Lecture 26/27: Random Numbers

BT 3051 – Data Structures and Algorithms for Biology

Karthik Raman
Department of Biotechnology
Bhupat and Jyoti Mehta School of Biosciences
Indian Institute of Technology Madras

Introduction

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"

— John von Neumann (1951)

Random Numbers

Numbers that are “chosen at random” are useful in many different kinds of applications

- Simulation
- Sampling
- Numerical analysis
  - Optimisation
  - Numerical integration
  - ...
- Computer programming
  - Test data
- Decision making
- Recreation
- ...

Pseudorandom Numbers

"The digital computer cannot generate random numbers, and it is generally not convenient to connect the computer to some external source of random events. For most applications in statistics, engineering, and the natural sciences, this is not a disadvantage if there is some source of pseudorandom numbers, samples of which seem to be randomly drawn from some known distribution. There are many methods that have been suggested for generating such pseudorandom numbers."

— James E. Gentle, in "Random number generation and Monte Carlo Methods", ISBN 0387001786

What is a *random* number I

Donald Knuth, Seminumerical Algorithms (TAOCP Vol. 2, Ch. 3)

People who think about this topic almost invariably get into philosophical discussions about what the word "random" means. In a sense, there is no such thing as a random number; for example, is 2 a random number? Rather, we speak of a sequence of independent random numbers with a specified distribution, *and this means loosely that each number was obtained merely by chance, having nothing to do with other numbers of the sequence, and that each number has a specified probability of falling in any given range of values.*

What is a *random* number II

Donald Knuth, Seminumerical Algorithms (TAOCP Vol. 2, Ch. 3)

A uniform distribution on a finite set of numbers is one in which each possible number is equally probable. A distribution is generally understood to be uniform unless some other distribution is specifically mentioned. Each of the ten digits 0 through 9 will occur about $\frac{1}{10}$ of the time in a (uniform) sequence of random digits. Each pair of two successive digits should occur about $\frac{1}{100}$ of the time, etc. Yet if we take a truly random sequence of a million digits, it will not always have exactly 100,000 zeros, 100,000 ones, etc. In fact, chances of this are quite slim; a sequence of such sequences will have this character on the average.

What is a *random* number III

Donald Knuth, Seminumerical Algorithms (TAOCP Vol. 2, Ch. 3)

Any specified sequence of a million digits is equally as probable as the sequence consisting of a million zeros. Thus, if we are choosing a million digits at random and if the first 999,999 of them happen to come out to be zero, the chance that the final digit is zero is still exactly $\frac{1}{10}$*, in a truly random situation. These statements seem paradoxical to many people, but there is really no contradiction involved.*

von Neumann's Middle-Square Method

- John von Neumann first suggested an arithmetic approach in about 1946, using the "middle-square" method
- His idea: take the square of the previous random number and to extract the middle digits
  - e.g. if we are generating 10-digit numbers and the previous value was 5772156649, we square it to get 3331792380594909201, and the next number is therefore 7923805949
- Many issues — one can't be too careful with random number generators!

Knuth's Algorithm K

Algorithm K ("Super-random" number generator). Given a 10-digit decimal number X, this algorithm may be used to change X to the number that should come next in a supposedly random sequence.

- K1. [Choose number of iterations.] Set $Y \leftarrow \lfloor X/10^9 \rfloor$, the most significant digit of X. (We will execute steps K2 through K13 exactly $Y + 1$ times; that is, *we will apply randomizing transformations a random number of times*.)
- K2. [Choose random step.] Set $Z \leftarrow \lfloor X/10^8 \rfloor \bmod 10$, the second most significant digit of X. Go to step K(3 + Z). (That is, we now jump to a random step in the program.)

Knuth's Algorithm K

- K3. [Ensure $\geq 5 \times 10^9$.] If $X < 5000000000$, set $X \leftarrow X + 5000000000$.
- K4. [Middle square.] Replace X by $\lfloor X^2/10^5 \rfloor \bmod 10^{10}$, that is, by the middle of the square of X.
- K5. [Multiply.] Replace X by $(1001001001\ X) \bmod 10^{10}$.
- K6. [Pseudo-complement.] If $X < 100000000$, then set $X \leftarrow X + 9814055677$; otherwise set $X \leftarrow 10^{10} X$.
- K7. [Interchange halves.] Interchange the low-order five digits of X with the high-order five digits; that is, set $X \leftarrow 10^5(X \bmod 10^5) + \lfloor X/10^5 \rfloor$, the middle 10 digits of $(10^{10} + 1)X$.

Knuth's Algorithm K

- K8. [Multiply.] Same as step K5.
- K9. [Decrease digits.] Decrease each non-zero digit of the decimal representation of X by one.
- K10. [99999 modify.] If $X < 10^5$, set $X \leftarrow X^2 + 99999$; otherwise set $X \leftarrow X - 99999$.
- K11. [Normalize.] (At this point X cannot be zero.) If $X < 10^9$, set $X \leftarrow 10X$ and repeat this step.

Knuth's Algorithm K

- K12. [Modified middle square.] Replace X by $\lfloor X(X1)/10^5 \rfloor \bmod 10^{10}$, that is, by the middle 10 digits of X(X − 1).
- K13. [Repeat?] If Y > 0, decrease Y by 1 and return to step K2. If Y = 0, the algorithm terminates with X as the desired "random" value.

Ridiculously, the algorithm almost immediately converged to the 10-digit value 6065038420, which — by extraordinary coincidence — is transformed into itself by the algorithm. With another starting number, the sequence began to repeat after 7401 values, in a cyclic period of length 3178.

Moral of the Story

"Random numbers should not be generated with a method chosen at random. Some theory should be used."

Linear Congruential Generator

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

We choose four "magic numbers":

- m, the modulus; $m > 0$
- a, the multiplier; $0 \leq a < m$
- c, the increment; $0 \leq c < m$
- X_0, the starting value; $0 \leq X_0 < m$.

How to choose these magic numbers?

Linear Congruential Generator

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

Try:

- $m = 10, a = 7, c = 7, X_0 = 7$
- $m = 4096, a = 109, c = 853, X_0 = 0$

Key Principle

- We perform an experiment many times (n) and count the number of occurrences of an event $\mathcal{A}$
- The *relative frequency* of occurrence of event $\mathcal{A}$ is $\frac{n_a}{n}$
- The *frequency theory* of probability asserts that the relative frequency converges as $n \to \infty$

$$\Pr(\mathcal{A}) = \lim_{n \to \infty} \frac{n_a}{n}$$

Random Variates

- A Random Variate is an algorithmically generated realisation of a random variable
- `x = random.random()` generates a *Uniform[0,1)* random variate
- How do we generate:
  - A *Uniform(a,b)* variate?
  - An equally likely integer in $[a, b]$?

Equally Likely Random Variates

- Uniform(0,1) random variates can be used to generate an *EquallyLikely[a,b]* random variate
- `u = random.random()` $\Leftrightarrow 0 \leq u < 1$

  $\Leftrightarrow 0 \leq (b-a)u < b-a$

  $\Leftrightarrow 0 \leq \lceil (b-a)u \rceil \leq b-a$

  $\Leftrightarrow a \leq a + \lceil (b-a)u \rceil \leq b$
- What happens if you have a `rand()` that generates uniform random variates in $(0, 1)$?
- Moral: Safely use `random.randint(a,b)`!
- Alert: Always try to use a built-in random generator!

How to generate the following variates?

- A coin toss?
- A die roll?
- A roll of two dice?
- A random[a] DNA sequence of length L?
- A random point *on a circle*?
- A random point *within a circle*?
- A random point within a square?

[a] *unless otherwised mentioned*, we mean **uniform**