Monte Carlo Methods

Spring Semester 2013/14, Department of Applied Mathematics, University of Crete
Instructor: Harmandaris Vagelis, email: vagelis@tem.uoc.gr

Part II: Monte Carlo Integration, Random Numbers generators
Introductory Examples: Calculate $\pi$

Calculation of number $\pi$ with the following method:

➢ Περικλείουμε κύκλο με ένα τετράγωνο. Δημιουργούμε $m$ τυχαία σημεία μέσα στο τετράγωνο.

➢ Βρίσκουμε τα σημεία που εμπεριέχονται και μέσα στον κύκλο, $n$.

➢ Αν $r = n/m$, τότε ο αριθμός $\pi$ προσεγγίζεται ως $\pi \approx 4r$. Όσο περισσότερα τα σημεία $m$ τόσο μεγαλύτερη ακρίβεια του υπολογισμού.
Introductory Examples: Calculate $\pi$

Algorithm:

```plaintext
npoints = 1000000
circle_count = 0
do j = 1, npoints
    generate 2 random numbers between 0 and 1
    xcoordinate = random1
    ycoordinate = random2
    if (xcoordinate, ycoordinate) inside circle then
        circle_count = circle_count + 1
    end if
end do
PI = 4.0*circle_count/npoints
```

- Ο χρόνος υπολογισμού είναι κυρίως ο χρόνος εκτέλεσης της επαναληπτικής διαδικασίας (loop).
- Αυτό οδηγεί σε (σχεδόν) ‘τέλειο παραλληλισμό’ (embarrassingly parallelism):
  ➢ Εντατικοί υπολογισμοί.
  ➢ Ελάχιστη επικοινωνία, ελάχιστο I/O.
Introductory Examples: Calculate π

Estimate π as a function of sample size:
Monte Carlo Integration

- Two major classes of numerical problems that arise in statistical inference
  - *optimization* problems
  - *integration* problems

- Although optimization is generally associated with the likelihood approach, and integration with the Bayesian approach, these are not strict classifications

- Generic problem of evaluating the integral

\[
E_f[h(X)] = \int_{\mathcal{X}} h(x) f(x) \, dx.
\]

- Based on previous developments, it is natural to propose using a sample \((X_1, \ldots, X_m)\) generated from the density \(f\)

- Approximate the integral by the empirical average

- This approach is often referred to as the *Monte Carlo method*
Monte Carlo Integration

**Strong Law**

- For a sample \((X_1, \ldots, X_m)\), the empirical average
  \[
  \bar{h}_m = \frac{1}{m} \sum_{j=1}^{m} h(x_j)
  \]
  converges almost surely to
  \[
  \mathbb{E}_f[h(X)]
  \]

- This is the Strong Law of Large Numbers
Central Limit Theorem

- Estimate the variance with
  \[ \text{var}(\overline{h}_m) = \frac{1}{m} \int_X (h(x) - \mathbb{E}_f[h(X)])^2 f(x) \, dx \]

- For \( m \) large,
  \[ \frac{\overline{h}_m - \mathbb{E}_f[h(X)]}{\sqrt{\text{var}_m}} \]
  is therefore approximately distributed as a \( \mathcal{N}(0, 1) \) variable

- This leads to the construction of a convergence test and of confidence bounds on the approximation of \( \mathbb{E}_f[h(X)] \).
Monte Carlo Integration: Example

Example: Calculate the integral of a function $h(x)$

$$h(x) = [\cos(50x) + \sin(20x)]^2$$

- To calculate the integral, we generate $U_1, U_2, \ldots, U_n$ iid $\mathcal{U}(0, 1)$ random variables, and approximate $\int h(x)dx$ with $\sum h(U_i)/n$.
- It is clear that the Monte Carlo average is converging, with value of 0.963 after 10,000 iterations.
Monte Carlo Integration: Example

Example: Estimators
Monte Carlo Integration

- Generalization of Integration: **Riemann sums vs MC method** (see hand notes).

\[
\int_0^1 f(x) \, dx = \int_0^1 \int_0^1 1 \, dt \, dx
\]

\[
= \int_0^1 \int \int 1 \, dt \, dx \quad \{ (x, t): t \leq f(x) \}
\]

\[
= \frac{\int_0^1 \int_1 1 \, dt \, dx \quad \{ (x, t): t \leq f(x) \}}{\int_0^1 \int_0^1 1 \, dt \, dx \quad \{ 0 \leq x, t \leq 1 \}}
\]

- Mapping: \( f: [0, 1] \rightarrow [0, 1] \)
Monte Carlo Integration

- Speed of convergence of Monte Carlo integration is $O_P(n^{-1/2})$.

- Speed of convergence of numerical integration of a one-dimensional function by Riemann sums is $O(n^{-1})$.

- Does not compare favourably for one-dimensional problems.

However:

- Order of convergence of Monte Carlo integration is independent of the dimension.

- Order of convergence of numerical integration techniques like Riemann sums deteriorates with the dimension increasing.

→ Monte Carlo methods can be a good choice for high-dimensional integrals.
Random Number Generators

- Philosophical paradox:
  - We need to reproduce randomness by a computer algorithm.
  - A computer algorithm is deterministic in nature.
  - "pseudo-random numbers"

- Pseudo-random number from \(U[0, 1]\) will be our only "source of randomness".

- Other distributions can be derived from \(U[0, 1]\) pseudo-random numbers using deterministic algorithms.
Pseudo-Random Number Generators

A pseudo-random number generator (RNG) should produce output for which the $U[0, 1]$ distribution is a suitable model.

The pseudo-random numbers $X_1, X_2, \ldots$ should thus have the same \textit{relevant} statistical properties as independent realisations of a $U[0, 1]$ random variable.

They should reproduce independence ("lack of predictability"): $X_1, \ldots, X_n$ should not contain any discernible information on the next value $X_{n+1}$. This property is often referred to as the lack of predictability.

The numbers generated should be spread out evenly across $[0, 1]$. 
A simple example: Congruential pseudo-RNG.

Algorithm 1.1: Congruential pseudo-random number generator

1. Choose \( a, M \in \mathbb{N} \), \( c \in \mathbb{N}_0 \), and the initial value ("seed") \( Z_0 \in \{1, \ldots, M - 1\} \).
2. For \( i = 1, 2, \ldots \)
   
   Set \( Z_i = (aZ_{i-1} + c) \mod M \), and \( X_i = Z_i / M \).

\( Z_i \in \{0, 1, \ldots, M - 1\} \), thus \( X_i \in [0, 1) \).
Consider the choice of $a = 81$, $c = 35$, $M = 256$, and seed $Z_0 = 4$.

\[
\begin{align*}
Z_1 &= (81 \cdot 4 + 35) \mod 256 = 359 \mod 256 = 103 \\
Z_2 &= (81 \cdot 103 + 35) \mod 256 = 8378 \mod 256 = 186 \\
Z_3 &= (81 \cdot 186 + 35) \mod 256 = 15101 \mod 256 = 253 \\
\ldots
\end{align*}
\]

The corresponding $X_i$ are $X_1 = 103/256 = 0.4023438$, $X_2 = 186/256 = 0.72656250$, $X_1 = 253/256 = 0.98828120$. 

MC Methods, Ch. 2: Integration, Optimization, Random Number Generators
Pseudo-Random Number Generators

- **RANDU**: A typical poor choice of RNG.

- Very popular in the 1970s (e.g. System/360, PDP-11).

- Linear congruential generator with $a = 2^{16} + 3$, $c = 0$, and $M = 2^{31}$.

- The numbers generated by RANDU lie on only 15 hyperplanes in the 3-dimensional unit cube!

According to a salesperson at the time: “We guarantee that each number is random individually, but we don’t guarantee that more than one of them is random.”
Pseudo-Random Number Generators

Flaw of the linear congruential RNG.

- "Crystalline" nature is a problem for every linear congruential generator.
- Sequence of generated values $X_1, X_2, \ldots$ viewed as points in an $n$-dimension cube lies on a finite, and often very small number of parallel hyperplanes.
- Marsaglia (1968): "the points [generated by a congruential generator] are about as randomly spaced in the unit $n$-cube as the atoms in a perfect crystal at absolute zero."
- The number of hyperplanes depends on the choice of $a$, $c$, and $M$.
- For these reasons do not use the linear congruential generator! Use more powerful generators (like e.g. the Mersenne twister, available in GNU R).
Another problematic example:

Linear congruential generator with \( a = 1229, c = 1, \) and \( M = 2^{11}. \)

Pairs of generated values \((X_{2k-1}, X_{2k})\) Transformed by Box-Muller method
Bibliography

