Data Analysis & Modeling Techniques

# Randomness, Simulations and Monte Carlo Methods

# Randomness

- Randomization in sample-based problem solutions can increase the performance in very complex problems
    - Random sampling has statistical properties that deterministic samples can not guarantee
- To use randomization it is necessary to generate random numbers that have particular properties
    - **Pseudo-Random numbers** are aimed at generating random sequences where the sequential samples appear as if they are independently generated from a particular random distribution
    - **Quasi-Random numbers** are aimed at generating samples that overall follow a particular distribution but might be correlated as a sequence

# Randomness and Monte Carlo Methods

- Introducing randomness in an algorithm can lead to improved efficiencies
  - Random sampling can provide probabilistically good results with relatively few samples
- Many random algorithms use stochastic simulation as part of their computation – Monte Carlo Methods
  - Exploit randomness to obtain statistical sample of outcomes
- Monte Carlo methods are particularly useful to study
  - Nondeterministic systems
  - Deterministic systems that are too complicated to model
  - Deterministic problems too high dimensional for discretization

# Randomness

- Randomness is often defined in terms of
  - **Incompressibility**
    - The random sequence is the shortest description of itself
  - **Unpredictability**
    - The next random number is not predictable from the previous ones
  - **Not repeatable**
    - Random sequences do not repeat (might not always be desirable)

# Random Number Generators

- To be used, the computer needs access to random numbers
  - **True random number generators**
    - To generate true random numbers, physical processes can be used
      - Radioactive decay
    - Tables with true random sequences can be (and have been) used
  - **Pseudo-random number generators**
    - Random numbers are generated using a deterministic algorithm
    - Sequence of numbers appears random without knowledge of the algorithm
      - Pseudo-random numbers are predictable if the algorithm is known
      - Pseudo-random numbers are repeatable and reproducible
      - Pseudo-random number sequences will eventually repeat
  - **Quasi-random number generators**
    - Quasi-random numbers sacrifice randomness of points and focuses on the uniformity of the sample sequence

# Simulation of Random Variables

**Monte Carlo Methods:**

Represent any complex distribution in terms of simpler distributions and use the given methods to generate long run samples to answer questions

**Simulating samples of Random Variables on the basis of samples from U(0,1)**

# Discrete Distributions(Known)

**Simulating samples of Random Variables on the basis of samples from U(0,1)**

- **Bernoulli(p)**
  1) If $u < p$ return 1 else return 0
- **Binomial(n, p)**
  1) Generate n samples from Bernoulli(p)
  2) Count the number of ‘1’ samples
- **Geometric(p)**
  1) Keep generating samples from Bernoulli(p) till ‘1’ sample is generated
  2) Return number of samples generated
- **Negative-Binomial(k, p)**
  1) Generate k samples from Geometric(p)
  2) Add the values together

# Discrete Distributions( General)

- **Method 1**

  1)Generate $U$

  2)Find $i$ such that $F(i-1) \leq U < F(i)$ where $F(x)$ is the cumulative distribution function

- **Method 2**

  1)Generate $U$

  2)Find the smallest possible value of $i$ such that $F(i) > U$, where $F(x)$ is the cumulative distribution function

# Continuous Distributions ( General)

- **Method 1 (Rejection Method)**
  1. Find $\mathrm{a}, \mathrm{b}, X$ such that $\mathrm{a}, \mathrm{b}$ and $0, c$ forms a bounding box on $f(x)$ where $f(x)$ is the probability distribution function $[\forall x: a \leq x \leq b\ , 0 \leq f(x) \leq c]$
  2. Generate $\mathrm{U}_1, \mathrm{U}_2$
  3. $X = a + (b - a)U_1$ and $Y = cU_2$
  4. If $\mathrm{Y} \leq f(x)$ accept $X$ as the desired sample. Else return to step 2

- **Method 2 (Inverse Transform Method)**
  1. Generate $U$
  2. Return $\mathrm{F}^{-1}(U)$ where $\mathrm{F}^{-1}(X)$ is the inverse of $F(X)$, the cumulative density function
  3. Note: Can also work for Discrete Distributions with Invertible $F(X)$

- **Continuous distributions (Known)**
  - Gamma: Generate α samples from Exponential(λ) and add them
  - Uniform $(a, b)$
    1. Generate $U$
    2. Return $U * (b - a) + a$

# Other Special Methods

- **Poisson (λ)**
  1. Generate $\mathrm{U}_1, \mathrm{U}_2$ , …
  2. Find the largest value $k$ for which $\mathrm{U}_1 * \mathrm{U}_2 \ * \cdots * \mathrm{U}_\mathrm{k} \geq \mathrm{e}^{-\lambda}$
  3. Return $k$

- **Normal(μ, σ) [Box-Mueller Transform]**
  1) Generate $\mathrm{U}_1, \mathrm{U}_2$
  2) $\mathrm{Z}_1 = \sqrt{-2\ln(\mathrm{U}_1)}\cos(2\pi\mathrm{U}_2)$
  3) $\mathrm{Z}_2 = \sqrt{-2\ln(\mathrm{U}_1)}\sin(2\pi\mathrm{U}_2)$
  4) $\mathrm{X}_1 = \mathrm{Z}_1\mathrm{\sigma} + \mathrm{\mu}$
  5) $\mathrm{X}_2 = \mathrm{Z}_2\mathrm{\sigma} + \mathrm{\mu}$

**Ex: Inverse Transform Methods**

- Geometric: $X = \left\lfloor \frac{\ln(1-U)}{\ln(1-P)} \right\rfloor$

- Exponential: $\mathrm{X} = -\frac{1}{\lambda}\ln(1-U)$

# Pseudo-Random Numbers

- A range of pseudo-random number generators are used, including
  - Congruential random number generator
    - Use a very simple equation to calculate the next pseudo-random number (as a Natural number) based on the previous pseudo-random number

$$x_{k+1} = (ax_k + c) \bmod m \quad , \quad u_{k+1} = x_{k+1}/m$$

      - Once a number repeats, the entire sequence repeats
  - Fibonacci generator
    - Next pseudo-random number is generated directly as a real number based on two previous pseudo-random numbers (as product, sum, difference, ...)

$$x_{k+1} = \begin{cases} x_{k-l_1} - x_{k-l_2} + 1 & if \quad x_{k-l_1} - x_{k-l_2} < 0 \\ x_{k-l_1} - x_{k-l_2} - 1 & if \quad x_{k-l_1} - x_{k-l_2} > 1 \\ x_{k-l_1} - x_{k-l_2} + 1 & otherwise \end{cases}$$

# Pseudo-Random Numbers -Congruential Generator

- Congruential random number generators are a very common type of generator.

$$x_{k+1} = (ax_k + c) \mod m \quad , \quad u_{k+1} = x_{k+1}/m$$

  - Performance depends on the choice of parameters a, c, and m
    - m determines the range of numbers that the random number generator can generate
    - Non-careful choice of a, b, and m can lead to statistically biased random number sequences
      - One example of this is the random generator used in early IBM computers: a=65539, b=0, m=$2^{31}$
  - m is often chosen as the maximum representable number ( to Minimizes repetition)
  - http://en.wikipedia.org/wiki/Linear_congruential_generator

# Pseudo-Random Numbers- Fibonacci Generator

- Fibonacci generators and their variations are replacing congruential pseudo-random number generators.
  - Fibonacci generators can directly generate floating point numbers as difference, sum, or product of previous numbers

$$x_k = (x_{k-i} \circ x_{k-j}) \ \ MOD \ \ m$$

$$e.g.: \quad x_k = (x_{k-i} - x_{k-j}) \ \ MOD \ \ 1$$

$$x_k = (x_{k-i} * x_{k-j}) \ \ MOD \ \ k$$

    - MOD operation ensures that numbers stay within the required range
  - Performance depends on the choice of parameters i, j, m
    - Common choice for a subtractive generator are i=17, j=5, m=1
  - Performance also depends on choice of initial elements

# Nonuniform Distributions

- Using a random number generator for uniform random numbers, a number of other distributions can be obtained
  - Shifted uniform distribution: To generate a uniform distribution in interval [a,b) we can simply transform a uniform random number in the interval [0,1)

$$x = (b-a)u + a$$

  - To achieve another distribution p(x) we can use its cumulative distribution P(x) and a uniform random number such that u=P(x), i.e. $x = P^{-1}(u)$
  - Exponential distribution: $p(x)=\lambda e^{-\lambda x}$

$$x = \frac{-\log(1-u)}{\lambda}$$

# Quasi-Random Numbers

Quasi-Random number generators generate numbers that uniformly cover the space but do not individually appear random

- Consecutive numbers are not unbiased

An example quasi random number generator:

- **Base-p low-discrepancy sequence (p is prime).** Have a sequence number i (the index). For the $i^{th}$ number use the base-p representation mirrored, and put after a decimal point:
- Base-3 is called the Halton sequence

| i | B-2 rep | Mirror | In dec |
|---|---|---|---|
| 1 | 1 | 0.1 | 0.5 |
| 2 | 10 | 0.01 | 0.25 |
| 3 | 11 | 0.11 | 0.75 |
| 4 | 100 | 0.001 | 0.125 |
| 5 | 101 | 0.101 | 0.625 |
| 6 | 110 | 0.011 | 0.375 |
| 7 | 111 | 0.111 | 0.875 |

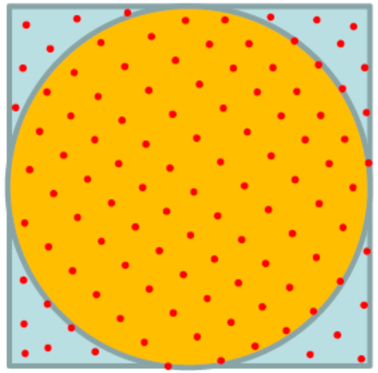
# Monte Carlo Methods

- Monte Carlo methods randomly draw samples from a distribution and determining values for each sample
  - Monte Carlo for expected value problems
    - Sample from the distribution and average the function values at the samples to get the expected value over the given distribution
  - Monte Carlo for ratio problems
    - Sample from a distribution and determine the ratio of valid vs. invalid samples to compute the desired ratio
- Monte Carlo methods provide increasingly precise solutions as the number of samples increases but require
  - Knowledge of relevant probability distribution and function
  - Access to random numbers

# Monte Carlo Ratios

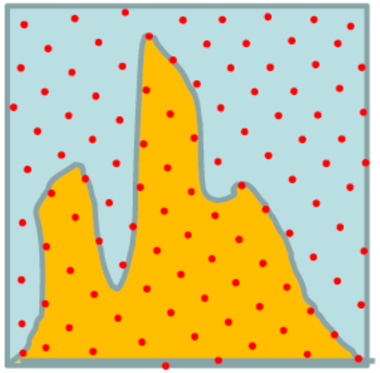
**“The usual” rain falls in a square example:**

- If the rainfall is uniform then the number of drops inside the circle vs. the number of total drops gives an estimate for the circle’s area and thus for $\pi$.



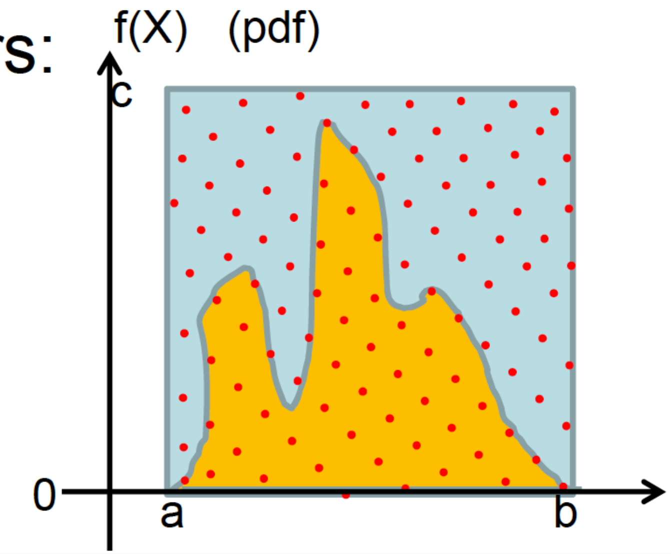
**Determining area of a function**

- Similarly, area (integral) of a function can be determined by a Monte-Carlo ratio method



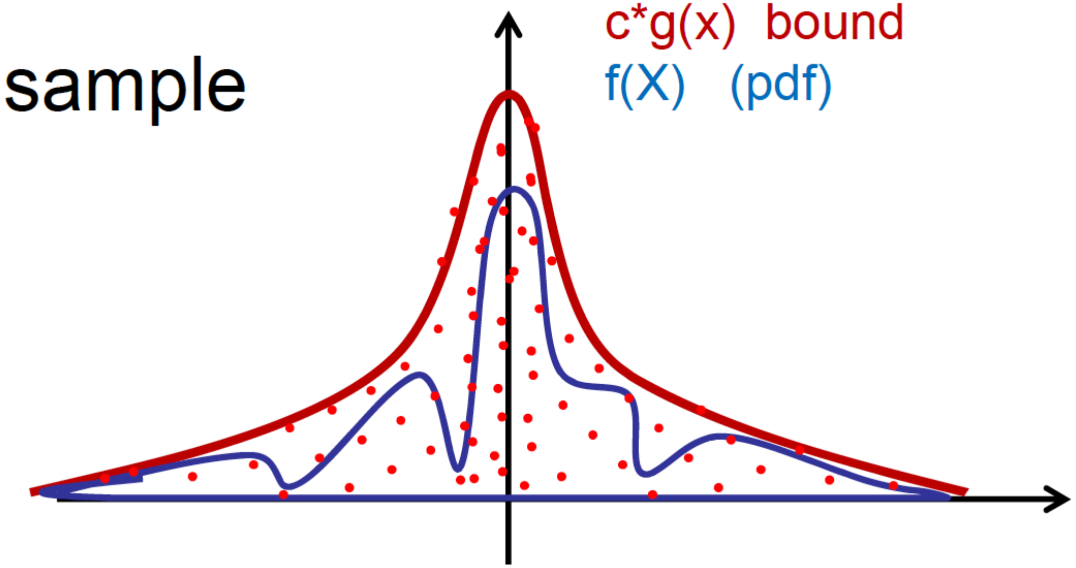
# Simple Rejection Sampling

- A “Monte Carlo Ratios” method.
  - Let’s say we have a random variable X with an “ugly” probability density function f(X).
  - We want to model this variable, i.e., draw sample from an f(X) distribution.
  - Draw two random uniform numbers:
    - u from U(0,c)
    - x from U(a,b)
  - Accept x as a sample from X iff u<f(x) (reject otherwise)



# Generalized Rejection Sampling

- We assumed that f(x) can be bounded by a rectangle. What if this is not true?
- We need to find a pdf g(x) that bounds f(x)
  $f(x) \le g(x)*c$ for $\forall x$
- Generate a sample x from a random variable with a pdf of g(x).
- Generate a uniform random sample u from U(0,1)
  accept x if $u < \frac{f(x)}{c*g(x)}$
  reject otherwise



# Importance Sampling for Monte-Carlo Methods

- Sometimes it is more efficient to **use a different distribution from the one needed** for the solution to generate the samples
  - E.g. while Monte-Carlo integration requires a uniform distribution, it might be more efficient for high-dimensional functions which are 0 for large parts of the space, to sample such that more samples are generated in areas with higher function values.
- Importance sampling allows to estimate the result of an evaluation with distribution density *q(x)* while taking samples from a distribution with density *p(x)* by re-weighing the samples with an importance weight
  - *p(x)* can not be 0 for any *x* at which *q(x)* is not 0

$$\frac{q(x_i)}{p(x_i)} f(x_i)$$

# Convergence Rates of Monte Carlo Methods

- Monte Carlo Simulations with pseudo-random numbers converge with the inverse of the square root of the number of samples

$$Error \propto 1/\sqrt{n}$$

  - This is the result of the way the expected value of the variance changes

$$E\left[\left(\frac{x_1 + \ldots + x_n}{n} - \hat{x}\right)^2\right] = \frac{1}{n^2}\sum E\left[\left(x_i - \hat{x}\right)^2\right] = \frac{1}{n^2} n\sigma^2 = \frac{\sigma^2}{n}$$

$$\sqrt{E\left[\left(\frac{x_1 + \ldots + x_n}{n} - \hat{x}\right)^2\right]} \propto \frac{1}{\sqrt{n}}$$

# Convergence Rates of Monte Carlo Methods

- Monte Carlo Simulations with quasi-random numbers converge at a different rate that depends on the dimensionality of the random number, d
  - Monte Carlo for expected value problems:

$$Error \;\propto\; (\ln\, n)^d \big/ n$$

  - Monte Carlo for ratio problems:

$$Error \;\propto\; 1 \Big/ n^{\frac{1}{2}+\frac{1}{2d}}$$

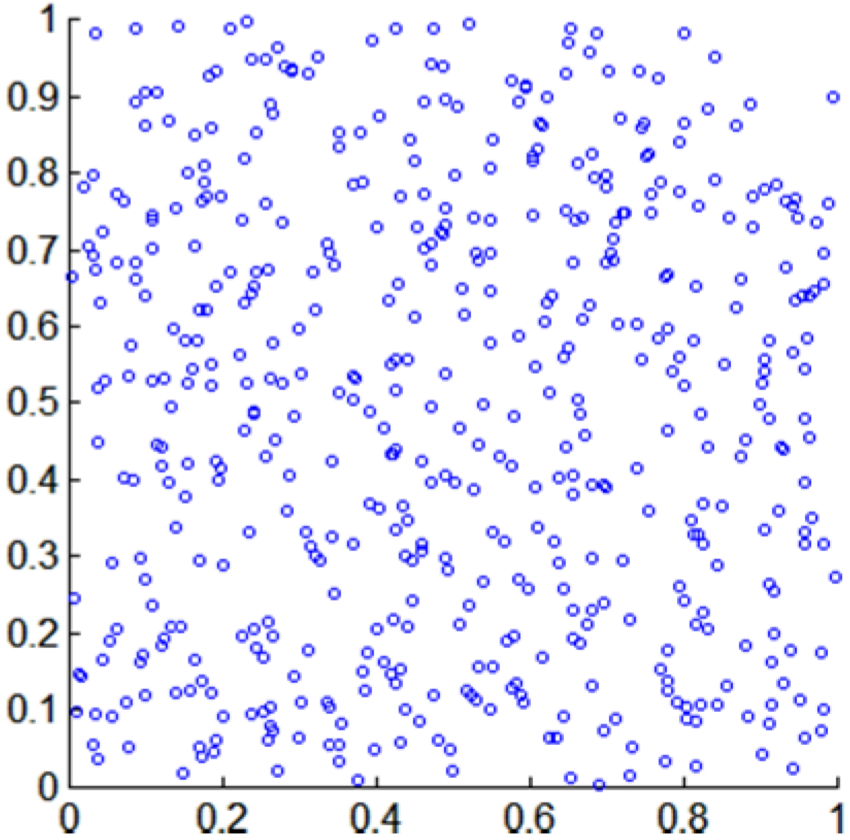
- **Monte Carlo methods converge faster with quasi-random numbers than with pseudo-random numbers**

# Pseudo-Random vs. Quasi-Random Numbers

- Pseudo-Random number generators are usually designed such that the sequence of numbers are uncorrelated and pass a number of statistical independence tests. Quasi-Random numbers will usually fail these sequence tests and show high sequence correlations.
- Quasi-Random numbers lead to faster convergence in Monte Carlo methods where only the uniformity of the distribution matters but not the randomness of the actual sequence.
    - E.g. Monte-Carlo integration
- Pseudo-Random numbers perform much better in situations where the randomness of the sequence of numbers matters
    - E.g. Monte-Carlo simulation of random processes

# Pseudo vs. Quasi-Random Numbers

Pseudo Uniform
Random Scatter




Quasi Random Scatter