Stochastic Simulation

Ö. Deniz Akyildiz

2022

Department of Mathematics
Imperial College London

PREFACE

This course is about stochastic simulation methods that underpin most of modern statistical inference, 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 an 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.

We will make use of the material in the following texts (cited within the text):

- Monte Carlo Statistical Methods, C. Robert and G. Casella (Robert and Casella, 2004)
- Independent Random Sampling Methods, L. Martino, D. Luengo, J. Miguez (Martino et al., 2018)
- Lecture Notes, Sinan Yildirim, Sabanci University (Yıldırım, 2017)

The course notes will be uploaded to Blackboard weekly – in an expanding manner. I won't, however, update past notes – so you don't have to worry about chapters expanding retrospectively.

There will be three assignments, accounting for 25% of the credit. The upload dates deadlines of these assignments are as follows:

- Assignment 1 (10%)
  - Upload: **26 Oct. 2022** – Deadline: **9 Nov. 2022**
- Assignment 2 (10%)
  - Upload: **16 Nov. 2022** – Deadline: **30 Nov. 2022**
- Assignment 3 (5%)
  - Upload: **30 Nov. 2022** – Deadline: **14 Dec. 2022**
- Final exam (75%)

Assignments and exam will have an extra question for M4R students (will be clarified before the exam). There will be a final exam which will account for 75% of the available credit. The primary course material is the lecture notes[1] and slides – however, we will also assign additional (optional) readings or complementary chapters where necessary.

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!

Ömer Deniz Akyildiz
London, 2022

[1]Cover photo by Evie Shaffer: https://www.pexels.com/photo/black-and-white-photoof-a-planet-2575278/

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. Before we introduce the necessity and importance of these tasks, we briefly set some notation up below for this section. More notation will be introduced in later sections as necessary.

1.1.1 NOTATION

In this course, the main objects we use are probability densities. We denote them with various letters, e.g., p, q. For a random variable X is drawn from p, we write $X \sim$ p. 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

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

We note the notation here $p(\varphi)$ denotes the expectation 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 $\mathsf{F}(x)$.

1.2 THE SAMPLING PROBLEM

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

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

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. Sampling problem has many motivations some of which described below.

- **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.** 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.

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.

1.3 NOTATION

This section will be the only section that can be retrospectively updated. So always download the latest version of the notes from the main page so that you can have the right notation section.

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.

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 generation. 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 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 test 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.

2.1 GENERATING UNIFORM RANDOM VARIATES

In this section, we will consider two main ways of generating pseudo-uniform random numbers. The first one is via the use of chaotic dynamical systems and the second one is the industry standard congruential linear random number generators.

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

$$x_{n+1} \equiv a x_n + b \pmod m$$

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.

For the rest of the course, we will use `np.random.uniform(0, 1)` to draw uniform random numbers as a suboptimal implementation can impact our simulation schemes.

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.

2.1.1 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 probabilities. In other words, if we draw $U \sim \text{Unif}(0, 1)$ and invert through the CDF, we will choose $x_1, x_2, \ldots$ according to their probabilities (see Fig. 2.1).

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

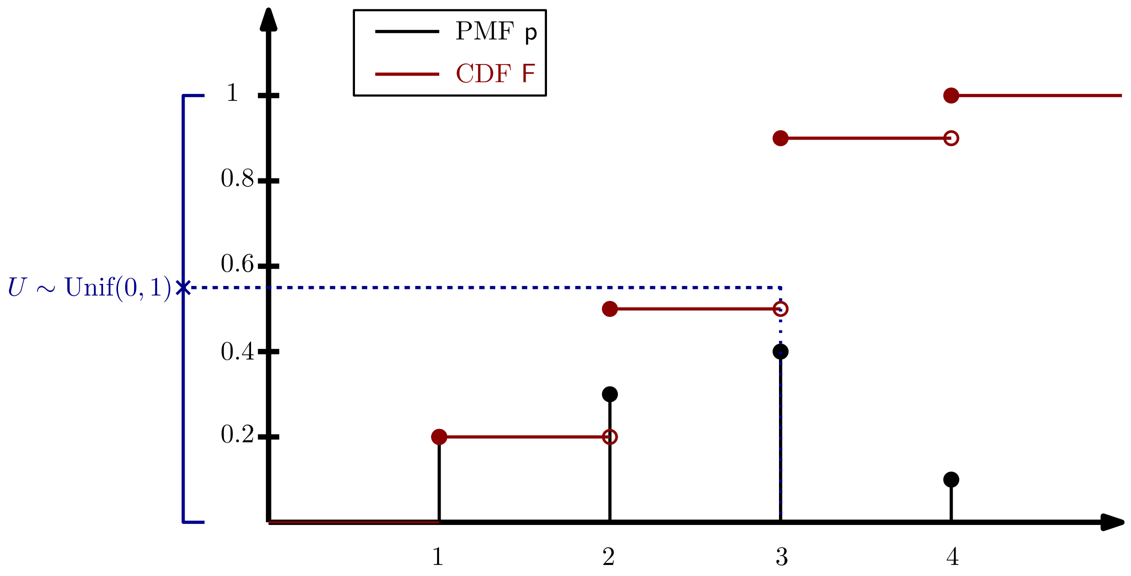

Figure 2.1: The inversion technique. The black is the probability mass function p whereas the red function is the CDF F. One can see that, drawing a uniform random number projected on the y axis ensures that we draw the samples $\{1, \dots, 4\}$ w.r.t. the probability if we follow the inverse of CDF.

Theorem 2.1. *Consider a random variable X with a CDF F_X. Then the random variable*

$$Y = F_X(X),$$

is uniformly distributed.

Proof. Consider any continuous random variable X, we define $Y = F_X(X)$. For $y \in [0, 1]$,

$$\begin{aligned} F_Y(y) &= \mathbb{P}(Y \le y) \\ &= \mathbb{P}(F_X(X) \le y) \\ &= \mathbb{P}(X \le F_X^{-1}(y)) \\ &= F_X(F_X^{-1}(y)) \\ &= y, \end{aligned}$$

which is the CDF of the standard uniform 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 $\sum_{k=1}^K w_k = 1$. In this case, the CDF is not continuous, therefore, we will use

[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,

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

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
```

- $U_i \sim u$
- $X_i = F_X^{-1}(U_i) = \min\{s_k \in \mathsf{S} : F_X(s_k) \geq u\}$.

This corresponds to something simple: Sample U_i and find the 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. 2.1).

Example 2.2. (Exponential) We would like to generate $X \sim \text{Exp}(x; \lambda) = \lambda e^{-\lambda x}$. 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

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

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

Example 2.4. (Poisson) Consider the Poisson distribution

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

The CDF is given as

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

This is similar to the discrete case.

- Sample $U \sim \mathrm{Unif}(0, 1)$
- 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.

2.1.2 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 2 Pseudocode for transformation method

```
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 $X_i = g(U_i)$
5: end for
```

algorithm Of course, 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.

Note 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

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

While the first term on the r.h.s. is similar to above, the last term 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. In our lecture, we will not need this formula for more than 2D and this case is exemplified in the examples.

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

Example 2.5. In this example, we consider Box-Müller method to sample Gaussians (Box and Müller, 1958). Let X_1 and X_2 be independent random variables where

$$X_1 \sim \text{Exp}\left(\frac{1}{2}\right),$$
$$X_2 \sim \text{Unif}(0, 2\pi),$$

Then, $Y_1 = \sqrt{X_1}\cos X_2$ and $Y_2 = \sqrt{X_1}\sin X_2$ are independent and standard normal. The following theorem provides the proof why this works.

Theorem 2.2. *(Box-Müller method) Let W, Z be independent r.v.'s respectively where*

$$X_1 \sim \text{Exp}\left(\frac{1}{2}\right),$$
$$X_2 \sim \text{Unif}(0, 2\pi),$$

Then $Y_1 = \sqrt{X_1}\cos X_2$ and $Y_2 = \sqrt{X_1}\sin X_2$ are independent and $\mathcal{N}(0,1)$-distributed.

Proof. 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)

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

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

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

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

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

Hence, the determinant is:

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

From the transformation of random variables formula

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

which concludes. □

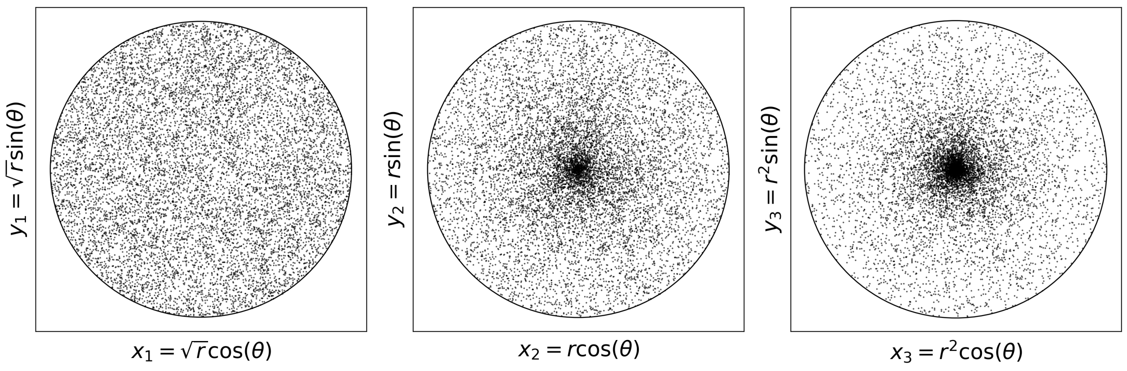

Figure 2.2: On the left, one can see the samples with the correct scaling $\sqrt{r}$. Some other intuitive formulas result in a non-uniform distribution.

Example 2.6. (Sampling uniformly on a circle) We draw

$$r \sim \text{Unif}(0,1),$$
$$\theta \sim \text{Unif}(0,2\pi).$$

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

$$x_1 = \sqrt{r}\cos\theta,$$
$$x_2 = \sqrt{r}\sin\theta.$$

We derive using the eq. (2.3)

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

We know that, since we use the same transformation as in Theorem 2.2, we have the Jacobian $|J_{g^{-1}}| = 2$ (see the proof of Theorem 2.2). We can then write

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

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 Unif$(\arctan(x_2/x_1); 0, 2\pi) = 1/2\pi$. 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. 2.2 for some examples (and alternatives discussed in the class).

We next consider the Gaussian example.

Example 2.7. A simple demonstration of the transformation of random variables formula (in 1-dimension) can be seen from a Gaussian density example. 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 $Y = \sigma X + \mu$. 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

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

by using the formula (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 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 hundreds 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).

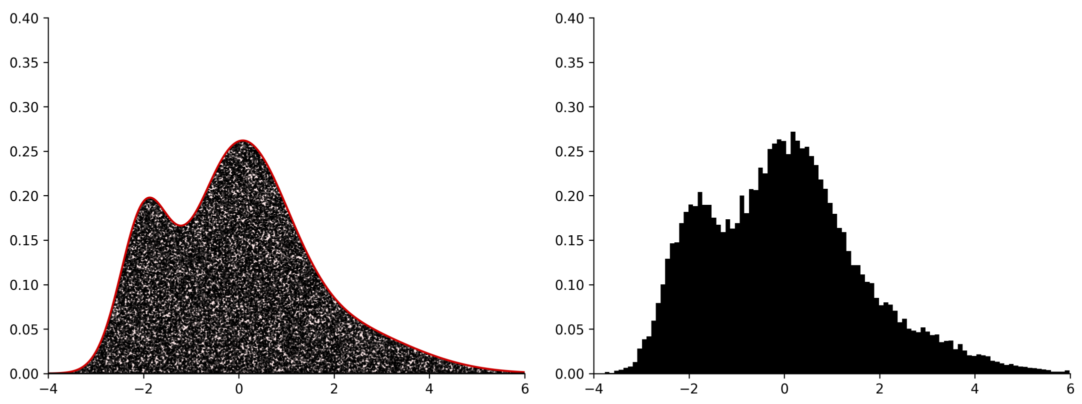

Figure 2.3: On the left, you can see a mixture of Gaussians (we will cover mixture distributions later) and samples uniformly distributed below the curve. Each black dot on under the curve is an (x, y) pair, hence you could denote those samples (X_i, Y_i). On the right, you can see the histogram of the x-marginal, which means, only X_i samples. This is the empirical demonstration of Theorem 2.3.

Theorem 2.3 (Fundamental Theorem of Simulation)**.** *(Martino et al., 2018, Theorem 2.2) Drawing samples from one dimensional random variable X with a density $p(x)$ is equivalent to sampling uniformly on the two dimensional region defined by*

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

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

Proof. The proof is simple but not necessary for us. See Martino et al. (2018, Theorem 2.2). □

An illustration of this theorem can be seen in Fig. 2.3. 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 2.3 suggests a quite intuitive sampling procedure: We can sample uniformly under the area of a density (or even an unnormalised negative 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

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

where $\Gamma(n) = (n - 1)!$ for integers. We will consider the special density $\mathsf{Beta}(2, 2)$. In order to design a uniform sampler under the area of the $\mathsf{Beta}(2, 2)$, we can use its special properties. For example, the Beta density is defined on $[0, 1]$ which makes it easy to design

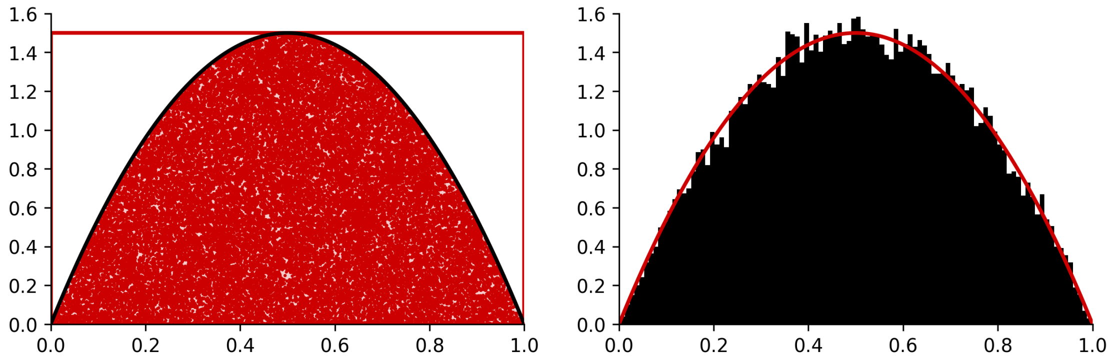
Figure 2.4: On the left, we plot the accepted samples (scattered) under the curve. On the right, we describe the histogram of x-marginal of these samples (so we just keep the first dimension of the two dimensional array).

a uniform sampler. A simple choice for this is the 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. 2.4 shows the result of this procedure together with the histogram.

2.2.1 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 sample from, denoted $q(x)$, which may cover our target density much better.

Rejection sampling is an algorithm just does that: 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 $q(x)$ with $M > 1$ should entirely cover $p(x)$. Depending on the choice of the proposal, the procedure will be much more efficient than simple boxing. Let us describe the conceptual algorithm.

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

$$a(X_i') = \frac{p(X_i')}{Mq(X_i')} \leq 1. \tag{2.7}$$

This algorithm might look simpler than what you would expect. Above we mentioned drawing samples uniformly under the curve, however, a simple look at the steps might not reveal the fact that this is precisely what this algorithm is doing. Let us look into this more carefully: The rejection sampler first generates $X' \sim q(x)$ and let us fix its value $X' = x'$[3]. Then in order to implement the *Accept* step, we should 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 $U \sim \text{Unif}(0, 1)$. Finally, you can see what the algorithm is doing behind the scenes:

- 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 2.3, this samples from the correct distribution!

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

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.

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

Algorithm 3 Pseudocode for rejection sampling without normalising constants

```
1: Input: The number of samples $n$ 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
```

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

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

where $Z = \int \bar{p}(x)\mathrm{d}x$. We describe a few examples below.

Example 2.9. It is easy to imagine why you could have an unnormalised density in the discrete case. Imagine, we have a bunch of numbers:

- People with black jumpers: 530
- People with red jumpers: 403
- People with yellow jumpers: 304

In this case, if somebody asked about the probability of seeing a "red jumper", we would *normalise* this number in order to obtain this probability:

$$p(\text{red jumper}) = \frac{530}{530 + 403 + 304} = 0.42.$$

Example 2.10. In physics, engineering, and even 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 2.3 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 samples $n$ and scaling factor $M$.
2: for $i = 1, \ldots, n$ do
3:     Generate $X' \sim q(x')$
4:     $U \sim$ 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
```

2.2.2 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 (2.7). 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 $\hat{a}$. Computing this probability is intuitive but we will not go into proof. For a proof, see Martino et al. (2018, Sec. 3.3.1) for a discussion. We give the acceptance rate below for the normalised and unnormalised cases.

ACCEPTANCE RATE FOR NORMALISED $p(x)$

When our density is normalised, the acceptance rate is given by

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

where $M > 1$ in order to satisfy the requirement that q covers p. It can be seen that smaller M is theoretically useful for us as it will mean higher acceptance rates.

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

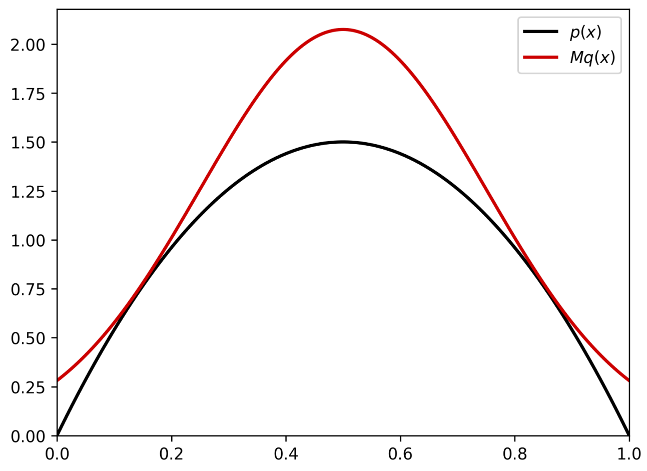

Figure 2.5: A better proposal for $p(x) = \text{Beta}(2,2)$

ACCEPTANCE RATE FOR UNNORMALISED $\bar{p}(x)$

In this case, denoting the normalising constant as

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

the acceptance rate is given as

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

2.2.3 EXAMPLES

In the following, we will develop examples where rejection sampling becomes crucial.

Example 2.11 (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). This will result in the graph shown in Fig. 2.5. In the lecture, we demonstrate that this proposal will result in higher acceptance rates than the box numerically.

Example 2.12 (Truncated Densities)**.** 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

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

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 $M = 1$). The resulting algorithm is extremely intuitive: All you need is to sample from $q(x) = \mathcal{N}(x;0,1)$ and reject if this sample is out of bounds $[-a,a]$.

Example 2.13. (A numerical example from Yıldırım (2017)) We are interested in sampling

$$X \sim \Gamma(\alpha, 1),$$

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

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

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

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

We therefore are interested in the parameter λ which gives us smallest M_λ. Let us write

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

This is maximised at (show this)

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

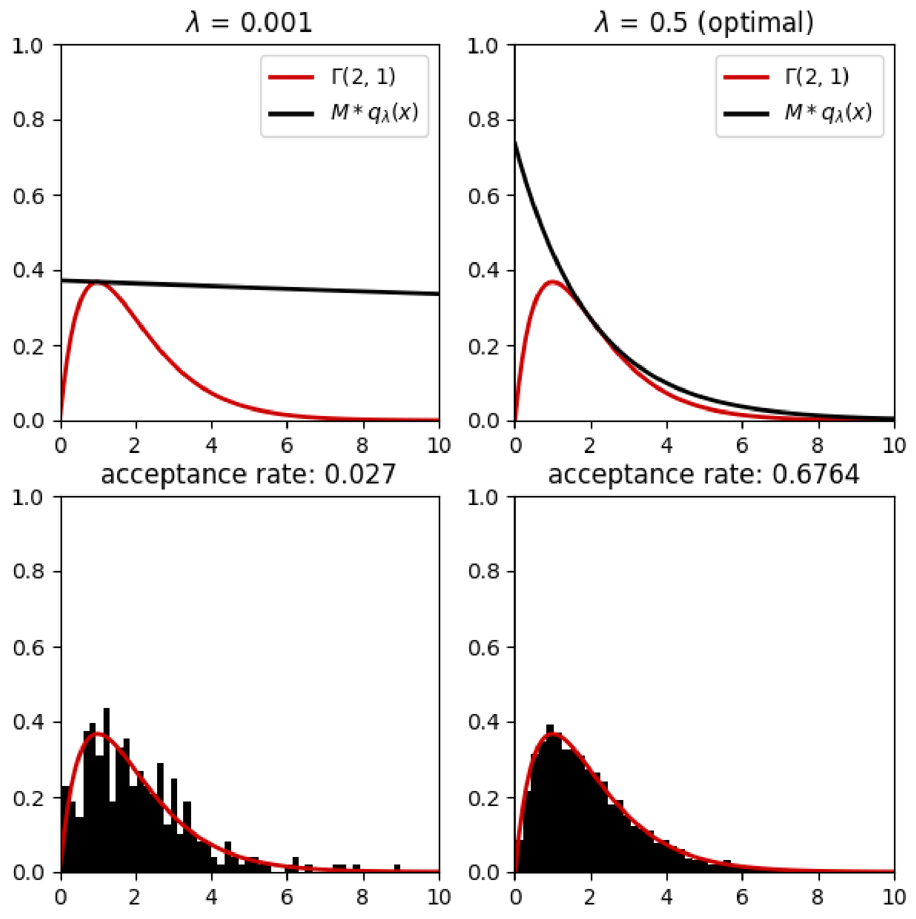

Figure 2.6: Two rejection sampling procedures for Example 2.13 with $\lambda = 0.001$ and optimal $\lambda = 1/\alpha$ (as derived in the example) for $n = 10000$.

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. 2.6.

2.3 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.

2.3.1 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. 2.7. 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: $p(k) = w_k$

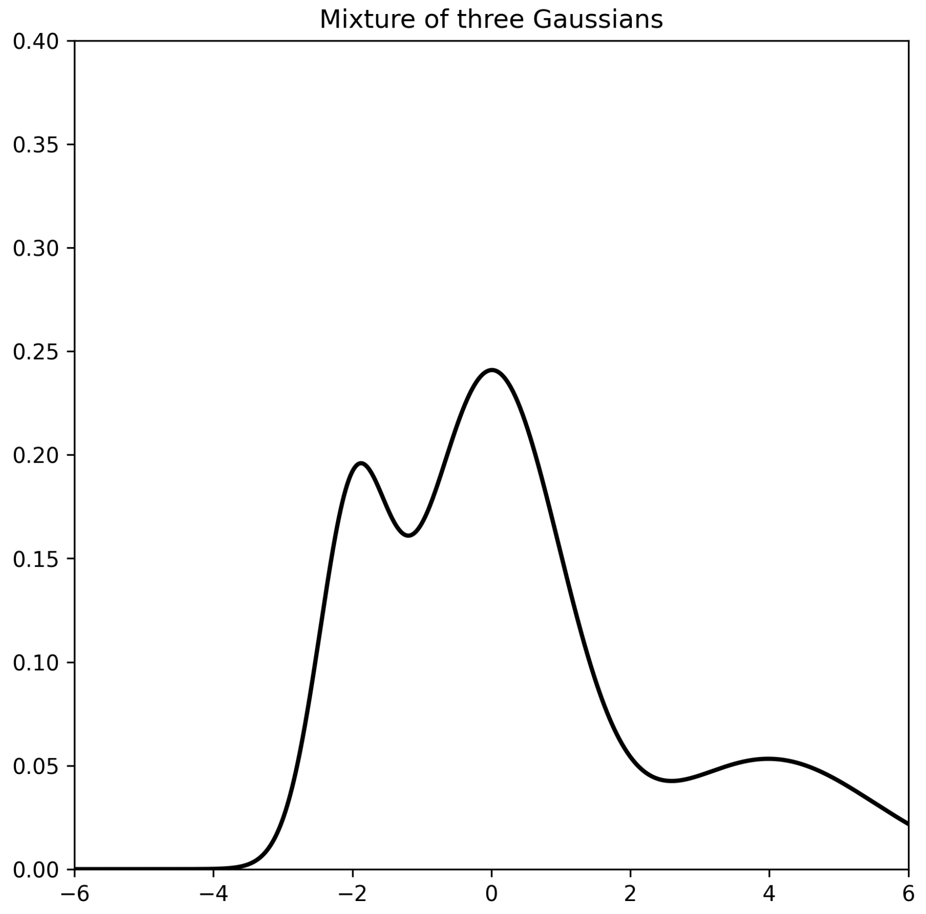

Figure 2.7: The density of a mixture of three Gaussians: $p(x) = \sum_{k=1}^{3} w_k \mathcal{N}(x; \mu_k, \sigma_k^2)$ with $\mu_1 = -2, \mu_2 = 0, \mu_3 = 4, \sigma_1 = 0.5, \sigma_2 = 1, \sigma_3 = 0.5, w_1 = 0.2, w_2 = 0.6, w_3 = 0.2$.

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, sampling a mixture of Gaussians is easy: Sample $k \sim p(k)$ from the

Algorithm 5 Sampling discrete mixtures

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

PMF consists of weights w_k, then sample from the selected Gaussian.

2.3.2 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.

2.3.3 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

$$X, Y \sim p(x, y) \tag{2.8}$$

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 (2.8), 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

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

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 2.1. 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.

2.3.4 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 2.14. Assume that

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

and

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

Then it can be shown that

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

This can be verified by

- 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

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

Implement and check this.

BIBLIOGRAPHY

Box, GEP and Müller, Mervin E. 1958. *A Note on the Generation of Random Normal Deviates*. In The Annals of Mathematical Statistics, vol. 29, no. 2, pp. 610–611. Cited on p. 10.

Devroye, Luc. 1986. *Non-Uniform Random Variate Generation*. Cited on p. 4.

Martino, Luca; Luengo, David; and Míguez, Joaquín. 2018. *Independent random sampling methods*. Springer. Cited on pp. i, 5, 12, 13, 16, and 17.

Robert, Christian P and Casella, George. 2004. *Monte Carlo statistical methods*. Springer. Cited on p. i.

Yıldırım, Sinan. 2017. *Sabanci University IE 58001 Lecture notes: Simulation Methods for Statistical Inference*. Cited on pp. i and 19.