Biostatistics 615/815 Lecture 16: Monte-carlo methods
Importance sampling

Hyun Min Kang

March 15th, 2011
Grading

- Midterm will be given by Thursday
- All the other homeworks will be given by next Tuesday

815 Update

- Send a brief progress update on the project
- Schedule meeting with instructor if needed
#include <iostream>
#include <cstdlib>

int main(int argc, char** argv) {
    int n = (argc > 1) ? atoi(argv[1]) : 1;
    int seed = (argc > 2) ? atoi(argv[2]) : 0;

    srand(seed); // set seed -- same seed, same pseudo-random numbers

    for(int i=0; i < n; ++i) {
        std::cout << (double)rand()/RAND_MAX << std::endl;
        // generate value between 0 and 1
    }

    return 0;
}
Recap: Good vs. bad random numbers

- Images using true random numbers from random.org vs. rand() function in PHP
- Visible patterns suggest that rand() gives predictable sequence of pseudo-random numbers
Recap: Generating uniform random numbers in C++

```cpp
#include <iostream>
#include <boost/random/uniform_int.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/variate_generator.hpp>
#include <boost/random/mersenne_twister.hpp>

int main(int argc, char** argv) {
    typedef boost::mt19937 prgType; // Mersenne-twister: a widely used lightweight pseudo-random-number-generator
    prgType rng; // lightweight pseudo-random-number-generator
    boost::uniform_int<> six(1,6); // uniform distribution from 1 to 6
    boost::variate_generator<prgType&, boost::uniform_int<> > die(rng,six);
    // die maps random numbers from rng to uniform distribution 1..6

    int x = die(); // generate a random integer between 1 and 6
    std::cout << "Rolled die: " << x << std::endl;

    boost::uniform_real<> uni_dist(0,1);
    boost::variate_generator<prgType&, boost::uniform_real<> > uni(rng,uni_dist);
    double y = uni(); // generate a random number between 0 and 1
    std::cout << "Uniform real: " << y << std::endl;
    return 0;
}
```
Today

Sampling from complex distributions

- Monte-Carlo Methods
- Importance Sampling
Monte-Carlo Methods

Informal definition

- Approximation by random sampling
- Randomized algorithms to solve deterministic problems approximately.

An example problem

Calculating

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

where \( f(x) \) is a complex function with \( 0 \leq f(x) \leq 1 \)

The problem is equivalent to computing \( E[f(u)] \) where \( u \sim U(0, 1) \).
The crude Monte-Carlo method

Algorithm

- Generate $u_1, u_2, \ldots, u_B$ uniformly from $U(0, 1)$.
- Take their average to estimate $\theta$

$$\hat{\theta} = \frac{1}{B} \sum_{i=1}^{B} f(u_i)$$
The crude Monte-Carlo method

Algorithm

- Generate $u_1, u_2, \cdots, u_B$ uniformly from $U(0, 1)$.
- Take their average to estimate $\theta$

$$\hat{\theta} = \frac{1}{B} \sum_{i=1}^{B} f(u_i)$$

Desirable properties of Monte-Carlo methods

- Consistency: Estimates converges to true answer as $B$ increases
- Unbiasedness: $E[\hat{\theta}] = \theta$
- Minimal Variance
Analysis of crude Monte-Carlo method

### Bias

\[
E[\hat{\theta}] = \frac{1}{B} \sum_{i=1}^{B} E[f(u_i)] = \frac{1}{B} \sum_{i=1}^{B} \theta = \theta
\]
Analysis of crude Monte-Carlo method

Bias

\[ E[\hat{\theta}] = \frac{1}{B} \sum_{i=1}^{B} E[f(u_i)] = \frac{1}{B} \sum_{i=1}^{B} \theta = \theta \]

Variance

\[ \sigma^2 = \frac{1}{B} \int_{0}^{1} (f(u) - \theta)^2 \, du = \frac{1}{B} E[f(u)^2] - \frac{\theta^2}{B} \]
Analysis of crude Monte-Carlo method

Bias

\[
E[\hat{\theta}] = \frac{1}{B} \sum_{i=1}^{B} E[f(u_i)] = \frac{1}{B} \sum_{i=1}^{B} \theta = \theta
\]

Variance

\[
\sigma^2 = \frac{1}{B} \int_{0}^{1} (f(u) - \theta)^2 \, du = \frac{1}{B} E[f(u)^2] - \frac{\theta^2}{B}
\]

Consistency

\[
\lim_{B \to \infty} \hat{\theta} = \theta
\]
Accept-reject (or hit-and-miss) Monte Carlo method

Algorithm

1. Define a rectangle $R$ between $(0,0)$ and $(1,1)$
2. Set $h = 0$ (hit), $m = 0$ (miss).
3. Sample a random point $(x, y) \in R$.
4. If $y < f(x)$, then increase $h$. Otherwise, increase $m$.
5. Repeat step 3 and 4 for $B$ times.
6. $\hat{\theta} = \frac{h}{h+m}$.
Bias

Let $u_i, v_i$ follow $U(0, 1)$.

$$E[\hat{\theta}] = E\left[ \frac{h}{h + m} \right] = \theta$$
Analysis of accept-reject Monte Carlo method

Bias

Let $u_i, v_i$ follow $U(0, 1)$.

$$E[\hat{\theta}] = E\left[ \frac{h}{h + m} \right] = \theta$$

Variance

$$\sigma^2 = \frac{\theta(1 - \theta)}{B}$$
Introduction

Integration

Importance sampling

Rejection sampling

Summary

Which method is better?

\[
\sigma_{AR}^2 - \sigma_{crude}^2 = \frac{\theta(1 - \theta)}{B} - \frac{1}{B} E[f(u)^2] + \frac{\theta^2}{B}
\]

\[
= \frac{\theta - E[f(u)]^2}{B}
\]

\[
= \frac{1}{B} \int_0^1 f(u)(1 - f(u)) \, du \geq 0
\]

The crude Monte-Carlo method has less variance than accept-rejection method
Revisiting The Crude Monte Carlo

\[ \theta = E[f(u)] = \int_0^1 f(u) \, du \]

\[ \hat{\theta} = \frac{1}{B} \sum_{i=1}^{B} f(u_i) \]

More generally, when \( x \) has pdf \( p(x) \), if \( x_i \) is random variable following \( p(x) \),

\[ \theta = E_p[f(x)] = \int f(x) p(x) \, dx \]

\[ \hat{\theta} = \frac{1}{B} \sum_{i=1}^{B} f(x_i) \]
Importance sampling

Let $x_i$ be random variable, and let $p(x)$ be an arbitrary function.

$$
\theta = E_u[f(x)] \int f(x) \, dx = \int \frac{f(x)}{p(x)} p(x) \, dx = E_p \left[ \frac{f(x)}{g(x)} \right]
$$

$$
\hat{\theta} = \frac{1}{B} \sum_{i=1}^{B} \frac{f(x_i)}{p(x_i)}
$$
Key Idea

- When \( f(x) \) is not uniform, variance of \( \hat{\theta} \) may be large.
- The idea is to pretend sampling from (almost) uniform distribution.
Importance sampling reduces the variance of $\theta$

Figure 1: Three different importance sampling functions (dotted lines) used to integrate the standard normal density (solid line) from -50 to 50. Top panels are the density curves and bottom panels are histograms of 5,000 Monte Carlo estimates of the area (which is exactly 1) using $n = 1,000$. 

@ERIC C. ANDERSON. Corrections, comments?→ eric@cqs.washington.edu
More on rejection sampling

Framework for random sampling with inversion CDF

- Draw \( u \sim U(0, 1) \).
- Set \( x = F^{-1}(u) \) for a CDF \( F \).
- Then \( x \) is a random variable such that \( x \sim F \).
More on rejection sampling

Framework for random sampling with inversion CDF

- Draw \( u \sim U(0, 1) \).
- Set \( x = F^{-1}(u) \) for a CDF \( F \).
- Then \( x \) is a random variable such that \( x \sim F \)

Rejection sampling

1. Sample \((x, u)\) from rectangle covering \( \min f(x) \leq u \leq \max f(x) \).
2. Accept \( x \) if \( u \leq f(x) \).
3. Otherwise, reject and repeat step 1 and 2 until accept
4. Repeat step 1-3 to obtain multiple random variable following \( x \sim F \)
Summary

- Crude Monte Carlo method: Sampling from uniform distribution for estimating $\theta$.
- Rejection sampling: Used for calculating $\theta$, or generating random samples
- Importance sampling: Reweight the probability distribution to reduce the variance in the estimation