Monte-Carlo and Empirical Methods for Statistical Inference, FMS091/MAFM11

L2 — Random number generation

Generating pseudo-random numbers

- Generating pseudo-random $U(0,1)$ numbers
- Inversion and transformation methods
- Rejection sampling
- Conditional methods
- $N(0,1)$ numbers.

What are pseudo-random numbers?

- Numbers exhibiting statistical randomness while being generated by a deterministic process.
- Easier to generate than true random numbers.

Last lecture - Inference

Given observations $x_i$ from a density $f(x; \theta)$, we want to construct an estimate $\theta^*$ and “describe” the estimate.

A maximum likelihood estimate is:

- Asymptotically unbiased: $\mathbb{E}(\theta^*) \to \theta$ as $n \to \infty$.
- Asymptotically efficient: No estimator has a smaller mean squared error.
- Asymptotically normal: $\sqrt{n}(\theta^* - \theta) \in \mathcal{N}(0, \mathbb{I}(\theta)^{-1})$ as $n \to \infty$.

Relying on tools from classic inference we are essentially left with two options:

1. Use a simple enough models that we can explicitly work out the distribution of $\theta^*(x_1, \ldots, x_n)$.
2. Obtain enough observations that the asymptotic’s of the ML-estimators are valid.

Good random numbers

LCG Mersenne

What is a good pseudo-random number?

- From the correct distribution (especially the tails)
- Long periodicity
- “Independent”
- Fast to generate

Most standard computing languages have packages or functions that generate either $U(0,1)$ random numbers, or integers on $U(0, 2^{32} - 1)$.

- `rand` and `unifrnd` in MATLAB
- `rand` in C/C++
- `gsl_rng_uniform` in the GNU Scientific Library
- `Random` in Java
Linear congruential generator (LCG)

The linear congruential generator is a simple, fast, and popular way of generating random numbers,

\[ X_k = (a \cdot X_{k-1} + c) \mod m. \]

This will generate integer random numbers on \([0, m - 1]\) which are mapped to \(U(0, 1)\) through division by \(m\). The period of the generator is \(m\) if

1. \(c\) and \(m\) are relatively prime.
2. \(a - 1\) is divisible by all prime factors of \(m\).
3. \(a - 1\) is divisible by 4 if \(m\) is divisible by 4.

The LCG has problems with high serial correlation, and is sensitive to the choice of parameters.

Mersenne twister

- Developed in 1997 by Makoto Matsumoto and Takuji Nishimura.
- Called Mersenne twister since it uses Mersenne prime numbers, \(2^p - 1\).
- The most commonly used version is the MT19937 algorithm.
- Standard generator in MATLAB (from ver. 7.4), Maple, R, and GNU Scientific Library.
- Period of the generated numbers is \(2^{19937} - 1\).
- Equidistributed in high dimensions (LCG has problems with \(d > 5\)).
- See en.wikipedia.org/wiki/Mersenne_twister for details.

Inversion method

We will now assume that we have access to \(U(0, 1)\) pseudo-random numbers, \(u_1, u_2, \ldots\)

- We want random numbers \(X \in \mathbb{R}\) from a continuous distribution \(F\).
- Does a simple transform \(x = h(u)\) exist?.
- Take \(x = F^{-1}(u)\)
- For discrete random variables \(F^{-1}(u)\) does not exist. Use \(\inf\{x; F(x) \geq u\}\)
- Limited to cases where:
  1. The inverse of the distribution function is easy to evaluate.
  2. We want univariate random numbers.

Some standard cases & RANDU

Some standard generators:

<table>
<thead>
<tr>
<th>Generator</th>
<th>(m)</th>
<th>(a)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerical Recipes</td>
<td>(2^{32})</td>
<td>1664525</td>
<td>1013904223</td>
</tr>
<tr>
<td>glibc (used by GCC)</td>
<td>(2^{32})</td>
<td>1103515245</td>
<td>12345</td>
</tr>
<tr>
<td>MATLAB pre v. 5</td>
<td>(2^{32} - 1)</td>
<td>7^5 = 16807</td>
<td>0</td>
</tr>
</tbody>
</table>

RANDU is an infamous LCG proposed and implemented by IBM in the 1960's:

\[ X_k = \left(65539 \cdot X_{k-1}\right) \mod 2^{31} = \left(2^{16} + 3 \cdot X_{k-1}\right) \mod 2^{31}, \]

with \(X_0\) odd.
Transformation methods

The inversion method is a special case of a broader class of methods based on the transformation of random numbers.

Theorem: Let $X$ have a continuous density $f$ and let $h$ be a differentiable function with inverse $g = h^{-1}$. Then the random variable $Z = h(X)$ has density

$$f(g(z)) |g'(z)| .$$

Can be generalized to multivariate densities, and cases where the inverse of $h$ does not exist.

Examples: $\text{Exp} (\theta)$, $\text{N} (\mu, \sigma^2)$, $\chi^2(1)$

Rejection sampling

Theorem: Let $f$ be the density function of a random variable on $\mathbb{R}^d$ and let $Z \in \mathbb{R}^{d+1}$ be a random variable that is uniformly distributed on the set $A = \{z : 0 \leq z_{d+1} \leq Mf(z_1, \ldots, z_d)\}$ for an arbitrary constant $M > 0$. Then the vector $Z_1, \ldots, Z_d$ has density $f$.

To generate a uniform random number $Z$ on $A$, we could sample $Z_1, \ldots, Z_d$ from $f$ and then sample $Z_{d+1}$ as $U (0, Mf(z_1, \ldots, z_d))$.

But this requires that we can sample from $f$!

Rejection sampling (cont.)

Algorithm: Given a density $g(x)$ such that $Mf(x) \leq Kg(x)$ do

1. Draw $(z_1, \ldots, z_d)$ from $g(x)$.
2. Draw $z_{d+1}$ from $U (0, Kg(z_1, \ldots, z_d))$.
3. If $z_{d+1} > Mf(z_1, \ldots, z_d)$ goto 1 (reject).
4. $(z_1, \ldots, z_d)$ is now a sample from $f(x)$.

The efficiency of the algorithm is given by $M/K$.

Examples: Tail of a $\text{N} (0, 1)$, $(x_1, x_2)$ uniform on $x_1^2 + x_2^2 \leq 1$. 

Transformation methods (cont.)

Theorem: Let $X$ and $Y$ be independent random variables that take values in $\mathbb{R}$ with densities $f_x$ and $f_y$, then the density of $Z = X + Y$ is

$$f_z(z) = \int f_x(z-t)f_y(t) \, dt.$$ 

Examples: $\Gamma (k, 1)$, $\chi^2(n)$, $\text{Bin} (n, p)$. 

Rejection sampling (cont.)
Conditional methods

If we can decompose a multivariate density into conditional parts, the problem of sampling from a multivariate density can be reduced to sampling from several univariate densities.

\[ f(x_1, \ldots, x_n) = f(x_n | x_{n-1}, \ldots, x_1) f(x_{n-1} | x_{n-2}, \ldots, x_1) \cdots f(x_2 | x_1) f(x_1) \]

The problem is that it may be hard to find the conditional densities, e.g.

\[ f(x_1) = \int \cdots \int f(x_1, \ldots, x_n) \, dx_2 \cdots x_n \]

**N (0, 1) numbers**

There are several ways of generating N (0, 1) numbers.

- Use CLT, \( x = (\sum_{i=1}^{n} u_i - n/2) \sqrt{2/n} \)
- The Box-Müller transform

  \[
  x_1 = \sqrt{-2 \ln u_2 \cos(2\pi u_1)} \\
  x_2 = \sqrt{-2 \ln u_2 \sin(2\pi u_1)}
  \]

- The polar coordinate method. Draw \( u_1, u_2 \) uniform on the circle \( u_1^2 + u_2^2 \leq 1 \) using rejection sampling, and take

  \[
  x_1 = u_1 \sqrt{\frac{-2 \ln(u_1^2 + u_2^2)}{u_1^2 + u_2^2}} \\
  x_2 = u_2 \sqrt{\frac{-2 \ln(u_1^2 + u_2^2)}{u_1^2 + u_2^2}}
  \]

**N (0, 1) numbers (cont.)**

George Marsaglia’s ziggurat algorithm.

www.mathworks.com/company/newsletters/news_notes/clevescorner/spring01_cleve.html