1: Inverse Transform

Inverse CDF for Continuous Distributions

If we recall, random variable X has continuous distribution if it has an associated density function $f(x)$ such that for any $A \subset \mathbb{R}$, we have:

$$P(X \in A) = \int_A f(x)dx$$

This means that the cdf $F(x) = \int_{-\infty}^{x} f(y)dy$, and by extension $F'(x) = f(x)$.

We can continue to use our same definition of an inverse cdf:

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}$$

but since the cdf will be a continuous function, this will specifically be equal to:

$$F^{-1}(u) = \inf\{x : F(x) = u\}$$

.

Inverse Transform

Proposition Let F be the cdf of a continuous random variable, and let F^{-1} be the inverse as described earlier. Let $U \sim Unif(0,1)$. Then $P(F^{-1}(U) \leq x) = F(x)$, that is, it has cdf $F(x)$.

Proof

$$\begin{aligned} P(F^{-1}(U) \leq x) &= P(F(F^{-1}(U) \leq F(x)) \\ &= P(U \leq F(x)) \\ &= F(x) \end{aligned}$$

Some basic examples

Example 1

Suppose

$$F(x) = \begin{cases} 0, & x \leq 0 \\ x^n, & 0 < x < 1 \\ 1, & x \geq 1 \end{cases}$$

We let $u = x^n$, which gives us the inverse cdf, $x = u^{1/n}$.

Example 2

Let $X \sim exp(\lambda)$, then $F(x) = 1 - exp(-\lambda x)$ for $x > 0$.

Letting $u = 1 - exp(-\lambda x)$, we find that $x = -\frac{\log(1-u)}{\lambda}$, which is equivalent in distribution to $-\frac{\log(U)}{\lambda}$.

Example 3

Suppose

$$f(x) = \begin{cases} x, & 0 < x \leq 1 \\ (2-x), & 1 < x < 2 \end{cases}$$

This means that the cdf will be:

$$F(x) = \begin{cases} 0, & x < 0 \\ \frac{x^2}{2}, & 0 < x \leq 1 \\ 1 - \frac{(2-x)^2}{2} & 1 < x < 2 \end{cases}$$

So we will have to treat this differently when $U \leq 0.5$ and when $U > 0.5$.

When $U < 0.5$, we set $u = \frac{x^2}{2}$, and find $X = \sqrt{2U}$. Conversely, when $U > 0.5$, we set $u = 1 - \frac{(2-x)^2}{2}$, and solving, we end up with: $x = 2 - \sqrt{2(1-u)}$.

This give us the following algorithm:

1. Let $U \sim Unif(0, 1)$
2. If $U \leq 0.5$, return $X = \sqrt{2U}$, else return $X = 2 - \sqrt{2(1-U)}$.

Sampling From Gamma Distributions

Note that if $X \sim Gamma(n, \lambda)$, then we have the cdf:

$$F(x) = \int_0^x \frac{\lambda \exp(-\lambda y)(\lambda y)^{n-1}}{\Gamma(n)} dy$$

Which we cannot solve in a closed form (except in trivial cases like $n = 1$), which will almost certainly will preclude us from inverting this directly.

However, in the case that n is an integer, we know that X can be represented as the sum of n independent $exp(\lambda)$ random variables.

Therefore, we can generate X as follows:

$$\begin{aligned} X &= -\frac{1}{\lambda} \sum_{i=1}^n \log(U_i) \\ &= -\frac{1}{\lambda} \log(\prod_{i=1}^n U_i) \end{aligned}$$

Let's compare the efficiency of this to numerically evaluating the cdf and numerically inverting:

```
import numpy as np
import time

from scipy.optimize import root_scalar
from scipy.stats import gamma, uniform

def inv_num(n, lmbda):
    u = uniform.rvs()
    # scipy uses mean parametrization
    f = lambda x: gamma.cdf(x, n , scale=1 / lmbda) - u
    f_prime = lambda x: gamma.pdf(x, n, scale=1 / lmbda)
    sol = root_scalar(f, x0=1, fprime=f_prime)
    return sol.root

def inv(n, lmbda):
    u = uniform.rvs(size=n)
    x = (-1 / lmbda) * np.log(np.product(u))
    return x


nsims = 100
n = 10
lmbda = 1

t1 = time.time()
X = [inv_num(n, lmbda) for i in range(0, nsims)]
t2 = time.time()
```

```
t3 = time.time()
X = [inv(n, lmbda) for i in range(0, nsims)]
t4 = time.time()

print(t2 - t1)
print(t4 - t3)
```