Generation of Random Numbers

Module DES130: Computational Statistics

Dr. Erick A. Chacón Montalván (echacon@uni.edu.pe)

Escuela de Profesional de Ingeniería Estadística
Facultad de Ingeniería Económica, Estadística y Ciencias Sociales
Universidad Nacional de Ingeniería (UNI)
Lima – Perú

Introduction

Generation of random numbers

Definition

Let X be a random variable with mass probability function $\Pr(X = x)$ or density function $f_X(x)$. The process of generating random numbers it to **obtain a realization** x of the random variable X.

Extension

We will also consider some situations where the density function is known up to a constant of proportionality such as

$$f(x) \propto q(x).$$

Applications

- **Monte Carlo methods**: generating random numbers are essential.
- **Bayesian inference**: it is required to generate random numbers when the posterior distribution is not analitically tractable.
- **Inference in physical models**: inference is done by simulating surrogate data from physical models like *aproximate bayesian computation*.
- **Stochastic processes**: random numbers are essential for simulating more complex stochastic processes.
- **Cryptography**: generate secure random numbers that are not easy to guess.

How to generate random numbers?

Statement (Probability integral transform)

Let X be a continuous random variable with cumulative distribution function $F_X(x)$.

$$\text{Let } Y := F_X(X), \text{ then } Y \sim \text{Uniform}(0, 1).$$

Proof

$$\begin{aligned} F_Y(y) &= \Pr(Y \leq y) \\ &= \Pr(F_X(X) \leq y) \\ &= \Pr\left(X \leq F_X^{-1}(y)\right) \\ &= F_X(F_X^{-1}(y)) \\ &= y. \end{aligned}$$

How to generate random numbers?

Exponential example

Let $X \sim \text{Exponential}(1)$, then $F(x) = 1 - \exp(-x)$. Hence

$$Y = 1 - \exp(-X) \sim \text{Uniform}(0, 1).$$

Steps to generate random numbers

Step 1: Generate realizations $y_i \overset{iid}{\sim} \text{Uniform}(0, 1)$.

Step 2: Apply some transformation $x_i = g(y_i) : x_i \sim f(x_i)$.

Uniform random number generators

Generation of uniform random numbers

Goal

In this section we deal with the topic of generating realizations

$$x_i \overset{iid}{\sim} \text{Uniform}(0, 1).$$

The mechanisms to obtain such realizations are usually known as **random number generators** (RNG).

Types of generators:

- **Physical generator**: realizations are obtained by observing physical processes.
- **Deterministic generator**: realizations are obtained by deterministic algorithms.

Physical generators

- Realizations are obtained by observing physical processes such as timing of succesive events, particle trajectories, so on.
- A test is required to evaluate if such realizations are a good approximation of iid uniform random variables.
- Disadvantages: Not reproducible, slow, and expensive.

Generators using deterministic algorithms

Definition

An RNG consists usually has the following structure

$$(\mathcal{S}, \mu, f, \mathcal{U}, g),$$

where

- $\mathcal{S}$ is a finite set of states,
- μ is a probability distribution on $\mathcal{S}$ to set an *initial state*,
- $f : \mathcal{S} \to \mathcal{S}$ is the *transition function*,
- $\mathcal{U}$ is the output space,
- $g : \mathcal{S} \to \mathcal{U}$ is the *output function*.

Details can be found at Gentle, Härdle, and Mori (2012).

Generators using deterministic algorithms

Algorithm

The algorithm consist on:

Step 1: generate the initial state s_0 using μ or set-up a *seed*,

For $i = 1, 2, \ldots, n$:

Step 2: generate $s_i = f(s_{i-1})$,

Step 3: generate $u_i = g(s_i)$.

The values $u_1, u_2, \ldots, u_n$ are considered *pseudo-random numbers* coming from a Uniform$(0, 1)$.

Period of an RNG

Is defined as the smallest positive j such as $s_{i+j} = s_i$. Good RNG have periods close to the upper bound $n(\mathcal{S})$.

Good properties of deterministic RNG

- *Long periods* to use most numbers of the state space $\mathcal{S}$.
- *Efficient* to run fast and use small memory.
- Able to *reproduce* the same sequence of numbers.
- *Portable* to work in different softwares and hardwares.

Statistical testing

RNGs have to pass a set of empirical statistical tests with **hypothesis**:

H_0 :μ_i are realizations of iid Uniform$(0, 1)$ random variables.

H_1 :μ_i are not realizations of iid Uniform$(0, 1)$ random variables.

For an statistic Y, they usually compute the **p-value**

$$p = \Pr(Y \geq y \mid H_0),$$

(L'Ecuyer and Simard 2007).

Decision:

- If p is extremely small, then the RNG fails.
- If p is not very close to 0 or 1, the RNG does not fails.
- If p is ambiguous the test proceeds until it becomes obvious.

Linear recurrence

The most popular RNGs use the following linear recurrence

$$x_i = (a_i x_{i-1} + \cdots + a_k x_{i-k}) \mod m.$$

with *modulus* m, *order* k and *coefficients* $a_1, \ldots, a_k \in \mathbb{Z}_m$. For m prime, the coefficients can be choosen to obtain a period $m^k - 1$ which is the largest possible value leading to

$$x_i = (a_r x_{i-r} + a_k x_{i-k}) \mod m.$$

Generators using linear recurrence

Multiple recursive generator (MRG)

Uses linear recurrence with a large value for m and defines

$$u_i = x_i/m.$$

Linear congruential generator (LCG)

Is a particular case for $k = 1$ such as

$$x_i = ax_{i-1} \mod m, \text{ and } u_i = x_i/m.$$

Example: $x_i = 65539x_{i-1} \mod 2^{31}$.

Other generators:

GFSR, Mersenne Twister, Xoshiro256++ (Julia's default). See Gentle, Härdle, and Mori (2012).

Non-uniform random number generators

Generation of non-uniform random numbers

Goal

In this section we deal with the topic of generating realizations

$$x_i \overset{iid}{\sim} f(x),$$

using realizations u_i from iid Uniform$(0, 1)$ random variables. The mechanisms to obtain such realizations are usually known as *sampling methods*.

Inverse CDF method

Let X be a continuous random variable with cumulative distribution function $F_X(x)$. Then

$$F_X(X) = U, \text{ where } U \sim \text{Uniform}(0, 1)$$
$$X = F_X^{-1}(U)$$
$$x_i = F_X^{-1}(u_i).$$

Exponential

$$F_X(X) = 1 - \exp\left(-\frac{X}{\theta}\right) = U$$
$$X = -\theta \ln(1 - U) = F_X^{-1}(U)$$
$$x_i = -\theta \ln(1 - u_i).$$

Inverse CDF method

Pareto

$$F_X(X) = 1 - \left(\frac{\lambda}{X}\right)^k = U$$

$$X = \frac{\lambda}{(1-U)^{1/k}} = F_X^{-1}(U)$$

$$x_i = \frac{\lambda}{(1-u_i)^{1/k}}.$$

This method can be applied for random variables with analitically known CDF such as Weibull, Geometric, so on. Cases like the normal, t-Student, chi-square good numerical approximations can be obtained for the inverse CDF.

Rejection sampling: algorithm

Let consider $\pi(x) = kJ(x)$, where $\pi(x)$ is a probability density and k is the normalising constant. We can obtain a sample from $\pi(x)$ using $J(x)$.

Algorithm:

1. Define a proposal $q(x)$ that holds the envelope property such as there is a constant M that holds the following condition:
$$J(x) \leq Mq(x)$$
2. Draw a sample $x \sim q(x)$
3. Compute the probability of acceptance:
$$r(x) = \frac{J(x)}{Mq(x)}$$

4. Accept x with probability $r(x)$

Rejection sampling: proof

$$
\begin{aligned}
\Pr(x \mid I = 1) &= \frac{q(x)\Pr(I = 1 \mid x)}{\Pr(I = 1)} \\
&= \frac{q(x)\Pr(I = 1 \mid x)}{\int q(x)\Pr(I = 1 \mid x)\, dx} \\
&= \frac{q(x)\frac{J(x)}{Mq(x)}}{\int q(x)\frac{J(x)}{Mq(x)} dx} \\
&= \frac{\frac{J(x)}{M}}{\int \frac{J(x)}{M} dx} \\
&= \frac{\frac{\pi(x)}{kM}}{\int \frac{\pi(x)}{kM} dx} = \frac{\frac{\pi(x)}{kM}}{\frac{1}{kM}} \\
&= \pi(x).
\end{aligned}
$$

Rejection sampling: Beta example

Let $X \sim Beta(\alpha, \beta)$ such as

$$f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1} \text{ for } 0 \leq x \leq 1$$
$$\propto x^{\alpha-1}(1-x)^{\beta-1} = J(x).$$

Let's use:

- $q(x) = \text{Uniform}(0, 1)$
- $x_{mode} = \arg\max J(x) = \frac{\alpha-1}{\alpha+\beta-2}$
- $M = J(x_{mode})$
- $r(x) = \frac{J(x)}{M} = \frac{J(x)}{J(x_{mode})}$

Rejection sampling: Point process example

Let consider the ocurrence of events in a period of time $[0, 10]$ with intensity function $\lambda(t) = sin(t) + 1$. Simulate 200 events of from this temporal point process.

Composition or convolution method

Let the random variables $X \sim \pi(x)$ such as:

$$\pi(x) = \sum_{i=1}^{k} p_i \pi_i(x)$$

$$\pi(x) = \int_y \pi(y)\pi(x \mid y)dy$$

Algorithm

Step 1: Generate $I = i$ with probability p_i. Or generate y from $\pi(y)$.

Step 2: Generate x from $\pi_i(x)$. Or generate x from $\pi(x \mid y)$.

Bivariate normal standard

Algorithm

Step 1: Generate u_1 and u_2 from Uniform$(0, 1)$.

Step 2: Compute $w = u_1^2 + u_2^2$.

Step 3: Accept u_1 and u_2 if $w \leq 1$.

Step 4: Compute $z = \sqrt{(-2\log(w))/w}$

Step 5: Compute $x_1 = u_1 z$ and $x_2 = u_2 z$.

More examples can be found in (Härdle, Okhrin, and Okhrin 2017).

Generating random vectors

Multivariate normal vector

Let $\boldsymbol{X}_q \sim N_q(\mu, \boldsymbol{\Sigma})$. If we decompose $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$, we can show that $\mu + \boldsymbol{L}\boldsymbol{Z} \sim N_q(\mu, \boldsymbol{\Sigma})$

Algorithm

1. Compute the Cholesky factorization $\boldsymbol{L}$ such as $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$.
2. Simulate a realization $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{I})$.
3. Compute $\mu + \boldsymbol{L}\boldsymbol{z}_i$.

Example: temporal Gaussian process

Sea $\{X(t) : t \in \{1, 2, \ldots, 100\}\}$ such as $E(X_t) = 0$,
$Cov(X(t), X(t')) = 9 \exp(-\frac{|t-t'|}{10})$.

Other examples

- Simulating correlation matrices (Whisart, Lkj distributions.).
- Simulating spatial point processes.
- Simulating Gaussian markov random fields.
- Simulating graphs or other more general objects (images).

References

References

Gentle, James E. 2009. *Computational Statistics*. Springer Science & Business Media.

Gentle, James E., Wolfgang Karl Härdle, and Yuichi Mori. 2012. *Handbook of Computational Statistics: Concepts and Methods*. Springer Science & Business Media.

Härdle, Wolfgang Karl, Ostap Okhrin, and Yarema Okhrin. 2017. *Basic Elements of Computational Statistics*. Springer.

L'Ecuyer, Pierre, and Richard Simard. 2007. "TestU01: A C Library for Empirical Testing of Random Number Generators." *ACM Transactions on Mathematical Software* 33 (4): 22:1–40. https://doi.org/10.1145/1268776.1268777.