Non-uniform random number generation in a computational framework with arbitrary finite precision arithmetic

Claude Gravel

March 22, 2020

This small set of routines perform exact random number generation of non-uniform discrete probability distributions up to an arbitrary finite precision of the probabilities. The precision and the binary representations are specified by the user. The algorithm can be found in Knuth and Yao [6]. The only limitation is the storage given the possibility to compute with arbitrary large finite precision the components of the probability vector. The routines use only functions from the C++ standard library including the generator for the source of raw (pseudo)-random unbiased i.i.d. bits. Some examples of probability distributions are given in the binary files together with this routine. Examples are explained below.

A lower bound on the time complexity can be given by the number of random coins needed to generate a given distribution. The last chapter 15 of Devroye [3] discuss the generation of random variables from a discrete source of i.i.i. unbiased bits. Lumbroso's thesis [2] explains how to implement efficiently Knuth and Yao's algorithm for the uniform discrete distribution under the name "The Dice Roller" by analogy to a multi-faceted non-biased dice.

A p-bit number is a representation for a real number x given by a pair of mantissa and exponent (m, p) where m is an odd integer such that $x = m2^{-p}$. For instance, the class RR from the NTL library [8], allows for arbitrary precision arithmetic by letting a user to specify p. *From NTL page*: $\triangleright$ The real number 0 is represented by $(0, 0)$. All arithmetic operations are implemented so that the effect is as if the result was computed exactly, and then rounded to p bits. If a number lies exactly half-way between two p-bit numbers, the "round to even" rule is used. So in particular, the computed result will have a relative error of at most 2^{-p}. The previous rounding rules apply to all arithmetic operations in this module, except for the following routines:

1. The transcendental functions: log, exp, log10, expm1, log1p, pow, sin, cos, ComputePi
2. The power function

3. The input and ascii to RR conversion functions when using "e"-notation

For these functions, a very strong accuracy condition is still guaranteed: the computed result has a relative error of less than 2^{-p+1} (and actually much closer to 2^{-p}). That is, it is as if the resulted were computed exactly, and then rounded to one of the two neighboring p-bit numbers (but not necessarily the closest). $\lhd$

The code given here only relies on the standard C++ library and let the user manage the computation of the binary representations. A few binary files described in the example below are given to test the algorithm and its speed. Details about the routines are contained in the C++ file "knuth_yao_1976_sampling_algo.cpp".

1 Examples

In the examples below, the user who executes the code on a file mentioned below is prompted for a sample size and the filename. Each file mentioned below contains the binary representation of the probability distribution. The estimation of the expected number of random i.i.d. unbiased bits needed per random variables is to be compared with the exact numerical entropy mentioned below. In all cases, for a not too big sample, the estimated number of random coins fall within the bounds given by [6] that is between H and $H+2$ where H is the entropy.

1.1 Discrete gaussian

The discrete gaussian mass function is given by

$$p_n = Ce^{-(\frac{n-\alpha}{\beta})^2} \text{ for } n \in \mathbb{Z}.$$

Given the impossibility to store an infinite support and the difficulty to compute exactly the normalization constant C, a truncation is considered for $\lfloor \alpha - 10\beta \rfloor \leq n \leq \lceil \alpha + 10\beta \rceil$.

The file "dis_nor.bit" contains the binary representation for $\alpha = e^{-1}$ and $\beta = 1000$. The minimal accuracy for each probability value is 1000 bits. The truncated range is $[-10001, 10001] \subset \mathbb{Z}$. The numerical entropy with given accuracy is therefore $11.512879869842728146\ldots$

The discrete gaussian is special case of the theta family for which $(\frac{n-\alpha}{\beta})^2$ is replaced by $-|\frac{n-\alpha}{\beta}|^k$ for $k \geq 1$. The case of $k = 1$ is a discrete analog to the Laplace distribution.

The discrete gaussian with large values of beta occurs in application like lattice-based cryptography [7].

1.2 Binomial

The file "bino.bit" contains the binary representation for binomial distribution with 2000 trials and occurrence probability 0.1. The minimal accuracy for each

probability value is 1000 bits. The numerical entropy is 5.7925934431983804672...

1.3 Zeta-Dirichlet related

For $u > 0$, consider

$$C_u = \sum_{n=3}^{\infty} \frac{1}{n(\log n)^{1+u}},$$

and the probabilities

$$p_n = \frac{1}{C_u} \frac{1}{n(\log n)^{1+u}} \text{ for } n \geq 3.$$

The entropy is unbounded for all $0 < u \leq 1$ and bounded for $u > 1$. (For $u \leq 0$, the sum C_u diverges. See Hardy and Riesz [5].)

A truncation with upper cutoff point $n = 2^{20}$ and with $u = 0.05$ is considered with proper re-normalization. Note that because the truncation has a finite support, then its entropy exists. The file containing the binary representations is "zeta.bit". The minimal accuracy is set at 1000 bits. For $0 < u \leq 1$, the entropy diverges slowly. The numerical entropy of the truncated probability distribution is 10.189437261652963733...

1.4 A basic three-mass distribution

We consider the probability vector (p_0, p_1, p_2) where

$$\begin{aligned} p_0 &= \frac{1}{\pi} \\ &= (0.010100010111110\ldots)_2. \\ p_1 &= \frac{1}{e} \\ &= (0.010111100010110\ldots)_2. \\ p_2 &= 1 - p_1 - p_2 \\ &= (0.010100000101010\ldots)_2. \end{aligned}$$

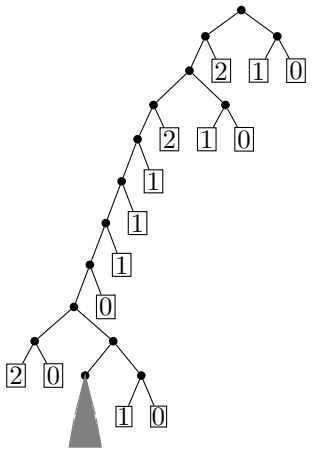
The execution the of the sampling algorithm is represented on Figure 1.4. The entropy is 1.581127402961993034...

1.5 Discretization of a singular continuous distribution

For $j \geq 1$, let X_j be non-identically, independently distributed Bernoulli random variable with

$$\begin{aligned} \mathbf{P}\{X_j = 0\} &= 1 - p_j \\ &= \frac{1}{e^{-1/2^j} + 1}, \end{aligned}$$

Figure 1: DDG tree



$$\begin{aligned}\mathbf{P}\{X_j = 1\} &= p_j \\ &= \frac{e^{-1/2^j}}{e^{-1/2^j} + 1}.\end{aligned}$$

Consider the continuous random variables

$$\begin{aligned}X &= \sum_{j=1}^{\infty} 2^{-2j} X_{2j}, \\ Y &= \sum_{j=1}^{\infty} 2^{-(2j-1)} X_{2j-1} \text{ and} \\ Z &= X + Y.\end{aligned}$$

By Kakutani's theorem [1], X and Y are singular continuous random variables. The random variable Z is a truncated exponential on the real interval $[0, 1]$ and hence is absolutely continuous.

We can discretize X to a certain desired accuracy, 15 bits of accuracy accuracy say, for which the corresponding discretization has 2^{15-1} atoms. Each atom is represented with 1000 bits precision. The file "sing.bit" contains the discretization of X.

To discretize a singular distribution is generally easier than to discretize an absolutely continuous distribution since it does not require numerical integration or known analytic formulas for the distribution function.

1.6 A quantum mechanical discrete distribution

Let $n > 1$, the sets of angles $\{\theta_j\}_{j=1}^n$ and $\{\varphi_j\}_{j=1}^n$. For $b = (b_1, \ldots, b_n) \in \{-1, +1\}^n$, an example of a quantum mechanical distribution from [4] is given

by

$$p(b) = \cos^2\left(\frac{\theta}{2}\right) p_1(b) + \sin^2\left(\frac{\theta}{2}\right) p_2(b) \text{ with} \tag{1}$$

$$\theta = \sum_{j=1}^{n} \theta_j \ ,$$

$$p_1(b) = \frac{1}{2}\big(\mathrm{a}_1(b) + \mathrm{a}_2(b)\big)^2 \ ,$$

$$p_2(b) = \frac{1}{2}\big(\mathrm{a}_1(b) - \mathrm{a}_2(b)\big)^2 \ , \tag{2}$$

$$\mathrm{a}_1(b) = \prod_{j=1}^{n} \cos\left(\tfrac{1}{2}\big(\varphi_j - \tfrac{\pi}{2} b_j\big)\right) \ ,$$

$$\mathrm{a}_2(b) = \prod_{j=1}^{n} -\sin\left(\tfrac{1}{2}\big(\varphi_j - \tfrac{\pi}{2} b_j\big)\right). \tag{3}$$

We observe that p is a convex combination of both p_1 and p_2.

The file "ghz.bit" contains the binary representations of $p(b)$ up to 1000 bits of accuracy for $n = 15$ with

$$\varphi_j = \begin{cases} {}^{\pi}/_{6} & \text{if } j \equiv 0 \pmod 3, \\ {}^{3\pi}/_{6} & \text{if } j \equiv 1 \pmod 3, \\ {}^{5\pi}/_{6} & \text{if } j \equiv 2 \pmod 3. \end{cases}$$

$$\theta_j = \begin{cases} {}^{2\pi}/_{10} & \text{if } j \equiv 0 \pmod 5, \\ {}^{6\pi}/_{10} & \text{if } j \equiv 1 \pmod 5, \\ {}^{10\pi}/_{10} & \text{if } j \equiv 2 \pmod 5, \\ {}^{14\pi}/_{10} & \text{if } j \equiv 3 \pmod 5, \\ {}^{18\pi}/_{10} & \text{if } j \equiv 4 \pmod 5. \end{cases}$$

The numerical entropy is $9.1127812445913279409\ldots$

References

[1] Shizuo Kakutani. On equivalence of infinite product measures. *Annals of Mathematics*, pages 214–224, 1948.

[2] Jérémie Lumbroso. *Probabilistic Algorithms for Data Sreaming and Random Generation*. PhD thesis, Université Pierre et Marie Curie - Paris 6, 2012.

[3] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[4] Gilles Brassard, Luc Devroye and Claude Gravel. Exact classical simulation of the quantum-mechanical GHZ distribution. *IEEE Trans. Inf. Theory*, 62(2):876–890, 2016.

[5] G.H. Hardy and M. Riesz. *The General Theory of Dirichlet's Series*. Cambridge Tracts in Mathematics and Mathematical Physics. Dover Publications, 2005.

[6] Donald E. Knuth and Andrew Chi-Chih Yao. The complexity of nonuniform random number generation. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results.*, pages 357–428, New York, 1976. Carnegie-Mellon University, Academic Press.

[7] Chris Peikert. A decade of lattice cryptography. `https://web.eecs.umich.edu/~cpeikert/pubs/lattice-survey.pdf`, 2016.

[8] Victor Shoup. NTL: A library for doing number theory. `https://www.shoup.net/ntl/`. Last checked on March 18, 2020.