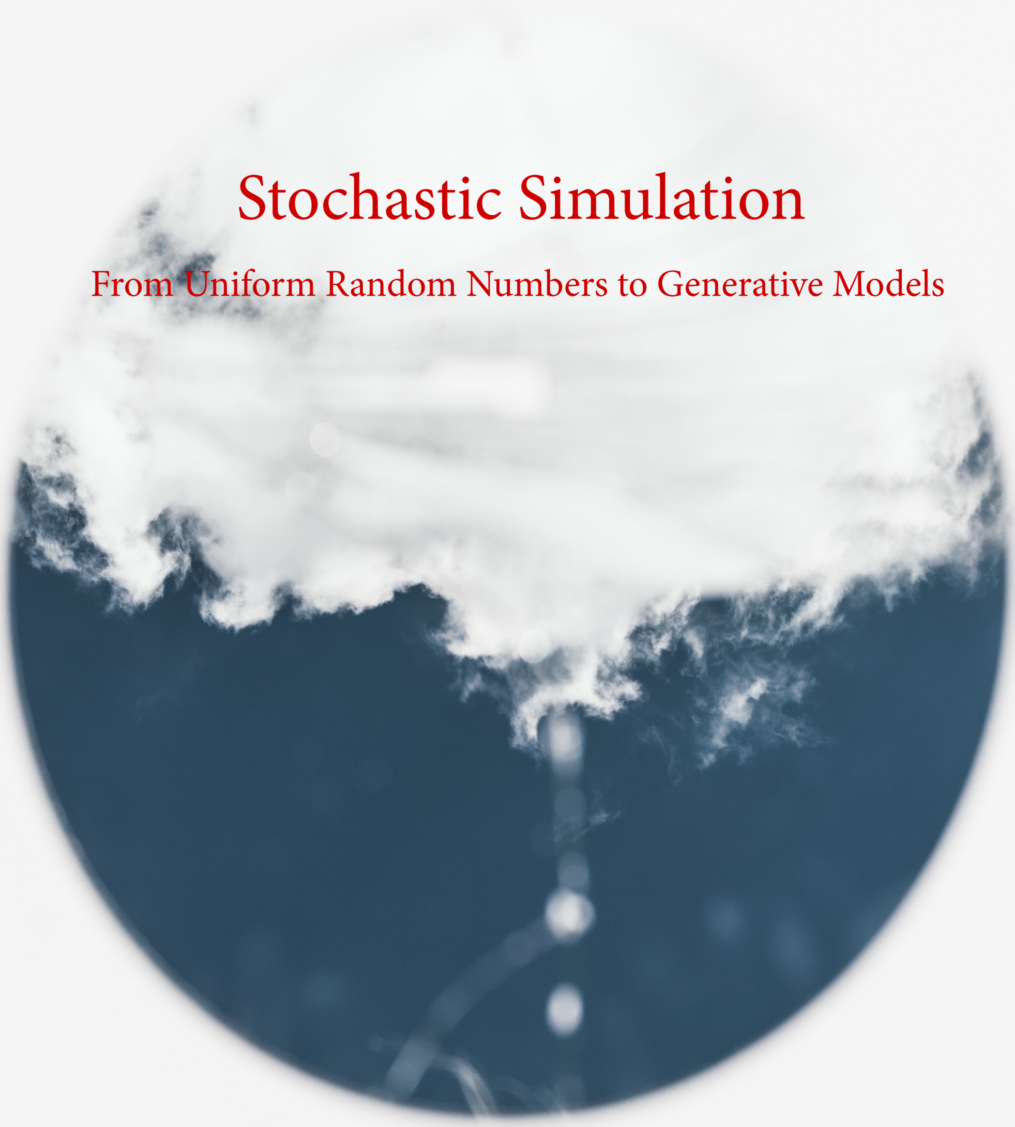
Stochastic Simulation

From Uniform Random Numbers to Generative Models



O. Deniz Akyildiz

Department of Mathematics
Imperial College London

PREFACE

This text is about stochastic simulation methods that underpin most of modern statistical inference, generative modelling, machine learning, and engineering applications. The material of this course serves an introduction to wide ranging areas and applications, from inference and estimation in a broad class of statistical models to modern generative modelling applications.

The core of this course is about sampling from a probability distribution that may have explicit, implicit, complicated, or intractable form. This problem arises in many fields of science and engineering, e.g., the probability distribution of interest can describe a posterior distribution in a statistical model or an unknown data distribution (in a generative model) from which we are interested in generating more samples. The sampling problem takes many forms, hence the solutions (*sampling algorithms*) is a broad topic, and this course is a fairly computational introduction to such methods. In order to develop tools to tackle such problems, we will be covering basics of simulation, starting from something as basic as simulating an independent random number from simple distributions (i.e. pseudo-sampling methods that underlie all stochastic simulations) to designing advanced sampling algorithms for more complicated models.

I hope that this course will strengthen your skills to conduct statistical research and become well-versed in the field of sampling, statistical inference, and generative modelling, no matter if you want to be an academic researcher or a practitioner!

O. Deniz Akyildiz
London, 2022

CONTENTS

1 INTRODUCTION

We introduce in this section the general ideas of this course, notation, and setup. We will also introduce the principles behind sampling and generative modelling and also try to answer the existential question from the beginning: Why is this course useful?

1.1 INTRODUCTION

This course is about *simulating random variables* or put it differently *sampling from probability distributions*. This seemingly simple task arises in a large number of scenarios in the real world and embedded in many critical applications. Furthermore, we look into generating samples from dependent processes (e.g. stochastic processes) as well as sampling from *intractable* distributions.

1.1.1 WHY IS THIS COURSE USEFUL?

This course is about simulating *structured randomness*. The course is a standard part of a Mathematics/Statistics curriculum, sometimes named differently, e.g., *Monte Carlo methods* or *Computational Statistics*. This course will teach you essential skills of simulating random systems and as such this can be very useful in a multitude of settings:

- **Simulation of complex systems:** It is often of interest to outright simulate systems to understand their behaviour. This is straightforward for simple systems but it can get complicated when one wants to model intricate systems. This sort of "forward simulation" approaches consist of designing the system and then simulating it. This is a very common approach in engineering and physics.
- **Statistical inference:** Now imagine a random system that is not fully known. This could be defined using a simulator or a generator as described above. But if some variables

in this system are not known and we have some *data* of other variables, we can always *condition* on the data and find out latent/hidden variables. Simulation and sampling methods come in handy in these cases. By carefully formulating these estimation problems as simulation/sampling problems, we can estimate the unknown variables. We will cover this in detail in the course.

- **Generative modelling:** We are often interested in sampling from a completely unknown probability measure p in the cases of generative models. Consider a given image dataset $\{x_1, \ldots, x_n\}$ and consider the problem of generating new images that mimic the properties of the image dataset. This problem can be framed as a sampling procedure $X \sim p$ where p is unknown but we have access to its samples $\{x_1, \ldots, x_n\}$ in the form of a dataset. Methods that address this challenge gave rise to very successful generative models, such as DALLE-2 or ChatGPT. We will **not** cover generative models in this course, however. The methods we introduce are the key foundations for working on these models.

1.1.2 WHAT WILL BE COVERED IN THIS COURSE?

We will first go through *exact* sampling methods. That is, we will learn the mechanisms that are usually under the hood when you call a function to generate random numbers. To begin, we will learn to generate random samples from simple distributions, e.g., uniform, Gaussian, and exponential. We will then move onto *integration* problems, one of the fundamental applications of sampling methods and will learn to use these samplers to estimate integrals. This will be useful in the context of Bayesian models and inference. Then, we will move onto Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) methods which are for intractable distributions that do not have a simple form.

1.2 THE SAMPLING PROBLEM

Given a probability distribution p, we are often interested in sampling from this distribution. This is simply denoted as drawing

$$X^{(i)} \sim p, \qquad i = 1, \ldots, N.$$

The main goal here is to draw these samples as accurately as possible, as often we may not have access to an exact sampling scheme. More technical applications of *sampling* include

- **Integration.** First reason sampling from a measure is interesting is that, even though one may have access to density p's analytic expression, computing certain integrals with respect to this density may be intractable. This is required, e.g., for estimating tail probabilities. Sampling from a distribution provides a way to compute this integrals (called Monte Carlo integration, which will be introduced later in this course). This motivation also holds for more general cases below.

- **Intractable normalising constants.** In many real life modelling scenarios, we have an access to a function $\bar{p}$ such that

$$p(x) = \frac{\bar{p}(x)}{Z},$$

where the normalising constant Z is unknown. In these cases, designing a sampler may be non-trivial and this is a big research area.

- **Generative models.** Given a trained generative model, we still want to generate samples as fast as possible.

We will first discuss *exact* sampling methods which are only applicable to simple distributions. However, these will be crucial for advanced sampling techniques – as all sampling methods rely on being able to draw realistic samples from simple distributions such as uniform or Gaussian distribution. We will then describe cases where the exact sampling is not possible and introduce advanced sampling algorithms (such as Markov chain Monte Carlo (MCMC) methods) to tackle this problem.

1.2.1 MOTIVATING EXAMPLE: ESTIMATING π

We mentioned repeatedly above that we can use *random sampling* methods to do various computation tasks. We can now demonstrate an intuitive application of this and will the idea of random variate generation to estimate π. This example is simple enough for us to understand the idea even if we have not seen any sampling methods yet.

Suppose that we want to obtain an estimate of π using computation. Normally, this number is available on any given software package, you can just query it, e.g., `np.pi` in Python. Let us assume that we do not have access to this number and we want to estimate it using a computer. We can use this through a neat geometric idea: Consider Fig. 1.1. We know that the area of a circle is given by πr^2 for any r. If we fit a square around a circle, then we also know that the square will have one side of length $2r$ and hence the area of the square will be $4r^2$. Given this, we can easily conclude

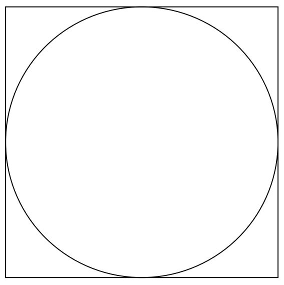


Figure 1.1: The circle and square.

$$\frac{\text{area of circle}}{\text{area of square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}.$$

We can again easily imagine that if we had uniformly distributed points in this square, the ratio of the number of points *within* the circle and the total number of points will be approxiamtely the same as the ratio of the areas. In other words, if we draw N points uniformly at random from the square, and N_c of these points fall within the circle, then

$$\frac{N_c}{N} \approx \frac{\pi}{4}.$$

This is a very simple *Monte Carlo* estimate which comes with lots of guarantees, in particular, we know that as $N \to \infty$, the estimate will converge to the true value of π.

To formalise this intuition, the trick at this stage is to *phrase this question probabilistically*. Can we convert this problem into estimating the probability of a set? Let us say, for clarity of the definition, this square is a set $\mathsf{A} = [-1, 1] \times [-1, 1]$. Let us define a 2-dimensional probability distribution, *uniform* on this set, defined as $\mathbb{P} = \text{Unif}([-1, 1]^{\otimes 2})$. This is a uniform distribution

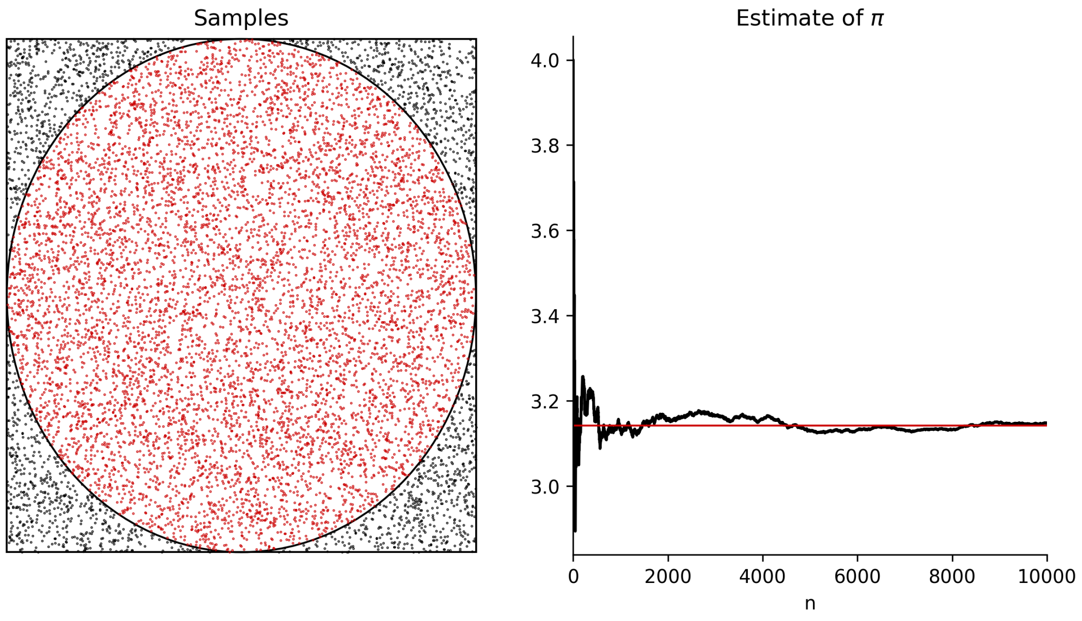


Figure 1.2: Estimating π using the Monte Carlo method.

on the square. Naturally, we have that $\mathbb{P}(\mathsf{A}) = 1$. Now we can see that, if the circle is defined as another set

$$\mathsf{B} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\},$$

then the probability of this set is

$$\mathbb{P}(\mathsf{B}) = \frac{\text{area of circle}}{\text{area of square}} = \frac{\pi}{4}.$$

We need to then find a way to formally estimate this set, using samples from $\mathbb{P}$. This is formally done by converting this probability into an expectation

$$\mathbb{P}(\mathsf{B}) = \mathbb{E}_{\mathbb{P}}[\mathbf{1}_{\mathsf{B}}(X)],$$

and estimating this expectation (integral) using samples. We will see later that this simple Monte Carlo procedure coincides with the intuitive solution: Count the samples within the circle and calculate the ratio to estimate $\pi/4$. We can see the result of this estimation procedure above.

1.3 PROBABILITY AND MEASURE: RECAP

Before we go and recap some of the necessary notions for this course, we briefly describe the notation.

1.3.1 DENSITY NOTATION

We will use $p(x)$ as a generic probability distribution. Normally, in probability text books, the notation used is something like $p_X(x)$ for one random variable X and $p_Y(y)$ for another random variable. This is done in order to stress the fact that the densities p_X and p_Y are different. However, this becomes tedious when doing more complex modelling. For example, a simple case appears in the Bayes' update for conditionals. Again in normal literature, this is written as

$$p_{X|Y}(x|y) = \frac{p_{Y|X}(y|x)p_X(x)}{p_Y(y)}.$$

Now consider even more general cases involving three or more variables and various dependences. This is going to become an infeasible notation.

Throughout these notes and the course, we will use $p(x)$ generically as a probability density of a random variable X. When we then write $p(y)$, this will mean *a different* density of another random variable (say Y). If we write the Bayes' formula in these terms

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)},$$

which is much cleaner. Of course, here, $p(x|y)$ and $p(y|x)$ are different functions, just as $p(x)$ and $p(y)$ are. We will however revert back to p_X and p_Y in cases where it is necessary, such as transformation of random variables.

We denote the expectation of a random variable with $\mathbb{E}_p[X]$ (or $\mathbb{E}_p X$ when there is no confusion). In general, we define the expectation of a function of a random variable $X \sim p$ as

$$(\varphi, p) = \mathbb{E}_p[\varphi(X)] = \int \varphi(x)p(x)\mathrm{d}x.$$

We note the notation here (φ, p) denotes the integration of φ w.r.t. p and we will use this in the sections regarding Monte Carlo integration heavily. Also note that we abuse the notation for denoting the measures and densities with the same letter. In other words, for a probability measure $p(\mathrm{d}x)$, we denote its density with $p(x)$. The cumulative density function (CDF) will generally be denoted as $\mathsf{F}(x)$. In this section, we review basic probability theory that will be required for the rest of the course.

1.3.2 BASIC PROBABILITY DEFINITIONS

Let X be a random variable that takes values on set X. We do not limit ourselves to any specific set X for now. We call this random variable a *discrete* random variable if X is a discrete set, i.e., a set with a finite or countable number of elements. We call it a *continuous* random variable if X is a set that is not countable. We will next define associated probability distributions.

Let us start from the case where a random variable is discrete. This means the set X is either finite or countable. A simple example is

$$\mathsf{X} = \{1, 2, 3, 4, 5, 6\},$$

which could denote, for example, the possible outcomes of a die roll. Now we define the probability mass function.

Definition 1.1 (Probability Mass Functions). *When a random variable is discrete, the probability mass function can be defined as*

$$p(x) = \mathbb{P}(X = x),$$

where $x \in \mathsf{X}$. We call $p(x)$ the probability mass function of X.

We note that in one dimensional case, the probability mass function is typically represented as a vector of probabilities when it comes to computations. Consider the following example.

Example 1.1. Assume that $\mathsf{X} = \{1, 2, 3, 4\}$ and

$$p(x) = \begin{cases} 0.1 & \text{if } x = 1, \\ 0.2 & \text{if } x = 2, \\ 0.3 & \text{if } x = 3, \\ 0.4 & \text{if } x = 4. \end{cases}$$

Describe how you would represent such a distribution on a computer.

Solution: We can see this as a table of probabilities

X	$\mathbb{P}(X = x)$
1	0.1
2	0.2
3	0.3
4	0.4

We can then define its *states* as

$$\mathsf{s} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix},$$

and its *probabilities* as

$$\mathbf{p} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}.$$

indexed by discrete variables. Of course, one can also define a *dictionary* (Python data type) in order to have more complicated states for the random variable.

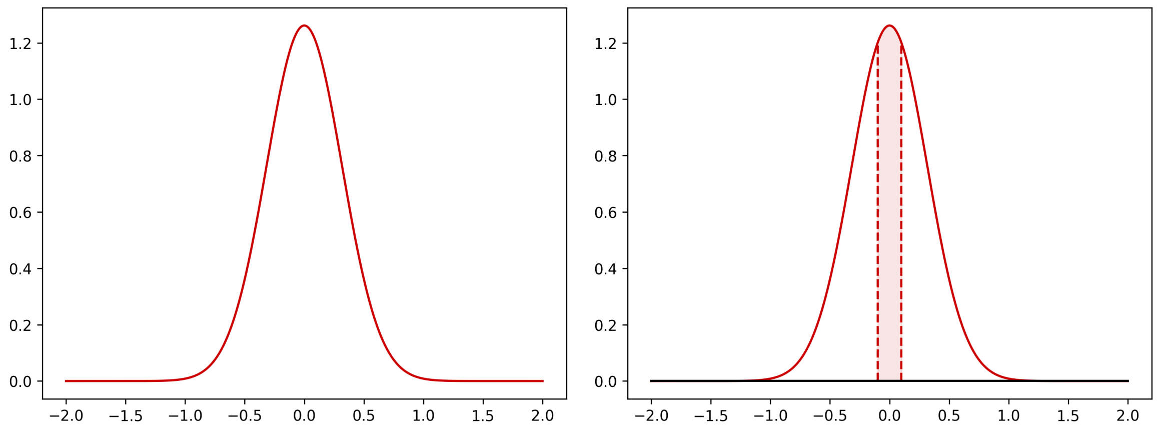


Figure 1.3: A Gaussian example: **Left:** It can be seen that $p(0) = 1.2615$ while we know $\mathbb{P}(X = 0) = 0$. This shows that the density values can be bigger than 1 as they are not probabilities without the integration. **Right:** One can only compute quantities like $\mathbb{P}(-0.1 \leq X \leq 0.1) = \int_{-0.1}^{0.1} p(x) \mathrm{d}x = 0.248$.

Next we define the probability density function in the case of continuous random variables.

Definition 1.2 (Measure and density)**.** *Assume* $\mathsf{X} \subset \mathbb{R}$ *and* $X \in \mathsf{X}$ *(for simplicity). Given the random variable* X*, we define the measure of* X *as*

$$\mathbb{P}(x_1 \leq X \leq x_2) = \mathbb{P}(X \in (x_1, x_2)).$$

The reason $\mathbb{P}$ *called a measure is that it measures the probability of sets. We have then the probability density function which has the following relationship with the probability measure*

$$\mathbb{P}(x_1 \leq X \leq x_2) = \int_{x_1}^{x_2} p(x) \, \mathrm{d}x.$$

We call $p(x)$ *the probability density function of* X*.*

Note that this notion generalises to higher dimensions straightforwardly. In the estimating π example, we basically tried to “measure” the probability of the 2D set that was a circle. Another important note about the measure/density distinction is demonstrated in the following remark.

Remark 1.1. Note that this difference in the continuous case matters. Consider the probability of a point $x' \in \mathbb{R}$ under a continuous probability distribution defined on $\mathbb{R}$. Given its density $p(x')$, we can surely evaluate $p(x') > 0$, but this is **not** the probability of x' under the distribution. The probability of a point x' is zero under a continuous distribution which follows from Definition 1.2 as $x_1 = x_2 = x'$. An example is demonstrated in Fig. 1.3.

When no confusion arises, we will write $p(\mathrm{d}x) = p(x)\mathrm{d}x$ and use this notation to denote the measure and density interchangeably. This will come in very handy as we will see Monte Carlo approximations as approximations to measures rather than densities.

1.3.3 JOINT PROBABILITY

We now define the joint probability distribution of two random variables X and Y. We will also focus on the discrete case first, and then move to the continuous case.

Definition 1.3 (Discrete Joint Probability Mass Function)**.** *Let X and Y be random variables and $\mathcal{X}$ and $\mathcal{Y}$ be the sets they live on. $\mathcal{X}$ and $\mathcal{Y}$ are at most countable sets. The joint probability mass function of X and Y is*

$$p(x, y) = \mathbb{P}(X = x, Y = y).$$

We call $p(x, y)$ the joint probability mass function of X and Y.

Example 1.2. Similar to the one dimensional case, we can now see the joint pmf $p(x, y)$ as a table of probabilities

	$Y = 0$	$Y = 1$	$Y = 2$	$Y = 3$	$p_X(x)$
$X = 0$	1/6	1/6	0	0	2/6
$X = 1$	1/6	0	1/6	0	2/6
$X = 2$	0	0	1/6	0	1/6
$X = 3$	0	0	0	1/6	1/6
$p_Y(y)$	2/6	1/6	2/6	1/6	1

Of course, on computer we can represent this as a matrix of probabilities

$$\mathbf{P} = \begin{bmatrix} 1/6 & 1/6 & 0 & 0 \\ 1/6 & 0 & 1/6 & 0 \\ 0 & 0 & 1/6 & 0 \\ 0 & 0 & 0 & 1/6 \end{bmatrix}.$$

This allows us to perform simple computations for marginalisation simply as sums of rows or columns. This is going to be a crucial tool when we study Markov models.

Let us finally define the probability density function $p(x, y)$ for continuous variables.

Definition 1.4 (Continuous Joint Probability Density Function)**.** *Let X and Y be random variables and $\mathcal{X}$ and $\mathcal{Y}$ be their ranges. We denote the joint probability measure as $\mathbb{P}(X \in A, Y \in B)$ and* *the density function $p(x, y)$ satisfies*

$$\mathbb{P}(X \in A, Y \in B) = \int_A \int_B p(x, y) \, \mathrm{d}x \, \mathrm{d}y.$$

The marginal probability densities from the joint density can be computed as

$$p(x) = \int_{\mathsf{Y}} p(x, y) \, \mathrm{d}y,$$

and

$$p(y) = \int_{\mathsf{X}} p(x, y) \, \mathrm{d}x.$$

From Fig. 1.4, you can see the visualisations of a joint density $p(x, y)$ and its marginals $p(x)$ and $p(y)$. Note that while given the joint, it is well-defined to compute marginals; given marginals, it is not a trivial task to compute the joint density and this is a big research area.

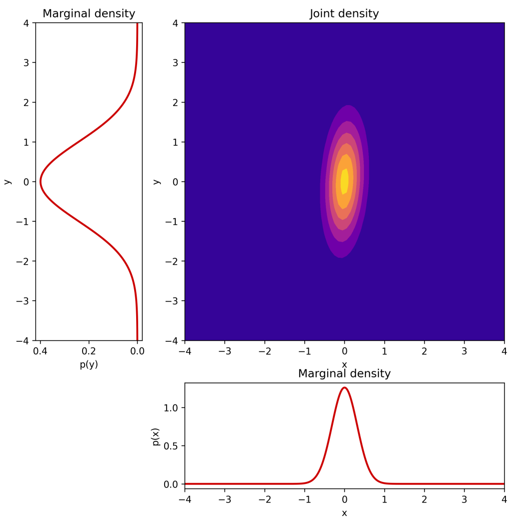


Figure 1.4: Visualisation of marginal densities and the joint density.

1.3.4 CONDITIONAL PROBABILITY

We now define the conditional probability of a random variable X given another random variable Y. As usual, we will first focus on the discrete case, and then move to the continuous case.

Definition 1.5 (Discrete Conditional Probability Mass Function). *Let X and Y be random variables and $\mathcal{X}$ and $\mathcal{Y}$ be their ranges. The conditional probability mass function of X given Y is*

$$p(x \mid y) = \mathbb{P}(X = x \mid Y = y).$$

We call $p(x \mid y)$ the conditional probability mass function of X given Y.

Example 1.3. Compute the conditional probability mass function $p(y|x = 2)$ from the table of probabilities of $p(x, y)$ given below.

	$X = 0$	$X = 1$	$X = 2$	$X = 3$	$p_Y(y)$
$Y = 0$	1/6	1/6	0	0	2/6
$Y = 1$	1/6	0	1/6	0	2/6
$Y = 2$	0	0	1/6	0	1/6
$Y = 3$	0	0	0	1/6	1/6
$p_X(x)$	2/6	1/6	2/6	1/6	1

Solution. Let us say we would like to compute $\mathbb{P}(Y = i|X = 2)$ for $i = 0, 1, 2, 3$. We can do this by simply dividing the joint probability mass function by the marginal probability mass function of X. Consider the following table

$p(x, y)$	$X = 0$	$X = 1$	$X = 2$	$X = 3$	$p_Y(y)$
$Y = 0$	1/6	1/6	**0**	0	2/6
$Y = 1$	1/6	0	**1/6**	0	2/6
$Y = 2$	0	0	**1/6**	0	1/6
$Y = 3$	0	0	**0**	1/6	1/6
$p_X(x)$	2/6	1/6	**2/6**	1/6	1

where the red entries are the joint probabilities of Y given $X = 2$. We can write the conditional probabilities as

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

As we can see that the conditional probability can also be represented as a vector

$$\mathbf{p} = [0, 1/2, 1/2, 0].$$

for implementation purposes.

One can compute conditional probability tables from the joint probability table.

Example 1.4. Derive the conditional probability table from the joint probability table given above.

Solution. The conditional probability table is given as

$p(y|x)$	$X = 0$	$X = 1$	$X = 2$	$X = 3$
$Y = 0$	1/2	1	0	0
$Y = 1$	1/2	0	1/2	0
$Y = 2$	0	0	1/2	0
$Y = 3$	0	0	0	1

Similarly, we can compute $p(x|y)$ as

$p(x\|y)$	$X = 0$	$X = 1$	$X = 2$	$X = 3$
$Y = 0$	1/2	1/2	0	0
$Y = 1$	1/2	0	1/2	0
$Y = 2$	0	0	1	0
$Y = 3$	0	0	0	1

We next define the continuous conditional density given $p(x, y)$.

Definition 1.6 (Continuous Conditional Probability Density Function)**.** *Let X and Y be random variables and $\mathcal{X}$ and $\mathcal{Y}$ be their ranges. The conditional probability density function of X given Y is*

$$p(y|x) = \frac{p(x, y)}{p(x)},$$

where we call $p(x \mid y)$ the conditional probability density function of X given Y. Similarly, we have

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

We call $p(x \mid y)$ the conditional probability density function of X given Y.

2

EXACT GENERATION OF RANDOM VARIATES

In this section, we will focus on exact sampling from certain classes of distributions, above all, uniform distribution. This chapter aims at developing an understanding for the basis for all simulation algorithms.

One of the central pillars of sampling algorithms is the ability to sample from the *uniform distribution*. This may sound straightforward, however, it is surprisingly difficult to sample a real uniform random number (Devroye, 1986). If the aim is to generate these numbers on a computer, one has to "listen" to some randomness[1] (e.g. thermal noise in the circuits of the computer) and even then, this random numbers have no guarantee to follow a uniform distribution. Therefore, much of the literature is devoted to generating *pseudo*-uniform random number. Furthermore, generation of random variables that follow popular distributions in statistics (such as Gaussian or exponential distribution) also requires pseudo-uniform random numbers as we will see in next sections.

Definition 2.1. *A sequence pseudo-random numbers $u_1, u_2, \ldots$ is a deterministic sequence of numbers whose statistical properties match a sequence of random numbers from a desired distribution.*

Why would we use pseudo-random numbers? They are (i) easier, quicker, and cheaper to generate, and more importantly, (ii) repeatable. This provides a crucial experimental advantage when it comes to testing algorithms based on random numbers – as we can re-run the same code (with the same pseudo-random numbers), e.g., for debugging.

In what follows, we will describe different methods for pseudo-uniform random number generators that can be used in practice.

[1] see https://www.random.org/ if you need real random numbers.

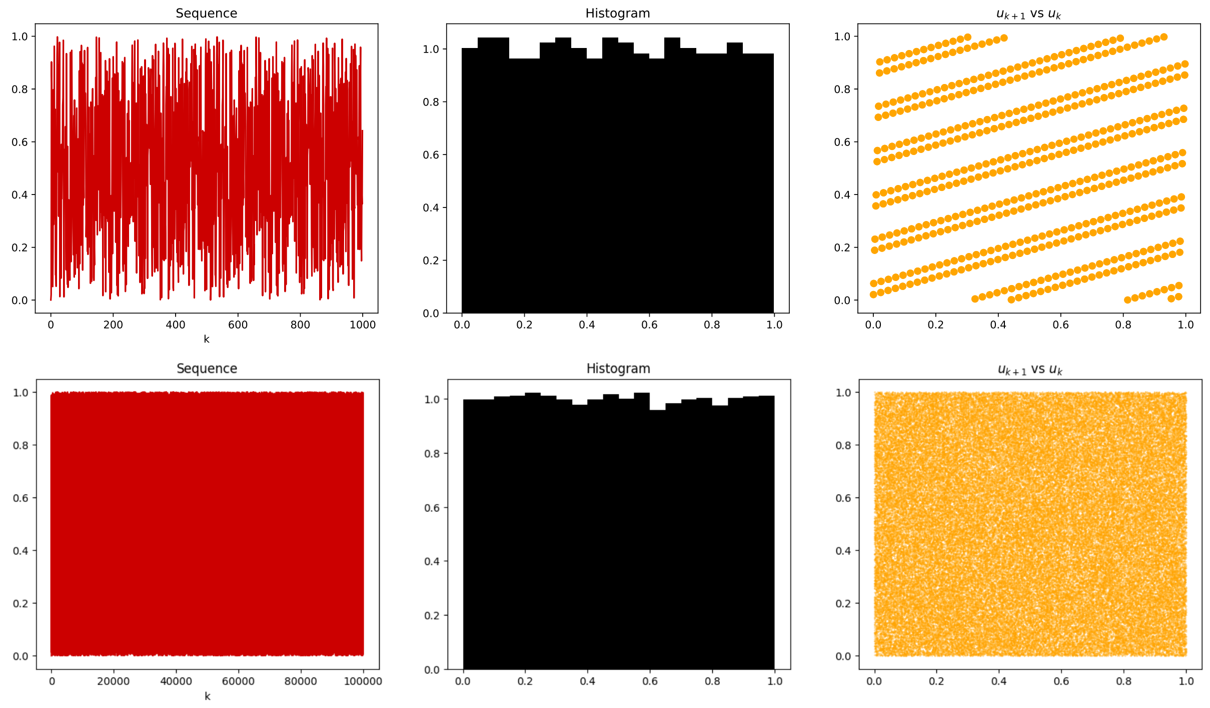


Figure 2.1: **Top Left:** Samples generated by an LCG with parameters $m = 2048, a = 43, b = 0, x_0 = 1$. **Top Middle:** The histogram of the samples showing that it conforms the uniform distribution. **Top Right:** However, plotting (u_k, u_{k+1}) pairs shows that the samples are not *random enough*. **Bottom Left:** Samples generated by an LCG with parameters $m = 2^{32}, a = 1664525, b = 1013904223$. **Bottom Middle:** The histogram of the samples showing that it conforms the uniform distribution. **Bottom Right:** Plotting (u_k, u_{k+1}) pairs shows that the samples do not look nonrandom.

2.1 GENERATING UNIFORM RANDOM VARIATES

The most popular (in practice) uniform random number generator is called the *linear congruential generator* (LCG). This method generates random numbers using a linear recursion

$$x_{n+1} \equiv ax_n + b \qquad (\text{mod } m)$$

where x_0 is the **seed**, m is the **modulus** of the recursion, b is the **shift**, and a is the **multiplier**. If $b = 0$, then the LCG is called a multiplicative generator and it is called a mixed generator when $b \neq 0$. We set m an integer and choose $x_0, a, b \in \{0, \ldots, m-1\}$. Defined this way, the recursion defines a class of generators and we have $x_n \in \{0, 1, \ldots, m-1\}$. The uniform numbers are then generated

$$u_n = \frac{x_n}{m} \in [0, 1) \qquad \forall n.$$

The sequence $(u_n)_{n\geq 0}$ is *periodic* with period $T \leq m$ (Martino et al., 2018). The goal is to choose the parameters a and b so that the sequence has the largest possible period, ideally, $T = m$ (full period). The choice of the modulus is determined by the precision, e.g., $m \approx 2^{32}$ for single-precision, and so on.

There are known shortcomings of LCGs, in particular that they generate *periodic* sequences. An illustration is given in Fig. 2.1. In particular, if a care is not taken, this can cause a significant problem in applications. For a typical issue, we demonstrate the impact of poor random samples

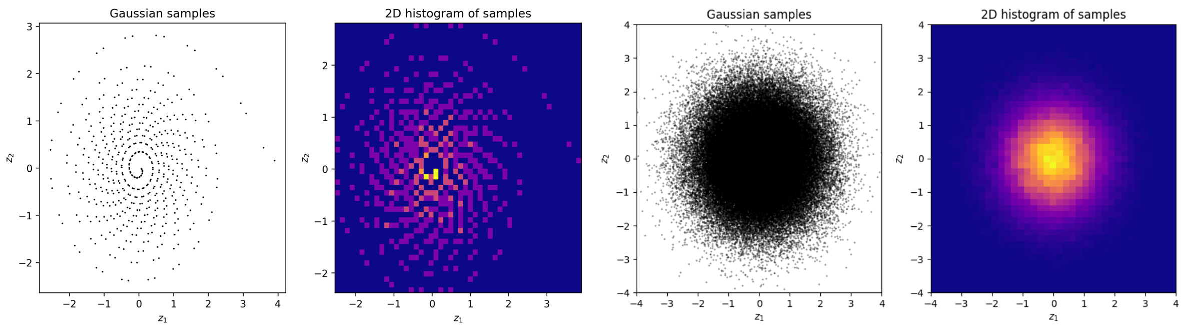


Figure 2.2: **Left two figures:** Gaussian samples generated using the samples in top row of Fig. 2.1. **Right two figures:** Gaussian samples generated using the samples in the bottom row Fig. 2.1 but a better parameterisation: $m = 2^{32}, a = 1664525, b = 1013904223$.

on other random number generators. We will see in the following sections that, for example, drawing Gaussian samples requires uniform random numbers. If we use the random samples generated in top row of Fig. 2.1, we will get bad samples from a "Gaussian distribution", see Fig. 2.2 leftmost two figures. Similarly, if we choose a better parameterisation $m = 2^{32}, a = 1664525, b = 1013904223$, whose samples are shown in the bottom row of Fig. 2.1, then we can get much better samples which can be seen from rightmost two figures of Fig. 2.2.

Unfortunately, LCGs fail to produce good samples in high-dimensions. The standard implemented method nowadays is Mersenne-Twister algorithm which is **not** an LCG. For the rest of the course, we will use `rng.uniform(0, 1)` to draw uniform random numbers, where `rng` is a properly initialised RNG (see online companion), as a suboptimal implementation can impact our simulation schemes.

2.2 TRANSFORMATION METHODS

Given a pseudo-uniform random number generator, we can now describe some exact sampling methods. In this section, we will describe *transformation* methods, where a uniform random number

$$U \sim \text{Unif}(0, 1),$$

can be transferred to a prescribed random variable $Y = g(U)$ using a deterministic transform g. Note that, in almost all of our exact sampling methods for nonuniform densities, we need a uniform random number generator – hence the samplers above are of crucial importance to stochastic simulation applications.

We will next start with the inversion method.

2.2.1 INVERSE TRANSFORM

This method considers the cumulative distribution function (CDF) F of a density p to draw samples from p given access to uniform random numbers. The intuition of the method is best seen on a discrete example first. Assume p is a *discrete* distribution on some finite set X taking values $x_1, x_2, x_3, \ldots$. The CDF is an increasing staircase function whose spacing in y axis reflects

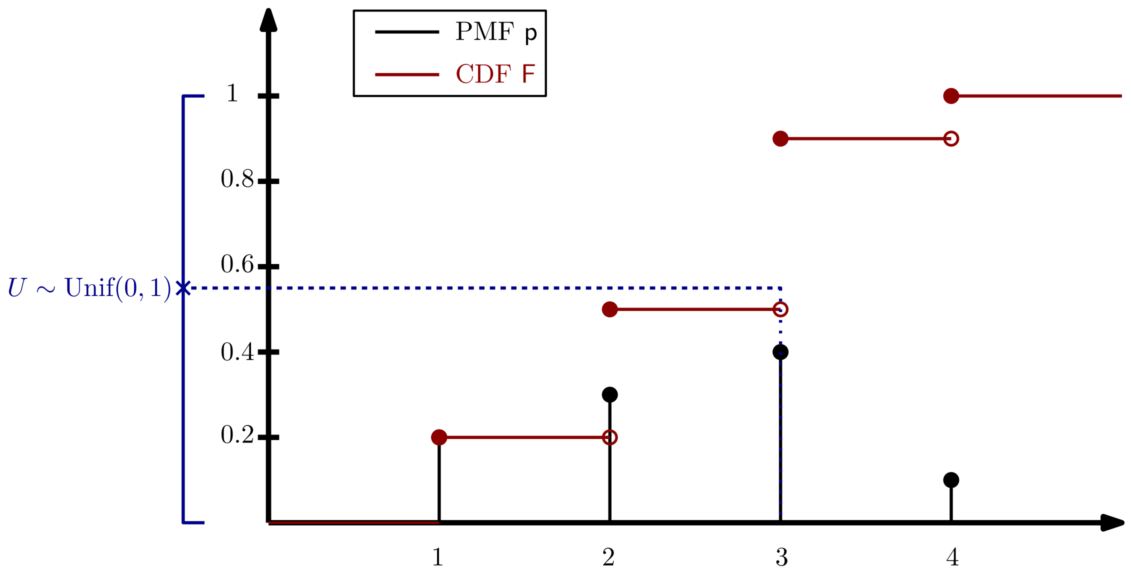


Figure 2.3: The inversion technique. The black is the probability mass function p whereas the red function is the CDF F. One can see that, drawing a uniform random number projected on the y axis ensures that we draw the samples $\{1, \dots, 4\}$ w.r.t. the probability if we follow the inverse of CDF.

probabilities. In other words, if we draw $U \sim \text{Unif}(0,1)$ and invert through the CDF, we will choose $x_1, x_2, \dots$ according to their probabilities (see Fig. 2.3).

This follows from a more general result called *probability integral transform*.

Theorem 2.1. *Consider a random variable X with a CDF F_X. Then the random variable $F_X^{-1}(U)$ where $U \sim \mathsf{Unif}(0,1)$, has the same distribution as X.*

Proof. The proof is one line:

$$\mathbb{P}(F_X^{-1}(U) \leq x) = \mathbb{P}(U \leq F_X(x)) = F_X(x).$$

which is the CDF of the target distribution[2]. □

This suggests then a sampling procedure for distributions where we can compute F_X^{-1}.

Example 2.1. (Discrete distribution) Let p be a discrete probability distribution defined on the set $\mathsf{S} = \{s_1, \dots, s_K\}$ (states) with probabilities $p(s_k) = w_k$ and $\sum_{k=1}^K w_k = 1$. Provide the inversion method.

[2]Note that above result is written for the case where F_X^{-1} exists, i.e., the CDF is continuous. If this is not the case, one can define the generalised inverse function,

$$F_X^{-}(u) = \min\{x : F_X(x) \geq u\},$$

for which the result holds.

Algorithm 1 Pseudocode for inverse transform sampling

```
1: Input: The number of samples $n$
2: for $i = 1, \dots, n$ do
3:     Generate $U_i \sim \text{Unif}(0, 1)$
4:     Set $X_i = F_X^{-1}(U_i)$
5: end for
```

Solution. In this case, the CDF is not continuous and the procedure is summarised in Fig. 2.3. Let us call X as our random variable to be sampled $X \sim p$. Then the sampling procedure as described above becomes (with generalised inverse):

- $U_i \sim \text{Unif}(0, 1)$
- Find $k_i = \min_k\{F_X(s_k) \geq u\}$.
- $X_i = s_{k_i}$.

This corresponds to something simple: Sample U_i and find the first state s_k that gives $F_X(s_k) \geq u$. Note that Bernoulli distribution corresponds to a special case of this with $s_1 = 0$, $s_2 = 1$ (see Fig. 2.3).

Example 2.2. (Exponential) Describe a sampler to generate $X \sim \text{Exp}(x; \lambda) = \lambda e^{-\lambda x}$.

Solution. We calculate the CDF

$$\begin{aligned}
F_X(x) &= \int_0^x p(x')\mathrm{d}x', \\
&= \lambda \int_0^x e^{-\lambda x'}\mathrm{d}x', \\
&= \lambda \left[-\frac{1}{\lambda} e^{-\lambda x'} \right]_{x'=0}^{x} \\
&= 1 - e^{-\lambda x}.
\end{aligned}$$

Given $F_X(x) = 1 - e^{-\lambda x}$ we start by deriving the inverse by

$$\begin{aligned}
& U = 1 - e^{-\lambda X} \\
\implies & X = -\frac{1}{\lambda} \log(1 - U), \\
\implies & F_X^{-1}(U) = -\lambda^{-1} \log(1 - U).
\end{aligned}$$

The algorithm is described next to draw samples from exponential distribution.

- Generate $U_i \sim \text{Unif}(0, 1)$
- Set $X_i = -\lambda^{-1} \log(1 - U_i)$.

Example 2.3. (Cauchy) Assume we want $X \sim$ Cauchy where the probability density is given as

$$p_X(x) = \frac{1}{\pi(1+x^2)}.$$

Describe the sampler using the inversion method.

Solution. The CDF is analytically available and given as

$$F_X(x) = \int_{-\infty}^{x} p_X(y)\mathrm{d}y = \frac{1}{2} + \pi^{-1}\tan^{-1} x$$

Furthermore, the inverse is also available

$$F_X^{-1}(u) = \tan\left[\pi\left(U - \frac{1}{2}\right)\right]$$

Given this, we can provide the algorithm (this should be obvious now!).

- Generate $U_i \sim \text{Unif}(0,1)$
- Set $X_i = \tan\left[\pi\left(U_i - \frac{1}{2}\right)\right]$.

Example 2.4. (Poisson) Consider the Poisson distribution

$$\mathbb{P}(X = k) = \text{Pois}(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}.$$

Describe the sampler using the inversion method.

Solution. The CDF is given as

$$F_X(k) = \mathbb{P}(X \leq k) = e^{-\lambda}\sum_{i=0}^{k}\frac{\lambda^i}{i!}.$$

This is similar to the discrete case.

- Sample $U \sim \text{Unif}(0,1)$
- Find the smallest k such that $F_X(k) \geq U$

then $k \in \mathbb{N}$ is our sample.

While this is a useful technique for sampling from many distributions, it is limited to the cases where F_X^{-1} exists, which is a very stringent condition. For example, consider the problem of sampling a standard normal, i.e., $X \sim \mathcal{N}(0,1)$. We know that the CDF is

$$F_X(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2} \mathrm{d}y.$$

We cannot find F_X^{-1}. Fortunately, for certain special cases, we can use another transformation to sample.

2.2.2 TRANSFORMATION METHOD

Transformation method is a generalisation of the inversion method, in the sense that, one can generalise the idea of sampling U and passing it through F_X^{-1} to using a more general transform g. In this case, we can describe the sampling procedure as the following algorithm Of course,

Algorithm 2 Pseudocode for transformation method

```
1: Input: The number of samples $n$.
2: for $i = 1, \dots, n$ do
3:     Generate $U_i \sim$ Unif$(0,1)$
4:     Set $X_i = g(U_i)$
5: end for
```

designing g is the crucial aspect of this method. This depends on the goal of the sampling method. A crucial tool to understand what happens with this kind of transformations is the formula describes *transformation of random variables* which is described below.

Remark 2.1. The transformation of random variables formula is an important formula for us describing the transformation of probability densities when we transform random variables. In other words, assume $X \sim p_X(x)$ and $Y = g(X)$, then $p_Y(y)$ has a certain density that is related to the density of X. The exact formula depends on the dimension of the random variables. For one-dimensional case, the relationship is given by

$$p_Y(y) = p_X(g^{-1}(y)) \left| \frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y} \right|. \tag{2.1}$$

This formula is simpler than it looks. One needs to explicitly find g^{-1} first (here is the weakness of this approach). Provided that, writing down $p_X(g^{-1}(y))$ simple (just write down the density of X, evaluated at $g^{-1}(y)$. The derivative is also often simple to compute, so is the absolute value.

For multidimensional (say n-dimensional) random variables (we will see one 2D example below), the formula is equally compact and simple, however, computations might become more involved. It is simply given as

$$p_Y(y) = p_X(g^{-1}(y)) \left|\det J_{g^{-1}}(y)\right| \tag{2.2}$$

While the first term on the r.h.s. is similar to above, the term $\det J_{g^{-1}}(y)$ now means *determinant of the Jacobian.* And in this case, the Jacobian would be given as

$$J_{g^{-1}} = \begin{bmatrix} \partial g_1^{-1}/\partial y_1 & \partial g_1{}^{-1}/\partial y_2 & \cdots & \partial g_1^{-1}/\partial y_n \\ \vdots & \cdots & \cdots & \vdots \\ \partial g_n^{-1}/\partial y_1 & \partial g_n{}^{-1}/\partial y_2 & \cdots & \partial g_n^{-1}/\partial y_n \end{bmatrix}$$

where $g^{-1} = (g_1^{-1}, \dots, g_n^{-1})$ is a multivariate function.

Next, we consider the example where we develop the method to sample Gaussian random variates.

Example 2.5. Let X_1 and X_2 be independent random variables where

$$X_1 \sim \text{Exp}\left(\frac{1}{2}\right),$$
$$X_2 \sim \text{Unif}(-\pi, \pi),$$

Then, prove that $Y_1 = \sqrt{X_1}\cos X_2$ and $Y_2 = \sqrt{X_1}\sin X_2$ are independent and standard normal.

Solution. This is a a transformation method with

$$(y_1, y_2) = g(x_1, x_2) = (\sqrt{x_1}\cos x_2, \sqrt{x_1}\sin x_2).$$

We use the transformation of random variables formula (from standard probability)

$$p_{y_1,y_2}(y_1, y_2) = p_{x_1,x_2}(g^{-1}(y_1, y_2))\,|\det J_{g^{-1}}(y_1, y_2)| \tag{2.3}$$

where $J_{g^{-1}}$ is the Jacobian of the inverse. We next derive g^{-1} by writing

$$x_1 = y_1^2 + y_2^2, \qquad \text{as } \cos^2 x_2 + \sin^2 x_2 = 1.$$

and

$$\frac{\sin x_2}{\cos x_2} = \frac{y_2}{y_1}$$

which leads to

$$x_2 = \arctan(y_2/y_1).$$

Therefore, $g^{-1} : \mathbb{R}^2 \to \mathbb{R}^2$

$$g^{-1}(y_1, y_2) = (g_1^{-1}, g_2^{-1}) = \left(y_1^2 + y_2^2, \arctan(y_2/y_1)\right).$$

We now compute the Jacobian

$$\begin{aligned} J_{g^{-1}} &= \begin{bmatrix} \partial g_1^{-1}/\partial y_1 & \partial g_1^{-1}/\partial y_2 \\ \partial g_2^{-1}/\partial y_1 & \partial g_2^{-1}/\partial y_2 \end{bmatrix} \\ &= \begin{bmatrix} 2y_1 & 2y_2 \\ \frac{1}{1+(y_2/y_1)^2}\frac{-y_2}{y_1^2} & \frac{1}{1+(y_2/y_1)^2}\frac{1}{y_1} \end{bmatrix} \end{aligned}$$

Hence, the absolute value of the determinant is:

$$|\det J_{g^{-1}}| = 2.$$

From the transformation of random variables formula

$$\begin{aligned} p_{y_1,y_2}(y_1, y_2) &= \text{Exp}(g_1^{-1}; 1/2)\text{Unif}(g_2^{-1}; -\pi, \pi)\, |J_{g^{-1}}| \\ &= \frac{1}{2} e^{-\frac{1}{2}(y_1^2+y_2^2)} \frac{1}{2\pi} 2 \\ &= \mathcal{N}(y_1; 0, 1)\mathcal{N}(y_2; 0, 1), \end{aligned}$$

which concludes.

Example 2.6. (Sampling uniformly on a circle) Consider drawing

$$\begin{aligned} r &\sim \text{Unif}(0, 1), \\ \theta &\sim \text{Unif}(-\pi, \pi). \end{aligned}$$

Prove that taking $X_1 = \sqrt{r}\cos\theta$ and $X_2 = \sqrt{r}\sin\theta$ results in a uniform distribution on a circle of radius r.

Solution. We will show now that using the same formula derived in the previous proof, we can describe a scheme to sample uniformly on a circle. We define

$$\begin{aligned} x_1 &= \sqrt{r}\cos\theta, \\ x_2 &= \sqrt{r}\sin\theta. \end{aligned}$$

We derive using the eq. (2.3)

$$p_{x_1,x_2}(x_1, x_2) = p_{r,\theta}(g^{-1}(x_1, x_2))|\det J_{g^{-1}}(x_1, x_2)|.$$

We know that, since we use the same transformation as in Example 2.5, we have the Jacobian $|\det J_{g^{-1}}| = 2$. We can then write

$$p_{x_1,x_2}(x_1, x_2) = \text{Unif}(x_1^2 + x_2^2; 0, 1)\text{Unif}(\arctan(x_2/x_1); -\pi, \pi)2.$$

If we pay attention to the first Uniform distribution in the above formula, we see that this would be 1 when $x_1^2 + x_2^2 < 1$. The second formula is $\arctan$ which takes values on $(-\pi/2, \pi/2)$, which means we always have $\text{Unif}(\arctan(x_2/x_1); -\pi, \pi) = 1/2\pi$. This results

$$p_{x_1,x_2}(x_1, x_2) = \frac{1}{\pi} \qquad \text{for } x_1^2 + x_2^2 < 1$$

and 0 otherwise, which is the uniform distribution within a circle. See Fig. 2.4 for some examples (and alternatives discussed in the class).

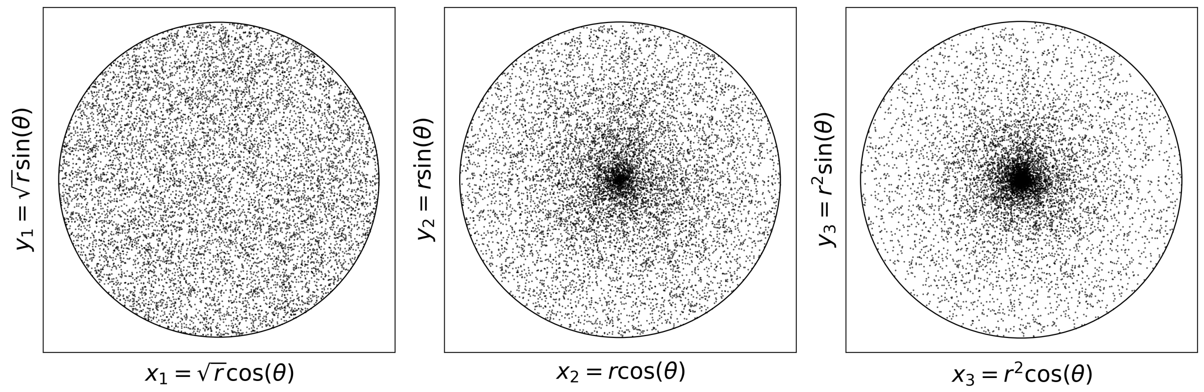


Figure 2.4: On the left, one can see the samples with the correct scaling $\sqrt{r}$. Some other intuitive formulas result in a non-uniform distribution.

We next consider the Gaussian example.

Example 2.7. Describe a transformation method to sample $Y \sim \mathcal{N}(y; \mu, \sigma^2)$ using $X \sim \mathcal{N}(x; 0, 1)$ and prove that $Y \sim \mathcal{N}(y; \mu, \sigma^2)$ as desired.

Solution. This is a simple demonstration of the transformation of random variables formula (in 1-dimension). Let $X \sim \mathcal{N}(x; 0, 1)$ where

$$\mathcal{N}(x; 0, 1) = p_X(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right). \tag{2.4}$$

Now let $Y = \sigma X + \mu$ which is our transformation, i.e.,

$$g(x) = \sigma x + \mu.$$

This is intuitive as we first scale the random variable with σ to increase or decrease its variability (variance) and then add some mean μ. The transformation formula in 1D is simpler:

$$p_Y(y) = p_X(g^{-1}(y)) \left| \frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y} \right| \tag{2.5}$$

where we have the absolute value of the derivative of the inverse function $g^{-1}(y)$. This is easy derive by leaving x alone starting from $y = g(x) = \sigma x + \mu$ and

$$x = \frac{y - \mu}{\sigma} = g^{-1}(y).$$

The derivative is then given by

$$\frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y} = \frac{1}{\sigma}.$$

Then using Eq. (2.3) we obtain

$$p_Y(y) = p_X(g^{-1}(y))\frac{1}{\sigma},$$
$$p_Y(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\frac{1}{\sigma},$$

as $\sigma > 0$ and by using Eq. (2.4) and plugging $x = (y-\mu)/\sigma$. We can already recognize the expression $p_Y(y) = \mathcal{N}(y; \mu, \sigma^2)$.

2.2.3 BOX-MÜLLER TRANSFORM

Box-Müller transform is a related transform to above, but provides a way to sample Gaussians directly from uniforms. In this case, we will just provide the algorithm.

Let $U_1, U_2 \sim \mathrm{Unif}(0,1)$ be independent. Then

$$Z_1 = \sqrt{-2\log U_1}\cos(2\pi U_2),$$
$$Z_2 = \sqrt{-2\log U_1}\sin(2\pi U_2),$$

are independent standard Normal random variables, i.e., $Z_1, Z_2 \sim \mathcal{N}(0,1)$. This transformation is called the Box-Müller transform. This algorithm is the standard way to draw Gaussian samples in practice. For correlated samples, we can use the Cholesky decomposition of the covariance matrix to transform the samples as will be covered in Section 2.5.

2.3 REJECTION SAMPLING

Inversion and the more general transformation method depend on the existence of specific transformations. Given a general $p(x)$, we may not have a specific transformation derived from simpler distributions or an inverse transform. We can still devise sampling methods in this case (in fact, there are many of them). In this section, we will look into one specific class called rejection samplers.

This specific class of methods are powered by a principle: If one can sample (x, y) pairs which are uniformly distributed *under the area* of $p(x)$, then the x-marginal of these samples exactly coming from $p(x)$. We formalise this intuition in the next theorem, adapted from Martino et al. (2018).

Theorem 2.2 (Fundamental Theorem of Simulation). *(Martino et al., 2018, Theorem 2.2) Drawing samples from one dimensional random variable X with a density $\bar{p}(x) \propto p(x)$ is equivalent to sampling uniformly on the two dimensional region defined by*

$$\mathsf{A} = \{(x,y) \in \mathbb{R}^2 : 0 \le y \le \bar{p}(x)\}. \tag{2.6}$$

In other words, if (x', y') is uniformly distributed on A, then x' is a sample from $p(x)$.

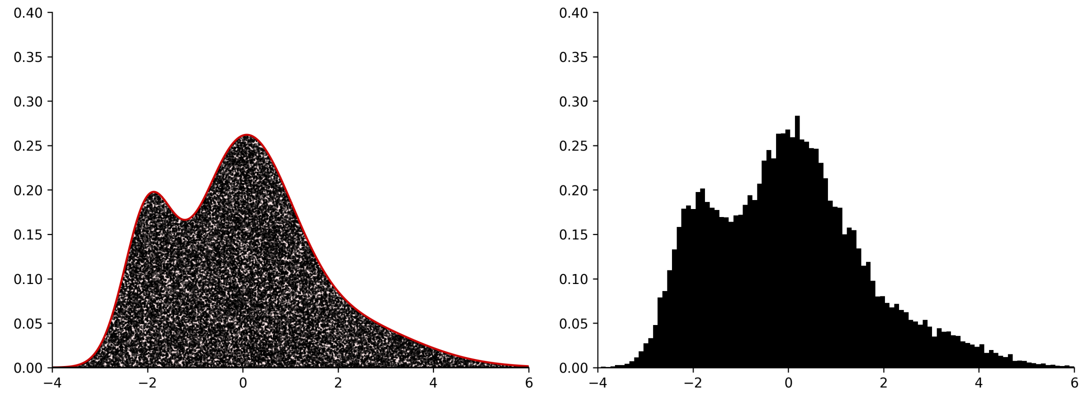


Figure 2.5: On the left, you can see a mixture of Gaussians (we will cover mixture distributions later) and samples uniformly distributed below the curve. Each black dot on under the curve is an (x, y) pair, hence you could denote those samples (X_i, Y_i). On the right, you can see the histogram of the x-marginal, which means, only X_i samples. This is the empirical demonstration of Theorem 2.2.

Proof. Consider the pair (X, Y) uniformly distributed on the region A. We denote their joint density as $q(x, y)$ as

$$q(x,y) = \frac{1}{|\mathsf{A}|}, \qquad \text{for } (x,y) \in \mathsf{A}. \tag{2.7}$$

where $|\mathsf{A}|$ is the area of the set A. We note that

$$p(x) = \frac{\bar{p}(x)}{|\mathsf{A}|}.$$

We use the standard formula for the joint density $q(x, y) = q(y|x)q(x)$. Note that, since (X, Y) is uniform in A, for fixed x, we have

$$q(y|x) = \frac{1}{\bar{p}(x)} \quad \text{for } (x,y) \in \mathsf{A}.$$

We therefore write

$$q(x,y) = q(y|x)q(x) = \frac{q(x)}{\bar{p}(x)} \qquad \text{for } (x,y) \in \mathsf{A}. \tag{2.8}$$

We consider now (2.7) and (2.8) which are both valid on $(x, y) \in \mathsf{A}$. Combining them gives

$$q(x) = \frac{\bar{p}(x)}{|\mathsf{A}|},$$

which means $q(x) = p(x)$. □

An illustration of this theorem can be seen in Fig. 2.5. This theorem extends to cases where we do not have the $p(x)$ exactly, but only have access to an unnormalised version (a very practical issue, as we will see in the following sections).

We will now describe some numerical methods which utilise the fact that if we manage to *uniformly under the area of a curve*, then we can sample from the probability density. Theorem 2.2

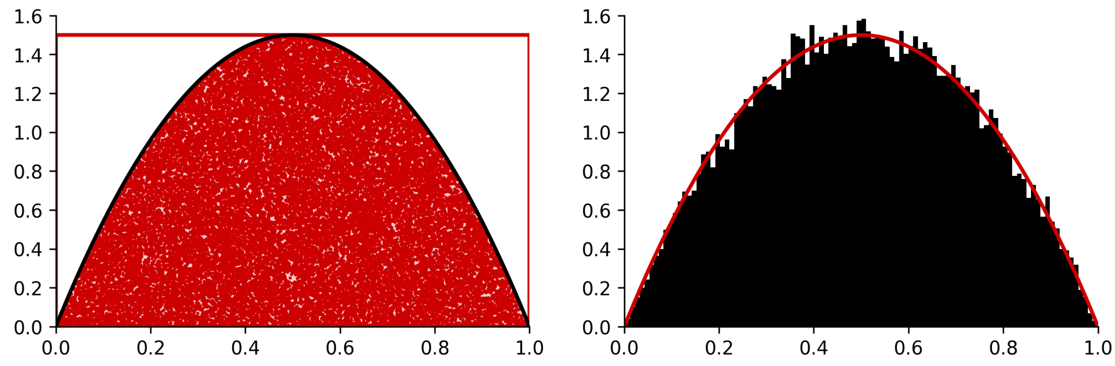


Figure 2.6: On the left, we plot the accepted samples (scattered) under the curve. On the right, we describe the histogram of x-marginal of these samples (so we just keep the first dimension of the two dimensional array).

suggests a quite intuitive sampling procedure: We can sample uniformly under the area of a density (or even an unnormalised nonnegative curve) and obtain samples from the (normalised) probability density by keeping the samples on the x-axis (this is sometimes called the x-marginal). One simple example that does this is described below.

Example 2.8 (Beta density under a box)**.** Consider the Beta density

$$p(x) = \text{Beta}(x; \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1},$$

where $\Gamma(n) = (n-1)!$ for integers. Design a sampler that samples uniformly under the curve $p(x) = \text{Beta}(x; 2, 2)$.

Solution. In order to design a uniform sampler under the area of the $\text{Beta}(2, 2)$, we can use its special properties. For example, the Beta density is defined on $[0, 1]$ which makes it easy to design a uniform sampler. The simplest choice for this is to design a "box" over the density. In order to design this box, we require the maximum of the density

$$p^\star = \max_x \text{Beta}(x; 2, 2) = 1.5.$$

We are of course lucky to have this number, which could be difficult to find analytically in general. In this case, we can design a box $[0, 1] \times [0, p^\star]$ and draw uniform random samples in this box. Let us suggestively denote these samples

$$(X', U') \sim \text{Unif}([0, 1] \times [0, p^\star]).$$

We can then check whether these samples are under the Beta density curve, which can be done by checking:

$$U' \leq p(X'),$$

and *accepting* the sample if this condition holds. Fig. 2.6 shows the result of this procedure together with the histogram.

2.3.1 Rejection sampler

The box example is nice, however, it is not optimal: It might be too inefficient to find a box of that type and for peaky densities, this could be horribly inefficient. We can however identify another probability density we can easily sample from (by choice), denoted $q(x)$, which may cover our target density much better (compared to the box). Rejection sampling is based on this idea: We identify a $q(x)$ to cover our target density $p(x)$. Of course, because $p(x)$ and $q(x)$ are both probability densities, $q(x)$ can never entirely cover $p(x)$. However, it will be sufficient for us if we can find an M such that

$$p(x) \leq Mq(x),$$

so the *scaled* version of $q(x)$ should entirely cover $p(x)$. Depending on the choice of the proposal, the procedure will be much more efficient than simple boxing.

Recall now that using the proposal q and the number M, we would like to sample uniformly under the curve $p(x)$. For this, we can first sample $X' \sim q(x)$ and fix its value $X' = x'$[3]. We are interested in then sampling a uniform between 0 and $Mq(x')$ since $Mq(x)$ dominates $p(x)$, i.e., we then sample $U' \sim \text{Unif}(0, Mq(x'))$. We can then accept the sample if $U' \leq p(x')$. More compactly:

- Sample $X' \sim q(X')$
- $U' \sim \text{Unif}(0, Mq(X'))$
- Accept if

$$U' \leq p(X').$$

This is exactly drawing a (X', U') pair and accepting the sample if it is under the curve of $p(x)$. By Theorem 2.2, this samples from the correct distribution!

The general implementation usually differs from the above procedure and takes the following form. The usual implementations first generate $X' \sim q(x)$ and then in order to implement the *Accept* step, generate $U \sim \text{Unif}(0, 1)$ and accept the sample if

$$U \leq a(x') = \frac{p(x')}{Mq(x')}.$$

This is what *accept with probability* $a(x')$ means. A closer look reveals, we could also write this (by playing with the above inequality)

$$Mq(x')U \leq p(x').$$

In order words, the lhs of this inequality is a uniform random variable multiplied by $Mq(x')$, so we could define $U' = Mq(x')U$ as

$$U' \sim \text{Unif}(0, Mq(x')),$$

since $U \sim \text{Unif}(0, 1)$. This shows the equivalence of the two procedures. Let us describe the conceptual algorithm.

[3]This is usual in probability: Capital letters are *random variables*, it is better to fix their values *after* they are sampled (now deterministic).

- Generate $X_i' \sim q(x)$
- Accept with probability

$$a(X_i') = \frac{p(X_i')}{Mq(X_i')} \leq 1. \tag{2.9}$$

So far we have written a few different versions of the method. Implementation however is made according to Algorithm 3 as mentioned above.

Algorithm 3 Pseudocode for rejection sampling

```
1: Input: The number of iterations $n$ and scaling factor $M$.
2: for $i = 1, \dots, n$ do
3:     Generate $X' \sim q(x')$
4:     $U \sim \text{Unif}(0, 1)$
5:     if $U \leq \frac{p(X')}{Mq(X')}$ then
6:         Accept $X'$                ▷ This should record the sample with other accepted samples
7:     end if
8: end for
9: Return accepted samples
```

REJECTION SAMPLING WITH UNNORMALISED DENSITIES

So far, we have assumed that we have access to the density we want to sample from $p(x)$: We could evaluate it, hence use it for rejection under the curve. However, imagine we know a probability density *up to a normalising constant*. This is one of the most common problems in computational statistics (Google: Estimating normalising constants) and it arises in multiple situations which will be described shortly.

We denote the unnormalised density associated to $p(x)$ as $\bar{p}(x)$. Usually, we write

$$p(x) = \frac{\bar{p}(x)}{Z},$$

where $Z = \int \bar{p}(x) \mathrm{d}x$. This is typical in in physics, engineering, machine learning, and optimisation, we do not start from densities, instead one defines:

$$p(x) \propto e^{-f(x)},$$

for some function f (which is generally called a *potential*). f usually comes from a rule which determines how probability mass should be spread (e.g. a multimodal function). However, in order to convert this into probabilities, we need normalise $e^{-f(x)}$.

The surprising fact about the rejection sampling is that it works *in the same way* for unnormalised densities $\bar{p}$. In other words, more generally, Theorem 2.2 holds for $\bar{p}$: As long as we sample uniformly under $\bar{p}$, we can obtain x-marginal which would be distributed w.r.t. $p(x)$ (Martino et al., 2018). This gives rejection samplers a massive advantage. Of course, needless to say, in this case, one should ensure that

$$\bar{p}(x) \leq Mq(x),$$

i.e. the *unnormalised* density is covered by our scaled proposal $Mq(x)$. We describe the algorithm for the unnormalised case in Algorithm 4.

Algorithm 4 Pseudocode for rejection sampling without normalising constants

```
1: Input: The number of iterations $n$ and scaling factor $M$.
2: for $i = 1, \ldots, n$ do
3:     Generate $X' \sim q(x')$
4:     $U \sim \text{Unif}(0, 1)$
5:     if $U \leq \frac{\bar{p}(X')}{Mq(X')}$ then
6:         Accept $X'$                ▷ This should record the sample with other accepted samples
7:     end if
8: end for
9: Return accepted samples
```

2.3.2 ACCEPTANCE RATE

An important aspect of this algorithm is the concept of *acceptance rate*, that is, the ratio of the number of accepted samples vs. the number of total samples. When the algorithm has a low acceptance rate, this hints that the proposal is poorly designed and most of the computational effort (sampling) goes unused and wasted[4].

Let us consider the normalised case first with a probability density $p(x)$. Note that the acceptance probability is a function of X' and defined as $a(X')$ in (2.9). We accept a sample when $U \leq a(X')$, in other words, when

$$U \leq \frac{p(X')}{Mq(X')},$$

where $X' \sim q(x')$. Let us denote the probability of acceptance (acceptance rate) as $\hat{a}$. This quantity is formalised below.

Proposition 2.1. *When the target density $p(x)$ is normalised and M is prechosen, the acceptance rate is given by*

$$\hat{a} = \frac{1}{M},$$

where $M > 1$ in order to satisfy the requirement that q covers p. For an unnormalised target density $\bar{p}(x)$ with the normalising constant $Z = \int \bar{p}(x)\mathrm{d}x$, the acceptance rate is given as

$$\hat{a} = \frac{Z}{M}.$$

Proof. We will first prove

$$\hat{a} = \mathbb{E}[a(X')] = \frac{1}{M} \tag{2.10}$$

[4]Acceptance rate will also be a crucial notion when we later study Markov chain Monte Carlo (MCMC) methods.

in the normalised case. Similarly, we will also prove

$$\hat{a} = \mathbb{E}[a(X')] = \frac{Z}{M} \tag{2.11}$$

for the unnormalised case where we use $\bar{p}(x)$ instead of $p(x)$.

For the first fact, we can prove (2.10) by noting that

$$\hat{a} = \mathbb{P}(\mathsf{accept}) = \mathbb{P}\left(U \leq \frac{p(X')}{Mq(X')}\right),$$

where the randomness here is w.r.t. U and X' jointly. We know however that, for a given $X' = x'$, we accept with the following probability

$$\mathbb{P}(\mathsf{accept}|X' = x') = \mathbb{P}\left(U \leq \frac{p(x')}{Mq(x')}\right) = \int_0^{p(x')/Mq(x')} \mathrm{d}u = \frac{p(x')}{Mq(x')} = a(x').$$

This means that we can compute the unconditional acceptance probability as

$$\begin{aligned}\mathbb{P}(\mathsf{accept}) &= \int \mathbb{P}(\mathsf{accept}|X' = x')q(x')\mathrm{d}x', \\ &= \mathbb{E}[a(X')].\end{aligned}$$

We can then easily show that

$$\begin{aligned}\mathbb{E}[a(X')] &= \int a(x')q(x')\mathrm{d}x' = \int \frac{p(x')}{Mq(x')}q(x')\mathrm{d}x' \\ &= \frac{1}{M}\int p(x')\mathrm{d}x' = \frac{1}{M}.\end{aligned}$$

For the unnormalised case, we can prove (2.11) with a similar argument above as

$$\begin{aligned}\hat{a} = \mathbb{E}[a(X')] &= \int a(x')q(x')\mathrm{d}x' = \int \frac{\bar{p}(x')}{Mq(x')}q(x')\mathrm{d}x' \\ &= \int Z\frac{p(x')}{Mq(x')}q(x')\mathrm{d}x' = \frac{Z}{M}\int p(x')\mathrm{d}x' \\ &= \frac{Z}{M}.\end{aligned}$$

□

Remark 2.2. Note the difference between two quantities a and $\hat{a}$. While $a(x)$ is the *acceptance ratio* $p(x)/Mq(x)$ (or the unnormalised version), $\hat{a}$ is the expectation of this quantity, called *acceptance rate*.

It can be seen that smaller M is theoretically useful for us as it will mean higher acceptance rates. Consider the following example.

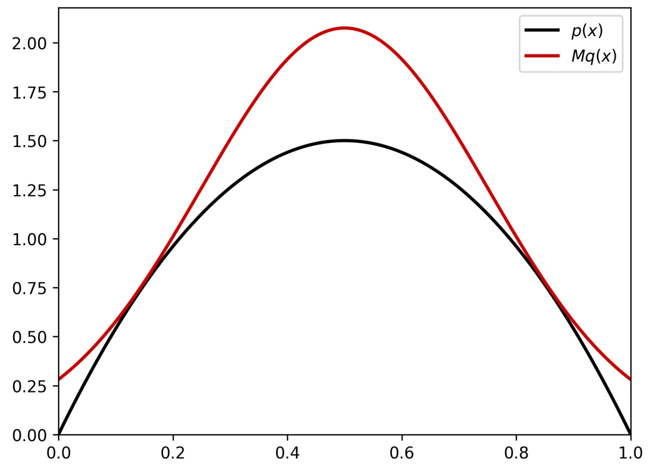


Figure 2.7: A better proposal for $p(x) = \text{Beta}(2, 2)$

Example 2.9 (Beta$(2, 2)$ density)**.** We can go back to our example Beta$(2, 2)$ density in Example 2.8. Instead of the box, we can now choose

$$q(x) = \mathcal{N}(x; 0.5, 0.25),$$

and $M = 1.3$ (this is optimised visually by plotting – we will compute these quantities explicitly below). This will result in the graph shown in Fig. 2.7. Compare the acceptance rate of this algorithm with the box example and demonstrate this numerically.

Solution. We can see now that in the box example, we chose $M_{\text{box}} = p^\star = 1.5$. By visual inspection for this example, for the Gaussian case, we chose $M_{\text{Gauss}} = 1.3$. We can now compute the acceptance rate for both cases. For the box example, we have

$$\hat{a}_{\text{box}} = \frac{1}{M_{\text{box}}} = \frac{1}{1.5} = 0.67,$$

and for the Gaussian case, we have

$$\hat{a}_{\text{Gauss}} = \frac{1}{M_{\text{Gauss}}} = \frac{1}{1.3} = 0.77.$$

These must be the precise numbers that the numerical simulations will give us.

Another example can be seen as follows.

Example 2.10 (Truncated Densities)**.** Describe a rejection sampler for a Truncated Gaussian target:

$$\bar{p}(x) = \mathcal{N}(x; 0, 1)\mathbf{1}_{|x|\leq a}(x).$$

Solution. The truncated densities arise in a number of applications where we may want to model something we know with a probability density $p(x)$ we are familiar with. However, it could also be the case that this variable X has strong constraints (e.g. positivity or boundedness). For example, we could consider an age distribution could be restricted this way with hard constraints. Imagine a Gaussian density $\mathcal{N}(x; 0, 1)$ and suppose we are interested in sampling this density between $[-a, a]$. We can write this truncated normal density as

$$p(x) = \frac{\bar{p}(x)}{Z} = \frac{\mathcal{N}(x; 0, 1)\mathbf{1}_{|x|\leq a}(x)}{\int_{-a}^{a} \mathcal{N}(y; 0, 1)\mathrm{d}y}. \tag{2.12}$$

Here are a few important things about this equation: $\mathbf{1}_A(x)$ denotes a function where

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise.} \end{cases}$$

Note that now we have access to our density evaluation in an unnormalised way: We can evaluate $\bar{p}(x)$ which equals to $\mathcal{N}(x; 0, 1)$ if $-a \leq x \leq a$ and to 0 otherwise. Rejection sampling is optimal for this task. Note here that, we can choose

$$q(x) = \mathcal{N}(x; 0, 1)$$

anyway, and we have $\bar{p}(x) \leq q(x)$ (i.e. we can take $M = 1$). Note a few interesting things for this case: First of all, since $p(x)$ is zero outside $[-a, a]$ and $\bar{p}(x)/Mq(x) = 1$ if $x \in [-a, a]$, we can simply reject any sample that is out of bounds and accept if they are in the bounds (without needing to sample U at all). Secondly, note that, this is one of the rare cases where we have $Z < 1$. Note that we can very intuitively also calculate the acceptance rate:

$$\hat{a} = \frac{Z}{M} = Z$$

where $Z = \int_{-a}^{a} \mathcal{N}(y; 0, 1)\mathrm{d}y$ as given in the density (2.12). It is very intuitive that the acceptance rate is this integral, as we literally accept a Gaussan sample from $q(x) = \mathcal{N}(x; 0, 1)$ if it falls into $[-a, a]$, the probability of sample falling into this interval is given by the integral $\int_{-a}^{a} \mathcal{N}(y; 0, 1)\mathrm{d}y = Z$.

2.3.3 DESIGNING THE OPTIMAL REJECTION SAMPLER

In the above examples, we have seen that choosing the right proposal $q(x)$ and M can improve the sampler quality significantly. In this section, we will discuss how to choose the optimal proposal $q(x)$ and M for a given target density $p(x)$ (or $\bar{p}(x)$).

CHOOSING M

In the above example, we have seen that choosing M is crucial for the acceptance rate. It is easy to see that we should choose M such that $Mq(x) \geq p(x)$ for all x. To choose smallest such M, we should find the number $M^\star$ such that

$$M_\star = \sup_x \frac{p(x)}{q(x)}.$$

This will ensure that $M_\star q(x) \geq p(x)$ for all x. This is the optimal choice of M and we can see that this is the smallest M that covers $p(x)$. Needless to say, for the unnormalised case, we just replace p with $\bar{p}$ in the above formula.

Example 2.11 (Choosing M)**.** Let our target be

$$\bar{p}(x) = e^{-x^2/2}$$

i.e., an unnormalised Gaussian and our proposal be

$$q(x) = \frac{1}{\pi}\frac{1}{1+x^2}.$$

Compute M for the optimal rejection sampler.

Solution. We need to compute

$$M = \sup_x \frac{\bar{p}(x)}{q(x)},$$

as usual. How? For this, let $R(x) = \bar{p}(x)/q(x)$ and note that we will first need to find

$$x^\star = \operatorname*{argmax}_x R(x) = \operatorname*{argmax}_x \log R(x),$$

as $\log$ is monotonic. Then we can find $M = R(x^\star)$. Let us now compute

$$\log R(x) = \log \bar{p}(x)/q(x) = -\frac{x^2}{2} + \log(1+x^2) + \log(1/\pi)$$

and find the roots by taking the derivative and setting it to zero

$$\frac{\mathrm{d}}{\mathrm{d}x}\log R(x) = -x + \frac{2x}{1+x^2} = 0$$
$$x = 0, \pm 1,$$

we see that we have three roots to decide. Which one is the maximum? To look at the answer, we need to check second derivatives. We compute the second derivative

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log \bar{p}(x)/q(x) = -1 + \frac{2(1-x^2)}{(1+x^2)^2}$$

- When $x = 0$, the second derivative is positive - which means $x = 0$ is a minimum.
- When $x = \pm 1$, the second derivative is negative - which means $x = \pm 1$ is a maximum.
- $x^\star = \pm 1$.

So we have

$$M = \frac{\bar{p}(1)}{q(1)} = 2\pi e^{-1/2}.$$

OPTIMISING THE PROPOSAL

Let us in this section parameterise our proposal q_θ where θ is the parameter of the proposal density q_θ. We have seen above that the choice of M will obviously depend on q_θ so we can denote it as M_θ. Since, in fact, we would like to achieve the optimal M_θ such that

$$p(x) \leq M_\theta q_\theta(x).$$

We have seen that we can choose M_θ as

$$M_\theta = \sup_x \frac{p(x)}{q_\theta(x)}.$$

Note that this is a function of θ, therefore allows us now to *optimise* our proposal to maximise the acceptance rate. Therefore, we can now solve the problem

$$\theta_\star = \operatorname*{argmax}_\theta \frac{1}{M_\theta}.$$

We always perform this optimisation in the $\log$-space[5] as the quantities we will obtain are more tractable, therefore, the problem often becomes

$$\theta_\star = \operatorname*{argmin}_\theta \log M_\theta,$$

as $\log(1/M_\theta) = -\log M_\theta$.

Example 2.12. (A numerical example from Yıldırım (2017)) Say we are interested in sampling

$$X \sim \text{Gamma}(\alpha, 1),$$

with $\alpha > 1$. The density is given by

$$p(x) = \frac{x^{\alpha-1}e^{-x}}{\Gamma(\alpha)}, \quad x > 0.$$

[5]We denote $\ln$ with $\log$, so all logaritms are w.r.t. to natural base.

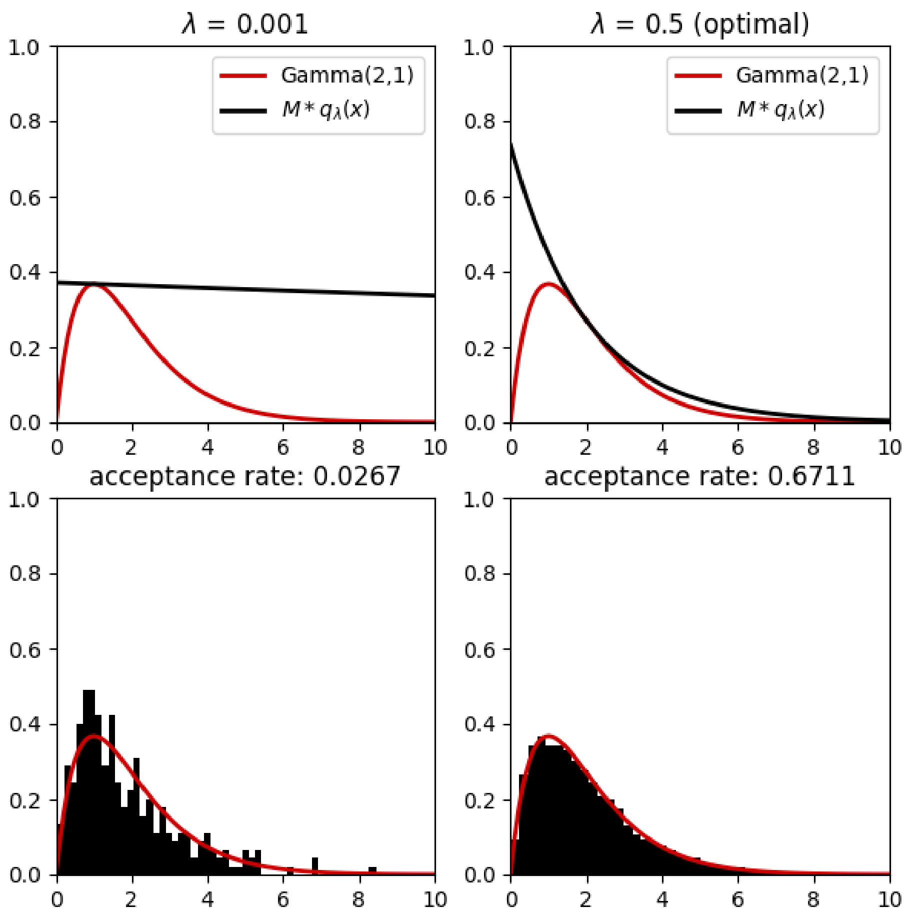


Figure 2.8: Two rejection sampling procedures for Example 2.12 with $\lambda = 0.001$ and optimal $\lambda = 1/\alpha$ (as derived in the example) for $n = 10000$.

As a *proposal*, let us choose exponential

$$q_\lambda(x) = \text{Exp}(x; \lambda) = \lambda e^{-\lambda x}, \quad x > 0$$

with $0 < \lambda < 1$ (for $\lambda > 1$, the ratio p/q_λ is unbounded). Derive the optimal rejection sampler for this problem.

Solution. We should ensure that $p(x) \leq Mq(x)$, therefore, the standard choice for M_λ is to compute

$$M_\lambda = \sup_x \frac{p(x)}{q_\lambda(x)}.$$

We know that

$$\frac{p(x)}{q_\lambda(x)} = \frac{x^{\alpha-1}e^{(\lambda-1)x}}{\lambda\Gamma(\alpha)}.$$

In order to compute M_λ, in practice, one needs to first find

$$x^\star = \operatorname*{argmax}_x \frac{p(x)}{q_\lambda(x)},$$

and then we can identify that $M_\lambda = p(x^\star)/q_\lambda(x^\star)$ for fixed λ. Denote $R_\lambda(x) = p(x)/q(x)$. To find such $x^\star$, we can compute

$$x^\star = \operatorname{argmax} \log p(x) - \log q_\lambda(x),$$

as log is monotonic. We can then compute

$$\log R(x) = \log p(x) - \log q_\lambda(x) = (\alpha - 1)\log x - x + (\lambda - 1)x - \log \lambda - \log \Gamma(\alpha).$$

We then take the derivative of this

$$\frac{\mathrm{d} \log R(x)}{\mathrm{d}x} = \frac{\alpha - 1}{x} + \lambda - 1,$$

and set it to zero, which leads to

$$x^\star = \frac{(\alpha - 1)}{(1 - \lambda)}.$$

Placing $x = x^\star$ in the ratio $p(x)/q_\lambda(x)$, we obtain

$$M_\lambda = \frac{\left(\frac{\alpha-1}{1-\lambda}\right)^{\alpha-1} e^{-(\alpha-1)}}{\lambda\Gamma(\alpha)}.$$

This leads to the acceptance probability

$$\frac{p(x)}{M_\lambda q_\lambda(x)} = \left(\frac{x(1-\lambda)}{\alpha - 1}\right)^{\alpha-1} e^{(\lambda-1)x+\alpha-1}.$$

Now, we have to minimise M_λ with respect to λ so that $\hat{a} = 1/M_\lambda$ would be maximised. It is easy to show that (show) M_λ is minimised by

$$\lambda^\star = \frac{1}{\alpha}.$$

Plugging this and computing

$$M_{\lambda^\star} = \frac{\alpha^\alpha e^{-(\alpha-1)}}{\Gamma(\alpha)}.$$

So we designed our *optimised* rejection sampler. In order to sample from $\Gamma(\alpha, 1)$, we perform

- Sample $X' \sim \text{Exp}(1/\alpha)$ and $U \sim \text{Unif}(0, 1)$
- If

  $$U \leq (x/\alpha)^{\alpha-1} e^{(1/\alpha-1)x+\alpha-1},$$

  accept X', otherwise start again.

We can see the results of this algorithm in Fig. 2.8.

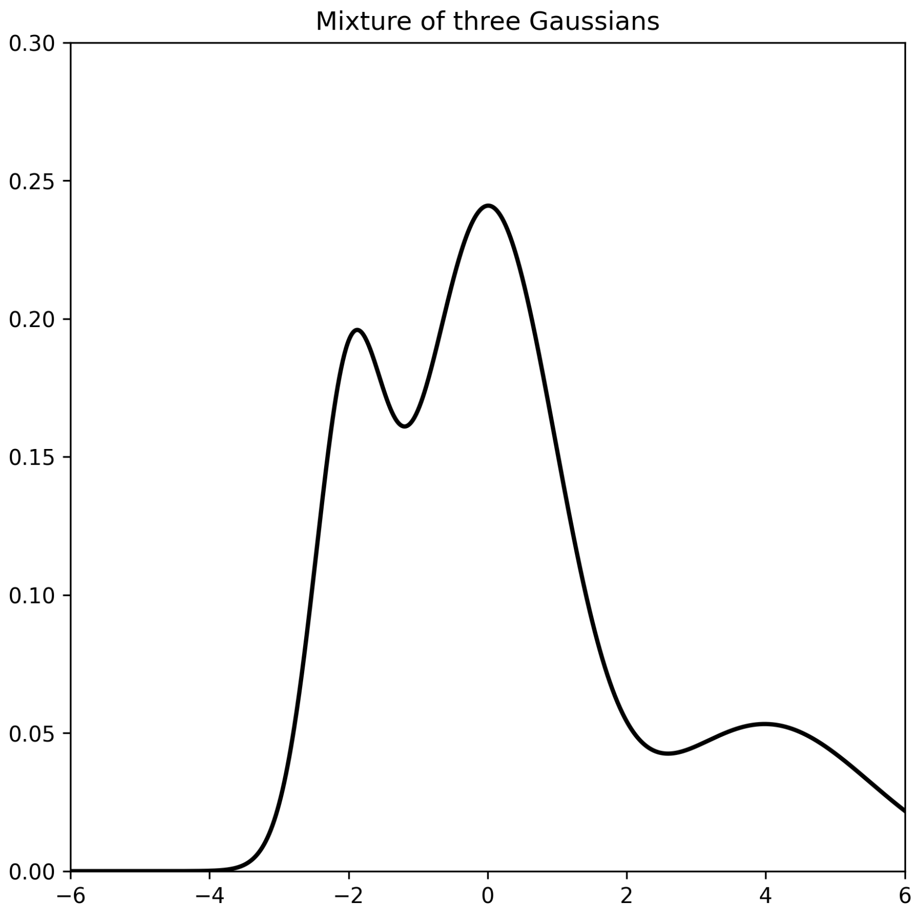


Figure 2.9: The density of a mixture of three Gaussians: $p(x) = \sum_{k=1}^{3} w_k \mathcal{N}(x; \mu_k, \sigma_k^2)$ with $\mu_1 = -2, \mu_2 = 0, \mu_3 = 4, \sigma_1 = 0.5, \sigma_2 = 1, \sigma_3 = 0.5, w_1 = 0.2, w_2 = 0.6, w_3 = 0.2$.

2.4 COMPOSITION

When the probability density $p(x)$ can be expressed in a composition of operations, we can still sample from such densities straightforwardly, albeit it may look complex at first look. In this section, we focus on *mixture densities*, i.e., densities that can be written as a weighted mixture of two probability densities. These objects are used to model subpopulations in a statistical population, modelling experimental error (e.g. localised in different regions), and heterogeneous populations. We will start from a discrete mixture and then will discuss the continuous case.

2.4.1 SAMPLING FROM DISCRETE MIXTURE DENSITIES

To start simple, consider the following probability density

$$p(x) = w_1 q_1(x) + w_2 q_2(x),$$

where $w_1 + w_2 = 1$ and q_1 and q_2 are probability densities. It is straightforward to verify that $p(x)$ is also a density

$$\begin{aligned}\int p(x)\mathrm{d}x &= w_1 \int q_1(x)\mathrm{d}x + w_2 \int q_2(x)\mathrm{d}x \\ &= w_1 + w_2 \\ &= 1.\end{aligned}$$

An example can be seen from Fig. 2.9. We can generalise this idea and define a general mixture distribution

$$p(x) = \sum_{k=1}^{K} w_k q_k(x),$$

with k mixtures. Sampling from such distributions are extremely easy with the techniques we know. We first sample from the probability mass function defined by weights: $p(k) = w_k$ where $\sum_{k=1}^{K} p(k) = 1$ (using inversion as we learned). This gives us an *index* $k \sim p(k)$, then we sample from associated density $X' \sim q_k(x)$, which gives us a sample from the mixture. For example,

Algorithm 5 Sampling discrete mixtures

```
1: Input: The number of samples $n$.
2: for $i = 1, \ldots, n$ do
3:     Generate $k \sim p(k)$
4:     Generate $X_i \sim q_k(x)$
5: end for
```

sampling a mixture of Gaussians is easy: Sample $k \sim p(k)$ from the PMF consists of weights w_k, then sample from the selected Gaussian.

2.4.2 SAMPLING FROM CONDITIONAL DENSITIES

Before we move on to the continuous mixture case, we clarify how one can sample from conditional distributions, denoted, generally, as $p(y|x)$. In this case, this is a density for every fixed x, therefore conditioned on x, sampling is same as the any other sampling problem. For example, consider

$$p(y|x) = \mathcal{N}(y; x, 1),$$

where the mean (parameter) of the Gaussian is denoted within the density as conditioned. This notation is useful if one assumes x is also random (will see later). However, for fixed x, the sampling is business usual:

$$y \sim p(y|x) = \mathcal{N}(y; x, 1),$$

is sampling a Gaussian with a fixed mean x.

2.4.3 SAMPLING FROM JOINT DISTRIBUTIONS

Sampling from a joint distribution $p(x, y)$ sounds straightforward but it might be still not obvious. Assume, we would like to draw

$$X, Y \sim p(x, y) \tag{2.13}$$

e.g., a two-dimensional sample from 2D Gaussian. It is often the case that the standard factorisation of joint densities

$$p(x, y) = p(y|x)p(x),$$

can be used. In order to realise (2.13), one can employ

$$\begin{aligned} X &\sim p(x), \\ Y|X = x &\sim p(y|x). \end{aligned}$$

Note the notation which implies that things should be done in this order. Once X is sampled, then it is fixed $X = x$. After that, Y is sampled conditioned on that specific x sample.

In particular, the idea can be generalised for n variables if one knows the full conditionals. For example, consider a joint distribution $p(x_1, \ldots, x_n)$, then any joint distribution of n variables satisfy the following.

$$p(x_1, \ldots, x_n) = p(x_n|x_{n-1}, \ldots, x_1)p(x_{n-1}|x_{n-2}, \ldots, x_1) \cdots p(x_2|x_1)p(x_1).$$

Therefore, simulating from a joint distribution can be done

$$\begin{aligned} X_1 &\sim p(x_1) \\ X_2|X_1 = x_1 &\sim p(x_2|x_1) \\ X_3|X_1 = x_1, X_2 = x_2 &\sim p(x_3|x_2, x_1) \\ &\vdots \\ X_n|X_1 = x_1, X_2 = x_2, \ldots, X_{n-1} = x_{n-1} &\sim p(x_n|x_1, \ldots, x_{n-1}). \end{aligned}$$

Of course the difficulty with this is that, it is often impossible to know these conditional distributions described above.

Remark 2.3. This idea can be taken to great generalisation, in fact, it is often the core of complex simulations. The core idea of probabilistic modelling is to factorise (assuming independence) some complex joint distribution $p(x_1, \ldots, x_n)$ with respect to the modelling assumptions. Simulation methods can then be used to sample these variables in the order that is assumed in the model and generate synthetic data.

2.4.4 SAMPLING FROM CONTINUOUS MIXTURES OR MARGINALISATION

It is a common case that a density can be written as an integral, instead of a sum (as in the discrete mixture case). Consider the fact that

$$p(y) = \int p(x, y) \mathrm{d}x,$$

for any joint density. This operation is called *marginalisation* and it is often of interest to compute marginal densities (and of course sampling from them).

For example, using the formula $p(x, y) = p(y|x)p(x)$ and given a conditional density $p(y|x)$ and $p(x)$, we can derive

$$p(y) = \int p(x, y) \mathrm{d}x = \int p(y|x)p(x) \mathrm{d}x.$$

Surprisingly enough, sampling from y is pretty straightforward: Sample from the joint $p(x, y)$ using the method above (i.e. $X \sim p(x)$ and $Y|X = x \sim p(y|x)$), then just keep Y samples. They will approximate $p(y)$!. Let us see an example.

Example 2.13. Let

$$p(x) = \mathcal{N}(x; \mu, \sigma^2)$$

and

$$p(y|x) = \mathcal{N}(y; x, 1).$$

Then it can be shown that (we will do this in future exercises):

$$p(y) = \mathcal{N}(y; \mu, \sigma^2 + 1).$$

Numerically verify that this is the case.

Solution. This can be verified by implementing two procedures. First,

- Sample $X \sim \mathcal{N}(x; \mu, \sigma^2)$,
- Sample $Y|X = x \sim \mathcal{N}(y; x, 1)$

and comparing resulting Y samples to

- $Y \sim p(y) = \mathcal{N}(y; \mu, \sigma^2 + 1)$.

Online companion contains the result.

Another example can be seen as follows.

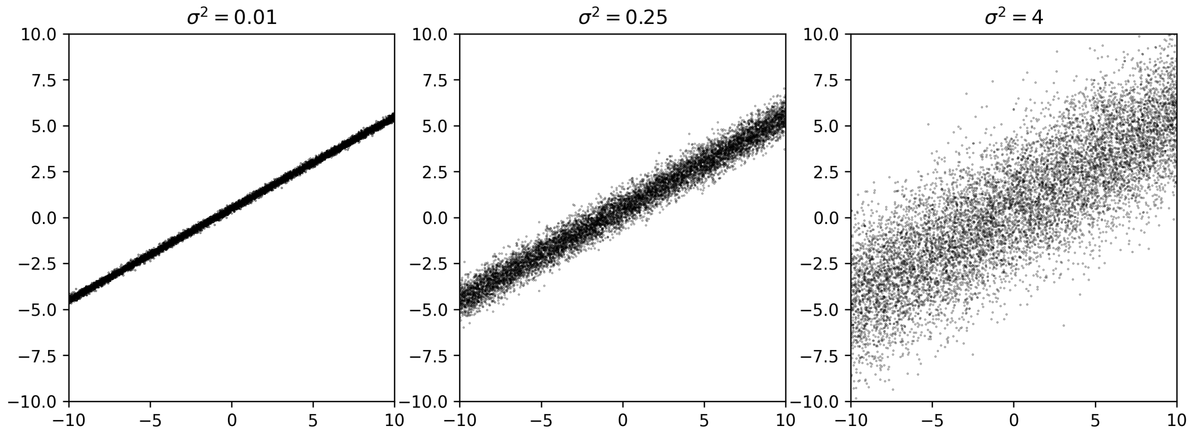


Figure 2.10: The data simulated from (2.15)–(2.16) using $a = 0.5$ and $b = 0.5$ with three different values for σ^2. As can be seen from the figures, the generated (x, y) pairs exhibit a clear linear relationship (as intended) with variance changing depending on our modelling choice.

Example 2.14 (Linear Model)**.** Linear models are of utmost importance in many fields of science. Describe a method to simulate (x, y) pairs that have a linear relationship.

Solution. We know that we can sample $x, y \sim p(x, y)$ by sampling $x \sim p(x)$ and $y|x \sim p(y|x)$ from the last chapter. We will now use this for a linear example.

To start intuitively, a typical linear relationship is described as

$$y = ax + b, \tag{2.14}$$

which describes a line where a is the slope and b is the intercept. In order to obtain a probabilistic model and generate data, we have to simulate both x and y variables. Since, from the equation, it is clear that y is generated *given* x, we should start from defining x. Now this depends on the application. For example, x can be a variable that may be uniform or a Gaussian. We denote its density as $p(x)$. The typical task is also to formulate $p(y|x)$. The linear equation suggests a deterministic relationship, however, real data often contains *noise*. To generate realistic data, we will instead assume

$$y = ax + b + n$$

where $n \sim \mathcal{N}(0, \sigma^2)$ is *noise* (often with small σ^2. Note that, given noise is zero mean and $ax + b$ is a deterministic number (given x), we can then write our full model

$$p(x) = \text{Unif}(x; -10, 10) \tag{2.15}$$
$$p(y|x) = \mathcal{N}(y; ax + b, \sigma^2). \tag{2.16}$$

where we chose our $p(x)$ distribution to be uniform on $[-10, 10]$. As a result, we have a full model to simulate variables with a linear relationship

$$X_i \sim p(x),$$
$$Y_i|X_i = x_i \sim p(y|x_i),$$

Algorithm 6 Sampling Multivariate Gaussian

```
1: Input: The number of samples $n$.
2: for $i = 1, \dots, n$ do
3:     Compute $L$ such that $\Sigma = LL^\top$. (Cholesky decomposition)
4:     Draw $d$ univariate independent normals $v_k \sim \mathcal{N}(0,1)$ to form the vector $v = [v_1, \dots, v_d]^\top$
5:     Generate $x_i = \mu + Lv$.
6: end for
```

where $p(x)$ could be a uniform, Gaussian, truncated Gaussian etc. depending on the nature of the modelled variable. The results of this generation can be seen in the scatter plot in Fig. 2.10.

2.5 SAMPLING MULTIVARIATE DENSITIES

Finally, we will discuss the sampling methods for multivariate distributions. As we saw earlier, a multivariate density is nothing but a joint density in d dimensions, i.e., can be defined as $p(x_1, \dots, x_d)$. The techniques mentioned in Sec. 2.4.3 might be used (sampling from conditionals) if they are known. Also, what we have seen as samplers, e.g., rejection sampling generalizes to many dimensions as you have already seen from the circle example. But, of course, most of the time, they are not known and very general independent sampling techniques are available, see Martino et al. (2018, Chapter 6). We will only cover one specific case.

2.5.1 SAMPLING A MULTIVARIATE GAUSSIAN

Define $x \in \mathbb{R}^d$, a multivariate Gaussian:

$$p(x) = (2\pi)^{-\frac{d}{2}} |\det \Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right),$$

where $\mu \in \mathbb{R}^d$ is the mean vector and $\Sigma \in \mathbb{R}^{d \times d}$ is a $d \times d$ symmetric positive definite matrix. Recall that, in the univariate case, $Y = \mu + \sigma X$ (where μ, σ are scalars) gave us a sample from $\mathcal{N}(\mu, \sigma^2)$. The same idea works here, however, since now we have the covariance instead of variance, we need to find a notion of a "square-root" of the covariance matrix Σ. In short, let $X \sim \mathcal{N}(x; 0, I)$ where now $X \in \mathbb{R}^d$ and the mean is a zero vector, and finally, I is the identity matrix. Such a multivariate Gaussian can be drawn by drawing each entry independently. We then can sample $Y \sim \mathcal{N}(y; \mu, \Sigma)$ where $\mu \in \mathbb{R}^d$ the mean vector and $\Sigma \in \mathbb{R}^{d \times d}$ the symmetric covariance matrix, as

$$Y = \Sigma^{1/2} X + \mu.$$

The computation of $\Sigma^{1/2}$ is done using a Cholesky decomposition[6]. The algorithm is provided in Algorithm 6.

[6]You do not need to know how to implement or compute this, it is perfectly fine to use `numpy.linalg.cholesky`.

3 PROBABILISTIC MODELLING AND INFERENCE

In this chapter, we will cover probabilistic modelling in more detail and then talk about inference. We will also review probability basics and a large range of applications the Bayesian viewpoint enables.

3.1 INTRODUCTION

In the previous chapter, we have seen how to generate data from a probabilistic model. Despite we have only simulated from a linear model as an example, the idea is general. We will see more about simulating models in other parts of the course. We have seen that

$$X_i \sim p(x), \tag{3.1}$$

$$Y_i | X_i = x_i \sim p(y|x_i), \tag{3.2}$$

generates the data according to the model $p(x, y) = p(y|x)p(x)$. It is important to stress that this can describe a very general situation: x variable can be multivariate (and even be time dependent), and y can describe any other process. We will see, though, that in *Bayesian modelling* (I use it simultaneously with probabilistic modelling), x generally denotes the *latent (hidden) states* or *parameters* of a model (or both) . The variable y typically denotes the *observed data*. So seeing the model (3.1) as a generative model, simulating from it can be seen as a way of generating synthetic data[1].

3.2 THE BAYES RULE AND ITS USES

In this section, we will discuss the Bayes rule in depth and its uses. The Bayesian formula is at the heart of many probabilistic modelling approaches. We start with the definition of the Bayes rule.

[1]This is a big deal in industry. Search for example for *synthetic data startups*.

Definition 3.1 (Bayes Theorem)**.** *Let X and Y be random variables with associated probability density functions $p(x)$ and $p(y)$, respectively. The Bayes rule is given by*

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \tag{3.3}$$

Note that the formula holds for continuous random variables as well as discrete random variables. Its importance comes from the fact that it provides us a natural way to incorporate or synthesise data into a probabilistic model. In this interpretation, we have three key concepts.

- **Prior:** In the formula (3.3), $p(x)$ is called the prior probability of X. Here X can be interpreted as a parameter of $p(y|x)$ or a hidden (unobserved) variable. The probability distribution $p(x)$ encodes our prior knowledge about this variable we cannot observe directly. This could be simple constraints, a distribution dictated by a real application (e.g. a physical variable can be only positive). In time series applications, $p(x)$ can be the distribution over an entire time series, it can even encode physical laws.
- **Likelihood:** $p(y|x)$ is called the likelihood of Y given X. This is the probability model of the process of *observation* – in other words, it describes how the underlying parameter or hidden variable is observed. For example, if Y is the number of observed cases of a disease in a population, then $p(y|x)$ is the probability of observing y cases given that the true number of cases is x.
- **Posterior:** $p(x|y)$ is called the posterior distribution of X given $Y = y$. This is the *updated* probability distribution after we see y observation and updated our prior knowledge $p(x)$ into $p(x|y)$.

We will see a number of examples where these quantities make sense.

Remark 3.1. Note the difference between *simulation* and *inference*. We can write down our *model* (sometimes we will call the forward model) $p(x)$ and $p(y|x)$ to describe the *data generation* process and can generate toy (synthetic) data with it as we have seen. But the essential goal of Bayes rule (also called Bayesian or probabilistic inference) is to *infer* the posterior distribution conditioned on already *observed* data. In other words, we can use a probabilistic model for two purposes:

- *Simulation:* We can generate synthetic data with a probabilistic model.
- *Inference:* We can infer the posterior distribution (implied by the model structure we impose) of a parameter or hidden variable given observed data.

Let us see the Bayes' rule on a discrete example.

Example 3.1. Suppose we have two fair dice, each with six faces. Suppose that we throw both dice and observe the sum as $y = 9$. Derive the posterior distribution of the first die X_1 and the second die X_2 given $Y = 9$.

Solution. Let us first write down the joint distribution of the two dice. Define the outcome of the first die as X_1 and the outcome of the second die as X_2. We can then describe their joint probability table as

$p(x_1, x_2)$	$X_1 = 1$	$X_1 = 2$	$X_1 = 3$	$X_1 = 4$	$X_1 = 5$	$X_1 = 6$
$X_2 = 1$	1/36	1/36	1/36	1/36	1/36	1/36
$X_2 = 2$	1/36	1/36	1/36	1/36	1/36	1/36
$X_2 = 3$	1/36	1/36	1/36	1/36	1/36	1/36
$X_2 = 4$	1/36	1/36	1/36	1/36	1/36	1/36
$X_2 = 5$	1/36	1/36	1/36	1/36	1/36	1/36
$X_2 = 6$	1/36	1/36	1/36	1/36	1/36	1/36

i.e., each combination is equally probable. Note that this is also the table of $p(x_1)p(x_2)$ due to independence. Suppose that we can only observe the sum of the two dice, $Y = X_1 + X_2$. This would result in a likelihood

$$p(y|x_1, x_2) = \begin{cases} 1 & \text{if } y = x_1 + x_2, \\ 0 & \text{otherwise.} \end{cases}$$

We can also denote this as an indicator function, i.e., let $\mathbf{1}_{\{y=x_1+x_2\}}(x_1, x_2)$ be the indicator function of the event $y = x_1 + x_2$, then we have $p(y|x_1, x_2) = \mathbf{1}_{\{y=x_1+x_2\}}(x_1, x_2)$. Suppose now we observe $Y = 9$ and would like to infer the posterior distribution of X_1 and X_2 given $Y = 9$. We can use the Bayes rule to write

$$\begin{aligned} p(x_1, x_2|y = 9) &= \frac{p(y = 9|x_1, x_2)p(x_1, x_2)}{p(y = 9)} \\ &= \frac{p(y = 9|x_1, x_2)p(x_1)p(x_2)}{p(y = 9)}. \end{aligned}$$

Let us first write out $p(y = 9|x_1, x_2)$ as a table

$p(y = 9|x_1, x_2)$	$X_1 = 1$	$X_1 = 2$	$X_1 = 3$	$X_1 = 4$	$X_1 = 5$	$X_1 = 6$
$X_2 = 1$	0	0	0	0	0	0
$X_2 = 2$	0	0	0	0	0	0
$X_2 = 3$	0	0	0	0	0	1
$X_2 = 4$	0	0	0	0	1	0
$X_2 = 5$	0	0	0	1	0	0
$X_2 = 6$	0	0	1	0	0	0

This is just the likelihood. In order to get the full joint (numerator of the Bayes theorem), we need to multiply the likelihood with the joint prior $p(x_1, x_2) = p(x_1)p(x_2)$. Multiplying this table with the joint probability table of X_1 and X_2 gives

$p(y=9\|x_1,x_2)p(x_1)p(x_2)$	$X_1=1$	$X_1=2$	$X_1=3$	$X_1=4$	$X_1=5$	$X_1=6$
$X_2=1$	0	0	0	0	0	0
$X_2=2$	0	0	0	0	0	0
$X_2=3$	0	0	0	0	0	1/36
$X_2=4$	0	0	0	0	1/36	0
$X_2=5$	0	0	0	1/36	0	0
$X_2=6$	0	0	1/36	0	0	0

This is just the numerator in the Bayes theorem, we now need to compute the probability $p(y=9)$ in order to finally arrive at the posterior distribution. We can compute this as

$$\begin{aligned}
p(y=9) &= \sum_{x_1,x_2} p(y=9|x_1,x_2)p(x_1)p(x_2) \\
&= \sum_{x_1,x_2} \mathbf{1}(y=x_1+x_2)p(x_1)p(x_2) \\
&= \sum_{x_1,x_2} \mathbf{1}(y=x_1+x_2) \times \frac{1}{6} \times \frac{1}{6} \\
&= \frac{1}{36} \sum_{x_1,x_2} \mathbf{1}(y=x_1+x_2), \\
&= \frac{1}{36} \times 4 \\
&= \frac{1}{9}.
\end{aligned}$$

Now we are ready to normalise $p(y=9|x_1,x_2)p(x_1)p(x_2)$ to obtain the posterior distribution as a table

$p(x_1,x_2\|y=9)$	$X_1=1$	$X_1=2$	$X_1=3$	$X_1=4$	$X_1=5$	$X_1=6$
$X_2=1$	0	0	0	0	0	0
$X_2=2$	0	0	0	0	0	0
$X_2=3$	0	0	0	0	0	1/4
$X_2=4$	0	0	0	0	1/4	0
$X_2=5$	0	0	0	1/4	0	0
$X_2=6$	0	0	1/4	0	0	0

Let us next see a continuous example adapted from Murphy (2007).

Example 3.2. Let

$$\begin{aligned}
p(x) &= \mathcal{N}(x;\mu_0,\sigma_0^2), \\
p(y|x) &= \mathcal{N}(y;x,\sigma^2),
\end{aligned}$$

where μ_0 and σ_0^2 are the prior mean and variance, respectively, and σ^2 is the variance of the likelihood. Derive the posterior distribution $p(x|y)$ using the Bayes' rule.

Solution. We have seen a similar example before (without the proof), where we computed the marginal likelihood $p(y)$. In this example, we will instead derive the posterior distribution $p(x|y)$. Now let us write

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}.$$

In order to derive the posterior, we first derive $p(y|x)p(x)$ as

$$\begin{aligned}
p(y|x)p(x) &= \mathcal{N}(y; x, \sigma^2)\mathcal{N}(x; \mu_0, \sigma_0^2) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-x)^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right) \\
&= \frac{1}{2\pi\sqrt{\sigma^2\sigma_0^2}} \exp\left(-\frac{(y-x)^2}{2\sigma^2} - \frac{(x-\mu_0)^2}{2\sigma_0^2}\right).
\end{aligned}$$

We know that

$$\begin{aligned}
p(x|y) &\propto p(y|x)p(x) \\
&\propto \exp\left(-\frac{(y-x)^2}{2\sigma^2} - \frac{(x-\mu_0)^2}{2\sigma_0^2}\right).
\end{aligned}$$

Recall now that we want a density on x. Let us expand this exponential

$$\begin{aligned}
p(x|y) &\propto \exp\left(-\frac{y^2}{2\sigma^2} + \frac{xy}{\sigma^2} - \frac{x^2}{2\sigma^2} - \frac{x^2}{2\sigma_0^2} + \frac{x\mu_0}{\sigma_0^2} - \frac{\mu_0^2}{2\sigma_0^2}\right) \\
&= \exp\left(-\left(\frac{1}{2\sigma^2} + \frac{1}{2\sigma_0^2}\right)x^2 + \left(\frac{y}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\right)x - \frac{y^2}{2\sigma^2} - \frac{\mu_0^2}{2\sigma_0^2}\right), \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{\sigma^2+\sigma_0^2}{\sigma^2\sigma_0^2}\right)x^2 + \left(\frac{y\sigma_0^2 + \mu_0\sigma^2}{\sigma_0^2\sigma^2}\right)x\right), \\
&= \exp\left(-\frac{1}{2}\left(\frac{\sigma^2+\sigma_0^2}{\sigma^2\sigma_0^2}\right)\left(x^2 - 2x\left(\frac{y\sigma_0^2 + \mu_0\sigma^2}{\sigma_0^2 + \sigma^2}\right)\right)\right), \\
&\propto \exp\left(-\frac{1}{2}\frac{\left(x - \frac{y\sigma_0^2+\mu_0\sigma^2}{\sigma_0^2+\sigma^2}\right)^2}{\frac{\sigma^2\sigma_0^2}{\sigma^2+\sigma_0^2}}\right).
\end{aligned}$$

This can be recognised as a Gaussian:

$$\begin{aligned}
\mu_p &= \frac{\sigma^2\mu_0 + \sigma_0^2 y}{\sigma^2 + \sigma_0^2}, \\
\sigma_p^2 &= \frac{\sigma^2\sigma_0^2}{\sigma^2 + \sigma_0^2}.
\end{aligned}$$

This gives us our Gaussian posterior. See Fig 3.1 for an illustration.

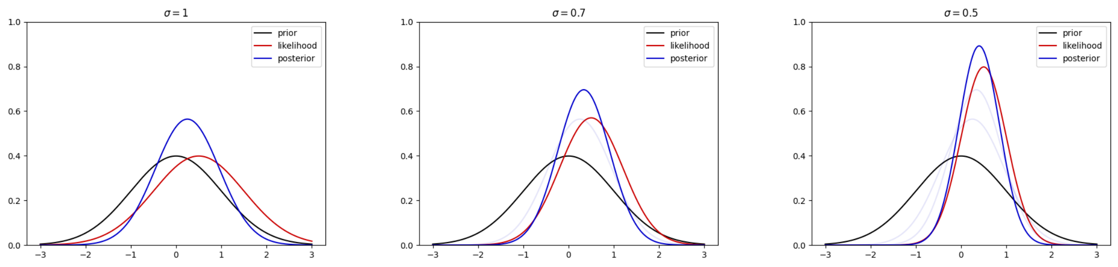


Figure 3.1: Posterior distribution of x given $\sigma = 1, \sigma = 0.7$ and $\sigma = 0.5$ respectively. One can see that as we shrink the likelihood variance, the posterior distribution becomes more peaked towards the observation $y = 0.5$. Old posteriors are also plotted in the second and third figure for comparison (in transparent blue).

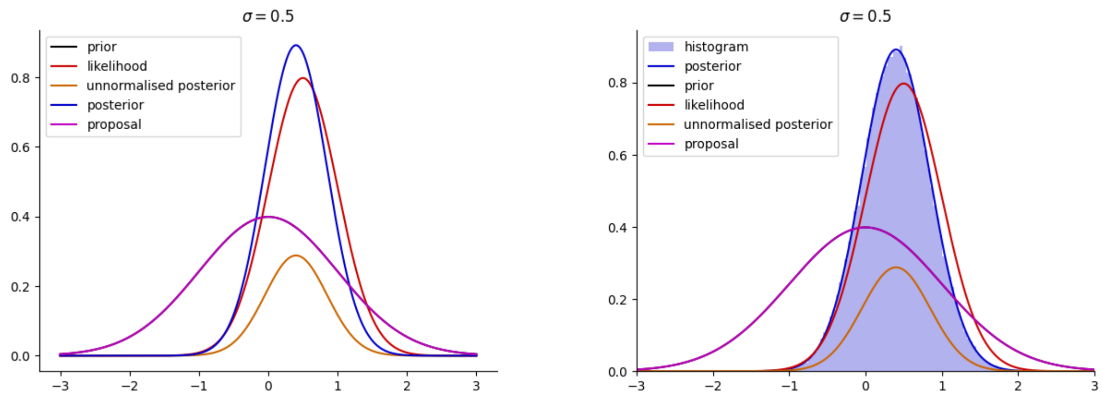


Figure 3.2: On the left, we plot all distributions of interest: prior, likelihood (with $y = 0.5$ with respect to x), the posterior, and the unnormalised posterior, and the proposal. Note that, the proposal should only cover the unnormalised posterior, even if the normalising constant is less than one. On the left, we plot the samples vs. the same quantities. One can see that we exactly sampled from the correct posterior.

This is an example of a *conjugate prior*, where the posterior distribution is of the same form as the prior. In the solved examples section, we will see more examples of this. As you have seen, the derivation of the posterior took some work. As opposed to this conjugate case, in the general case, we will not be able to derive the posterior. Let us see one example now how we can avoid computing the normalised posterior but still sample from it.

Example 3.3. Assume that we have a prior distribution $p(x) = \mathcal{N}(x; \mu_0, \sigma_0^2)$ and a likelihood $p(y|x) = \mathcal{N}(y; x, \sigma^2)$. We want to sample the posterior distribution $p(x|y)$ without going through the derivation. Derive the rejection sampler for this purpose.

Solution. We know the posterior is given by

$$p(x|y) \propto p(y|x)p(x) = \mathcal{N}(x; \mu_0, \sigma_0^2)\mathcal{N}(y; x, \sigma^2).$$

Recall that we would like to sample from the posterior $p(x|y)$ without necessarily computing the Bayes rule. We can pose this problem as a *rejection sampling* problem. We would like to sample from the posterior distribution conditioned on y. In our case, the unnormalised posterior is

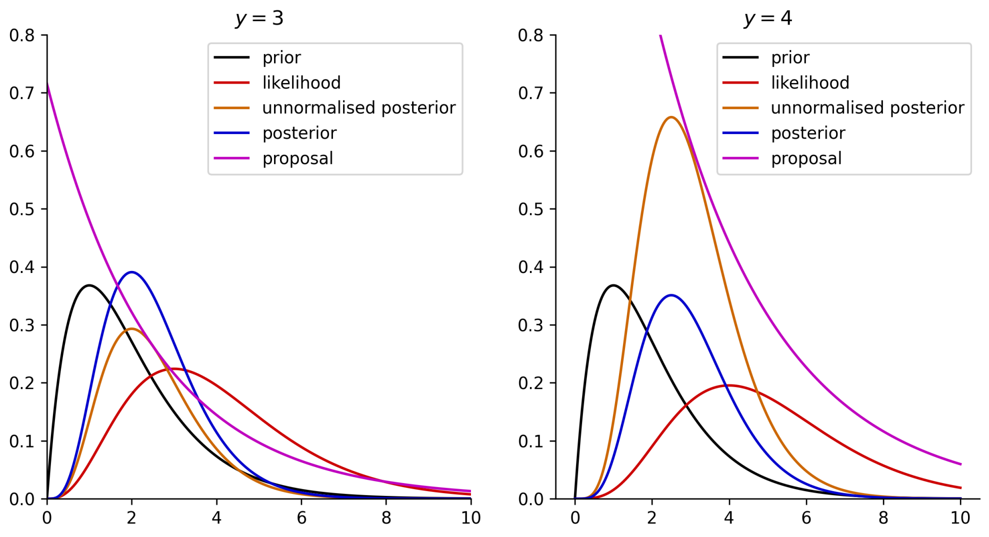


Figure 3.3: Illustration of the prior, posterior, likelihood, and the proposal distribution.

given by

$$\bar{p}(x|y) = p(y|x)p(x)$$

Note that we *evaluate* the likelihood at the observation y and hence it becomes a function of x. Below, for clarity, we will use the r.h.s. of above equation in acceptance rate, instead of $\bar{p}(x)$ as we usually did before. For this example, we also set $\mu_0 = 0$, $\sigma_0 = 1$, and $\sigma = 0.5$. Next, we need to design a proposal distribution $q(x)$. This could be tricky as we do not know the posterior. For now, we can choose another simple Gaussian (we could also optimise this):

$$q(x) = \mathcal{N}(x; \mu_q, \sigma_q^2).$$

Let us choose $\mu_q = 0$ and $\sigma_q = 1$ (note again that this is the standard deviation!) and $M = 1$. An illustration of this is shown in Fig 3.2. We can now sample from the posterior

- Sample $X' \sim q(x)$
- Sample $U \sim \text{Unif}(0, 1)$
- If $U \leq \frac{p(y|X')p(X')}{Mq(X')}$, accept X'. Otherwise, reject X' and go back to step 1.

We can see the results of this procedure from Fig 3.2. As seen from the figure, we exactly sample from the posterior $p(x|y = 0.5)$ without ever computing the correct posterior. We have also plotted the correct posterior in the figure for comparison.

Let us see another example.

Example 3.4. Assume that we have a Poisson observation model:

$$p(y|x) = \text{Pois}(y; x) = \frac{x^y e^{-x}}{y!},$$

and a Gamma prior:

$$p(x) = \text{Gamma}(x; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}.$$

We want to sample from the posterior distribution $p(x|y)$. Derive the posterior distribution $p(x|y)$.

Solution. We know that the posterior is proportional to

$$\begin{aligned} p(x|y) &\propto p(y|x)p(x) \\ &= \text{Pois}(y; x)\text{Gamma}(x; \alpha, \beta), \\ &\propto x^{\alpha-1+y} e^{-\beta x - x}, \end{aligned}$$

where we ignored all the normalising constants. We can see that the posterior is also a Gamma density:

$$p(x|y) = \text{Gamma}(x; \alpha + y, \beta + 1).$$

Let us sample from this posterior with rejection sampling as we did before for the Gaussian.

Example 3.5. Assume that we have a Gamma prior:

$$p(x) = \text{Gamma}(x; \alpha, 1) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x},$$

with $\alpha > 0$. Next, we define our Poisson observation model as before

$$p(y|x) = \text{Pois}(y; x) = \frac{x^y e^{-x}}{y!}.$$

Derive the rejection sampler for the posterior without explicitly computing the posterior distribution.

Solution. We know that the posterior is proportional to

$$\begin{aligned} p(x|y) &\propto p(y|x)p(x) = \text{Pois}(y; x)\text{Gamma}(x; \alpha, 1), \\ &\propto x^{\alpha-1+y} e^{-2x}. \end{aligned}$$

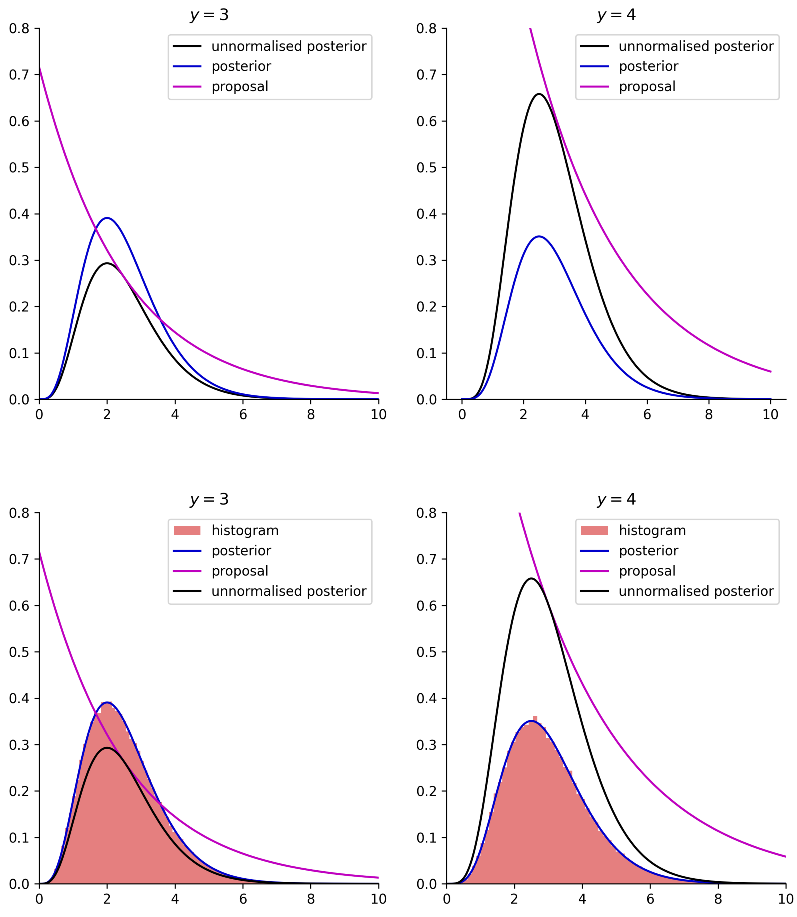


Figure 3.4: Histogram of the samples drawn using rejection sampling.

In short, we will choose this as our unnormalised posterior

$$\bar{p}(x|y) = x^{\alpha-1+y} e^{-2x}.$$

Now we will design our proposal distribution. We choose the proposal as an exponential distribution:

$$q_\lambda(x) = \text{Exp}(x;\lambda) = \lambda e^{-\lambda x}.$$

Now we derive the acceptance probability. As usual, we need to first find

$$M_\lambda = \sup_x \frac{\bar{p}(x|y)}{q_\lambda(x)}.$$

First we need to optimise the ratio:

$$\begin{aligned}\frac{\bar{p}(x|y)}{q_\lambda(x)} &= \frac{x^{\alpha-1+y} e^{-2x}}{\lambda e^{-\lambda x}} \\ &= \frac{x^{\alpha-1+y} e^{-(2-\lambda)x}}{\lambda}.\end{aligned}$$

Aiming at optimising this w.r.t. x, we first compute its log:

$$\begin{aligned}\log \frac{\bar{p}(x|y)}{q_\lambda(x)} &= \log x^{\alpha-1+y} + \log e^{-(2-\lambda)x} - \log \lambda \\ &= (\alpha - 1 + y)\log x - (2-\lambda)x - \log\lambda.\end{aligned}$$

We now take the derivative of this w.r.t. x:

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[(\alpha-1+y)\log x - (2-\lambda)x - \log\lambda\right] = \frac{\alpha-1+y}{x} - (2-\lambda),$$

and set it to zero:

$$\frac{\alpha-1+y}{x} - (2-\lambda) = 0.$$

This gives us the maximiser

$$x^\star = \frac{\alpha-1+y}{2-\lambda}.$$

We can now compute M_λ:

$$\begin{aligned}M_\lambda &= \frac{\bar{p}(x^\star|y)}{q_\lambda(x^\star)} \\ &= \frac{x^{\star\alpha-1+y}e^{-(2-\lambda)x^\star}}{\lambda} \\ &= \frac{1}{\lambda}\left(\frac{\alpha-1+y}{2-\lambda}\right)^{\alpha-1+y} e^{-(2-\lambda)\left(\frac{\alpha-1+y}{2-\lambda}\right)} \\ &= \frac{1}{\lambda}\left(\frac{\alpha-1+y}{2-\lambda}\right)^{\alpha-1+y} e^{-(\alpha-1+y)}.\end{aligned}$$

We can now optimise this further to choose our optimal proposal. We will first compute the log of M_λ:

$$\begin{aligned}\log M_\lambda &= \log\frac{1}{\lambda} + (\alpha-1+y)\log\left(\frac{\alpha-1+y}{2-\lambda}\right) - (\alpha-1+y) \\ &= -\log\lambda + (\alpha-1+y)\log\left(\frac{\alpha-1+y}{2-\lambda}\right) - (\alpha-1+y).\end{aligned}$$

Taking the derivative of this w.r.t. λ, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\log M_\lambda = -\frac{1}{\lambda} + \frac{(\alpha-1+y)}{2-\lambda}$$

Setting this to zero, we obtain

$$\frac{1}{\lambda} = \frac{(\alpha-1+y)}{2-\lambda},$$

which implies that

$$\lambda^\star = \frac{2}{\alpha+y}.$$

Therefore, we can choose our optimal proposal in terms of α and y depends on the observed sample. See Fig . 3.4 for the histogram of the samples drawn using rejection sampling.

3.3 CONDITIONAL INDEPENDENCE

The step forward from the simple Bayes rule to modelling complex dependencies and interactions is to understand the notion of conditional independence. Simply put, conditional independence is a notion of independence of two random variables *conditioned* on a third random variable. Of course, this can be extended to arbitrary number of variables, defining a full probabilistic model. It is important to note that these models *everywhere* in science and engineering.

Let us first define the notion of conditional independence.

Definition 3.2. *Let X, Y and Z be random variables. We say that X and Y are conditionally independent given Z if*

$$p(x, y|z) = p(x|z)p(y|z).$$

This definition is of course the same as plain independence, just written in terms of conditional probabilities. Note that, in general, X and Y are not independent if we do not condition on Z. We note the important corollary.

Corollary 3.1. *If X and Y are conditionally independent given Z, then*

$$p(x|y, z) = p(x|z),$$

and

$$p(y|x, z) = p(y|z).$$

Proof. See Exercise 4.2 solution. □

We can now describe the notion of conditional independence in terms of joint distributions.

Proposition 3.1. *Let X, Y and Z be random variables. If X and Y are conditionally independent given Z, then*

$$p(x, y, z) = p(x|z)p(y|z)p(z).$$

Proof. Recall that we have described the chain rule for conditional probabilities in Sec. 2.4.3

$$p(x_1, \ldots, x_n) = p(x_n|x_{n-1}, \ldots, x_1)p(x_{n-1}|x_{n-2}, \ldots, x_1) \cdots p(x_2|x_1)p(x_1).$$

This relationship is as important as in inference as in simulation. We can now use this to show that

$$\begin{aligned} p(x, y, z) &= p(x|y, z)p(y|z)p(z) \\ &= p(x|z)p(y|z)p(z), \end{aligned}$$

where the last line follows from Corollary 3.1. □

This idea can be extended to arbitrary number of variables. This kind of factorisations are at the core of probabilistic modelling. In other words, a probabilistic modeller (scientist) poses a set of conditional independence assumptions which then allows them to factorise the joint distribution into a product of conditional distributions. From then on, the modeller can use the conditional distributions to compute any desired marginal or conditional distributions. This is the essence of probabilistic modelling.

3.3.1 BAYES RULE FOR CONDITIONALLY INDEPENDENT OBSERVATIONS

So far, we have seen an example of prior to posterior update for a single observation in Sec. 3.2 and the definition of conditional independence. We can now combine these two ideas to obtain the Bayes update for conditionally independent observations. This is a standard use case for conditional independence: Typically, given an unobserved variable x, we can obtain multiple measurements related to a single latent variable x.

Let us define the general Bayes update for this case. Assume that we have observed $y_1, \ldots, y_n \sim p(y|x)$ (these can be thought of as conditionally i.i.d samples from the likelihood)[2]. Given a prior of x, denoted $p(x)$, we want to compute the posterior $p(x|y_1, \ldots, y_n)$. We know that the posterior is given by

$$p(x|y_{1:n}) = \frac{p(y_{1:n}|x)p(x)}{p(y_{1:n})}. \tag{3.4}$$

Under the conditional independence assumption of observations, we can just use Definition 3.2 to arrive at

$$p(y_{1:n}|x) = \prod_{i=1}^{n} p(y_i|x).$$

Plugging this in back to the Bayes update (3.4), we can see that the posterior is proportional to the product

$$\begin{aligned} p(x|y_1, \ldots, y_n) &\propto p(y_1, \ldots, y_n|x)p(x) \\ &= \prod_{i=1}^{n} p(y_i|x)p(x), \end{aligned}$$

Again, in many occasions, we will not be able to compute the normalising constant. However, we can still sample from the posterior. In this particular example, let us continue with the Gaussian prior and likelihood. In this case, we can exactly compute the posterior too.

Example 3.6 (Gaussian Bayes update for conditionally independent observations)**.** Let us assume the following probabilistic model

$$\begin{aligned} X &\sim \mathcal{N}(x; \mu_0, \sigma_0^2) = p(x) \\ Y_i|X = x &\sim \mathcal{N}(y_i; x, \sigma^2) = p(y_i|x), \quad i = 1, \ldots, n, \end{aligned}$$

[2]We define the following notation. Let $y_1, \ldots, y_n$ be n observations. We collectively denote these variables as $y_{1:n} := (y_1, \ldots, y_n)$. This will be also used in the following sections.

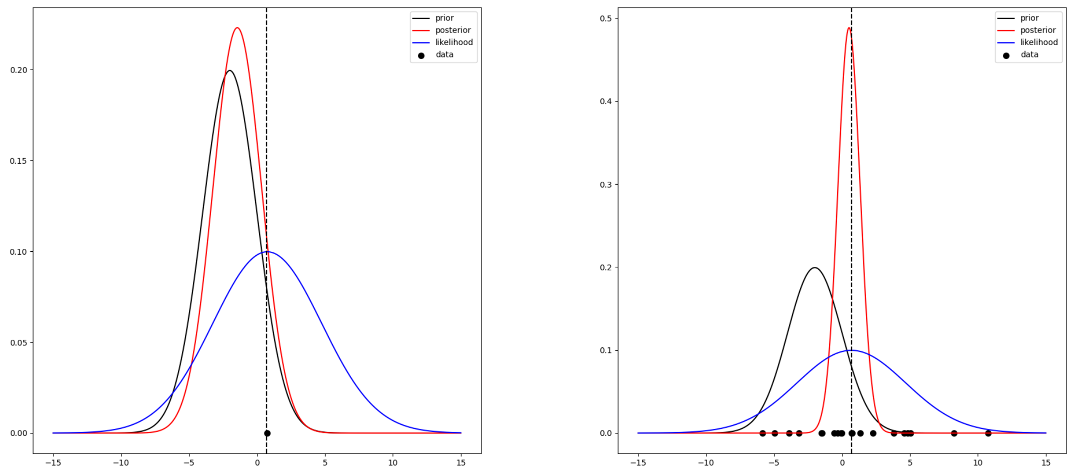


Figure 3.5: Bayes update for conditionally independent observations.

where Y_i are conditionally independent given $X = x$. Derive the posterior distribution $p(x|y_1, \ldots, y_n)$.

Solution. Here each observation is assumed to be conditionally independent given x. Note that this model is very different than the one where we simulated (X_i, Y_i) pairs in Example 2.14. The point in Example 2.14 was to simulate pairs exhibiting linear relationship, each (Y_i, X_i) was an independent draw from the joint distribution. Here, we assume that the observations are sampled conditioned on a *single* x – in essence, the sequence $y_1, \ldots, y_n$ are dependent. They are only conditionally independent given x.

Having observed $y_1, \ldots, y_n$, we would like to compute the posterior $p(x|y_1, \ldots, y_n)$. Let us first compute the likelihood

$$\begin{aligned} p(y_{1:n}|x) &= \prod_{i=1}^{n} p(y_i|x) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x)^2}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - x)^2\right). \end{aligned}$$

Using the same derivations as in Example 3.2, we can compute the posterior

$$\begin{aligned} p(x|y_{1:n}) &= \frac{p(y_{1:n}|x)p(x)}{p(y_{1:n})} \\ &= \frac{p(y_{1:n}|x)p(x)}{\int p(y_{1:n}|x)p(x)dx} \end{aligned}$$

where $p(x|y_{1:n}) = \mathcal{N}(x; \mu_p, \sigma_p^2)$, with (Murphy, 2007)

$$\mu_p = \frac{\sigma_0^2 \sum_{i=1}^n y_i + \sigma^2 \mu_0}{\sigma_0^2 n + \sigma^2}$$
$$\sigma_p^2 = \frac{\sigma_0^2 \sigma^2}{\sigma_0^2 n + \sigma^2}.$$

The posterior with conditioned data can be seen from Fig. 3.5.

3.3.2 CONDITIONAL BAYES RULE

It is important to realise that the Bayes rule can be used *conditionally*. Consider three variables X, Y, Z without specifying any conditional independence assumptions. In this case, the Bayes rule for $p(x|y, z)$ can be written entirely on z (of course, this is true if we swap the variables and condition on x or y). We can write in this case the conditional Bayes rule.

Proposition 3.2. *Given X, Y, Z without any conditional independence assumptions, the conditional Bayes rule is*

$$p(x|y, z) = \frac{p(y|x, z)p(x|z)}{p(y|z)}.$$

Proof. See the solution of Exercise 4.1. □

This is of course true if we write the same rule for x or y conditioned.

3.4 MARGINAL LIKELIHOOD

The notion of marginal likelihood is left unexplored so far and we will now investigate it. We can go back to the Bayes theorem and write

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}.$$

In this formula, we have been discussing the posterior $p(x|y)$, the prior $p(x)$, and the likelihood $p(y|x)$ in past sections. However, the normalising constant, which we assumed to be intractable, is also of interest. This quantity, $p(y)$, is called the marginal likelihood and it is given by

$$p(y) = \int p(y|x)p(x)\mathrm{d}x.$$

For fixed y, the interpretation of this term is that it gives us the *probability of data y* under the model[3]. For more complicated models (where x can be multiple variables or multiple other

[3] Aside from its usual interpretation as the normalising constant.

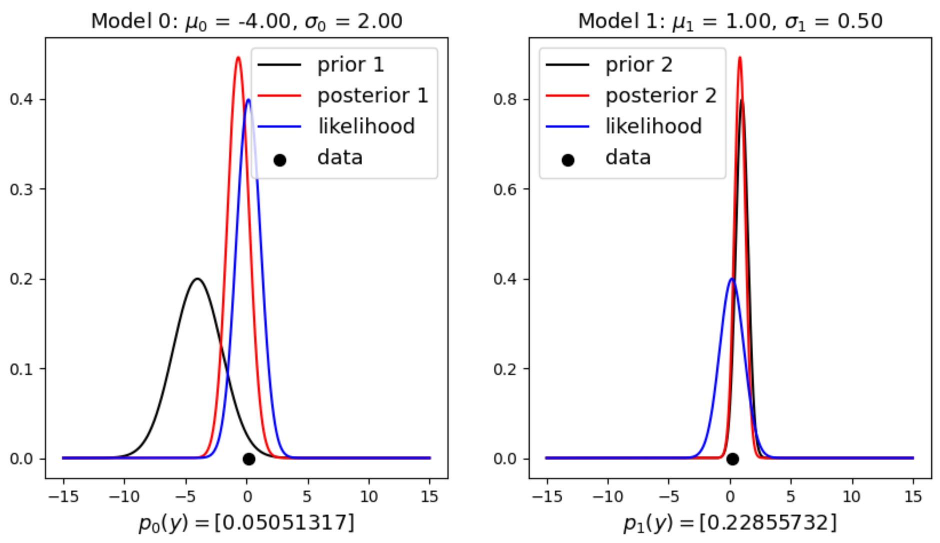


Figure 3.6: Marginal likelihood for model comparison. For observed data, we can compute the marginal likelihood for each model. The model with the highest marginal likelihood is the best model for the observed data.

distributions may exist), the quantity $p(y)$ becomes crucial to determine the quality of the model for the observed data. While itself does not mean much, it gives us a comparative measure to compare different models. We will discuss this with an example.

Example 3.7 (Marginal likelihood for two Gaussian models)**.** Consider two different models:

$$\begin{aligned} p_0(x) &= \mathcal{N}(x; \mu_0, \sigma_0^2) \\ p(y|x) &= \mathcal{N}(y; x, \sigma_y^2) \end{aligned}$$

and

$$\begin{aligned} p_1(x) &= \mathcal{N}(x; \mu_1, \sigma_1^2) \\ p(y|x) &= \mathcal{N}(y; x, \sigma_y^2). \end{aligned}$$

Consider observing y (a single data point). How can you find out which model is more likely?

Solution. Recall that, for these models, we have computed $p(y)$ analytically before. We can compute for both models:

$$\begin{aligned} p_0(y) &= \int p(y|x)p_0(x)\mathrm{d}x \\ &= \int \mathcal{N}(y; x, \sigma_y^2)\mathcal{N}(x; \mu_0, \sigma_0^2)\mathrm{d}x \\ &= \mathcal{N}(y; \mu_0, \sigma_0^2 + \sigma_y^2) \end{aligned}$$

and

$$\begin{aligned}
p_1(y) &= \int p(y|x)p_1(x)\mathrm{d}x \\
&= \int \mathcal{N}(y; x, \sigma_y^2)\mathcal{N}(x; \mu_1, \sigma_1^2)\mathrm{d}x \\
&= \mathcal{N}(y; \mu_1, \sigma_1^2 + \sigma_y^2)
\end{aligned}$$

We will say Model 1 is better than Model 0 if $p_1(y) > p_0(y)$ for fixed y and similarly, we will say Model 0 is better if $p_1(y) < p_0(y)$. This, however, as you can see depends on various parameters. Let us choose that $\sigma = 1$, $\mu_0 = -4$, $\sigma_0 = 2$, and $\mu_1 = 1$, $\sigma_1 = 0.5$. The computed marginal likelihoods can be seen from Fig. 3.6. It can be seen that Model 1 is a much better fit to the data than Model 0.

3.5 SEQUENTIAL BAYESIAN UPDATING

Before concluding this section, we will pay special attention to a case where the conditional Bayes rule is used together with many conditionally i.i.d. observations. Consider a scenario where we would like to obtain the distribution of a random variable X given a sequence of datapoints $y_{1:n}$, i.e., $p(x|y_{1:n})$. We can see that, to repeat, this can be done using Eq. (3.4), i.e.,

$$p(x|y_{1:n}) = \frac{p(y_{1:n}|x)p(x)}{p(y_{1:n})}.$$

However, in many cases, we are interested in $(y_n)_{n\geq 1}$ arriving *sequentially*, that is, one at a time. In this case, we can use the conditional Bayes rule to update the posterior sequentially.

Assume that we start from the prior $p(x)$ before we *observe* any data. If we then observe y_1, one can construct the posterior of x as

$$p(x|y_1) = \frac{p(y_1|x)p(x)}{p(y_1)},$$

as we have seen before. Suppose now that we observe a new data point y_2 and would like to obtain $p(x|y_{1:2})$ without reprocessing y_1. We can do this using the Bayes rule as

$$\begin{aligned}
p(x|y_{1:2}) &= \frac{p(y_{1:2}|x)p(x)}{p(y_{1:2})} \\
&= \frac{p(y_2|x, y_1)p(y_1|x)p(x)}{p(y_2|y_1)p(y_1)} \\
&= \frac{p(y_2|x, \cancel{y_1})p(y_1|x)p(x)}{p(y_2|y_1)p(y_1)} \\
&= \frac{p(y_2|x)p(x|y_1)}{p(y_2|y_1)}.
\end{aligned}$$

One can see that this is equivalent to using $p(x|y_1)$ as a prior. Also this is the conditional Bayes update conditioned on y_1.

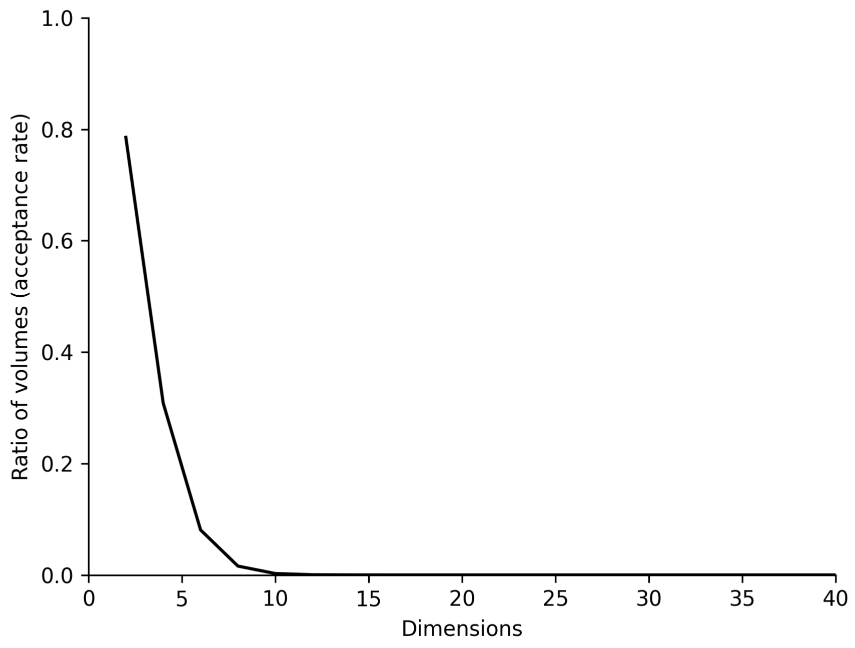


Figure 3.7: The curse of dimensionality for the sampling example for rejection sampling.

The generalisation of this process goes as follows. Assume that we have $p(x|y_{1:n-1})$, i.e., the conditional posterior distribution of $n-1$ observations. Upon receiving y_n, we can update our posterior as

$$\begin{aligned} p(x|y_{1:n}) &= p(x|y_n, y_{1:n-1}), \\ &= \frac{p(y_n|x, \cancel{y_{1:n-1}})p(x|y_{1:n-1})}{p(y_n|y_{1:n-1})}, \\ &= \frac{p(y_n|x)p(x|y_{1:n-1})}{p(y_n|y_{1:n-1})}. \end{aligned}$$

where we obtain the *sequential Bayesian updating* rule. This is a very important result as it allows us to update our posterior sequentially without reprocessing the data. This is especially useful in online learning scenarios where we would like to update our posterior as we receive new data points. This will be of crucial use when we consider sequential Monte Carlo towards the end of the course. We will have one exercise about a Gaussian example in this setting.

3.6 CONCLUSION

In this section, we briefly discussed the Bayes rule and its application to probabilistic inference. This is a vast topic and we have only scratched the surface. If you are curious about the topic, Bishop (2006) is a good book to read. Some other very nices ones are Barber (2012) and Murphy (2022). We will finish this chapter by discussing why rejection samplers as we introduced it would not be an appropriate candidate for sampling in more complicated models we discussed in this chapter.

Given all these derivations, it is natural to ask whether we can use rejection samplers for Bayesian inference. Let us assume that we have $y_1, \ldots, y_n$ observed and our unnormalised posterior is given by

$$\bar{p}(x|y_{1:n}) = p(x) \prod_{i=1}^{n} p(y_i|x).$$

Let us assume that we have a proposal distribution $q(x)$ and assume that we have been lucky to identify some M such that

$$\bar{p}(x|y_{1:n}) \leq Mq(x).$$

We can now perform rejection sampling as follows:

1. Sample $X' \sim q(x)$
2. Sample $U \sim$ Unif$(0, 1)$
3. If $U \leq \frac{\bar{p}(X'|y_{1:n})}{Mq(X')} = \frac{p(X') \prod_{i=1}^{n} p(y_i|X')}{Mq(X')}$ then accept X'
4. Otherwise reject X' and go back to step 1.

What could be an immediate problem as n grows? The multiplication $\prod_{i=1}^{n} p(y_i|X')$ would not be numerically stable. This would result in numerical underflow as the multiplication of small probabilities gets smaller and smaller. In order to mitigate this, one solution is to work with log-probabilities. This means that we can still perform rejection sampling (provided that $\bar{p}(x|y) \leq Mq(x)$) as follows:

1. Sample $X' \sim q(x)$
2. Sample $U \sim$ Unif$(0, 1)$
3. Compute log-acceptance probability

$$\begin{aligned}\log a(X') = \log \frac{\bar{p}(X'|y_{1:n})}{Mq(X')} &= \log \frac{p(X') \prod_{i=1}^{n} p(y_i|X')}{Mq(X')}, \\ &= \log p(X') + \sum_{i=1}^{n} \log p(y_i|X') - \log M - \log q(X').\end{aligned}$$

4. If $\log U \leq \log a(X')$ then accept X'

However, this would also not often solve our issues as

- It is often impossible to find M and $q(x)$ such that $\bar{p}(x|y) \leq Mq(x)$.
- Bounds found to log-unnormalised posterior can be very loose
  - Super low acceptance probability

This is also not the only failure mode of the rejection sampling. It is often the case that rejection sampling is very inefficient in high dimensions even if one manages to find a good proposal q. Consider the rejection sampling in 2D for sampling the circle within a square. The acceptance probability for this case:

$$\hat{a} = \frac{\text{area of the circle}}{\text{area of the square}} = \frac{\pi}{4} \approx 0.78.$$

Next, consider the same sampler for the sphere and the cube (3D). The acceptance probability for this case:

$$\hat{a} = \frac{\text{volume of the sphere}}{\text{volume of the cube}} = \frac{\pi}{6} \approx 0.52.$$

If we were doing this in d dimensions, the acceptance rate would be

$$\hat{a} = \frac{\text{volume of the unit ball}}{\text{volume of the unit cube}} = \frac{\pi^{d/2}}{\Gamma(d/2+1)}.$$

However, this ratio goes to zero incredibly fast as d grows (see Fig. 3.7) In other words, rejection samplers have very poor acceptance rates in high dimensions. This will lead us to look at other sampling methods.

4 MONTE CARLO INTEGRATION

In this section, we introduce Monte Carlo integration and importance sampling in detail. We will show how these ideas can be applied to a variety of problems such as computing integrals, computing expectations, sampling from complex distributions, and computing marginal likelihoods.

4.1 INTRODUCTION TO MONTE CARLO INTEGRATION

We have repeatedly highlighted that we are interested in sampling from a probability measure p. One reason we are interested in this is to *estimate expectations* of certain measures, i.e., we can estimate moments of distributions. Of course, so far, we have been considering drawing samples from known distributions (for which moments might be readily available). However, it is often the case in sampling applications that, in most cases, the primary goal *is* to compute expectations for distributions which are not available to us in closed form.

We will call this task as *Monte Carlo integration*. Briefly, given a probability distribution p, we are interested in computing expectations of the form

$$\bar{\varphi} = \mathbb{E}_p[\varphi(X)] = \int \varphi(x)p(x)\mathrm{d}x = (\varphi, p),$$

where $\varphi(x)$ is called a *test function*. For example, $\varphi(x) = x$ would give us the mean, $\varphi(x) = x^2$ the second moment, or $\varphi(x) = \log(x)$ would give us the entropy. For example, given $X^{(1)}, \ldots, X^{(N)} \sim p$ i.i.d, we know that (intuitively, at this point) the mean estimator is given by

$$\mathbb{E}_p[X] = \int xp(x)\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} X^{(i)},$$

which is simply the empirical average of the samples. While this can be intuitive, it underlies a certain choice about the approximation of the probability measure $p(\mathrm{d}x) := p(x)\mathrm{d}x$ using its samples. In order to do this, we build an *empirical distribution* of the samples, using

$$p^N(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^N \delta_{X^{(i)}}(\mathrm{d}x). \tag{4.1}$$

In order to understand how this works, we first need to understand the Dirac delta measure δ_x. The Dirac delta measure is defined as

$$f(y) = \int f(x)\delta_y(\mathrm{d}x) \tag{4.2}$$

Here, the Dirac can be thought as a *point mass* at y. In other words, the Dirac delta measure is a measure which is concentrated at a single point. To understand it intuitively, the object $\delta_y(x)$ can be informally thought as a function centered at y (and only takes value 1 at y) [1]

$$\delta_y(x) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases}$$

One can see that then p^N is a *sample based approximation* of p, where the samples are equally weighted. While we never may use this particular approximation of as a density approximation, it is useful to build estimates of expectations. Generalising the above scenario, let us consider the estimation of the general expectation

$$\bar{\varphi} = (\varphi, p) = \mathbb{E}_p[\varphi(X)] = \int \varphi(x)p(x)\mathrm{d}x.$$

Given samples $X^{(1)}, \ldots, X^{(N)}$, we can build p^N as in (4.1) and approximate this expectation as

$$\begin{aligned}
\bar{\varphi} &= \mathbb{E}_p[\varphi(x)] = (\varphi, p) \\
&= \int \varphi(x)p(x)\mathrm{d}x \\
&\approx \int \varphi(x)p^N(\mathrm{d}x) = (\varphi, p^N) \\
&= \int \varphi(x)\frac{1}{N}\sum_{i=1}^N \delta_{X_i}(\mathrm{d}x) \\
&= \frac{1}{N}\sum_{i=1}^N \int \varphi(x)\delta_{X_i}(\mathrm{d}x) \\
&= \frac{1}{N}\sum_{i=1}^N \varphi(X_i) = \hat{\varphi}^N_{\mathrm{MC}},
\end{aligned} \tag{4.3}$$

where we have used (4.2) in the approximate integral to arrive at the final expression. Note that this generalises the example above about the mean (which was $\varphi(x) = x$ case). In this course, we will also be interested in the properties of these estimators.

[1] This is not correct rigorously – just for intuition! Note that the Diracs always make sense with an integral attached to them.

Remark 4.1. As we can see that, the Monte Carlo estimator can be used to estimate expectations. We can also use this idea to estimate integrals. Consider a standard integration problem

$$I = \int f(x) \mathrm{d}x,$$

where $f(x)$ is a function. We can use the Monte Carlo (MC) estimator to estimate this integral as

$$\begin{aligned} I &= \int \frac{f(x)}{p(x)} p(x) \mathrm{d}x \\ &\approx \int \frac{f(x)}{p(x)} p^N(\mathrm{d}x) && \text{where } p^N(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^N \delta_{X^{(i)}}(\mathrm{d}x) \\ &= \frac{1}{N}\sum_{i=1}^N \frac{f(X_i)}{p(X_i)}. && \text{using (4.1)} \end{aligned}$$

In this case, we have $\varphi(x) = \frac{f(x)}{p(x)}$. This is particularly easy for the integrals of type

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

where $f(x)$ is a function. In this case, we can use the uniform distribution as the base distribution p and use the Monte Carlo estimator to estimate the integral without needing to compute any ratios.

In the following, we prove some results about the properties of the Monte Carlo estimator (4.3) when samples are i.i.d from p.

Proposition 4.1. *Let $X_1, \ldots, X_N$ be i.i.d samples from p. Then, the Monte Carlo estimator*

$$\hat{\varphi}^N_{MC} = \frac{1}{N}\sum_{i=1}^N \varphi(X_i)$$

is unbiased, i.e.,

$$\mathbb{E}_p[\hat{\varphi}^N_{MC}] = \bar{\varphi}.$$

Proof. We have

$$\begin{aligned}\mathbb{E}_p[\hat{\varphi}^N_{\text{MC}}] &= \mathbb{E}_p\left[\frac{1}{N}\sum_{i=1}^N \varphi(X_i)\right] \\ &= \frac{1}{N}\sum_{i=1}^N \mathbb{E}_p[\varphi(X_i)] \\ &= \frac{1}{N}\sum_{i=1}^N \int \varphi(x)p(x)\mathrm{d}x \\ &= \int \varphi(x)p(x)\mathrm{d}x \\ &= \bar{\varphi},\end{aligned}$$

which proves the result. □

Next, we can also compute the variance of the Monte Carlo estimator.

Proposition 4.2. *Let $X_1, \ldots, X_N$ be i.i.d samples from p. Then, the Monte Carlo estimator*

$$\hat{\varphi}^N_{MC} = \frac{1}{N}\sum_{i=1}^N \varphi(X_i)$$

has variance

$$\mathsf{var}_p[\hat{\varphi}^N_{MC}] = \frac{1}{N}\mathsf{var}_p[\varphi(X)].$$

where

$$\mathsf{var}_p[\varphi(X)] = \int (\varphi(x) - \bar{\varphi})^2 p(x)\mathrm{d}x.$$

Proof. We have

$$\begin{aligned}\mathsf{var}_p[\hat{\varphi}^N_{\text{MC}}] &= \mathsf{var}_p\left[\frac{1}{N}\sum_{i=1}^N \varphi(X_i)\right] \\ &= \frac{1}{N^2}\sum_{i=1}^N \mathsf{var}_p[\varphi(X_i)] \\ &= \frac{1}{N^2}\sum_{i=1}^N \int (\varphi(x) - \bar{\varphi})^2 p(x)\mathrm{d}x \\ &= \frac{1}{N}\mathsf{var}_p[\varphi(X)] = \frac{\sigma^2_\varphi}{N}\end{aligned}$$

Provided that $\mathsf{var}_p[\varphi(X)] < \infty$ and the estimator is consistent as $N \to \infty$. This proves the result.
□

Remark 4.2. The expression $\mathsf{var}_p[\hat{\varphi}^N_{\mathrm{MC}}]$ is the variance of the MC estimator but this expression requires the true mean $\bar{\varphi}$ to be known. In practice, we do not know the true mean but also have an MC estimator for it. We can plug this estimator into the variance in order to obtain an empirical variance estimator. Note that

$$\begin{aligned}
\mathsf{var}_p[\hat{\varphi}^N_{\mathrm{MC}}] &= \frac{1}{N}\mathsf{var}_p[\varphi(X)] \\
&= \frac{1}{N}\int (\varphi(x) - \bar{\varphi})^2 p(x)\mathrm{d}x \\
&\approx \frac{1}{N^2}\sum_{i=1}^N (\varphi(X_i) - \hat{\varphi}^N_{\mathrm{MC}})^2 \\
&= \sigma^2_{\varphi,N}.
\end{aligned}$$

This estimator then can be used to estimate the variance of the MC estimator.

We can therefore obtain a central limit theorem for our estimator, i.e.,

$$\frac{(\hat{\varphi}^N_{\mathrm{MC}} - \bar{\varphi})}{\sigma_{\varphi,N}} \to \mathcal{N}(0,1) \qquad \text{as} \qquad N \to \infty.$$

This can be used to build empirical confidence intervals for the estimators. However, this is not a principled estimate and may not be valid in many scenarios. We can also see that we have a standard deviation estimate (which follows from the variance estimate) given by

$$\mathsf{std}_p[\hat{\varphi}^N_{\mathrm{MC}}] = \sqrt{\mathsf{var}_p[\hat{\varphi}^N_{\mathrm{MC}}]} = \frac{\sigma_\varphi}{\sqrt{N}}.$$

This is a typical display of a convergence rate $\mathcal{O}(1/\sqrt{N})$.

Remark 4.3. One of the most common application of sampling is to estimate probabilities. We have seen that different choices of φ can lead to estimating different quantities such as the mean and nth moments. However, the MC estimators can also be used to estimate probabilities. In order to see this, assume that we would like to estimate $\mathbb{P}(X \in A)$ where $X \sim p$. We know that this is given as

$$\mathbb{P}(X \in A) = \int_A p(x)\mathrm{d}x,$$

see, e.g., Definition 1.2. For example, A can simply be an interval. Given the definition above, we can write

$$\begin{aligned}
\mathbb{P}(X \in A) &= \int_A p(x)\mathrm{d}x \\
&= \int \mathbf{1}_A(x)p(x)\mathrm{d}x,
\end{aligned}$$

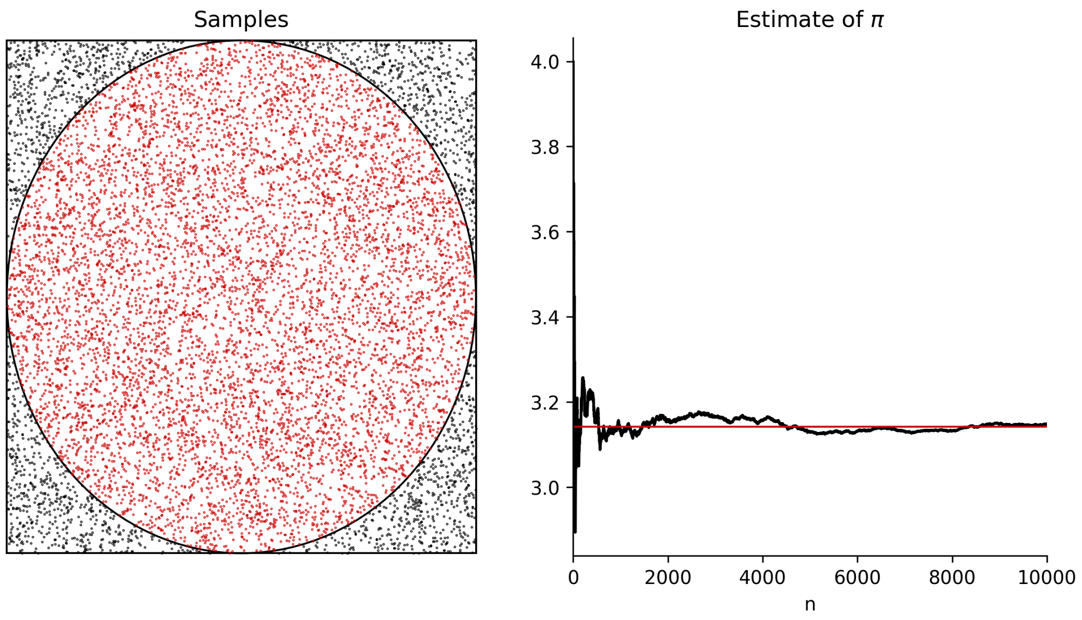


Figure 4.1: Estimating π using the Monte Carlo method.

where $\mathbf{1}_A(x)$ is the indicator function of A. We can therefore set $\varphi(x) = \mathbf{1}_A(x)$ and given the samples from p, we can build an estimator

$$\begin{aligned}
\mathbb{P}(X \in A) &= \int \mathbf{1}_A(x) p(x) \mathrm{d}x, \\
&\approx \int \mathbf{1}_A(x) p^N(\mathrm{d}x), \\
&= \int \mathbf{1}_A(x) \frac{1}{N} \sum_{i=1}^N \delta_{X_i}(\mathrm{d}x), \\
&= \frac{1}{N} \sum_{i=1}^N \int \mathbf{1}_A(x) \delta_{X_i}(\mathrm{d}x), \\
&= \frac{1}{N} \sum_{i=1}^N \mathbf{1}_A(X_i).
\end{aligned}$$

This estimator also leads to an intuitive procedure: We sample $X_1, \ldots, X_N$ from p and we effectively just count the samples in A and divide it by N.

We can now return to the example of estimating π using the Monte Carlo method.

Example 4.1. Recall the problem of estimating π using the Monte Carlo method. Frame it as a Monte Carlo integration problem and derive the algorithm rigorously.

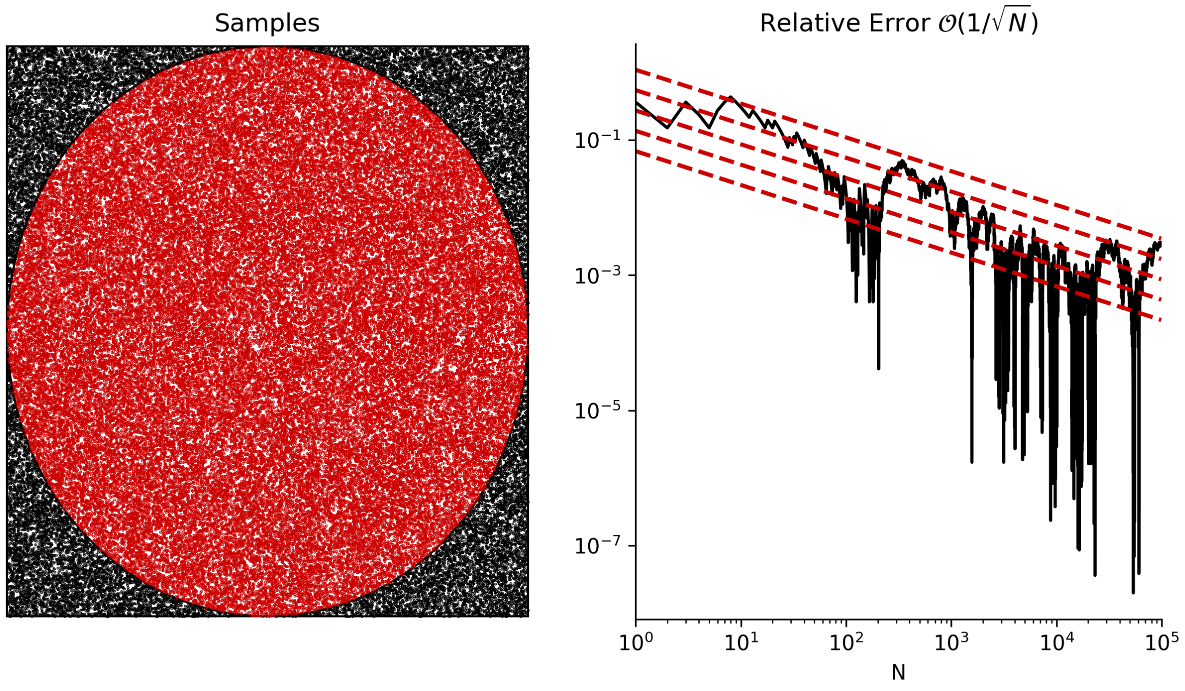


Figure 4.2: Relative error (see next section) of the Monte Carlo estimate provided by sampling within the circle.

Solution. The logic that was used in this example was to estimate the area of a circle that lies within a square. To be precise, consider the square $[-1, 1] \times [-1, 1]$ and define the uniform distribution on this square as $p(x, y) = \text{Unif}([-1, 1] \times [-1, 1])$. We can simply phrase the problem as estimating the area of the circle which we define as $\mathsf{A} \subset [-1, 1] \times [-1, 1]$. The set A is given as

$$\mathsf{A} = \{(x, y) \in [-1, 1] \times [-1, 1] \mid x^2 + y^2 \leq 1\}.$$

We can then formalise this problem as estimating the probability that a point lies within the circle. This is given as

$$\begin{aligned}\mathbb{P}(X \in \mathsf{A}) &= \int_{\mathsf{A}} p(x, y) \mathrm{d}x \mathrm{d}y, \\ &= \int \mathbf{1}_{\mathsf{A}}(x, y) p(x, y) \mathrm{d}x \mathrm{d}y.\end{aligned}$$

Sampling $(X_i, Y_i) \sim p(x, y)$ (a uniform sample within a square), we can estimate this integral using the standard MC method. More formally, we can write

$$\begin{aligned}\mathbb{P}(\mathsf{A}) &= \int_A p(x, y) \mathrm{d}x \mathrm{d}y \\ &= \int \mathbf{1}_{\mathsf{A}}(x, y) p(x, y) \mathrm{d}x \mathrm{d}y, \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{\mathsf{A}}(X_i, Y_i) \to \frac{\pi}{4} \qquad \text{as } N \to \infty.\end{aligned}$$

A trajectory of the estimation procedure π can be seen from Fig. 4.1 w.r.t. varying sample size.

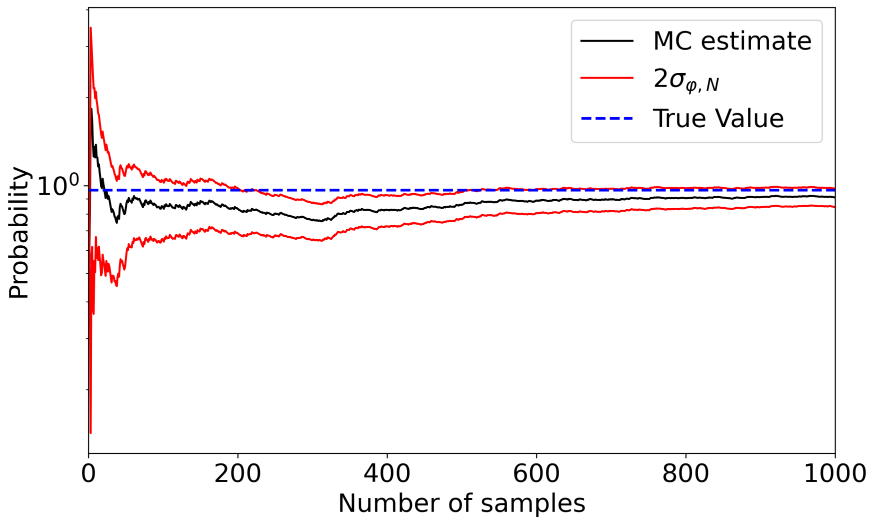


Figure 4.3: Monte Carlo integration of $h(x) = [\cos(50x) + \sin(20x)]^2$

Nonasymptotic results showing the convergence rate of $\mathcal{O}(1/\sqrt{N})$ are also available (see, e.g., Akyildiz (2019, Corollary 2 .1)) – see Fig. 4.2 for a demonstration.

Example 4.2 (Example 3.4 from Robert and Casella (2004))**.** Let us consider an example of estimating an integral:

$$I = \int_0^1 h(x)\mathrm{d}x = \int_0^1 [\cos(50x) + \sin(20x)]^2 \mathrm{d}x.$$

The exact value of this integral is 0.965. Describe a Monte Carlo method to estimate this integral.

Solution. We can just choose $p(x) = \text{Unif}(0, 1)$ and set $\varphi(x) = h(x)$. We can then write

$$\begin{aligned} I &= \int_0^1 h(x)\mathrm{d}x, \\ &= \int_0^1 \varphi(x)p(x)\mathrm{d}x, \end{aligned}$$

and apply the standard MC estimator. The results (together with the empirical variance estimate) can be seen from Fig. 4.3.

Finally, we provide an example of estimating the probability of a random variable.

Example 4.3. Consider $X \sim \mathcal{N}(0, 1)$ and we would like to estimate the probability that $X > 2$. Describe the MC method.

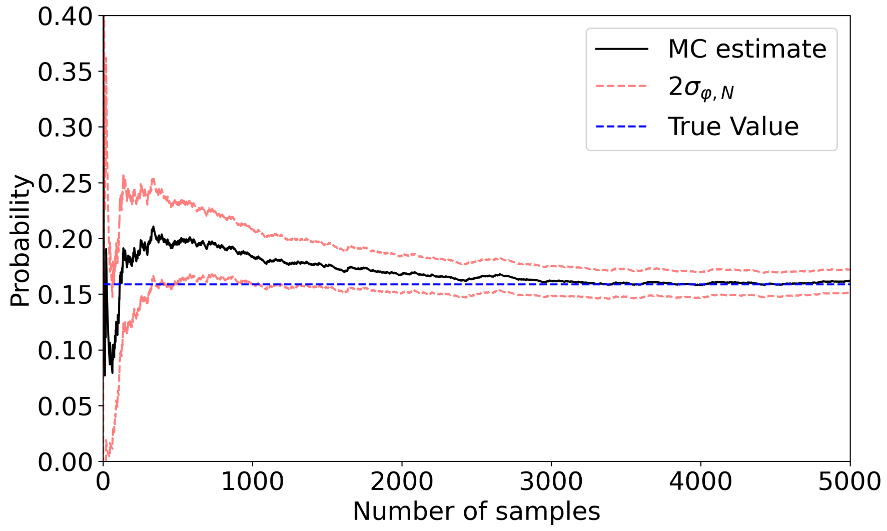


Figure 4.4: Monte Carlo estimation of the tail probability $X > 2$. The “true value” is computed via numerical integration.

Solution. The way to do this is to choose

$$p(x) = \mathcal{N}(0,1), \quad \varphi(x) = \mathbf{1}_{\{x>2\}}(x).$$

We can then write that

$$\begin{aligned}\mathbb{P}(X > 2) &= \int_{-\infty}^{\infty} \mathbf{1}_{\{x>2\}}(x)\mathcal{N}(0,1)\mathrm{d}x, \\ &\approx \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{X_i>2\}}(X_i).\end{aligned}$$

where $X_1, \ldots, X_N \sim \mathcal{N}(0,1)$. The results can be seen from Fig. 4.4.

4.2 ERROR METRICS

In order to quantify convergence, we will have a number of error metrics in the following section. Let us generically denote $\hat{\varphi}^N$ as an estimator of $\bar{\varphi}$ within this section (which can be standard Monte Carlo estimator or any other estimator we will introduce later.

We start with defining the bias as

$$\mathsf{bias}(\hat{\varphi}^N) = \mathbb{E}[\hat{\varphi}^N] - \bar{\varphi}, \tag{4.4}$$

where $\bar{\varphi}$ is the *true value*. We call an estimator *unbiased* if the bias is zero. In the case where we sample i.i.d from $p(x)$, we can build unbiased estimators of expectations and integrals. We recall the variance

$$\mathsf{var}(\hat{\varphi}^N) = \mathbb{E}[(\hat{\varphi}^N - \mathbb{E}[\hat{\varphi}^N])^2]. \tag{4.5}$$

If the estimator is unbiased, we can then replace $\mathbb{E}[\hat{\varphi}^N]$ with $\bar{\varphi}$. Next, we define the mean squared error (MSE)

$$\mathsf{MSE}(\hat{\varphi}^N) = \mathbb{E}[(\hat{\varphi}^N - \bar{\varphi})^2]. \tag{4.6}$$

One can see that the MSE and the variance coincides if the estimator is unbiased. We have also the following decomposition of the MSE

$$\mathsf{MSE}(\hat{\varphi}^N) = \mathsf{bias}(\hat{\varphi}^N)^2 + \mathsf{var}(\hat{\varphi}^N). \tag{4.7}$$

We define the root mean square error (RMSE) as

$$\mathsf{RMSE}(\hat{\varphi}^N) = \sqrt{\mathsf{MSE}(\hat{\varphi}^N)}. \tag{4.8}$$

Finally, we define the relative absolute error (RAE) as

$$\mathsf{RAE}(\hat{\varphi}^N) = \frac{|\hat{\varphi}^N - \bar{\varphi}|}{|\bar{\varphi}|}. \tag{4.9}$$

We usually plot the absolute error of the estimator, as we only run the experiment once in general[2]. We note that this absolute error $|\hat{\varphi}^N - \bar{\varphi}|$ is a random variable (since no expectations are taken). However, this quantity provably converges with a rate of $\mathcal{O}(1/\sqrt{N})$ (see, e.g., Akyildiz (2019, Corollary 2.1)) for the standard Monte Carlo estimator. More precisely, we can write

$$|\hat{\varphi}^N_{\text{MC}} - \bar{\varphi}| \leq \frac{V}{\sqrt{N}}, \tag{4.10}$$

where V is an almost surely finite random variable. This error rate will be displayed empirically in the following sections (see also Fig . 4.2).

Example 4.4 (Marginal Likelihood estimation)**.** Recall that, given a prior $p(x)$ and a likelihood $p(y|x)$, we can compute the marginal likelihood $p(y)$ as

$$p(y) = \int p(y|x)p(x)\mathrm{d}x.$$

Describe the MC method to estimate $p(y)$.

Solution. This defines a nice integration problem that we can solve using MC. Assume that we are given the following model

$$\begin{aligned} p(x) &= \mathcal{N}(x; \mu_0, \sigma_0^2), \\ p(y|x) &= \mathcal{N}(y; x, \sigma^2). \end{aligned}$$

[2]However, if you were to do a proper experimentation, then you would have to run the same experiment M times (Monte Carlo runs) and average the error to estimate the RMSE.

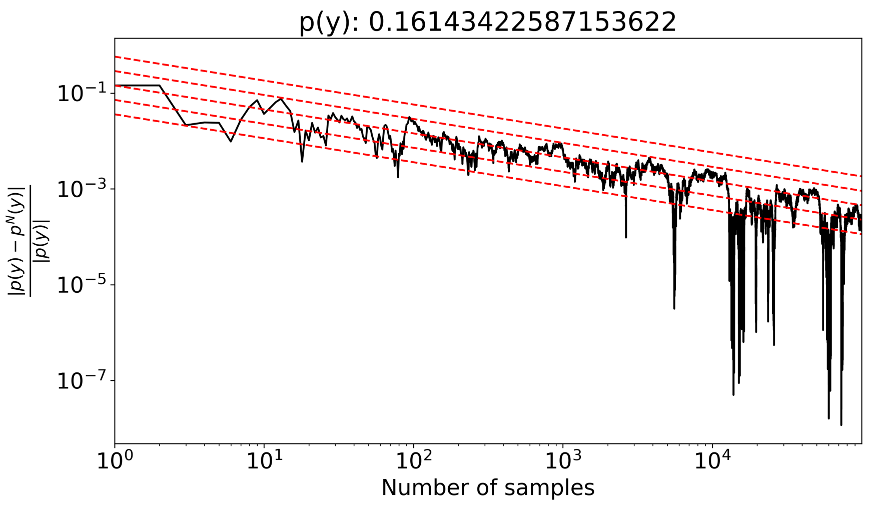


Figure 4.5: Estimating the marginal likelihood $p(y)$ for $y = 1$. One can clearly see the displayed error rate that is $\mathcal{O}(1/\sqrt{N})$.

Assume that $\mu_0 = 0$, $\sigma_0 = 1$, $\sigma = 2$, and $y = 1$. For example, for fixed $y = 1$, we would want to estimate $p(y = 1)$. This integral becomes

$$p(y = 1) = \int p(y = 1|x)p(x)\mathrm{d}x,$$

where we can set $\varphi(x) = p(y = 1|x)$. We can then compute the integral using MC estimation procedure as

$$p^N(y = 1) = \frac{1}{N}\sum_{i=1}^{N} p(y = 1|X_i),$$

where $X_1, \ldots, X_N \sim p(x)$. The results can be seen from Fig. 4.5.

We will next consider another example of estimating a probability where we show how to quantify the variance using the true value.

Example 4.5. Consider the following density

$$p(x) = \frac{1}{\pi(1 + x^2)}.$$

We would like to compute the probability of $X \sim p(x)$ being larger than 2, i.e., $\mathbb{P}(X > 2)$. Describe the method and suggest an improvement.

Solution. We can compute this probability using MC estimation as we can set

$$\varphi(x) = \mathbf{1}_{\{x>2\}}(x).$$

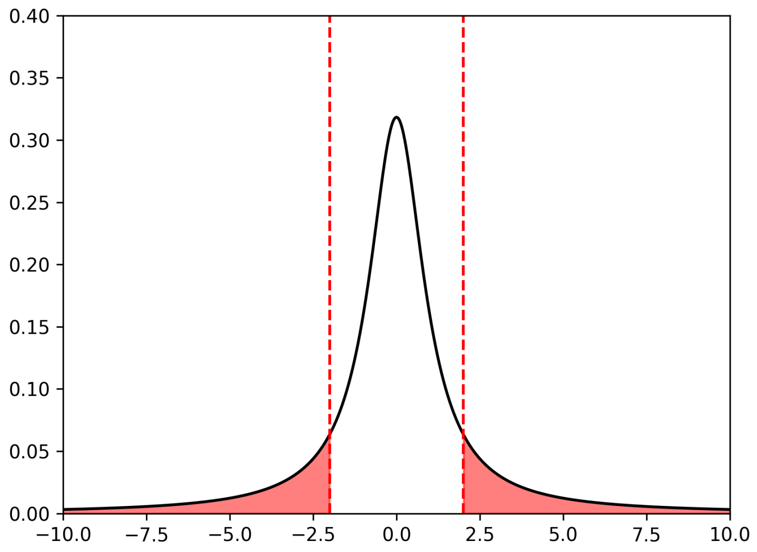


Figure 4.6: Cauchy density of Example 4.5.

We can compute

$$\begin{aligned}\mathbb{P}(X > 2) &= \int_2^\infty p(x)\mathrm{d}x \\ &= \int \mathbf{1}_{\{x>2\}}(x)p(x)\mathrm{d}x.\end{aligned}$$

We can also compute the real value of this integral as (see Example 2.3 for the CDF of this density)

$$I = \bar{\varphi} = \int_2^\infty p(x)\mathrm{d}x = F_X(\infty) - F_X(2) = \frac{1}{2} - \frac{1}{\pi}\tan^{-1}(2) = 0.1476.$$

Let us compute the variance of the Monte Carlo estimator for $N = 10$ samples:

$$\mathsf{var}(\hat{\varphi}^N_{\mathrm{MC}}) = \frac{\mathsf{var}_p(\varphi)}{N}$$

So we need to compute:

$$\begin{aligned}\mathsf{var}_p(\varphi) &= \int \varphi(x)^2 p(x)\mathrm{d}x - \left(\int \varphi(x)p(x)\mathrm{d}x\right)^2 \\ &= \int \mathbf{1}_{\{x>2\}}(x)^2 p(x)\mathrm{d}x - \left(\int \mathbf{1}_{\{x>2\}}(x)p(x)\mathrm{d}x\right)^2 \\ &= \int \mathbf{1}_{\{x>2\}}(x)p(x)\mathrm{d}x - \left(\int \mathbf{1}_{\{x>2\}}(x)p(x)\mathrm{d}x\right)^2 \\ &= 0.1476 - 0.1476^2 = 0.125.\end{aligned}$$

The variance of the estimator then

$$\mathsf{var}(\hat{\varphi}^N_{\mathrm{MC}}) = \frac{0.125}{10} = 0.0125.$$

Could we do better? An idea is to use the fact that the density is symmetric around zero: This means $P(X > 2) = P(X < -2)$ (see Fig . 4.6). So we could compute:

$$\mathbb{P}(|X| > 2) = \mathbb{P}(X > 2) + \mathbb{P}(X < -2) = 2I.$$

Therefore, our new problem is $I = \frac{1}{2}\mathbb{P}(|X| > 2)$. Let us write it as

$$\begin{aligned} I &= \frac{1}{2}\int_{|x|>2} p(x)\mathrm{d}x, \\ &= \int \frac{1}{2}\mathbf{1}_{\{|x|>2\}}(x)p(x)\mathrm{d}x, \end{aligned}$$

Now define the test function

$$\varphi(x) = \frac{1}{2}\mathbf{1}_{\{|x|>2\}}(x).$$

As before, we need to compute $\mathsf{var}_p(\varphi)$:

$$\begin{aligned} \mathsf{var}_p(\varphi) &= \int \varphi(x)^2 p(x)\mathrm{d}x - \left(\int \varphi(x)p(x)\mathrm{d}x\right)^2 \\ &= \int \frac{1}{4}\mathbf{1}^2_{\{|x|>2\}}p(x)\mathrm{d}x - \left(\int \frac{1}{2}\mathbf{1}_{\{|x|>2\}}p(x)\mathrm{d}x\right)^2 \\ &= \int \frac{1}{4}\mathbf{1}_{\{|x|>2\}}p(x)\mathrm{d}x - \left(\int \frac{1}{2}\mathbf{1}_{\{|x|>2\}}p(x)\mathrm{d}x\right)^2 \\ &= \frac{1}{4} \times 2 \times 0.1476 - \frac{1}{4} \times (2 \times 0.1476)^2, \\ &= 0.052. \end{aligned}$$

Therefore, the variance of the estimator for $N = 10$ samples is

$$\mathsf{var}(\hat{\varphi}^N_{\text{MC}}) = \frac{0.052}{10} = 0.0052.$$

Improvement over the previous estimator! This kind of variance improvements are crucial in safety critical applications.

4.3 IMPORTANCE SAMPLING

While the estimators constructed using samples exactly coming from p has desirable properties as we have seen above, in the majority of cases, we need to employ more complex sampling strategies. A few cases where we need this are summarised below.

- A typical problem arises when computing tail probabilities (also called *rare events*). We may have access to samples directly from $p(x)$, however, sampling from the tail of $p(x)$ might be extremely difficult. For example, consider the Gaussian random variable X

with mean 0 and variance 1. The probability of X being larger than 4 is very small, i.e., $\mathbb{P}(X > 4) \approx 0.00003$. Sampling from the tail of this density directly would be very inefficient without further tricks.

- Another typical scenario where we may want to compute expectations with respect to $p(x)$ when we do not have direct samples from it. The standard example for this is the Bayesian setting. Given a prior $p(x)$ and a likelihood $p(y|x)$, we may want to compute the expectations w.r.t. the posterior density $p(x|y)$, i.e., $\mathbb{E}_{p(x|y)}[\varphi(X)]$. In this case, we do not have access to samples from $p(x|y)$ so we need to employ other strategies.

A strategy we will pursue in this section is specific to *Monte Carlo integration*. In other words, we will next describe a strategy where we can compute integrals and expectations w.r.t. a probability density without having access to samples from it. This is slightly different than directly aiming at *sampling* from the density (which can also be used to estimate integrals). While we will look at sampling methods in the following chapters, it is important to note that importance sampling is primarily an integration technique.

4.3.1 BASIC IMPORTANCE SAMPLING

Consider the basic task of estimating the integral

$$\bar{\varphi} = \mathbb{E}_p[\varphi(X)] = \int \varphi(x) p(x) \mathrm{d}x.$$

In this section, as opposed to previous sections, we assume that we cannot sample from p directly (or exactly, e.g., using rejection sampling[3]). However, we assume (in this section) we can evaluate the density $p(x)$ pointwise. We can still estimate this expectation (and compute integrals more generally), using samples from an instrumental, *proposal* distribution q. In other words, we can sample from a *proposal* and we can repurpose these samples to estimate expectations w.r.t. $p(x)$. This resembles the rejection sampling where we have also used a proposal to accept-reject samples. However, in this case, we will employ a different strategy of *weighting* samples and will not throw any of the samples away. The weights we will compute will *weight samples* so that the integral estimate gets closer to the true integral. In order to see how to do this, we compute

$$\begin{aligned}
\bar{\varphi} &= \int \varphi(x) p(x) \mathrm{d}x, \\
&= \int \varphi(x) \frac{p(x)}{q(x)} q(x) \mathrm{d}x, \qquad \text{“identity trick”} && (4.11) \\
&= \int \varphi(x) w(x) q(x) \mathrm{d}x, && (4.12)
\end{aligned}$$

where $w(x) = p(x)/q(x)$ (which is called the *weight function*). We know from Section 4.1 that we can estimate the integral in (4.12) using samples from q. Let $X_i \sim q$ be i.i.d samples from q for $i = 1, \ldots, N$. We can then estimate the integral in (4.12), hence the expectation $\bar{\varphi}$ using

$$\begin{aligned}
\bar{\varphi} &= \int \varphi(x) w(x) q(x) \mathrm{d}x \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \mathsf{w}_i \varphi(X_i) = \hat{\varphi}_{\text{IS}}^N, && (4.13)
\end{aligned}$$

[3]Recall that rejection sampling draws i.i.d samples from the density, not *approximate*.

where $\mathsf{w}_i = w(X_i) = p(X_i)/q(X_i)$ are called the *weights*. The weights will play a crucial role throughout this section. The key idea of importance sampling is that, instead of throwing away the samples by rejection, we could reweight them according to their importance. This is why this strategy is called *importance sampling* (IS).

The importance sampling algorithm for this case then can be described relatively straightforwardly. Given $p(x)$ (which we can evaluate), we choose a proposal $q(x)$. Then, we sample $X_i \sim q(x)$ for $i = 1, \ldots, N$ and compute the IS estimator as

$$\hat{\varphi}_{\mathrm{IS}}^N = \frac{1}{N} \sum_{i=1}^N \mathsf{w}_i \varphi(X_i),$$

where $\mathsf{w}_i = \frac{p(X_i)}{q(X_i)}$ for $i = 1, \ldots, N$ are the importance weights. We summarise the method in Algorithm 7. In what follows, we will discuss some details of the method.

Algorithm 7 Pseudocode for basic importance sampling

```
1: Input: The number of samples $N$
2: for $i = 1, \ldots, N$ do
3:     Sample $X_i \sim q(x)$
4:     Compute weights $\mathsf{w}_i = \frac{p(X_i)}{q(X_i)}$
5: end for
6: Report the estimator
       $$\hat{\varphi}_{\mathrm{IS}}^N = \frac{1}{N} \sum_{i=1}^N \mathsf{w}_i \varphi(X_i).$$
```

Remark 4.4. Note that one can also consider IS approximations as the approximation to the probability measure $p(\mathrm{d}x)$ as

$$p_{\mathrm{IS}}^N(\mathrm{d}x) = \frac{1}{N} \sum_{i=1}^N \mathsf{w}_i \delta_{X_i}(\mathrm{d}x),$$

and hence $\hat{\varphi}_{\mathrm{IS}}^N = (\varphi, p_{\mathrm{IS}}^N)$. This is a useful way to think about the IS estimator later on in some proofs.

Remark 4.5. Unlike rejection sampling, in importance sampling, the proposal does not have to dominate the target density. Instead, the crucial requirement for the IS is that the support of the proposal should be the same as the support of the density. More precisely, we need $q(x) > 0$ whenever $p(x) > 0$. This is far less restrictive than the requirement of rejection sampling. Of course, the choice of proposal can still effect the performance of the IS. We will discuss this in more detail.

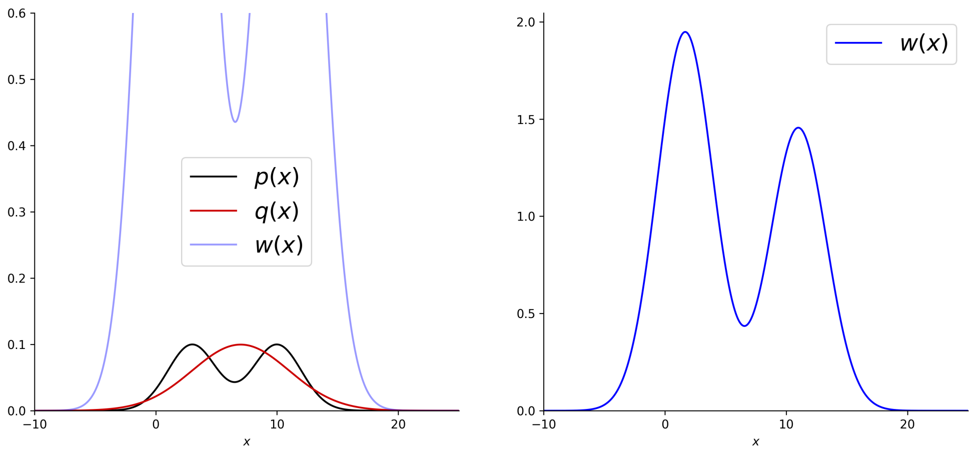


Figure 4.7: An example of target density $p(x)$, the proposal $q(x)$ and the associated weight function $w(x)$. One can see that if $q(x) < p(x)$ (which means fewer samples would be drawn from $q(x)$ in this region), then $w(x) > 1$ to account for this effect. The opposite is also true, since if $q(x) > p(x)$, this means that we would draw more samples than necessary, which should be downweighted, hence $w(x) < 1$ in these regions.

From Fig. 4.7, one can see an example plot of the target density $p(x)$, the proposal $q(x)$ and the associated weight function $w(x)$. See the caption for more details and intuition.

Example 4.6. Consider the problem of estimating $\mathbb{P}(X > 4)$ for $X \sim \mathcal{N}(0,1)$. Describe a potential problem of using the naive MC method.

Solution. While we can exactly sample from this density, given that

$$\mathbb{P}(X > 4) = 3.16 \times 10^{-5},$$

it will be the case that very few of the samples from exact distribution will fall into this tail (Note that, while we know the exact value in this case, we will not know this in general – this is just a demonstrative example). In fact, a standard run with $N = 10000$ gives exactly zero samples that satisfy $X_i > 4$, hence provides the estimate as zero! It is obvious that this is not a great way to estimate the probability and we can use importance sampling for this. Consider a proposal $q(x) = \mathcal{N}(6,1)$. This will draw a lot of samples from the region $X > 4$ and we can reweight this samples w.r.t. the target density using the IS estimator in (4.13). A standard run in this case with $N = 10000$ results in

$$\hat{\varphi}_{\text{IS}}^N = 3.18 \times 10^{-5},$$

which is obviously a much closer number to the truth.

One can next prove that the estimator $\hat{\varphi}_{\text{IS}}^N$ is unbiased.

Proposition 4.3. *The estimator $\hat{\varphi}_{\mathrm{IS}}^N$ is unbiased, i.e.,*

$$\mathbb{E}_{q(x)}[\hat{\varphi}_{\mathrm{IS}}^N] = \bar{\varphi}.$$

Proof. We simply write

$$\begin{aligned}
\mathbb{E}_q[\hat{\varphi}_{\mathrm{IS}}^N] &= \mathbb{E}_{q(x)}\left[\frac{1}{N}\sum_{i=1}^N \mathsf{w}_i \varphi(X_i)\right] \\
&= \mathbb{E}_q\left[\frac{1}{N}\sum_{i=1}^N \frac{p(X_i)}{q(X_i)}\varphi(X_i)\right] \\
&= \frac{1}{N}\sum_{i=1}^N \mathbb{E}_q\left[\frac{p(X_i)}{q(X_i)}\varphi(X_i)\right] \\
&= \frac{1}{N}\sum_{i=1}^N \int \frac{p(x)}{q(x)}\varphi(x)q(x)\mathrm{d}x \qquad \text{since } X_i \sim q(x) \\
&= \int \varphi(x)p(x)\mathrm{d}x, \\
&= \bar{\varphi},
\end{aligned}$$

which completes the proof. □

An important quantity in IS is the *variance* of the estimator $\hat{\varphi}_{\mathrm{IS}}^N$. The variance of the estimator is a measure of how much the estimator fluctuates around its expected value. The variance of the IS estimator (4.13) is given by the following proposition.

Proposition 4.4. *The variance of the estimator $\hat{\varphi}_{\mathrm{IS}}^N$ is given by*

$$\mathsf{var}_q[\hat{\varphi}_{\mathrm{IS}}^N] = \frac{1}{N}\left(\mathbb{E}_q[w^2(X)\varphi^2(X)] - \bar{\varphi}^2\right).$$

Proof. Next we write out the estimator $\hat{\varphi}_{\mathrm{IS}}^N$ in (4.13)

$$\begin{aligned}
\mathsf{var}_q[\hat{\varphi}_{\mathrm{IS}}^N] &= \mathsf{var}_q\left[\frac{1}{N}\sum_{i=1}^N \mathsf{w}_i\varphi(X_i)\right] \\
&= \frac{1}{N^2}\mathsf{var}_q\left[\sum_{i=1}^N w(X_i)\varphi(X_i)\right] \\
&= \frac{1}{N}\mathsf{var}_q\left[w(X)\varphi(X)\right] \qquad \text{where } X \sim q(x) \\
&= \frac{1}{N}\left(\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right] - \mathbb{E}_q\left[w(X)\varphi(X)\right]^2\right) \\
&= \frac{1}{N}\left(\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right] - \bar{\varphi}^2\right),
\end{aligned}$$

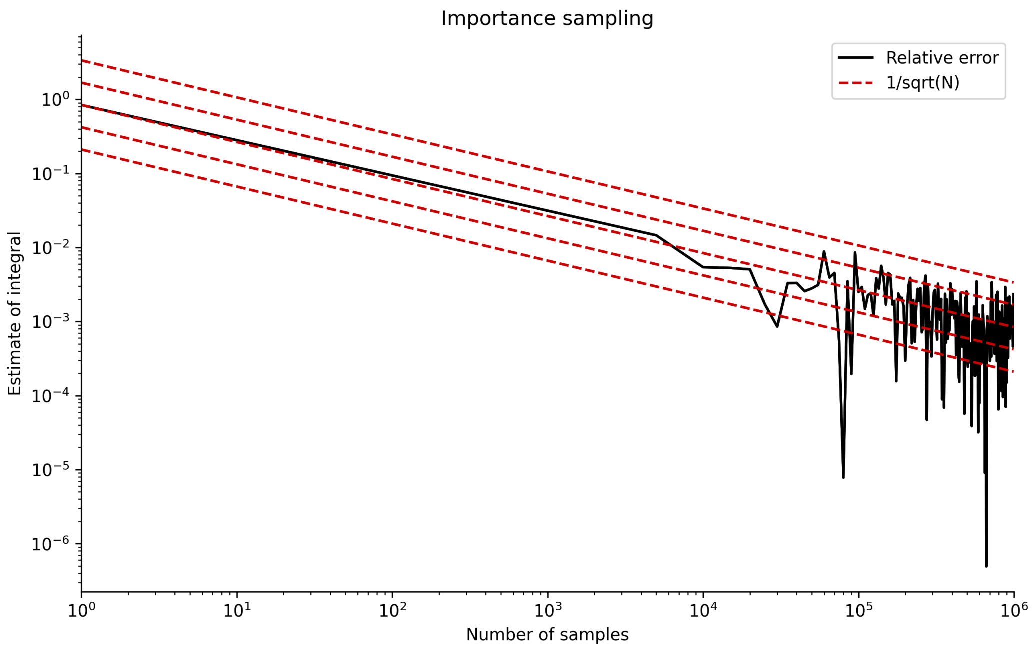


Figure 4.8: The importance sampling estimator $\hat{\varphi}_{\text{IS}}^N$ is plotted against the number of samples N for the example in in Fig . 4.7, for $\varphi(x) = x^2$. This demonstrates that the random error in the IS case also satisfies $\mathcal{O}(1/\sqrt{N})$ convergence rate.

which concludes the proof. We have used the fact that the variance of the sum of independent random variables is the sum of the variances. □

One can see that this easily leads to the bound for the standard devation $\text{std}_q[\hat{\varphi}_{\text{IS}}^N] \leq \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$. Also, we still have the result for the relative absolute error as

$$|\hat{\varphi}_{\text{IS}}^N - \bar{\varphi}| \leq \frac{V}{\sqrt{N}},$$

where V is an almost surely finite random variable. As in the perfect MC case, we will not prove this result as it is beyond our scope, but curious reader can refer to Corollary 2.2 in Akyildiz (2019) (which also holds for the self normalised case which will be introduced below). A demonstration of this rate for importance sampling can be seen from Fig. 4.8.

We can see that the variance of the IS estimator is finite if

$$\mathbb{E}_q[w^2(X)\varphi^2(X)] < \infty.$$

This implies that

$$\int w^2(x)\varphi^2(x)q(x)\mathrm{d}x = \int \frac{p(x)}{q(x)}\varphi^2(x)p(x)\mathrm{d}x < \infty.$$

In other words, for our importance sampling estimate to be well-defined, the ratio

$$\frac{p^2(x)}{q(x)}\varphi^2(x)$$

has to be integrable. We will see next an example where this condition is not satisfied.

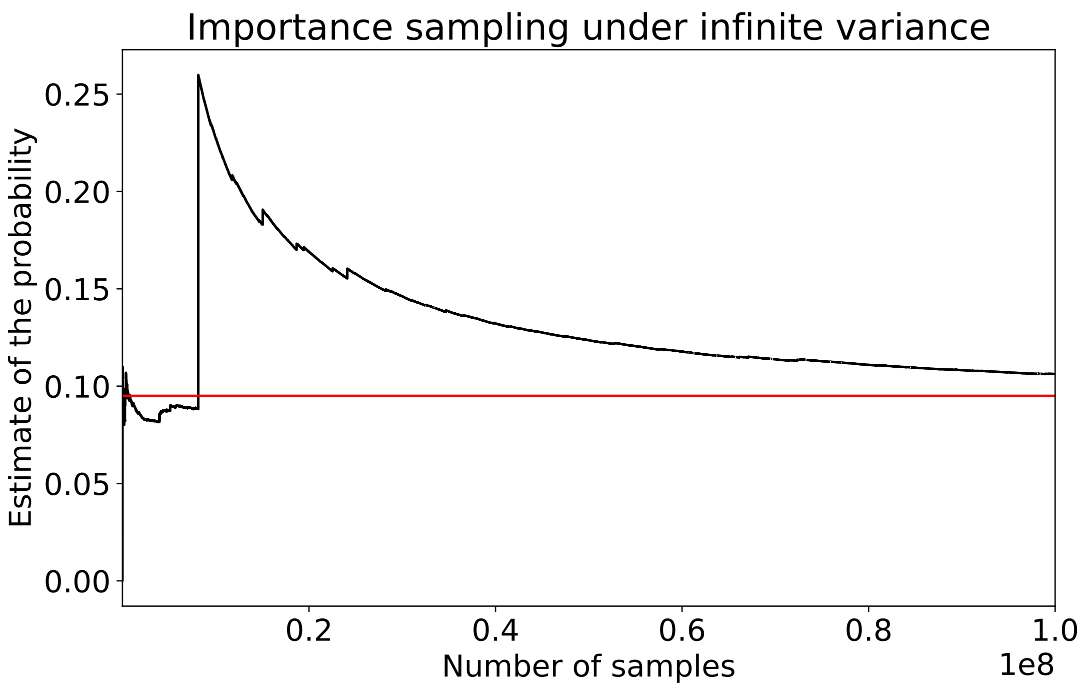


Figure 4.9: Estimating $\mathbb{P}(2 < X < 6)$ where X is Cauchy with $q(x) = \mathcal{N}(0, 1)$. The true value is plotted in red and the estimator value in black.

Example 4.7 (Infinite variance IS, Example 3.8 from Robert and Casella (2010))**.** Consider the target

$$p(x) = \frac{1}{\pi}\frac{1}{1+x^2},$$

which is the Cauchy density. Let us choose the proposal

$$q(x) = \mathcal{N}(x; 0, 1) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right).$$

Discuss whether the IS estimator is well-defined.

Solution. We can compute the ratio $\frac{p(x)}{q(x)} \propto \exp(x^2/2)/(1 + x^2)$ and see that it is explosive (precisely, its integral is infinite). This can result in problematic situations even if φ ensures that the variance is finite. For example, consider the problem of estimating $\mathbb{P}(2 < X < 6)$. One example run for this case can be seen from Fig. 4.9. One can see that the estimator in this case is unstable and cannot be reliably used.

Remark 4.6 (Optimal proposal)**.** We can try to inspect the variance expression to figure out which proposals can give us variance reduction. From Prop. 4.4, it follows that we have

$$\mathsf{var}_q[\hat{\varphi}^N_{\text{IS}}] = \frac{1}{N}\mathsf{var}_q\left[w(X)\varphi(X)\right].$$

This means that minimising the variance of the IS estimator is the same as minmising the variance of the function $w(x)\varphi(x)$. Moreover, looking at the expression,

$$\mathsf{var}_q[\hat{\varphi}^N_{\text{IS}}] = \frac{1}{N}\left(\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right] - \bar{\varphi}^2\right),$$

we can see that since $\bar{\varphi}^2 > 0$ (which is independent of the proposal), we should choose a proposal that minimises $\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right]$. We can lower bound this quantity using Jensen's inequality:

$$\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right] \geq \mathbb{E}_q\left[w(X)|\varphi(X)|\right]^2,$$

where we used the fact that $(\cdot)^2$ is a convex function (For a convex function f, Jensen's inequality states that $\mathbb{E}_q[f(X)] \geq f(\mathbb{E}_q[X])$.). Using $w(x) = p(x)/q(x)$, we arrive at the following lower bound:

$$\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right] \geq \mathbb{E}_p\left[|\varphi(X)|\right]^2. \tag{4.14}$$

Now let us expand the term $\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right]$ out and write

$$\begin{aligned}
\mathbb{E}_q\left[w^2(X)\varphi^2(X)\right] &= \mathbb{E}_q\left[\frac{p^2(X)}{q^2(X)}\varphi^2(X)\right] \\
&= \int \frac{p^2(x)}{q^2(x)}\varphi^2(x)q(x)\mathrm{d}x \\
&= \int p(x)\frac{p(x)}{q(x)}\varphi^2(x)\mathrm{d}x, \\
&= \mathbb{E}_p\left[w(X)\varphi^2(X)\right]. \qquad (4.15)
\end{aligned}$$

The last equation, eq. (4.15), suggests that we can choose a proposal such that we attain the lower bound of this function (4.14) (which means that it would be the minimiser). In particular, if we choose a proposal $q(x)$ such that

$$w(x) = \frac{p(x)}{q(x)} = \frac{\mathbb{E}_p[|\varphi(X)|]}{|\varphi(x)|}$$

is satisfied, then (4.15) would be equal to the lower bound (4.14). This implies that

$$q_\star(x) = p(x)\frac{|\varphi(x)|}{\mathbb{E}_p[|\varphi(X)|]}, \tag{4.16}$$

would minimise the variance of the importance sampling estimator.

Choosing $q_\star$ as the proposal, one can see that the variance of the IS estimator satisfies

$$\begin{aligned}\mathsf{var}_{q_\star}[\hat{\varphi}_{\text{IS}}^N] &= \frac{1}{N}\mathbb{E}_p\left[|\varphi(X)|\right]^2 - \frac{1}{N}\bar{\varphi}^2 \\ &\leq \frac{1}{N}\mathbb{E}_p\left[\varphi^2(X)\right] - \frac{1}{N}\bar{\varphi}^2 \\ &= \mathsf{var}_p[\hat{\varphi}_{\text{MC}}^N],\end{aligned}$$

therefore we obtain

$$\mathsf{var}_{q_\star}[\hat{\varphi}_{\text{IS}}^N] \leq \mathsf{var}_p[\hat{\varphi}_{\text{MC}}^N],$$

i.e., a variance reduction. In fact, one can show that, if $\varphi(x) \geq 0$ for all $x \in \mathbb{R}$, then the variance of the IS estimator with optimal proposal $q_\star$ is equal to zero.

We note that this optimal construction of the proposal (4.16) is not possible to implement in practice. It requires the knowledge of the very quantity we want to estimate, namely, $\mathbb{E}_p[|\varphi(X)|]$! But in general, we can choose proposals that minimise the variance of the IS estimator where possible. This idea has been used in the literature to construct proposals that minimise the variance of the estimator, see, e.g., Akyildiz and Míguez (2021) and references therein. Within the context of this course, we will construct some simple examples for this purpose later.

4.3.2 SELF-NORMALISED IMPORTANCE SAMPLING

As mentioned several times in past chapters, in many scenarios, we have access to the *unnormalised* density, i.e., given p, we can evaluate it up to a normalising constant. As usual, we denote this density $\bar{p}(x)$ and recall that it is related to p by

$$p(x) = \frac{\bar{p}(x)}{Z}$$

where $Z = \int \bar{p}(x)\mathrm{d}x$. In the context of Bayesian inference, we usually have an unnormalised posterior density $\bar{p}(x|y) \propto p(y|x)p(x)$. In the previous section, we have built an importance sampling estimator for the case where we have access to the normalised density. In this section, we will generalise the idea and assume we only have access to the unnormalised density.

Consider, again, the problem of estimating expectations of a given density p. For the case where we can only evaluate $\bar{p}(x)$, one way to estimate this expectation is to sample from a proposal distribution q and rewrite the integral as

$$\begin{aligned}\bar{\varphi} &= \int \varphi(x)p(x)\mathrm{d}x, \\ &= \frac{\int \varphi(x)\frac{\bar{p}(x)}{q(x)}q(x)\mathrm{d}x}{\int \frac{\bar{p}(x)}{q(x)}q(x)\mathrm{d}x},\end{aligned} \tag{4.17}$$

where we use the fact that $p(x) = \bar{p}(x)/Z$. This gives us two separate integration problems, one to estimate the numerator and one to estimate the denominator. We will estimate both quantities using samples from $q(x)$.

Algorithm 8 Pseudocode for self-normalised importance sampling

```
1: Input: The number of samples $N$
2: for $i = 1, \ldots, N$ do
3:     Sample $X_i \sim q(x)$
4:     Compute unnormalised weights $W(X_i) = \frac{\bar{p}(X_i)}{q(X_i)}$
5:     Normalise:
           $$\bar{\mathsf{w}}_i = \frac{W(X_i)}{\sum_{i=1}^N W(X_i)}.$$
6: end for
7: Report the estimator
       $$\hat{\varphi}^N_{\text{SNIS}} = \sum_{i=1}^N \bar{\mathsf{w}}_i \varphi(X_i).$$
```

Let us now introduce the unnormalised weight function $W(x)$

$$W(x) = \frac{\bar{p}(x)}{q(x)},$$

which is analogous to the normalised weight function $w(x)$ in the previous section. Using $X_i \sim q(x)$ and building the Monte Carlo estimator of the numerator and denominator, we arrive at the following estimator of the (4.17):

$$\begin{aligned}
\hat{\varphi}^N_{\text{SNIS}} &= \frac{\frac{1}{N}\sum_{i=1}^N \varphi(X_i)W(X_i)}{\frac{1}{N}\sum_{i=1}^N W(X_i)}, \\
&= \frac{\sum_{i=1}^N \varphi(X_i)W(X_i)}{\sum_{i=1}^N W(X_i)} \\
&= \sum_{i=1}^N \bar{\mathsf{w}}_i \varphi(X_i), \qquad (4.18)
\end{aligned}$$

where

$$\bar{\mathsf{w}}_i = \frac{W(X_i)}{\sum_{i=1}^N W(X_i)},$$

are the *normalised* weights. This estimator (4.18) is called the self-normalised importance sampling (SNIS) estimator The full scheme given in Algorithm 8.

Remark 4.7 (Bias and variance)**.** As opposed to the normalised case, the estimator $\hat{\varphi}^N_{\text{SNIS}}$ is *biased*. The reason of this bias can be seen by recalling the integral (4.17). By sampling from $q(x)$, we can construct unbiased estimates of the numerator and denominator. However, the ratio of these two quantities is biased in general. However, it can be shown that the bias of the SNIS estimator decreases with a rate $\mathcal{O}(1/N)$ (Agapiou et al., 2017).

Since the SNIS estimator is biased, we cannot use the same variance formula as in the previous section. It makes sense to consider the $\mathsf{MSE}(\hat{\varphi}^N_{\text{SNIS}})$ instead. However, this quantity is challenging to control in general – without bounded test functions. With bounded test functions, it is possible to show that the $\mathsf{MSE}(\hat{\varphi}^N_{\text{SNIS}})$ is controlled with a rate $\mathcal{O}(1/N)$ (Agapiou et al., 2017; Akyildiz and Míguez, 2021), which we show below.

In the following, we provide the MSE bound for the SNIS estimator see, e.g., Agapiou et al. (2017) or Akyildiz and Míguez (2021). We use here the notation of Akyildiz and Míguez (2021, Theorem 1) for the proof.

Proposition 4.5. *Assume for simplicity that* $\sup_x |\varphi(x)| \leq 1$. *Then,*

$$\mathsf{MSE}(\hat{\varphi}^N_{SNIS}) = \mathbb{E}_q\left[(\hat{\varphi}^N_{SNIS} - \bar{\varphi})^2\right] \leq \frac{4}{N}\frac{\mathbb{E}_q\left[W^2(X)\right]}{\left(\mathbb{E}_q\left[W(X)\right]\right)^2}.$$

Proof. We start by recalling our notation for integrals $(\varphi, p) := \int \varphi(x)p(x)\mathrm{d}x$ and Monte Carlo approximations $(\varphi, p^N) := (1/N)\sum_{i=1}^N \varphi(X_i)$ for $X_i \sim p$ which will come in handy in this proof. Let us write our SNIS approximation of the measure p as (recall Remark 4.4)

$$p^N_{\text{SNIS}}(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^N \bar{\mathsf{w}}_i \delta_{X_i}(\mathrm{d}x),$$

hence that $\hat{\varphi}^N_{\text{SNIS}} = (\varphi, p^N_{\text{SNIS}})$. Let us also recall in this notation

$$\bar{\varphi} = (\varphi, p) = \int \varphi(x)p(x)\mathrm{d}x = \frac{\int \varphi(x)\frac{\bar{p}(x)}{q(x)}q(x)\mathrm{d}x}{\int \frac{\bar{p}(x)}{q(x)}q(x)\mathrm{d}x} = \frac{(\varphi W, q)}{(W, q)}.$$

With a similar derivation, we can also write

$$\hat{\varphi}^N_{\text{SNIS}} = \frac{(\varphi W, q^N)}{(W, q^N)}.$$

Now next we write

$$\begin{aligned}
|(\varphi, p) - (\varphi, p^N_{\text{SNIS}})| &= \left|\frac{(\varphi W, q)}{(W, q)} - \frac{(\varphi W, q^N)}{(W, q^N)}\right| \\
&= \left|\frac{(\varphi W, q)}{(W, q)} - \frac{(\varphi W, q^N)}{(W, q^N)} + \frac{(\varphi W, q^N)}{(W, q)} - \frac{(\varphi W, q^N)}{(W, q)}\right| \\
&\leq \left|\frac{(\varphi W, q)}{(W, q)} - \frac{(\varphi W, q^N)}{(W, q)}\right| + \left|\frac{(\varphi W, q^N)}{(W, q)} - \frac{(\varphi W, q^N)}{(W, q^N)}\right| \\
&= \frac{1}{(W, q)}\left|(\varphi W, q) - (\varphi W, q^N)\right| + \left|(\varphi W, q^N)\right|\left|\frac{1}{(W, q)} - \frac{1}{(W, q^N)}\right|.
\end{aligned}$$

At this point, we use $\sup_x |\varphi(x)| \leq 1$ (note that, this is without loss of generality for bounded test functions) and get $|(\varphi W, q^N)| \leq |(W, q^N)|$. Using this, we can write

$$\begin{aligned}
|(\varphi, p) - (\varphi, p^N_{\text{SNIS}})| &\leq \frac{1}{(W,q)} \left|(\varphi W, q) - (\varphi W, q^N)\right| + |(W, q^N)| \left| \frac{(W,q^N) - (W,q)}{(W,q^N)(W,q)} \right| \\
&= \frac{1}{(W,q)} \left(\left|(\varphi W, q) - (\varphi W, q^N)\right| + \left|(W, q^N) - (W,q)\right| \right).
\end{aligned}$$

Next, we take squares of both sides (since both sides are nonnegative), use $(a+b)^2 \leq 2(a^2+b^2)$ and take expectations

$$\begin{aligned}
\mathbb{E}\left[\left|(\varphi, p) - (\varphi, p^N_{\text{SNIS}})\right|^2\right] &\leq 2\mathbb{E}\left[\frac{1}{(W,q)^2}\left(\left|(\varphi W, q) - (\varphi W, q^N)\right|^2 + \left|(W, q^N) - (W,q)\right|^2 \right)\right] \\
&= \frac{2}{(W,q)^2} \left(\text{var}_q[\varphi W, q] + \text{var}_q[W,q]\right), \\
&\leq \frac{2}{(W,q)^2} \left(\mathbb{E}_q[\varphi^2(X) W^2(X)] + \mathbb{E}_q[W^2(X)]\right),
\end{aligned}$$

by using $\sup_x |\varphi(x)| \leq 1$, we conclude the proof. □

4.4 BAYESIAN INFERENCE WITH IMPORTANCE SAMPLING

In this section, we demonstrate the applications of the Monte Carlo and importance sampling estimators to Bayesian inference. Self normalised IS is a natural choice for Bayesian inference. Assume that we have a prior $p(x)$ and a likelihood $p(y|x)$. The posterior is given by

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)\mathrm{d}x}.$$

Let $\bar{p}(x|y) := p(y|x)p(x)$ as usual and design an importance sampler to estimate expectations of the form

$$\mathbb{E}_{p(x|y)}[\varphi(x)] = \int \varphi(x)p(x|y)\mathrm{d}x.$$

Assume that we choose $q(x)$ and decided to perform SNIS. We first sample $X_1, \ldots, X_N \sim q(x)$ and construct

$$\mathsf{W}_i = \frac{\bar{p}(X_i|y)}{q(X_i)} = \frac{p(y|X_i)p(X_i)}{q(X_i)}.$$

We can now normalise these weights and obtain

$$\bar{\mathsf{w}}_i = \frac{\mathsf{W}_i}{\sum_{i=1}^N \mathsf{W}_i},$$

which will give us the Monte Carlo estimator:

$$\mathbb{E}_{p(x|y)}[\varphi(x)] \approx \sum_{i=1}^N \bar{\mathsf{w}}_i \varphi(X_i).$$

It is useful to recall that this estimator is biased, since it is a SNIS estimator. However, as a byproduct of this estimator, we can also obtain an **unbiased** estimate of the marginal likelihood $p(y)$. Note that, this is already provided by the SNIS estimator

$$p(y) \approx \frac{1}{N} \sum_{i=1}^{N} \mathsf{W}_i.$$

Proposition 4.6. *The marginal likelihood estimator given by*

$$p^N(y) = \frac{1}{N} \sum_{i=1}^{N} \mathsf{W}_i,$$

is an unbiased estimator of the marginal likelihood $p(y)$*.*

Proof. We can easily show this

$$\begin{aligned}
\mathbb{E}_q\left[\sum_{i=1}^{N} \mathsf{W}_i\right] &= \sum_{i=1}^{N} \mathbb{E}_q\left[\frac{p(y|X_i)p(X_i)}{q(X_i)}\right] \\
&= N\mathbb{E}_q\left[\frac{p(y|X_i)p(X)}{q(X)}\right] \\
&= N \int \frac{p(y|x)p(x)}{q(x)} q(x)\mathrm{d}x \\
&= Np(y).
\end{aligned}$$

□

As we have seen before, a number of interesting problems require computing normalising constants, including model selection and prediction. SNIS estimators are very useful in the sense that they provide an unbiased estimate of it. Let us now look at an example using the optimal importance proposal as introduced in Remark 4.6.

Example 4.8 (Marginal likelihood using optimal importance proposal)**.** We have seen that we can get unbiased estimates of the marginal likelihood. Find the optimal proposal $q_\star$ for estimating the marginal likelihood.

Solution. We will now see how to find the optimal importance proposal to compute the marginal likelihood. Note that, we have

$$p(y) = \int p(y|x)p(x)\mathrm{d}x,$$

for some prior $p(x)$ and likelihood $p(y|x)$. Note, as we mentioned before, in this case $p(x)$ can be seen as the distribution to sample from and $\varphi(x) = p(y|x)$ to obtain the standard problem of integration $\int \varphi(x)p(x)\mathrm{d}x$. A naive way to approximate this quantity (as we have seen before) is to sample i.i.d from $p(x)$ and approximate the integral, i.e., $X_1, \ldots, X_N \sim p(x)$ and write

$$p_{\text{MC}}^N(y) = \frac{1}{N}\sum_{i=1}^N \varphi(X_i) = \frac{1}{N}\sum_{i=1}^N p(y|X_i).$$

We can now look at the variance of this estimator

$$\mathsf{var}_p\left[p_{\text{MC}}^N(y)\right] = \frac{1}{N}\mathsf{var}_{p(x)}[p(y|x)].$$

This quantity may depend on the prior-likelihood selection and can be large.

Let us take Remark 4.6 seriously as the question suggested and search for the optimal proposal $q_\star$. From (4.16), we can see that

$$q_\star(x) = p(x)\frac{|\varphi(x)|}{\mathbb{E}_p(x)[|\varphi(x)|]}.$$

In this case, however, we have $\varphi(x) = p(y|x)$ (and $|\varphi(x)| = \varphi(x)$ since the likelihood is positive everywhere). We can now write

$$q_\star(x) = p(x)\frac{p(y|x)}{\mathbb{E}_p[p(y|x)]} = p(x)\frac{p(y|x)}{p(y)}.$$

In other words, the optimal proposal is the posterior itself! Now we can compute the IS estimator variance where we plug $q_\star = p(x|y)$. Note to explore variance, we write

$$\mathsf{var}_{q_\star}[p_{\text{IS}}^N(y)] = \frac{1}{N}\left(\mathbb{E}_{q_\star}\left[\left(\frac{p(x)}{q_\star(x)}\right)^2 p(y|x)^2\right] - p(y)^2\right).$$

We compute the first term in brackets,

$$\begin{aligned}\mathbb{E}_{q_\star}\left[\left(\frac{p(x)}{q_\star(x)}\right)^2 p(y|x)^2\right] &= \int \frac{p^2(x)}{q_\star^2(x)}p(y|x)^2 q_\star(x)\mathrm{d}x \\ &= \int \frac{p^2(x)}{q_\star(x)}p(y|x)^2\mathrm{d}x \\ &= \int \frac{p^2(x)}{p(x|y)}p(y|x)^2\mathrm{d}x \\ &= \int \frac{p^2(x)p(y)}{p(y|x)p(x)}p(y|x)^2\mathrm{d}x \\ &= p(y)\int p(x)p(y|x)\mathrm{d}x \\ &= p(y)^2.\end{aligned}$$

Plugging this back into the above variance expression $\mathsf{var}_{q_\star}[p_{\text{IS}}^N(y)]$, we obtain

$$\mathsf{var}_{q_\star}[p_{\text{IS}}^N(y)] = \frac{1}{N}\left(p(y)^2 - p(y)^2\right) = 0.$$

It can be seen that we can achieve zero variance, but as we mentioned before, this required us to know the posterior density.

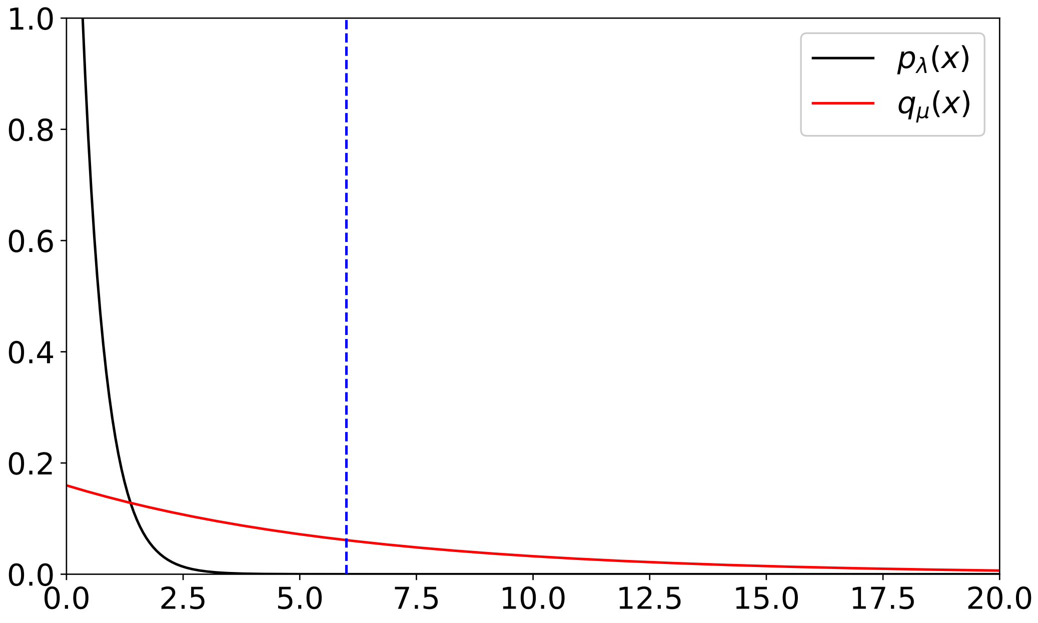


Figure 4.10: The density of the exponential distribution p_λ, the proposal q_μ and $K = 6$

4.5 CHOOSING THE OPTIMAL IMPORTANCE SAMPLING PROPOSAL WITHIN A FAMILY

While we may not be able to choose the optimal proposal $q_\star$ as shown above (as it requires us to know the target density), we can still optimise the proposal within a family of distributions. Recall the discussion for this in the case of rejection sampling, where we optimised the acceptance rate $\hat{a}$ in order to choose the optimal parameter of a proposal. In the case of importance sampling, the target is to *minimise* the variance. Let q_θ be a parametric family of proposals. Recall that in Prop. 4.4, we have defined the variance of the IS estimator as

$$\mathsf{var}_{q_\theta}[\hat{\varphi}^N_{\text{IS}}] = \frac{1}{N}\left(\mathbb{E}_{q_\theta}\left[w_\theta^2(X)\varphi^2(X)\right] - \bar{\varphi}^2\right),$$

where $\bar{\varphi} = \mathbb{E}_p[\varphi(X)]$ and

$$w_\theta(x) = \frac{p(x)}{q_\theta(x)}.$$

Note that $1/N$ does not change the location of the minimum and $\bar{\varphi}^2$ is independent of θ. Therefore, we drop these terms and see that in order to minimise this variance w.r.t. θ, we need to solve the following optimisation problem

$$\theta_\star = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{q_\theta}\left[w_\theta^2(X)\varphi^2(X)\right].$$

We will now demonstrate this with an example.

Example 4.9 (Minimum variance IS). We are given an exponential distribution

$$p_\lambda(x) = \lambda \exp(-\lambda x),$$

and want to compute $\mathbb{P}(X > K)$. For example, $\lambda = 2$ and $K = 6$, we can analytically compute $\mathbb{P}(X > 6) = 6.144 \times 10^{-6}$. Therefore, the standard MC estimator would be very inefficient to compute this probability. In order to mitigate the problem, we would like to use another exponential proposal $q_\mu(x)$ which may have higher probability concentration around 6. We would like to design our proposal using the minimum variance criterion (see Remark 4.6). Find the optimal $\mu_\star$ that would make the estimator variance minimal.

Solution. Accordingly, we would like to find μ such that

$$\mu_\star = \operatorname*{argmin}_\mu \mathbb{E}_q \left[w^2(X)\varphi^2(X) \right].$$

In this case, note that we have $\varphi(x) = \mathbf{1}_{\{x>K\}}(x)$. In order to do this, we write next

$$\begin{aligned}
\mathbb{E}_{q_\mu}\left[\left(w^2(X)\varphi^2(X)\right)\right] &= \int \frac{p_\lambda(x)^2}{q_\mu(x)^2} q_\mu(x)\varphi^2(x)\mathrm{d}x, \\
&= \int_K^\infty \frac{p_\lambda(x)}{q_\mu(x)} p_\lambda(x)\mathrm{d}x, \\
&= \int_K^\infty \frac{\lambda^2 e^{-2\lambda x}}{\mu e^{-\mu x}}\mathrm{d}x, \\
&= \frac{\lambda^2}{\mu}\int_K^\infty e^{-(2\lambda-\mu)x}\mathrm{d}x.
\end{aligned}$$

Note at this stage that in order for this integral to be finite, we need to have $2\lambda - \mu > 0$. Therefore, we limit for $\mu \in (0, 2\lambda)$. In order to compute this, we can multiply and divide by $(2\lambda - \mu)$ and obtain

$$\mathbb{E}_{q_\mu}\left[\left(w^2(X)\varphi^2(X)\right)\right] = \frac{\lambda^2}{\mu(2\lambda-\mu)}\int_K^\infty (2\lambda-\mu)e^{-(2\lambda-\mu)x}\mathrm{d}x,$$

and using the CDF of the exponential distribution, we obtain

$$g(\mu) = \mathbb{E}_{q_\mu}\left[\left(w^2(X)\varphi^2(X)\right)\right] = \frac{\lambda^2}{\mu(2\lambda-\mu)}\left[1 - 1 + e^{-(2\lambda-\mu)K}\right]. \tag{4.19}$$

Now we optimise $g(\mu)$ w.r.t. μ. As usual, we compute first log (and drop the terms unrelated to μ as they will not matter in optimisation)

$$\log g(\mu) \stackrel{c}{=} -\log\mu - \log(2\lambda-\mu) + \mu K.$$

Computing

$$\frac{\mathrm{d}}{\mathrm{d}\mu}\log g(\mu) = -\frac{1}{\mu} + \frac{1}{2\lambda-\mu} + K,$$

Setting this to zero, we obtain

$$\begin{aligned}
&-\frac{1}{\mu} + \frac{1}{2\lambda - \mu} + K = 0,\\
\Rightarrow &-(2\lambda - \mu) + \mu + K\mu(2\lambda - \mu) = 0,\\
\Rightarrow &K\mu^2 - 2K\mu\lambda + 2\lambda - 2\mu = 0,\\
\Rightarrow &K\mu^2 - 2(K\lambda + 1)\mu + 2\lambda = 0.
\end{aligned}$$

This is a quadratic equation, therefore we will have two solutions:

$$\begin{aligned}
\mu &= \frac{2(K\lambda + 1) \pm \sqrt{(2K\lambda + 2)^2 - 8K\lambda}}{2K},\\
&= \frac{2(K\lambda + 1) \pm \sqrt{4K^2\lambda^2 + 4}}{2K}.
\end{aligned}$$

If we inspect this solution, if we choose μ to be the sum of the two terms, we will then have $\mu > 2\lambda$ which is a violation of a condition condition we imposed for the integral to be finite. Therefore, we arrive at

$$\mu_\star = \frac{2(K\lambda + 1) - \sqrt{4K^2\lambda^2 + 4}}{2K}.$$

After this tedious computation, we can now verify the reduction in variance and estimation quality. Let us now set $K = 6$ and $\lambda = 2$. See Fig. 4.10 for plot of p_λ, $K = 6$ and $q_{\mu_\star}$ (the optimal exponential proposal). We can see that the proposal puts much higher mass to the right of K. A standard run for $N = 10^5$ samples gives us **zero** samples in the region of $X > 6$, therefore the standard MC estimate is zero! Compared to $\hat{\varphi}^N_{\text{MC}} = 0$, using $q_{\mu_\star}$ as a proposal, we obtain $\hat{\varphi}^N_{\text{IS}} = 6.08 \times 10^{-6}$ which is a much better estimate.

Let us compare the theoretical variances of two estimators. The standard variance of $\hat{\varphi}^N_{\text{MC}}$ is

$$\mathsf{var}_p(\hat{\varphi}^N_{\text{MC}}) = \frac{1}{N}\mathsf{var}_p(\varphi(X)),$$

where

$$\begin{aligned}
\mathsf{var}_p(\varphi(X)) &= \int \varphi(x)^2 p_\lambda(x)\mathrm{d}x - \left(\int \varphi(x)p_\lambda(x)\mathrm{d}x\right)^2,\\
&= \int_K^\infty p_\lambda(x)\mathrm{d}x - \left(\int_K^\infty p_\lambda(x)\mathrm{d}x\right)^2.
\end{aligned}$$

Using CDFs, we can compute this quantity hence can obtain the estimate of the variance for $\hat{\varphi}^N_{\text{MC}}$.

Now set $\mu = \mu_\star$. The variance of $\hat{\varphi}^N_{\text{IS}}$ is given by (see Prop. 4.4)

$$\mathsf{var}_q[\hat{\varphi}^N_{\text{IS}}] = \frac{1}{N}\left(\mathbb{E}_q[w^2(X)\varphi^2(X)] - \bar{\varphi}^2\right),$$

We have already computed the term $\mathbb{E}_q[w^2(X)\varphi^2(X)]$ in Eq. (4.19). The second term is the true integral, which we also summarised how to compute above, i.e., $\bar{\varphi} = \int_K^\infty p_\lambda(x)\mathrm{d}x$ which can be computed using the exponential CDF. In this particular case, we compute

$$\mathsf{var}_q[\hat{\varphi}^N_{\text{IS}}] = \frac{1}{N}\left(g(\mu_\star) - \bar{\varphi}^2\right).$$

The theoretical variance of the naive MC estimator is given by 6.14×10^{-7} for $N = 10$ samples vs. the IS estimator variance is 6.04×10^{-11} for the same amount of samples. This is a huge improvement in the variance of the estimation.

4.6 IMPLEMENTATION, ALGORITHMS, DIAGNOSTICS

When implementing the IS or SNIS, there are several numerical considerations that need to be taken into account. Especially for SNIS, where the weight normalisation takes place, several numerical problems can arise for complex distributions that would prevent us from implementing them successfully.

4.6.1 COMPUTING WEIGHTS

In the case of SNIS, we have stated that the weights are computed as

$$\bar{\mathsf{w}}_i = \frac{W(X_i)}{\sum_{i=1}^N W(X_i)},$$

where $W(x) = \frac{\bar{p}(x)}{q(x)}$. However, this formula can be numerically ill-behaved for complex distributions. To mitigate this, the weighting step is implemented as follows. We first compute log *unnormalised* weights:

$$\log \mathsf{W}_i = \log \bar{p}(X_i) - \log q(X_i).$$

However, directly applying exponentiation and normalisation can also lead to numerical problems. Note that, we will normalise these weights (e.g. multiplying them with a constant does not change the result), we can use this to our advantage. A common numerical trick is to subtract the maximum log weight from all weights:

$$\log \tilde{\mathsf{W}}_i = \log \bar{p}(X_i) - \log q(X_i) - \max_{i=1,\ldots,N} \log \mathsf{W}_i.$$

This ensures that the maximum weight is 0 and all other weights are negative. We can now exponentiate the weights and normalise them:

$$\bar{\mathsf{w}}_i = \frac{\exp(\log \tilde{\mathsf{W}}_i)}{\sum_{i=1}^N \exp(\log \tilde{\mathsf{W}}_i)}.$$

Note that, this does not change the computation, just done for numerical stability.

4.6.2 SAMPLING IMPORTANCE RESAMPLING

We can also use the SNIS estimator as a sampler (Robert and Casella, 2004). Recall that, the SNIS estimator provides us an estimator of the distribution $p(x)$ as

$$p(x)\mathrm{d}x \approx \tilde{p}^N(\mathrm{d}x) = \sum_{i=1}^N \bar{\mathsf{w}}_i \delta_{X_i}(\mathrm{d}x).$$

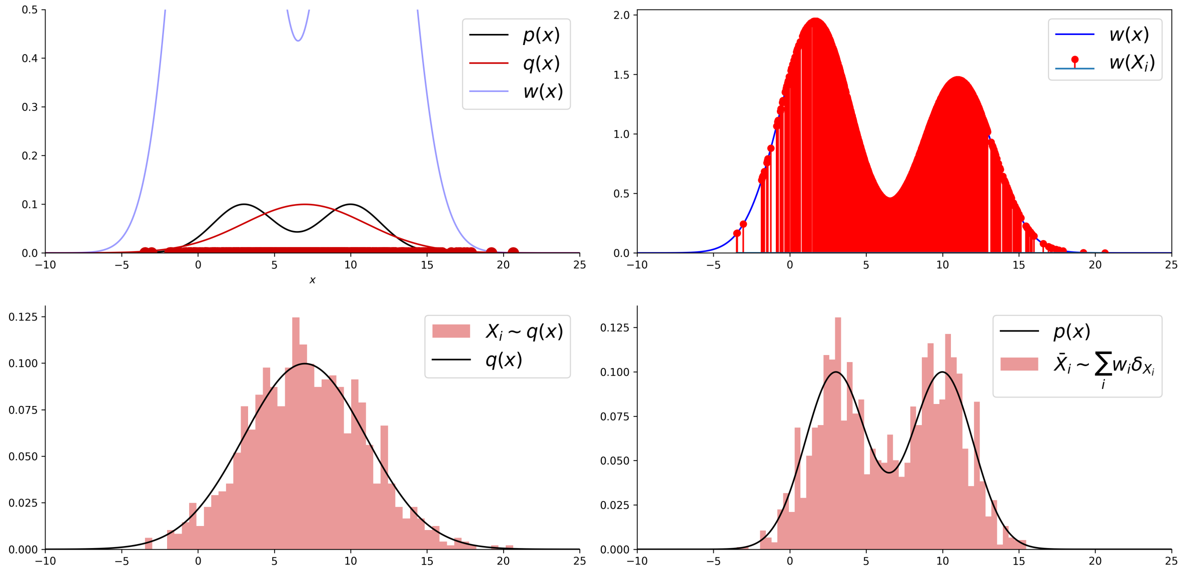


Figure 4.11: Top left shows the target density, the proposal, and the weight function. Top right shows the samples with their respective weights. Bottom left shows that these samples are indeed approximately distribution w.r.t. $q(x)$ (just with attached weights). Bottom right shows that we can resample these existing samples to obtain a new set of samples $\bar{X}_i$ that are distributed (approximately)according to $p(x)$.

This p^N can be seen as a weighted distribution. By drawing samples from this distribution, we may also approximately sample from $p(x)$ (recall that IS based ideas here are just introduced for integration so far). We can then draw[4]

$$k \sim \mathsf{Discrete}(\bar{\mathsf{w}}_1, \ldots, \bar{\mathsf{w}}_N),$$

and set new samples

$$\bar{X}_i = X_k.$$

This amounts to *resampling* the existing samples w.r.t. their weights. A demonstration of this idea can be seen from Figure 4.11.

4.6.3 DIAGNOSTICS FOR IMPORTANCE SAMPLING

It is important to have a good intuition and diagnostic tools to understand the performance of the IS and SNIS estimators. We first start with the effective sample size (ESS).

Definition 4.1 (Effective Sample Size). *To measure the sample efficiency, one measure that is used in the literature is the effective sample size (ESS) which is given by*

$$\mathsf{ESS}_N = \frac{1}{\sum_{i=1}^N \bar{\mathsf{w}}_i^2},$$

for the SNIS estimator.

[4]Note that here the weights $\bar{w}_i$ are normalised. Even in the basic IS case, we need to normalise weights (**just for resampling**) as they do not naturally sum up to one.

In order to see the meaning of the ESS, consider the case where $\bar{\mathsf{w}}_i = 1/N$ where we have an equally weighted sample. This means all samples are equally considered and in this case we have $\mathsf{ESS}_N = N$. On the other hand, if we have a sample X_i where $\bar{\mathsf{w}}_i = 1$ and, hence, $\bar{\mathsf{w}}_j = 0$ for every $j \neq i$, we obtain $\mathsf{ESS}_N = 1$. This means, we *effectively* have one sample which is the goal of the estimator. ESS is used to measure importance samplers and importance sampling-based estimators in the literature (Elvira et al., 2018). Note that the ESS_N takes values between 1 and N, i.e., $1 \leq \mathsf{ESS}_N \leq N$.

4.6.4 MIXTURE IMPORTANCE SAMPLING

Sometimes the target density $p(x)$ can be multimodal, therefore it is beneficial to use mixture densities as proposals (Owen, 2013). We have seen in previous chapters how to sample from a mixture. Let us define a proposal

$$q_\alpha(x) = \sum_{k=1}^{K} \alpha_k q_k(x),$$

where $\alpha_k \geq 0$ and $\sum_{k=1}^{K} \alpha_k = 1$. In this version of the method, we just sample from the mixture proposal $X_i \sim q_\alpha(x)$ and then, given an unnormalised target $\bar{p}$, compute the importance weights as

$$\bar{\mathsf{w}}_i = \frac{W(X_i)}{\sum_{i=1}^{N} W(X_i)},$$

where

$$W(X_i) = \frac{\bar{p}(X_i)}{\sum_{k=1}^{K} \alpha_k q_k(X_i)}.$$

The computational concerns may arise in this situation too, as the denominator as a sum of densities and its log can be tricky to compute. In these cases, we can use the log-sum-exp trick to compute the log of the denominator.

5

MARKOV CHAIN MONTE CARLO

In this chapter, we introduce Markov chains and then Markov Chain Monte Carlo (MCMC) methods. These methods are at the heart of Bayesian statistics, generative modelling, statistical physics, and many other fields. We will introduce the Metropolis-Hastings algorithm and then introduce the celebrated Gibbs sampler and, if time permits, some others.

In this chapter, we introduce a new sampling methodology - namely using Markov chains for sampling problems. This is a very powerful and widely used idea in statistics and machine learning. The idea is to set up Markov chains with prescribed stationary distributions. These distributions will be our target distributions.

In this chapter, we will adapt our notation and modify it to suit the new setting. From now on, we denote stationary/invariant distributions of Markov chains (which are also coincide with our target distributions) as $p_\star$. We will introduce discrete space Markov chains next.

5.1 DISCRETE STATE SPACE MARKOV CHAINS

A good setting for an introduction to Markov chains is the discrete space setting. In this setting, we have a finite set of states X where the cardinality of X is finite. We first define the Markov chain in this context.

Definition 5.1 (Markov chain). *A discrete Markov chain is a sequence of random variables $X_0, X_1, \ldots, X_n$ such that*

$$\mathbb{P}(X_{n+1} = x_{n+1} \mid X_0 = x_0, \ldots, X_n = x_n) = \mathbb{P}(X_{n+1} = x_{n+1} \mid X_n = x_n).$$

In other words, a Markov chain is a sequence of random variables such that a given state at time n is conditionally independent of all previous states given the state at time $n-1$. One can see that this describes many systems in the real world – as evolution of many systems can be summarised with the current state of the system and the evolution law.

An important quantity in the study of Markov chains is the transition matrix (or kernel in the continuous space case). This matrix defines the evolution structure of the chain and determines all of its properties. The transition matrix is defined as follows.

Definition 5.2 (Transition matrix)**.** *The transition matrix of a Markov chain is a matrix M such that*

$$M_{ij} = \mathbb{P}(X_{n+1} = j \mid X_n = i).$$

A usual way to depict Markov chains is the following conditional independence structure which sums the structure of the Markov chain up. We note that we will only consider the case

$$\begin{tikzcd} X_0 \arrow[r] & X_1 \arrow[r] & X_2 \arrow[r] & \cdots \arrow[r] & X_n \end{tikzcd}$$

Figure 5.1: A Markov chain with n states.

where the transition matrix is time-homoegeneous, i.e., the transition matrix is the same for all times. We can see then that a Markov chain's behaviour is completely determined by its initial distribution and the transition matrix. We will denote the initial distribution of the chain as p_0 and note that this is a discrete distribution over the state space X (in this case)[1]. The transition matrix M is a matrix of size $d \times d$ where $d = |\mathsf{X}|$.

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1d} \\ M_{21} & M_{22} & \cdots & M_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ M_{d1} & M_{d2} & \cdots & M_{dd} \end{bmatrix}.$$

We note that this matrix is stochastic, i.e. each row sums to 1:

$$\sum_{j=1}^{d} M_{ij} = 1,$$

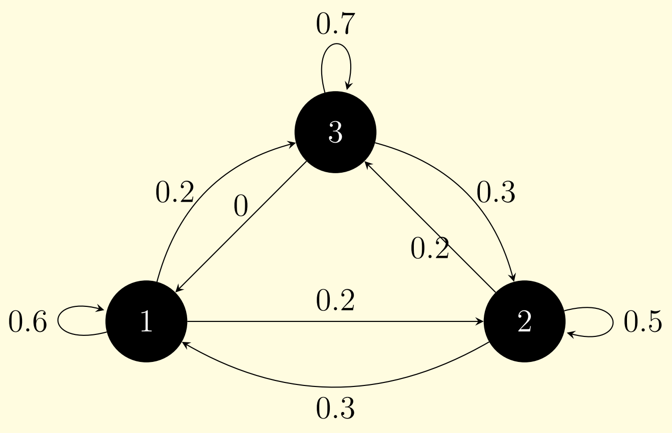
since $M_{ij} = \mathbb{P}(X_{n+1} = j | X_n = i)$ and

$$\sum_{j=1}^{d} \mathbb{P}(X_{n+1} = j | X_n = i) = 1.$$

Next we will consider an example.

[1]When we move to continuous spaces, we will use the same notation for densities.

Example 5.1 (Discrete space Markov chain)**.** Consider the state transition diagram



of a Markov chain. Write out the transition matrix of this chain and decsribe the simulation procedure.

Solution. We can write the transition matrix as

$$M = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.3 & 0.5 & 0.2 \\ 0 & 0.3 & 0.7 \end{bmatrix}, \qquad \text{where } \mathsf{X} = \{1, 2, 3\}.$$

The state transition diagram of this matrix can be described as follows.

This Markov chain can be simulated using the following idea. Given the above diagram, we can denote its transition matrix as a table:

M	$X_t = 1$	$X_t = 2$	$X_t = 3$
$X_{t-1} = 1$	0.6	0.2	0.2
$X_{t-1} = 2$	0.3	0.5	0.2
$X_{t-1} = 3$	0	0.3	0.7

Given $X_0 = 1$, how to simulate this chain? This boils down to just selecting the correct row from this matrix and then sampling using the discrete distribution given by that row. For example, if we sample from the first row, we get $X_1 = 1$ with probability 0.6, $X_1 = 2$ with probability 0.2 and $X_1 = 3$ with probability 0.2. We can then repeat this process for X_2 and so on. This is a simple way to simulate a Markov chain. The precise sampler is given below.

$$X_t | X_t = x_{t-1} \sim \text{Discrete}(M_{x_{t-1},\cdot}),$$

where the notation $M_{x_{t-1},\cdot}$ denotes the x_{t-1}th row of the transition matrix M (where $x_{t-1} \in \{1, 2, 3\}$ naturally).

Markov chains can be seen everywhere in the real world, hence we will skip their applications. But one exciting example can be provided from the field of language modelling. Let us describe a character-level Markov model of English.

Example 5.2 (Character-level Markov model of English)**.** Consider the task of modelling an English language text as a Markov model. Describe a possible way of doing this on a character-level.

Solution. We can model the English language as a Markov chain on a character-level. In this case, we can define the state space as $\mathsf{X} = \{\mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}, .\}$ (where for simplicity the character dot models the space character and no other special characters are considered). We can then define the transition matrix as follows. Let x_{t-1} be the previous character and x_t be the current character. Of course, writing out the transition matrix by hand is an impossible task, hence we can *learn* it from data. For this, we simply choose a large corpus of English text and count the number of times a character is followed by another character. We have provided an example code for this in our online companion. By estimating the transition matrix from data, we can then simulate English text by sampling from this Markov chain. For example, one simulated text reads as

```
g.l.se.t.bin.s.lese.ry..wolked.t.hered.e.ly.hr.impefatrt.mofe.mouroreand
```

and goes on (see lecture for a longer simulation). We can see that while this captures very vague structure, it is not a good model of English. However, this character-level model can still be useful for some applications. Because we can use this model to estimate the probability of a given text. For example, we can compute

$$\mathbb{P}(\text{the quick brown fox jumps over the lazy dog})$$

by breaking down this text into characters and 'reading' the transition matrix. This is a very simple example of a Markov chain model of English.

Let us look at a more intelligent way of using Markov chains to model language.

Example 5.3 (Word-level Markov model of English)**.** Consider the task of modelling an English language text as a Markov model. Describe a possible way of doing this on a word-level.

Solution. We can model the English language as a Markov chain on a word-level. As you can see, there is no single state-space for this (as we can expand our word list (state-space) possibly indefinitely). A good idea is to pick a book or a large text corpus and count all possible word transitions. Note that modern language models do *not* use words, but rather sub-word units, called tokens. Here we will stick to words and we will pick the book 'A Study in Scarlet' by Sir Arthur Conan Doyle. We have provided an example code for this in our online companion. By estimating the transition matrix from data, we can then simulate English text by sampling from this Markov chain. For example, one simulated text reads as

```
    Sherlock Holmes, "because you will place where you remember, in his
cudgel, only one in summer are grey with a vague shadowy terrors which
hung over my hand, and eyeing the strange contrast to the name a walking
in time before twelve, you are."
```

While this is much more intelligible and seems to follow some English rules, it is nothing but a simulation from a Markov chain given a transition matrix as estimated in our online companion. Modern language models, however, do not use Markov models either – their structure is much more complex and they are based on neural networks.

We can also compute n-step transition matrix:

$$M^{(n)} = \mathbb{P}(X_n = j | X_0 = i),$$

where $M^{(n)}$ is a matrix of size $d \times d$. For this, see that n-step transition matrix can be written as:

$$\begin{aligned} M_{ij}^{(n)} &= \mathbb{P}(X_n = j | X_0 = i) \\ &= \sum_k \mathbb{P}(X_n = j, X_1 = k | X_0 = i) \\ &= \sum_k \mathbb{P}(X_n = j | X_1 = k, X_0 = i)\mathbb{P}(X_1 = k | X_0 = i) \\ &= \sum_k \mathbb{P}(X_n = j | X_1 = k)\mathbb{P}(X_1 = k | X_0 = i) \\ &= \sum_k M_{ik} M_{kj}^{(n-1)}. \end{aligned}$$

Therefore, $M^{(n)} = M^n$ which is the nth power of the transition matrix. Note that we can compute in general the conditional distributions of the Markov chain by summing out the variables in the middle. For example, in order to compute $\mathbb{P}(X_{n+2} = x_{n+2} | X_n = x_n)$, we can write

$$\mathbb{P}(X_{n+2} = x_{n+2} | X_n = x_n) = \sum_{x_{n+1}} \mathbb{P}(X_{n+2} = x_{n+2} | X_{n+1} = x_{n+1})\mathbb{P}(X_{n+1} = x_{n+1} | X_n = x_n).$$

This will lead us to define the Chapman-Kolmogorov equation, which is a generalization of the n-step transition matrix:

$$\begin{aligned} M^{(m+n)} &= \mathbb{P}(X_{m+n} = j | X_0 = i) \\ &= \sum_k \mathbb{P}(X_{m+n} = j | X_n = k)\mathbb{P}(X_n = k | X_0 = i) \\ &= \sum_k M_{ik}^{(m)} M_{kj}^{(n)}. \end{aligned}$$

Therefore, we can write $M^{m+n} = M^m M^n$.

It is also important to define the evolution of the chain. Note that we defined our initial distribution as p_0 and it is important to quantify how this distribution evolves over time. We denote the distribution at time n as p_n and write Then, the density of the chain at time n is given by:

$$\begin{aligned} p_n(i) &= \mathbb{P}(X_n = i) \\ &= \sum_k \mathbb{P}(X_n = i, X_{n-1} = k) \\ &= \sum_k \mathbb{P}(X_n = i | X_{n-1} = k)\mathbb{P}(X_{n-1} = k) \\ &= \sum_k M_{ki} p_{n-1}(k). \end{aligned}$$

This implies that

$$p_n = p_{n-1} M.$$

Therefore,

$$p_n = p_0 M^n.$$

These are important equations, which will have corresponding equations in the continuous case (however, they will be integrals).

Since we have expressed our interest in Markov chains because of their potential utility in sampling, we will now discuss the properties we need to ensure that we can use Markov chains for sampling. In short, we need Markov chains that have (i) invariant distributions, (ii) their convergence to invariant distributions are ensured, (iii) the invariant distribution is unique. We will now discuss the properties we need to ensure these in detail.

5.1.1 IRREDUCIBILITY

The first property we need to ensure is that the Markov chain is irreducible. This means that there is a path from any state to any other state . To be precise, let $x, x' \in \mathsf{X}$ be any two states. We write $x \rightsquigarrow x'$ if there is a path from x to x':

$$\exists n > 0, \text{ s.t. }, \mathbb{P}(X_n = x' | X_0 = x) > 0.$$

If $x \rightsquigarrow x'$ and $x' \rightsquigarrow x$, then we say that x and x' *communicate*. We then define the *communication class* $C \subset \mathsf{X}$ which is a set of states such that $x \in C$ and $x' \in C$ if and only if $x \rightsquigarrow x'$ and $x' \rightsquigarrow x$. A chain is irreducible if X is a single communication class. This simply means that there is a positive probability of moving around to every other state. This makes sense as without ensuring this, we won't be sampling from the full support.

5.1.2 RECURRENCE AND TRANSIENCE

We now define the notion of recurrence and transience. A state $i \in \mathsf{X}$ is *recurrent* if there is a positive probability of returning to i. In order to see define this, consider the return time

$$\tau_i = \inf\{n \geq 1 : X_n = i\}.$$

We say that the state i is recurrent if

$$\mathbb{P}(\tau_i < \infty | X_0 = i) = 1.$$

In other words, the probability of waiting time being finite is 1. If a chain is not recurrent, it is said to be transient. We can also further define the *positive recurrence* which is a slightly stronger (better) condition. We say that i is positively recurrent if

$$\mathbb{E}[\tau_i | X_0 = i] < \infty.$$

This means that the expected waiting time is finite. If a chain is recurrent but not positive recurrent, then it is called null recurrent.

5.1.3 INVARIANT DISTRIBUTIONS

In the discrete time case, a distribution $p_\star$ is called invariant if

$$p_\star = p_\star M.$$

This means that the chain is reach stationarity, i.e., evolving further (via M) does not change the distribution. We have then the following theorem (Yıldırım, 2017).

Theorem 5.1. *If M is irreducible, then M has a unique invariant distribution if and only if it is positive recurrent.*

This is encouraging however for actual convergence of the chain to this distribution, we will need more conditions.

5.1.4 REVERSIBILITY AND DETAILED BALANCE

We define the detailed balance condition as

$$p_\star(i) M_{ij} = p_\star(j) M_{ji}.$$

This trivially implies that $p_\star = p_\star M$, hence the invariance of $p_\star$. We will have a more detailed discussion of this condition in the continuous state space case.

5.1.5 CONVERGENCE TO INVARIANT DISTRIBUTION

Finally, we need the ergodicity condition to ensure that the chain converges to the invariant distribution. For this, we require the chain to be aperiodic, which is defined as follows. A state i is called aperiodic if

$$\{n > 0 : \mathbb{P}(X_{n+1} = i | X_1 = i) > 0\}$$

has no common divisor other than 1. A Markov chain is called aperiodic if all states are aperiodic. An irreducible Markov chain is called ergodic if it is positive recurrent and aperiodic. If $(X_n)_{n\in\mathbb{N}}$ is an ergodic Markov chain with initial p_0 and $p_\star$ as its invariant distribution, then

$$\lim_{n\to\infty} p_n(i) = p_\star(i).$$

Moreover, for $i, j \in \mathsf{X}$

$$\lim_{n\to\infty} \mathbb{P}(X_n = i | X_1 = j) = p_\star(i).$$

In other words, the chain will converge to its invariant distribution from every state.

5.2 CONTINUOUS STATE SPACE MARKOV CHAINS

Our main interest is in the continuous case. However, it is important to understand the definitions above – as we will not go into analogous definitions in the continuous case. The reason for this is that, in continuous cases, the individual states have zero probability (i.e. a point has zero probability) and all the notions above are defined using sets and measure theoretic concepts. We focus on simulation methods within this course, therefore, we will not go into reviewing this material. A couple of very good books for this are Douc et al. (2018) and Douc et al. (2013).

Let X be an uncountable set from now on, e.g., $\mathsf{X} = \mathbb{R}$ or $\mathsf{X} = \mathbb{R}^d$. We denote the initial density as $p_0(x)$ as usual, the transition kernel with $K(x|x')$, the marginal density of the chain at time n as $p_n(x)$.

We can write the Markov property in this case as follows. For any measurable A

$$\mathbb{P}(X_n \in A | X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \mathbb{P}(X_n \in A | X_{n-1} = x_{n-1}).$$

This implies that if we write down the joint distribution of $X_{1:n}$, then the following factorisation holds

$$p(x_0, \ldots, x_n) = \prod_{k=0}^{n} p(x_k | x_{k-1}),$$

where $p(x_0|x_{-1}) := p_0(x_0)$. We also assume that the transition kernel has a density which we denote as $K(x_n|x_{n-1})$ at time n. Similarly to the discrete case, we will assume that the density is time-homogeneous (i.e. same for every n). Note that the transition density is a density in its first variable, i.e.,

$$\int_{\mathsf{X}} K(x_n|x_{n-1}) dx_n = 1.$$

It is a function of x_{n-1} otherwise. We give an example of a continuous state-space Markov chain in what follows.

Example 5.4 (Simulation of a Markov process)**.** Consider the following Markov chain with $X_0 = 0$

$$X_n | X_{n-1} = x_{n-1} \sim \mathcal{N}(x_n; a x_{n-1}, 1), \tag{5.1}$$

with $0 < a < 1$. Describe the simulation procedure for this chain in terms of a recursion.

Solution. We can simulate this chain by

$$\begin{aligned} X_1 &\sim \mathcal{N}(0, 1) \\ X_2 &\sim \mathcal{N}(aX_1, 1) \\ X_3 &\sim \mathcal{N}(aX_2, 1) \\ &\vdots \\ X_n &\sim \mathcal{N}(aX_{n-1}, 1). \end{aligned}$$

How to do this? We also note that Eq. (5.1) can also be expressed as

$$X_n = aX_{n-1} + \epsilon_n$$

where $\epsilon_n \sim \mathcal{N}(0,1)$. This is also called AR(1) process. From the last equation, it must be clear how to simulate this as you only need a `for` loop and samples from $\mathcal{N}(0,1)$.

Similar to the continuous case, we can define the distribution of X_n given a past variable X_{n-k} by integrating out the variables in between . It is important to note that, X_n is independent of past variables **if** (and a big if) $X_{n-1} = x_{n-1}$ is given. Otherwise, we can write down the densities as

$$p(x_n|x_{n-k}) = \int \cdots \int K(x_n|x_{n-1})K(x_{n-1}|x_{n-2}) \cdots K(x_{n-k+1}|x_{n-k}) dx_{n-1} \cdots dx_{n-k+1}.$$

We define the m-step transition kernel as

$$K^{(m)}(x_{m+n}|x_n) = \int_{\mathsf{X}} K(x_{m+n}|x_{m+n-1}) \cdots K(x_{n+1}|x_n)\, \mathrm{d}x_{m+n-1} \cdots \mathrm{d}x_{n+1}.$$

We now provide the definition of invariance in this context, w.r.t. to the transition kernel.

Definition 5.3 (K-invariance)**.** *A probability measure $p_\star$ is called K-invariant if*

$$p_\star(x) = \int_{\mathsf{X}} K(x|x')\, p_\star(x')\mathrm{d}x'. \tag{5.2}$$

It can be seen that $p_\star$ being invariant means that the kernel operating on $p_\star$ results in the same distribution $p_\star$ (the integral against the kernel can be seen as a transformation, similar to the matrix product in the discrete case). Finally, we get to the detailed balance condition.

Definition 5.4 (Detailed balance)**.** *A transition kernel K is said to satisfy detailed balance if*

$$K(x'|x)p_\star(x) = K(x|x')p_\star(x'). \tag{5.3}$$

Remark 5.1. It is also important to note the more general version of the detailed balance written using the integral form. In general, the transition kernel is a probability measure w.r.t. first argument so the above relationship can be written as

$$\int f(x,x')K(x'|x)p_\star(x)\mathrm{d}x'\mathrm{d}x = \int f(x,x')K(x|x')p_\star(x')\mathrm{d}x'\mathrm{d}x,$$

for f defined on $\mathsf{X} \times \mathsf{X}$.

We note that this is a sufficient condition for stationarity of $p_\star$.

Proposition 5.1 (Detailed balance implies stationarity)**.** *If K satisfies detailed balance, then $p_\star$ is the invariant distribution.*

Proof. The proof is a one-liner:

$$\int p_\star(x) K(x'|x) \mathrm{d}x' = \int p_\star(x') K(x|x') \mathrm{d}x',$$

which is just integrating both sides after writing the detailed balance condition. The lhs of this equation is $p_\star(x)$ since $K(x'|x)$ integrates to 1 which leaves us with the definition of K-invariance as given in (5.2). □

Let us see an example of a continuous space Markov chain (or rather go back to AR(1) example).

Example 5.5. Consider again the Markov chain with the following transition kernel

$$K(x_n|x_{n-1}) = \mathcal{N}(x_n; a x_{n-1}, 1).$$

We can also describe the evolution this chain as a recursion, as mentioned before

$$X_n = a X_{n-1} + \epsilon_n$$

where $\epsilon_n \sim \mathcal{N}(0,1)$. Show that

$$p_\star(x) = \mathcal{N}(x; 0, \frac{1}{1-a^2}),$$

by checking the detailed balance condition (we will prove this directly later). Prove that the m-step transition kernel is given by

$$K^{(m)}(x_{m+n}|x_n) = \mathcal{N}(x_{m+n}; a^m x_n, \frac{1-a^{2m}}{1-a^2}).$$

Solution. The proofs of these results will be asked as exercises (as usual, solutions will be posted). We can note that the last result implies trivially that

$$p_\star(x) = \lim_{m\to\infty} K^{(m)}(x|x').$$

for any x'. In other words, starting from any x', the chain will reach stationarity.

We have now almost everything we need to move on to discuss Metropolis-Hastings method.

5.3 METROPOLIS-HASTINGS ALGORITHM

We finally have all the ingredients to define the celebrated Metropolis-Hastings algorithm. We will not need more technicalities in defining it.

The Metropolis-Hastings (MH) algorithm is a remarkable method which allows us to define transition kernels (defined implicitly via the algorithm) where the detailed balance is satisfied w.r.t. any $p_\star$ we wish to sample from. I call this *remarkable* because it rids us of the need of designing Markov kernels for specific probability distributions and provides a generic way to design samplers that will target any measure we want. The algorithm relies on the idea of using *local* proposals $q(x'|x)$ and accepting them with a certain acceptance ratio. The acceptance ratio is designed so that the resulting samples $X_1, \ldots, X_n$ from the method form a Markov chain that leaves $p_\star$ invariant. We will provide the algorithm below, as seen from Algorithm 9. Note,

Algorithm 9 Pseudocode for Metropolis Hastings method

```
1: Input: The number of samples $N$, and starting point $X_0$.
2: for $n = 1, \ldots, N$ do
3:     Propose (sample): $X' \sim q(x'|X_{n-1})$
4:     Accept the sample $X'$ with probability
           $$\alpha(X_{n-1}, X') = \min\left\{1, \frac{p_\star(X')q(X_{n-1}|X')}{p_\star(X_{n-1})q(X'|X_{n-1})}\right\}.$$
5:     Otherwise reject the sample and set $X_n = X_{n-1}$.
6: end for
7: Discard first burnin samples and return the remaining samples.
```

as mentioned in the lecture, the last step of the method: When a sample is rejected, we do not sample again – we set $X_n = X_{n-1}$ and continue sampling the next sample. This means that, if the rejection rate is high, there will be a lot of duplicated samples and this is the expected behaviour. Another important note is about the `burnin` period. Any Markov chain started at a random point will take some time to reach stationarity (the whole magic is to be able to make them converge faster). Therefore, we discard the first `burnin` samples and only return the remaining ones. This is a common practice in MCMC methods.

We define the acceptance ratio as

$$\mathsf{r}(x, x') = \frac{p_\star(x')q(x|x')}{p_\star(x)q(x'|x)}. \tag{5.4}$$

We also note that in the practical algorithm, one does not need to implement the min operation. For accepting with a certain probability (like in the rejection sampling), we draw $U \sim \text{Unif}(0,1)$ and check if $U \leq \alpha(X_{n-1}, X')$. However, if the ratio $\mathsf{r}(X_{n-1}, X')$ is greater than 1, this sample is always going to be accepted anyway. The min operation is important however for theoretical properties of the kernel to hold.

As we mentioned above, the algorithm provides us with an implicit kernel $K(x_n|x_{n-1})$ – if you think about it, it is just a way to get X_n given X_{n-1}. The specific structure of the algorithm, however, ensures that we leave the right kind of distribution invariant – i.e. $p_\star$ – that is our target measure. We elucidate this in the following proposition.

Proposition 5.2 (Metropolis-Hastings satisfies detailed balance). *The Metropolis-Hastings algorithm satisfies detailed balance w.r.t. $p_\star$, i.e.,*

$$p_\star(x)K(x'|x) = p_\star(x')K(x|x'),$$

where K is the kernel defined by the MH algorithm.

Proof. We first define the kernel induced by the MH algorithm. This can be seen by inspecting the algorithm - and we write it intuitively as (by writing Dirac functions instead of measures)

$$K(x'|x) = \alpha(x,x')q(x'|x) + (1 - a(x))\delta_x(x'),$$

where δ_x is the Dirac delta function and

$$a(x) = \int_{\mathsf{X}} \alpha(x,x')q(x'|x)\mathrm{d}x',$$

is the probability of accepting a sample (hence $1 - a(x)$ is the probability of rejecting a new sample while at point x). See Sec. 2.3.1 of Douc et al. (2018) for a rigorous derivation. Given this, to check detailed balance, we use Remark 5.1 and write for a function f

$$\begin{aligned}\int f(x,x')p_\star(x)K(x'|x)\mathrm{d}x\mathrm{d}x' &= \int f(x,x')p_\star(x)q(x'|x)\alpha(x,x')\mathrm{d}x\mathrm{d}x' \\ &\quad + \int f(x,x')p_\star(x)(1-a(x))\delta_x(\mathrm{d}x')\mathrm{d}x,\end{aligned}$$

and check if the equality in Remark 5.1 holds. For the first term, we can easily see that

$$\begin{aligned}p_\star(x)q(x'|x)\alpha(x,x') &= p_\star(x)q(x'|x)\min\left\{1, \frac{p_\star(x')q(x|x')}{p_\star(x)q(x'|x)}\right\} \\ &= \min\left\{p_\star(x)q(x'|x), p_\star(x')q(x|x')\right\} \\ &= \min\left\{\frac{p_\star(x)q(x'|x)}{p_\star(x')q(x|x')}, 1\right\} p_\star(x')q(x|x'), \\ &= p_\star(x')q(x|x')\alpha(x',x).\end{aligned}$$

Since the second term involves a Dirac measure, we will check it against the integral form directly:

$$\begin{aligned}\int f(x,x')p_\star(x)(1-a(x))\delta_x(\mathrm{d}x')\mathrm{d}x &= \int f(x,x)p_\star(x)(1-a(x))\mathrm{d}x \\ &= \int f(x',x')p_\star(x')(1-a(x'))\mathrm{d}x' \\ &= \int f(x,x')p_\star(x')(1-a(x'))\delta_{x'}(\mathrm{d}x)\mathrm{d}x'.\end{aligned}$$

All in all, we can see that

$$\int f(x,x')p_\star(x)K(x'|x)\mathrm{d}x\mathrm{d}x' = \int f(x,x')p_\star(x')K(x|x')\mathrm{d}x\mathrm{d}x',$$

which is the detailed balance condition. □

One can see that the algorithm works just the same with unnormalised densities, i.e., recall

$$p_\star(x) = \frac{\bar{p}_\star(x)}{Z},$$

where Z is the normalisation constant. In this case, the acceptance ratio becomes

$$\mathsf{r}(x, x') = \frac{\bar{p}_\star(x')q(x|x')}{\bar{p}_\star(x)q(x'|x)},$$

without any change as the normalising constants cancel out in (5.4). We will next describe certain classes of proposals to sample from various kinds of distributions and assess their performance.

5.3.1 INDEPENDENT PROPOSALS

An important class of proposals that is used in practice is the independent proposal. Note that in general we denoted our proposal with $q(x'|x)$, in particular, we would sample from $q(x'|x_{n-1})$ implying that in general the proposal uses the current state of the chain. This does not have to be the case and we can as well chose just an independent proposal $q(x')$ to ease computations. The acceptance ratio in this specific case becomes

$$\mathsf{r}(x, x') = \frac{\bar{p}_\star(x')q(x)}{\bar{p}_\star(x)q(x')}.$$

In the algorithm, this means that we compute

$$\alpha(X_{n-1}, X') = \min\left\{1, \frac{\bar{p}_\star(X')q(X_{n-1})}{\bar{p}_\star(X_{n-1})q(X')}\right\}.$$

We will see one example as follows.

Example 5.6 (Independent Gaussian proposal)**.** Consider a Gaussian (artificial) target:

$$p_\star(x) = \mathcal{N}(x; \mu, \sigma^2)$$

Assume we want to use MH to sample from it. Choose a proposal

$$q(x) = \mathcal{N}(x; \mu_q, \sigma_q^2).$$

Compute the acceptance ratio.

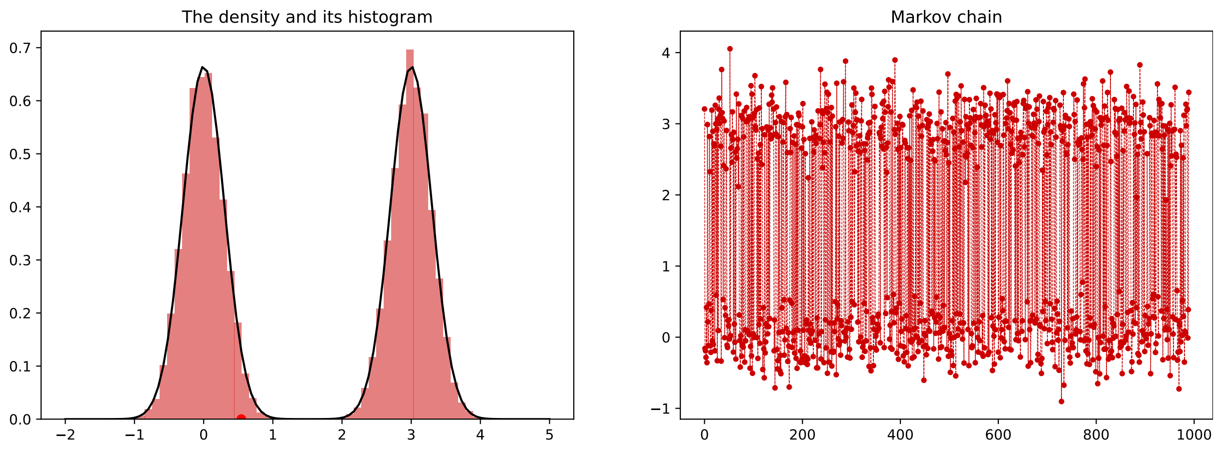


Figure 5.2: Random walk Gaussian proposal for a mixture of two Gaussians.

Solution. The acceptance ratio can be computed in this case as:

$$\begin{aligned}
\mathsf{r}(x, x') &= \frac{p_\star(x')q(x)}{p_\star(x)q(x')} \\
&= \frac{\mathcal{N}(x'; \mu, \sigma^2)\mathcal{N}(x; \mu_q, \sigma_q^2)}{\mathcal{N}(x; \mu, \sigma^2)\mathcal{N}(x'; \mu_q, \sigma_q^2)} \\
&= \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x'-\mu)^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x'-\mu_q)^2}{2\sigma_q^2}\right)} \\
&= \frac{\exp\left(-\frac{(x'-\mu)^2}{2\sigma^2}\right) \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right)}{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \exp\left(-\frac{(x'-\mu_q)^2}{2\sigma_q^2}\right)} \\
&= e^{\left(-\frac{1}{2\sigma^2}\left[(x'-\mu)^2-(x-\mu)^2\right]\right)} e^{\left(-\frac{1}{2\sigma_q^2}\left[(x-\mu_q)^2-(x'-\mu_q)^2\right]\right)}
\end{aligned}$$

5.3.2 Random walk (symmetric) proposals

Another important class of proposals is the random walk proposal. In this case, the proposal does use the current state X_{n-1} to define a proposal $q(x'|x_{n-1})$. These proposals in the random walk (and more generally symmetric) case result in a density that is symmetric, i.e., $q(x'|x) = q(x|x')$. This leads to a considerable simplification in the acceptance ratio calculations. We will see some examples below.

Example 5.7 (Random walk Gaussian proposal). Consider a mixture Gaussian target:

$$p_\star(x) = w_1\mathcal{N}(x; \mu_1, \sigma_1^2) + w_2\mathcal{N}(x; \mu_2, \sigma_2^2),$$

with some $w_1, w_2 > 0$ and $w_1 + w_2 = 1$. Assume we want to use MH to sample from it. Choose a proposal

$$q(x'|x) = \mathcal{N}(x'; x, \sigma_q^2),$$

and derive the acceptance ratio.

Solution. This proposal is symmetric so we can write

$$\begin{aligned}
\mathsf{r}(x, x') &= \frac{p_\star(x') q(x|x')}{p_\star(x) q(x'|x)} \\
&= \frac{p_\star(x')}{p_\star(x)}, \\
&= \frac{w_1 \mathcal{N}(x'; \mu_1, \sigma_1^2) + w_2 \mathcal{N}(x'; \mu_2, \sigma_2^2)}{w_1 \mathcal{N}(x; \mu_1, \sigma_1^2) + w_2 \mathcal{N}(x; \mu_2, \sigma_2^2)},
\end{aligned}$$

which is a considerable simplification. See Fig. 5.2 for a demonstration.

5.3.3 GRADIENT BASED (LANGEVIN) PROPOSALS

One of the powerful proposal alternatives is to choose the proposal based on the gradient of the target distribution $p_\star$. Note that we can compute $\nabla \log p_\star(x)$ without necessarily needing the normalising constant, since

$$\nabla \log p_\star(x) = \nabla \log \bar{p}_\star(x) - \underbrace{\nabla \log Z}_{0}$$

Therefore, without doing much more than what we are already doing (using unnormalised density), we can *inform* the proposal by using the gradient of the target distribution:

$$q(x'|x) = \mathcal{N}(x'; x + \gamma \nabla \log p_\star(x), 2\gamma I),$$

This algorithm is widely popular in the fields of statistics and especially in machine learning. This approach is called *Metropolis adjusted Langevin algorithm* (MALA)

5.3.4 BAYESIAN INFERENCE WITH METROPOLIS-HASTINGS

We can finally use the Metropolis-Hastings method for Bayesian inference. In what follows, we will provide some examples for this and visualisations resulting from the sampling procedures.

Recall that, with conditionally independent observations $y_1, \ldots, y_n$, we have the Bayes theorem as

$$p(x|y_{1:n}) = \frac{p(y_{1:n}|x)p(x)}{p(y_{1:n})} = \frac{\prod_{i=1}^n p(y_i|x)p(x)}{p(y_{1:n})}.$$

We write

$$p(x|y_{1:n}) \propto \prod_{i=1}^n p(y_i|x)p(x),$$

and set

$$\bar{p}_\star(x) = \prod_{i=1}^n p(y_i|x)p(x),$$

which is the unnormalised posterior density. We can then use the Metropolis-Hastings algorithm to sample from this posterior density. A generic Metropolis-Hastings method for Bayesian inference is described in Algorithm 10.

Algorithm 10 Pseudocode for Metropolis Hastings method for Bayesian inference

```
1: Input: The number of samples $N$, and starting point $X_0$.
2: for $i = 1, \dots, N$ do
3:     Propose (sample): $X' \sim q(x'|X_{n-1})$
4:     Accept the sample $X_n = X'$ with probability
           $$\alpha(X_{n-1}, X') = \min\left\{1, \frac{\bar{p}_\star(x')q(x_{n-1}|x')}{\bar{p}_\star(x_{n-1})q(x'|x_{n-1})}\right\}.$$
5:     Otherwise reject the sample and set $X_n = X_{n-1}$.
6: end for
7: Discard first burnin samples and return the remaining samples.
```

Example 5.8 (Source localisation)**.** This example is taken from Cemgil (2014). Consider the problem of source localisation in the presence of three sensors with three noisy observations. The setup in this example can be seen from the left part of Fig. 5.3. We have three sensors surrounding an object we are trying to locate. The sensors receive noisy observations on $\mathbb{R}^2$. We are trying to locate the object based on these observations. We define our prior rather broadly: $p(x) = \mathcal{N}(x; \mu, \sigma^2 I)$ where $\mu = [0, 0]$ and $\sigma^2 = 20$. We assume that the observations are coming from

$$p(y_i|x, s_i) = \mathcal{N}(y_i; \|x - s_i\|, \sigma_y^2),$$

where s_i is the location of the ith sensor on $\mathbb{R}^2$ for $i = 1, 2, 3$. We assume that the observations are independent and that the noise is independent of the location of the object (of course, for the sake of the example, we simulate our observations from the true model which is not the case in the real world). Devise a MH sampler to sample from the posterior density of x, i.e., the distribution over the location of the hidden object.

Solution. We can write the posterior density as

$$p(x|y_1, y_2, y_3, s_1, s_2, s_3) \propto p(y_1, y_2, y_3|x, s_1, s_2, s_3)p(x),$$

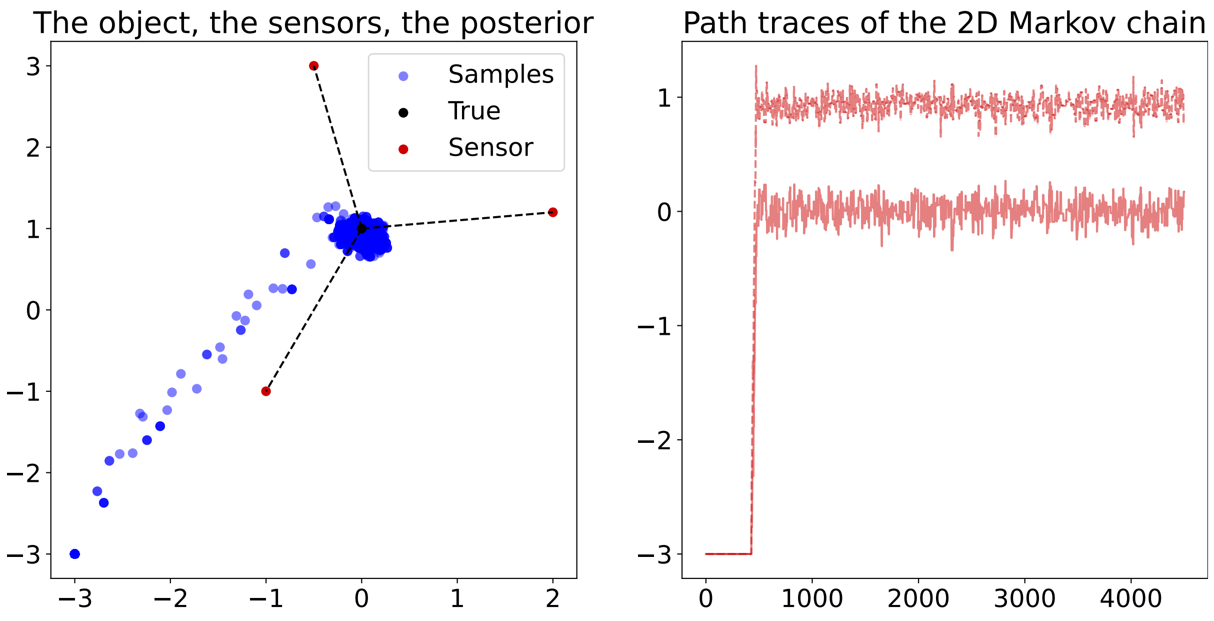


Figure 5.3: Solution of the source localisation problem.

and given conditional independence we have

$$p(x|y_1, y_2, y_3, s_1, s_2, s_3) \propto p(x) \prod_{i=1}^{3} p(y_i|x, s_i).$$

This sort of Bayes update follows from the conditional Bayes rule introduced in Prop. 3.2. In order to design the MH scheme, therefore, we need to just evaluate the likelihood and the prior for MH. We choose a random walk proposal:

$$q(x'|x) = \mathcal{N}(x'; x, \sigma^2 I).$$

This is symmetric so the acceptance ratio is:

$$\mathsf{r}(x, x') = \frac{p(x')p(y_1|x', s_1)p(y_2|x', s_2)p(y_3|x', s_3)}{p(x)p(y_1|x, s_1)p(y_2|x, s_2)p(y_3|x, s_3)}.$$

An example solution to this problem can be seen from Fig. 5.3 and the code can be accessed from our online companion.

Example 5.9 (Gaussian with unknown mean and variance Example 5.13 in Yıldırım (2017))**.** Assume that we observe

$$Y_1, \ldots, Y_n|z, s \sim \mathcal{N}(y_i; z, s)$$

where we do not know z and s. Assume we have an independent prior on z and s:

$$p(z)p(s) = \mathcal{N}(z; m, \kappa^2)\mathcal{IG}(s; \alpha, \beta).$$

where $\mathcal{IG}(s; \alpha, \beta)$ is the inverse Gamma distribution

$$\mathcal{IG}(s; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} s^{-\alpha-1} \exp\left(-\frac{\beta}{s}\right).$$

Design an MH algorithm to sample from the posterior distribution $p(z, s|y_{1:n})$.

Solution. We can explicitly write the priors as

$$p(z)p(s) = \frac{1}{\sqrt{2\pi\kappa^2}} \exp\left(-\frac{(z-m)^2}{2\kappa^2}\right) \frac{\beta^\alpha}{\Gamma(\alpha)} s^{-\alpha-1} \exp\left(-\frac{\beta}{s}\right).$$

We are after the posterior distribution

$$\begin{aligned} p(z, s|y_1, \ldots, y_n) &\propto p(y_1, \ldots, y_n|z, s)p(z)p(s), \\ &= \prod_{i=1}^n \mathcal{N}(y_i; z, s)\mathcal{N}(z; m, \kappa^2)\mathcal{IG}(s; \alpha, \beta). \end{aligned}$$

Let us call our unnormalised posterior as $\bar{p}_\star(z, s|y_{1:n})$. In order to do this, we need to design proposals over z and s. We choose a random walk proposal for z:

$$q(z'|z) = \mathcal{N}(z'; z, \sigma_q^2).$$

and an independent proposal for s:

$$q(s') = \mathcal{IG}(s'; \alpha, \beta).$$

The joint proposal therefore is

$$q(z', s'|z, s) = \mathcal{N}(z'; z, \sigma_q^2)\mathcal{IG}(s'; \alpha, \beta).$$

When we design the MH algorithm, we see that the acceptance ratio is

$$\begin{aligned} \mathsf{r}(z, s, z', s') &= \frac{\bar{p}_\star(z', s'|y_{1:n})q(z, s|z', s')}{\bar{p}_\star(z, s|y_{1:n})q(z', s'|z, s)} \\ &= \frac{p(z')p(s')\left[\prod_{k=1}^n \mathcal{N}(y_k; z', s')\right]\mathcal{N}(z; z', \sigma_q^2)p(s)}{p(z)p(s)\left[\prod_{k=1}^n \mathcal{N}(y_k; z, s)\right]\mathcal{N}(z'; z, \sigma_q^2)p(s')} \\ &= \frac{\mathcal{N}(z'; m, \kappa^2)\left[\prod_{k=1}^n \mathcal{N}(y_k; z', s')\right]}{\mathcal{N}(z; m, \kappa^2)\left[\prod_{k=1}^n \mathcal{N}(y_k; z, s)\right]} \end{aligned}$$

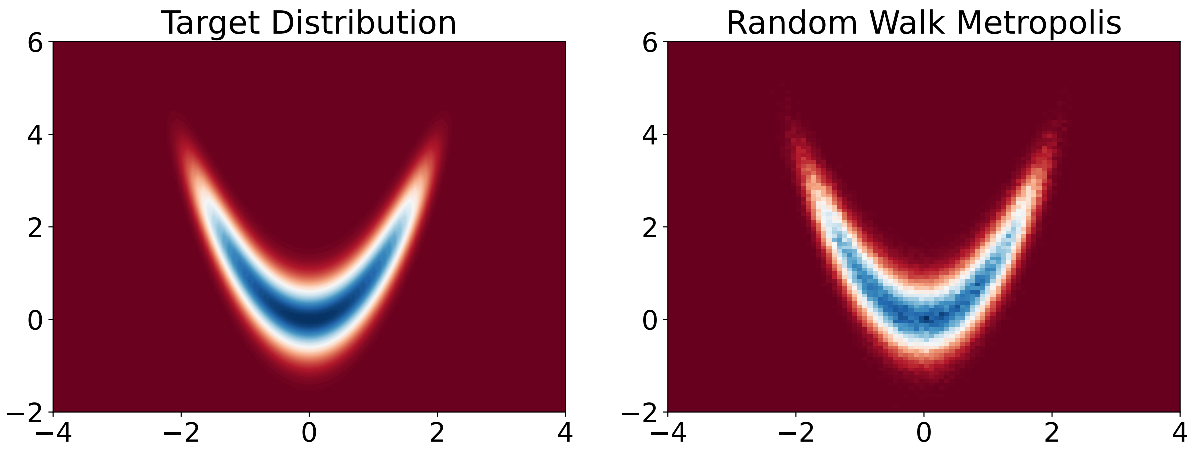


Figure 5.4: Banana density estimation using Random walk metropolis and plotting the histogram.

Example 5.10 (The banana density)**.** Consider the following density:

$$p(x, y) \propto \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y - x^2)^2\right).$$

This is only available in unnormalised form and it is an excellent test problem for many algorithms to fail. Design an MH algorithm for this density.

Solution. We have

$$\bar{p}_\star(x, y) = \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y - x^2)^2\right).$$

and let us choose the proposal

$$q(x', y'|x, y) = \mathcal{N}(x'; x, \sigma_q^2)\mathcal{N}(y'; y, \sigma_q^2).$$

This is a symmetric proposal so the acceptance ratio is

$$\mathsf{r}(x, y, x', y') = \frac{\bar{p}_\star(x', y')}{\bar{p}_\star(x, y)}.$$

Note that it makes sense to only compute log-acceptance ratio here

$$\log \mathsf{r}(x, y, x', y') = \log \bar{p}_\star(x', y') - \log \bar{p}_\star(x, y),$$

and implement the acceptance rate by drawing $U \sim \text{Unif}(0, 1)$ and accepting if $\log U < \log \mathsf{r}(x, y, x', y')$. The result can be seen from Fig. 5.4.

5.4 GIBBS SAMPLING

We will now go into another major class of MCMC samplers, called Gibbs samplers. The idea of Gibbs samplers is that, given a joint distribution of many variables $p(x_1, \ldots, x_d)$, one can build a Markov chain that samples from this distribution by sampling from the conditional distributions. This will also allow us straightforwardly sample from high-dimensional distributions. The downside of this approach is that, one has to derive the conditional distributions, which can be difficult. However, if one can do this, then the Gibbs sampler can be a very efficient method.

In this chapter, we denote our target similarly as $p_\star(x)$ where $x \in \mathbb{R}^d$ and define the *full conditional* distributions as

$$p_{m,\star}(x_m|x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_d) = p_{m,\star}(x_m|x_{1:m-1}, x_{m+1:d}) = p_{m,\star}(x_m|x_{-m}),$$

where $x_{-m} = (x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_d)$ is the vector of all variables except x_m. For a moment, assume that the full conditionals are available. Also assume, we obtain $X_{n-1} \in \mathbb{R}^d$ at the $n-1$'th iteration of the algorithm. To denote individual components, we use $X_{n-1,m}$ to denote the m'th component of X_{n-1}. Of course, the key aspect of the Gibbs sampler is to derive the

Algorithm 11 Pseudocode for the Gibbs sampler

```
1: Input: The number of samples $N$, and starting point $X_0 \in \mathbb{R}^d$.
2: for $n = 1, \ldots, N$ do
3:     Sample
           $X_{n,1} \sim p_{1,\star}(X_{n,1}|X_{n-1,2}, \ldots, X_{n-1,d})$
           $X_{n,2} \sim p_{2,\star}(X_{n,2}|X_{n,1}, X_{n-1,3}, \ldots, X_{n-1,d})$
           $X_{n,3} \sim p_{3,\star}(X_{n,3}|X_{n,1}, X_{n,2}, X_{n-1,4}, \ldots, X_{n-1,d})$
           $X_{n,d} \sim p_{d,\star}(X_{n,d}|X_{n,1}, X_{n,2}, \ldots, X_{n,d-1})$
4: end for
5: Discard first burnin samples and return the remaining samples.
```

full conditional distributions. We will come back to this point, but we will first investigate why the Gibbs sampling approach provides us a valid MCMC kernel, in other words, how the Gibbs sampler satisfies the detailed balance.

Let us denote $x = (x_m, x_{-m})$. It is easy to see from Algorithm 11 that the Gibbs sampler for every iteration (at time n) is defined as d separate operations, each sampling from the conditional distribution. We can first look at what goes on in each of these d updates . It is also easy to see that, the kernel defined in each of these d updates is given as

$$K_m(x'|x) = p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}),$$

where $\delta_{x_{-m}}(x'_{-m})$ is the Dirac delta function. Intuitively, each step samples from the full conditional $p_{m,\star}(\cdot|x_{-m})$ for mth dimension where $m \in \{1, \ldots, d\}$ and leaves others unchanged, which is enforced by the term $\delta_{x_{-m}}(x'_{-m})$. One can then see that the entire Gibbs kernel can be written as

$$K = K_1 K_2 \ldots K_d.$$

Note that each kernel is an *integral operator* – therefore the above equation is almost symbolic, it does not mean multiplication of kernels. We will now show that the Gibbs kernel satisfies the detailed balance.

Proposition 5.3. *The Gibbs kernel K leaves the target distribution $p_\star$ invariant.*

Proof. We first show that each kernel K_m satisfies the detailed balance condition:

$$\begin{aligned} p_\star(x)K_m(x'|x) &= p_\star(x)p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}) \\ &= p_\star(x_{-m})p_{m,\star}(x_m|x_{-m})p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}) \\ &= p_\star(x'_{-m})p_{m,\star}(x'_m|x'_{-m})p_{m,\star}(x_m|x'_{-m})\delta_{x'_{-m}}(x_{-m}) \\ &= p_\star(x')K_m(x|x'). \end{aligned}$$

The steps of this derivation follows from the fact that the use of Dirac allows us to exchange variables x and x'. This shows that K_m satisfies the detailed balance condition, therefore, we have

$$\int K_m(x'|x)p_\star(x)\mathrm{d}x = p_\star(x'),$$

i.e., K_m leaves $p_\star$ invariant. Let us denote now the integral

$$(K_m, p_\star) = \int K_m(x'|x)p_\star(x)\mathrm{d}x = p_\star.$$

One can see that we have $(K_2, (K_1, p_\star)) = (K_2, p_\star) = p_\star$, which is true for all $m = 1, \ldots, d$. Therefore, we see that application of d kernels $K_1, \ldots, K_d$ will leave $p_\star$ invariant. □

We can also see why Gibbs sampling works by relating it to the Metropolis-Hastings algorithm. Recall that we can see our sampling from the conditional as a proposal, i.e.,

$$q_m(x'|x) = p_{m,\star}(x'_m|x_{-m})\delta_{x_{-m}}(x'_{-m}).$$

If we calculate the acceptance ratio for this proposal

$$\begin{aligned} \alpha_m(x'|x) &= \min\left\{1, \frac{p_\star(x')q_m(x|x')}{p_\star(x)q_m(x'|x)}\right\}, \\ &= \min\left\{1, \frac{p_\star(x')K_m(x|x')}{p_\star(x)K_m(x'|x)}\right\}. \end{aligned}$$

We see that this is equal to 1 as the detailed balance is satisfied for q_m (which is K_m – see the proof of Proposition 5.3).

As we noted before, we have shown that the kernel K would leave $p_\star$ invariant, but this would not give us proper convergence guarantees. Note that the version of the algorithm we presented is called *deterministic scan* Gibbs sampler. The reason for this is that the algorithm in Alg. 11 is implemented so that we sample $x_1, \ldots, x_d$ in order, *scanning* the variables deterministically. It turns out, while this sampler's convergence guarantees cannot be established easily, there is an algorithmic fix which results in a procedure that is also guaranteed to converge. Instead of scanning the variables deterministically, we can sample them in a random order. This is called the *random scan* Gibbs sampler. The algorithm is given in Alg. 12. We will now see an example.

Algorithm 12 Random scan Gibbs sampler

```
1: Input: The number of samples $N$, and starting point $X_0 \in \mathbb{R}^d$.
2: for $n = 1, \ldots, N$ do
3:     Sample $j \sim \{1, \ldots, d\}$
           $X_{n,j} \sim p_{j,\star}(X_{n,j} | X_{n,-j})$,
4: end for
```

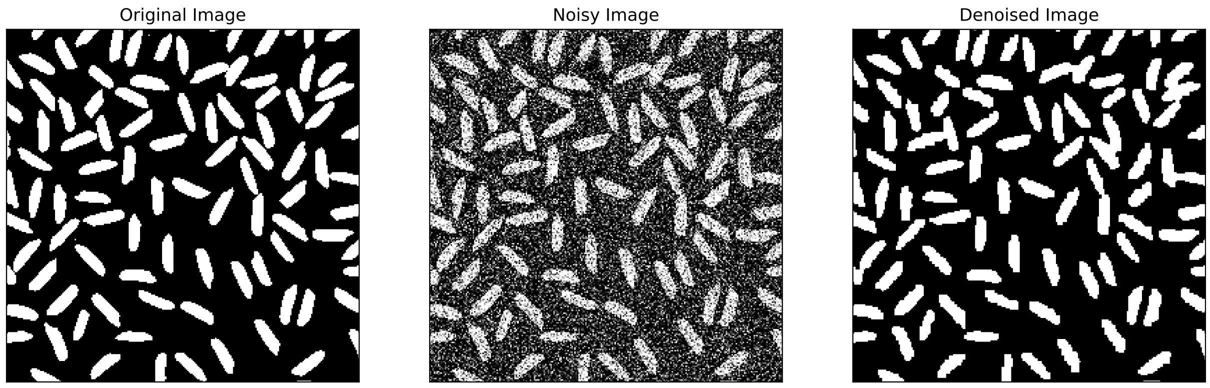


Figure 5.5: Denoising of an image using Gibbs sampler. The left column shows the original image, the middle column shows the noisy image, and the right column shows the denoised image. I used $\sigma = 1$, $J = 4$ for this and the Gibbs sampler scanned the entire image only 10 times.

Example 5.11 (Image denoising)**.** A biologist knocked your door as some of the images from the microscope were too noisy. You decide to help and use the Gibbs sampler for this. The model is given as follows.

Consider a set of random variables X_{ij} for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. This is a matrix modeling an $m \times n$ image. We assume that we have an image that takes values $X_{ij} \in \{-1, 1\}$ – note that this is an "unusual" image, as the images usually take values between $[0, 255]$ (or $[0, 1]$). We assume that the image is corrupted by noise, i.e., we have a noisy image

$$Y_{ij} = X_{ij} + \sigma \epsilon_{ij},$$

where $\epsilon_{ij} \sim \mathcal{N}(0, 1)$ and σ is the standard deviation of the noise. We assume that the noise is independent of the image. We want to recover the image X_{ij} from the noisy image Y_{ij} and utilise Gibbs sampler for this purpose.

The goal is obtain (conceptually) $p(X|Y)$, i.e., samples from $p(X|Y)$ given Y. For this, we need to specify a prior $p(X)$. We take this from the literature and place as a prior a smooth Markov random field (MRF) assumption. This is formalised as

$$p(X_{ij}|X_{-ij}) = \frac{1}{Z} \exp(J X_{ij} W_{ij}),$$

where W_{ij} is the sum of the X_{ij}'s in the neighbourhood of X_{ij}, i.e.,

$$W_{ij} = \sum_{kl:\text{neighbourhood of } (i,j)} X_{kl} = X_{i-1,j} + X_{i+1,j} + X_{i,j-1} + X_{i,j+1}.$$

This is an intuitive model of the image, making the current value of the pixel depend on the values of its neighbours. The exercise is to design the Gibbs sampler for this problem.

Solution. We aim at using a Gibbs sampler approach from sampling the posterior $p(X|Y)$. Note that now we need to sample from full conditionals, e.g., for each (i, j), we need to sample from $X_{ij} \sim p(X_{ij}|X_{-ij}, Y_{ij})$. We derive the full conditional as

$$p(X_{ij} = k|X_{-ij}, Y_{ij}) = \frac{p(Y_{ij}|X_{ij} = k)p(X_{ij} = k|X_{-ij})}{\sum_{k\in\{-1,1\}} p(Y_{ij}|X_{ij} = k)p(X_{ij} = k|X_{-ij})},$$

where $p(Y_{ij}|X_{ij} = k) = \mathcal{N}(Y_{ij}; k, \sigma^2)$ is the likelihood of the noisy image given the value of the pixel. We can easily compute these probabilities since each term in the Bayes rule is computable (and $1/Z$ cancels). Therefore, we can get explicit expressions for $q = p(X_{ij} = 1|X_{-ij}, Y_{ij})$ and $1 - q = p(X_{ij} = -1|X_{-ij}, Y_{ij})$. We can then sample from the full conditional as

$$X_{ij} \sim \begin{cases} 1 & \text{with probability } q, \\ -1 & \text{with probability } 1 - q. \end{cases}$$

We can now loop over (i, j) (to sample from each full conditional) and sample from the full conditionals. This is the Gibbs sampler algorithm as described above. The results of this procedure can be seen from Fig. 5.5.

Let us consider another example.

Example 5.12 (Beta-Binomial Gibbs sampler)**.** Consider the following model

$$p(\theta) = \text{Beta}(\theta; \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1},$$

and

$$p(x|\theta) = \text{Bin}(x; n, \theta) = \binom{n}{x}\theta^x(1-\theta)^{n-x}.$$

Derive the Gibbs sampler to sample from the joint distribution $p(x, \theta)$.

Solution. We know that, for this, we need full conditionals, i.e., we need $p(x|\theta)$ and $p(\theta|x)$. We can see that $p(x|\theta)$ is already provided in the definition of the model. Therefore, we only need to derive the posterior. We can write the joint distribution as

$$\begin{aligned} p(x, \theta) = p(x|\theta)p(\theta) &= \binom{n}{x}\theta^x(1-\theta)^{n-x}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\binom{n}{x}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1}. \end{aligned}$$

For Bayes theorem $p(\theta|x) = p(x|\theta)p(\theta)/p(x)$, we also need to compute $p(x)$. This is given by

$$\begin{aligned} p(x) = \int_0^1 p(x,\theta)\mathrm{d}\theta &= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\binom{n}{x}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1}\mathrm{d}\theta, \\ &= \binom{n}{x}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\frac{\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(n+\alpha+\beta)}. \end{aligned}$$

Therefore, we can compute the posterior as

$$\begin{aligned} p(\theta|x) = \frac{p(x,\theta)}{p(x)} &= \frac{\cancel{\binom{n}{x}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1}}{\cancel{\binom{n}{x}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}}\frac{\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(n+\alpha+\beta)}}, \\ &= \mathrm{Beta}(\theta; x+\alpha, n-x+\beta). \end{aligned}$$

Therefore we can sample from $p(\theta|x)$ using any method to simulate a Beta variable. The Gibbs sampler is then defined as follows:

- Initialise x_0, θ_0
- For $k = 1, 2, \ldots$:
  - Sample $\theta_k \sim p(\theta|x_{k-1})$
  - Sample $x_k \sim p(x|\theta_k)$
- Return x_k, θ_k for $k = 1, 2, \ldots$.

We also note that simulated x_k are approximately from $p(x)$ which also gives us a way to approximate $p(x)$.

5.4.1 METROPOLIS-WITHIN-GIBBS

One remarkable feature of the Gibbs sampler is that when we cannot derive the full conditionals (or too lazy to do it), we can instead target the full conditional with a single Metropolis step at each iteration. This is called the Metropolis-within-Gibbs algorithm and, remarkably, it samples from the correct posterior!

Consider a generic target $p(x, y)$. In many situations, it is easier to write *unnormalised* full conditionals, i.e., we would have access to $\bar{p}(x|y)$ and $\bar{p}(y|x)$, but not $p(x|y)$ and $p(y|x)$. In this case, we can use the Metropolis-within-Gibbs algorithm – meaning that instead of sampling from the full conditional, we can take a Metropolis step to sample from the full conditional. The algorithm would be summarised as follows. Given (x_{n-1}, y_{n-1})

1. Sample $x' \sim q_x(x'|x_{n-1})$ and accept x' with probability

   $$\mathsf{r}_x = \frac{\bar{p}(x'|y_{n-1})q_x(x_{n-1}|x')}{\bar{p}(x_{n-1}|y_{n-1})q_x(x'|x_{n-1})}$$

   i.e., set $x_n = x'$ with probability r_x and $x_n = x_{n-1}$ otherwise.

2. Sample $y' \sim q_y(y'|y_{n-1})$ and accept y' with probability

$$\mathsf{r}_y = \frac{\bar{p}(y'|x_n)q_y(y_{n-1}|y')}{\bar{p}(y_{n-1}|x_n)q_y(y'|y_{n-1})}$$

i.e., set $y_n = y'$ with probability r_y and $y_n = y_{n-1}$ otherwise.

Example 5.13 (Metropolis-within-Gibbs for Example 5.9)**.** Let us return to Example 5.9. To recall the model, assume that we observe

$$Y_1, \ldots, Y_n | z, s \sim \mathcal{N}(y_i; z, s)$$

where we do not know z and s. Assume we have an independent prior on z and s:

$$p(z)p(s) = \mathcal{N}(z; m, \kappa^2)\mathcal{IG}(s; \alpha, \beta).$$

where $\mathcal{IG}(s; \alpha, \beta)$ is the inverse Gamma distribution

$$\mathcal{IG}(s; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} s^{-\alpha-1} \exp\left(-\frac{\beta}{s}\right).$$

In other words, we have

$$p(z)p(s) = \frac{1}{\sqrt{2\pi\kappa^2}} \exp\left(-\frac{(z-m)^2}{2\kappa^2}\right) \frac{\beta^\alpha}{\Gamma(\alpha)} s^{-\alpha-1} \exp\left(-\frac{\beta}{s}\right).$$

We are after the posterior distribution

$$\begin{aligned} p(z, s|y_1, \ldots, y_n) &\propto p(y_1, \ldots, y_n|z, s)p(z)p(s), \\ &= \prod_{i=1}^n \mathcal{N}(y_i; z, s)\mathcal{N}(z; m, \kappa^2)\mathcal{IG}(s; \alpha, \beta). \end{aligned}$$

Let us call our unnormalised posterior as $\bar{p}_\star(z, s|y_{1:n})$. Now instead of MH or defining Gibbs (requires us to derive full conditionals), derive the Metropolis-within-Gibbs algorithm.

Solution. For this, note the unnormalised full conditionals:

$$\bar{p}_\star(z|s, y_{1:n}) = \prod_{i=1}^n \mathcal{N}(y_i; z, s)\mathcal{N}(z; m, \kappa^2),$$

and

$$\bar{p}_\star(s|z, y_{1:n}) = \prod_{i=1}^n \mathcal{N}(y_i; z, s)\mathcal{IG}(s; \alpha, \beta).$$

In order to do this, we need to design proposals over z and s to target $\bar{p}(z|s, y_{1:n})$ and $\bar{p}(s|z, y_{1:n})$ respectively. This step will be a standard Metropolis as if we are solving each problem independently. We choose a random walk proposal for z:

$$q(z'|z) = \mathcal{N}(z'; z, \sigma_q^2).$$

and an independent proposal for s:

$$q(s') = \mathcal{IG}(s'; \alpha, \beta).$$

Therefore, we Metropolis-within-Gibbs can be implemented as follows

- Initialise z_0, s_0
- For $k = 1, 2, \ldots$:
- Metropolis step for z-marginal:
  - Sample $z' \sim q(z'|z_{k-1})$
  - Accept z' and set $z_k = z'$ with probability
$$\mathsf{r}_z = \frac{\bar{p}_\star(z'|s_{k-1}, y_{1:n})}{\bar{p}_\star(z_{k-1}|s_{k-1}, y_{1:n})}$$
    which is simplified due to the symmetric proposal.
  - Otherwise set $z_k = z_{k-1}$.
- Metropolis step for s-marginal:
  - Sample $s' \sim q(s')$
  - Accept s' and set $s_k = s'$ with probability
$$\begin{aligned}\mathsf{r}_s &= \frac{\bar{p}_\star(s'|z_k, y_{1:n})q(s_{k-1})}{\bar{p}_\star(s_{k-1}|z_k, y_{1:n})q(s')} \\ &= \frac{\prod_{i=1}^n \mathcal{N}(y_i; z, s')}{\prod_{i=1}^n \mathcal{N}(y_i; z, s_{k-1})}\end{aligned}$$
  - Otherwise set $s_k = s_{k-1}$.
- Return z_k, s_k for $k = 1, 2, \ldots$.

5.5 LANGEVIN MCMC METHODS

We briefly introduced Metropolis-adjusted Langevin algorithm (MALA) in Sec. 5.3.3. MALA is just one example of a more general class of samplers, called Langevin MCMC algorithms, which are based on Langevin dynamics. These approaches are at the forefront of modern MCMC methods, used in a variety of settings, including sampling from high-dimensional targets, deep learning, sampling from Bayesian neural networks, and so on. We will now introduce the Langevin MCMC methods and see how they work.

Consider again our target $p_\star$ defined on $\mathbb{R}^d$. It turns out, we can use stochastic differential equations (SDEs) to sample from $p_\star$ (recall that SDEs are differential equations with a stochastic term). Consider the following SDE:

$$\mathrm{d}X_t = \nabla \log p_\star(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t, \tag{5.5}$$

where B_t is a standard Brownian motion. It turns out the marginal distributions of X_t driven by this SDE converge to $p_\star$ as $t \to \infty$. In other words, in many suitable metrics, we can quantify the convergence $d(p_t, p_\star) \to 0$ as $t \to \infty$. This means that all we need to draw samples from $p_\star$ is to numerically solve this SDE which can be done with a variety of numerical methods (akin to ODE solvers) . One caveat in this situation is that, while the SDE would target $p_\star$, its discretisation would incur bias. This is why MALA is "Metropolised".

Let us recall the MALA algorithm. We start with a point X_0 and then define the proposal

$$q(x_n|x_{n-1}) = \mathcal{N}(x_n; x_{n-1} + \gamma\nabla \log p_\star(x_{n-1}), 2\gamma I_d), \tag{5.6}$$

where $\gamma > 0$ is a step size. We then sample from q to obtain X_n. We then accept X_n with probability

$$\alpha(X_n, X_{n-1}) = \min\left(1, \frac{p_\star(X_n)q(X_{n-1}|X_n)}{p_\star(X_{n-1})q(X_n|X_{n-1})}\right). \tag{5.7}$$

Recall that the MALA proposal would not define a symmetric proposal, therefore, we would need to compute the ratio. Now one can see that, Eq . (5.6) can be equivalently written as

$$X_n = X_{n-1} + \gamma\nabla \log p_\star(X_{n-1}) + \sqrt{2\gamma}W_n, \tag{5.8}$$

where $W_n \sim \mathcal{N}(0, I)$. The relationship of Eq. (5.8) to (5.5) can be seen by noting that the discretisation of the SDE in (5.5) would exactly take the form of Eq. (5.8). Therefore, MALA uses this Langevin SDE as the proposal and then accept/reject its samples. This has a beneficial effect of correcting the bias of the discretisation.

However, the Metropolis step can be computationally infeasible in higher dimensions, just as in the case of rejection sampling. Higher dimensional problems cause the acceptance rate to vanish, which results in slow convergence. To remedy this situation, a common approach is to simply drop the Metropolis step and use the following iteration:

$$X_n = X_{n-1} + \gamma\nabla \log p_\star(X_{n-1}) + \sqrt{2\gamma}W_n. \tag{5.9}$$

This is a simple (and biased) MCMC method - which is called the unadjusted Langevin algorithm (ULA). The ULA is a discretisation of the SDE, and as such, its stationary measure is not $p_\star$. However, under various conditions, it can be shown that the limiting distribution of ULA $p_\star^\gamma$ can be made arbitrarily close to $p_\star$ as $\gamma \to 0$. This means that the ULA can be a viable alternative.

Example 5.14. Consider the target $p_\star(x) = \mathcal{N}(x; \mu, \sigma^2)$. Write down the ULA for this target and derive the stationary distribution of the chain.

Solution. We can write the ULA as

$$\frac{\partial}{\partial x} \log p_\star(x) = -\frac{x-\mu}{\sigma^2}.$$

We can then write the iterates of ULA as

$$\begin{aligned} X_n &= X_{n-1} + \gamma\frac{\partial}{\partial x}\log p_\star(X_{n-1}) + \sqrt{2\gamma}W_n, \\ &= X_{n-1} - \gamma\frac{X_{n-1}-\mu}{\sigma^2} + \sqrt{2\gamma}W_n, \\ &= \left(1 - \frac{\gamma}{\sigma^2}\right)X_{n-1} + \frac{\gamma}{\sigma^2}\mu + \sqrt{2\gamma}W_n, \end{aligned}$$

where $W_n \sim \mathcal{N}(0,1)$. In this simple case, we can compute the stationary distribution of the chain and analyse its relationship to the true target $p_\star$. Let

$$a = 1 - \frac{\gamma}{\sigma^2}, \quad b = \frac{\gamma}{\sigma^2}\mu.$$

We can write now the iterates beginning at x_0 as

$$\begin{aligned}
x_1 &= ax_0 + b + \sqrt{2\gamma}W_1, \\
x_2 &= \underbrace{a^2x_0 + ab + a\sqrt{2\gamma}W_1}_{ax_1} + b + \sqrt{2\gamma}W_2, \\
x_3 &= \underbrace{a^3x_0 + a^2b + a^2\sqrt{2\gamma}W_1 + ab + a\sqrt{2\gamma}W_2}_{ax_2} + b + \sqrt{2\gamma}W_3, \\
&\vdots \\
x_n &= a^n x_0 + \sum_{k=0}^{n-1} a^k b + \sum_{k=0}^{n-1} a^k \sqrt{2\gamma} W_{n-k}.
\end{aligned}$$

We can compute the expected value

$$\mathbb{E}[X_n] = a^n x_0 + \sum_{k=0}^{n-1} a^k b,$$

since W_k are zero mean. As $n \to \infty$, we have

$$\mu_\infty = \lim_{n\to\infty} \mathbb{E}[X_n] = \sum_{k=0}^{\infty} a^k b = \frac{b}{1-a} = \mu.$$

since $0 < a < 1$. The variance of the iterates as $n \to \infty$ can also be computed. Note that for finite n, we have

$$\begin{aligned}
\mathsf{var}(x_n) &= \mathsf{var}\left(\sum_{k=0}^{n-1} a^k \sqrt{2\gamma} W_k\right), \\
&= 2\gamma \sum_{k=0}^{n-1} (a^2)^k, \\
&= 2\gamma \frac{1-a^{2n}}{1-a^2}.
\end{aligned}$$

Therefore, we obtain the limiting variance as

$$\begin{aligned}
\lim_{n\to\infty} \mathsf{var}(x_n) &= 2\gamma \frac{1}{1-a^2} \\
&= 2\gamma \frac{1}{1-\left(1-\frac{\gamma}{\sigma^2}\right)^2} \\
&= 2\gamma \frac{1}{\frac{2\gamma}{\sigma^2} - \frac{\gamma^2}{\sigma^4}}, \\
&= \frac{2\sigma^4}{2\sigma^2 - \gamma}.
\end{aligned}$$

Therefore, we obtained the target measure of ULA as

$$p_\star^\gamma(x) = \mathcal{N}\left(x; \mu, \frac{2\sigma^4}{2\sigma^2 - \gamma}\right),$$

which is different than $p_\star$. Note that in this particular case, the means of the $p_\star^\gamma$ and $p_\star$ agree. It can be seen that the bias enters the picture through the variance, but this vanishes as $\gamma \to 0$.

Let us look at the ULA for the Banana density.

Example 5.15. Consider the Banana density

$$p(x, y) \propto \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y - x^2)^2\right).$$

Derive ULA for this density.

Solution. Note the density is only available in unnormalised form:

$$\bar{p}_\star(x, y) = \exp\left(-\frac{x^2}{10} - \frac{y^4}{10} - 2(y - x^2)^2\right).$$

Recall that $\nabla \log \bar{p}_\star(x, y) = \nabla \log p_\star(x, y)$. Therefore, we will directly compute the unnormalised gradients:

$$\nabla \log p_\star(x, y) = \begin{bmatrix} \frac{-x}{5} + 8x(y - x^2) \\ \frac{-2y^3}{5} - 4(y - x^2) \end{bmatrix}.$$

Therefore, the update in 2D is given by

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \gamma \nabla \log p_\star(x_n, y_n) + \sqrt{2\gamma} V_n$$

where $V_n \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$.

Example 5.16 (Bayesian inference with ULA)**.** Show that we can also straightforwardly perform Bayesian inference in this setting by derive ULA for a generic posterior density given $p(y|x)$ and $p(x)$.

Solution. Recall the target posterior density in this setting

$$\bar{p}_\star(x|y) = p(x) \prod_{k=1}^{n} p(y_k|x),$$

where $p(x)$ is the prior density and $p(y_k|x)$ is the likelihood where observations are conditionally i.i.d given x. We can write the ULA iterates as

$$\begin{aligned} X_n &= X_{n-1} + \gamma \nabla \log \bar{p}_\star(X_{n-1}|y) + \sqrt{2\gamma} V_n, \\ &= X_{n-1} + \gamma \left(\nabla \log p(X_{n-1}) + \sum_{k=1}^{n} \nabla \log p(y_k|X_{n-1}) \right) + \sqrt{2\gamma} V_n. \end{aligned}$$

A common problem arising in machine learning and statistics is *big data*, where the number of observations n is large. In this case, both ULA and MALA are infeasible as both require the iterates above to be evaluated, e.g., each iteration involves summing n terms. If n is order of millions, this is computationally infeasible. In this case, we can use the *stochastic* gradients. This is only applicable in the setting of ULA as we will see below, which is one reason why ULA-type methods are more popular than MALA-type methods.

5.5.1 STOCHASTIC GRADIENT LANGEVIN DYNAMICS

The problem of large number of data points arise in the setting of ULA as a sum, therefore, we should look for estimating large sums with something cheaper. Consider the following sum of (arbitrary) numbers:

$$g = \frac{1}{n} \sum_{k=1}^{n} g_i.$$

If n is simply too large to compute this sum efficiently, we can instead resort to *unbiased* estimates of it. This can be done by sampling $i_1, \ldots, i_K \sim \{1, \ldots, n\}$ uniformly and constructing

$$\hat{g} = \frac{1}{K} \sum_{k=1}^{K} g_{i_k}.$$

This estimate is an unbiased estimate of g, i.e.,

$$\mathbb{E}[\hat{g}] = \mathbb{E}\left[\frac{1}{K} \sum_{k=1}^{K} g_{i_k} \right] = \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}[g_{i_k}] = g.$$

This idea can be used to construct stochastic gradients. We provide some examples below.

Example 5.17 (Large scale Bayesian inference)**.** Recall the problem setting in Example 5.16:

$$X_n = X_{n-1} + \gamma \left(\nabla \log p(x) + \sum_{k=1}^{n} \nabla \log p(y_k|X_{n-1}) \right) + \sqrt{2\gamma} V_n.$$

Design the stochastic gradient sampler for this case.

Solution. Assume we sample uniformly $i_1, \ldots, i_K$ from $\{1, \ldots, n\}$, we can then approximate the sum

$$\sum_{k=1}^{n} \nabla \log p(y_k | X_{n-1}) \approx \frac{n}{K} \sum_{k=1}^{K} \nabla \log p(y_{i_k} | X_{n-1}).$$

Note that (n/K) factor comes here as the sum itself did not have $(1/n)$ term (as opposed to the sum example above). Therefore, the stochastic gradient Langevin dynamics (SGLD) iterate can be written as

$$X_n = X_{n-1} + \gamma \left(\nabla \log p(x) + \frac{n}{K} \sum_{k=1}^{K} \nabla \log p(y_{i_k} | X_{n-1}) \right) + \sqrt{2\gamma} V_n.$$

This is also called *data subsampling* as one can see that the gradient only uses a subset of the data. Every iteration is cheap and computable as we only need to compute K terms. This is a very popular method in Bayesian inference and is used in many applications.

5.6 MONITORING AND POSTPROCESSING MCMC OUTPUT

There are a number of ways to monitor the MCMC samples to ensure that the algorithm is working as expected. We will discuss a few of them here.

5.6.1 TRACE PLOTS

The simplest way to monitor the MCMC output is to plot the trace of the samples. This is a plot of the samples against the iteration number. This is what we have been doing in previous examples. If the trace plots show you that the chain is still "moving", then you can conclude that the chain is not yet converged. On the other hand, a trace plot from MH can also show you that the chain is stuck. It is then straightforward to conclude simple convergence issues from trace plots.

5.6.2 AUTOCORRELATION PLOTS

The autocorrelation plot is a plot of the autocorrelation function of the samples. The autocorrelation function is defined as

$$\rho_k = \frac{\text{Cov}(X_t, X_{t+k})}{\text{Var}(X_t)}.$$

This can be empirically computed on the samples coming from the Markov chain $(x_k)_{k \in \mathbb{N}}$. Since the aim of MCMC is to obtain nearly independent samples from a target $p_\star$, we expect a good MCMC chain to exhibit low autocorrelation. A bad chain which is not *mixing* well will exhibit high autocorrelation. An example can be seen from Fig. 5.6 and see its caption for more details. One way to choose the proposal variance is to ensure that the chain has a low autocorrelation. This is a very simple way to monitor the chain.

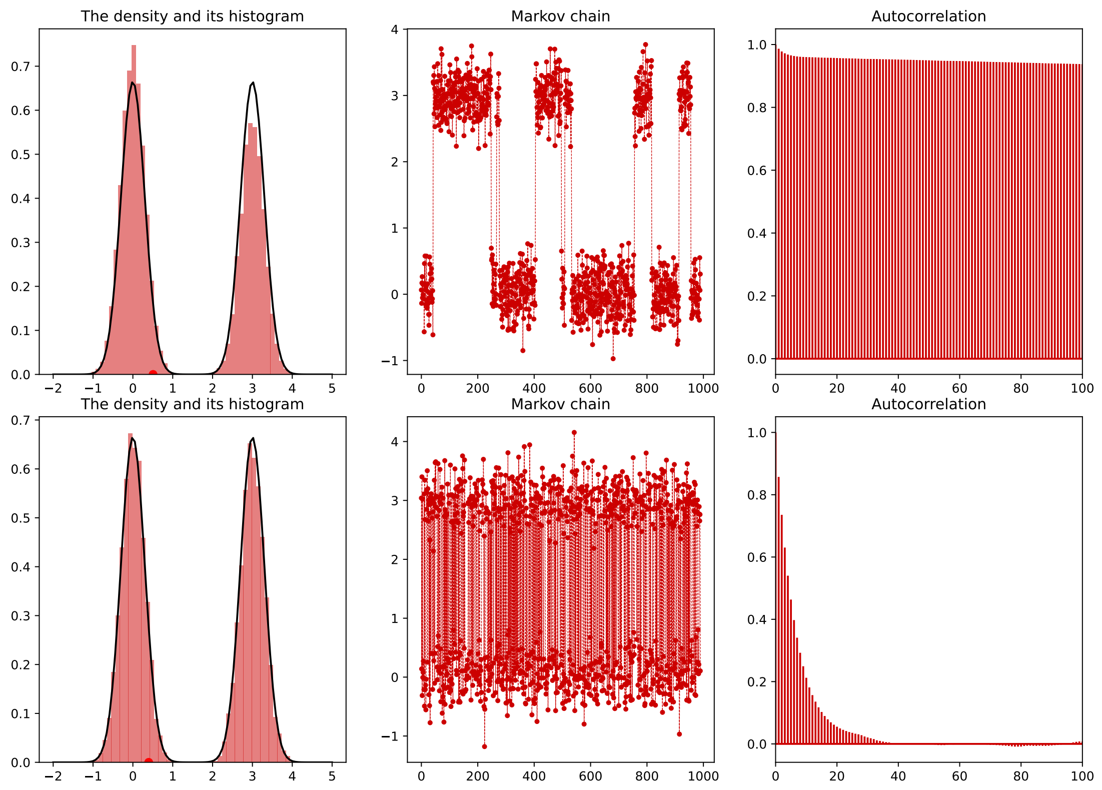


Figure 5.6: Random walk Metropolis-Hastings for a mixture of two Gaussians. The top panel shows the situation where $\sigma_q = 0.5$, so chain gets stuck in modes. This causes a high autocorrelation, and as such, this sampler is not considered to be a good one. When we set $\sigma_q = 4$, then the chain exhibits low autocorrelation and is a good sampler.

5.6.3 Effective sample size

There is a notion of effective sample size for MCMC methods. However, its computation is trickier than the IS one and it is usually implemented using software packages. The definition of the ESS for MCMC chains is given as

$$\text{ESS} = \frac{N}{1 + 2\sum_{k=1}^{\infty} \rho_k},$$

where ρ_k is the autocorrelation function. The ESS is an approximate measure of the number of independent samples that we have. For example, if the chain exhibits no autocorrelation, then the ESS is equal to the number of samples. If the chain exhibits high autocorrelation, then the ESS will be very low, as the sum in denominator will be large.

The computation of effective sample size in MCMC is usually done by software packages. We will not go into the details of this computation here.

5.6.4 Thinning the MCMC output

One way to reduce the autocorrelation of the MCMC samples is to *thin* them. This is a post-processing step that is done after the MCMC chain has been generated. The idea is to discard some of the samples and keep only a subset of them. This is done by keeping every kth sample. Since autocorrelation in an MCMC chain decays naturally over time, after reaching stationary,

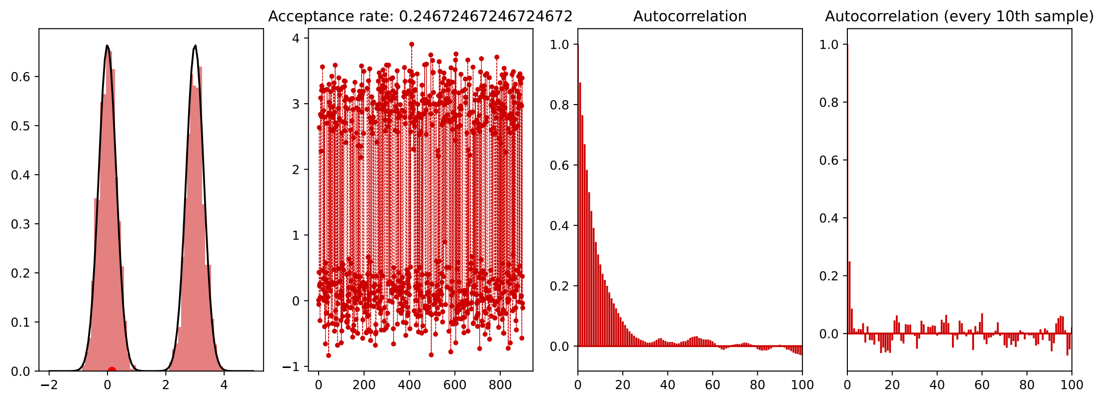


Figure 5.7: Thinning of MCMC samples. We keep every 10th sample for the same mixture of Gaussians example with $\sigma_q = 2$. It can be seen that the thinned MCMC chain exhibits significantly lower autocorrelation.

we can choose every kth of them and discard the rest. This will still give us a chain with the same stationary measure but with a lower autocorrelation. A demonstration of this can be seen from Fig .5.7.

6

MONTE CARLO FOR OPTIMISATION AND LEARNING

In this chapter, we draw connections between sampling algorithms we have introduced so far and the task of optimising a cost function. We will see that there are rich connections between the fields of sampling and optimisation which motivates most of the modern algorithmic tools in machine learning.

6.1 AN INTRODUCTION TO OPTIMISATION

Optimisation is a fundamental task we face repeatedly in numerical computation, virtually all fields of engineering, machine learning, statistics, and so on. The task is to find the minima of a cost function $f : \mathbb{R}^d \to \mathbb{R}$. More formally, in this section, we will look at simulation-based methods to solve the problem of the form

$$x^\star \in \operatorname*{argmin}_{x \in \mathsf{X}} f(x). \tag{6.1}$$

We will usually take $\mathsf{X} = \mathbb{R}^d$, although simulation methods will adapt easily to the case where X could be a closed, bounded set. The typical algorithms to solve (6.1) are *gradient-based*. This means that the typical numerical way to solve the problem (6.1) is to construct a sequence $(x_t)_{t\geq 0}$ so that

$$x_{n+1} = x_n - \gamma \nabla f(x_n), \tag{6.2}$$

where $\gamma > 0$ is a step size. This is the *gradient descent* algorithm. Particularly relevant to this chapter is the fact that the gradient descent algorithm can be seen as a discretisation of a continuous-time ordinary differential equation (ODE) called the *gradient flow*:

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = -\nabla f(x_t), \tag{6.3}$$

which can also be written informally as

$$\mathrm{d}x_t = -\nabla f(x_t)\mathrm{d}t.$$

The gradient flow has the intriguing property of converging to the minimisers exponentially fast for convex functions f.

The gradient descent algorithm is a simple and effective way to solve optimisation problems. However, it has its own limitations. For example, the gradient descent algorithm can get stuck in local minima, saddle points, or plateaus. In this section, we will see how we can use MCMC methods to solve the *global* optimisation problem (6.1) – that is, to find the global minima of the cost function f without getting stuck in local minima.

6.2 MCMC FOR OPTIMISATION

MCMC methods were originally motivated by optimisation problems. These methods are a good candidate to solve challenging, nonconvex optimisation problems with multiple minima due to the intrinsic noise in the algorithms. In this section, we will briefly look at two MCMC methods that can be used for optimisation: (i) simulated annealing and (ii) Langevin MCMC.

6.2.1 BACKGROUND

It is important to note that a sampler can be used as an optimiser in the following context. Consider the target density

$$p_\star^\beta(x) \propto \exp(-\beta f(x)),$$

where $\beta > 0$ is a parameter. It is known in the literature that the density $p_\star^\beta(x)$ concentrates around the minima of f as $\beta \to \infty$ (Hwang, 1980). This connection between probability distributions and optimisation spurred the development of sampling methods for optimisation. In what follows, we describe two methods that exploit this connection.

6.2.2 SIMULATED ANNEALING

Consider now a *sequence* of target distributions defined as

$$p_\star^{\beta_t}(x) \propto \exp(-\beta_t f(x)),$$

where $\beta_t > 0$ is a sequence of increasing parameters. This algorithm *anneals* the target distribution so that $p_\star^{\beta_t}(x)$ becomes concentrated around the minima of f. Typically an increasing sequence of β_t's are chosen, so that as $t \to \infty$, $\beta_t \to \infty$, making the target distribution concentrated on the minima. Drawing from previous chapter's MH algorithm, we can easily develop the simulated annealing (SA) algorithm as seen in Algorithm 13.

One can see that this simulated annealing method takes a special and intuitive case for optimisation. If we look at the acceptance ratio

$$\frac{\bar{p}_\star^{\beta_t}(X')}{\bar{p}_\star^{\beta_t}(X_{t-1})} = \frac{\exp(-\beta_t f(X'))}{\exp(-\beta_t f(X_{t-1}))} = \exp(\beta_t(f(X_{t-1}) - f(X'))),$$

Algorithm 13 Simulated Annealing

```
1: $X_0$.
2: for $t = 1, 2, \ldots$ do
3:     $X' \sim q(x|X_{t-1})$ (symmetric proposal, e.g., random walk)
4:     Set $\beta_t$ (e.g. $\beta_t = \sqrt{1+t}$)
5:     Accept $X_t$ with probability
           $$\min\left\{1, \frac{\bar{p}_\star^{\beta_t}(X')}{\bar{p}_\star^{\beta_t}(X_{t-1})}\right\}.$$
6:     Otherwise set $X_t = X_{t-1}$.
7: end for
```

we can see that the acceptance ratio is a function of the difference in the objective function values. If $f(X') \leq f(X_{t-1})$, this proposal will take higher values, possibly bigger than 1 depends on the improvement. If, however, $f(X') \geq f(X_{t-1})$, the acceptance ratio will be small as it should be. Scheduling of $(\beta_t)_{t\geq 0}$ is a design problem that depends on the specific cost function under consideration.

Example 6.1. Consider the following challenging cost function

$$f(x) = -\left(\cos(50x) + \sin(20x)\right)^2 \exp(-5x^2), \qquad x \in [-1, 1]. \tag{6.4}$$

This is a function with multiple local minima and is nonconvex. The function has one global minimum and we aim at finding it. Describe the simulated annealing method with a schedule $\beta_t = \sqrt{1+t}$.

Solution. We see that $\beta_t \to \infty$ as t grows. We use a random walk proposal with a standard deviation of $\sigma_q = 0.1$. We implement this on the log domain. We initialise $X_0 \sim \text{Unif}(-1, 1)$. The algorithm is implemented as, given X_{t-1}

- $X' \sim q(x|X_{t-1}) = \mathcal{N}(x; X_{t-1}, \sigma_q^2)$
- Sample $u \sim \text{Unif}(0, 1)$
- Accept if

$$\log(u) < \beta_t(f(X_{t-1}) - f(X'))$$

- Otherwise set $X_t = X_{t-1}$.

The result can be seen from Figure 6.1. We can see that the algorithm is able to find the global minima.

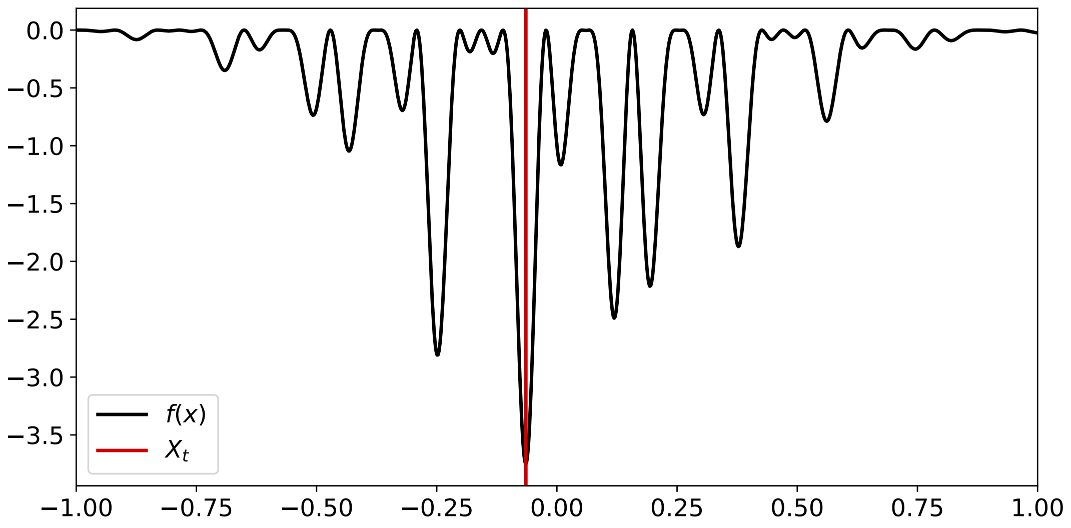


Figure 6.1: Simulated annealing for the function in Eq. 6.4. The red line shows the final estimate of SA algorithm.

6.2.3 LANGEVIN MCMC FOR OPTIMISATION

In this section, we will describe how the Langevin MCMC based methods can be designed and used for optimisation. This class of methods are very popular in machine learning literature, both as a global optimiser and as a model of stochastic gradient descent (SGD) which is a popular optimisation algorithm in machine learning.

The use Langevin MCMC for optimisation is motivated by the fact that the following Langevin SDE

$$\mathrm{d}X_t = -\nabla f(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}W_t, \tag{6.5}$$

leaves the target distribution $p_\star(x) \propto \exp(-\beta f(x))$ invariant. We have already seen that this sort of target concentrates around the minima of f as $\beta \to \infty$, therefore we can use this as an optimisation algorithm. A simple Euler discretisation of this SDE gives

$$X_{n+1} = X_n - \gamma \nabla f(X_n) + \sqrt{\frac{2\gamma}{\beta}} W_n, \tag{6.6}$$

where $W_n \sim \mathcal{N}(0, I)$ and γ is the step size. This algorithm can be shown to converge to the global minima of f arbitrarily close for large β and small γ (Zhang et al., 2023).

The family of MCMC methods can be used for optimisation as well. We will showcase one example.

Example 6.2 (ULA for Optimisation). Assume that we try to solve the following problem:

$$\arg\min_{x\in\mathbb{R}} f(x),$$

where $f(x) = \frac{1}{2\sigma^2}(x-\mu)^2$ and μ and σ are known. Of course, (i) we do not really need the scaling factor $\frac{1}{2\sigma^2}$ and (ii) we can simply solve this problem exactly (no surprise, the minimiser is μ). For the sake of this example, design a Langevin MCMC method to optimise this cost function and show its properties.

Solution. We can convert the optimisation problem into a sampling problem by defining the target density as

$$p_\star(x) \propto \exp(-\beta f(x)).$$

One can argue that we will not know the normalising constant – which is exactly true. In general, to solve the problem $\min_{x\in\mathbb{R}^d} f(x)$, one constructs a target density $p_\star(x) \propto e^{-\beta f(x)}$. To sample from this density, we resort to a modified version

$$X_{n+1} = X_n + \gamma \nabla \log p_\star(X_n) + \sqrt{\frac{2\gamma}{\beta}} V_n,$$

where β is a parameter that is called the inverse temperature. We can see following the same logic in Example 5.14 that, we have a target distribution

$$p_\star^\beta(x) = \mathcal{N}\left(x; \mu, \frac{2\sigma^4}{\beta(2\sigma^2 - \gamma)}\right).$$

One can see that as $\beta \to \infty$, we have $p_\star^\beta(x) \to \delta_\mu(x)$, i.e., the target distribution is a Dirac delta at μ. This is an example of a more general result where sampling from $p_\star^\beta(x) \propto \exp(-\beta f(x))$ (as it is what the sampler is doing) leads to distributions that concentrate on the minima of $f(x)$ as $\beta \to \infty$.

In our case, for large β, the distribution would be concentrated around μ, that is maximum. Therefore, samples from this distribution would be very close to μ. The error can be verified and quantified in a number of challenging and nonconvex settings (Zhang et al., 2023).

6.3 MAXIMUM MARGINAL LIKELIHOOD ESTIMATION

Another application where sampling methods come in handy is *model learning* problems. So far, we have covered probabilistic inference problems where we typically introduced a prior $p(x)$ and a likelihood $p(y|x)$. Our sampling algorithms usually handled *unnormalised* likelihoods – therefore, we could just set $\bar{p}_\star(x) = p(y|x)p(x)$ and run the sampling algorithm (say MCMC). However, in many cases, we are interested in learning the parameters of the model. In fact, in fields like generative models (e.g. energy-based models), the density of interest could be completely parameterised with a flexible neural network (as we will see in the next section). In this section, we will see such methods and general principles to learn parametric probabilistic models from data and the use of stochastic simulation methods for this task.

Let us start, again, from the simple Bayesian inference scenario. Assume that, we have a prior $p_{\theta_1}(x)$ and a likelihood $p_{\theta_2}(y|x)$, where θ_1 and θ_2 are the parameters of the prior and the likelihood, respectively. Let $\theta = (\theta_1, \theta_2)$ and we will typically from now on write the joint density $p_\theta(x, y)$ to leave the dependence on θ explicit. Assume that y is observed and fixed data. Then, the maximum marginal likelihood estimation (MMLE) problem is defined as

$$\theta^\star = \arg\max_\theta \log p_\theta(y) = \arg\max_\theta \log \int p_\theta(x, y)\mathrm{d}x.$$

We note that, despite its conceptual similarity, this is a very different problem than a pure maximum likelihood estimation (MLE) problem. In MLE, we would have $\log p_\theta(y)$ available and we could treat this as a simple optimisation problem. In contrast, the MMLE problem above involves an intractable integral. This is suitable for the earlier methods we introduced in this course, as in this case, we need to solve the integration and optimisation problem together.

Example 6.3. Consider the following model

$$\begin{aligned} p_\theta(x) &= \mathcal{N}(x; \theta, \sigma_0^2), \\ p(y|x) &= \mathcal{N}(y; ax, \sigma^2). \end{aligned}$$

Assume that σ_0^2, σ^2 are known and θ is unknown. Given the observation y, derive the MMLE estimate for this model.

Solution. We can write the marginal likelihood as

$$\begin{aligned} p_\theta(y) &= \int p(y|x)p_\theta(x)\mathrm{d}x, \\ &= \mathcal{N}(y; a\theta, a^2\sigma_0^2 + \sigma^2). \end{aligned}$$

Computing the $\log p_\theta(y)$, we have

$$\log p_\theta(y) = -\frac{1}{2}\log(2\pi(a^2\sigma_0^2 + \sigma^2)) - \frac{(y - a\theta)^2}{2(a^2\sigma_0^2 + \sigma^2)}.$$

Compute now

$$\frac{\partial \log p_\theta(y)}{\partial \theta} = \frac{y - a\theta}{a\sigma_0^2 + \sigma^2}.$$

By setting this to zero, we obtain $\theta^\star = y/a$.

6.3.1 MONTE CARLO GRADIENT ESTIMATION FOR MMLE

As we have seen many times before, however, exact marginalisation like this is not possible in many cases. Of course, then, a standard approach would be optimisation. But note that, we cannot perform standard optimisation here, as $\log p_\theta(y)$ in general not available as $p_\theta(y)$ is given as an integral. Therefore, a direct computation of $\nabla \log p_\theta(y)$ is not possible. Luckily, we can prove the following result.

Proposition 6.1 (Fisher's identity)**.** *The gradient of the log marginal likelihood can be written as*

$$\nabla_\theta \log p_\theta(y) = \int p_\theta(x|y) \nabla_\theta \log p_\theta(x, y),$$

in other words, this gradient can be written as an expectation:

$$\nabla_\theta \log p_\theta(y) = \mathbb{E}_{p_\theta(x|y)}[\nabla_\theta \log p_\theta(x, y)].$$

Proof. The proof of this result is simple. We first use the fact that

$$\nabla_\theta \log p_\theta(y) = \frac{\nabla_\theta p_\theta(y)}{p_\theta(y)}.$$

Then, we can substitute $p_\theta(y) = \int p_\theta(x, y) dx$ and exchange differentiation and integration

$$\begin{aligned}
\nabla_\theta \log p_\theta(y) &= \frac{\nabla_\theta \int p_\theta(x, y) dx}{\int p_\theta(x, y) dx}, \\
&= \frac{\int \nabla_\theta p_\theta(x, y) dx}{\int p_\theta(x, y) dx}, \\
&= \frac{\int \nabla_\theta \log p_\theta(x, y) p_\theta(x, y) dx}{\int p_\theta(x, y) dx}, \\
&= \int \nabla_\theta \log p_\theta(x, y) p_\theta(x|y) dx,
\end{aligned}$$

where we used the fact that $p_\theta(x|y) = \frac{p_\theta(x,y)}{p_\theta(y)}$ in the last line. □

This shows that in order to implement the gradient descent iteration

$$\theta_{n+1} = \theta_n + \gamma \nabla_\theta \log p_{\theta_n}(y),$$

for the MMLE, one can use the ideas from previous chapters. In particular, for the case when $p_\theta(x|y)$ is tractable, this leads to a simple algorithm as the gradient can be approximated as

$$\nabla_\theta \log p_\theta(y) = \mathbb{E}_{p_\theta(x|y)}[\nabla_\theta \log p_\theta(x, y)] \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log p_\theta(x_k, y),$$

where $x_k \sim p_\theta(x|y)$ for $k = 1, \ldots, K$. This is a simple (and unbiased) Monte Carlo approximation of the gradient.

Oftentimes, we will not have $p_\theta(x|y)$ analytically available, thus we have to resort to Monte Carlo methods we have seen within this course to approximate this gradient. A good overview for the use of these Monte Carlo estimation methods in machine learning is given in Mohamed et al. (2020).

7
SEQUENTIAL MONTE CARLO

In this chapter, we introduce sequential Monte Carlo (SMC) methods. These methods are used to approximate a sequence of target distributions rather than just a single, fixed target. This can have a number of applications, including filtering and smoothing in state space models. We will briefly introduce state-space models, SMC and its connection to importance sampling, and application of SMC to filtering in state-space models, which is also called particle filtering.

7.1 INTRODUCTION

In this section, we depart from our standard setting where we have a single, fixed target $p_\star(x)$. In many problems in the real world, the target distributions are *evolving* over time. For example, consider the example of tracking a target, a straightforward extension of the source localisation problem we discussed in Example 5.8. Instead of a fixed target and fixed measurements, we could have easily the case of a moving target and fixed/moving sensors. In this case, we could recompute our posterior every time we get a new measurement, however, this could become very prohibitive (imagine every time you get new data, you need to run a new MCMC chain!). However, the applications of this framework is not limited to simple localisation examples, it broadly generalises to many dynamical systems. A few examples are volatility estimation in financial time series, robotics (tracking and control of moving arms), infectious disease modelling (tracking the spread of a disease), and many more. The idea of evolving sequence of distributions can also be used to target static problems, as we have seen in the example of simulated annealing.

Our running example in this section will be *state-space* models. A good example within this setting will be the target tracking example which summarises the notion of a *hidden state* and a sequence of *observations*. However, it is crucial to observe that the example generalises to any situation where a hidden, evolving quantity to be estimated (out in the wild) and a stream of data is received to update our latest belief on the state of the object.

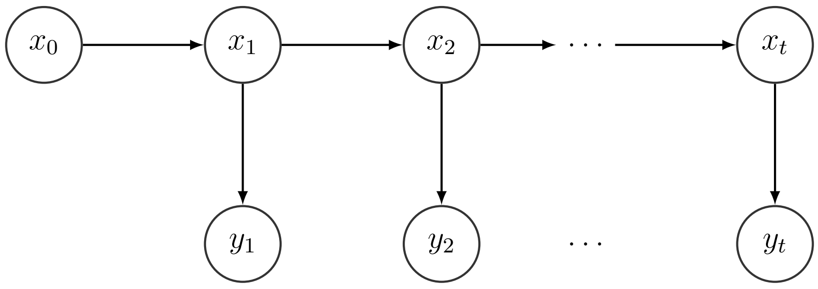


Figure 7.1: The conditional independence structure of a state-space model.

7.2 STATE-SPACE MODELS

Consider a Markov process $(X_t)_{t\geq 0}$ defined on the measurable space X with $\mathsf{X} \subset \mathbb{R}^{d_x}$. This process denotes the signal of interest, e.g., the state of an object, the velocity field of a partial differential equation (PDE), hence we call it *the signal process*. Similarly, we define another sequence of random variables $(y_t)_{t\geq 1}$, defined on $\mathsf{Y} \subset \mathbb{R}^{d_y}$, to denote our observation sequence, or *the observation process*. This sequence denotes the observed data coming from the signal process and it can typically consist of noisy sensor measurements or noisy observations. Based on this two sequences, we can define a *model* which we name as a state-space model. This is typically by three distributions (Doucet et al., 2000)

$$\begin{aligned} X_0 &\sim \mu(x_0) \\ X_t | \{X_{t-1} = x_{t-1}\} &\sim f(x_t|x_{t-1}), \\ Y_t | \{X_t = x_t\} &\sim g(y_t|x_t), \end{aligned}$$

where μ is called the prior distribution, f is a Markov transition kernel defined on X, and g as the likelihood function. For convenience, we always assume the densities exist in this document but a general construction is possible. See Fig. 7.1 for the conditional independence structure of this class of models.

An important consequence of the conditional independence structure described above is that we can write the joint distribution as

$$\bar{\pi}_t(x_{0:t}, y_{1:t}) = \mu(x_0) \prod_{k=1}^{t} f(x_k|x_{k-1}) g(y_k|x_k).$$

This is going to be important. To see the analogy with standard Bayesian inference, recall that with a given prior $p(x)$ and a likelihood $p(y|x)$, we could write the joint distribution as $p(x, y) = p(y|x)p(x)$. This is a similar structure above for the joint distribution.

Recall also that we have used joint distributions as unnormalised densities throughout. This will be the same in this section where the joint distribution above will act like an unnormalised density. Another way to look at it is to consider $y_{1:t}$ completely fixed. This also makes it clear that we have distributions of the form $\pi_t(x_{0:t})$ (normalised) and $\bar{\pi}_t(x_{0:t})$ (unnormalised). We will then apply what we have covered in Chapter 4 to this setting. The final thing to note is the definition of the marginal likelihood in this setting. Let $p(y_{1:t})$ denote the marginal likelihood of the observations $y_{1:t}$, then we can write

$$p(y_{1:t}) = \int \mu(x_0) \prod_{k=1}^{t} f(x_k|x_{k-1}) g(y_k|x_k) \mathrm{d}x_{0:t}. \tag{7.1}$$

7.2.1 THE FILTERING PROBLEM

Given a sequence of observations $y_{1:t}$, a typical problem is to estimate the conditional distributions of the signal process $(X_t)_{t\geq 0}$ given the observed data. We denote this distribution with $\pi_t(x_t|y_{1:t})$ which is called *the filtering distribution*. The problem of sequentially updating the sequence of filtering distributions $(\pi_t(x_t|y_{1:t}))_{t\geq 1}$ is called *the filtering problem*.

To introduce the idea intuitively, consider the scenario of tracking a target. We denote the states of the target with $(x_t)_{t\geq 0}$ which may include positions and velocities. We assume that the target moves in space w.r.t. f, i.e., the transition model of the target is given by $f(x_t|x_{t-1})$. Observations may consist of the locations of the target on $\mathbb{R}^2$ or power measurements with associated sensors (which may result in high-dimensional observations). At each time t, we receive a measurement vector y_t conditional on the true state of the system x_t. The likelihood of each observation is assumed to follow $g(y_t|x_t)$.

We now provide a simple recursion to demonstrate one possible solution to the filtering problem. Assume that we are given the distribution at time $t-1$ (to define our sequential recursion) and would like to incorporate a recent observation y_t. One way to do so is to first perform *prediction*

$$\xi_t(x_t|y_{1:t-1}) = \int f(x_t|x_{t-1})\pi_{t-1}(x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1}, \tag{7.2}$$

and obtain the predictive measure and then perform *update*

$$\pi_t(x_t|y_{1:t}) = \xi_t(x_t|y_{1:t-1})\frac{g(y_t|x_t)}{p(y_t|y_{1:t-1})}, \tag{7.3}$$

where $p(y_t|y_{1:t-1}) = \int \xi_t(x_t|y_{1:t-1})g(y_t|x_t)\mathrm{d}x_t$ is the incremental marginal likelihood.

Remark 7.1. We remark that the celebrated *Kalman filter* (Kalman, 1960) exactly implements recursions (7.2)–(7.3) in the case of

$$\begin{aligned}
\mu(x_0) &= \mathcal{N}(x_0; \mu_0, \Sigma_0), \\
f(x_t|x_{t-1}) &= \mathcal{N}(x_t; Ax_{t-1}, Q), \\
g(y_t|x_t) &= \mathcal{N}(y_t; Cx_t, R).
\end{aligned}$$

For this Gaussian system, computing the integral (7.2) and the update (7.3) is analytically tractable, which results in Kalman filtering recursions of the mean and the covariance of the filtering distribution $\pi_t(x_t|y_{1:t})$. We skip the update rules of the Kalman filter, as our main aim is to focus on sequential Monte Carlo in this course.

Finally, we can move on to show how to update joint filtering distribution of the states $x_{0:t}$.

To see this, note the recursion

$$\begin{aligned}\pi_t(x_{0:t}|y_{1:t}) &= \frac{\bar{\pi}_t(x_{0:t}, y_{1:t})}{p(y_{1:t})} \\ &= \frac{\bar{\pi}_{t-1}(x_{0:t-1}, y_{1:t-1})}{p(y_{1:t-1})} \frac{f(x_t|x_{t-1})g(y_t|x_t)}{p(y_t|y_{1:t-1})} \\ &= \pi_{t-1}(x_{0:t-1}|y_{1:t-1}) \frac{f(x_t|x_{t-1})g(y_t|x_t)}{p(y_t|y_{1:t-1})}.\end{aligned}$$

This recursion will be behind the sequential Monte Carlo method we use for filtering in the next sections.

7.3 SEQUENTIAL MONTE CARLO FOR FILTERING

7.3.1 IMPORTANCE SAMPLING: RECAP

Before we introduce the sequential Monte Carlo sampling for filtering, we recall the basic importance sampling idea and its terminology accounting for the change of notation within this chapter. Assume that we aim at estimating expectations of a given density π, i.e., we would like to compute

$$\mathbb{E}_\pi[\varphi(X)] = \int \varphi(x)\pi(x)\mathrm{d}x.$$

We also assume that sampling from this density is not possible and we can only evaluate the *unnormalised* density $\bar{\pi}(x)$. One way to estimate this expectation is to sample from a proposal measure q and rewrite the integral as

$$\begin{aligned}\mathbb{E}_\pi[\varphi(X)] &= \int \varphi(x)\pi(x)\mathrm{d}x, \\ &= \frac{\int \varphi(x)\frac{\bar{\pi}(x)}{q(x)}q(x)\mathrm{d}x}{\int \frac{\bar{\pi}(x)}{q(x)}q(x)\mathrm{d}x}, \\ &\approx \frac{\frac{1}{N}\sum_{i=1}^N \varphi(x^{(i)})\frac{\bar{\pi}(x^{(i)})}{q(x^{(i)})}}{\frac{1}{N}\sum_{i=1}^N \frac{\bar{\pi}(x^{(i)})}{q(x^{(i)})}}, \qquad x^{(i)} \sim q, \quad i = 1, \ldots, N. \end{aligned} \tag{7.4}$$

Let us now introduce the unnormalised weight function[1]

$$W(x) = \frac{\bar{\pi}(x)}{q(x)}. \tag{7.5}$$

With this, the Eq. (7.4) becomes

$$\begin{aligned}\hat{\varphi}_{\mathrm{IS}}^N &= \frac{\frac{1}{N}\sum_{i=1}^N \varphi(x^{(i)})W(x^{(i)})}{\frac{1}{N}\sum_{i=1}^N W(x^{(i)})}, \qquad x^{(i)} \sim q, \quad i = 1, \ldots, N, \\ &= \frac{\sum_{i=1}^N \varphi(x^{(i)})\mathsf{W}^{(i)}}{\sum_{i=1}^N \mathsf{W}^{(i)}}, \qquad x^{(i)} \sim q, \quad i = 1, \ldots, N,\end{aligned}$$

[1]More technically, these weights are the evaluations of the Radon-Nikodym derivative $W(x) = \frac{\mathrm{d}\gamma}{\mathrm{d}q}(x)$ (which, in this case, is just a ratio as we assume absolute continuity implicitly).

where $\mathsf{W}^{(i)} = W(x^{(i)})$ are called *the unnormalised weights*. Finally, we can obtain the estimator in a more convenient form,

$$\hat{\varphi}_{\text{IS}}^N = \sum_{i=1}^N \mathsf{w}^{(i)} \varphi(x^{(i)}),$$

by introducing the *normalised importance weights*

$$\mathsf{w}^{(i)} = \frac{\mathsf{W}^{(i)}}{\sum_{i=1}^N \mathsf{W}^{(i)}}, \tag{7.6}$$

for $i = 1, \ldots, N$. We note that the particle approximation of π in this case is given as

$$\pi^N(\mathrm{d}x) = \sum_{i=1}^N \mathsf{w}^{(i)} \delta_{x^{(i)}}(\mathrm{d}x). \tag{7.7}$$

In the following section, we will derive the importance sampler aiming at building particle approximations of $\pi_t(x_{0:t}|y_{1:t})$ for a state-space model.

7.3.2 IMPORTANCE SAMPLING FOR STATE-SPACE MODELS: THE EMERGENCE OF THE GENERAL PARTICLE FILTER

In this section, we simply derive an importance sampler for the joint filtering distribution $\pi_t(x_{0:t}|y_{1:t})$. We will see in the process that the particle filter is a special case of this conceptually simple importance sampler (defined just in many variables instead of one) and the infamous bootstrap particle filter is a further simplified case.

Let us assume that, in order to build an estimator of $\pi_t(x_{0:t}|y_{1:t})$, we have a proposal distribution over the entire path space $x_{0:t}$ denoted $q(x_{0:t})$. Note that, we also denote the unnormalised distribution of $x_{0:t}$ as $\bar{\pi}(x_{0:t}, y_{1:t})$ which is given as

$$\bar{\pi}(x_{0:t}, y_{1:t}) = \mu(x_0) \prod_{k=1}^t f(x_k|x_{k-1}) g(y_k|x_k). \tag{7.8}$$

This simply the joint distribution of all variables $(x_{0:t}, y_{1:t})$. Just as in the regular importance sampling case in eq. (7.5), we write

$$W_{0:t}(x_{0:t}) = \frac{\bar{\pi}(x_{0:t}, y_{1:t})}{q(x_{0:t})}.$$

Obviously, given samples from the proposal $x_{0:t}^{(i)} \sim q(x_{0:t})$, one can easily build the same weighted measure as in (7.7) on the path space by evaluating the weight $\mathsf{W}_{0:t}^{(i)} = W_{0:t}(x_{0:t}^{(i)})$ for $i = 1, \ldots, N$ and building a particle approximation

$$\pi^N(\mathrm{d}x_{0:t}) = \sum_{i=1}^N \mathsf{w}_{0:t}^{(i)} \delta_{x_{0:t}^{(i)}}(\mathrm{d}x_{0:t}).$$

where

$$\mathsf{w}_{0:t}^{(i)} = \frac{\mathsf{W}_{0:t}^{(i)}}{\sum_{i=1}^N \mathsf{W}_{0:t}^{(i)}}.$$

However, this would be an undesirable scheme: We would need to store all variables in memory which is infeasible as t grows. Furthermore, with the arrival of a new observation y_{t+1}, this would have to be re-done, as this importance sampling procedure does not take into account the dynamic properties of the SSM. Therefore, implementing this sampler to build estimators sequentially is out of question.

Fortunately, we can design our proposal in certain ways so that this process can be done sequentially, starting from 0 to t. Furthermore, this would allow us to run the filter *online* and incorporate new observations. The clever choices of the proposal here lead to a variety of different *particle filters* as we shall see next. Let us consider a decomposition of the proposal

$$q(x_{0:t}) = q(x_0) \prod_{k=1}^{t} q(x_k | x_{1:k-1}).$$

Note that, based on this, we can build a recursion for the function $W(x_{0:t})$ by writing

$$\begin{aligned}
W_{0:t}(x_{0:t}) &= \frac{\bar{\pi}(x_{0:t}, y_{1:t})}{q(x_{0:t})}, \\
&= \frac{\bar{\pi}(x_{0:t-1}, y_{1:t-1})}{q(x_{0:t-1})} \frac{f(x_t|x_{t-1}) g(y_t|x_t)}{q(x_t|x_{0:t-1})}, \\
&= W_{0:t-1}(x_{0:t-1}) \frac{f(x_t|x_{t-1}) g(y_t|x_t)}{q(x_t|x_{0:t-1})}, \\
&= W_{0:t-1}(x_{0:t-1}) W_t(x_{0:t}). \qquad (7.9)
\end{aligned}$$

That is, under this scenario, the weights can be computed *recursively* – given the weights of time $t-1$, one can evaluate $W_{0:t}(x_{0:t})$ and update the weights. However, this would not solve the infeasibility problem mentioned earlier, as the cost of evaluating using the whole path of samples is still out of question. Finally, to remedy this, we can further simplify our proposal

$$q(x_{0:t}) = q(x_0) \prod_{k=1}^{t} q(x_k | x_{k-1}).$$

by removing dependence to the past, essentially choosing a Markov process as a proposal. This allows us to obtain purely recursive weight computation

$$\begin{aligned}
W_{0:t}(x_{0:t}) &= \frac{\bar{\pi}(x_{0:t}, y_{1:t})}{q(x_{0:t})}, \qquad (7.10) \\
&= \frac{\bar{\pi}(x_{0:t-1}, y_{1:t-1})}{q(x_{0:t-1})} \frac{f(x_t|x_{t-1}) g(y_t|x_t)}{q(x_t|x_{t-1})}, \qquad (7.11) \\
&= W_{0:t-1}(x_{0:t-1}) \frac{f(x_t|x_{t-1}) g(y_t|x_t)}{q(x_t|x_{t-1})}, \qquad (7.12) \\
&= W_{0:t-1}(x_{0:t-1}) W_t(x_t, x_{t-1}), \qquad (7.13)
\end{aligned}$$

using only the samples from time $t-1$ and time t. The advantage of this scheme is explicit in the notation: Note that the final weight function W_t only depends on (x_t, x_{t-1}), but not the whole past as in (7.9). The function $W_t(x_t, x_{t-1})$ is called the incremental weight function.

7.3.3 SEQUENTIAL IMPORTANCE SAMPLING

We can now see how the one-step update of this sampler works given a new observation. Assume that we have computed the unnormalised weights $\mathsf{W}_{1:t-1}^{(i)} = W(x_{0:t-1}^{(i)})$ recursively and obtained samples $x_{0:t-1}^{(i)}$. As we mentioned earlier, we only need the last sample $x_{t-1}^{(i)}$ to obtain the weight update given in (7.13). And also note that $\mathsf{W}_{1:t-1}^{(i)}$ for $i = 1, \ldots, N$ are just numbers, they do not need the storage of previous samples. Given this, we can now sample from the Markov proposal $x_t^{(i)} \sim q(x_t|x_{t-1}^{(i)})$ and compute the weights of the path sampler at time t as

$$\mathsf{W}_{1:t}^{(i)} = \mathsf{W}_{1:t-1}^{(i)} \times \mathsf{W}_t^{(i)},$$

where

$$\mathsf{W}_t^{(i)} = \frac{f(x_t^{(i)}|x_{t-1}^{(i)})g(y_t|x_t^{(i)})}{q(x_t^{(i)}|x_{t-1}^{(i)})}.$$

What we described in other words is that, given the samples $x_{t-1}^{(i)}$, we first perform sampling step

$$x_t^{(i)} \sim q(x_t|x_{t-1})$$

and then compute

$$\mathsf{W}_t^{(i)} = \frac{f(x_t^{(i)}|x_{t-1}^{(i)})g(y_t|x_t^{(i)})}{q(x_t^{(i)}|x_{t-1}^{(i)})}.$$

and update

$$\mathsf{W}_{1:t}^{(i)} = \mathsf{W}_{1:t-1}^{(i)} \times \mathsf{W}_t^{(i)}.$$

These are unnormalised weights and we normalise them to obtain,

$$\mathsf{w}_{1:t}^{(i)} = \frac{\mathsf{W}_{1:t}^{(i)}}{\sum_{i=1}^N \mathsf{W}_{1:t}^{(i)}},$$

which finally leads to the empirical measure,

$$\pi^N(\mathrm{d}x_{0:t}) = \sum_{i=1}^N \mathsf{w}_{1:t}^{(i)} \delta_{x_{0:t}^{(i)}}(\mathrm{d}x_{0:t}).$$

The full scheme is given in Algorithm 14. This method is called sequential importance sampling (SIS). This is not very popular in the literature due to the well known *weight degeneracy* problem. We next introduce a resampling step to this method and will obtain the first particle filter in this lecture.

7.3.4 SEQUENTIAL IMPORTANCE SAMPLING WITH RESAMPLING: THE GENERAL PARTICLE FILTER

We finally describe the general particle filter by extending the above method with a resampling step employed after the weighting step. We will show in a practical session that the SIS method

Algorithm 14 Sequential Importance Sampling (SIS)

```
1: Sample $x_0^{(i)} \sim q(x_0)$ for $i = 1, \dots, N$.
2: for $t \geq 1$ do
3:     Sample: $x_t^{(i)} \sim q(x_t | x_{t-1}^{(i)})$,
4:     Compute weights:
           $$\mathsf{W}_t^{(i)} = \frac{f(x_t^{(i)} | x_{t-1}^{(i)}) g(y_t | x_t^{(i)})}{q(x_t^{(i)} | x_{t-1}^{(i)})}.$$
       and update
           $$\mathsf{W}_{1:t}^{(i)} = \mathsf{W}_{1:t-1}^{(i)} \times \mathsf{W}_t^{(i)}.$$
       Normalise weights,
           $$\mathsf{w}_{1:t}^{(i)} = \frac{\mathsf{W}_{1:t}^{(i)}}{\sum_{i=1}^N \mathsf{W}_{1:t}^{(i)}}.$$
5:     Report
           $$\pi_t^N(\mathrm{d}x_{0:t}) = \sum_{i=1}^N \mathsf{w}_{1:t}^{(i)} \delta_{x_{0:t}^{(i)}}(\mathrm{d}x_{0:t}).$$
6: end for
```

without resampling easily degenerates, i.e., after some time, only a single weight approximates to 1 and others to 0, rendering the method a point estimate. To keep the particle diversity, a resampling method is introduced in between weighting and sampling steps. This step does not introduce a systematic bias, although, it adds additional terms to the overall L_p error.

With the additional resampling step, the sequential importance sampling with resampling (SISR) takes the form given in Algorithm 15. We note that, effectively, resampling step sets $\mathsf{W}_{1:t-1}^{(i)} = 1/N$ for $i = 1, \dots, N$. Therefore, we only need to compute the last incremental weight and weight our particles with the current weight. Also, note that the resampling step does introduce extra error but does not induce bias, since moments of π_t^N does not change.

7.3.5 THE BOOTSTRAP PARTICLE FILTER

In the general particle filter, the proposal $q(x_t | x_{t-1})$ is a design choice to be made and this depends on our specific knowledge of a good proposal for a given system. For example, one can incorporate future observations into this proposal in an ad-hoc or use the proposal choices like in the auxiliary particle filter (APF).

A generic choice exists, however, that is simply setting $q(x_t | x_{t-1}) = f(x_t | x_{t-1})$, i.e., using the transition density of the SSM under consideration as a proposal. The algorithm simplifies considerably in this case and the resulting method is called the bootstrap particle filter (BPF) which is given in Alg. 16. This algorithm has multiple appealing intuitive explanations beyond

Algorithm 15 Sequential Importance Sampling with Resampling (SISR)

```
1: Sample $x_0^{(i)} \sim q(x_0)$ for $i = 1, \dots, N$.
2: for $t \geq 1$ do
3:     Sample: $\tilde{x}_t^{(i)} \sim q(x_t | x_{t-1}^{(i)})$,
4:     Compute weights:
           $$\mathsf{W}_t^{(i)} = \frac{f(\tilde{x}_t^{(i)} | x_{t-1}^{(i)}) g(y_t | \tilde{x}_t^{(i)})}{q(\tilde{x}_t^{(i)} | x_{t-1}^{(i)})}.$$
       Normalise weights,
           $$\mathsf{w}_t^{(i)} = \frac{\mathsf{W}_t^{(i)}}{\sum_{i=1}^N \mathsf{W}_t^{(i)}}.$$
5:     Report
           $$\pi_t^N(x_t)\mathrm{d}x_t = \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x_t).$$
6:     Resample:
           $$x_t^{(i)} \sim \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x_t).$$
7: end for
```

the derivation we provided based on importance sampling here. It can be most generally thought as an evolutionary method. To uncover some of this intuition, see Fig. 7.2.

To elaborate the interpretation, consider a set of particles $x_{t-1}^{(i)}$ representing the state of the system at time $t - 1$. If our state-space transition model $f(x_t | x_{t-1})$ is well-specified (that is, if the underlying system we aim at tracking does indeed move according to f), then the first intuivite step we can do to predict where the state would be at time t would be to move particles according to f, that is sampling $\tilde{x}_t^{(i)} \sim f(x_t | x_{t-1}^{(i)})$ which is the first step of the BPF. This gives us a predictive distribution which consists of $\tilde{x}_t^{(i)}$ for $i = 1, \dots, N$. The prediction step (naturally) does not require to observe the data point at y_t. Once we observe the data point y_t, we can then use this data point to evaluate a fitness measure for our particles. In other words, if a predictive particle $\tilde{x}_t^{(i)}$ is a good fit to the observation, we would expect its likelihood $g(y_t | \tilde{x}_t^{(i)})$ to be high. Otherwise, this likelihood would be low. Thus, it intuitively makes sense to use our likelihood evaluations as "weights", that is to compute a measure of fitness for each particle. That is exactly what the BPF does at the second step by computing weights using the likelihood evaluations. The final step is then to use these relative weights to *resample* – a step that is used to refine the cloud of particles we have. Simply, the resampling step removes some of the particles with low weights (that are bad fits to the observation) and regenerates the particles with high weights.

The connection to evolutionary terms are clearer within this interpretation. The sampling step in the BPF can be seen as "mutation" that introduces changes to an individual particle according

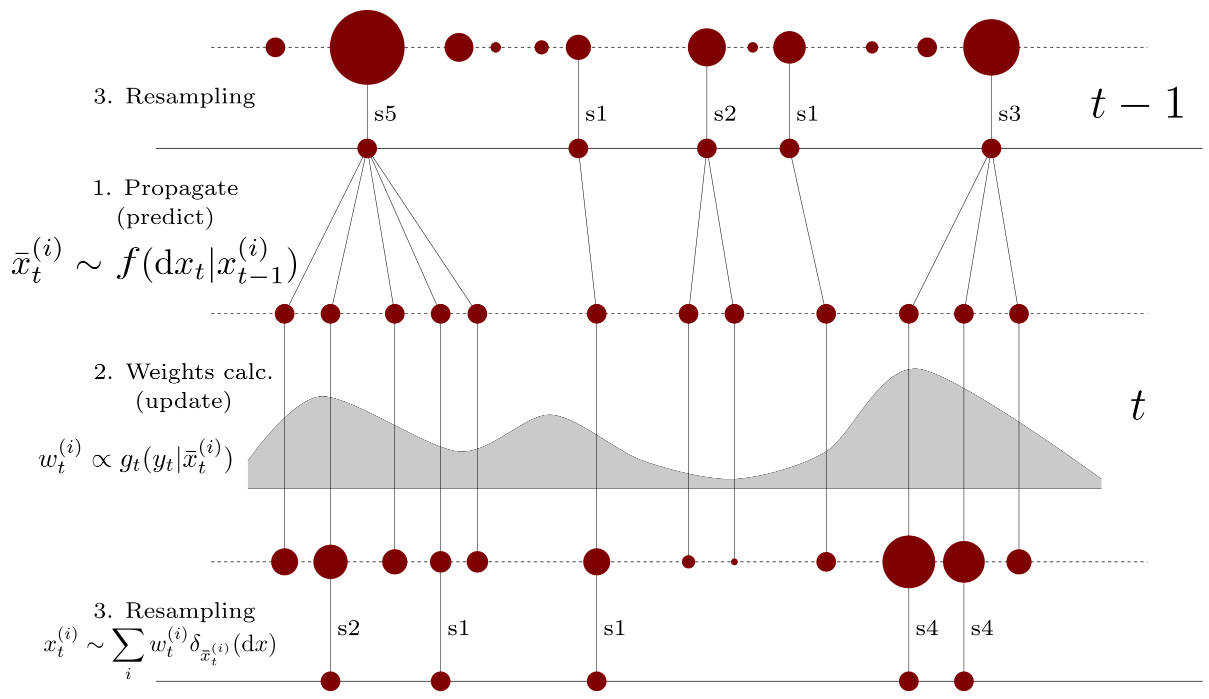

3. Resampling
s5
s1
s2
s1
s3
$t-1$
1. Propagate (predict)
$\bar{x}_t^{(i)} \sim f(\mathrm{d}x_t|x_{t-1}^{(i)})$
2. Weights calc. (update)
$w_t^{(i)} \propto g_t(y_t|\bar{x}_t^{(i)})$
t
3. Resampling
$x_t^{(i)} \sim \sum_i w_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x)$
s2
s1
s1
s4
s4

Figure 7.2: Intuitive model of BPF (Figure courtesy Victor Elvira).

to some mutation mechanism (in our case, the dynamics). Then, weighting and resampling correspond to "selection" step, where individual particles are evaluated w.r.t. a fitness measure coming from the environment (defined by an observation) and individuals are reproduced in a random manner w.r.t. their fitness.

7.3.6 PRACTICAL IMPLEMENTATION OF THE BPF

Of course, the BPF can become numerically unstable if the weights are too small or too large. This is in line with the theme we have seen about computing small or large numbers (especially involving normalisation) throughout this course. To avoid a problem here, too, we need to perform the comptutations in the log-domain. For example, after sampling from the proposal $\tilde{x}_t^{(i)} \sim f(x_t|x_{t-1})$, we can compute the log-weights as

$$\log \mathsf{W}_t^{(i)} = \log g(y_t|\tilde{x}_t^{(i)})$$

We can then compute the normalised weights $\mathsf{w}_t^{(i)}$ using the trick introduced in Sec. 4.6. This will ensure the stable computation of weights and prevent instability.

7.3.7 MARGINAL LIKELIHOOD COMPUTATION WITH BPF

The BPF can naturally be used to compute $p(y_{1:t})$ sequentially. In order to see that we have the decomposition

$$p(y_{1:t}) = p(y_{1:t-1})p(y_t|y_{1:t-1}).$$

Algorithm 16 Bootstrap particle filter (BPF)

```
1: Sample $x_0^{(i)} \sim q(x_0)$ for $i = 1, \ldots, N$.
2: for $t \geq 1$ do
3:     Sample: $\tilde{x}_t^{(i)} \sim f(x_t | x_{t-1}^{(i)})$,
4:     Compute weights:
           $$\mathsf{W}_t^{(i)} = g(y_t | \tilde{x}_t^{(i)}).$$
       Normalise weights,
           $$\mathsf{w}_t^{(i)} = \frac{\mathsf{W}_t^{(i)}}{\sum_{i=1}^N \mathsf{W}_t^{(i)}}.$$
5:     Report
           $$\pi_t^N(x_t)\mathrm{d}x_t = \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x_t).$$
6:     Resample:
           $$x_t^{(i)} \sim \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x_t).$$
7: end for
```

In order to recursively compute $p(y_{1:t})$, we need to estimate $p(y_t|y_{1:t-1})$ using the BPF. This is possible via the use of the predictive density

$$p(y_t|y_{1:t-1}) = \int g(y_t|x_t)\xi(x_t|y_{1:t-1})\mathrm{d}x_t. \tag{7.14}$$

We note that the predictive density can be built using the particles after sampling (as explained above). In short, we can build the predictive density

$$\xi^N(\mathrm{d}x_t|y_{1:t-1}) = \frac{1}{N}\sum_{i=1}^N \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x_t),$$

if the resampling is done at every iteration (otherwise, the weights from the previous iteration has to be used). Plugging this back into Eq. (7.14), we arrive at the empirical estimate of the predictive density

$$p^N(y_t|y_{1:t-1}) = \frac{1}{N}\sum_{i=1}^N g(y_t|\tilde{x}_t^{(i)}).$$

Finally, the full marginal likelihood can be computed as

$$p^N(y_{1:t}) = \prod_{k=1}^t p^N(y_k|y_{1:k-1}). \tag{7.15}$$

Incredibly, this estimate is unbiased (see Lemma 2 of (Crisan et al., 2018) for a proof). This is incredibly useful for many things, including model selection.

Note that, computation of the marginal likelihood requires numerical care, as the product of likelihoods can easily underflow. To avoid this, we can use `logsumexp` trick to compute the log-marginal likelihood as

$$\begin{aligned}\log p^N(y_{1:t}) &= \log \sum_{i=1}^N \exp\left(\log g(y_t|\tilde{x}_t^{(i)})\right) - \log N, \\ &= \log \sum_{i=1}^N \exp\left(\log \mathsf{W}_t^{(i)})\right) - \log N\end{aligned}$$

7.4 PARTICLE MARKOV CHAIN MONTE CARLO

The unbiasedness property mentioned above of the marginal likelihood estimator can also be used to build MCMC algorithms for parameter inference in these models. We will look at one such method here, called particle MCMC (Andrieu et al., 2010). The idea relies on the fact that one can use "unbiased" estimates of various terms in the acceptance ratio and this would still leave the original target invariant (Andrieu and Roberts, 2009).

To formalise, the estimator in Eq. (7.15) satisfies

$$p(y_{1:T}) = \mathbb{E}\left[p^N(y_{1:T})\right].$$

Now let us see how we can use this for parameter inference in state-space models. Assume that we have a parameter θ that we would like to infer in the SSM. We further have a prior distribution $p(\theta)$. The existence of parameter means that the transition density $f_\theta(x_t|x_{t-1})$ and the likelihood $g_\theta(y_t|x_t)$ are now parameterised by θ. Let us for now fix θ and write the marginal likelihood of the SSM as given in (7.1) for fixed θ as

$$p(y_{1:T}|\theta) = \int \bar{\pi}_\theta(x_{0:T}, y_{1:T}) \mathrm{d}x_{0:T}.$$

It is obvious that for every θ, we can estimate this using the BPF and the estimator in (7.15). To be completely explicit, for every θ, we can get estimators $p^N(y_{1:T}|\theta)$ of the marginal likelihood such that

$$p(y_{1:T}|\theta) = \mathbb{E}\left[p^N(y_{1:T}|\theta)\right].$$

Assume next for simplicity that we have the knowledge of $p(y_{1:T}|\theta)$ and $p(\theta)$ and want to design a Metropolis-Hastings sampler for the posterior $p(\theta|y_{1:T})$. As we have seen before, we can do this via choosing a proposal $q(\theta'|\theta)$ and writing the Metropolis-Hastings method as

- Sample $\theta' \sim q(\theta'|\theta_n)$.
- Compute the acceptance ratio

$$\alpha(\theta_n, \theta') = \min\left\{1, \frac{p(y_{1:T}|\theta')p(\theta')q(\theta_n|\theta')}{p(y_{1:T}|\theta_n)p(\theta_n)q(\theta'|\theta_n)}\right\}.$$

- Accept θ' with probability $\alpha(\theta_n, \theta')$ and set $\theta_{n+1} = \theta'$.

- Otherwise, reject θ' and set $\theta_{n+1} = \theta_n$.

We have everything we want to sample from the parameter posterior $p(\theta|y_{1:T})$ except the fact that we do not have $p(y_{1:T}|\theta)$ in closed form. However, we can use the unbiased estimator $p^N(y_{1:T}|\theta)$ instead. This would still leave the target invariant (Andrieu and Roberts, 2009; Andrieu et al., 2010). Assume now that we have run the particle filter and obtained $p^N(y_{1:T}|\theta_n)$ for the current parameter θ_n. Then the particle MH algorithm can be summarised as

- Sample $\theta' \sim q(\theta'|\theta_n)$.
- Run the particle filter with θ' and compute the estimator:

$$p^N(y_{1:T}|\theta') = \prod_{k=1}^T p^N(y_k|y_{1:k-1}, \theta').$$

- Compute the acceptance ratio

$$\alpha(\theta_n, \theta') = \min\left\{1, \frac{p^N(y_{1:T}|\theta')p(\theta')q(\theta_n|\theta')}{p^N(y_{1:T}|\theta_n)p(\theta_n)q(\theta'|\theta_n)}\right\}.$$

- Accept θ' with probability $\alpha(\theta_n, \theta')$ and set $\theta_{n+1} = \theta'$.
- Otherwise, reject θ' and set $\theta_{n+1} = \theta_n$.

Remarkably, this can work out of the box, in the sense that, it functions as a valid MCMC method and will sample from the marginal $p(\theta|y_{1:T})$ asymptotically!

7.5 EXAMPLES

We will next consider some examples of the BPF in action.

Example 7.1 (Tracking a moving target in 2D)**.** Let us assume that we would like to track a 2D moving target. The model is given by In this experiment, we consider a tracking scenario where a target is observed through sensors collecting radio signal strength (RSS) measurements contaminated with additive heavy-tailed noise. The target dynamics are described by the model,

$$x_t = Ax_{t-1} + u_t,$$

where $x_t \in \mathbb{R}^4$ denotes the target state, consisting of its position $r_t \in \mathbb{R}^2$ and its velocity, $v_t \in \mathbb{R}^2$, hence $x_t = \begin{bmatrix} r_t \\ v_t \end{bmatrix} \in \mathbb{R}^4$. Each element in the sequence $\{u_t\}_{t\in\mathbb{N}}$ is a zero-mean Gaussian random vector with covariance matrix Q. The parameters A and Q are selected as

$$A = \begin{bmatrix} I_2 & \kappa I_2 \\ 0 & 0.99 I_2 \end{bmatrix},$$

and

$$Q = \begin{bmatrix} \frac{\kappa^3}{3} I_2 & \frac{\kappa^2}{2} I_2 \\ \frac{\kappa^2}{2} I_2 & \kappa I_2 \end{bmatrix},$$

where I_2 is the 2×2 identity matrix and $\kappa = 0.04$. The observation model is given by

$$y_t = H x_t + v_t,$$

where $y_t \in \mathbb{R}^2$ is the measurement assumed to be noisy. Implement the particle filter for this problem (see the code companion).

Example 7.2 (Tracking Lorenz 63 system)**.** In this example, we consider the tracking of a discretised stochastic Lorenz 63 system given by

$$\begin{aligned} x_{1,t} &= x_{1,t-1} - \gamma \mathsf{s}(x_{1,t} - x_{2,t}) + \sqrt{\gamma}\xi_{1,t}, \\ x_{2,t} &= x_{2,t-1} + \gamma(\mathsf{r} x_{1,t} - x_{2,t} - x_{1,t}x_{3,t}) + \sqrt{\gamma}\xi_{2,t}, \\ x_{3,t} &= x_{3,t-1} + \gamma(x_{1,t}x_{2,t} - \mathsf{b} x_{3,t}) + \sqrt{\gamma}\xi_{3,t}, \end{aligned}$$

where $\gamma = 0.01$, $\mathsf{r} = 28$, $\mathsf{b} = 8/3$, $\mathsf{s} = 10$, and $\xi_{1,t}, \xi_{2,t}, \xi_{3,t} \sim \mathcal{N}(0,1)$ are independent Gaussian random variables. The observation model is given by

$$y_t = [1, 0, 0] x_t + \eta_t,$$

where $\eta_t \sim \mathcal{N}(0, \sigma_y^2)$ is a Gaussian random variable. Implement the particle filter for this problem (see the code companion).

BIBLIOGRAPHY

Agapiou, Sergios; Papaspiliopoulos, Omiros; Sanz-Alonso, Daniel; and Stuart, Andrew M. 2017. *Importance sampling: Intrinsic dimension and computational cost*. In Statistical Science, pp. 405–431. Cited on p. 82.

Akyildiz, Omer Deniz. March 2019. *Sequential and adaptive Bayesian computation for inference and optimization*. Ph.D. thesis, Universidad Carlos III de Madrid. Can be accessed from: http://akyildiz.me/works/thesis.pdf. Cited on pp. 67, 69, and 77.

Akyildiz, Ömer Deniz and Míguez, Joaquín. 2021. *Convergence rates for optimised adaptive importance samplers*. In Statistics and Computing, vol. 31, no. 2, pp. 1–17. Cited on pp. 80 and 82.

Andrieu, Christophe; Doucet, Arnaud; and Holenstein, Roman. 2010. *Particle Markov chain Monte Carlo methods*. In Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 72, no. 3, pp. 269–342. Cited on pp. 143 and 144.

Andrieu, Christophe and Roberts, Gareth O. 2009. *The pseudo-marginal approach for efficient Monte Carlo computations*. In . Cited on pp. 143 and 144.

Barber, David. 2012. *Bayesian reasoning and machine learning*. Cambridge University Press. Cited on p. 57.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer. Cited on p. 57.

Cemgil, A Taylan. 2014. *A tutorial introduction to Monte Carlo methods, Markov Chain Monte Carlo and particle filtering*. In Academic Press Library in Signal Processing, vol. 1, pp. 1065–1114. Cited on p. 107.

Crisan, Dan; Míguez, Joaquín; and Ríos-Muñoz, Gonzalo. 2018. *On the performance of parallelisation schemes for particle filtering*. In EURASIP Journal on Advances in Signal Processing, vol. 2018, pp. 1–18. Cited on p. 143.

Devroye, Luc. 1986. *Non-Uniform Random Variate Generation*. Cited on p. 12.

Douc, Randal; Moulines, Eric; Priouret, Pierre; and Soulier, Philippe. 2018. *Markov chains*. Springer. Cited on pp. 99 and 103.

Douc, Randal; Moulines, Éric; and Stoffer, David. 2013. *Nonlinear Time Series: Theory, Methods and Applications with R Examples*. Chapman & Hall. Cited on p. 99.

Doucet, Arnaud; Godsill, Simon; and Andrieu, Christophe. 2000. *On sequential Monte Carlo sampling methods for Bayesian filtering*. In Statistics and computing, vol. 10, no. 3, pp. 197–208. Cited on p. 133.

Elvira, Víctor; Martino, Luca; and Robert, Christian P. 2018. *Rethinking the effective sample size*. In International Statistical Review. Cited on p. 91.

Hwang, Chii-Ruey. 1980. *Laplace's method revisited: weak convergence of probability measures*. In The Annals of Probability, pp. 1177–1182. Cited on p. 126.

Kalman, Rudolph Emil. 1960. *A new approach to linear filtering and prediction problems*. In Journal of Fluids Engineering, vol. 82, no. 1, pp. 35–45. Cited on p. 134.

Martino, Luca; Luengo, David; and Míguez, Joaquín. 2018. *Independent random sampling methods*. Springer. Cited on pp. 13, 22, 26, and 40.

Mohamed, Shakir; Rosca, Mihaela; Figurnov, Michael; and Mnih, Andriy. 2020. *Monte carlo gradient estimation in machine learning*. In Journal of Machine Learning Research, vol. 21, no. 132, pp. 1–62. Cited on p. 131.

Murphy, Kevin P. 2007. *Conjugate Bayesian analysis of the Gaussian distribution*. In def, vol. 1, no. $2\sigma 2$, p. 16. Cited on pp. 44 and 54.

———. 2022. *Probabilistic machine learning: an introduction*. MIT press. Cited on p. 57.

Owen, Art B. 2013. *Monte Carlo theory, methods and examples*. Cited on p. 91.

Robert, Christian P and Casella, George. 2004. *Monte Carlo statistical methods*. Springer. Cited on pp. 67 and 89.

———. 2010. *Introducing Monte Carlo methods with R*, vol. 18. Springer. Cited on p. 78.

Yıldırım, Sinan. 2017. *Sabanci University IE 58001 Lecture notes: Simulation Methods for Statistical Inference*. Cited on pp. 32, 98, and 108.

Zhang, Ying; Akyildiz, Ömer Deniz; Damoulas, Theodoros; and Sabanis, Sotirios. 2023. *Nonasymptotic estimates for stochastic gradient Langevin dynamics under local conditions in nonconvex optimization*. In Applied Mathematics & Optimization, vol. 87, no. 2, p. 25. Cited on pp. 128 and 129.