COS 302 Precept 7

Spring 2020

Princeton University
Motivation for Today’s Precept

How to generate samples from a given distribution?

\[ p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \]
Outline

Pseudo-random number generation

Inverse transform sampling
Deterministic algorithms that produce a sequence of “relatively random” numbers:

\[ X_1: \text{initial input} \quad \text{seed} \]
\[ X_2; \]
\[ X_3 \]
\[ \vdots \]
Pseudo-random number generator

- Needs to design an algorithm that could output “relative random” numbers and the numbers should be made as random as possible
- The seed totally determines the output sequence of numbers. Hence the name “pseudo-random”.
Pseudo-random number generator

Linear Congruential Generator

- Defined by the following recurrence relation:

\[ X_{n+1} = (aX_n + c) \mod m \]

\( m \): modulus
\( a \): multiplier
\( c \): increment
Linear Congruential Generator example:

\[ a = 5, \quad c = 0, \quad m = 32 \]

\[ X_1 = 1 \]
\[ X_2 = (5 \times 1 + 0) \mod 32 = 5 \]
\[ X_3 = (5 \times 5 + 0) \mod 32 = 25 \]
\[ X_4 = 29 \]
\[ X_5 = 17 \]
\[ X_6 = 21 \]
\[ X_7 = 9 \]
\[ X_8 = 13 \]
\[ X_9 = 1 \]

Period = 8
Pseudo-random number generation

Choice of $a$, $c$ and $m$ for LCG are important:

---

The following table lists the parameters of LCGs in common use, including built-in `rand()` functions in runtime libraries of various compilers. This table is to show popularity, not examples to emulate; *many of these parameters are poor.* Tables of good parameters are available.\[8\][3]

<table>
<thead>
<tr>
<th>Source</th>
<th>modulus $m$</th>
<th>multiplier $a$</th>
<th>increment $c$</th>
<th>output bits of seed in <code>rand()</code> or <code>Random(L)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerical Recipes</td>
<td>$2^{32}$</td>
<td>1664525</td>
<td>1013904223</td>
<td></td>
</tr>
<tr>
<td>Borland C/C++</td>
<td>$2^{32}$</td>
<td>22695477</td>
<td>1</td>
<td>bits 30..16 in <code>rand()</code>, 30..0 in <code>lrand()</code></td>
</tr>
<tr>
<td>glibc (used by GCC)[15]</td>
<td>$2^{31}$</td>
<td>1103515245</td>
<td>12345</td>
<td>bits 30..0</td>
</tr>
<tr>
<td>ANSI C: Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++ [16] C90, C99, C11: Suggestion in the ISO/IEC 9899,[17] C18</td>
<td>$2^{31}$</td>
<td>1103515245</td>
<td>12345</td>
<td>bits 30..16</td>
</tr>
<tr>
<td>Borland Delphi, Virtual Pascal</td>
<td>$2^{32}$</td>
<td>134775813</td>
<td>1</td>
<td>bits 63..32 of ($seed \times L$)</td>
</tr>
<tr>
<td>Turbo Pascal</td>
<td>$2^{32}$</td>
<td>134775813 (0x8088405)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Microsoft Visual/Quick C/C++</td>
<td>$2^{32}$</td>
<td>214013 (343FD$_{16}$)</td>
<td>2531011 (269EC3$_{16}$)</td>
<td>bits 30..16</td>
</tr>
<tr>
<td>Microsoft Visual Basic (6 and earlier)[18]</td>
<td>$2^{24}$</td>
<td>1140671485 (43FD43FD$_{16}$)</td>
<td>12820163 (C39EC3$_{16}$)</td>
<td></td>
</tr>
</tbody>
</table>
Pseudo-random number generation allow us to sample from uniform distribution on $[0, 1]$.

\[ X_1, \ldots, X_n \sim [0, m] \text{ uniformly} \]
\[ \frac{X_1}{m}, \ldots, \frac{X_n}{m} \sim [0, 1] \text{ uniformly} \]
Outline

Pseudo-random number generation

Inverse transform sampling
Inverse transform sampling

Setup:

- We have samples from the uniform distribution on $[0, 1]$
- We want to sample from arbitrary distribution with given cumulative distribution function (CDF) $F_X$. 
Inverse transform sampling

1. Assume random variable $U$ follows uniform distribution in the interval $[0,1]$, i.e. Unif[0,1].

2. Find the inverse of the CDF for the desired distribution, e.g. $F_X^{-1}(y)$. 

3. Compute $X = F_X^{-1}(U)$. The computed random variable $X$ has the distribution $F_X(x)$. 
Inverse transform sampling

Basic idea:

\[ u \sim \text{unif}[0, 1] \]

\[ x = T(u) \quad \text{\(T\): monotonically increasing} \]

\[ F_x(x) = P(x \leq x) = P(T(u) \leq x) = P(u \leq T^{-1}(x)) \]

\[ = T^{-1}(x) \]

\[ T = F_x^{-1} \]
Inverse transform sampling

Example: Exponential distribution \((\lambda = 1)\)

\[
f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}
\]

\[
F_x(x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}
\]

\[
y = 1 - e^{-\lambda x}
\]

\[
e^{-\lambda x} = 1 - y
\]

\[
-\lambda x = \ln(1 - y)
\]

\[
\lambda = 1, \quad F_x^{-1}(y) = \frac{\ln(1 - y)}{-1}
\]
Inverse transform sampling

Example: Normal distribution

\[ f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

\[ F_x(x) = \frac{1}{2} \left[ 1 + \text{erf} \left( \frac{x-\mu}{\sigma \sqrt{2}} \right) \right] \]

\[ \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, dt \]

\[ y = \frac{1}{2} \left[ 1 + \text{erf} \left( \frac{x-\mu}{\sigma \sqrt{2}} \right) \right] \]

\[ 2y - 1 = \text{erf} \left( \frac{x-\mu}{\sigma \sqrt{2}} \right) \]

\[ \frac{x-\mu}{\sigma \sqrt{2}} = \text{erf}^{-1} (2y-1) \]

\[ x = \sqrt{2} \sigma \text{erf}^{-1} (2y-1) + \mu \]

\[ F_x^{-1}(y) = \sqrt{2} \sigma \text{erf}^{-1} (2y-1) + \mu \]
Inverse transform sampling

Example: Discrete distribution

\[ P(x = 1) = 0.3 \]
\[ P(x = 2) = 0.2 \]
\[ P(x = 3) = 0.5 \]

\[ F_x(x) = \begin{cases} 
0, & x < 1 \\
0.3, & 1 \leq x < 2 \\
0.5, & 2 \leq x < 3 \\
1, & x \geq 3 
\end{cases} \]

\[ F_x^{-1}(u) = \begin{cases} 
1, & 0 < u \leq 0.3 \\
2, & 0.3 < u \leq 0.5 \\
3, & 0.5 < u \leq 1 
\end{cases} \]

\[ F_x^{-1}(y) = \min \left\{ x \mid F_x(x) \geq y \right\} \quad u \sim (0, 1) \]