Thesis 6

Anna Lucia Lamacchia
Statistics

January 11, 2024

Algorithms for Random Variates Generation

1 Some Algorithms

Algorithms for random variates generation play a crucial role in simulation and modeling across various fields, including statistics, computer science, and engineering. These algorithms are designed to generate random numbers that follow specific probability distributions.
These algorithms are fundamental tools for Monte Carlo simulations, probabilistic modeling, and various applications that involve the generation of random variates following specific probability distributions. The choice of an algorithm depends on the characteristics of the target distribution and the computational requirements of the application.
Here is a summary of key aspects related to algorithms for random variates generation:

1. **Inverse Transform Method**:
   - **Idea**: Utilizes the cumulative distribution function (CDF) of a distribution to transform uniform random variables into random variables following the desired distribution.
   - **Process**: Involves solving for the inverse CDF and applying it to uniform random variates.
   - **Advantages**: Conceptually simple and applicable to a wide range of distributions.
   - **Limitations**: May not be computationally efficient for distributions with complex CDFs.
2. **Acceptance-Rejection Method**:

- **Idea**: Uses an envelope distribution to generate random variates by accepting or rejecting sampled points.
- **Process**: Generates candidate points from the envelope distribution and accepts them based on a comparison with the target distribution.
- **Advantages**: Applicable to distributions where the inverse transform method is impractical.
- **Limitations**: Requires a suitable envelope distribution, and the method may involve rejection, making it less efficient.

3. **Box-Muller Transform**:
   - **Idea**: Converts pairs of independent uniform random variables into pairs of independent standard normal random variables.
   - **Process**: Applies trigonometric transformations to uniform variates.
   - **Advantages**: Efficient for generating standard normal variates.
   - **Limitations**: Limited to normal distributions.

4. **Marsaglia Polar Method**:
   - **Idea**: Similar to the Box-Muller transform, generates pairs of standard normal variates.
   - **Process**: Uses rejection to discard points outside the unit circle.
   - **Advantages**: Efficient for generating standard normal variates.
   - **Limitations**: Limited to normal distributions.

5. **Alias Method**:
   - **Idea**: Efficiently generates discrete random variates by using precomputed tables and random selections.
   - **Process**: Combines multiple values into an alias table and uses uniform random variables for selection.
   - **Advantages**: Particularly useful for discrete distributions.
   - **Limitations**: Requires additional memory for alias tables.

6. **Sampling from Beta and Gamma Distributions**:
   - **Idea**: Utilizes properties of gamma and beta functions for efficient sampling.

- **Process**: Relies on properties of gamma and beta distributions to generate random variates.
- **Advantages**: Efficient for generating variates from gamma and beta distributions.
- **Limitations**: Specialized to specific distributions.

7. **Pseudo-Random Number Generators (PRNGs)**:
   - **Idea**: PRNGs are algorithms that generate sequences of pseudo-random numbers that exhibit statistical properties similar to true random variables.
   - **Process**: These algorithms involve iterative mathematical operations starting from an initial seed to produce a sequence of apparently random numbers.
   - **Advantages**: Efficient, deterministic, and widely applicable in various simulation contexts.
   - **Limitations**: PRNGs are deterministic, meaning that given the same seed, they produce the same sequence. This property can be a limitation in some applications, and the quality of PRNGs is crucial for ensuring statistical properties.
8. **Markov Chain Monte Carlo (MCMC)**:
   - **Idea**: MCMC is a sampling technique that constructs a Markov chain to generate sequences of dependent random variates.
   - **Process**: It involves iteratively updating the state of the chain based on acceptance probabilities, which allows for the simulation of complex distributions.
   - **Advantages**: Effective for simulating complex distributions, provides dependent samples suitable for Bayesian inference.
   - **Limitations**: MCMC methods can be computationally intensive, and convergence diagnostics are necessary to ensure the Markov chain has reached a stationary distribution.
9. **Poisson Disk Sampling**:
   - **Idea**: Poisson Disk Sampling is a spatially aware random sampling algorithm used in computer graphics and computational geometry.

- **Process**: It generates points in space such that each point is a minimum distance away from others, ensuring a spatially uniform distribution.
- **Advantages**: Useful for applications where spatial distribution matters, such as computer graphics and mesh generation.
- **Limitations**: The method can be computationally demanding, especially in high-dimensional spaces, and the choice of minimum distance influences the generated patterns.

2 Simulation of Box-Muller Transform in Python

In this simulation, the Box-Muller transform is used to generate pairs of independent standard normal random variables (z0 and z1) from pairs of independent, uniformly distributed random variables. The resulting histogram visually demonstrates the characteristics of the standard normal distribution.

```python
import numpy as np
import matplotlib.pyplot as plt

# Number of samples
num_samples = 1000

# Generate pairs of independent uniform random variables
u1 = np.random.rand(num_samples)
u2 = np.random.rand(num_samples)

# Box-Muller Transform
z0 = np.sqrt(-2 * np.log(u1)) * np.cos(2 * np.pi * u2)
z1 = np.sqrt(-2 * np.log(u1)) * np.sin(2 * np.pi * u2)

# Plot histogram of generated normal variates
plt.hist(z0, bins=30, density=True, alpha=0.5, color='blue', label='z0')
plt.hist(z1, bins=30, density=True, alpha=0.5, color='orange', label='z1')
plt.xlabel('Value')
plt.ylabel('Probability Density')
plt.title('Box-Muller Transform Simulation')
plt.legend()
plt.show()
```

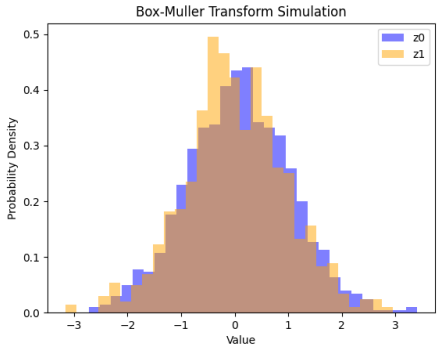

Figure 1: Simulation