Accurate Functions for Probability Distributions

This version of the document is dated 2020-07-04.

Peter Occil

1 Introduction

This page is intended to contain implementations of probability density and inverse cumulative distribution functions (inverse CDFs) for popular probability distributions, such as the normal, gamma, and beta distributions. The difference here is that they return accurate answers to an arbitrary precision (rather than just for the commonly-used floating-point formats called *binary32* and *binary64*). They are needed in order to implement the rejection sampling and inversion methods described by Devroye and Gravel ((2016)**(1)** and (2015)**(2)**, respectively), which generate arbitrary-precision random numbers that follow a distribution as closely as possible. Both samplers are **implemented in Python** as the `numbers_from_dist` and the `numbers_from_dist_inversion` methods, respectively.

A word about probability density functions and inverse CDFs:

- The ***probability density function***, or *PDF*, is, roughly and intuitively, a curve of weights 0 or greater, where for each number, the greater its weight, the more likely a number close to that number is randomly chosen.
- The *inverse CDF* (also known as *quantile function*) maps numbers in the interval [0, 1) to numbers in the distribution, from low to high.

Both functions normally take one number in and put one number out, but in the case of the sampling methods by Devroye and Gravel, special versions of these functions are needed. Specifically:

- For the rejection sampler, `EPDF(min_x, max_x, precision)` calculates tight bounds of the PDF anywhere in a region of interest, namely, in the interval [`min_x` * $2^{-\texttt{precision}}$, `max_x` * $2^{-\texttt{precision}}$]. `EPDF` returns two values: the first is the greatest lower bound of the PDF anywhere in the region of interest, and the second is the least upper bound of the PDF anywhere in that region.
- For the inversion sampler, `EICDF(u, ubits, digitplaces)` calculates an accurate approximation to the quantile. Specifically, `EICDF` returns a number that is within $\texttt{base}^{-\texttt{digitplaces}}$ of the true quantile of `u` * $\texttt{base}^{-\texttt{ubits}}$, and is monotonic for a given value of `digitplaces`. `base` is the digit base of the accuracy, and `numbers_from_dist_inversion` can take any digit base (such as 2 for binary or 10 for decimal).

2 Functions

Generally, `EICDF` can make use of interval arithmetic, which provides rigorous error bounds on the result of calculations. The following Python code is an example of `EICDF` that produces an accurate approximation of the exponential quantile, to within $10^{-\texttt{precision}}$, using interval arithmetic.

```
def expoicdf(u, ubits, precision):
    """ Inverse CDF for the exponential distribution, implemented
         accurately using interval arithmetic. """
    intv = Fraction(u, 10 ** ubits)
    threshold = Fraction(1, 10 ** precision)
    while True:
       ret = 1 - intv
       ret = -(Interval(ret, prec=precision).log())
       if ret.isAccurateTo(threshold):
           return ret.midpoint()
       precision += 8
```

3 Notes

(1) Devroye, L., Gravel, C., "The expected bit complexity of the von Neumann rejection algorithm", arXiv:1511.02273v2 [cs.IT], 2016.

(2) Devroye, L., Gravel, C., "**Sampling with arbitrary precision**", arXiv:1502.02539v5 [cs.IT], 2015.

4 License