Inverse CDF Sampling

Troy Stribling

July 8, 2018

1 Introduction

Inverse CDF sampling is a method for obtaining samples from both discrete and continuous probability distributions that requires the CDF to be invertible. The method assumes values of the CDF are Uniform random variables on [0, 1]. CDF values are generated and used as input into the inverted CDF to obtain samples with the distribution defined by the CDF.

2 Sampling Discrete Distributions

A discrete probability distribution consisting of a finite set of N probability values is defined by, $\{p_i\}_N = \{p_1, p_2, \ldots, p_N\}$ with $p_i \geq 0, \forall i$ and $\sum_{i=1}^{N} p_i = 1$. The CDF specifies the probability that $i \leq n$ and is given by,

$$P(i \leq n) = P(n) = \sum_{i=1}^{n} p_i, \tag{1}$$

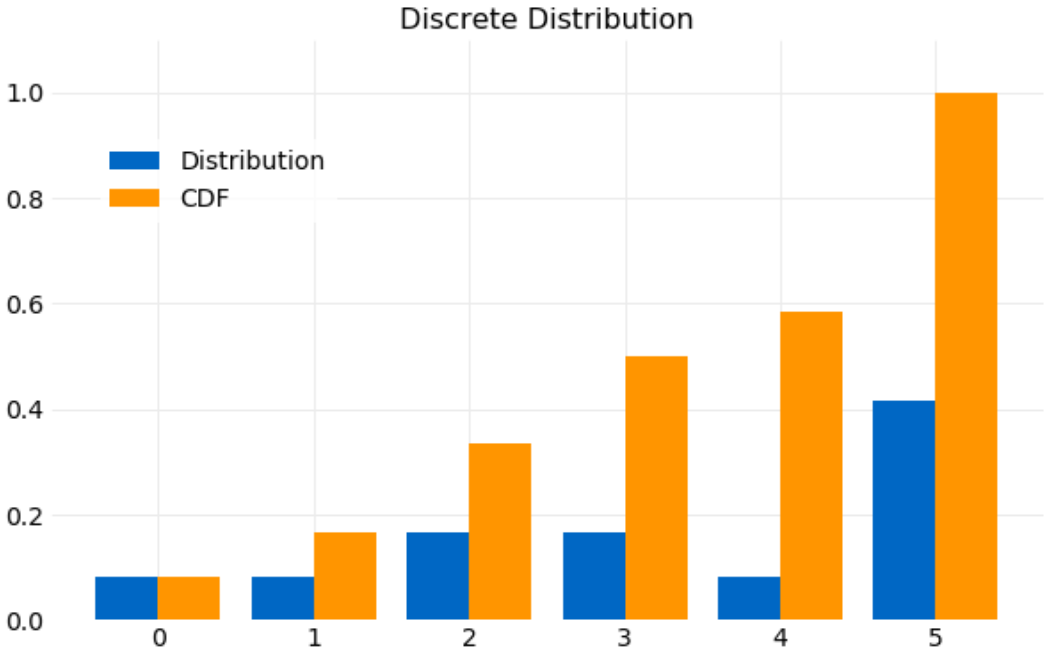
where $P(N) = 1$.

For a given generated CDF value, U, Equation (1) can always be inverted by evaluating it for each n and searching for the value of n that satisfies, $P(n) \geq U$. It can be seen that the generated samples will have distribution $\{p_i\}_N$ since the intervals $P(n) - P(n-1) = p_n$ are Uniformly sampled.

Consider the example distribution,

$$\left\{\frac{1}{12}, \frac{1}{12}, \frac{1}{6}, \frac{1}{6}, \frac{1}{12}, \frac{5}{12}\right\} \tag{2}$$

It is shown in the following plot with its CDF. Note that the CDF is a monotonically increasing function.



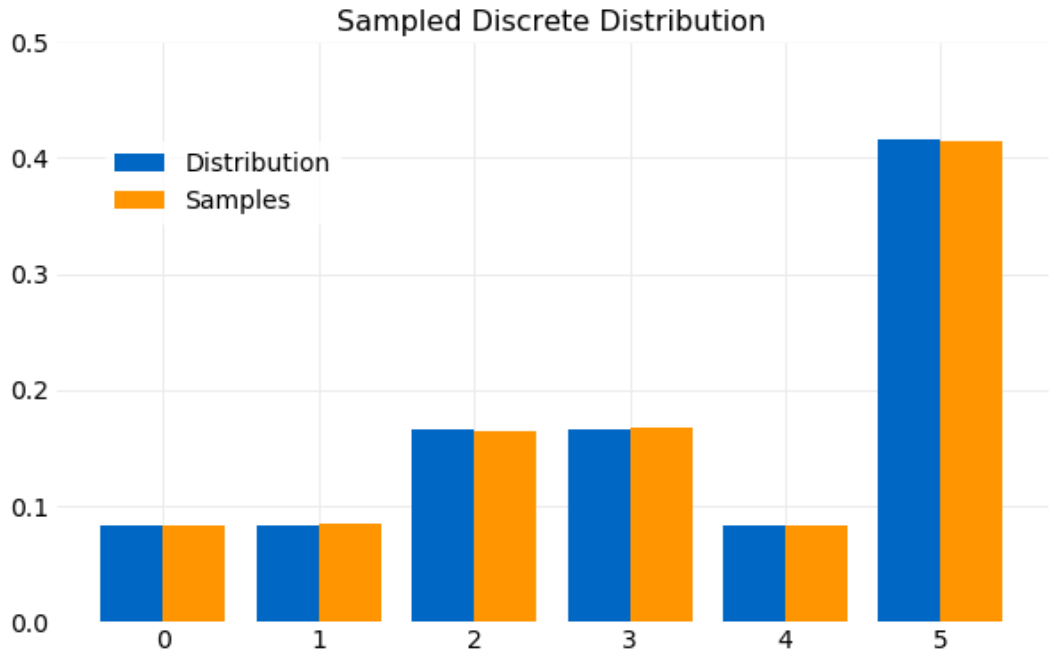
A sampler using the Inverse CDF method on the distribution $\{p_i\}_N$ implemented in Python is shown below. The program first stores the CDF computed from each of the sums $P(n)$ in an array. Next, CDF samples using $U \sim \textbf{Uniform}(0, 1)$ are generated. Finally, for each sampled CDF value, U, the array containing $P(n)$ is scanned for the value of n where $P(n) \geq U$. The resulting values of n will have the distribution $\{p_i\}_N$.

```
import numpy

n = 10000
df = numpy.array([1/12, 1/12, 1/6, 1/6, 1/12, 5/12])
cdf = numpy.cumsum(df)

samples = [numpy.flatnonzero(cdf >= u)[0] for u in numpy.random.rand(n)]
```

The figure below favorably compares generated samples and distribution (2),



It is also possible to directly sample $\{p_n\}$ using the `multinomial` sampler from `numpy`,

```
import numpy

n = 10000
df = numpy.array([1/12, 1/12, 1/6, 1/6, 1/12, 5/12])
samples = numpy.random.multinomial(n, df, size=1)/n
```

The number of operations required for generating samples using Inverse CDF sampling from a discrete distribution will scale $O(N_{samples}N)$ where $N_{samples}$ is the desired number of samples and N is the number of terms in the discrete distribution.

3 Sampling Continuous Distributions

A continuous probability distribution is defined by the PDF, $f_X(x)$, where $f_X(x) \geq 0, \forall x$ and $\int f_X(x)dx = 1$. The CDF is a monotonically increasing function that specifies the probability that $X \leq x$, namely,

$$P(X \leq x) = F_X(x) = \int^x f_X(w)dw. \tag{3}$$

3.1 Proof that Inverse CD Sampling Works

To prove that Inverse CDF sampling works for continuous distributions it must be shown that,

$$P[F_X^{-1}(U) \leq x] = F_X(x), \tag{4}$$

where $F_X^{-1}(x)$ is the inverse of $F_X(x)$ and $U \sim \textbf{Uniform}(0, 1)$.

A more general result needed to complete this proof is obtained using a change of variable on a CDF. If $Y = G(X)$ is a monotonically increasing invertible function of X then,

$$P(X \leq x) = P(Y \leq y) = P[G(X) \leq G(x)]. \tag{5}$$

To prove this note that $G(x)$ is monotonically increasing so the ordering of values is preserved,

$$X \leq x \implies G(X) \leq G(x).$$

Consequently, the order of the integration limits is maintained by the transformation. Further, since $G(x)$ is invertible, $x = G^{-1}(y)$ and $dx = \frac{dG^{-1}}{dy}dy$, so

$$\begin{aligned} P(X \leq x) &= \int^x f_X(w)dw \\ &= \int^y f_X(G^{-1}(z))\frac{dG^{-1}}{dz}dz \\ &= \int^y f_Y(z)dz \\ &= P(Y \leq y) \\ &= P[G(X) \leq G(x)], \end{aligned}$$

where,

$$f_Y(y) = f_X(G^{-1}(y))\frac{dG^{-1}}{dy}$$

The desired proof of Equation (4) follows from Equation (5) by noting that $U \sim$ **Uniform**$(0, 1)$ so $f_U(u) = 1$,

$$\begin{aligned} P[F_X^{-1}(U) \leq x] &= P[F_X(F_X^{-1}(U)) \leq F_X(x)] \\ &= P[U \leq F_X(x)] \\ &= \int_0^{F_X(x)} f_U(w)dw \\ &= \int_0^{F_X(x)} dw \\ &= F_X(x). \end{aligned}$$

3.2 Example

Consider the Weibull Distribution, with density

$$f_X(x; k, \lambda) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{\left(\frac{-x}{\lambda}\right)^k} & x \geq 0 \\ 0 & x < 0, \end{cases}$$

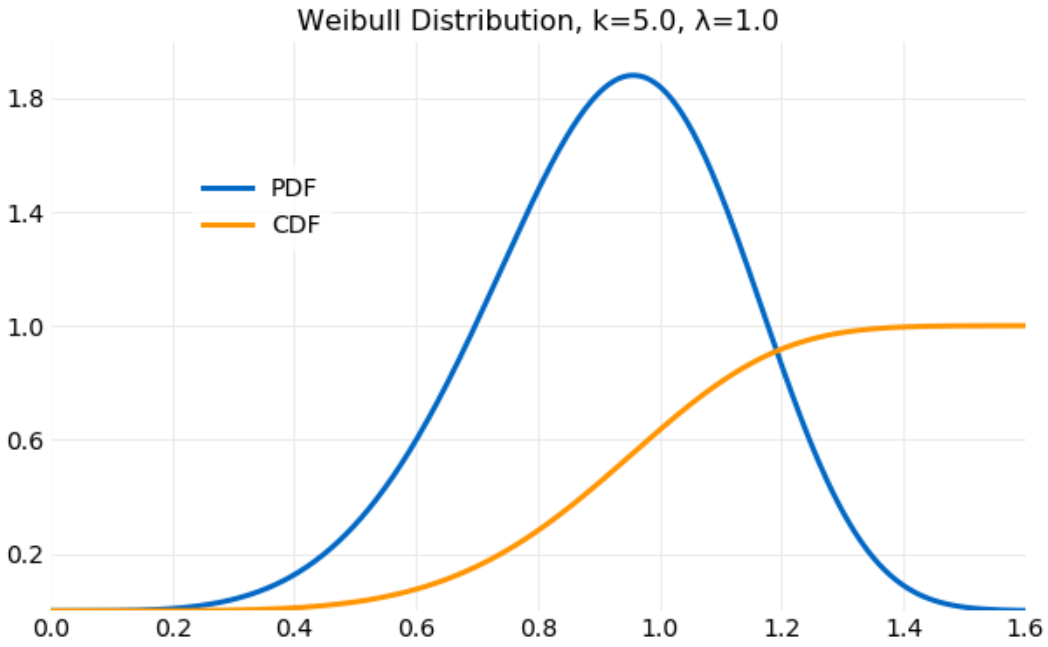
where k is the shape parameter and λ the scale parameter. The CDF is given by,

$$F_X(x; k, \lambda) = \begin{cases} 1 - e^{\left(\frac{-x}{\lambda}\right)^k} & x \geq 0 \\ 0 & x < 0. \end{cases}$$

The CDF can be inverted to yield,

$$F_X^{-1}(u; k, \lambda) = \begin{cases} \lambda \ln\left(\frac{1}{1-u}\right)^{\frac{1}{k}} & 0 \leq u \leq 1 \\ 0 & u < 0 \text{ or } u > 1. \end{cases}$$

In the example described here it will be assumed that $k = 5.0$ and $\lambda = 1.0$. The following plot shows the PDF and CDF using these values.



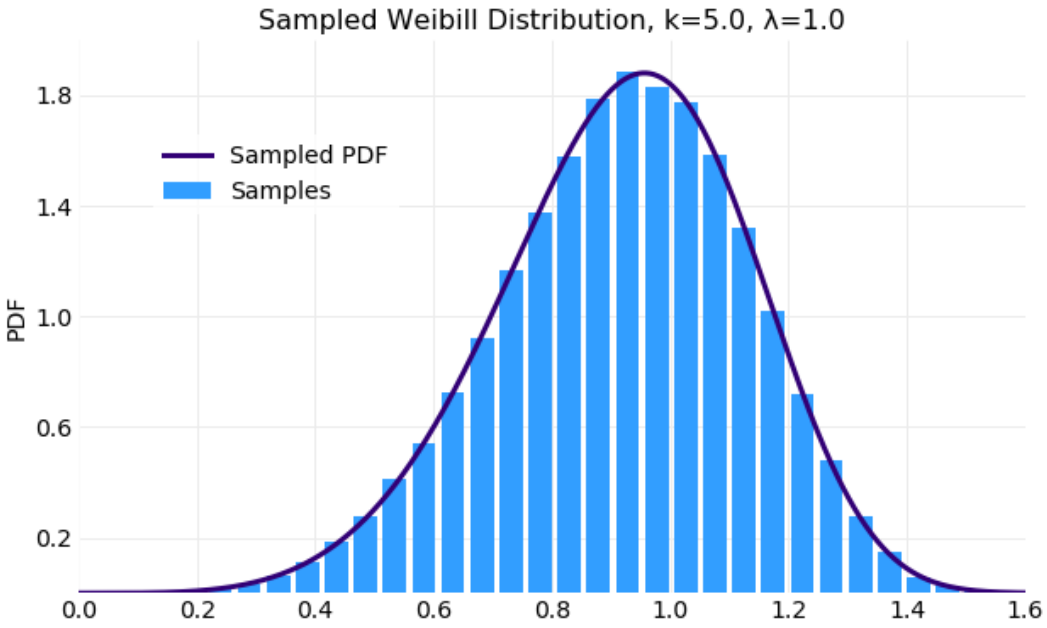
The sampler implementation for the continuous case is simpler than for the discrete case. Just as in the discrete case CDF samples with distribution $U \sim \textbf{Uniform}(0, 1)$ are generated. The desired samples with the Weibull distribution are then computed using the CDF inverse. Below an implementation of the sampler in Python is listed.

```
import numpy

k = 5.0
l = 1.0
nsamples = 100000

cdf_inv = lambda u: l * (numpy.log(1.0/(1.0 - u)))**(1.0/k)
samples = [cdf_inv(u) for u in numpy.random.rand(nsamples)]
```

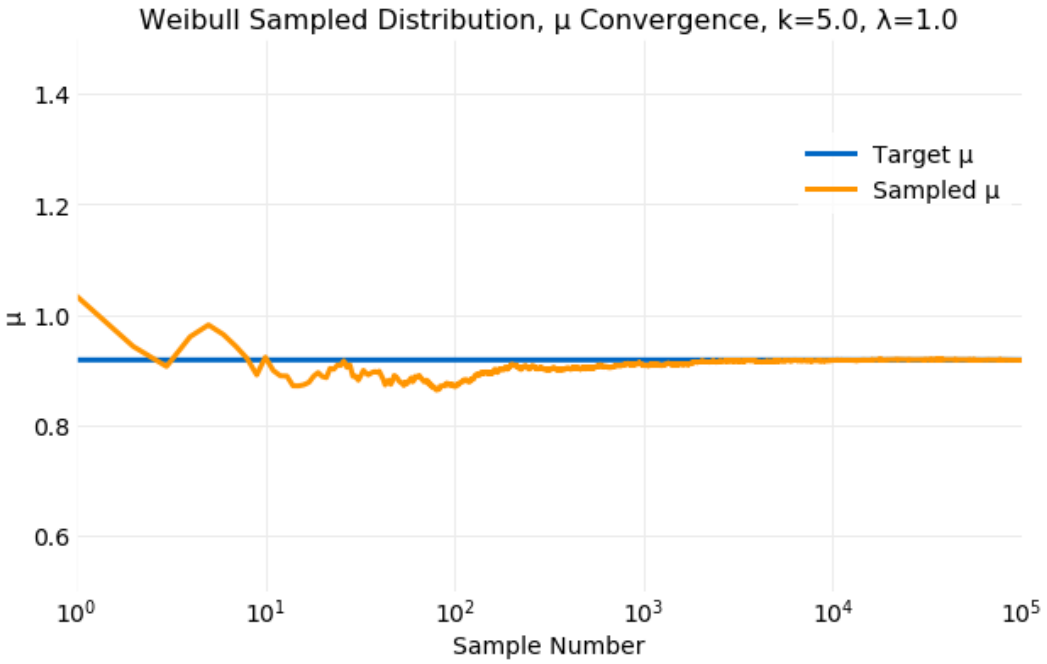
The following plot compares a histogram of the samples generated by the sampler above. The fit is quite good. The subtle asymmetry of the Weibull distribution is captured.

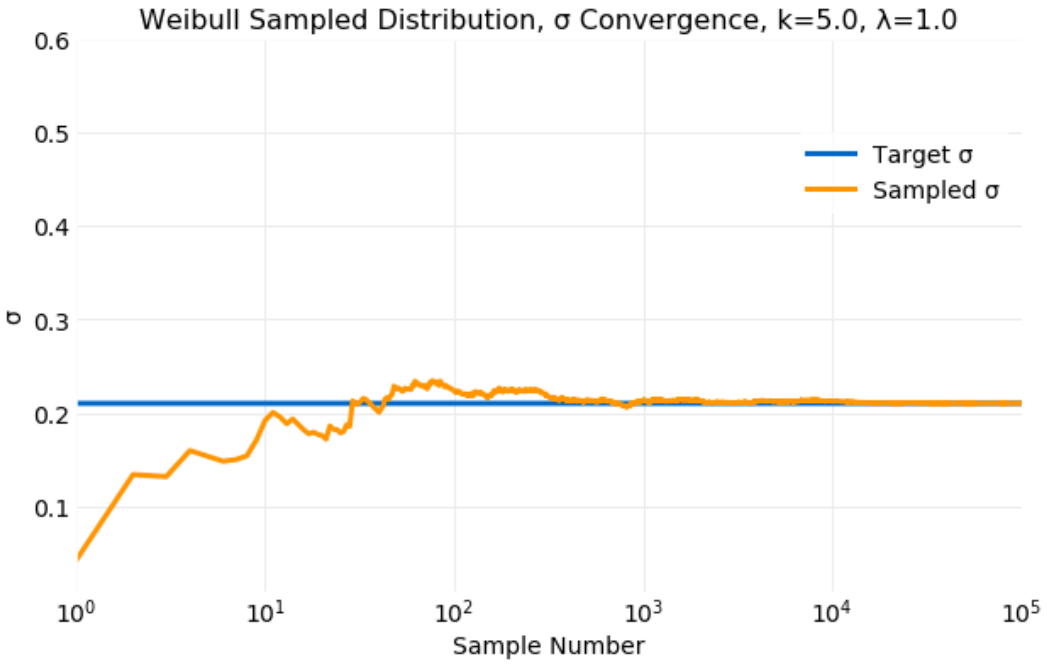


A measure of convergence of the samples to the target distribution can be obtained by comparing the cumulative moments of the distribution computed from the samples with the value computed analytically. For the Weibull distribution the first and second moments are given by,

$$\mu = \lambda\Gamma\left(1+\frac{1}{k}\right)$$
$$\sigma^2 = \lambda^2\left[\Gamma\left(1+\frac{2}{k}\right) - \left(\Gamma\left(1+\frac{1}{k}\right)\right)^2\right],$$

where $\Gamma(x)$ is the Gamma function. The following plots perform this comparison. The first shows the convergence of μ and the second the convergence of σ. Within only 1000 samples both μ and σ computed from samples is comparable to the analytic value.





3.3 Performance

Any continuous distribution, $f_X(x)$, can be approximated by the discrete distribution, $\{f_X(x_i)\Delta x_i\}_N$ for $i = 1, 2, 3, \ldots, N$, where $\Delta x_i = (x_{max} - x_{min})/(N-1)$ and $x_i =$

$x_{min} + (i-1)\Delta x_i$. This method has a couple of drawbacks compared to using Inverse CDF sampling on the continuous distribution. First, a bounded range for the samples must be assumed when in general the range of the samples can be unbounded. The Inverse CDF method can sample an unbounded range. Second, the performance for sampling a discrete distribution scales $O(N_{samples}N)$ while sampling the continuous distribution scales $O(N_{samples})$.