}(0,1)$ random variables, $U_1, \ldots, U_N$.

(ii) Return

$$\widehat{I} = \frac{1}{N}\sum_{i=1}^{N} e^{-U_i^2}.$$

This is easily implemented in Matlab.

Listing 1.1: Solving a simple integral

```
N = 10^6;
U = rand(N,1);
Y = exp(-U.^2);
I_hat = sum(Y)/N
```

The estimates returned by this program are very close to the true answer (roughly 0.7468).

1.1.2 Example 2

Finding a good estimate of π is a very classical problem (going back, at least, to ancient Egypt and India). Of course, we have very good estimates now. But it is a nice example of a problem that can be solved quickly and easily using Monte Carlo methods.

As in the first example, to solve this problem using Monte Carlo methods we need to write it as a problem involving random numbers. Recalling that π is the area of a circle with radius 1, we observe that

$$\begin{aligned}\pi &= \int_{-1}^{1}\int_{-1}^{1} \mathbb{I}\{x^2+y^2 \le 1\}\,\mathrm{d}v\,\mathrm{d}u = 4\int_{0}^{1}\int_{0}^{1} \mathbb{I}\{x^2+y^2 \le 1\}\,\mathrm{d}v\,\mathrm{d}u \\ &= 4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{I}\{x^2+y^2 \le 1\}\mathbb{I}\{u \in (0,1)\}\mathbb{I}\{v \in (0,1)\}\,\mathrm{d}v\,\mathrm{d}u \\ &= 4\mathbb{E}\mathbb{I}\{U^2+V^2 \le 1\},\end{aligned}$$

where $U \sim \mathcal{U}(0,1)$ and $V \sim \mathcal{U}(0,1)$ are independent.

Using the SLLN again, we have the following recipe for estimating π.

Algorithm 1.1.2. Estimating π.

(i) Simulate $2N$ independent $\mathcal{U}(0,1)$ random variables, $U_1,\ldots,U_N$ and $V_1,\ldots,V_N$.

(ii) Return

$$\widehat{\pi} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\{U_i^2+V_i^2 \le 1\}.$$

Again, this is easily implemented in Matlab.

Listing 1.2: Estimating π

```
N = 10^6;
U = rand(N,1);
V = rand(N,1);
Y = ((U.^2 + V.^2) <= 1);
p_hat = 4 * sum(Y) / N
```

Here the estimate is OK up to the 3rd decimal place. We will spend a lot of time in this course focusing on ways to improve the accuracy of Monte Carlo estimators.

Chapter 2

Generating Uniform Random Numbers

2.1 How do we make random numbers?

The two examples we have considered so far both required, as input, sequences of i.i.d. $\mathcal{U}(0,1)$ random variables. As it turns out, these are really the essential ingredient in Monte Carlo. This is because almost any random object can be made using $\mathcal{U}(0,1)$ random objects as building blocks.

So, to begin, we should think about just how we might go about making a stream of i.i.d. $\mathcal{U}(0,1)$ random variables. A good method of generating such random numbers should have the following properties:

(i) The random numbers should have a $\mathcal{U}(0,1)$ distribution.

(ii) The random numbers should be independent.

(iii) The method should be very fast and not require a large amount of computer memory.

Unfortunately, there is a big problem in finding such a method of generating random numbers on a computer: nothing about a computer is random. Thus, we are left with two options:

(i) Use a physical device to generate random numbers (e.g., via radioactive decay).

(ii) Generate a sequence of numbers that are deterministic (i.e., not at all random) but "act like random numbers".

2.1.1 Physical generation

Although the physical idea sounds like a good idea, it actually does not work well. There are a few reasons for this.

(i) In order to turn physical data into uniform random variables, we need to know the distribution of the physical data. Usually we do not know this exactly.

(ii) There are measurement errors which can distort the physical information.

(iii) These methods are slow.

Strangely enough, even though they are not random, numbers made by a computer can often be much more convincingly random than "real" random numbers.

2.1.2 Pseudo-random numbers

Thus, the best idea in practice is usually to use a deterministic sequence of numbers generated by a computer that "act like random numbers". Such numbers are called *pseudo-random numbers*. What exactly is means to "act like random numbers" is something of a philosophical question. However, people now more or less agree that it is sufficient that the numbers fool as many statistical tests as possible into thinking that:

(i) Each random number is $\mathcal{U}(0,1)$ distributed.

(ii) The random numbers are independent of one another.

It turns out that there are a number of ways to make such numbers.

2.2 Pseudo-random numbers

2.2.1 Abstract setting

Most random number generators can be described by a general algorithm. This algorithm has the following ingredients:

- A *state space*, S, which is a finite set.
- A *transition function*, $f : S \to S$.
- A *seed*, $S_0 \in S$.
- An *output space*, U (normally $U = (0,1)$).
- An *output function*, $g : S \to U$.

The general random number generation technique is then given in Algorithm 2.2.1.

Algorithm 2.2.1 (Abstract random number generator)**.**
Given the above ingredients,

1. **Initialize:** Set $X_1 = S_0$. Set $n = 2$.
2. **Transition:** Set $X_n = f(X_{n-1})$.
3. **Output:** Set $U_n = g(X_n)$.
4. **Repeat:** Set $n = n + 1$. Repeat from step 2.

2.2.2 Linear Congruential Generators

One of the simplest and most widely used random number generators is the *linear congruential generator*. This is described in Algorithm 2.2.4.

Algorithm 2.2.2 (Linear congruential generator)**.**
Given a, c and m,

1. **Initialize:** Set $X_1 = S_0$. Set $n = 2$.
2. **Transition:** Set $X_n = (aX_{n-1} + c) \mod m$.
3. **Output:** Set $U_n = X_n/m$.
4. **Repeat:** Set $n = n + 1$. Repeat from step 2.

Here, a is called the *multiplier*, c is called the *increment* and m is called the *modulus*. Eventually, such a sequence will start to repeat. The length of the sequence before it starts repeating is called the *period*.

Example 2.2.1 (A very simple LCG)**.**
Take $a = 6$, $m = 11$, $c = 0$ and $S_0 = 1$. Then,

$$\begin{aligned} X_1 &= 1 \\ X_2 &= (6 \cdot 1 + 0) \mod 11 = 6 \\ X_3 &= (6 \cdot 6 + 0) \mod 11 = 3 \\ &\vdots \end{aligned}$$

If we write out the whole sequence, we get $1, 6, 3, 7, 9, 10, 5, 8, 4, 2, 1, 6, \ldots$ (notice the sequence starts repeating once a number reoccurs). There are a total of 10 numbers before it starts repeating. That is, the period of the LCG is 11. This generator is easily implemented in Matlab.

Listing 2.1: A simple LCG

```
N = 12;
a = 6; m = 11; c = 0;
S_0 = 1;
X = zeros(N,1); U = zeros(N,1);
X(1) = S_0; U(1) = X(1) / m;
for i = 2:N
    X(i) = mod(a*X(i-1) + c,m);
    U(i) = X(i) / m;
end

X'
U'
```

What happens if we take $a = 3$ instead? Then,

$$\begin{aligned} X_1 &= 1 \\ X_2 &= (3 \cdot 1 + 0) \mod 11 = 3 \\ X_3 &= (3 \cdot 6 + 0) \mod 11 = 9 \\ &\vdots \end{aligned}$$

If we write out the whole sequence, we have $1, 3, 9, 5, 4, 1, 3, \ldots$. This has only got a period of 5.

A natural question to ask when consider an LCG is what determines its period. In particular, we want to know when an LCG has *full period* (i.e., for a given choice of m it is not possible to find a larger period). In general, the maximum period is the size of the state space, S. Now, for fixed m and $c = 0$, the largest possible state space is $S = \{1, \ldots, m-1\}$ (because, if $X = 0$ then $aX = 0$ and the LCG becomes stuck). Thus, the biggest possible period is $m - 1$. For $c = 0$, $X = 0$ is allowed because $aX + c \neq 0$. Thus, for $c \neq 0$, the maximum period is m.

For LCGs with $c = 0$, we have the following result.

Theorem 2.2.2. An LCG with $c = 0$ has full period, $m - 1$, if

(i) $S_0 \neq 0$,

(ii) $a^{m-1} - 1$ is a multiple of m,

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

The example with $a = 3$ and $m = 11$ fails because $3^{6-1} \mod 11 = 0$. For LCGs with $c \neq 0$, the following theorem holds.

Theorem 2.2.3. An LCG with $c \neq 0$ has full period, m, if

(i) c and m are relatively prime (they have no common divisors other than 1),

(ii) every prime number that divides m divides $a - 1$,

(iii) $a - 1$ is divisible by 4 if m is.

This theorem is satisfied if c is odd, m is a power of 2 and $a = 4m + 1$.

Example 2.2.4 (Minimal standard LCG)**.** Many early LCGs had periods of size $2^{31} - 1 \approx 2.1 \times 10^9$. One example is the so-called "minimal standard LCG". This has $a = 7^5 = 16807$, $c = 0$ and $m = 2^{31}$. Of course, these days such a small period would be totally inacceptable.

2.2.3 Improvements on LCGs

A significant disadvantage of LCGs is that they have small periods unless m is chosen very large. This is a significant problem because these days it is common to carry out Monte Carlo experiments using 10^{15} or more samples. Clearly the period of a RNG should be at least as long as the number of samples required in a simulation. As a rule of thumb, if N is the required sample size then the period should be bigger than N^2 or (more conservatively) N^3. This means modern RNGs should have a period of at least 10^{30} or 10^{45}. Using 64-bit numbers, an LCG would have a maximum period of $2^{64} \approx 1.8 \times 10^9$ which is clearly too small. For this reason, researchers developed a number of modifications of the LCG that have much larger periods. The basic idea of these generators is to increase the size of the state space. The first one we will consider is the *multiple recursive generator.*

Algorithm 2.2.3 (Multiple Recursive Generator)**.**
Given $a_1, \ldots, a_k$ and m,

1. **Initialize:** Set $X_1 = S_0^{(1)}, \ldots, X_k = S_0^{(k)}$. Set $i = k + 1$.

2. **Transition:** Set $X_i = (a_1 X_{i-1} + \cdots + a_k X_{i-k}) \mod m$.

3. **Output:** Set $U_i = X_i/m$.

4. **Repeat:** Set $i = i + 1$. Repeat from step 2.

We can place this in the abstract framework from above. Here, the state space is $S = \{0, 1, \ldots, m-1\}^k \setminus (0, \ldots, 0)$, which is of size $m^k - 1$. A single state is then given by $\mathbf{x} = (X^{(1)}, \ldots, X^{(k)})$. The update is of the form

$$\mathbf{x}_n = f(\mathbf{X}_{n-1}) = (X_{n-1}^{(2)}, \ldots, X_{n-1}^{(k)}, \mathbf{a}^\intercal \mathbf{X}_{n-1}),$$

where $\mathbf{a} = (a_k, \ldots, a_1)$. Given the size of the state space, the maximum period of an MRG is $m^k - 1$.

Alternative, one can obtain a bigger period by combining LCGs with different periods. An example of this is the Wichman-Hill generator.

Algorithm 2.2.4 (Wichman-Hill Generator)**.**
Given a_1, a_2, a_3 and m_1, m_2, m_3,

1. Set $X_0 = S_0^X$, $Y_0 = S_0^Y$ and $Z_0 = S_0^Z$. Set $n = 2$.

2. Set $X_n = (a_1 X_{n-1}) \mod m_1$.

3. Set $Y_n = (a_2 Y_{n-1}) \mod m_2$.

4. Set $Z_n = (a_3 Z_{n-1}) \mod m_3$.

5. Set
$$U_i = \left(\frac{X_n}{m_1} + \frac{Y_n}{m_2} + \frac{Z_n}{m_3} \right) \mod 1$$

6. Set $i = i + 1$. Repeat from step 2.

Note that $\mod 1$ means return the decimal part of a number.

A natural idea is to combine multiple recursive generators. The MRG32k3a generator, developed by Pierre L'Ecuyer, is one such generator. It is implemented in Matlab and has very good properties. This generator uses two multiple recursive generators with update functions

$$X_n = (1403580 X_{n-2} - 810728 X_{n-3}) \mod m_1.$$

with $m_1 = 2^{32} - 209$ and

$$Y_n = (527612 Y_{n-1} - 1370589 Y_{n-3}) \mod m_2,$$

with $m_2 = 2^{32} - 22853$. These are combined by the output function

$$U_n(X_n, Y_n) = \begin{cases} \frac{X_n - Y_n + m_1}{m_1 + 1} & \text{if } X_n \le Y_n, \\ \frac{X_n - Y_n}{m_1 + 1} & \text{if } X_n > Y_n. \end{cases}$$

The period of this generator is approximate 3×10^{57}.

Chapter 3

Non-Uniform Random Variable Generation

The basic ingredient in all Monte Carlo simulations is a stream of iid $\mathcal{U}(0,1)$ random numbers. We need to transform these into interesting random objects. The first such objects we will consider are random variables with non-uniform distributions. Two generic methods exists for generating random variables: the *inverse transform* method and the *acceptance-rejection* method.

3.1 The Inverse Transform

The distribution of a random variable is determined by its *cumulative distribution function* (cdf), $F(x) = \mathbb{P}(X \leq x)$. A cdf has the properties given in Theorem 3.1.1.

Theorem 3.1.1. F is a cdf if and only if

(i) $F(x)$ is non-decreasing,

(ii) $F(x)$ is right-continuous (i.e. when there is a jump discontinuity, the function takes the right-hand value at the location of the jump),

(iii) $\lim_{x\to-\infty} F(x) = 0$ and $\lim_{x\to\infty} F(x) = 1$.

The *inverse cdf* corresponding to the cdf $F(x)$ is defined as $F^{-1}(y) = \inf\{x : F(x) \geq y\}$. The *inverse cdf* is sometimes called the generalized inverse or quantile function.

Theorem 3.1.2. If $F(x)$ is a cdf with inverse $F^{-1}(u) = \inf\{x : F(x) \geq u\}$ and $U \sim \mathcal{U}(0,1)$, then $X = F^{-1}(U)$ is a random variable with cdf F.

Proof. To see this, consider the cdf of $F^{-1}(U)$. This is given by

$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x),$$

where the first equality follows from the definition of the inverse cdf and the second equality follows from the fact that a cdf takes values in $[0,1]$ and $\mathbb{P}(U \leq x) = x$ for $0 \leq x \leq 1$. $\square$

This leads us to the inverse transform algorithm.

1. Generate $U \sim \mathcal{U}(0,1)$.
2. Return $X = F^{-1}(U)$.

In order to use the inverse-transform method, we need to find a convenient expression for the inverse, $F^{-1}(u)$. If X is a continuous random variable then F is bijective. This means we can find $F^{-1}(u)$ by solving $F(x) = u$.

Example 3.1.3 (Exponential distribution)**.** Suppose we wish to simulate replicates of an exponential random variable, X. The cdf of the exponential distribution is given by $F(x) = 1 - \exp(-\lambda x)$. We can solve for $F^{-1}(u)$ as follows.

$$F(x) = u = 1 - e^{\lambda x} \Rightarrow x = -\frac{1}{\lambda}\log(1-u) = F^{-1}(u).$$

This immediately gives us a recipe for simulating exponential random variables with parameter λ: generate $U \sim \mathcal{U}(0,1)$ then return $X = -\frac{1}{\lambda}\log(1-U)$. Note that we can simplify this by setting $X = -\frac{1}{\lambda}\log(U)$. This is because, if $U \sim \mathcal{U}(0,1)$ then $1 - U \sim \mathcal{U}(0,1)$. We can easily simulate an exponential random variable in Matlab.

Listing 3.1: Matlab Code

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

Example 3.1.4 (Uniform distribution on $\mathcal{U}(0,b)$)**.** Consider the uniform distribution on $\mathcal{U}(0,b)$, where $b > 0$. This has the cdf

$$F(x) = \mathbb{P}(X \le x) = \begin{cases} 0 & x < 0 \\ \frac{x}{b} & 0 \le x < b \\ 1 & x \ge b \end{cases}.$$

We can solve $F(x) = u$ to get $F^{-1}(u)$:

$$F(x) = u = \frac{x}{b} \text{ for all } u \in (0,1) \Rightarrow x = bu.$$

Thus, we can generate $X \sim \mathcal{U}(0,b)$ by generating $U \sim \mathcal{U}(0,1)$ and setting $X = bU$.

When considering discrete random variables, $F^{-1}(u)$ is a step function (from a programming perspective, this is a series of if statements).

Example 3.1.5 (Bernoulli distribution)**.** Let X be a Bernoulli random variable with success probability 1. Then, the cdf is given by

$$F(x) = \mathbb{P}(X \le x) = \begin{cases} 0 & x < 0 \\ 1-p & 0 \le x < 1 \\ 1 & x \ge 1 \end{cases}.$$

The inverse cdf is then given by

$$F^{-1}(u) = \begin{cases} 0 & u < 1-p \\ 1 & u \ge p \end{cases}.$$

In Matlab, this can be written as follows.

Listing 3.2: Matlab Code

```
p = 0.5;
U = rand;
if U < 1 - p
  X = 0;
else
  X = 1;
end
```

In practice, this is often simplified to the following.

Listing 3.3: Matlab Code

```
X = (rand <= p);
```

Note that this is not exactly inverting the cdf (but does exactly the same thing in the end).

In addition to continuous and discrete random variables, there are random variables that are neither discrete nor continuous, but rather a mixture of both. Care must be taken when working with such random variables (an example can be found in your exercises).

3.1.1 The table lookup method

As mentioned above, generating discrete random variables by the inverse transform method essentially just involves evaluating a lot of if statements. This can become quite computationally intensive. The table lookup method is a method for reducing the time spent generating a discrete random variable taking values in a finite set. It has a cost, however, as some precomputation is required. Such precomputation makes sense if the random variable in question needs to be generated a lot (as we then save time in the long run). The idea is to arrange all the possible values of the random variable in a table (often entering a value multiple times) so that drawing a value uniformly at random from the table results in a random variable with the correct distribution. In order to implement this method, we first need to be able to draw uniformly from a finite set $\{1, \ldots, K\}$.

Example 3.1.6 (Uniform random variables on $\{1, \ldots, K\}$)**.** Suppose we want to draw a random variable uniformly from the set $\{1, \ldots, K\}$. We can use the following lemma to do this.

Lemma 3.1.7. *If $U \sim \mathcal{U}(0,1)$, then $X = \lceil kU \rceil$ is uniformly distributed on $\{1, \ldots, K\}$.*

Proof. We have

$$\mathbb{P}(X = k) = \mathbb{P}(\lceil KU \rceil = k) = \mathbb{P}\left(\frac{k-1}{K} < U \leq \frac{k}{K}\right) = \frac{k}{K} - \frac{k-1}{K} = \frac{1}{K}.$$

□

This suggests the obvious recipe: draw $U \sim \mathcal{U}(0,1)$ then return $X = \lceil KU \rceil$. This is easily implemented in Matlab.

Listing 3.4: Matlab Code

```
K = 10;
X = ceil(K*rand);
```

We can now implement the table lookup method. We suppose that we have a random variable X taking values on the finite set $\{1, \ldots, K\}$ (or any set that can be labeled from 1 to K). We further require that the probabilities are all rational. In this case, all probabilities can be written in the form

$$p_i = \mathbb{P}(X = i) = \frac{n_i}{M} \quad \text{for all } i \in \{1, \ldots, K\},$$

where M is the same for all values of i. We make a table with M rows in it. The first n_1 of these rows contain the value 1, the next n_2 rows contain the value 2 and so on. We then use the following algorithm to sample values of X:

1. Draw $U \sim \mathcal{U}(0, 1)$.
2. Set $I = \lceil MU \rceil$.
3. Return the value in the Ith row of the table.

Example 3.1.8. Suppose X is a random variable such that

$$\mathbb{P}(X = 0) = \frac{1}{3}, \quad \mathbb{P}(X = 1) = \frac{1}{2}, \quad \mathbb{P}(X = 2) = \frac{1}{6}.$$

We can write all these probabilities as ratios involving $M = 6$. We can then generate from this distribution using the following code.

Listing 3.5: Matlab Code

```
N = 10^5;
M = 6;
T = [0 0 1 1 1 2];
X = zeros(N,1);
for i = 1:N
  X(i) = T(ceil(rand*M));
end
hist(X,3)
```

3.2 Integration on Various Domains and Properties of Estimators

3.2.1 Integration over $(0, b)$

We already considered the problem of estimating integrals of functions over the interval $(0, 1)$. What do we do in the case where the interval is arbitrary?

First, lets consider the problem of estimating $\ell = \int_0^b S(x)\,\mathrm{d}x$, for some arbitrary function S. As before, we rewrite this by multiplying the inside by 1 (this time the ratio of a pdf with itself). Using the uniform density on $(0, b)$ — which is $f(x) = 1/b$ — we write

$$\ell = \int_0^b S(x)\mathrm{d}x = \int_0^b \frac{(f(x)}{f(x)} S(x)\,\mathrm{d}x = \int_0^b \frac{b}{b} S(x)\mathrm{d}x = b\int_0^b S(x)\frac{1}{b}\,\mathrm{d}x = b\mathbb{E}S(X),$$

where $X \sim \mathcal{U}(0,b)$. This suggests an estimator of the form

$$\widehat{\ell} = \frac{b}{N}\sum_{i=1}^{N} S(X_i),$$

where the $\{X_i\}$ are iid $\mathcal{U}(0,b)$ random variables.

3.2.2 Properties of estimators

An important question is whether the above estimator is any good. Two important measures of estimator quantity are *bias* and *variance*. An estimator, $\widehat{\ell}$, of a quantity ℓ is said to be *unbiased* if $\mathbb{E}\widehat{\ell} = \ell$. If $\mathbb{E}\widehat{\ell} < \ell$, it is negatively-biased. If $\mathbb{E}\widehat{\ell} > \ell$, it is positively biased.

It is generally straightforward to check if an estimator is unbiased. In our example, using the properties of iid random variables, we can write

$$\mathbb{E}\widehat{\ell} = \mathbb{E}\left[\frac{b}{N}\sum_{i=1}^{N} S(X_i)\right] = \frac{b}{N}\sum_{i=1}^{N} S(X_i) = b\mathbb{E}S(X_1) = b\int_0^b S(x)\frac{1}{b}\,\mathrm{d}x = \int_0^b S(x)\,\mathrm{d}x,$$

demonstrating that the estimator is unbiased.

It is usually difficult to get an explicit expression for the variance. In our case, using the iid properties of the random variables, we can obtain

$$\begin{aligned}\mathrm{Var}(\widehat{\ell}) &= \mathrm{Var}\left(\frac{b}{N}\sum_{i=1}^{N} S(X_i)\right) = \frac{b^2}{N^2}\mathrm{Var}\left(\sum_{i=1}^{N} S(X_i)\right) \\ &= \frac{b^2}{N^2}\sum_{i=1}^{N}\mathrm{Var}(S(X_i)) = \frac{b^2}{N}\mathrm{Var}(S(X_1)).\end{aligned}$$

Observe that, in order to calculate $\mathrm{Var}(S(X_1))$ we need to know $\mathbb{E}S(X_1)$ (which is exactly what we are trying to calculate). In practice, we usually estimate the variance of the estimator based on the sample.

Example 3.2.1. Suppose we want to estimate $\int_0^2 e^{-x^2}\,\mathrm{d}x$. We can use the estimator above. This leads to the following Matlab code.

Listing 3.6: Matlab Code

```
N=10^4;
b = 2;
X = zeros(N,1);
Y = zeros(N,1);
for i = 1:N
  X(i) = b*rand;
  Y(i) = b*exp(-X(i)^2);
end
ell = erf(2) * sqrt(pi) / 2
ell_hat = mean(Y)
ell_hat_var = b^2/N * var(Y)
ell_hat_std = b/sqrt(N) * sqrt(Y)
```

3.2.3 Integration over unbounded domains

We can extend the idea of multiplying the integrand by the ratio of a pdf with itself to integration problems with possibly unbounded domains. Let $X \sim g$ be a random variable with *support* I (the support of X is $\{x \in \mathbb{R} : g(x) > 0\}$). For example, we could have $I = (-\infty, \infty)$ if g is the density of a normal random variable or $I = (0, \infty)$ if g is the density of an exponential random variable.

Then, we can write

$$\ell = \int_I S(x)\,\mathrm{d}x = \int_i S(x)\frac{g(x)}{g(x)}\,\mathrm{d}x = \mathbb{E}\frac{S(X)}{g(X)},$$

where $X \sim g$. If $\mathbb{E}|\frac{S(X)}{g(X)}|$, then we can use the SLLN to obtain the estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N}\frac{S(X_i)}{g(X_i)},$$

where the $\{X_i\}$ are iid with density g.

Example 3.2.2. Suppose we want to estimate $\ell = \int_0^\infty e^{-x^2/2}\,\mathrm{d}x$ using exponential random variables. Then, rewriting the integral as above, we have

$$\int_0^\infty e^{-x^2/2}\frac{e^{-x}}{e^{-x}}\,\mathrm{d}x = \int_0^\infty e^{-x^2/2+x}e^{-x}\,\mathrm{d}x = \mathbb{E}\exp\{-X^2/2 + X\},$$

where $X \sim \mathsf{Exp}(1)$. This leads to the estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N}\exp\{-X_i^2 + X_i\},$$

where the $\{X_i\}$ are iid $\mathsf{Exp}(1)$.

Listing 3.7: Matlab Code

```matlab
N=10^4;
X=zeros(N,1);
Y=zeros(N,1);
for i = 1:N
  X(i) = -log(rand);
  Y(i) = exp(-X(i)^2/2 + X(i));
end
ell = sqrt(pi/2)
ell_hat = mean(Y)
ell_hat_std = std(Y)/sqrt(N)
```

In order to this approach to be effective, the density $g(x)$ must be chosen correctly. We will talk about good choices for g later in this course.

3.3 The Acceptance-Rejection Method

3.3.1 The continuous case

Suppose we want to sample from a given pdf, f, (which we know at least up to its normalizing constant). Suppose, furthermore, that this is a non-standard density such that no standard algorithm for sampling from it exists. The idea of the acceptance rejection method is to instead sample random variables from a density, g, from which it is easy to sample (e.g., the exponential density). However, we only accept some of these samples. The decision of whether to accept a given sample or not is based on a Bernoulli experiment. The probabilities of these experiments are chosen such that the final sample is a sample from the density f.

The acceptance-rejection algorithm is (at least from one perspective) based on the fact that if one could sample a random vector (X, U) uniformly from the region under the graph of f, then $X \sim f$. This is shown by the following lemma.

Lemma 3.3.1. Let (X, U) be a uniform random sample from $G = \{(x, u) \in \mathbb{R}^2 : f(x) > 0, u \in (0, f(x))\}$. Then, $X \sim f$.

Proof. We first write out $h(x, u)$, the joint pdf of (X, U). This is given by

$$h(x,u) = \frac{\mathbb{I}\{(x,u) \in G)\}}{|G|} = \mathbb{I}\{(x,u) \in G)\},$$

as the area under a density is 1. Now, $h(x)$, the marginal density of X, is given by

$$\int h(x,u)\,\mathrm{d}u = \int_0^{f(x)} 1\,\mathrm{d}u = f(x).$$

$\square$

Thus, if we could sample uniformly under the graph of f, we could generate random variables with the desired density. In general, however, this is not to easy to do. Fortunately, the converse of the above lemma also holds. That is, if $X \sim f$ and $U \sim \mathcal{U}(0, f(X))$, then (X, U) is distributed uniformly on G. This suggests the following strategy. We could draw random variables from another density g, which is easy to sample from. We could then obtain samples uniformly distributed under the graph of g. We could, indeed, get samples under the graph of Cg where $C \geq 1$ is some constant by drawing $\widetilde{X}$ from g then $U \sim \mathcal{U}(0, Cg(X))$. If we could choose C such that $Cg(x) \geq f(x)$ for all $x : f(x) > 0$ then the area under Cg would include all of the area under f. By taking that part of our uniform random sample which lies under f — that is, those $\widetilde{X}$ where $U \leq f(\widetilde{X})$ — we would obtain a uniform sample in the area under f, giving us a sample from the desired density. This leads to the acceptance-rejection algorithm:

1. Draw $\widetilde{X} \sim g$.
2. Draw $U \sim \mathcal{U}(0, 1)$ (note that $Cg(X)U \sim \mathcal{U}(0, Cg(X))$).
3. If
$$U \leq \frac{f(\widetilde{X})}{Cg(\widetilde{X})},$$
return $X = \widetilde{X}$. Otherwise, return to step 1.

The density g is often called the *proposal density*. The ingredients of the acceptance-rejection method are illustrated in Figure 3.3.1.

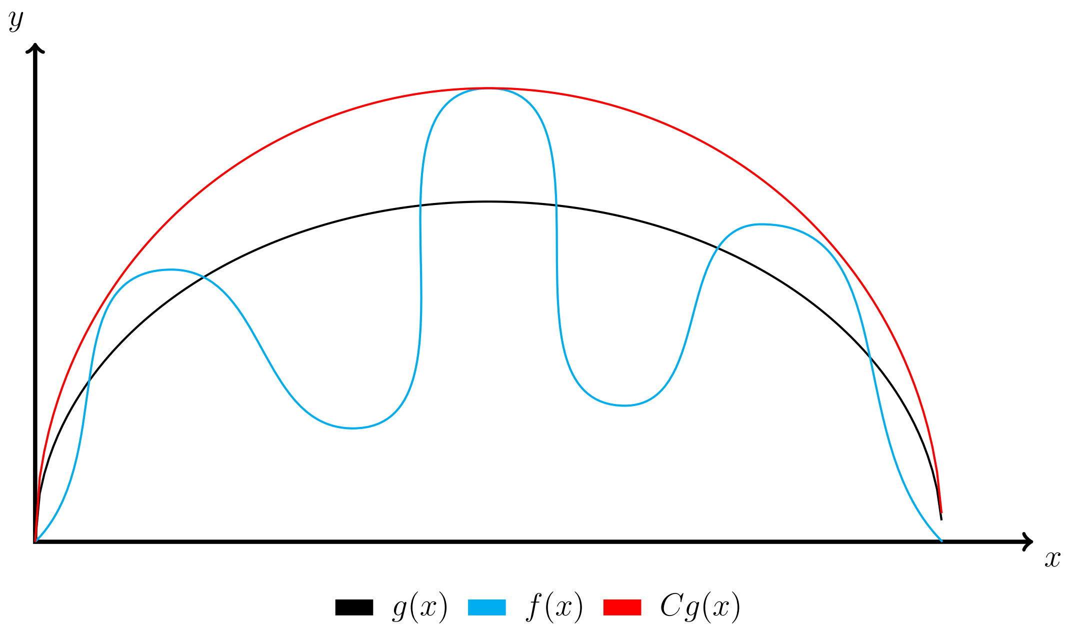


Figure 3.3.1: Illustration of the components of the acceptance-rejection method. We sample $\widetilde{X}$ from g, then accept it with probability $f(\widetilde{X})/Cg(\widetilde{X})$.

Of course, the above justification of the acceptance-rejection algorithm is more or less just heuristic. We should check that it actually works. Theorem 3.3.2 shows that it does.

Theorem 3.3.2. Given a $C < \infty$ such that $Cg(x) \geq f(x)$ for all $x : f(x) > 0$, the acceptance rejection algorithm described above returns a random variable $X \sim f$.

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

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

Using the continuous law of total probability, the numerator can be written as

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

The denominator can be written as

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

So

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

□

The constant C and the acceptance probability

In order for acceptance-rejection to work we need to be able to find a finite constant C such that $Cg(x) \geq f(x)$ for all $x \in \mathbb{R}$. We can obtain an obvious choice for such a constant by rearranging the above inequality to get that C must satisfy

$$C \geq \frac{f(x)}{g(x)}.$$

This is clearly satisfied by

$$C = \sup_{x\in\mathbb{R}} \frac{f(x)}{g(x)}.$$

We then just need to check that $\sup_{x\in\mathbb{R}} f(x)/g(x) < \infty$. Note that this will not always be the case. For example, if we use a normal density for g and an exponential density for f, we have

$$\frac{f(x)}{g(x)} \propto \frac{e^{-x}}{e^{-x^2/2}} = e^{x^2/2-x} \to \infty \text{ as } x \to \infty.$$

In general, for acceptance-rejection to work, the tails of the density g should go to 0 faster than the tails of f. That is $g(x)$ should go to 0 faster than $f(x)$ when $x \to \infty$ and $x \to -\infty$.

The constant C that is chosen is very important, because it determines the probability we accept a given random variable generated from g. This probability is just the area under f divided by the area under Cg. That is,

$$\mathbb{P}(\text{accept}) = \frac{\int_{\mathbb{R}} f(x)\,\mathrm{d}x}{\int_{\mathbb{R}} Cg(x)\,\mathrm{d}x} = \frac{1}{C}.$$

Clearly, the higher the probability that we accept that random variable, the less work our algorithm has to do. It would be bad, for example, if our algorithm only accepts every 10 millionth random variable on average. The best possible choice of C is the smallest choice that satisfies the conditions. Note that this choice is exactly $C = \sup_{x\in\mathbb{R}} f(x)/g(x)$, which is, by definition, the smallest value that satisfies the inequality.

Acceptance-rejection and normalizing constants

One of the nice things about acceptance rejection is that we do not need to know the normalizing constant of the density we are sampling from (which is often something difficult to calculate). That is, suppose $f(x)$ can be written in the form

$$f(x)\frac{h(x)}{Z},$$

where $h(x) \geq 0$ for all x and $Z = \int h(x)\,\mathrm{d}x$. Given a gensity g, The obvious choice of C is

$$C = \sup_x \frac{h(x)/Z}{g(x)} = \frac{1}{Z}\sup_x \frac{h(x)}{g(x)} = \frac{1}{Z}C^*,$$

wher $C^* = \sup_x \frac{h(x)}{g(x)}$. Plugging this into the inequality needed for acceptance, we see that

$$U \leq \frac{f(x)}{Cg(x)} \Leftrightarrow U \leq \frac{h(x)/Z}{C^* g(x)/Z} = \frac{h(x)}{C^* g(x)}.$$

Thus we do not need to know Z, only $h(x)$.

Example 3.3.3. Suppose we want to generate an exponential random variable with parameter 1 conditional on it being less than 1. We can write this density as

$$f(x) = \frac{e^{-x}\mathbb{I}\{x \leq 1\}}{\mathbb{P}(X \leq 1)} \propto e^{-x}\mathbb{I}\{x \leq 1\}.$$

Here, $h(x) = e^{-x}$ and $Z = \mathbb{P}(X \leq 1) = 1 - e^{-1}$. Let's use a uniform random variable as our proposal. That is, lets take $g(x) = \mathbb{I}x \in (0,1)$. We can then calculate C^*. This is given by

$$C^* = \sup_{x\in(0,1)} \frac{h(x)}{g(x)} = \sup_{x\in(0,1)} \frac{e^{-x}}{1}.$$

As the supremum of e^{-x} on $(0,1)$ is at $x = 0$, we have that $C^* = e^0 = 1$. Thus, we will accept our proposal $\widetilde{X}$ when

$$U \leq \frac{h(x)}{Cg(x)} = \frac{e^{-x}}{1} = e^{-x}.$$

We can implement this in Matlab as follows.

Listing 3.8: Matlab Code

```
N=10^6;
X = zeros(N,1);

for i = 1:N
  X_tilde = rand;
  while rand > exp(-X_tilde)
    X_tilde = rand;
  end
  X(i) = X_tilde;
end
```

3.3.2 The discrete case

Suppose we want to a sample a random variable, X, taking values in some countable set $\mathcal{X}$ with *probability mass function* (pmf) $\mathbf{p}$, where

$$\mathbb{P}(X = i) = p_i.$$

In some cases, such a distribution could be difficult to sample from. For example, if the size of $\mathcal{X}$ is quite large and the distribution of X is non-standard. In such a case, we can use a discrete version of acceptance rejection. In order for this to work, we need to be able to simulate realizations of another random variable, $\widetilde{X}$, taking values in $\mathcal{X}$ with pmf $\mathbf{q}$. That is,

$$\mathbb{P}(\widetilde{X} = i) = q_i.$$

Analogously to the continuous case, for acceptance rejection to work, we need a constant, $C < \infty$, such that $Cq_i \geq p_i$ for all $i \in \mathcal{X}$. If so, we can use the following algorithm:

(i) Draw $\widetilde{X}$ according to $\mathbf{q}$.

(ii) Draw $U \sim \mathcal{U}(0, 1)$. If
$$U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}}$$
return $X = \widetilde{X}$. Otherwise, go to step 1.

Note that an optimal C can be very difficult to calculate in the discrete setting.

As in the continuous case, we should make sure our algorithm works.

Theorem 3.3.4. Discrete acceptance rejection works.

Proof. Analogously to the discrete case, we have to find

$$\mathbb{P}\left(\widetilde{X} = i \,|\, U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}}\right) = \frac{\mathbb{P}\left(\widetilde{X} = i, U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}}\right)}{\mathbb{P}\left(U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}}\right)}$$

for all $i \in \mathcal{X}$. Let $\mathcal{X}_+ = \{i \in \mathcal{X} : p_i > 0\}$. Then, for all $i \in \mathcal{X}_+$,

$$\begin{aligned}\frac{\mathbb{P}\left(\widetilde{X} = i, U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}}\right)}{\mathbb{P}\left(U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}}\right)} &= \frac{\mathbb{P}\left(U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}} \,|\, \widetilde{X} = i,\right)\mathbb{P}(\widetilde{X} = i)}{\sum_{j \in \mathcal{X}_+} \mathbb{P}\left(U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}} \,|\, \widetilde{X} = j,\right)\mathbb{P}(\widetilde{X} = j)} \\ &= \frac{\frac{p_i}{Cq_i} q_i}{\sum_{j \in \mathcal{X}_+} \frac{p_j}{Cq_j} q_j} = p_i\end{aligned}$$

If, instead $p_i = 0$, then

$$\mathbb{P}\left(\widetilde{X} = i \,|\, U \leq \frac{p_{\widetilde{X}}}{Cq_{\widetilde{X}}}\right) = 0 = p_i.$$

Thus, we have shown the random variable we simulate has the desired distribution. □

Example 3.3.5 (A Poisson random variable condition on being less than 11)**.** Suppose we want to generate a Poisson random variable, X, with parameter λ, conditional on the event $\{X \leq 10\}$. The pmf of X is given by

$$\mathbb{P}(X = i) = p_i = \frac{e^{-\lambda}\frac{\lambda^i}{i!}\mathbb{I}\{i \leq 10\}}{\sum_{j=0}^{10} e^{-\lambda}\frac{\lambda^j}{j!}} \propto e^{-\lambda}\frac{\lambda^i}{i!}\mathbb{I}\{i \leq 10\}.$$

A straightforward choice of the proposal pmf, $\mathbf{q}$, is a uniform distribution on $\{0, \dots, 10\}$. That is, $q_i \propto \mathbb{I}\{i \in \{0, \dots, 10\}$. The optimal choice of C^* (the C calculated ignoring normalizing constants) is given by

$$C^* = \max i \in \{0, \dots, 10\}\frac{p_i}{Cq_i}.$$

For general λ, this is very difficult to compute (you can try it yourself). So, to make life easier, lets take $\lambda = 1$. In this case,

$$C^* = \max i \in \{0, \dots, 10\}\frac{p_i}{Cq_i} = \max i \in \{0, \dots, 10\}\frac{e^{-1}}{i!} = e^{-1}.$$

For $\widetilde{X} = i$, we accept if

$$U \leq \frac{p_i}{Cq_i} = \frac{\frac{e^{-1}}{i!}}{e^{-1}} = \frac{1}{i!}.$$

Implementing this in Matlab, we have the following

Listing 3.9: Matlab Code

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

for i = 1:N
  X_tilde = floor(11*rand);
  while rand > 1/factorial(X_tilde)
    X_tilde = floor(11*rand);
  end
  X(i) = X_tilde;
end

hist(X,max(X)+1)
```

3.3.3 Multivariate acceptance-rejection

Acceptance-rejection extends easily to multi-dimensional densities. I will only describe the continuous setting, but note that everything also holds in the discrete setting.

Consider a multi-dimensional density $f : \mathbb{R}^d \to [0, \infty)$. Suppose we want to sample from this density, but do not know how to. Assume that there exists another d-dimensional density, g, that we can sample from easily (for example, the multivariate normal density, which we will discuss later) and that there exists a $C < \infty$ such that $Cg(\mathbf{x}) \geq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^d$. Then, we can use the same recipe as above. That is:

(i) Draw $\widetilde{\mathbf{X}} \sim g$.

(ii) Draw $U \sim \mathcal{U}(0,1)$. If

$$U \leq \frac{f(\widetilde{\mathbf{X}})}{Cg(\widetilde{\mathbf{X}})},$$

then return $\mathbf{X} = \widetilde{\mathbf{X}}$. Otherwise, go to step 1.

The proof that acceptance-rejection works in a multi-dimensional setting is essentially the same as in the standard continuous case. Just change the integrals to integrals over $\mathbb{R}^d$.

3.3.4 Drawing uniformly from complicated sets

One of the most useful applications of the acceptance-rejection method is to sampling uniformly from a complicated set. I will explain this for the continuous case (but the discrete case is completely analagous).

Suppose we want to draw uniformly from some set A, but are not able to do so easily. Suppose, however, that we can draw from a larger set, $\mathcal{X}$, with $A \subset \mathcal{X}$. In addition, we impose the condition that, if $\widetilde{X}$ is distributed uniformly on $\mathcal{X}$, then $\mathbb{P}(\widetilde{X} \in A) > 0$. Putting this in the framework of acceptance-rejection, observe that

$$f(x) = \frac{\mathbb{I}(x \in A)}{|A|} \propto \mathbb{I}(x \in A)$$

and

$$g(x) = \frac{\mathbb{I}(x \in \mathcal{X})}{|\mathcal{X}|} \propto \mathbb{I}(x \in \mathcal{X}).$$

Then, the optimal C is given by

$$C = \sup_{x \in \mathcal{X}} \frac{|\mathcal{X}|}{|A|} = \frac{|\mathcal{X}|}{|A|} > 1.$$

This is, of course, often hard to determine (as we may not know much about A). However, it turns out that we do not need C. Given $\widetilde{X}$ and $U \sim \mathcal{U}(0,1)$, we accept if

$$U \leq \frac{1}{C}\frac{f(\widetilde{X})}{g(\widetilde{X})} = \frac{|A|}{|\mathcal{X}|}\frac{\mathbb{I}(\widetilde{X} \in A)/|A|}{\mathbb{I}(\widetilde{X} \in \mathcal{X})/|\mathcal{X}|} = \mathbb{I}(\widetilde{X} \in A).$$

That is, we accept if $\widetilde{X} \in A$ and reject otherwise. Note that the acceptance probability is $|A|/|\mathcal{X}|$.

Example 3.3.6 (Sampling uniformly from a sphere)**.** An easy way to sample uniformly in the unit sphere $S = \{(x,y,z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 \leq 1\}$ (which is a little tricky using direct means) is by drawing uniformly from the cube $[-1,1]^3$ and then using acceptance-rejection. This can be done in Matlab as follows:

Listing 3.10: Matlab Code

```
N=10^3;
X = zeros(N,3);

for i = 1:N
```

```
  X_tilde = 2*rand(1,3) - 1;
  while(X_tilde(1)^2 + X_tilde(2)^2 + X_tilde(3)^2 > 1)
    X_tilde = 2*rand(1,3) - 1;
  end
  X(i,:) = X_tilde;
end

scatter3(X(:,1),X(:,2),X(:,3))
```

If we know the size of $\mathcal{X}$ and have a reasonable estimate, $\widehat{\alpha}$, of the acceptance probability (which we can get simply by recording the number of proposals we accept), we can use this to estimate the size of A. We have

$$\widehat{|A|} = \widehat{alpha}|X|.$$

This is very useful as it allow us to estimate the size of the set, which may be of great interest (for example, it could be the number of solutions to a problem or the number of combinations of something).

Example 3.3.7 (Pythagorean triples)**.** Suppose we want to estimate the number of Pythagorean triples where all values are under 100 (counting each permutation separately). That is, we want to estimate how many integer triples, $x, y, z \leq 100$ exist such that $x^2 + y^2 = z^2$. We can sample such triples using acceptance-rejection (drawing each of x,y and z uniformly at randomfrom $\{1, \ldots, 100\}$. We can then estimate the acceptance probability to get an estimate of the number of triples. This is implemented in Matlab as follows.

Listing 3.11: Matlab Code

```
N=10^3;
accept = zeros(N,1);

for i = 1:N
  X = ceil(100*rand);
  Y = ceil(100*rand);
  Z = ceil(100*rand);
  if((X^2 + Y^2) == Z^2)
    [X Y Z]
    accept(i) = 1;
  end
end

num_sol_hat = mean(accept) * 100^3.
```

3.3.5 The limitations of high-dimensional acceptance-rejection

Although acceptance-rejection is a very useful method, it unfortunately often does not perform well in high-dimensions. To see an example of this, consider the problem of sampling from a d-dimensional sphere by sampling uniformly from the hypercube $[-1, 1]^d$. The probability of acceptance, α, is given by

$$\alpha = \frac{|\{\mathbf{x} \in \mathbb{R}^d : \|x\| \leq 1\}|}{|[-1, 1]^d|} = \frac{\frac{\pi^{d/2}}{d/2\Gamma(d/2)}}{2^d} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)}.$$

This goes to zero very quickly as d gets larger. Essentially, the problem is that the error in an approximating a sphere by a box gets proportionally much bigger in higher dimensions. This limitation means that in high-dimensional settings, it is often necessary to consider an alternative method of generating random variables.

3.4 Location-Scale Families

The most efficient methods to generate random variables are usually very specific. That is, unlike inverse-transform and acceptance-rejection, these methods only work for a specific distribution. Often, these methods are even more specific: they only work for a given choice of parameters. For example, as we will see, some algorithms are specifically designed to generate $\mathsf{N}(0,1)$ random variables.

For this reason, the *location-scale* property of some distributions turns out to be very useful. Essentially, this property says that if a distribution is location-scale we can easily transform random variables generated with one parameter set to random variables generated with another parameter set.

Formally, a family of probability densities $\{f(x;\mu,\sigma)\}$ indexed by a location parameter μ and a scale parameter σ is said to be location-scale if

$$f(x;\mu,\sigma) = \frac{1}{\sigma} f\left(\frac{x-\mu}{\sigma};0,1\right) = \frac{1}{\sigma}\mathring{f}\left(\frac{x-\mu}{\sigma}\right),$$

where $\mathring{f}$ is the 'standard' form of the density with location parameter $\mu = 0$ and scale parameter $\sigma = 1$ (e.g., the standard normal density). Note that the location is not always the expected value and the scale is not always the standard deviation.

The usefulness of the location-scale property comes from the following lemma.

Lemma 3.4.1. Let $\{f(x;\mu,\sigma)\}$ be a location-scale family. Then, $Z \sim \mathring{f}$ implies $X = \mu + \sigma Z \sim f(x;\mu,\sigma)$.

Proof. We can check the cdf of X is as desired. We have

$$\mathbb{P}(X \leq x) = \mathbb{P}(\mu + \sigma Z \leq x) = \mathbb{P}\left(Z \leq \frac{x-\mu}{\sigma}\right) = \int_{-\infty}^{\frac{x-\mu}{\sigma}} \mathring{f}(z)\,\mathrm{d}z.$$

We can change variables from z to $u = \mu + \sigma z$ to get

$$\int_{-\infty}^{\frac{x-\mu}{\sigma}} \mathring{f}(z)\,\mathrm{d}z = \int_{\infty}^{x} \frac{1}{\sigma}\mathring{f}\left(\frac{-\mu}{\sigma}\right)\mathrm{d}u.$$

Now, using the location-scale property, we can write this as

$$\int_{-\infty}^{x} \frac{1}{\sigma}\mathring{f}\left(\frac{u-\mu}{\sigma}\right)\mathrm{d}u = \int_{-\infty}^{x} f(u;\mu,\sigma)\,\mathrm{d}u,$$

which is the desired cdf. □

Example 3.4.2 (The normal distribution)**.** Consider the normal distribution. This is location-scale with location parameter μ and scale parameter σ. We can see this by writing

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} = \frac{1}{\sigma}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{\left(\frac{x-\mu}{\sigma}\right)^2}{2}\right\} = \frac{1}{\sigma}\mathring{f}\left(\frac{x-\mu}{\sigma}\right),$$

where $\mathring{f}$ is the density of a $\mathsf{N}(0,1)$ random variable. This leads to the following recipe for producing $\mathsf{N}(\mu,\sigma)$ random variables:

(i) Simulate $Z \sim \mathsf{N}(0,1)$.

(ii) Return $X = \mu + \sigma Z$.

Example 3.4.3 (The uniform distribution on (a,b))**.** Consider the uniform distribution on (a,b). This is location-scale with location parameter $\mu = a$ and scale parameter $\sigma = (b-a)$. To see this, observe that the density is

$$f(x;\mu,\sigma) = \frac{\mathbb{I}\{x\in(a,b)\}}{|b-a|} = \frac{1}{|b-a|}\mathbb{I}\left\{\frac{x-a}{b-a}\in(0,1)\right\} = \frac{1}{\sigma}\mathring{f}\left(\frac{x-\mu}{\sigma}\right),$$

where $\mathring{f}$ is the density of a $\mathcal{U}(0,1)$ random variable. This leads to the following recipe for producing $\mathcal{U}(a,b)$ random variables:

(i) Simulate $U \sim \mathsf{U}(0,1)$.

(ii) Return $X = a + (b-a)U$.

Many useful distributions are location-scale, including:

- the normal distribution
- the student-t distribution,
- the Laplace distribution,
- the Cauchy distribution,
- the logistic distribution,
- the continuous uniform distribution.

3.5 Generating Normal Random Variables

The normal distribution is one of the most important probability distributions. For this reason, quite specialist methods have been developed for generating normal random variables. These methods usually produced standard normals (i.e., those with a $\mathsf{N}(0,1)$ distribution), which are then transformed using the location-scale property of the normal distribution.

In practice, normal random variables are usually generated via the inverse-transform method, using very accurate approximations of the inverse cdf of the standard normal distribution. Such methods are difficult to implement and should be left to the professionals (i.e., don't write your own normal random variable generator). Nevertheless, it is nice to look at a simple and elegant method that does exist for generating normal random variables.

3.5.1 Box-Muller method

The Box-Muller method is a very well known (but seldom used) method for generating normal random variables. It is based on the observation that, if Z_1 and Z_2 are standard normals, then the level sets of their joint density

$$f_{Z_1,Z_2}(z_1, z_2) = \frac{1}{2\pi}\exp\left\{-\frac{(z_1^2 + z_2^2)}{2}\right\}$$

form circles. Thus, working with polar coordinates, two independent standard normals can be produced by sampling an angle uniformly on $(0, 2\pi)$ then sampling a radius from the appropriate distribution.

The algorithm is as follows:

(i) Draw $\theta \sim \mathcal{U}(0, 2\pi)$.

(ii) Draw $R^2 \sim \mathsf{Exp}(1/2)$.

(iii) Return $Z_1 = \sqrt{R^2}\cos\theta$ and $Z_2 = \sqrt{R^2}\sin\theta$.

Lemma 3.5.1. The Box-Muller algorithm returns two independent $\mathsf{N}(0, 1)$ random variables.

Proof. To see this, we can rewrite the joint density of R and θ to get the density of Z_1 and Z_2 (which will be that of two independent normal random variables). As R and θ are independent, their joint density is the product of their marginals. Now,

$$f_\theta(\theta) = \frac{1}{2\pi}.$$

The pdf of R is a bit more complicated. Observe that $\mathbb{P}(R^2 \leq r^2) = 1 - e^{\frac{-r^2}{2}}$. As the radius, R, must be positive, we have that $\mathbb{P}(R \leq r) = \mathbb{P}(R^2 \leq r^2)$. To get the pdf of R we differentiate. This gives

$$f_R(r) = r\exp\left\{-\frac{r^2}{2}\right\}.$$

Thus, the joint density is

$$f_{\theta,R}(\theta, R) = \frac{1}{2\pi}\, r\exp\left\{-\frac{r^2}{2}\right\}.$$

Carrying out a standard change of variables from polar coordinates (with $R = \sqrt{Z_1^2 + Z_2^2}$), we get

$$f_{Z_1,Z_2}(z_1, z_2) = \frac{1}{2\pi}\exp\left\{-\frac{z_1^2 + z_2^2}{2}\right\},$$

which is the joint density of two independent standard normal random variables. $\square$

3.6 Big O notation

When discussing algorithms (e.g. for generating specific random variables) it is useful to be able to measure how much work they require, where work is, e.g., the number of uniform random variables generated or the number of iterations of the algorithm that are required. Often, this is easiest to describe in an asymptotic setting (for example as a parameter of the algorithm goes to zero or to infinity. One way to describe the way that a quantity grows in an asymptotic setting is using big O notation (sometimes called Langau notation).

We say $f(x) = O(g(x))$, or f is of *order* $g(x)$, if there exists $C > 0$ and $x_0 > 0$ such that

$$|f(x)| \leq Cg(x)$$

for all $x \geq x_0$ as $x \to \infty$ (or, sometimes, for all $x \leq x_0$ as $x \to 0$).

Example 3.6.1. The quadratic $x^2 + 3x + 1$ is $O(x^2)$, as, for $x \geq x_0 = 1$, we have $x^2 + 3x + 1 \leq x^2 + 3x^2 + x^2 = 5x^2$, so $x^2 + 3x + 1 \leq Cx^2 = Cg(x)$ where $C = 5$ and $g(x) = x^2$.

If we can break a function into a sum of other functions (e.g., $f(x) = x^2 + 3x = f_1(x) + f_2(x)$, where $f_1(x) = x^2$ and $f_2(x) = 3x$), then the order of f is the order of the component function with the biggest order. In our example, $f_1(x) = O(x^2)$ and $f_2(x) = O(x)$, so $f(x) = O(x^2)$.

We use big O notation to describe the behavior of algorithms and also the behavior of errors. For example, if we measure the dimension of a problem by n — for example, the number of items in a list that we have to sort — then the work done by an algorithm will typically be a function of n. Usually, we prefer algorithms with smaller growth rates for the work they have to do. For example, if algorithm 1 accomplishes a task with $O(n)$ work and algorithm 2 needs $O(n^2)$ work, then we will tend to prefer algorithm 1. However, if the work done by algorithm 1 is $f_1(n) = 10^6 n$ and the work done by algorithm 2 is $f_2(n) = n^2$, then the second algorithm is actually better if $n < 10^6$, even though its order is worse.

It is worth noting that $x^2 = O(x^2)$, but it is also true that $x^2 = O(x^3)$, so the equals sign is something of an abuse of notation.

Example 3.6.2 (The Standard Deviation of the Monte Carlo Estimator)**.** Consider the quantity

$$\ell = \int_{\mathbb{R}} S(x)\,\mathrm{d}x = \int_{\mathbb{R}} \frac{S(x)}{g(x)} g(x)\,\mathrm{d}x,$$

where $g(x)$ is a density such that $S(x) = 0 \Rightarrow g(x) = 0$. The standard estimator of this is

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} \frac{S(X_i)}{g(X_i)},$$

where the $\{X_i\}$ are iid random variables with density g. The standard deviation of this estimator (which is a rough measure of its error) is given by

$$\frac{1}{\sqrt{N}}\sqrt{\mathrm{Var}\left(\frac{S(X_1)}{g(X_1)}\right)} = O(N^{-1/2}).$$

This describes the rate at which the error gets smaller as the sample size gets larger.

3.7 Some useful tools from probability

3.7.1 Conditional expectation (discrete case)

Let X be a discrete random variable with state space $\mathcal{X}$ and Y be another random variable with the same state space. If $\mathbb{P}(Y = y) > 0$, we can define the conditional distribution of X

given the event $\{Y = y\}$by

$$\mathbb{P}(X = i \,|\, Y = y) = \frac{\mathbb{P}(X = i, Y = y)}{\mathbb{P}(Y = y)} \quad \text{for all } i \in \mathcal{X}.$$

We can then define the conditional expectation of X given $\{Y = y\}$ as the expectation of X with respect to the conditional distribution. That is,

$$\psi(y) = \mathbb{E}[X \,|\, Y = y] = \sum_{i\in\mathcal{X}} i \, \mathbb{P}(X = i \,|\, Y = y).$$

This is deterministic (i.e., it is a fixed number). However, we could replaced the deterministic argument y with the random variable Y. Then we get

$$\mathbb{E}[X \,|\, Y] = \psi(Y).$$

This is called the conditional expectation of X given Y. Note that this is no longer deterministic but is instead a random variable. Note that this is well defined because Y will only take values with non-zero probabilities (this would not be true if Y was continuous).

Example 3.7.1. Suppose someone flips two biased coins. These are modeled by Bernoulli random variables $X_1 \sim \mathsf{Ber}(p)$ and $X_2 \sim \mathsf{Ber}(p)$. Let Y be the sum of these two variables (i.e., $Y = X_1 + X_2$). Then, we can easily define the conditional distribution of Y given $X_1 = x$. For example,

$$\mathbb{P}(Y = 1 \,|\, X_1 = 0) = \frac{\mathbb{P}(Y = 1, X_1 = 0)}{\mathbb{P}(X_1 = 0)} = \frac{\mathbb{P}(X_2 = 1)\mathbb{P}(X_1 = 0)}{\mathbb{P}(X_1 = 0)} = \mathbb{P}(X_2 = 1) = p.$$

We can then work out the conditional expectations of Y given $\{X_1 = 0\}$ and $\{X_1 = 1\}$. We have

$$\begin{aligned}\mathbb{E}\left[Y \,|\, X_1 = 0\right] &= 2 \cdot \mathbb{P}(Y = 2 \,|\, X_1 = 0) + 1 \cdot \mathbb{P}(Y = 1 \,|\, X_1 = 0) + 0 \cdot \mathbb{P}(Y = 0 \,|\, X_1 = 0) \\ &= 2 \cdot 0 + 1 \cdot p + 0 \cdot (1 - p) = p.\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}\left[Y \,|\, X_1 = 1\right] &= 2 \cdot \mathbb{P}(Y = 2 \,|\, X_1 = 1) + 1 \cdot \mathbb{P}(Y = 1 \,|\, X_1 = 1) + 0 \cdot \mathbb{P}(Y = 0 \,|\, X_1 = 1) \\ &= 2 \cdot p + 1 \cdot (1 - p) + 0 \cdot (1 - p) = 2p + (1 - p) = 1 + p.\end{aligned}$$

The conditional expectation of Y given X_1 is

$$\mathbb{E}\left[Y \,|\, X_1\right] = p + X_1.$$

Lemma 3.7.2 (Tower property)**.** We have that $\mathbb{E}\left[\mathbb{E}\left[X \,|\, Y\right]\right] = \mathbb{E}X$.

Proof.

$$\begin{aligned}\mathbb{E}\left[\mathbb{E}\left[X \,|\, Y\right]\right] &= \sum_{\{y\,:\,\mathbb{P}(Y=y)>0\}} \mathbb{E}\left[X \,|\, Y = y\right] \mathbb{P}(Y = y) \\ &= \sum_{\{y\,:\,\mathbb{P}(Y=y)>0\}} \sum_{x\in\mathcal{X}} x \, \frac{\mathbb{P}(X = x, Y = y)}{\mathbb{P}(Y = y)} \mathbb{P}(Y = y) \\ &= \sum_{x\in\mathcal{X}} x \sum_{\{y\,:\,\mathbb{P}(Y=y)>0\}} \mathbb{P}(X = x, Y = y) \\ &= \sum_{x\in\mathcal{X}} x \, \mathbb{P}(X = x) = \mathbb{E}X.\end{aligned}$$

□

If we condition on a random variable, then we can treat that random variable as a constant inside the conditional expectation (and, like a constant, if possible we can pull it out of the expectation). This simplifies many calculations. There are are a number of results formalizing this. For example,

Lemma 3.7.3. $\mathbb{E}[XY \mid Y] = Y\mathbb{E}[X \mid Y]$.

3.7.2 Probability generating functions

Generating functions are tools in probability theory that are very helpful when trying to prove facts about the distributions of random variables. There are a number of different types of generating functions. *Probability generating functions* (pgfs) are used when the distribution of interest is discrete.

Let X be a discrete random variable. Then the probability generating function of X is given by

$$G(s) = \sum_{i \in \mathcal{X}} s^i \mathbb{P}(X = i), \quad s \in \mathbb{C}.$$

By taking derivatives of $G(s)$ and evaluating them at $s = 1$ we recover the so-called *factorial moments* of X.

Theorem 3.7.4. $G^{(k)}(1) = \mathbb{E}[X(X-1)\cdots(X-k+1)]$, where $G^{(k)}$ is the kth derivative of G.

Example 3.7.5. Geometric distribution Let X have pmf $\mathbb{P}(X = i) = p(1-p)^{i-1}$. Then,

$$\begin{aligned} G(s) = \mathbb{E}s^X &= \sum_{i=1}^{\infty} s^i (1-p)^{i-1} p \\ &= sp \sum_{i=1}^{\infty} s^{i-1}(1-p)^{i-1} \\ &= sp \sum_{j=0}^{\infty} (s(1-p))^j \\ &= \frac{sp}{1 - s(1-p)}. \end{aligned}$$

Note that this is only defined when $s(1-p) < 1$ (i.e., $s < (1-p)^{-1}$).

Example 3.7.6. Poisson distribution Let X have pmf $\mathbb{P}(X = i) = e^{-\lambda}\frac{\lambda^i}{i!}$. Then,

$$\begin{aligned} G(s) = \mathbb{E}s^X &= \sum_{i=0}^{\infty} s^i e^{-\lambda} \frac{\lambda^i}{i!} \\ &= \sum_{i=0}^{\infty} e^{-\lambda} \frac{(s\lambda)^i}{i!} \\ &= e^{\lambda(1-s)} \sum_{i=0}^{\infty} e^{-\lambda s} \frac{(s\lambda)^i}{i!} \\ &= e^{\lambda(1-s)}. \end{aligned}$$

Theorem 3.7.7. If $G_X = G_Y$ for all $s \in \mathbb{C}$ then X and Y have the same distribution.

3.7.3 Moment generating function

Let X be a random variable. The *moment generating function* (mgf) of X is given by $M(\theta) = \mathbb{E}e^{\theta X}$, where $\theta \in \mathbb{R}$. Thus, in the discrete case,

$$M(\theta) = \sum_{i \in \mathcal{X}} e^{\theta i} \mathbb{P}(X = i)$$

and, in the continuous case,

$$M(\theta) = \int_{\mathbb{R}} e^{\theta u} f(u)\, \mathrm{d}u.$$

Note that the moment generating function may not exist for all $\theta \in \mathbb{R}$ (as the expectation may not be finite).

Example 3.7.8 (Geometric distribution)**.** Let X have pmf $\mathbb{P}(X = i) = p(1-p)^{i-1}$. Then,

$$\begin{aligned} M(\theta) = \mathbb{E}e^{\theta X} &= \sum_{i=1}^{\infty} e^{\theta i}(1-p)^{i-1} p \\ &= p\, e^{\theta} \sum_{i=1}^{\infty} (e^{\theta}(1-p))^{i-1} \\ &= p\, e^{\theta} \sum_{j=0}^{\infty} (e^{\theta}(1-p))^{i} \\ &= \frac{pe^{\theta}}{1 - e^{\theta}(1-p)}. \end{aligned}$$

Note that this is only defined for $e^{\theta}(1-p) < 1$ (i.e., $\theta < -\log(1-p)$).

Example 3.7.9 (Bernoulli distribution)**.** Let X have a $\mathsf{Ber}(p)$ distribution. Then

$$M(\theta) = \mathbb{E}e^{\theta X} = e^{\theta}\mathbb{P}(X = 1) + e^{0}\mathbb{P}(X = 0) = pe^{\theta} + 1 - p.$$

As the name suggests, one can determine the moments of a distribution from its moment generating function.

Lemma 3.7.10. $M^{(k)}(0) = \mathbb{E}X^k$.

Theorem 3.7.11. If $M(\theta)$ exists in a neighborhood of 0 then $M(\theta)$ determines the distribution of X (and, thus, if two random variables have the same moment generating function, they have the same distribution).

Example 3.7.12 (Putting things together)**.** A popular bar offers a drink promotion where, every time someone buys a beer, a biased coin (where the probability of heads if p) is flipped. If the coin comes up heads, the customer pays for the beer. But, if the coin is tails, the customer gets the beer for free. If a Poisson (λ) number of people arrive in an hour, what is the distribution of the number of people who have to pay? This can be written in the form of a sum of a random number of random variables. More precisely, let X be the number of people paying for beer, N be the (random) number of people that arrive in an hour and $\{Z_i\}$ be a sequence of iid $\mathsf{Ber}(p)$ random variables. Then,

$$X = \sum_{i=1}^{N} Z_i$$

Considering the mgf of X, we have (by the tower property)

$$\mathbb{E}\exp\{\theta X\} = \mathbb{E}\left[\mathbb{E}\left[\exp\{\theta X\} \,|\, N\right]\right] = \mathbb{E}\left[\mathbb{E}\left[\exp\left\{\sum_{i=1}^{N} \theta Z_i\right\} \,|\, N\right]\right].$$

Now, because we are conditioning on N we can treat it as a constant within the conditional expectation and, thus, pull it out to get

$$\mathbb{E}\left[\mathbb{E}\left[\sum_{i=1}^{N} \exp\{Z_i\} \,|\, N\right]\right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{i=1}^{N} \exp\{\theta Z_i\}\right]\right] = \mathbb{E}\left[\left(\mathbb{E}\exp\{\theta Z_1\}\right)^N\right]$$

Note that the inner expectation is just the mgf of a Bernoulli random variable, which was calculated above. Thus, we have

$$\mathbb{E}\exp\{\theta X\} = \mathbb{E}\left[\left(pe^\theta + 1 - p\right)^N\right].$$

Observe that this is the formula for the generating function of a Poisson random variable (with $s = pe^\theta + 1 - p$). Thus,

$$\mathbb{E}\exp\{\theta X\} = \exp\left\{\lambda(1 - pe^\theta - 1 + p)\right\} = \exp\left\{p\lambda(e^\theta - 1)\right\}.$$

This is the mgf of a Poisson distribution with parameter λp. Thus the number of people that pay in an hour is a Poisson (λp) random variables.

3.8 Discrete Distributions

3.8.1 The geometric distribution

Consider a sequence of independent $\mathsf{Ber}(p)$ experiments (for example, a sequence of coin tosses). A geometric random variable gives the number of failure before a success occurs. There are two versions of this distribution. The first counts the success along with the failures. The pmf for this version is

$$\mathbb{P}(X = i) = p(1-p)^{i-1} \quad i \in \{1, 2, \ldots\}$$

The second only counts the failures. The pmf of this version is

$$\mathbb{P}(X = i) = p(1-p)^{i} \quad i \in \{0, 1, \ldots\}.$$

Note that the first can be transformed to the second by subtracting 1. Thus, we will only consider methods for generating the first distribution.

Often, a simple way to find an algorithm for generating a random variable is to think about what the random variable is describing and mimic that. In this case, the obvious algorithm is to simulate Bernoulli random variables until a success occurs and then return the number of variables we simulated. The algorithm is:

1. Set $n = 0$.
2. Simulate $Z \sim \mathsf{Ber}(p)$ and set $n = n + 1$.
3. If $Z = 1$, return $X = n$. Otherwise, go to step 2.

Now that we have some tools, we should think about how much work this algorithm needs to do. An obvious measure of work here is the number of random variables that needs to be generated. To make things easy, we will consider the expected number of these variables. Of course, this is just the expected value of a Geometric random variables, which is $1/p$. This means that

$$\mathsf{work} = O\left(\frac{1}{p}\right).$$

What does this mean? Essentially, it says that the work required grows at a rate that is inversely proportional to the success probability. So, the smaller p is, the more work we need to do. Imagine if $p = 10^{-6}$. In this case, we would need to simulate 10^6 random variables on average before we produce a geometric random variable. If we needed 10^{10} such random variables, this would require 10^{16} random variables in total, which corresponds to a lot of computing time.

A better algorithm might be one where the amount of work does not grow as p gets smaller. Note, however, that such an algorithm might be worse for values of p close to 1.

The work done by the following algorithm does not grow as a function of p:

1. Draw $Z \sim \mathsf{Exp}\left(-\log(1-p)\right)$.
2. Return $X = \lceil Z \rceil$.

What is the expected work done by this algorithm as a function of p? It does not grow, so

$$\mathsf{work} = O(1).$$

Note, however, that for p sufficiently close to 1, this algorithm might not be a better choice than the first one, because we need to do some more work in calculating logarithms.

Of course, we should check that this algorithm works.

Lemma 3.8.1. The above algorithm returns a geometric random variable with parameter p.

Proof. The easiest way to show this is to check the pmf of the variable we produce corresponds to the pmd of a geometric p random variable. Now, we have

$$\mathbb{P}(X = i) = \mathbb{P}\left(\lceil Z \rceil = i\right) = \mathbb{P}\left(i - 1 < Z \le i\right).$$

Using the representation of exponentials in terms of uniforms, we have

$$\begin{aligned}\mathbb{P}\left(i-1 < Z \le i\right) &= \mathbb{P}\left(i - 1 < \frac{-\log(U)}{-\log(1-p)} \le i\right) \\ &= \mathbb{P}\left(i \log(1-p) \le \log(U) < (i-1)\log(1-p)\right) \\ &= \mathbb{P}\left((1-p)^i \le U < (1-p)^{i-1}\right) \\ &= (1-p)^{i-1} - (1-p)^i = (1-(1-p))(1-p)^{i-1} = p(1-p)^{i-1}\end{aligned}$$

□

3.8.2 Binomial distribution

A binomial (n, p) random variable describes the number of successes in n independent $\mathsf{Ber}(p)$ experiments. The pmf is

$$\mathbb{P}(X = i) = \binom{n}{i} p^i (1-p)^{i-1}, \quad i \in \{0, \ldots, n\}.$$

The most obvious way to simulate such a random variable is simply to simulate n Bernoulli random variables and count the number of successes. The algorithm is

1. Set $m = 0$ and $s = 0$.
2. Simulate $Z \sim \mathsf{Ber}(p)$ and set $m = m + 1$.
3. If $Z = 1$ set $s = s + 1$.
4. If $m = n$ return $X = s$. Otherwise, go to step 2.

What is the work done here as a function of the distribution parameters n and p? This algorithm does not depend on p. But the amount of work clearly grows proportionally to n. Thus,

$$\mathsf{work} = O(n).$$

For large values of n, it makes sense to look for another algorithm. In the case where n is large but p is small, we can use geometric random variables to generate binomial random variables. Observe that the event $\{X = i\}$, where X is binomial (n, p), is equivalent to the event $\{\sum_{j=1}^{i} Z_j \leq n < \sum_{j=1}^{i+1} Z_j\}$, where the $\{Z_j\}$ are independent geometric random variables. This suggests the following algorithm

1. Set $m = 0$ and $s = 0$.
2. Draw $Z \sim$ geometric (p).
3. Set $m = m + Z$.
4. If $m > n$ return $X = s$. Otherwise, set $s = s + 1$ and go to step 2.

What is the work done here? It depends on both n and p. The expected number of geometric random variables we need to generate is one plus the expected value of the binomial random variable, which is np. Thus,

$$\mathsf{work} = O(np).$$

Given that $p \leq 1$, this is better than the work done by the first algorithm (ignoring the additional cost of generating the geometric random variables). For $p \ll 1$, the algorithm will be much more effective! For large n, one possibility is to use a normal approximation based on the central limit theorem. Let X_n be binomial (n, p). Then,

$$\frac{X_n - np}{\sqrt{np(1-p)}} \xrightarrow{D} Z,$$

as $n \to \infty$, where $Z \sim \mathsf{N}(0, 1)$. This suggests the following algorithm.

1. Generate $Z \sim \mathsf{N}(0, 1)$.
2. Return $X = \max\{0, \lfloor np + \sqrt{np(1-p)}Z \rfloor\}$.

Note that this method of generating binomial random variables is only approximate. For small n it may be quite incorrect. A rule of thumb for the binomial distribution is that it is roughly normal for $np > 5$ and $n(1-p) > 5$. But much larger values of n should be taken in order to be certain that things are correct. In practice, more sophisticated algorithms are often used for large n.

3.8.3 The Poisson distribution

The Poisson distribution is a natural model for the number of independent arrivals in a period of time (among many other things). The pmf of a $\mathsf{Poi}(\lambda)$ random variable is

$$\mathbb{P}(X = i) = e^{-\lambda}\frac{\lambda^i}{i!}, \quad i \in \{0, 1, \ldots\}.$$

One way to simulate a Poisson random variable is to exploit an intimate link between Poisson and exponential random variables.

Lemma 3.8.2. Let $\{Z_j\}$ be a sequence of iid $\mathsf{Exp}(\lambda)$ random variables and define

$$X = \max\left\{n : \sum_{j=1}^{n} Z_j \leq 1\right\}.$$

Then, $X \sim \mathsf{Poi}(\lambda)$.

Proof. We have that

$$\mathbb{P}(X = i) = \mathbb{P}\left(\sum_{j=1}^{i} Z_j \leq 1 < \sum_{j=1}^{i+1} Z_j\right) = \int_0^1 \mathbb{P}\left(Z_{i+1} \geq 1 - u \,\middle|\, \sum_{j=1}^{i} Z_j = u\right) f(u)\,\mathrm{d}u,$$

where f is the density of $\sum_{j=1}^{i} Z_j$. Now, we know that $\sum_{j=1}^{i} Z_j \sim \mathsf{Gam}(i, \lambda)$ is independent of $Z_i \sim \mathsf{Exp}(\lambda)$. Thus,

$$\begin{aligned}
&\int_0^1 \mathbb{P}\left(Z_{i+1} \geq 1 - u \,\middle|\, \sum_{j=1}^{i} Z_j = u\right) f(u)\,\mathrm{d}u \\
&= \int_0^1 e^{-\lambda(1-u)} \frac{\lambda^i u^{i-1} e^{-\lambda u}}{(i-1)!}\,\mathrm{d}u \\
&= \frac{e^{-\lambda}\lambda^i}{(i-1)!}\int_0^1 u^{i-1}\,\mathrm{d}u = \frac{e^{-\lambda}\lambda^i}{i!}
\end{aligned}$$

□

This suggests the following algorithm.

1. Set $s = 0$. Set $n = 0$.

2. Draw $Z \sim \mathsf{Exp}(\lambda)$.
3. Set $s = s + Z$.
4. If $s > 1$ then return n. Otherwise, set $n = n + 1$ and go to step 2.

The work done by this algorithm is proportional to the expected value of a $\mathsf{Poi}(\lambda)$ random variable. That is,

$$\mathsf{work} = O\left(\lambda\right).$$

As a result, this algorithm is not suitable for large values of λ. An alternative is the following.

1. Set $m = \lfloor 7/8\lambda \rfloor$.
2. Generate $Y \sim \mathsf{Gam}(m, 1)$.
3. If $Y \leq \lambda$, generate $Z \sim \mathsf{Poi}(\lambda - Y)$ and set $X = m + Z$. Otherwise, generate $X \sim \mathsf{Bin}(m-1, \lambda/Y)$.

Chapter 4

Variance Reduction

4.1 Measuring the Error of an Estimator

Because Monte Carlo methods are based on random numbers, the error of a given estimator will not be deterministic but rather random. As such, statements about the errors of Monte Carlo estimators are probabilistic.

When an estimator is unbiased (so its expected value is the correct answer), the most natural measure of its error is variance (or, equivalently, the easier to interpret standard deviation). There are a number of reasons why variance is a natural measure of error.

4.1.1 Asymptotic Error Bounds

Variance naturally occurs in one of the most fundamental theorems in probability: the central limit theorem. This tells us that the value of an empirical average of iid random variables is asymptotically normally distributed around the expect value of these variables.

Theorem 4.1.1 (Central Limit Theorem)**.** Let $\{X_i\}_{i=1}^\infty$ be a sequence of i.i.d. random values taking values in $\mathbb{R}$ such that $\mathbb{E}X_1 = \mu$ and $\mathrm{Var}(X_1) = \sigma^2$, with $0 < \sigma^2 < \infty$. Then, the random variables

$$Z_n = \frac{\sum_{i=1}^n X_i - n\mu}{\sigma\sqrt{N}}$$

converge in distribution to a random variable $Z \sim \mathsf{N}(0,1)$ as $n \to \infty$.

Thus, if we have an unbiased estimator of the form

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N},$$

where the $\{X_i\}$ are i.i.d. with $\mathbb{E}X_1 = \ell$ and $\mathrm{Var}(X_1) < \infty$, we can construct confidence intervals. For example, we know that

$$\mathbb{P}\left(\widehat{\ell} - 1.96\sqrt{\frac{\mathrm{Var}(X_1)}{N}} \leq \ell \leq \widehat{\ell} + 1.96\sqrt{\frac{\mathrm{Var}(X_1)}{N}}\right) \approx 0.95,$$

where the $\approx$ reflects the fact that $\widehat{\ell}$ is only approximately normal.

4.1.2 Non-Asymptotic Error Bounds

The central limit theorem is an asymptotic result. This means that it is not strictly speaking true that $\widehat{\ell}$ is normally distributed for finite N (though, of course, in practice its distribution is usually very close to a normal distribution). Non-asymptotic results allow us to say definite things about errors without making the approximations involved in asymptotic results. Of course, a cost of working in a non-asymptotic setting is that what we can say is usually a little less precise. The most famous non-asymptotic bounds in probability theory are Markov's inequality and its corollary, Chebyshev's inequality.

Theorem 4.1.2 (Markov's inequality)**.** Given a random variable, X, taking values in $\mathbb{R}$, a non-negative function $g : \mathbb{R} \to [0, \infty)$ and a constant $a > 0$, we have

$$\mathbb{P}(g(X) \geq a) \leq \frac{\mathbb{E}g(X)}{a}.$$

Proof. It is clear that $g(X) \geq a\mathbb{I}(g(X) \geq a)$, so $\mathbb{E}g(X) \geq \mathbb{E}a\mathbb{I}(g(X) \geq a) = a\mathbb{E}\mathbb{I}(g(X) \geq a) = a\mathbb{P}(g(X) \geq a)$. □

If we set $g(x) = (x - \mathbb{E}X)^2$ and $a = \epsilon^2$, where $\epsilon > 0$, we have

$$\mathbb{P}((X - \mathbb{E}X)^2 \geq \epsilon^2) \leq \frac{(X - \mathbb{E}X)^2}{\epsilon^2} \Rightarrow \mathbb{P}(|X - \mathbb{E}X| \geq \epsilon) \leq \frac{\mathrm{Var}(X)}{\epsilon^2}.$$

This is Chebyshev's inequality.

Theorem 4.1.3 (Chebyshev's inequality)**.** Given a random variable X taking values in $\mathbb{R}$ with $\mathrm{Var}(X) < \infty$ and $\epsilon > 0$, we have

$$\mathbb{P}(|X - \mathbb{E}X| \geq \epsilon) \leq \frac{\mathrm{Var}(X)}{\epsilon^2}.$$

Note that in both the asymptotic and non-asymptotic settings, we usually do not know the actual value of $\mathrm{Var}(X_1)$ as this would presumably require us to know $\mathbb{E}X_1$, which is the very quantity we are trying to estimate. Thus, we usually have to replace the above bounds with weaker ones (involving bounds on $\mathrm{Var}(X_1)$). In practice, we can usually estimate $\mathrm{Var}(X_1)$ and $\mathrm{Std}(X_1)$ using the empirical variance estimated from $\{X_i\}_{i=1}^N$ but care needs to be taken in doing so, as we will see later.

4.1.3 Estimating Probabilities

Now that we can generate a few different types of random variables, we can begin thinking about using Monte Carlo to answer some interesting questions. One thing that we often wish to estimate is the probability of a given event occurring. For example, we might want to know the probability that it rains tomorrow or the probability that a company goes bankrupt in the next 12 months. This can be done in a reasonably straightforward manner using Monte Carlo methods. As usual, the key to estimating such a quantity is to write it as an expectation. Observe that

$$\ell = \mathbb{P}(X \in A) = \mathbb{E}\mathbb{I}(X \in A).$$

Thus, we can estimate ℓ with the estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(X_i \in A),$$

where the $\{X_i\}$ are iid random variables with $X_1 \overset{D}{=} X$. The variance of such an estimator is

$$\text{Var}(\widehat{\ell}) = \frac{1}{N}\text{Var}(\mathbb{I}(X_i \in A)).$$

Now, observe that $\mathbb{I}(X_i \in A)$ is a Bernoulli random variable with success probability ℓ. Thus, $\text{Var}(\mathbb{I}(X_i \in A)) = \ell(1-\ell)$ and, as a result,

$$\text{Var}(\widehat{\ell}) = \frac{1}{N}\ell(1-\ell)$$

and

$$\text{Std}(\widehat{\ell}) = \frac{1}{\sqrt{N}}\sqrt{\ell(1-\ell)}.$$

Because ℓ is generally small, it is hard to interpret what the size of $\text{Std}(\widehat{\ell})$ means in terms of the error. For this reason, the standard deviation of the estimator is generally normalized by the quantity of interest. This gives the relative error

$$\mathsf{RE} = \frac{1}{\sqrt{N}}\frac{\sqrt{\ell(1-\ell)}}{\ell}.$$

Note that,

$$\mathsf{RE} = \frac{1}{\sqrt{N}}\frac{\sqrt{\ell(1-\ell)}}{\ell} \leq \frac{1}{\sqrt{N}}\frac{\sqrt{\ell}}{\ell} = \frac{1}{\sqrt{N}}\frac{1}{\sqrt{\ell}},$$

so, for fixed N,

$$\mathsf{RE} = O\left(\frac{1}{\sqrt{\ell}}\right)$$

and, for fixed ℓ,

$$\mathsf{RE} = O\left(\frac{1}{\sqrt{N}}\right).$$

If we fix the relative error (i.e., set $\mathsf{RE} = \epsilon$) we have

$$N = \frac{1}{\epsilon^2}\frac{\ell(1-\ell)}{\ell^2} = \frac{1}{\epsilon^2}\frac{1-\ell}{\ell} \leq \frac{1}{\epsilon^2}\frac{1}{\ell}.$$

So, as a function of ϵ, we have

$$N = O\left(\frac{1}{\epsilon^2}\right)$$

and, as a function of ℓ, we have

$$N = O\left(\frac{1}{\ell}\right).$$

Using the Chebyshev inequality, we obtain

$$\mathbb{P}\left(|\widehat{\ell} - \ell| > \epsilon\right) \leq \frac{\text{Var}(\widehat{\ell})}{\epsilon^2} = \frac{\ell(1-\ell)}{N\epsilon^2}.$$

This is not so meaningful, however, as we do not know *a priori* how small we want $|\widehat{\ell} - \ell|$ to be. A more meaningful bound (which we obtain using the Chebyshev inequality with constant $\ell\epsilon$) is

$$\mathbb{P}\left(\frac{|\widehat{\ell} - \ell|}{\ell} > \epsilon\right) \leq \frac{(1-\ell)}{N\epsilon^2\ell},$$

which allows us to estimate the probability that, say, the error is more than 10% of the true value of ℓ (this is done by setting $\epsilon = 0.1$).

Example 4.1.4 (Estimating an exceedance probability (and estimating the relative error))**.** Suppose we wish to estimate $\ell = \mathbb{P}(X > \gamma)$, where $\gamma \in \mathbb{N}$ and $X \sim \mathsf{Poi}(\lambda)$. We can do this as follows.

Listing 4.1: Matlab Code

```
N = 10^4;
lambda = 10; gamma = 16;
indics = zeros(N,1);

for i = 1:N
    X = poissrnd(lambda);
    indics(i) = (X > gamma);
end

ell = 1 - poisscdf(gamma, lambda)
ell_hat = mean(indics)
RE = sqrt(ell * (1 - ell)) / (ell * sqrt(N))
RE_hat = std(indics) / (ell_hat * sqrt(N))
```

4.2 More on Conditional Expectation and Conditional Variance

We here give a rough description of conditional expectation for continuous random variables. For more details on conditional probability and expectation, consult a good probability textbook.

Suppose X and Y are continuous $\mathbb{R}$-valued random variables with joint distribution $f(x,y)$. Then, for x such that $f(x) > 0$, we can define the conditional density of Y given $\{X = x\}$ by

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

Using the pdf, the conditional expectation of Y given $\{X = x\}$ is given by

$$\psi(x) = \mathbb{E}[Y \,|\, X = x] = \int_{\mathbb{R}} y f(y \,|\, X = x)\,\mathrm{d}y.$$

The conditional expectation of Y given X is then given by

$$\mathbb{E}[Y \,|\, X] = \psi(X).$$

Note that, as in the discrete case, the conditional expectation $\mathbb{E}[Y \mid X]$ is a random variable. It satisfies the tower property (which is a defining property of conditional expectation). That is,

$$\mathbb{E}\psi(X) = \mathbb{E}\left[\mathbb{E}\left[Y \mid X\right]\right] = \mathbb{E}Y.$$

Again, as in the discrete case, we can treat any X inside $\mathbb{E}[Y \mid X]$ as if it were a constant.

Using conditional expectation, we can define the conditional expectation of a random variable X given another random variable Y. This is given (analogous to the standard definition of variance) by

$$\mathrm{Var}(X \mid Y) = \mathbb{E}\left[X^2 \mid Y\right] - \mathbb{E}[X \mid Y]^2.$$

A key result about conditional variance is the following.

Theorem 4.2.1 (Law of Total Variance)**.** Let X and Y be two random variables on the same probability space. Then

$$\mathrm{Var}(X) = \mathbb{E}[\mathrm{Var}(X \mid Y)] + \mathrm{Var}\left(\mathbb{E}[X \mid Y]\right).$$

Proof. The easiest way to show this is to expand the right side. Observe that,

$$\begin{aligned}
&\mathbb{E}[\mathrm{Var}(X \mid Y)] + \mathrm{Var}\left(\mathbb{E}[X \mid Y]\right) \\
&= \mathbb{E}\left[\mathbb{E}[X^2 \mid Y]\right] - \mathbb{E}\left[\mathbb{E}[X \mid Y]^2\right] + \mathbb{E}\left[\mathbb{E}[X \mid Y]^2\right] - \mathbb{E}\left[\mathbb{E}[X \mid Y]\right]^2 \\
&= \mathbb{E}\left[\mathbb{E}[X^2 \mid Y]\right] - \mathbb{E}\left[\mathbb{E}[X \mid Y]\right]^2 \\
&= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\
&= \mathrm{Var}(X).
\end{aligned}$$

□

4.3 Conditional Monte Carlo

It is always desirable to maximize the accuracy of a given estimator (for a fixed computational cost). Doing so allows us to be more confident about the correctness of our result and it also, hopefully, allows us to use less samples (i.e. less computer time) to achieve a desired level of accuracy. As the above discussion demonstrates, in order to make Monte Carlo methods more accurate we should aim to find a Monte Carlo estimator with a smaller variance than the standard Monte Carlo estimator. The first variance reduction technique we will consider is called conditional Monte Carlo.

We know from the tower property that $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}X$. This suggests that, so long as we can generate realizations of the random variable Y and calculate $\mathbb{E}[X \mid Y]$, we can estimate

$$\ell = \mathbb{E}X$$

using the estimator

$$\widehat{\ell}_{\mathsf{cond}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[X \mid Y_i],$$

where the $\{Y_i\}_{i=1}^N$ are iid replicates of Y.

Why is this estimator better than the standard Monte Carlo estimator (assuming that we can easily generate Y and calculate $\mathbb{E}[X \mid Y]$)? To see this, first consider what the variance of $\widehat{\ell}_{\sf cond}$ looks like:

$$\mathrm{Var}\left(\widehat{\ell}_{\sf cond}\right) = \mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[X \mid Y_i]\right) = \frac{\mathrm{Var}\left(\mathbb{E}[X \mid Y]\right)}{N}$$

Now, observe that, by rearranging the law of total variance, we see that

$$\mathrm{Var}\left(\mathbb{E}[X \mid Y]\right) = \mathrm{Var}(X) - \mathbb{E}[\mathrm{Var}(X \mid Y)] \leq \mathrm{Var}(X).$$

As the variance of the standard Monte Carlo estimator is given by

$$\mathrm{Var}\left(\widehat{\ell}\right) = \frac{\mathrm{Var}(X)}{N}.$$

Thus, $\mathrm{Var}(\widehat{\ell}_{\sf cond}) \leq \mathrm{Var}(\widehat{\ell})$.

Example 4.3.1. Suppose you want to estimate $\ell = \mathbb{P}(X_1 + X_2 > \gamma)$ where $X_1 \sim \mathsf{Exp}(\lambda_1)$ and $X_2 \sim \mathsf{Exp}(\lambda_2)$. Now, observe that

$$\ell = \mathbb{E}\mathbb{I}(X_1 + X_2 > \gamma) = \mathbb{E}\mathbb{I}(X_2 > \gamma - X_1).$$

Thus, we can consider the conditional expectation

$$\mathbb{E}\left[\mathbb{I}(X_2 > \gamma - X_1) \mid X_1\right] = \begin{cases} 1 & \text{if } \gamma - X_1 < 0, \\ e^{-\lambda_2(\gamma - X_2)} & \text{otherwise.} \end{cases}$$

We then estimate $\mathbb{E}\mathbb{E}\left[\mathbb{I}(X_2 > \gamma - X_1) \mid X_1\right]$ by generating realizations of $X_1 \sim \mathsf{Exp}(\lambda_1)$ and averaging over the results. This is implemented in the following Matlab code (along with the conditional estimator given X_2 and the standard Monte Carlo estimator).

Listing 4.2: Matlab Code

```matlab
N = 10^6; gamma = 5;
results = zeros(N,1);
results_cond1 = zeros(N,1);
results_cond2 = zeros(N,1);

lambda_1 = 1;
lambda_2 = 2;

for i = 1:N
    X_1 = -log(rand) / lambda_1;
    X_2 = -log(rand) / lambda_2;

    results(i) = (X_1 + X_2 > gamma);

    if (gamma - X_1 < 0)
        results_cond1(i) = 1;
    else
        results_cond1(i) = exp(-lambda_2*(gamma-X_1));
    end

    if (gamma - X_2 < 0)
        results_cond2(i) = 1;
    else
        results_cond2(i) = exp(-lambda_1*(gamma-X_2));
    end
end

clc
std_ell_hat = mean(results)
std_var_hat = var(results) / N
cond1_ell_hat = mean(results_cond1)
cond1_var_hat = var(results_cond1) / N
cond2_ell_hat = mean(results_cond2)
cond2_var_hat = var(results_cond2) / N
```

A sample run of the above estimator with $\lambda_1 = 1$ and $\lambda_2 = 2$ gives the following results (where $\widehat{\ell}_1$ is the conditional estimator given X_1 and $\widehat{\ell}_2$ is the conditional estimator given X_2).

$$\widehat{\text{Var}}(\widehat{\ell}) = 1.33 \times 10^{-8} \quad \widehat{\text{Var}}(\widehat{\ell}_1) = 8.90 \times 10^{-9} \quad \widehat{\text{Var}}(\widehat{\ell}_2) = 3.07 \times 10^{-10}.$$

Notice that the variance of the estimator conditioned on X_2 is much smaller than the variance of the other conditional estimator. It is worth thinking about why this might be. Note that, because $\lambda_1 < \lambda_2$, X_1 will generally be larger than X_2. Thus, by only considering the 'randomness' coming from X_2 (which is what we do when we condition on X_2) we remove most of the randomness from the system – and, thus, the variance should be smaller.

More precisely, observe that in order to minimize the variance of the conditional estimator we want to condition on a Y such that

$$\mathbb{E}[\text{Var}(X \mid Y)] = \mathbb{E}X^2 - \mathbb{E}[\mathbb{E}[X \mid Y]^2] \leq \text{Var}(X)$$

is maximized. Because $\mathbb{E}X^2 - \mathbb{E}[\mathbb{E}[X \mid Y]^2] \leq \text{Var}(X)$, we know that $\mathbb{E}[\mathbb{E}[X \mid Y]^2] \leq \mathbb{E}X^2$. Thus, if we can find a Y such that $\mathbb{E}[\mathbb{E}[X \mid Y]^2] \approx \mathbb{E}X^2$, we will have almost perfect variance reduction. One way to satisfy this condition is to take Y independent of X. Although this seems a bit strange it actually makes a lot of sense – what it means is that we know how to calculate $\mathbb{E}[X \,|Y] = \mathbb{E}X$ (i.e. we know the answer already). A general rule of thumb is to try to condition on the variable that gives the least information about X while still allowing us to calculate the conditional expectation.

In summary, conditional Monte Carlo is a powerful method which can theoretically result in estimators with variance arbitrarily close to zero. However, it suffers from some limitations. In particular, one must be able to easily generate replicates of Y and then calculate $\mathbb{E}[X \mid Y]$. Note that, in some settings, it might still be worthwhile to use conditional Monte Carlo when $\mathbb{E}[X \mid Y]$ needs to be calculated numerically.

4.4 Antithetic sampling

Antithetic sampling is a simple technique that can often reduce the variance of an estimator by a non-trivial amount. The basic idea is to try to ensure that the samples spread out sufficiently over the sample space. This is done by generating negatively correlated (antithetic) pairs of random variables.

Suppose, again, that we want to estimate

$$\ell = \mathbb{E}X.$$

In antithetic sampling, instead of generating replicates of X, we generate pairs of correlated random variables, $Z^{(1)}, Z^{(2)}$, such that $Z^{(1)} \overset{D}{=} Z^{(2)} \overset{D}{=} X$ (i.e., so that both $Z^{(1)}$ and $Z^{(2)}$ have the same marginal distribution as X). We then use the estimator

$$\widehat{\ell}_{\mathsf{anti}} = \frac{1}{M}\sum_{i=1}^{M} \frac{Z_i^{(1)} + Z^{(2)}}{2},$$

where $(Z_1^{(1)}, Z_1^{(2)}), \ldots, (Z_m^{(1)}, Z_M^{(2)})$ are iid replicates of $(Z^{(1)}, Z^{(2)})$. This is clearly an unbiased estimator of ℓ.

In order to do this, we need to be able to generate $Z^{(1)}$ and $Z^{(2)}$ such that they are negatively correlated and have the correct marginals. This can often be done using the inverse transform method. Recall that, if $U \sim \mathcal{U}(0,1)$, then $1-U$ is also $\mathcal{U}(0,1)$. Thus, if F is the cdf of X, then $F^{-1}(U)$ and $F^{-1}(1-U)$ both produce variables with distribution F. They are also clearly negatively correlated (though it can be difficult to calculate this correlation exactly).

What is the variance of this estimator? If we choose $M = N/2$ (so that we use the same number of samples as a standard MC estimator with sample size N), then (using the fact that $Z^{(1)}$ and $Z^{(2)}$ have the same marginal distribution as X)

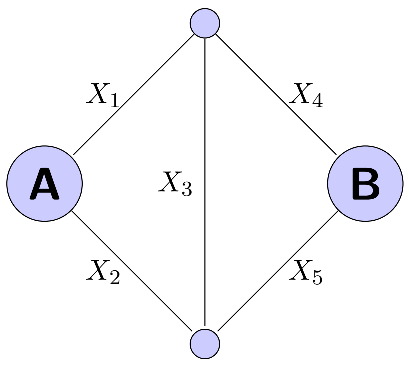
$$\begin{aligned}
\mathrm{Var}\left(\widehat{\ell}_{\mathsf{anti}}\right) &= \frac{1}{(N/2)^2}\sum_{i=1}^{N/2}\frac{1}{4}\mathrm{Var}\left(Z^{(1)} + Z^{(2)}\right) \\
&= \frac{2}{N}\frac{1}{4}\left[\mathrm{Var}(Z^{(1)}) - 2\mathrm{Cov}(Z^{(1)}, Z^{(2)}) + \mathrm{Var}(Z^{(2)})\right] \\
&= \frac{1}{N}\frac{1}{2}\left[\mathrm{Var}(X) - 2\rho\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(X)} + \mathrm{Var}(X)\right] \\
&= \frac{1}{N}(1-\rho)\mathrm{Var}(X),
\end{aligned}$$

where $\rho \in [-1,1]$ is the correlation coefficient. Thus, if the correlation is negative, the variance will be smaller than that of the standard Monte Carlo estimator.

Example 4.4.1 (Expected length of the shortest path)**.**
Consider the network illustrated in Figure 4.4.1. In this network, the 'length' of every edge is given by random variable (note these lengths do not have a geometric interpretation). These 'lengths' could, for example, represent the lifetimes of components or the length of time taken for certain industrial processes. Suppose we want to find the shortest path between A and B when $X_1, \ldots, X_5$ are iid $\mathcal{U}(0,1)$ random variables. This is given by

$$\ell = \mathbb{E}X = \mathbb{E}H(X_1, \ldots, X_5) = \min\{X_1 + X_4, X_1 + X_3 + X_5, X_2 + X_3 + X_4, X_2 + X_5\}.$$

We can generate an antithetic version of X using $H(1 - X_1, \ldots, 1 - X_5)$.



Listing 4.3: Matlab Code

```
N = 10^5;
M = N / 2;

results = zeros(N,1);
results_anti = zeros(M,1);
Zs = zeros(M,2);

for i = 1:N
    U = rand(1,5);
    Paths = [U(1) + U(4), U(1) + U(3) + U(5), ...
        U(2) + U(3) + U(4), U(2) + U(5)];
    results(i) = min(Paths);
end

for i = 1:M
    U = rand(1,5);
    Paths = [U(1) + U(4), U(1) + U(3) + U(5), ...
        U(2) + U(3) + U(4), U(2) + U(5)];
    Z_1 = min(Paths);
    U = 1 - U;
    Paths_2 = [U(1) + U(4), U(1) + U(3) + U(5), ...
        U(2) + U(3) + U(4), U(2) + U(5)];
    Z_2 = min(Paths_2);
    results_anti(i) = (Z_1 + Z_2) / 2;
    Zs(i,:) = [Z_1 Z_2];
end

ell_hat = mean(results)
std_hat = std(results)/sqrt(N)
ell_hat_anti = mean(results_anti)
std_hat_anti = std(results_anti)/sqrt(M)
cor_matrix = corr(Zs);
rho = cor_matrix(1,2)
```

4.5 Stratified Sampling

Stratified sampling is an extension of the idea of anithetic sampling in the sense that it seeks to ensure that the samples are spread appropriately across the state space. The idea is as follows. Suppose we can wish to estimate $\ell = \mathbb{E}S(X)$ where $S(\cdot)$ is some function and X is a random variable / vector with a straightforward distribution (most problems can be written in this form with a little work). Further suppose that we can divide the range of X into disjoint sets $A_1, \ldots, A_K$ (for example, if $X \sim \mathcal{U}(0,1)$ we could divides its range into $A_1 = (0, .5]$ and $A_2 = (.5, 1)$). These sets are called strata. Then, we can write

$$\ell = \mathbb{E}S(X) = \sum_{i=1}^{K} \mathbb{E}[S(X) \mid X \in A_i]\mathbb{P}(X \in A_i).$$

We can estimate this by

$$\widehat{\ell}_{\mathsf{strat}} = \sum_{i=1}^{K} p_i \widehat{\ell}^{(i)},$$

where $p_i = \mathbb{P}(X \in A_i)$ and $\widehat{\ell}$ is the standard estimator of $\ell_i = \mathbb{E}[S(X) \mid X \in A_i]$. That is,

$$\widehat{\ell}^{(i)} = \frac{1}{N_i} \sum_{j=1}^{N_i} S(X_j^{(i)}),$$

where $\{X_j^{(i)}\}$ is a sequence of iid random variables drawn from the conditional distribution of X given that $X \in A_i$. Thus, in full one can write the estimator in the form

$$\widehat{\ell}_{\mathsf{strat}} = \sum_{i=1}^{K} p_i \frac{1}{N_i} \sum_{j=1}^{N_i} S(X_j^{(i)}).$$

Assuming we have a division of the range of X into sets $A_1, \ldots, A_K$, we should try to choose the sample sizes, $N_1, \ldots, N_K$, in order to minimize the variance. How can we do this?

In order to decide, we first need to calculate the variance of the stratified sampling estimator. This is given by

$$\begin{aligned}\operatorname{Var}\left(\widehat{\ell}_{\mathsf{strat}}\right) &= \operatorname{Var}\left(\sum_{i=1}^{K} p_i \frac{1}{N_i} \sum_{j=1}^{N_i} S(X_j^{(i)})\right) \\ &= \sum_{i=1}^{K} p_i^2 \frac{1}{N_i} \operatorname{Var}\left(S(X^{(i)})\right) = \sum_{i=1}^{K} p_i^2 \frac{\sigma_i^2}{N_i},\end{aligned}$$

where $\sigma_i^2 = \operatorname{Var}(X^{(i)}$.

4.5.1 Proportional allocation

The first strategy we consider is to allocate samples to the ith strata proportional to the probability that X is in A_i. That is, we take

$$N_i \approx p_i N,$$

where N is the total sample size. What is the variance in this case?

We know that

$$\operatorname{Var}\left(\widehat{\ell}_{\mathsf{strat}}\right) = \sum_{i=1}^{K} p_i^2 \frac{\sigma_i^2}{N_i}.$$

So, plugging in $N_i = p_i N$, we obtain

$$\operatorname{Var}\left(\widehat{\ell}_{\mathsf{strat}}\right) = \sum_{i=1}^{K} p_i^2 \frac{\sigma_i^2}{p_i N} = \sum_{i=1}^{K} p_i \frac{\sigma_i^2}{N}.$$

Now, we need to compare this to the variance of the standard MC estimator. In order to do this, we need to rewrite $\operatorname{Var}(\widehat{\ell})$ in terms of the p_i and A_i. Observing that

$$\operatorname{Var}(\widehat{\ell}) = \frac{\mathbb{E}S(X)^2 - (\mathbb{E}S(X))^2}{N}.$$

Now,

$$\mathbb{E}S(X)^2 = \sum_{i=1}^{K} p_i \mathbb{E}[S(X)^2 \,|\, X \in A_i] = \sum_{i=1}^{K} p_i(\sigma_i^2 + \ell_i^2) = \sum_{i=1}^{K} p_i\sigma_i^2 + \sum_{i=1}^{K} p_i \ell_i^2$$

and

$$\mathbb{E}S(X) = \sum_{i=1}^{K} p_i \mathbb{E}[S(X) \,|\, Y] = \sum_{i=1}^{K} p_i \ell_i.$$

So,

$$\operatorname{Var}(\widehat{\ell}) = \frac{\mathbb{E}S(X)^2 - (\mathbb{E}S(X))^2}{N} = \frac{\sum_{i=1}^{K} p_i\sigma_i^2 + \sum_{i=1}^{K} p_i\ell_i^2 - \left(\sum_{i=1}^{K} p_i\ell_i\right)^2}{N}.$$

Observe that this differs from $\operatorname{Var}(\widehat{\ell}_{\mathsf{strat}})$ by $\sum_{i=1}^{K} p_i\ell_i^2 - (\sum_{i=1}^{K} p_i\ell_i)^2$. So, if we can show that

$$\sum_{i=1}^{K} p_i\ell_i^2 \le (\sum_{i=1}^{K} p_i\ell_i)^2$$

then we have shown that $\operatorname{Var}(\widehat{\ell}_{\mathsf{strat}}) \le \operatorname{Var}(\widehat{\ell})$. In order to show this, we simply apply Jensen's inequality. Thus, proportional sampling results in an estimator that always has a variance less than or equal to the variance of the standard Monte Carlo estimator. This is a very useful result, because it is often very straightforward to apply proportional sampling. Note that, in general, $p_i N$ will not be an integer. However, rounding these values will not make a significant different if N is sufficiently large.

4.5.2 Optimal allocation

In order to determine the optimal allocation of samples to each strata, we set $N_i = q_i N$ and again consider the variance of the stratified estimator, which is given by

$$\operatorname{Var}(\widehat{\ell}_{\mathsf{strat}}) = \sum_{i=1}^{K} \frac{p_i^2\sigma_i^2}{N} = \sum_{i=1}^{K} \frac{p_i^2\sigma_i^2}{q_i^2 N}.$$

We can optimize this, under the constraint that $\sum_{i=1}^K q_i = 1$ by solving

$$\frac{\partial}{\partial q_i}\left[\sum_{i=1}^K \frac{p_i^2\sigma_i^2}{q_i^2 N} + \lambda\left(\sum_{i=1}^K q_i = 1\right)\right] = 0$$

for $q_1, \ldots, q_K$ and

$$\frac{\partial}{\partial \lambda}\left[\sum_{i=1}^K \frac{p_i^2\sigma_i^2}{q_i^2 N} + \lambda(\sum_{i=1}^K q_i = 1)\right] = 0.$$

This gives

$$q_i^* = \frac{p_i\sigma_i}{\sum_{i=1}^K p_i\sigma_i} \qquad \text{for } i = 1, \ldots, K,$$

which can be seen to be a minimize by checking second order conditions. Thus, the optimal allocation is to allocate the sample to the ith strata proportional to its variance times the probability that X takes a value in this strata. Of course, to use this allocation we need to know the conditional variances for each strata. In general, finding these is a more difficult problem than the original problem, so the optimal allocation is not useable in practice (though it does give a good indication of what an efficient allocation should look like).

What is the variance of the optimal allocation scheme? Plugging in for N_i, we see that

$$\mathrm{Var}(\widehat{\ell}_{\mathsf{strat}}) = \sum_{i=1}^K \frac{p_i^2\sigma_i^2}{Np_i\sigma_i/\sum_{j=1}^K p_j\sigma_j} = \frac{(\sum_{i=1}^K p_i\sigma_i)(\sum_{j=1}^K p_j\sigma_j)}{N} = \frac{(\sum_{i=1}^K p_i\sigma_i)^2}{N}.$$

As expected, this is bounded above by the variance of the proportional allocation scheme, as

$$\frac{(\sum_{i=1}^K p_i\sigma_i)^2}{N} \leq \sum_{i=1}^K p_i\frac{\sigma_i^2}{N}$$

by Jensen's inequality. Note that, in general, $p_i\sigma_i N$ will not be an integer. However, rounding these values will not make a significant different if N is sufficiently large.

4.6 Control variates

Recall that we saw, in the case of conditional Monte Carlo, that we reduce the variance of an estimator by reducing the randomness as much as possible. This is the basic idea of control variates, where we try to use as much information as possible about the quantity we want to estimate, $\ell = \mathbb{E}S(X)$. If we use this information properly, then the difference between what this information tells us and what we wish to estimate will hopefully be small (and thus have a small variance).

In order to use control variances, we write

$$\ell = \mathbb{E}S(X) = \mathbb{E}S(X) - \alpha\mathbb{E}H(X) + \alpha\mathbb{E}H(X) = \mathbb{E}[S(X) - \alpha H(X)] + \alpha h,$$

where $H(X)$ is the control variate, which is chosen such that $h = \mathbb{E}H(X)$ is a quantity that we know. This gives the estimator

$$\widehat{\ell}_{\mathsf{cont}} = \frac{1}{N}\sum_{i=1}^N [S(X_i) - \alpha H(X_i)] + \alpha h,$$

where the $\{X_i\}$ are iid replicates of X.

In order to use control variates effectively, we need to choose an appropriate value of α. As usual, we do this by writing out the variance of our estimator and taking derivatives. The variance is given by

$$\mathrm{Var}(\widehat{\ell}_{\mathsf{cont}}) = \frac{\mathrm{Var}(S(X) - \alpha H(X))}{N} = \frac{\mathrm{Var}(S(X)) - 2\alpha \mathrm{Cov}(S(X), H(X)) + \alpha^2 \mathrm{Var}(H(X))}{N}.$$

Solving for

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} \frac{\mathrm{Var}(S(X)) - 2\alpha \mathrm{Cov}(S(X), H(X)) + \alpha^2 \mathrm{Var}(H(X))}{N} = 0,$$

we find that the optimal choice of α for minimizing the variance (where second order conditions show that it is indeed a minimizer) is

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

Note that this is the same as the regression coefficient that would be obtained if we tried to describe $S(X)$ using the simple linear regression model $S(X) = \alpha_0 + \alpha^* H(X) + \epsilon$.

Plugging α^* into the variance formula, we see that (using the optimal α)

$$\begin{aligned}
&N\mathrm{Var}(\widehat{\ell}_{\mathsf{cont}}) \\
&= \left[\mathrm{Var}(S(X)) - 2\frac{\mathrm{Cov}(S(X), H(X))}{\mathrm{Var}(H(X))}\mathrm{Cov}(S(X), H(X)) + \frac{\mathrm{Cov}(S(X), H(X))^2}{\mathrm{Var}(H(X))^2}\mathrm{Var}(H(X))\right] \\
&= \left[\mathrm{Var}(S(X)) - 2\frac{\mathrm{Cov}(S(X), H(X))^2}{\mathrm{Var}(H(X))} + \frac{\mathrm{Cov}(S(X), H(X))^2}{\mathrm{Var}(H(X))}\right] \\
&= \left[\mathrm{Var}(S(X)) - \frac{\mathrm{Cov}(S(X), H(X))^2}{\mathrm{Var}(H(X))}\right] \\
&= N\mathrm{Var}(S(X))(1 - \rho^2).
\end{aligned}$$

Thus, the variance of the control variates estimator (assuming the optimal α is used) will be less than the variance of the standard Monte Carlo estimator so long as ρ, the correlation between $S(X)$ and $H(X)$ is not equal to zero.

Note that (as always seems to be the case with variance reduction) we do not know usually know α^*, as it requires us to know the covariance of $H(X)$ and $S(X)$, which is typically hard to calculate than $\mathbb{E}S(X)$. However, we can obtain a rough estimate of α^* by simulating a small number of values of $H(X)$ and $S(X)$ and estimating their covariance. We then carry out the main simulation using new samples.

Example 4.6.1 (A simple integration example)**.** Suppose we wish to estimate

$$\ell = \int_0^\infty e^{-x^2}\,\mathrm{d}x = \mathbb{E}S(X),$$

where $S(X) = e^{-X^2}$ and $X \sim \mathcal{U}(0,1)$. A good choice of control variate is $H(X) = e^{-X}$ as we know that

$$h = \mathbb{E}H(X) = \int_0^\infty e^{-x}\,\mathrm{d}x = 1 - e^{-1}.$$

The control variate approach is easily implemented in Matlab.

Listing 4.4: Matlab Code

```
N_init = 10^3;
N_ctrl = 10^5;
N = N_init + N_ctrl;

SH = zeros(N_init,2);
results_ctrl = zeros(N_ctrl,1);
results = zeros(N,1);

for i = 1:N_init
    X = rand;
    SH(i,1) = exp(-X^2);
    SH(i,2) = exp(-X);
end

cov_mat = cov(SH);
alpha_star_hat = cov_mat(1,2) / cov_mat(2,2);

for i = 1:N_ctrl
    X = rand;
    results_ctrl(i) = exp(-X^2) - alpha_star_hat * exp(-X);
end

for i = 1:N
    X = rand;
    results(i) = exp(-X^2);
end

ell_hat_ctrl = mean(results_ctrl) + 1 - exp(-1)
std_hat_ctrl = std(results_ctrl) / sqrt(N_ctrl)
ell_hat = mean(results)
std_hat = std(results) / sqrt(N)
```

Chapter 5

Importance Sampling

Importance sampling is the most powerful of the general variance reduction methods. However, this power comes at a cost (which is that things can go quite wrong), so care should be exercised when using it.

The easiest way to introduce importance sampling is in the context of rare events. Suppose we want to estimate

$$\ell = \mathbb{P}_f(X \in A),$$

where X is an $\mathbb{R}$-valued random variable with density f, $A \subset \mathbb{R}$, and $\mathbb{P}_f$ is used to denote that the probability is calculated assuming $X \sim f$. Furthermore, suppose that ℓ is small (so the event $\{X \in A\}$ is rare). In this case, the relative error of the standard Monte Carlo estimator will be quite high because many samples are required in order to observe enough occurrences of $\{X \in A\}$. The idea of importance sampling in this context is to reweight the probabilities of certain events happening so that $\{X \in A\}$ (which is the important event) happens much more often. Obviously, this needs to be done properly or the resulting estimator will be biased. This is done as follows. Suppose we can find a density g such that, when X has density g the event $\{X \in A\}$ is more likely and such that $g(x) = 0 \Rightarrow f(x)\mathbb{I}(X \in A) = 0$. Then, we can write

$$\begin{aligned}\ell = \mathbb{E}_f \mathbb{I}(X \in A) &= \int_{\mathbb{R}} \mathbb{I}(x \in A) f(x)\,\mathrm{d}x \\ &= \int_{\mathbb{R}} \frac{f(x)}{g(x)} g(x) \mathbb{I}(x \in A)\,\mathrm{d}x = \mathbb{E}_g \frac{f(X)}{g(X)} \mathbb{I}(X \in A),\end{aligned}$$

where $\mathbb{E}_f$ is calculated with $X \sim f$ and $\mathbb{E}_g$ is calculated with $X \sim g$. The term $f(X)/g(X)$, which reweights things so that everything is unbiased, is called the likelihood ratio.

Using this result, we obtain the estimator

$$\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{g(X_i)} \mathbb{I}(X \in A),$$

where the $\{X_i\}$ are iid random variables with density g.

Example 5.0.2. Recall that the basic idea is to choose g such that $\{X \in A\}$ is more likely. For example, suppose we want to estimate

$$\ell = \mathbb{P}(X \in [5, \infty)),$$

where $X \sim \mathsf{Exp}(1)$. We could make the event of interest more likely by choosing g to be the density of a $\mathsf{Exp}(1/5)$ random variable (so that $\mathbb{E}_g X = 5$). Suppose we have $f(x) = e^{-x}$ and $g(x) = \lambda e^{-\lambda x}$, then

$$\frac{f(x)}{g(x)} = \frac{1}{\lambda}\exp\{(\lambda - 1)x\}.$$

This approach is implemented in Matlab for the general problem of estimating $\ell = \mathbb{P}(X > \gamma)$, using an exponential distribution with $\lambda = 1/\gamma$.

Listing 5.1: Matlab Code

```
N = 10^5; gamma = 5;

lambda = 1/gamma;
results = zeros(N,1);
results_IS = zeros(N,1);

for i = 1:N
    X = -log(rand);
    X_IS = -log(rand) / lambda;
    results(i) = (X > gamma);
    results_IS(i) = exp((lambda-1)*X_IS) / lambda * (X_IS > gamma);
end

ell = exp(-5)
ell_hat = mean(results)
re_hat = std(results) / (sqrt(N) * ell_hat)
ell_hat_IS = mean(results_IS)
re_hat_IS = std(results_IS) / (sqrt(N) * ell_hat_IS)
re_hat / re_hat_IS
```

More generally, we can use importance sampling to estimate any problem of the form

$$\ell = \mathbb{E}_f S(X),$$

using an estimator of the form

$$\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{g(X_i)} S(X_i),$$

so long as $g(x) = 0 \Rightarrow f(x)S(x) = 0$.

5.1 The variance of the importance sampling estimator

What can we say about the variance of the importance sampling estimator? We have that

$$\mathrm{Var}_g\left(\widehat{\ell}_{\mathsf{IS}}\right) = \mathrm{Var}_g\left(\frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{g(X_i)} S(X_i)\right) = \frac{1}{N}\mathrm{Var}_g\left(\frac{f(X)}{g(X)} S(X)\right).$$

Now,

$$\mathrm{Var}_g\left(\frac{f(X)}{g(X)}S(X)\right) = \mathbb{E}_g\left(\frac{f(X)}{g(X)}S(X)\right)^2 - \left(\mathbb{E}_g\frac{f(X)}{g(X)}S(X)\right)^2.$$

As the importance sampling estimator is, by construction, unbiased, the second term on the right is simply equal to ℓ^2 (which matches exactly the equivalent term in the variance of the standard estimator). This means that, the variance of the importance sampling estimator will be small than the variance of the standard Monte Carlo estimator so long as

$$\mathbb{E}_g\left(\frac{f(X)}{g(X)}S(X)\right)^2 < \mathbb{E}_f S(X)^2.$$

We can make this inequality more meaningful by writing the second moment of the IS estimator as an expectation with respect to f. To do this, observe that

$$\begin{aligned}\mathbb{E}_g\left(\frac{f(X)}{g(X)}S(X)\right)^2 &= \int_{\mathbb{R}} \frac{f(x)^2}{g(x)^2}S(x)^2 g(x)\,\mathrm{d}x \\ &= \int_{\mathbb{R}} \frac{f(x)}{g(x)}S(x)^2 f(x)\,\mathrm{d}x = \mathbb{E}_f\frac{f(X)}{g(X)}S(X)^2.\end{aligned}$$

Thus, we will achieve variance reduction so long as

$$\mathbb{E}_f\frac{f(X)}{g(X)}S(X)^2 < \mathbb{E}_f S(X)^2.$$

In the special case of rare-event probability estimation, the requirement is that

$$\mathbb{E}_f\frac{f(X)}{g(X)}\mathbb{I}(X\in A) < \mathbb{E}_f\mathbb{I}(X\in A),$$

which will be satisfied is, for example, g gives a large weight to all x such that $x \in A$. We can event take this further and see that the above condition is equivalent to

$$\begin{aligned}&\mathbb{E}_f\left[\frac{f(X)}{g(X)}\mathbb{I}(X\in A)\middle| X\in A\right]\mathbb{P}(X\in A) < \mathbb{P}(X\in A) \\ &\Rightarrow \mathbb{E}_f\left[\frac{f(X)}{g(X)}\mathbb{I}(X\in A)\middle| X\in A\right] < 1.\end{aligned}$$

That is, we require that the mean value of the likelihood ratio on the set A should be less than 1.

5.2 The zero-variance density

What is the best possible choice of g? As usual with variance reduction, this choice will require us to know the answer already. However, it will give valuable insight into good choices of g.

If $S(X) \geq 0$ for all x then, we can choose

$$g^*(x) = \frac{f(x)S(x)}{\mathbb{E}_f S(X)} = \frac{f(x)S(x)}{\ell}.$$

We need to check that this is a density. Clearly, under our assumptions, $g^*(x) \geq 0$. Thus, we simply need to check it integrates to one, which is immediately true because $\int S(x)f(x)\,\mathrm{d}x = \ell$.

The resulting estimator is of the form

$$\widehat{\ell}_{g^*} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{f(X_i)S(X_i)/\ell} S(X_i) = \frac{1}{N}\sum_{i=1}^{N} \ell,$$

which has zero-variance as it is a constant. Note that we cannot use this estimator in practice, as we need to know ℓ in order to calculate the likelihood ratio.

In the special case where $S(x) = \mathbb{I}(x \in A)$, we have

$$g^* = \frac{f(x)\mathbb{I}(x \in A)}{\mathbb{P}(X \in A)} = f(x \,|\, X \in A).$$

That is, the zero-variance density is the original density conditioned on the event of interest happening. This is a very useful thing to know, as we are often able to find a density that is quite close to the conditional density.

5.3 Exponential tilting

Trying to find the best choice of g is essentially an infinite dimensional optimization problem. Such problems are generally very difficult (though one can use the calculus of variations and theory of optimal control). In practice, it is often easy to try to turn the problem into a finite dimensional one. This can be done by restricting the choice of g to a parametric family of densities $\{g(x;\theta)\}_{\theta \in \Theta}$.

The most useful way to create such a family of densities is by so-called exponential tilting. Recall that the moment generating function of a random variable $X \sim f$ is given by

$$M(\theta) = \mathbb{E}_f \exp\{\theta X\},$$

which is defined so long as the expectation on the right is finite. The cumulant generating function of a random variable is given by

$$\kappa(\theta) = \log \mathbb{E}_f \exp\{\theta X\} = \log M(\theta).$$

Using this, one can define a family of densities by

$$f(x;\theta) = \exp\{\theta x - \kappa(\theta)\} f(x),$$

where $\theta \in \Theta = \{\theta : M(\theta) < \infty\}$. One can see these are densities as the exponential function is non-negative and the density integrates to 1.

Example 5.3.1 (Tilting the normal density)**.** The moment generating function of a normal distribution with mean μ and variance σ^2 is given by

$$M(\theta) = \exp\left\{\mu\theta + \frac{\sigma^2\theta^2}{2}\right\}.$$

So,

$$\kappa(\theta) = \mu\theta + \frac{\sigma^2\theta^2}{2}.$$

This means that the tilted normal distribution is of the form

$$\begin{aligned} f(x;\theta) &\propto \exp\left\{\theta x - \kappa(\theta)\right\} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \\ &= \exp\left\{\theta x - \mu\theta + \frac{\sigma^2\theta^2}{2}\right\} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \\ &= \exp\left\{-\frac{1}{2\sigma^2}\left(x-(\mu+\sigma^2\theta)\right)^2\right\}, \end{aligned}$$

which is the pdf of a normal distribution with mean $\mu+\sigma^2\theta$ and variance σ^2. Thus, exponentially tilting a normal distribution is equivalent to shifting its mean by $\sigma^2\theta$.

Using an exponential tilt, one obtains the estimator

$$\widehat{\ell}_\theta = \frac{1}{N}\sum_{i=1}^N \frac{f(X_i)}{f(X_i;\theta)} S(X_i) = \frac{1}{N}\sum_{i=1}^N \exp\{\kappa(\theta) - \theta X_i\} S(X_i).$$

The obvious question is how to choose θ. There are a number of possibilities here. One is to try to find the θ that minimizes the variance of the estimator. That is, we choose

$$\theta_{\mathsf{VM}} = \operatorname*{argmin}_{\theta\in\Theta} \mathrm{Var}_g\left(\exp\{\kappa(\theta)-\theta X\}S(X)\right).$$

This is equivalent to minimizing the second moment of the estimator. Using the results from the section on variance above, the problem is thus

$$\theta_{\mathsf{VM}} = \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_f \exp\{\kappa(\theta)-\theta X\}S(X).$$

In the case where $S(x) = \mathbb{I}(x \in A)$, this problem reduces further to

$$\theta_{\mathsf{VM}} = \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_f \left[\exp\{\kappa(\theta)-\theta X\} \,\middle|\, X \in A\right].$$

Note that this can be estimated by first using acceptance rejection to sample from $f(x \mid X \in A)$ then using numerical optimization to find the best choice of θ.

Example 5.3.2. Suppose we wish to estimate $\ell = \mathbb{P}(X > \gamma)$ where $X \sim \mathsf{N}(0,1)$. We can use the variance minimization approach to carry out importance sampling. First, we simulate a small initial simulation and use numerical optimization to find the optimal parameter. We then use this paramter to obtain a tilted distribution with which we perform the importance sampling. The code for the main file is:

Listing 5.2: Matlab Code

```
gamma = 3; mu = 0; sigma = 1;
N_init = 10^4;
N = 10^5;

X_init = mu + sigma * randn(N_init,1);
X_sample = X_init(find(X_init > gamma));
theta_vm = fminsearch(@(theta) second_moment(theta,mu,sigma,X_sample),0);
```

```

X_is = theta_vm + randn(N,1);
results_IS = exp(mu*theta_vm + sigma^2*theta_vm^2/2 - theta_vm*X_is)...
    .* (X_is > gamma);
X_CMC = randn(N + N_init,1);
results_CMC = (X_CMC > gamma);

clc
ell = 1 - normcdf((gamma-mu)/sigma)
ell_hat_IS = mean(results_IS)
re_hat_IS = std(results_IS) / (sqrt(N)* ell_hat_IS)
ell_hat_CMC = mean(results_CMC)
re_hat_CMC = std(results_CMC) / (sqrt(N+N_init)* ell_hat_CMC)
```

The optimization code calls a function which returns an estimate of $\mathbb{E}_f[\exp\{\kappa(\theta) - \theta X\}\,|X > \gamma]$ given the sample. The code for this function is given by

Listing 5.3: Matlab Code

```
function [ val ] = second_moment( theta, mu, sigma, X )
    kappa_theta = mu * theta + sigma^2 * theta^2 / 2;
    val = mean(exp(kappa_theta - theta*X));
end
```

An alternative way to choose θ is so that the mean of the tilted density is the point of interest (e.g., in the case where we wish to estimate $\mathbb{P}(X > \gamma)$ we tilt the density so that $\mathbb{E}_\theta X = \gamma$. To do this, we make use of the following result.

Lemma 5.3.3. If $X \sim f$ with cumulant generating function κ, then provided that it is OK to exchange differentiation and expectation,

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\kappa(\theta) = \mathbb{E}_\theta X.$$

Proof. Observe that

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\theta}\kappa(\theta) &= \frac{\mathrm{d}}{\mathrm{d}\theta}\log \mathbb{E}e^{\theta X} = \frac{1}{e^{\kappa(\theta)}}\frac{\mathrm{d}}{\mathrm{d}\theta}\mathbb{E}e^{\theta X} \\ &= \frac{1}{e^{\kappa(\theta)}}\mathbb{E}\frac{\mathrm{d}}{\mathrm{d}\theta}e^{\theta X} = \frac{1}{e^{\kappa(\theta)}}\mathbb{E}Xe^{\theta X} \\ &= e^{-\kappa(\theta)}\int x\,e^{\theta x}f(x)\,\mathrm{d}x = \int x\,e^{\theta x-\kappa(\theta)}f(x)\,\mathrm{d}x = \mathbb{E}_\theta X.\end{aligned}$$

□

Using the above lemma, it is clear we can choose the tilted density to have mean γ by choosing

$$\theta_{\mathsf{ms}} = \theta \text{ such that } \frac{\mathrm{d}}{\mathrm{d}\theta}\kappa(\theta) = \gamma.$$

Example 5.3.4. Recall for a normal density the cumulant generating function is given by $\kappa(\theta) = \mu\theta + \sigma^2\theta^2/2$. Thus,

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\kappa(\theta) = \mu + \sigma^2\theta \Rightarrow \theta_{\mathsf{ms}} = \frac{\gamma - \mu}{\sigma^2}.$$

Using this, we can then estimate $\mathbb{P}(X > \gamma)$ via importance sampling as follows.

Listing 5.4: Matlab Code

```
gamma = 4; mu = 0; sigma = 1;
N = 10^5;

theta = (gamma - mu ) / sigma^2;

X_is = mu + sigma^2*theta+ sigma*randn(N,1);
results_IS = exp(mu*theta + sigma^2*theta^2/2 - theta*X_is) .* (X_is > gamma);
X_CMC = mu + sigma * randn(N,1);
results_CMC = (X_CMC > gamma);

clc
ell = 1 - normcdf((gamma-mu)/sigma)
ell_hat_IS = mean(results_IS)
re_hat_IS = std(results_IS) / (sqrt(N)* ell_hat_IS)
ell_hat_CMC = mean(results_CMC)
re_hat_CMC = std(results_CMC) / (sqrt(N)* ell_hat_CMC)
```

A first hint of why this approach works very well can be found in the Chernoff bound.

Lemma 5.3.5 (Chernoff Inequality)**.** If X has cumulant generating function κ, then

$$\mathbb{P}(X > \gamma) \leq \inf_{\theta > 0} \exp\left\{\kappa(\theta) - \theta\gamma\right\}.$$

Proof. We have that, for all $\theta > 0$

$$\begin{aligned}\mathbb{P}(X > \gamma) &= \mathbb{E}\mathbb{I}(X > \gamma) = \mathbb{E}e^{\theta X - \theta X}\mathbb{I}(X > \gamma) \\ &\leq e^{-\theta\gamma}\mathbb{E}e^{\theta X}\mathbb{I}(X > \gamma) \leq e^{-\theta\gamma}\mathbb{E}e^{\theta X} = e^{\kappa(\theta) - \theta\gamma}.\end{aligned}$$

The bound then follows by taking the infimum over all $\theta > 0$. □

Note that, in general, this bound is minimized by choosing θ such that

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left[\kappa(\theta) - \theta\gamma\right] = 0 \Rightarrow \frac{\mathrm{d}}{\mathrm{d}\theta}\kappa(\theta = \gamma.$$