MFC CDT Probability and Statistics
Week 6

O. Deniz Akyildiz

Mathematics for our Future Climate: Theory, Data and Simulation (MFC CDT).

November 11, 2024

IMPERIAL

https://akyildiz.me/

𝕏: @odakyildiz

So far, you have learned about probability theory and statistics.

So far, you have learned about probability theory and statistics.

The rest of this course will teach you how to simulate (pseudo) random numbers that attain certain statistical properties.

So far, you have learned about probability theory and statistics.

The rest of this course will teach you how to simulate (pseudo) random numbers that attain certain statistical properties.

This course will also teach you how to estimate certain quantities of interest using these random numbers.

So far, you have learned about probability theory and statistics.

The rest of this course will teach you how to simulate (pseudo) random numbers that attain certain statistical properties.

This course will also teach you how to estimate certain quantities of interest using these random numbers.

- Expectations with respect to intractable distributions
- Tail probabilities
- Sampling from posterior distributions of Bayesian models

Timeline

- Week 6: Exact sampling methods, rejection samplers, importance samplers

Timeline

- Week 6: Exact sampling methods, rejection samplers, importance samplers
- Week 7: Markov chain Monte Carlo methods

Timeline

- Week 6: Exact sampling methods, rejection samplers, importance samplers
- Week 7: Markov chain Monte Carlo methods
- Week 8: Langevin Monte Carlo, Energy-Based Models, Score-Based Generative Models

Timeline

- Week 6: Exact sampling methods, rejection samplers, importance samplers
- Week 7: Markov chain Monte Carlo methods
- Week 8: Langevin Monte Carlo, Energy-Based Models, Score-Based Generative Models
- Week 9: Stochastic Filtering, Sequential Monte Carlo, Particle Filters

Timeline

- Week 6: Exact sampling methods, rejection samplers, importance samplers
- Week 7: Markov chain Monte Carlo methods
- Week 8: Langevin Monte Carlo, Energy-Based Models, Score-Based Generative Models
- Week 9: Stochastic Filtering, Sequential Monte Carlo, Particle Filters
- Week 10: Parameter Estimation in State-Space Models: Maximum Likelihood and Bayesian Estimation

The first focus will be on *independent exact sampling* methods.

The first focus will be on *independent exact sampling* methods.

- We will discuss and design algorithms that sample directly from basic distributions, such as

The first focus will be on *independent exact sampling* methods.

- We will discuss and design algorithms that sample directly from basic distributions, such as
  - Uniform distribution

The first focus will be on *independent exact sampling* methods.

- We will discuss and design algorithms that sample directly from basic distributions, such as
  - Uniform distribution
  - Gaussian distribution

The first focus will be on *independent exact sampling* methods.

- We will discuss and design algorithms that sample directly from basic distributions, such as
  - Uniform distribution
  - Gaussian distribution
  - Exponential distribution

  and others.

The first focus will be on *independent exact sampling* methods.

- We will discuss and design algorithms that sample directly from basic distributions, such as
  - Uniform distribution
  - Gaussian distribution
  - Exponential distribution

  and others.

These random number generation techniques are at the core of many fields, e.g., statistical inference and generative models.

Then, we will move on to Monte Carlo integration and Markov chain Monte Carlo methods:

Then, we will move on to Monte Carlo integration and Markov chain Monte Carlo methods:

- Importance sampling

Then, we will move on to Monte Carlo integration and Markov chain Monte Carlo methods:

- Importance sampling
- Sampling from intractable distributions by forming Markov chains and targeting them

Then, we will move on to Monte Carlo integration and Markov chain Monte Carlo methods:

- Importance sampling
- Sampling from intractable distributions by forming Markov chains and targeting them
- Computation of integrals, expectations

Then, we will move on to Monte Carlo integration and Markov chain Monte Carlo methods:

- Importance sampling
- Sampling from intractable distributions by forming Markov chains and targeting them
- Computation of integrals, expectations

Then finally, we will finalize with sequential Monte Carlo (if time permits).

In computational Bayesian statistics, we are interested in synthesising *the model* and *the data* (among other things).

In computational Bayesian statistics, we are interested in synthesising *the model* and *the data* (among other things).

One very effective way is to use Bayesian statistical methodology.

In computational Bayesian statistics, we are interested in synthesising *the model* and *the data* (among other things).

One very effective way is to use Bayesian statistical methodology.

For this, we are often interested in sampling from posterior distributions of the form

$$p(x|y) \propto p(y|x)\pi(x), \tag{1}$$

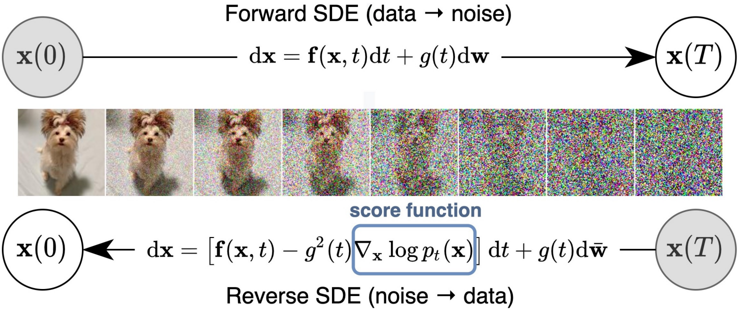
and estimating expectations w.r.t. them.

In this case, we have samples $\{Y_i\}_{i=1}^n$ from a dataset. Underlying data distribution $Y_i \sim p_{\text{data}}$ is not accessible in any way.

In this case, we have samples $\{Y_i\}_{i=1}^n$ from a dataset. Underlying data distribution $Y_i \sim p_{\text{data}}$ is not accessible in any way.

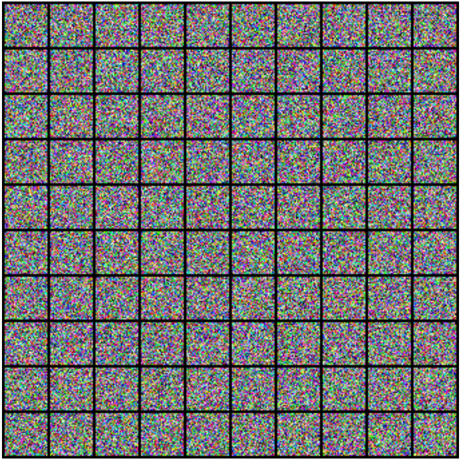
Goal: Sample from p_{data} by only accessing data $\{Y_i\}_{i=1}^n$.

The standard way to do it is to run forward and backward stochastic differential equations[1]



[1] Figure from: `https://yang-song.net/blog/2021/score/`

The standard way to do it is to run forward and backward stochastic differential equations[2]



[2]Figure from: `https://yang-song.net/blog/2021/score/`

In summary, this course can be immensely useful for you to go into any of these areas.

In summary, this course can be immensely useful for you to go into any of these areas.

We will balance computation and theory for practical and conceptual understanding.

In summary, this course can be immensely useful for you to go into any of these areas.

We will balance computation and theory for practical and conceptual understanding.

Let's get to our first motivating example.

In this course, a core focus will be estimating certain quantities (probabilities, expectations, etc.) using *sampling*.

Estimating π

Without being too clever

In this course, a core focus will be estimating certain quantities (probabilities, expectations, etc.) using *sampling*.

Sampling here means *random variate generation*.

Estimating π

Without being too clever

In this course, a core focus will be estimating certain quantities (probabilities, expectations, etc.) using *sampling*.

Sampling here means *random variate generation*.

Let's try to solve a simple problem to illustrate the methodology: Estimating π.

Estimating π

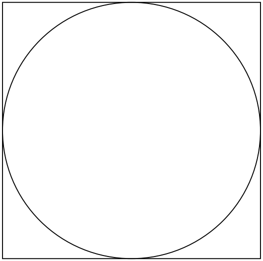
Without being too clever

In this course, a core focus will be estimating certain quantities (probabilities, expectations, etc.) using *sampling*.

Sampling here means *random variate generation*.

Let's try to solve a simple problem to illustrate the methodology: Estimating π.

Can we estimate π using sampling? Any ideas?

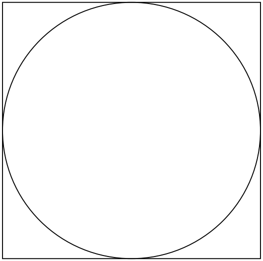


Given the knowledge that:

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

Estimating π

Without being too clever



Given the knowledge that:

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

Can we phrase this question probabilistically?

Estimating π

Without being too clever

What does this mean?

- Write down the estimation problem as

Estimating π

Without being too clever

What does this mean?

- Write down the estimation problem as
  - a probability (of a set)

Estimating π

Without being too clever

What does this mean?

- Write down the estimation problem as
  - a probability (of a set)
  - an expectation

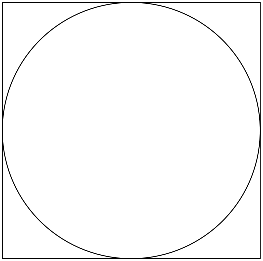
Estimating π

Without being too clever

What does this mean?

- Write down the estimation problem as
  - a probability (of a set)
  - an expectation

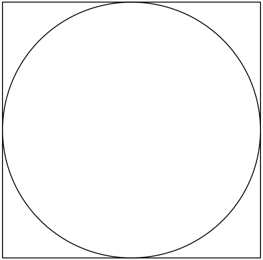
Most of the time, expectation is the most general way.



Consider a 2D uniform distribution on $[-1, 1] \times [-1, 1]$.

Estimating π

Without being too clever

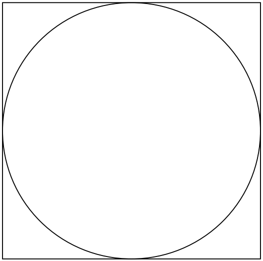


Consider a 2D uniform distribution on $[-1, 1] \times [-1, 1]$.

- The 'probability' of the square (whole space) is 1.

Estimating π

Without being too clever



Consider a 2D uniform distribution on $[-1, 1] \times [-1, 1]$.

- The 'probability' of the square (whole space) is 1.
- The 'probability of the circle' (set) is precisely the ratio of areas.

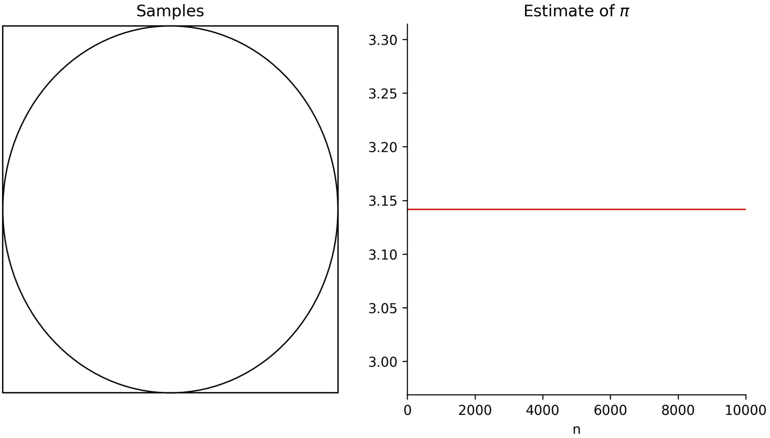
$$\mathbb{P}(\text{Circle}) = \frac{\pi}{4}.$$

Last question:

Can we estimate the probability of this set, if we had access to samples from $\mathrm{Unif}([-1, 1] \times [-1, 1])$?

Estimating π

Without being too clever



We have used here the most basic idea of estimating an integral (we will clarify shortly).

[3] This is different from π the number!

We have used here the most basic idea of estimating an integral (we will clarify shortly).

Consider now a target measure $\pi(x)\mathrm{d}x$[3] and a function $\varphi(x)$. If we have access to i.i.d samples from $X_i \sim \pi(x)$, then

$$(\varphi, \pi) := \int \varphi(x)\pi(x)\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \varphi(X_i),$$

using a particle approximation

$$\pi^N(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X_i}(\mathrm{d}x).$$

[3]This is different from π the number!

Note by definition of the Dirac measure

$$\varphi(y) = \int \varphi(x)\delta_y(\mathrm{d}x).$$

Therefore, given the approximation $\pi^N(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^N \delta_{X_i}(\mathrm{d}x)$, we have

$$(\varphi, \pi) \approx (\varphi, \pi^N) = \int \varphi(x)\frac{1}{N}\sum_{i=1}^N \delta_{X_i}(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^N \varphi(X_i).$$

Let $\mathrm{X} = [-1, 1] \times [-1, 1]$ and define the uniform measure such that $\mathbb{P}(\mathrm{X}) = 1$.

Let A be the "circle" s.t. $A \subset \mathrm{X}$. Now, the probability of A is given

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

where $x_i \sim \mathbb{P}$.

In general, we will be interested in sampling from general (unnormalized) distributions:

$$\pi(x) \propto \frac{\gamma(x)}{Z},$$

where $Z = \int \gamma(x)\mathrm{d}x$ is the normalizing constant. Our general aim throughout this course is to compute

$$(\varphi, \pi) = \int \varphi(x)\pi(x)\mathrm{d}x,$$

for $\varphi : \mathsf{X} \to \mathbb{R}$ a measurable function. $\varphi(x) = x^n$ for moments, $\varphi(x) = \mathbf{1}_A(x)$ for probabilities... In Bayesian inference

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

An estimator of the form

$$(\varphi, \pi^N) = \frac{1}{N} \sum_{i=1}^{N} \varphi(X_i),$$

is desirable.

An estimator of the form

$$(\varphi, \pi^N) = \frac{1}{N} \sum_{i=1}^{N} \varphi(X_i),$$

is desirable. Because

- The estimator is unbiased $\mathbb{E}[(\varphi, \pi^N)] = (\varphi, \pi)$.

Perfect Monte Carlo

The general problem

An estimator of the form

$$(\varphi, \pi^N) = \frac{1}{N}\sum_{i=1}^{N}\varphi(X_i),$$

is desirable. Because

- The estimator is unbiased $\mathbb{E}[(\varphi, \pi^N)] = (\varphi, \pi)$.
- The variance of this estimator is

  $$\mathrm{var}((\varphi, \pi^N)) = \frac{1}{N}\mathrm{var}(\varphi(X)),$$

  which is decreasing in N.

An estimator of the form

$$(\varphi, \pi^N) = \frac{1}{N} \sum_{i=1}^{N} \varphi(X_i),$$

is desirable. Because

- The estimator is unbiased $\mathbb{E}[(\varphi, \pi^N)] = (\varphi, \pi)$.
- The variance of this estimator is

  $$\mathrm{var}((\varphi, \pi^N)) = \frac{1}{N} \mathrm{var}(\varphi(X)),$$

  which is decreasing in N.
- Strong LLN holds

  $$(\varphi, \pi^N) \to (\varphi, \pi) \quad \text{a.s. as } N \to \infty.$$

An estimator of the form

$$(\varphi, \pi^N) = \frac{1}{N} \sum_{i=1}^{N} \varphi(X_i),$$

is desirable. Because

- The estimator is unbiased $\mathbb{E}[(\varphi, \pi^N)] = (\varphi, \pi)$.
- The variance of this estimator is

  $$\operatorname{var}((\varphi, \pi^N)) = \frac{1}{N} \operatorname{var}(\varphi(X)),$$

  which is decreasing in N.
- Strong LLN holds

  $$(\varphi, \pi^N) \to (\varphi, \pi) \quad \text{a.s. as } N \to \infty.$$

- CLT holds

  $$\sqrt{N}\left((\varphi, \pi^N) - (\varphi, \pi)\right) \to \mathcal{N}(0, \sigma^2(\varphi, \pi)) \quad \text{as } N \to \infty.$$

In terms of theoretical guarantees, we will favor L_p bounds, i.e., for perfect MC, one can show that

$$\|(\varphi, \pi) - (\varphi, \pi^N)\|_p \leq \frac{c_p \|\varphi\|_\infty}{\sqrt{N}},$$

for bounded test functions φ, i.e., $\|\varphi\|_\infty < \infty$.

In terms of theoretical guarantees, we will favor L_p bounds, i.e., for perfect MC, one can show that

$$\|(\varphi, \pi) - (\varphi, \pi^N)\|_p \leq \frac{c_p \|\varphi\|_\infty}{\sqrt{N}},$$

for bounded test functions φ, i.e., $\|\varphi\|_\infty < \infty$.

We are mostly interested in $p = 2$ case, which is square root of the MSE in general.

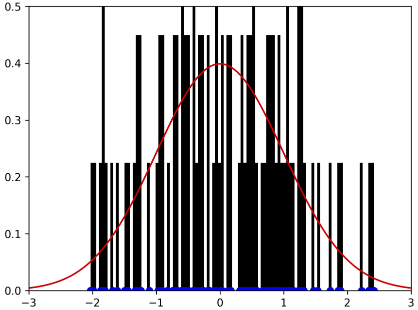
In order to perform perfect and approximate Monte Carlo integration, we need to be able to generate random variables from distributions.

In order to perform perfect and approximate Monte Carlo integration, we need to be able to generate random variables from distributions.

Next up: Pseudo uniform random number generation.

What are pseudo-random numbers?

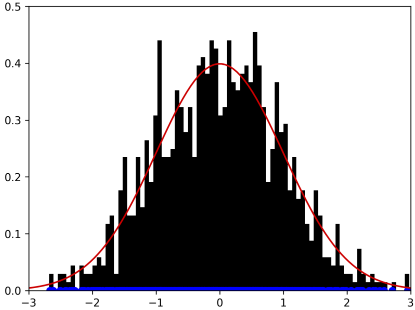
Why do we need them?



$$X^{(i)} \sim \pi(x)$$

What are pseudo-random numbers?

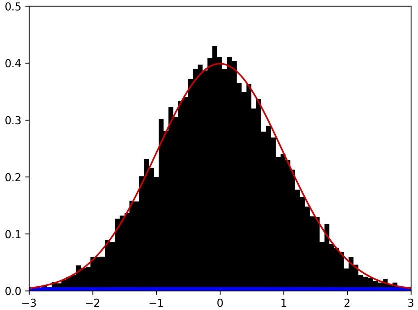
Why do we need them?



$$X^{(i)} \sim \pi(x)$$

What are pseudo-random numbers?

Why do we need them?



$$X^{(i)} \sim \pi(x)$$

What are pseudo-random numbers?

Why do we need them?

It is (literally) impossible to generate genuinely random numbers on computers.

What are pseudo-random numbers?

Why do we need them?

It is (literally) impossible to generate genuinely random numbers on computers.

- You can flip a coin every time you need a binary number
  - Is it really unbiased though?[4]
- Throw a die

[4] Diaconis, P., Holmes, S., & Montgomery, R. (2007). Dynamical bias in the coin toss. *SIAM review*, 49(2), 211-235.

What are pseudo-random numbers?

Why do we need them?

It is (literally) impossible to generate genuinely random numbers on computers.

- You can flip a coin every time you need a binary number
  - Is it really unbiased though?[4]
- Throw a die

What other things can give you a truly random number?

[4]Diaconis, P., Holmes, S., & Montgomery, R. (2007). Dynamical bias in the coin toss. *SIAM review*, 49(2), 211-235.

What are pseudo-random numbers?

Why do we need them?

It is (literally) impossible to generate genuinely random numbers on computers.

- You can flip a coin every time you need a binary number
  - Is it really unbiased though?[4]
- Throw a die

What other things can give you a truly random number?

- You can use `www.random.org`

[4] Diaconis, P., Holmes, S., & Montgomery, R. (2007). Dynamical bias in the coin toss. *SIAM review*, 49(2), 211-235.

What are pseudo-random numbers?

Why do we need them?

It is (literally) impossible to generate genuinely random numbers on computers.

- You can flip a coin every time you need a binary number
  - Is it really unbiased though?[4]
- Throw a die

What other things can give you a truly random number?

- You can use `www.random.org`
- On a computer
  - Try to measure some inner thermal noise (of circuits)
  - Measure atmospheric noise

[4] Diaconis, P., Holmes, S., & Montgomery, R. (2007). Dynamical bias in the coin toss. *SIAM review*, 49(2), 211-235.

What are pseudo-random numbers?

Why do we need them?

It is (literally) impossible to generate genuinely random numbers on computers.

- You can flip a coin every time you need a binary number
  - Is it really unbiased though?[4]
- Throw a die

What other things can give you a truly random number?

- You can use `www.random.org`
- On a computer
  - Try to measure some inner thermal noise (of circuits)
  - Measure atmospheric noise

As you can see, these are not very practical.

[4] Diaconis, P., Holmes, S., & Montgomery, R. (2007). Dynamical bias in the coin toss. *SIAM review*, 49(2), 211-235.

What are pseudo-random numbers?

Why do we need them?

If we want to simulate randomness, we need to obtain a way that is

What are pseudo-random numbers?

Why do we need them?

If we want to simulate randomness, we need to obtain a way that is

- Repeatable

What are pseudo-random numbers?

Why do we need them?

If we want to simulate randomness, we need to obtain a way that is

- Repeatable
- Cheap

What are pseudo-random numbers?

Why do we need them?

If we want to simulate randomness, we need to obtain a way that is

- Repeatable
- Cheap

It has become an entire research topic to design *deterministic* algorithms which gives samples that match the desired characteristics.

We will start from the simplest: The uniform distribution.

Uniform pseudo-random numbers

The most important sampling task

The key to simulate many (many) other random variables is to be able to simulate uniform random numbers.

[5] Note that current state-of-the-art is not based on this.

Uniform pseudo-random numbers

The most important sampling task

The key to simulate many (many) other random variables is to be able to simulate uniform random numbers.

We denote the task

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

More precisely

$$U \sim p(u) = 1 \quad \text{for } 0 \leq u \leq 1.$$

[5] Note that current state-of-the-art is not based on this.

The key to simulate many (many) other random variables is to be able to simulate uniform random numbers.

We denote the task

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

More precisely

$$U \sim p(u) = 1 \quad \text{for } 0 \leq u \leq 1.$$

We will look into an old way of doing it:

- Linear congruential random number generators

These methods are based on generating a *deterministic linear recursion* with a careful design [5].

[5] Note that current state-of-the-art is not based on this.

Linear congruential generators (LCGs from now on) are based on simulating a recursion:

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

where x_0 is the **seed**, m is the **modulus**, b is the **shift**, and a is the **multiplier**.

Uniform pseudo-random numbers

The most important sampling task

Linear congruential generators (LCGs from now on) are based on simulating a recursion:

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

where x_0 is the **seed**, m is the **modulus**, b is the **shift**, and a is the **multiplier**.

- m is an integer
- $x_0, a, b \in \{0, \ldots, m-1\}$.

Given $x_n \in \{0, \ldots, m-1\}$, we generate the uniform random numbers

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

Uniform pseudo-random numbers

The most important sampling task

Example code (try and make it work!)

```
import numpy as np
import matplotlib.pyplot as plt

def lcg(a, b, m, n, x0):
    x = np.zeros(n)
    u = np.zeros(n)
    x[0] = x0
    u[0] = x0 / m
    for k in range(1, n):
        x[k] = (a * x[k - 1] + b) % m
        u[k] = x[k] / m
    return u
```

Uniform pseudo-random numbers

The most important sampling task

A few things to know about LCGs:

- They generate *periodic* sequences.

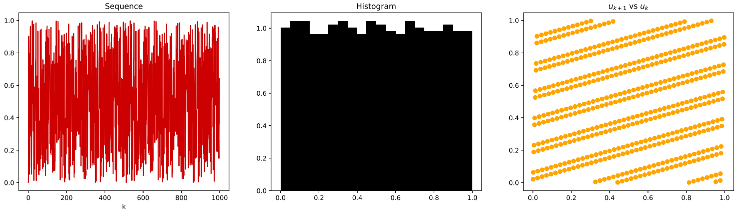


Figure: $m = 2048$, $a = 43$, $b = 0$, $x_0 = 1$.

period $T \leq m$ (m: the modulus).

- Full period: $T = m$

Choice of good parameters rely on some theory, some art.

Wikipedia has a list of parameters for professional implementations:

Parameters in common use [edit]

The following table lists the parameters of LCGs in common use, including built-in *rand()* functions in runtime libraries of various compilers. This table is to show popularity, not examples to emulate; *many of these parameters are poor.* Tables of good parameters are available.[10][2]

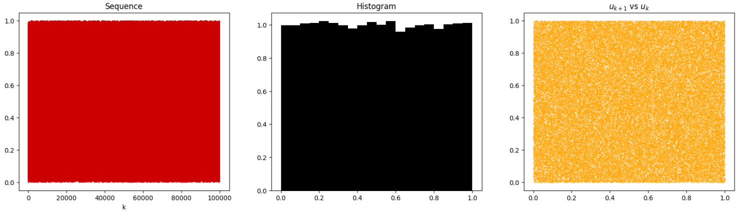
Source	modulus *m*	multiplier *a*	increment *c*	output bits of seed in *rand()* or *Random(L)*
ZX81	$2^{16}+1$	75	74	
Numerical Recipes from the "quick and dirty generators" list, Chapter 7.1, Eq. 7.1.6 parameters from Knuth and H. W. Lewis	2^{32}	1664525	1013904223	
Borland C/C++	2^{32}	22695477	1	bits 30..16 in *rand()*, 30..0 in *lrand()*
glibc (used by GCC)[17]	2^{31}	1103515245	12345	bits 30..0
ANSI C: Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++[18] C90, C99, C11: Suggestion in the ISO/IEC 9899,[19] C17	2^{31}	1103515245	12345	bits 30..16
Borland Delphi, Virtual Pascal	2^{32}	134775813	1	bits 63..32 of *(seed × L)*
Turbo Pascal	2^{32}	134775813 (8088405_{16})	1	
Microsoft Visual/Quick C/C++	2^{32}	214013 ($343FD_{16}$)	2531011 ($269EC3_{16}$)	bits 30..16
Microsoft Visual Basic (6 and earlier)[20]	2^{24}	1140671485 ($43FD43FD_{16}$)	12820163 ($C39EC3_{16}$)	
RtlUniform from Native API[21]	$2^{31}-1$	2147483629 ($7FFFFFED_{16}$)	2147483587 ($7FFFFFC3_{16}$)	
Apple CarbonLib, C++11's				

from: `https://en.wikipedia.org/wiki/Linear_congruential_generator`

Uniform pseudo-random numbers

The most important sampling task

Better parameters:



Going forward, we will mostly assume that we will have access to a uniform random number generator.

Going forward, we will mostly assume that we will have access to a uniform random number generator.

- When implementing $U \sim \text{Unif}(0, 1)$, you can instead use

  ```
  rng.uniform(0, 1, n)
  ```

  where `n` is the number of samples you want to draw and `rng` is appropriately initialised random number generator.

Going forward, we will mostly assume that we will have access to a uniform random number generator.

- When implementing $U \sim \text{Unif}(0, 1)$, you can instead use

```
rng.uniform(0, 1, n)
```

  where `n` is the number of samples you want to draw and `rng` is appropriately initialised random number generator.

Next up: Exact sampling methods

We will cover methods to sample more general nonuniform $\pi(x)$:

- Inversion method

We will cover methods to sample more general nonuniform $\pi(x)$:

- Inversion method
- Transformation method

We will cover methods to sample more general nonuniform $\pi(x)$:

- Inversion method
- Transformation method
- Rejection method

We will cover methods to sample more general nonuniform $\pi(x)$:

- Inversion method
- Transformation method
- Rejection method

Next up: Sampling via inversion.

Exact sampling of distributions

How to simulate from any $\pi(x)$?

Simulating from a given $\pi(x)$ is an endless research area (simulation, sampling, generative models) and still flourishing.

Exact sampling of distributions

How to simulate from any $\pi(x)$?

Simulating from a given $\pi(x)$ is an endless research area (simulation, sampling, generative models) and still flourishing.

We will start by describing some general methods to sample from more general distributions.

THE INSTITUTE FOR ADVANCED STUDY
SCHOOL OF MATHEMATICS
PRINCETON, NEW JERSEY

May 21, 1947

Mr. Stan Ulam
Post Office Box 1663
Santa Fe
New Mexico

Dear Stan:

Thanks for your letter of the 19th. I need not tell you that Klari and I are looking forward to the trip and visit at Los Alamos this Summer. I have already received the necessary papers from Carson Mark. I filled out and returned mine yesterday; Klari's will follow today.

I am very glad that preparations for the random numbers work are to begin soon. In this connection, I would like to mention this: Assume that you have several random number distributions, each equidistributed in $0, 1$: $(x^i), (y^i), (z^i), \ldots$. Assume that you want one with the distribution function (density) $f(\xi)\,d\xi : (\xi^i)$. One way to form it is to form the cumulative distribution function: $g(\xi) = \int^{\xi} f(\xi)\,d\xi$ to invert it $h(x) = \xi \rightleftarrows x = g(\xi)$, and to form $\xi^i = h(x^i)$ with this $h(x)$, or some approximant polynomial. This is, as I see, the method that you have in mind.

The inversion technique is based on the following theorem (Theorem 2.1 of notes):

Theorem 1

Consider a random variable X with a CDF F_X. Then the random variable $F_X^{-1}(U)$ where $U \sim \text{Unif}(0,1)$, 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. ∎

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,

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

Going back to statement: If $U \sim \text{Unif}(0, 1)$ then $X' = F_X^{-1}(U)$ has the desired distribution, i.e.,

$$X' \sim p_X(x).$$

Exact sampling of distributions

Inversion

Going back to statement: If $U \sim \mathrm{Unif}(0, 1)$ then $X' = F_X^{-1}(U)$ has the desired distribution, i.e.,

$$X' \sim p_X(x).$$

Then this suggests an algorithm:

- Sample $U \sim \mathrm{Unif}([0, 1])$,
- Draw $X = F_X^{-1}(U)$.

Going back to statement: If $U \sim \text{Unif}(0,1)$ then $X' = F_X^{-1}(U)$ has the desired distribution, i.e.,

$$X' \sim p_X(x).$$

Then this suggests an algorithm:

- Sample $U \sim \text{Unif}([0,1])$,
- Draw $X = F_X^{-1}(U)$.

Of course, this is limited to the cases where we *can* invert the CDF.

Exact sampling of distributions

Inversion: Discrete (categorical) distribution

Let us consider some examples.

The most generic one is the discrete (categorical) distribution. For $K \geq 1$ (integer), define K states $s_1, \ldots, s_K$ where

$$p(s_k) \in [0, 1] \qquad \text{where} \qquad \sum_{k=1}^{K} p(s_k) = 1.$$

Simpler than it looks, consider the die:

$$s_k = k \text{ (the face of die)}$$

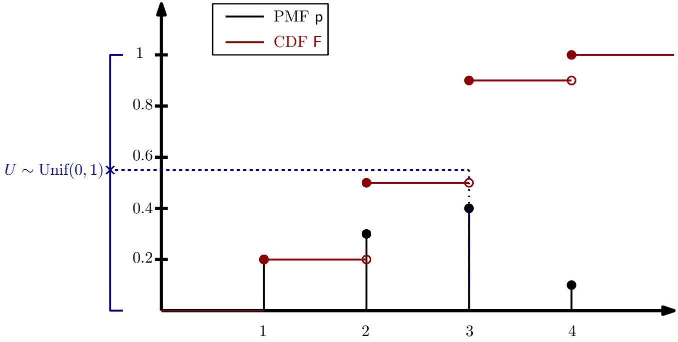
and their probabilities

$$p(s_k) = 1/6.$$

Exact sampling of distributions

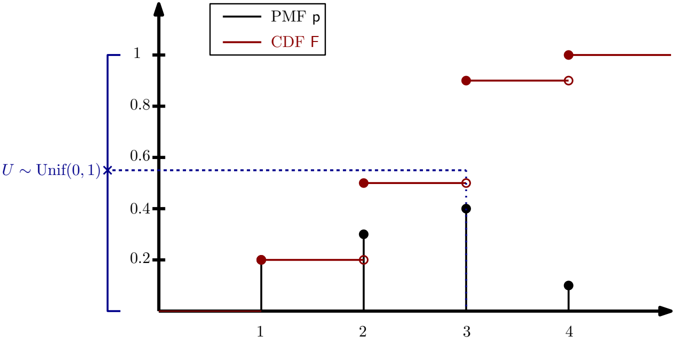
Inversion: Discrete (categorical) distribution

How does the sampling work?



- Draw $U \sim \mathrm{Unif}(0,1)$
- Choose $F_X^{-}(u) = \min\{x : F_X(x) \geq u\}$

How does the sampling work?



- Draw $U \sim \text{Unif}(0,1)$
- Choose $F_X^-(u) = \min\{x : F_X(x) \geq u\}$ generic for discrete dist.

Exact sampling of distributions

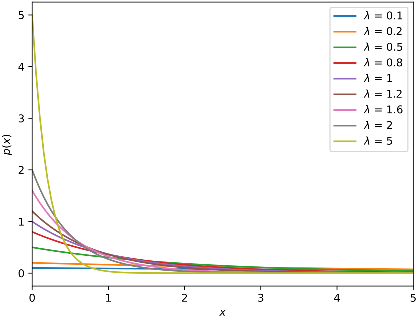
Inversion: Discrete (categorical) distribution

```
import numpy as np
import matplotlib.pyplot as plt

w = np.array([0.2, 0.3, 0.4, 0.1]) # pmf
s = np.array([1, 2, 3, 4])         # support (states)

def discrete_cdf(w):
return np.cumsum(w)

cw = discrete_cdf(w)

def plot_discrete_cdf(w, cw):
fig, ax = plt.subplots(1, 2, figsize=(20, 5))
ax[0].stem(s, w)
ax[1].plot(s, cw, 'o-', drawstyle='steps-post')
plt.show()

plot_discrete_cdf(w, cw)
```

The exponential density

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

for $x \geq 0$. Otherwise $\pi(x) = 0$.



Exact sampling of distributions

Inversion: Exponential distribution

$$\pi(x) = \mathrm{Exp}(x; \lambda) = \lambda e^{-\lambda x}.$$

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

We calculate the CDF

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

We calculate the CDF

$$F_X(x) \quad = \quad \int_0^x \pi(x')\mathrm{d}x',$$

$$\pi(x) = \mathrm{Exp}(x;\lambda) = \lambda e^{-\lambda x}.$$

We calculate the CDF

$$\begin{aligned} F_X(x) &= \int_0^x \pi(x')\mathrm{d}x', \\ &= \lambda \int_0^x e^{-\lambda x'}\mathrm{d}x', \end{aligned}$$

$$\pi(x) = \mathrm{Exp}(x; \lambda) = \lambda e^{-\lambda x}.$$

We calculate the CDF

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

Exact sampling of distributions

Inversion: Exponential distribution

$$\pi(x) = \mathrm{Exp}(x; \lambda) = \lambda e^{-\lambda x}.$$

We calculate the CDF

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

Exact sampling of distributions

Inversion: Exponential distribution

Deriving the inverse:

$$u = 1 - e^{-\lambda x}$$

Deriving the inverse:

$$\begin{aligned} u &= 1 - e^{-\lambda x} \\ \implies x &= -\frac{1}{\lambda}\log(1-u) \end{aligned}$$

Deriving the inverse:

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

Deriving the inverse:

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

So what is the algorithm?

Deriving the inverse:

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

So what is the algorithm?

- Generate $u_i \sim \text{Unif}([0,1])$
- $x_i = -\lambda^{-1}\log(1-u_i)$.

Deriving the inverse:

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

So what is the algorithm?

- Generate $u_i \sim \text{Unif}([0,1])$
- $x_i = -\lambda^{-1}\log(1-u_i)$.

Exact sampling of distributions

Inversion: Is Gaussian possible?

Let $\pi(x) = \mathcal{N}(x; \mu, \sigma^2)$. Can we use inversion?

Exact sampling of distributions

Inversion: Is Gaussian possible?

Let $\pi(x) = \mathcal{N}(x; \mu, \sigma^2)$. Can we use inversion?

No. F_X^{-1} is impossible to compute and hard to approximate.

Inversion is a special case of a general method called *transformation method*.

Inversion is a special case of a general method called *transformation method*.

Transformation method:

- Sample $U_i \sim \text{Unif}(u; 0, 1)$

Inversion is a special case of a general method called *transformation method*.

Transformation method:

- Sample $U_i \sim \text{Unif}(u; 0, 1)$
- Transform: $X_i = g(U_i)$.

Inversion is a special case of a general method called *transformation method*.

Transformation method:

- Sample $U_i \sim \text{Unif}(u; 0, 1)$
- Transform: $X_i = g(U_i)$.

Inversion is just setting $g = F_X^{-1}$.

The simplest example can be seen from sampling a uniform on $[a, b]$ using a uniform on $[0, 1]$.

The simplest example can be seen from sampling a uniform on $[a, b]$ using a uniform on $[0, 1]$.

- Draw $U_i \sim \text{Unif}(u; 0, 1)$
- Set $X_i = g(U_i) = (b - a)U_i + a$

then $X_i \sim \text{Unif}(x; a, b)$.

For general g, how do we compute the density?

Exact sampling of distributions

Transformation method

If $X \sim p_X(x)$ and $Y = g(X)$, what is $p_Y(y)$?

If $X \sim p_X(x)$ and $Y = g(X)$, what is $p_Y(y)$?

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

If $X \sim p_X(x)$ and $Y = g(X)$, what is $p_Y(y)$?

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

where J is the Jacobian of the inverse mapping g^{-1}, evaluated at y:

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

Exact sampling of distributions

Transformation method: An exercise

If $X \sim \mathcal{N}(0, 1)$, derive the distribution of

$$Y = \sigma X + \mu.$$

The inverse transform is:

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

The inverse transform is:

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

Therefore,

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

which is

Exact sampling of distributions

Transformation method: An exercise

The inverse transform is:

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

Therefore,

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

which is

$$p_Y(y) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(y-\mu)^2}{2\sigma^2}) \frac{1}{\sigma} = \mathcal{N}(\mu, \sigma^2)$$

How to sample Gaussians from uniforms?

Box-Müller method

Finally, we provide the Box-Müller method for Gaussians: Let $U_1, U_2 \sim \mathrm{Unif}(0,1)$ be independent. Then

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

are independent $\mathcal{N}(0,1)$-distributed random variables.

Next step?

Follow von Neumann

… polynomial. This is, as I see, the method that you have in mind.

An alternative, which works if ξ and all values of $f(\xi)$ lie in 0, 1, is this: Scan pairs x^i, y^i and use or reject x^i, y^i according to whether $y^i \leq f(x^i)$ or not. In the first case, put $\xi^j = x^i$ in the second case form no ξ^j at that step.

The second method may occasionally be better th…

Rejection sampling

Recap: We have seen

Recap: We have seen

- Uniform random variate generation

Recap: We have seen

- Uniform random variate generation
- Direct sampling from variety of distributions

Recap: We have seen

- Uniform random variate generation
- Direct sampling from variety of distributions
  - Inversion method
    - Draw $U \sim \text{Unif}(0, 1)$
    - Compute $X = F^{-}(U) = \min\{x : F_X(x) \geq u\}$

Recap: We have seen

- Uniform random variate generation
- Direct sampling from variety of distributions
  - Inversion method
    - Draw $U \sim \text{Unif}(0, 1)$
    - Compute $X = F^{-}(U) = \min\{x : F_X(x) \geq u\}$
  - Transformation method.
    - Draw $U \sim \text{Unif}(0, 1)$,
    - Obtain $X = g(U)$ for some general transformation g.

Recap: We have seen

- Uniform random variate generation
- Direct sampling from variety of distributions
  - Inversion method
    - Draw $U \sim \text{Unif}(0, 1)$
    - Compute $X = F^{-}(U) = \min\{x : F_X(x) \geq u\}$
  - Transformation method.
    - Draw $U \sim \text{Unif}(0, 1)$,
    - Obtain $X = g(U)$ for some general transformation g.

However, those methods required a quite specific structure for us to be able to sample.

What if inverse of CDF or a nice transformation is not available?

What if inverse of CDF or a nice transformation is not available?

What if we cannot evaluate $\pi(x)$ – only evaluate an unnormalised density $\gamma(x)$ where

$$\pi(x) = \frac{\gamma(x)}{Z}.$$

What if inverse of CDF or a nice transformation is not available?

What if we cannot evaluate $\pi(x)$ – only evaluate an unnormalised density $\gamma(x)$ where

$$\pi(x) = \frac{\gamma(x)}{Z}.$$

Can we still do *exact* sampling?

The Fundamental Theorem of Simulation

Is there a more general structure?

Theorem 2 (Theorem 2.2, Martino et al., 2018)

Drawing samples from one dimensional random variable X with a density $\pi(x) \propto \gamma(x)$ is equivalent to sampling uniformly on the two dimensional region defined by

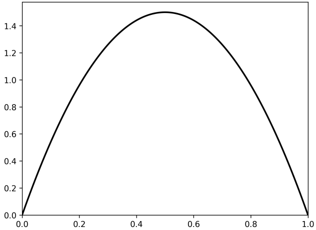
$$\mathrm{A} = \{(x, y) \in \mathbb{R}^2 : 0 \leq y \leq \gamma(x)\}. \tag{2}$$

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

Let

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

where $\Gamma(n) = (n-1)!$ for integers. For $\text{Beta}(2, 2)$:



Its maximum is 1.5 in this specific case. Can we sample uniformly?

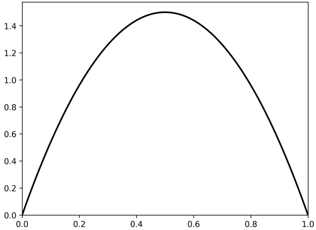
Sampling Beta density

Testing the theorem

Let

$$\pi(x) = \mathrm{Beta}(\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1},$$

where $\Gamma(n) = (n-1)!$ for integers. For $\mathrm{Beta}(2, 2)$:



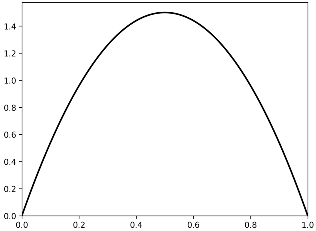
Its maximum is 1.5 in this specific case. Can we sample uniformly?

- Sample from the box $[0, 1] \times [0, 1.5]$ and keep the ones inside.

Let

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

where $\Gamma(n) = (n-1)!$ for integers. For $\text{Beta}(2, 2)$:

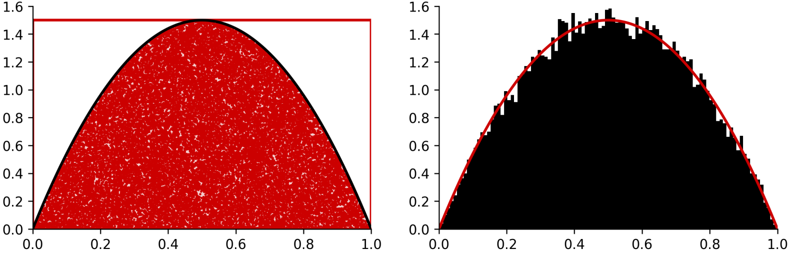


Its maximum is 1.5 in this specific case. Can we sample uniformly?

- Sample from the box $[0, 1] \times [0, 1.5]$ and keep the ones inside.
- Note though our aim is to 'test the x-marginal'

Sampling Beta density

Testing the theorem



Note the key aspects:

Note the key aspects:

- We have not used any inverse CDF or transformation but

Note the key aspects:

- We have not used any inverse CDF or transformation but
  - We have used the expression of $\pi(x)$

Note the key aspects:

- We have not used any inverse CDF or transformation but
  - We have used the expression of $\pi(x)$

We can get away with an unnormalised density $\gamma(x)$ (as FTS suggests).

Using a box wrapping the density is very inefficient:

Rejection sampling

More than a box

Using a box wrapping the density is very inefficient:

- You need the maximum of the density

$$p^\star = \max \pi(x)$$

which could be as hard as the sampling problem!

Rejection sampling

More than a box

Using a box wrapping the density is very inefficient:

- You need the maximum of the density

$$p^\star = \max \pi(x)$$

  which could be as hard as the sampling problem!
- For densities that are peaky, this could be wildly inefficient

Rejection sampling

More than a box

Using a box wrapping the density is very inefficient:

- You need the maximum of the density

$$p^\star = \max \pi(x)$$

  which could be as hard as the sampling problem!
- For densities that are peaky, this could be wildly inefficient

Idea: Design a *proposal* density that tightly wraps the target density

Rejection sampling

Choice of the proposal

Consider a (target) density $\pi(x)$ and a *proposal* density $q(x)$.

Consider a (target) density $\pi(x)$ and a *proposal* density $q(x)$.

For rejection sampling, we always choose a proposal such that

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

for $M \geq 1$.

Consider a (target) density $\pi(x)$ and a *proposal* density $q(x)$.

For rejection sampling, we always choose a proposal such that

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

for $M \geq 1$. Intuitively, the $Mq(x)$ curve should be **above** $\pi(x)$.

But how to obtain a sample under the curve?

But how to obtain a sample under the curve?

We can do

But how to obtain a sample under the curve?

We can do

- Sample $x' \sim q(x)$

Rejection sampling

The algorithm

But how to obtain a sample under the curve?

We can do

- Sample $x' \sim q(x)$
- Sample $u' \sim \text{Unif}(u'; 0, Mq(x'))$

Rejection sampling

The algorithm

But how to obtain a sample under the curve?

We can do

- Sample $x' \sim q(x)$
- Sample $u' \sim \text{Unif}(u'; 0, Mq(x'))$
- Accept if

$$u' \leq \pi(x'),$$

This would give us (x', u') uniformly under the curve (hence x' samples would be distributed w.r.t. $\pi(x)$)

Rejection sampling

The algorithm: A closer look

To implement the method:

- Sample $x' \sim q(x)$,

To implement the method:

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$

To implement the method:

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$
- Accept if

$$u \leq \frac{\pi(x')}{Mq(x')}.$$

The rejection sampler:

The rejection sampler:

- $X' \sim q(x)$,

Rejection sampling

The algorithm

The rejection sampler:

- $X' \sim q(x)$,
- Accept the sample X' with probability

$$a(X') = \frac{\pi(X')}{Mq(X')} \leq 1.$$

We have seen that rejection sampling works for general densities but requires the evaluation of $\pi(x)$.

We have seen that rejection sampling works for general densities but requires the evaluation of $\pi(x)$.

In many (many) cases, we cannot evaluate $\pi(x)$!

We have seen that rejection sampling works for general densities but requires the evaluation of $\pi(x)$.

In many (many) cases, we cannot evaluate $\pi(x)$!

Luckily, we can evaluate $\pi(x)$ up to a normalising constant:

$$\pi(x) = \frac{\gamma(x)}{Z}.$$

We have seen that rejection sampling works for general densities but requires the evaluation of $\pi(x)$.

In many (many) cases, we cannot evaluate $\pi(x)$!

Luckily, we can evaluate $\pi(x)$ up to a normalising constant:

$$\pi(x) = \frac{\gamma(x)}{Z}.$$

Note the terminology and convention:

We have seen that rejection sampling works for general densities but requires the evaluation of $\pi(x)$.

In many (many) cases, we cannot evaluate $\pi(x)$!

Luckily, we can evaluate $\pi(x)$ up to a normalising constant:

$$\pi(x) = \frac{\gamma(x)}{Z}.$$

Note the terminology and convention:

- $\gamma(x)$ is called the *unnormalised* density

We have seen that rejection sampling works for general densities but requires the evaluation of $\pi(x)$.

In many (many) cases, we cannot evaluate $\pi(x)$!

Luckily, we can evaluate $\pi(x)$ up to a normalising constant:

$$\pi(x) = \frac{\gamma(x)}{Z}.$$

Note the terminology and convention:

- $\gamma(x)$ is called the *unnormalised* density
- Z is called the normalising constant
  - It is a super important quantity for many other purposes

We have seen that rejection sampling works for general densities but requires the evaluation of $\pi(x)$.

In many (many) cases, we cannot evaluate $\pi(x)$!

Luckily, we can evaluate $\pi(x)$ up to a normalising constant:

$$\pi(x) = \frac{\gamma(x)}{Z}.$$

Note the terminology and convention:

- $\gamma(x)$ is called the *unnormalised* density
- Z is called the normalising constant
  - It is a super important quantity for many other purposes
- We write $\pi(x) \propto \gamma(x)$ to say p is proportional to $\gamma(x)$ but normalised to integrate (or sum) to one.

Rejection sampling

The algorithm: A closer look

To implement, choose M and q such that $\gamma(x) \leq Mq(x)$

- Sample $x' \sim q(x)$,

To implement, choose M and q such that $\gamma(x) \leq Mq(x)$

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$

To implement, choose M and q such that $\gamma(x) \leq Mq(x)$

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$
- Accept if

$$u \leq \frac{\gamma(x')}{Mq(x')}.$$

Rejection sampling

The algorithm: A closer look

To implement, choose M and q such that $\gamma(x) \leq Mq(x)$

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$
- Accept if

$$u \leq \frac{\gamma(x')}{Mq(x')}.$$

Exactly same – γ used instead of p provided that $\gamma(x) \leq Mq(x)$

Rejection sampler:

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$
- Accept if

$$u \leq \frac{\gamma(x')}{Mq(x')}.$$

Rejection sampler:

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$
- Accept if

$$u \leq \frac{\gamma(x')}{Mq(x')}.$$

In order for this algorithm to be implemented, we do not want many rejections (as we want many accepted samples to build our distribution).

Rejection sampler:

- Sample $x' \sim q(x)$,
- Sample $u \sim \text{Unif}(u; 0, 1)$
- Accept if

$$u \leq \frac{\gamma(x')}{Mq(x')}.$$

In order for this algorithm to be implemented, we do not want many rejections (as we want many accepted samples to build our distribution).

How to compute acceptance rate?

Proposition 1

When the target density $\pi(x)$ 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 π. For an unnormalised target density $\gamma(x)$ with the normalising constant $Z = \int \gamma(x)\mathrm{d}x$, the acceptance rate is given as

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

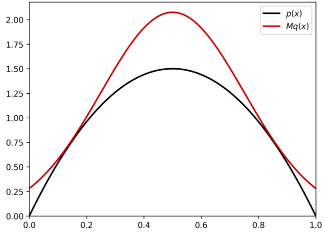
Rejection sampling

Examples: Same Beta(2, 2), better proposal

Choose

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

with $M = 1.3$.



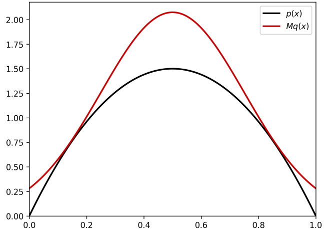
Rejection sampling

Examples: Same Beta(2, 2), better proposal

Choose

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

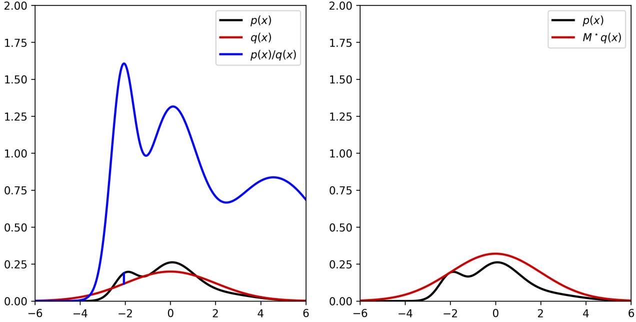
with $M = 1.3$.



Simulation.

A standard choice for M is

$$M^\star = \sup_x \frac{\pi(x)}{q(x)}.$$



Given $\mathcal{N}(x; 0, 1)$, suppose we are interested in sampling this density between $[-a, a]$. We can write this truncated normal density as

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

We can choose $q(x) = \mathcal{N}(x; 0, 1)$ anyway, and we have $\gamma(x) \leq q(x)$ (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 $[-a, a]$.

We have covered the rejection sampler:

- Sample $X' \sim q(x)$
- Sample $U \sim \text{Unif}(0,1)$
- If $U \leq \gamma(X')/Mq(X')$,
  - Accept X'

We have covered the rejection sampler:

- Sample $X' \sim q(x)$
- Sample $U \sim \text{Unif}(0,1)$
- If $U \leq \gamma(X')/Mq(X')$,
  - Accept X'

While very popular in 90s, it is extremely hard to compute M for modern large scale problems.

See you next week!

Thanks!

References I

Martino, Luca, David Luengo, and Joaquín Míguez (2018). *Independent random sampling methods*. Springer.