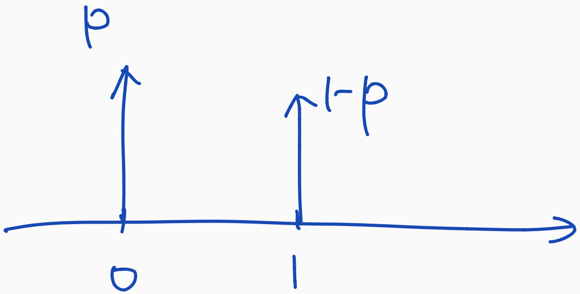
COS 302 Precept 7

Spring 2020

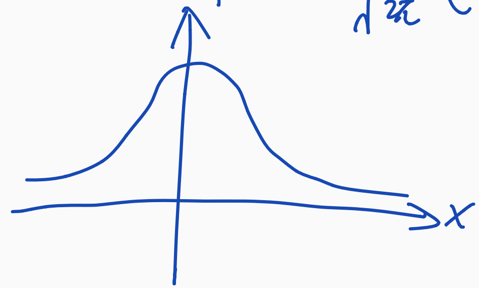
Princeton University

Motivation for Today's Precept

How to generate samples from a given distribution?



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



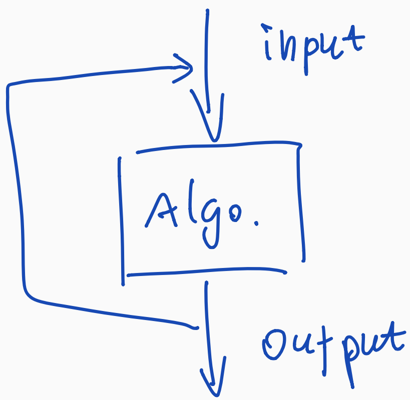
Outline

Pseudo-random number generation

Inverse transform sampling

Pseudo-random number generator

Deterministic algorithms that produce a sequence of "relatively random" numbers:



X_1: intitial input seed

X_2:

X_3

⋮

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 recurrance relation:

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

m : modulus

a : multiplier

c : increment

Pseudo-random number generation

Linear Congruential Generator example:

$a = 5, \quad c = 0, \quad m = 32$

$X_1 = 1$

$X_2 = (5 \times 1 + 0) \bmod 32 = 5$

$X_3 = (5 \times 5 + 0) \bmod 32 = 25$

$X_4 = 29$

$X_5 = 17$

$X_6 = 21$

$X_7 = 9$

$X_8 = 13$

$X_9 = 1$

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]

Source	modulus m	multiplier a	increment c	output bits of seed in *rand()* or *Random(L)*
Numerical Recipes	2^{32}	1664525	1013904223	
Borland C/C++	2^{32}	22695477	1	bits 30..16 in *rand()*, 30..0 in *lrand()*
glibc (used by GCC)[15]	2^{31}	1103515245	12345	bits 30..0
ANSI C: Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++ [16] C90, C99, C11: Suggestion in the ISO/IEC 9899,[17] C18	2^{31}	1103515245	12345	bits 30..16
Borland Delphi, Virtual Pascal	2^{32}	134775813	1	bits 63..32 of *(seed × L)*
Turbo Pascal	2^{32}	134775813 ($0x8088405_{16}$)	1	
Microsoft Visual/Quick C/C++	2^{32}	214013 ($343FD_{16}$)	2531011 ($269EC3_{16}$)	bits 30..16
Microsoft Visual Basic (6 and earlier)[18]	2^{24}	1140671485 ($43FD43FD_{16}$)	12820163 ($C39EC3_{16}$)	

Pseudo-random number generation

Pseudo-random number generation allow us to sample from uniform distribution on $[0, 1]$.

$x_1, \ldots, x_n \sim [0, m]$ uniformly

$\frac{x_1}{m}, \ldots, \frac{x_n}{m} \sim [0, 1]$ 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. $\text{Unif}[0, 1]$.

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:

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

$X = T(u)$ $\qquad\qquad$ T: monotonically increasing

$$F_X(x) = P(X \leq x) = P(T(u) \leq x) = P(u \leq T^{-1}(x))$$

$$= T^{-1}(x)$$

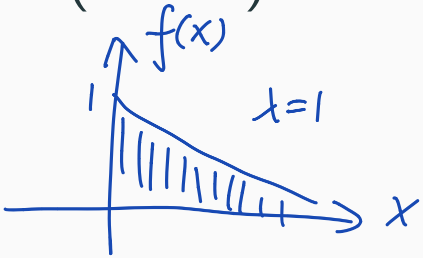
$T = F_X^{-1}$

Inverse transform sampling

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

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

λ



$$F_X(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & 0 \end{cases}$$

$$y = 1 - e^{-\lambda x}$$

$$e^{-\lambda x} = 1 - y$$

$$-\lambda x = \ln(1-y)$$

$$\lambda = 1, \quad F_X^{-1}(y) = \frac{\ln(1-y)}{-1}$$

Inverse transform sampling

Example: Normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

μ mean

σ^2 variance

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

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt$$

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

$$2y - 1 = \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)$$

$$\frac{x-\mu}{\sigma\sqrt{2}} = \operatorname{erf}^{-1}(2y-1)$$

$$x = \sqrt{2}\,\sigma\,\operatorname{erf}^{-1}(2y-1) + \mu$$

$$F_X^{-1}(y) = \sqrt{2}\,\sigma\,\operatorname{erf}^{-1}(2y-1) + \mu$$

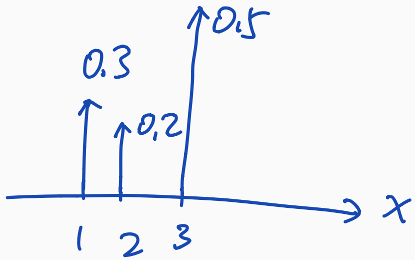
Inverse transform sampling

Example: Discrete distribution

$P(X=1) = 0.3$

$P(X=2) = 0.2$

$P(X=3) = 0.5$



PMF

$$F_X(x) = \begin{cases} 0, & x<1 \\ 0.3, & 1\le x<2 \\ 0.5, & 2\le x<3 \\ 1, & x\ge 3 \end{cases}$$

$$F_X^{-1}(y) = \min\{x \mid F_X(x) \ge y\}$$

$u \sim [0,1]$

$$F_X^{-1}(u) = \begin{cases} 1, & 0<u\le 0.3 \\ 2, & 0.3<u\le 0.5 \\ 3, & 0.5<u\le 1 \end{cases}$$

$F_X(1) = 0.3 \ge u$

$F_X(0.99) = 0 < u$

$F_X(x) \ge u$