Generation of Non-Uniform Random Numbers

Refs: Chapter 8 in Law and book by Devroye (watch for typos)

Peter J. Haas

CS 590M: Simulation
Spring Semester 2020

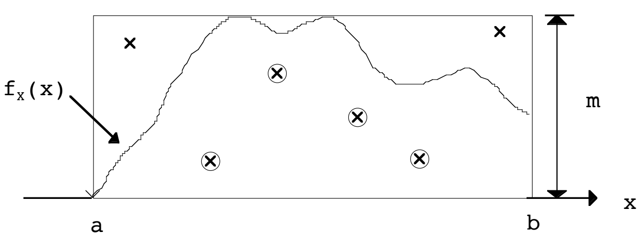
Acceptance-Rejection

Goal: Generate a random variate X having pdf f_X

- Avoids computation of $F^{-1}(u)$ as in inversion method
- Assumes f_X is easy to calculate

Special case: $f_X(x) > 0$ only on $[a, b]$ (finite support)

- Throw down points uniformly in enclosing rectangle R, reject points above f_X curve



- Return x-coordinate of accepted point

Acceptance-Rejection, Continued

Claim:
x-coordinate of an accepted point has pdf f_X

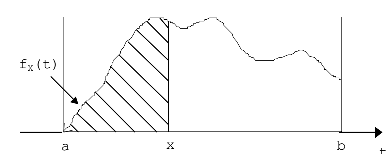
Proof

1. Let (Z_1, Z_2) be the (x, y)-coordinates of a random point distributed uniformly in R and fix $x \in [a, b]$
2. Then $P(Z_1 \le x, \text{acceptance}) = P\big(Z_1 \le x, Z_2 \le f_X(Z_1)\big)$
3. But $P\big(Z_1 \le x, Z_2 \le f_X(Z_1)\big) =$ prob that (Z_1, Z_2) falls in shaded region:

$P(Z_1 \le x, \text{acceptance}) =$

$P(\text{acceptance}) =$

$P(Z_1 \le x \mid \text{acceptance}) =$



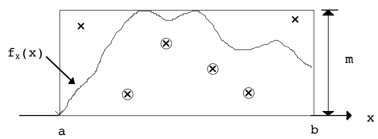
Acceptance-Rejection, Continued

Acceptance-Rejection Algorithm (Finite Support)

1. Generate $U_1, U_2 \overset{D}{\sim} \text{Uniform}(0,1)$ (U_1 and U_2 are independent)
2. Set $Z_1 = a + (b-a)U_1$ and $Z_2 = mU_2$ (inversion method)
3. if $Z_2 \le f_X(Z_1)$, return $X = Z_1$, else go to step 1

How many (U_1, U_2) pairs must we generate?

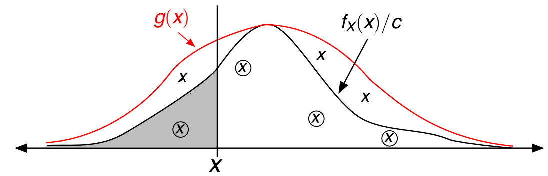
- N (= number pairs generated) has geometric dist'n: $P(N=k) = p(1-p)^{k-1}$ where $p = 1/(\text{area of } R)$
- So $E[N] = 1/p = (\text{area of } R) = (b-a)m$
- So make m as small as possible



Generalized Acceptance-Rejection: Infinite Support

Find density g that majorizes f_X

- There exists a constant c such that $f_X(x)/c \le g(x)$ for all x
- Smallest such constant is $c = \sup_x \big(f_X(x)/g(x)\big)$



Generalized Acceptance-Rejection Algorithm

1. Generate $Z \overset{D}{\sim} g$ and $U \overset{D}{\sim} \text{Uniform}(0,1)$ (Z, U independent)
2. if $U\,g(Z) \le f_X(Z)/c$, return $X = Z$, else go to step 1

Expected number of (Z, U) pairs generated: $E[N] = c$

Convolution Method

Goal: Generate X where $X = Y_1 + \cdots + Y_m$ $[Y_1, \ldots, Y_m \text{ i.i.d.}]$

- $f_X = f_{Y_1} * f_{Y_2} * \cdots * f_{Y_m}$
- Where the convolution $f * g$ is defined by $(f*g)(x) = \int_{-\infty}^{\infty} f(x-y)g(y)\,dy$

Convolution Algorithm

- Generate $Y_1, \ldots, Y_m$
- Return $X = Y_1 + \cdots + Y_m$

Example: Binomial Distribution

- Suppose $X \overset{D}{\sim} \text{Binom}(m,p)$
- Then $X = Y_1 + \cdots + Y_m$ where $Y_i \overset{D}{\sim} \text{Bernoulli}(p)$
- Often part of a more complex algorithm (e.g., do something else if m large)

Composition Method

Suppose that we can write

- $F_X(x) = p_1 F_{Y_1}(x) + p_2 F_{Y_2}(x) + \cdots + p_m F_{Y_m}(x)$ or
- $f_X(x) = p_1 f_{Y_1}(x) + p_2 f_{Y_2}(x) + \cdots + p_m f_{Y_m}(x)$

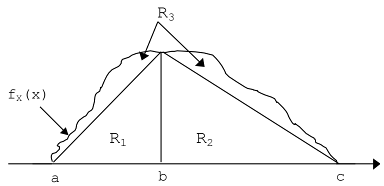
where p_i's are nonnegative and $\sum_{i=1}^m p_i = 1$

Composition Method

1. Generate a discrete RV J where $P(J=j) = p_j$ for $1 \le j \le m$
2. Generate Y_J from F_{Y_J} or f_{Y_J}
3. Return $X = Y_J$

Can lead to very fast generation algorithms

Composition Method: Example



- Let p_i = area of R_i for $1 \le i \le 3$
- Then $f_X = p_1 f_{Y_1} + p_2 f_{Y_2} + p_3 f_{Y_3}$, where

$$f_{Y_1}(x) = \begin{cases} \frac{2(x-a)}{(b-a)^2} & \text{if } a \le x \le b; \\ 0 & \text{otherwise} \end{cases}$$

$$f_{Y_2}(x) = \begin{cases} \frac{2(c-x)}{(c-b)^2} & \text{if } b \le x \le c; \\ 0 & \text{otherwise} \end{cases}$$

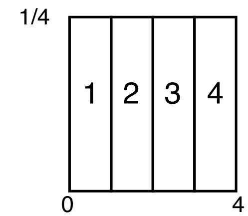
$$f_{Y_3}(x) = (1/p_3)\big(f_X(x) - p_1 f_{Y_1}(x) - p_2 f_{Y_2}(x)\big)$$

- Easy to generate Y_1 and Y_2 (the "usual case"):
  - $Y_1 = \max(U_1, U_2)$ and $Y_2 = \min(U_1, U_2)$ or use inversion

Alias Method for Discrete Random Variables

Goal: Generate X **with** $P(X = x_i) = p_i$ **for** $1 \le i \le n$

Easy case: $p_1 = p_2 = \cdots = p_n$

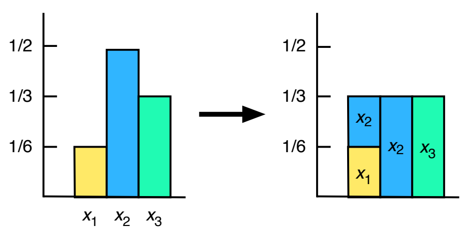


Algorithm

1. Generate $U \overset{D}{\sim} \text{Uniform}(0,1)$
2. Return x_J, where $J = \lceil nU \rceil$

$\lceil x \rceil$ = smallest integer $\ge x$ (ceiling function)

Alias Method: General Case

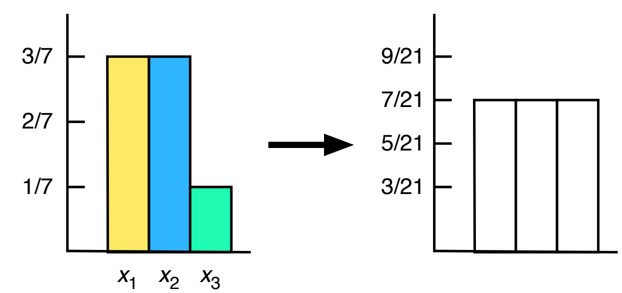


i	x_i	p_i	a_i	r_i
1	x_1	1/6	x_2	0.5
2	x_2	1/2	x_2	1.0
3	x_3	1/3	x_3	1.0

Alias Algorithm

1. Generate $U_1, U_2 \overset{D}{\sim} \text{Uniform}(0,1)$
2. Set $I = \lceil nU_1 \rceil$
3. If $U_2 \le r_I$ return $X = x_I$ else return $X = a_I$

Alias Method: Another Example



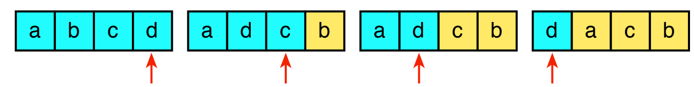
i	x_i	p_i	a_i	r_i
1	x_1	3/7		
2	x_2	3/7		
3	x_3	1/7		

Random Permutations: Fisher-Yates Shuffle

Goal: Create random permutation of array x of length n

Fisher-Yates Algorithm

1. Set $i \leftarrow n$
2. Generate random integer N between 1 and i (e.g., as $\lceil iU \rceil$)
3. Swap $x[N]$ and $x[i]$
4. Set $i \leftarrow i - 1$
5. If $i = 1$ then exit



Q: What about this algorithm:
For $i = 1$ to n: swap $x[i]$ with random entry

Other random objects: graphs, matrices, random vectors, ...

Sequential Bernoulli Sampling

- Stream of items $x_1, x_2, \dots,$
- Insert each item into sample with probability p
- Expected sample size after nth item $= np$
- Fast implementation: generate skips directly (geometrically distributed)

Bernoulli Sampling:

Generate $U \overset{D}{\sim} \text{Uniform}(0,1)$
Set $\Delta \leftarrow 1 + \lfloor \ln U / \ln(1-p) \rfloor$ [Geometric on $\{1, 2, \dots\}$, HW #4]
Set $m \leftarrow \Delta$
Upon arrival of x_i:

- if $i = m$, then
  - Include x_i in sample
  - Generate $U \overset{D}{\sim} \text{Uniform}(0,1)$
  - Set $\Delta \leftarrow 1 + \lfloor \ln U / \ln(1-p) \rfloor$
  - set $m \leftarrow m + \Delta$

Reservoir Sampling

- Stream of items $x_1, x_2, \dots,$
- Maintain a uniform random sample of size N w.o.replacement

Reservoir Sampling:

Upon arrival of x_i:

- if $i \le N$, then include x_i in sample
- if $i > N$, then
  - Generate $U \overset{D}{\sim} \text{Uniform}(0,1)$
  - If $U \le N/i$, then include x_i in sample, replacing randomly chosen victim

- Can generate skips directly using acceptance-rejection [JS Vitter, ACM Trans. Math. Softw., 11(1): 37–57, 1985]

Reservoir Sampling: Simple Example

- Sample size = 1
- S_i = sample state after processing jth item (called i_j)
- accept item i_1 into S_1 with probability 1

$$P(i_1 \in S_1) = 1$$

- accept item i_2 into S_2 with probability $1/2$

$$P(i_1 \in S_2) = P(i_1 \in S_1) \times P(i_2 \notin S_2) = (1)(1/2) = 1/2$$
$$P(i_2 \in S_2) = 1/2$$

- accept item i_3 into S_3 with probability $1/3$

$$P(i_1 \in S_3) = P(i_1 \in S_2) \times P(i_3 \notin S_3) = (1/2)(2/3) = 1/3$$
$$P(i_2 \in S_3) = P(i_2 \in S_2) \times P(i_3 \notin S_3) = (1/2)(2/3) = 1/3$$
$$P(i_3 \in S_2) = 1/3$$

Non-Homogeneous Poisson Processes: Thinning

Ordinary (Homogeneous) Poisson process

- Times between successive events are i.i.d. Exp(λ)
- Event probabilities for disjoint time intervals are independent (Markov property)
- $P(\text{event in } t + \Delta t) \approx \lambda \Delta t$ for Δt very small
- $P(T_n > y + z \mid T_{n-1} = y) = e^{-\lambda z}$

$P(n \text{ events in } [t, t+\Delta t]) = (\lambda \Delta t)^n e^{-\lambda \Delta t}/n!$

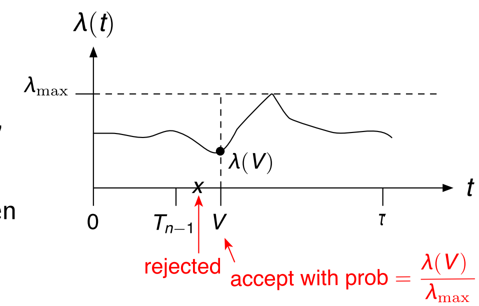
Non-Homogeneous Poisson process

- Event probabilities for disjoint time intervals are independent
- $P(\text{event in } t + \Delta t) \approx \lambda(t) \Delta t$ for Δ very small
  [$\lambda(t)$ is sometimes called a hazard function]
- $P(T_n > y + z \mid T_{n-1} = y) = e^{-\int_y^{y+z} \lambda(u)\, du}$
- Can capture, e.g., time-of-day effects

Thinning, Continued

Suppose that $\lambda(t) \le \lambda_{\max}$ **for** $t \in [0, \tau]$

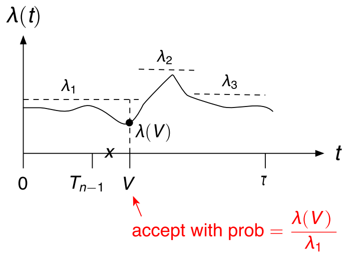
- Idea: Generate "too many" events according to a Poisson($\lambda_{\max}$) process, then reject some of the events



Thinning Algorithm:

1. Set $T_0 = 0$, $V = 0$, and $n = 0$
2. Set $n \leftarrow n + 1$ [Generate T_n]
3. Generate $E \overset{D}{\sim} \text{Exp}(\lambda_{\max})$ and $U \overset{D}{\sim} \text{Uniform}(0, 1)$
4. Set $V \leftarrow V + E$ [Proposed event time]
5. If $U \le \lambda(V)/\lambda_{\max}$ then set $T_n = V$ else go to Step 3
6. If $T_n < \tau$ then go to Step 2 else exit

Thinning, Continued



Ross's Improvement

- Piecewise-constant upper-bounding to reduce rejections
- Correction for event times that span segments

Many other approaches

- Inversion (see HW #4)
- Idiosyncratic methods:
  Exploit special properties of Poisson process