LA-UR-16-24304
Approved for public release; distribution is unlimited.

Title: Fundamentals of Monte Carlo

Author(s): Wollaber, Allan Benton

Intended for: Lecture material for Parallel Computing summer school

Issued: 2016-06-17

Fundamentals of Monte Carlo

Allan B. Wollaber

June 16, 2016

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Introduction

- Background
- A Simple Example: Estimating π

Why does this even work?

- The Law of Large Numbers
- The Central Limit Theorem

How to Sample

- Inverse Transform Sampling
- Rejection

An Example from Particle Transport

Outline

Outline

What is Monte Carlo?

Welcome to Los Alamos, the birthplace of “Monte Carlo” for computational physics.

- Stanislaw Ulam, John von Neumann, and Nicholas Metropolis are credited as the founders of modern Monte Carlo methods
- The name “Monte Carlo” was chosen in reference to the Monte Carlo Casino in Monaco (purportedly a place where Ulam’s uncle went to gamble)
- The central idea (for us) – to use computer-generated “random” numbers to determine expected values or estimate equation solutions – has since spread to many fields

What is Monte Carlo?

The first thoughts and attempts I made to practice [the Monte Carlo Method] were suggested by a question which occurred to me in 1946 as I was convalescing from an illness and playing solitaires. The question was what are the chances that a Canfield solitaire laid out with 52 cards will come out successfully? After spending a lot of time trying to estimate them by pure combinatorial calculations, I wondered whether a more practical method than "abstract thinking" might not be to lay it out say one hundred times and simply observe and count the number of successful plays... Later [in 1946], I described the idea to John von Neumann, and we began to plan actual calculations.

– Stanislaw Ulam

Definitions

You will need to know what a *probability density function*, or PDF is. For a continuous random variable $x \in (-\infty, \infty)$ and scalars $a < b$, an implicit definition to remember is:

$$Pr\,(a \leq x \leq b) = \int_a^b p(x)\,dx$$

You should always mentally associate $p(x)$ with dx (integrated or not). This will help you if you ever want to change variables – it is not enough to substitute x for u without taking care of dx/du!
The PDF $p(x)$ has the properties that it is *non-negative* and that

$$\int_{-\infty}^{\infty} p(x)\,dx = 1\,.$$

Note that $p(x)$ need not be less than 1.

Definitions

The integrated version of a PDF is a *cumulative distribution function*, or CDF (note that the D's are different in PDF and CDF).

$$Pr(x \leq y) = P(y) = \int_{-\infty}^{y} p(x)\,dx\,.$$

All CDFs must be non-decreasing and limit to a maximum value of 1.0.

We frequently use this fact to "normalize" a function in order to convert it into a PDF from which we can sample.

Definitions

PDFs can be arbitrarily complicated, but we can understand a lot about them by describing them with a handful of scalar quantities. We'll consider two "moments" of PDFs in this talk (you can go higher if you like),

The *mean* or *expected value*, μ is the first moment of $p(x)$

$$\mu = \int_{-\infty}^{\infty} xp(x)\,dx\,, \quad \text{or, for discrete,} \quad \mu = \sum_i x_i p_i$$

This weighting is one way to tell you the "central tendency" of $p(x)$. The next is the square difference of the PDF from the mean, or the *variance*, σ^2,

$$\sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)\,dx\,, \text{ or, for discrete, } \sigma^2 = \sum_i (x_i-\mu)^2 p_i\,,$$

We use σ^2 because σ is the *standard deviation*.

Definitions

If we could analytically compute those integrals, our job would be easy. Very often, we are only concerned with μ or $E[f(x)]$ in computational physics (e.g., the expected value of energy deposited in some detector or a mesh cell). We use Monte Carlo to try to estimate μ by repeatedly sampling $p(x)$ (even if we can't write $p(x)$ explicitly!).

The *sample mean* of x for N random samples of $p(x)$ is defined as

$$m = \frac{1}{N} \sum_{i=1}^{N} x_i \,,$$

and *sample variance* as

$$s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - m)^2 \,.$$

so that s is the *sample standard deviation*.

Definitions

And, the last definition we will need is the *variance of the sample mean* after N trials

$$\sigma_m^2 = \frac{\sigma^2}{N},$$

or

$$\sigma_m = \frac{\sigma}{\sqrt{N}},$$

which pithily tells you the main criticism of Monte Carlo – *to obtain an extra digit of accuracy, you must run 100X more samples*. There is a tremendous amount of research in building clever ways to reduce σ^2 (this is called "variance reduction").

Let's try a simple example...

Outline

Estimating π

To help "build intuition", we'll start with a well-used example:

- Let's assume that we know the formula for a unit circle ($x^2 + y^2 = 1$), and that we know that this circle inscribes the area π, but we don't know the value of π.
- The idea is to randomly "throw darts" at a square that inscribes the circle (presumably, we can easily calculate the area of the square, but not the circle).
- We count the darts that land inside the circle by scoring the area of the square.
- The darts that land outside the circle score *zero*

Estimating π

Assuming we throw darts truly randomly, our probabilities are:

- Event 1: Land inside the circle with probability $p_1 = \frac{\pi}{4}$, score 4.0
- Event 2: Land outside the circle with probability $p_2 = 1 - \frac{\pi}{4}$, score 0.0

By the definition of the mean, μ,

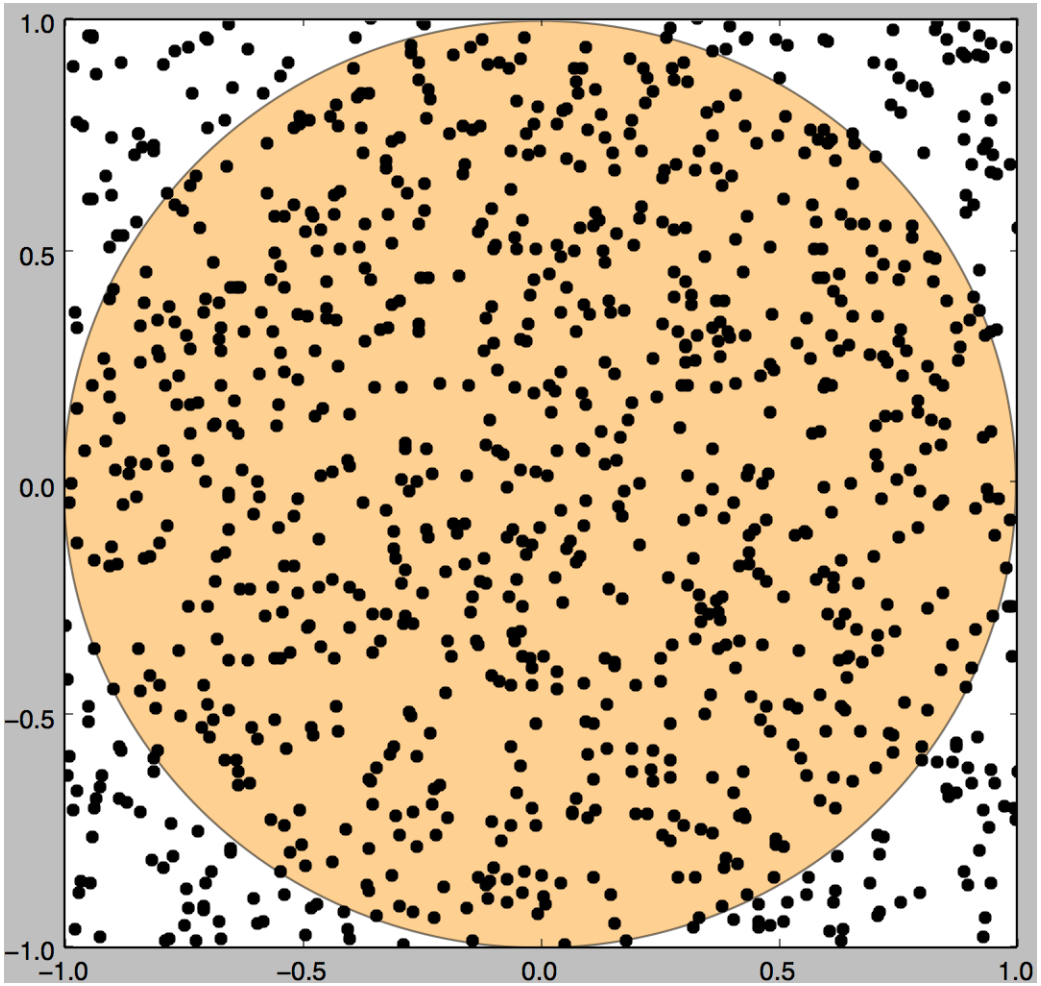
$$\mu = 4\frac{\pi}{4} + 0\left(1 - \frac{\pi}{4}\right) = \pi$$

and, by the definition of the variance, σ^2,

$$\sigma^2 = (4-\pi)^2\frac{\pi}{4} + (0-\pi)^2\left(1 - \frac{\pi}{4}\right) = \pi(4-\pi) \approx 2.69$$

$$\sigma \approx 1.64$$

Estimating π



An example with 1000 points

Example Python Code

```
import numpy as np

Ns = np.logspace(2.0, 7.0, num=6)
# Initialize RNG for repeatable results
my_rng = np.random.RandomState(1234)

for N in Ns:
   #Each loop provides a single estimate of PI
   tally = 0.0    # This is what we score into
   tally2 = 0.0   # This is the square of what we score into
   for i in range(int(N)):
      x = 2.0*my_rng.uniform()-1.0
      y = 2.0*my_rng.uniform()-1.0
      r2 = x*x + y*y
      if (r2 < 1.0):   # If the point is inside the circle...
         tally += 4.0
         tally2 += 16.0

   my_pi = tally / N;
   pi_err = np.sqrt((tally2/N - my_pi*my_pi)/((N-1)))
   print "The value of pi for ", N, " estimates is ", my_pi,
   print "+/- ", pi_err, ", off by ",  my_pi - np.pi
```

Example Python Code Output

```
The value of pi for 100.0 estimates is 3.32 +/-
0.151010067227 , off by 0.17840734641
The value of pi for 1000.0 estimates is 3.036 +/-
0.0541260901381 , off by -0.10559265359
The value of pi for 10000.0 estimates is 3.1192 +/-
0.0165760855651 , off by -0.0223926535898
The value of pi for 100000.0 estimates is 3.1454 +/-
0.00518467522691 , off by 0.00380734641021
The value of pi for 1000000.0 estimates is 3.141584 +/-
0.00164219020449 , off by -8.65358979318e-06
The value of pi for 10000000.0 estimates is 3.1406656 +/-
0.000519507676446 , off by -0.000927053589793
```

Outline

Well, it almost always works

We are going to *assume* two things about the probability distributions we intend to use for Monte Carlo:

1. They are “IID”: Independent and Identically Distributed
2. They have finite first and second moments (at least)

There are exceptions to these rules, but be very careful if you intend to use Monte Carlo when these do not apply!

Outline

The Law of Large Numbers

There are many of these (Strong, Weak, Uniform, Borel's...), but the idea in words is that, as you increase the number of samples of a PDF to estimate a mean, it converges (in some sense) to the true mean. The "weak" law is stated as:

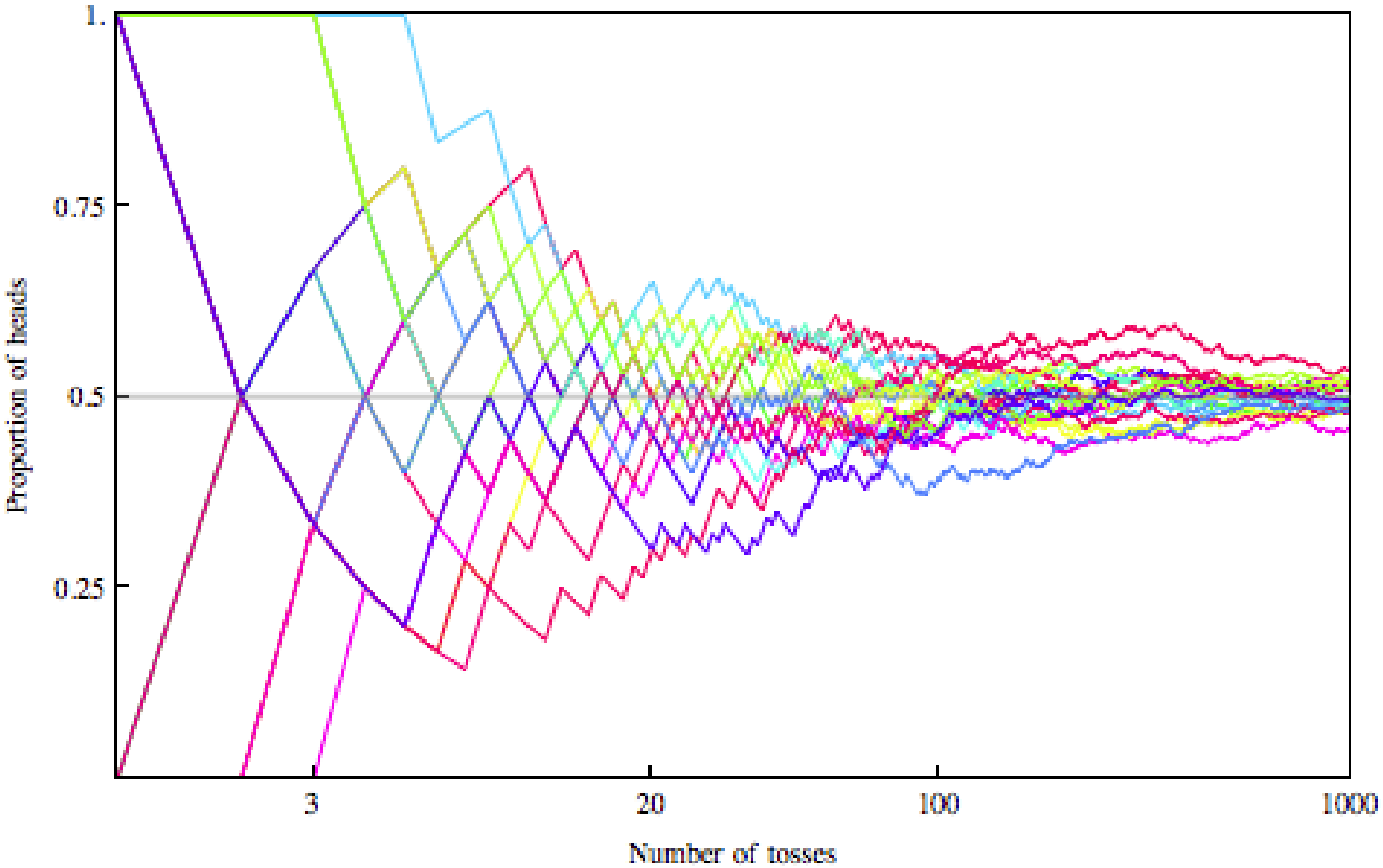
$$\lim_{N\to\infty} Pr\left(|m_N - \mu| > \epsilon\right) = 0\,,$$

where m_N is meant to indicate the sample mean for N samples. Essentially, this gives us the foundation to estimate μ with m.

However, it does not tell us how quickly m_N approaches μ.

The Law of Large Numbers

Proportion of heads from fair coin tosses



http://demonstrations.wolfram.com/SimulatedCoinTossingExperimentsAndTheLawOfLargeNumbers/

Outline

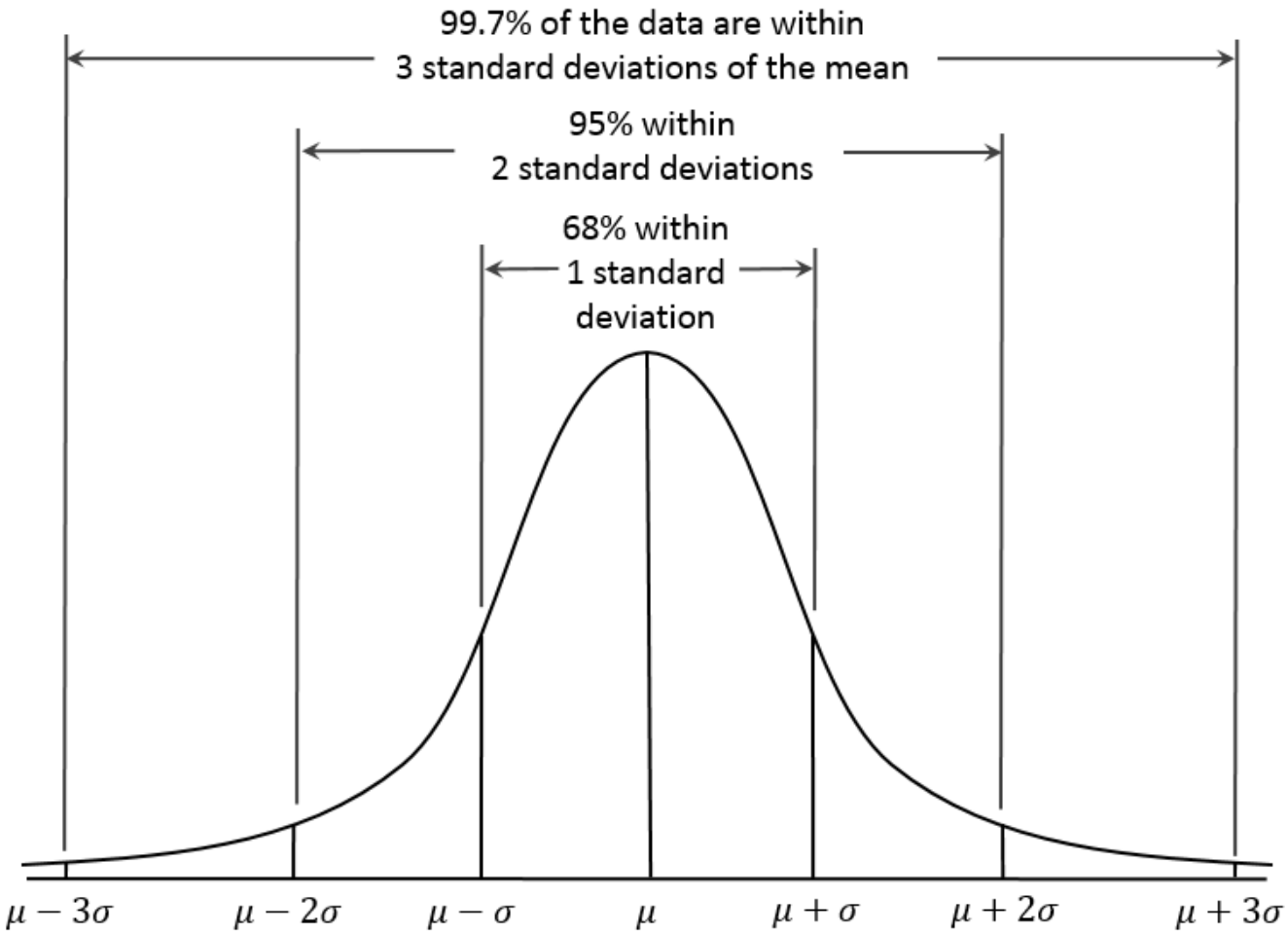
The Central Limit Theorem

I think this is one of the most beautiful results of probability theory. You have hopefully heard of the *standard normal distribution*, which is itself a PDF,

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2},$$

that has mean 0 and standard deviation 1. It is also called the Gaussian, or the “bell curve” due to its shape, and it is worth memorizing a few of its characteristics.

The Central Limit Theorem



The Standard Normal Distribution

By Dan Kernler - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=36506025

The Central Limit Theorem

In words, the Central Limit Theorem essentially states that the distribution of the sample means of *any* PDF limits to a Normal distribution *regardless of the underlying shape of the PDF itself*. The Lindeberg-Levy version is written as

$$\lim_{N\to\infty} Pr\left(\sqrt{N}(m_N - \mu) \leq z\right) = \Phi(z/\sigma)\,,$$

where Φ is the CDF for ϕ. There are also many versions and proofs of the CLT.

The Central Limit Theorem

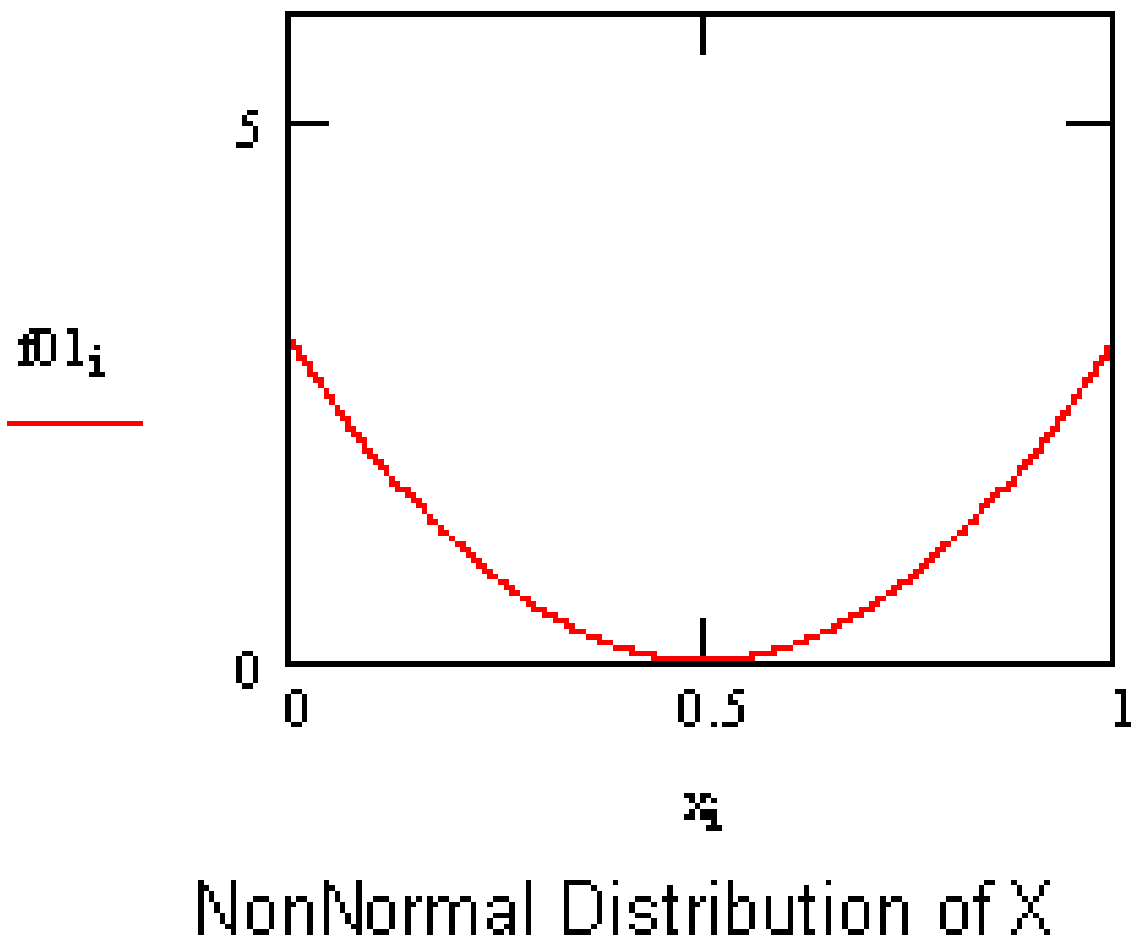
As an example of the Central Limit Theorem, we consider estimating means of a very non-normal distribution, taken from:

`http://www.statisticalengineering.com/central_limit_theorem_(parabola).html`

The example shows how estimating the mean from increasingly larger numbers of samples from a parabolic distribution. The first figure depicts the parabolic PDF. The next figure depicts the result of repeatedly computing the sample mean of two samples and binning the distributions.

The subsequent figures depict the results of binning 3, 4, 8, 16, and 32 samples (these should animate).

The Central Limit Theorem



http://www.statisticalengineering.com/central_limit_theorem_(parabola).html

Bounding our Errors

- Because of this fact, you can make precise statements about your estimates of the sample mean. For instance, we saw that m_{10^7} for π was 3.1406656.
- The standard deviation of the sample mean for this many samples is $\sigma/\sqrt{10^7} \approx 5.19 \times 10^{-4}$,
- Then our estimate was $1.7\sigma_{m_N}$ off the true mean. This happens with about 5% frequency.
- However, only 0.1% of the time should we expect an answer greater than $\pi + 3\sigma_{m_N}$, or 3.1431 . . . What should you do if your code estimates this as the answer?

Knowledge like this can come in handy for software testing.

Outline

"Random" Numbers

In practice, we don't really use random numbers, we use *pseudorandom* numbers from a *random number generator* (RNG)

- These are deterministic sequences of numbers that pass certain tests for randomness
- Most commonly, they are streams of *uniform random variates* (URVs), ξ_i, distributed on $[0, 1]$, $\xi_1, \xi_2, \xi_3, \ldots$
- Eventually, they repeat – this is the *period* of the generator
- A *seed* value is provided to reproducibly initialize the generator

You should be aware of the period of your RNG and its statistical integrity

Very Strange Things can happen if you use a "bad" RNG or otherwise find that the ξ_i are correlated.

Outline

Inverting a CDF

The most straightforward way to choose an x from a PDF $p(x)$ on a computer (if you can do it) is *inverse transform sampling*. The approach is

- Get a URV, ξ, from a random number generator
- Compute the value of x such that $\xi = P(x) = \int_{-\infty}^{x} p(x)\,dx$. In other words, compute the inverse $x = P^{-1}(\xi)$

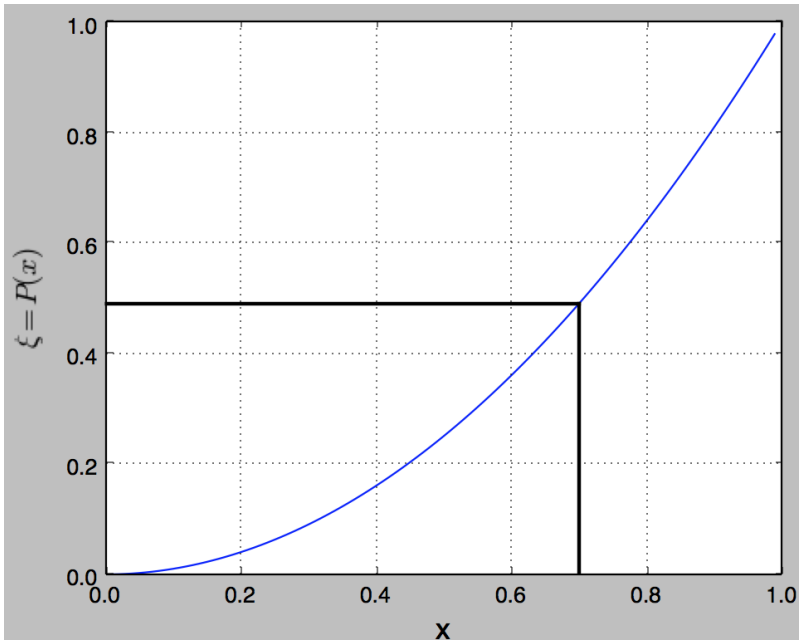
Roughly, this works because the URVs ξ are distributed uniformly on $[0, 1]$, which is the range of the CDF:

$$\xi = \int_0^{\xi} d\xi' = \int_{-\infty}^{x} p(x')\,dx' = P(x)$$

Unfortunately, this only works if $P(x)$ is invertible, and it can be expensive.

Inverting a CDF

As an example, consider $p(x) = 2x$ for $x \in [0, 1]$.[1] Then $P(x) = \int_0^x 2x' \, dx' = x^2$, so we can sample $p(x)$ by solving $\xi = x^2$, or $x = \sqrt{\xi}$. In this example, ξ is 0.49, so x is chosen to be 0.7



[1] This PDF comes up when sampling angle cosines for isotropically incident radiation according to Lambert's cosine law

Outline

Sampling with Rejection

Rejection sampling is the method of choice whenever $p(x)$ is sufficiently complicated and can be bounded above by some $f(x)$ that is simpler to sample from and has the same *support* as $p(x)$. The approach is to

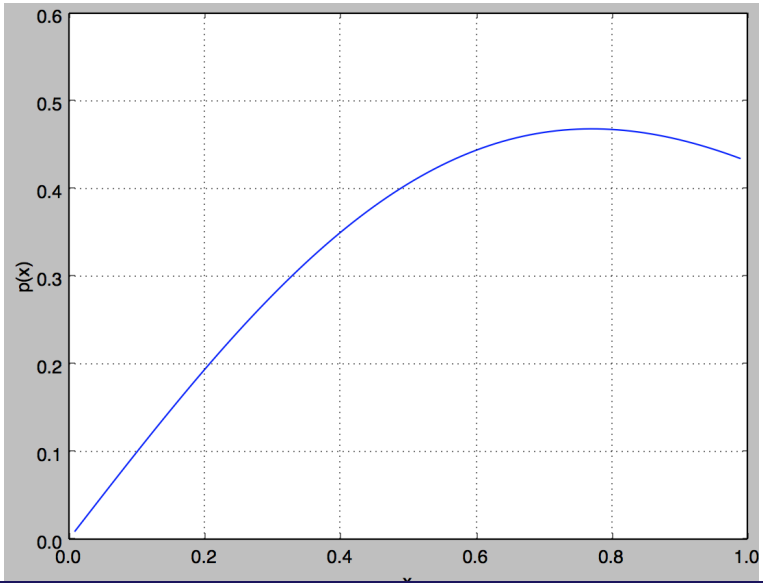
1. Calculate a ξ, and sample an x from $f(x)$ (such as by inverting it)
2. Compute the ratio $p(x)/f(x)$, which, by construction, is between 0 and 1
3. Pull another ξ from the generator
4. Accept x if $\xi \leq p(x)/f(x)$, otherwise return to step 1.

A geometric example would be repurposing the π example to produce points that uniformly sample the unit circle.

Sampling with Rejection

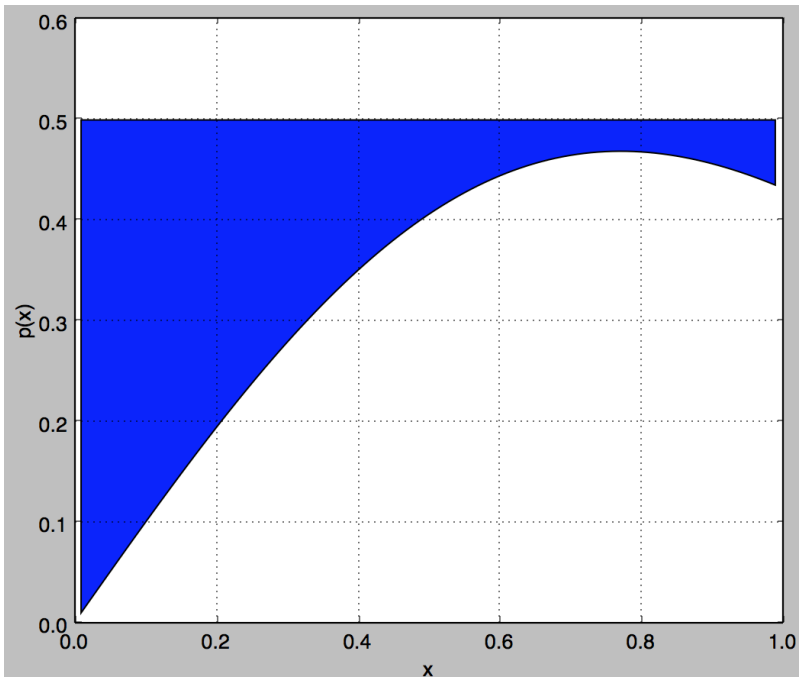
- A feature of rejection is that there is no need to normalize the function.
- The *efficiency* of a rejection method is the number of accepted x's over the total number of samples drawn.
- The choice of $f(x)$ is extremely important, as discarded x's are "wasted" computation

Consider sampling from $e^{-x^2}\sinh(x)\ldots$



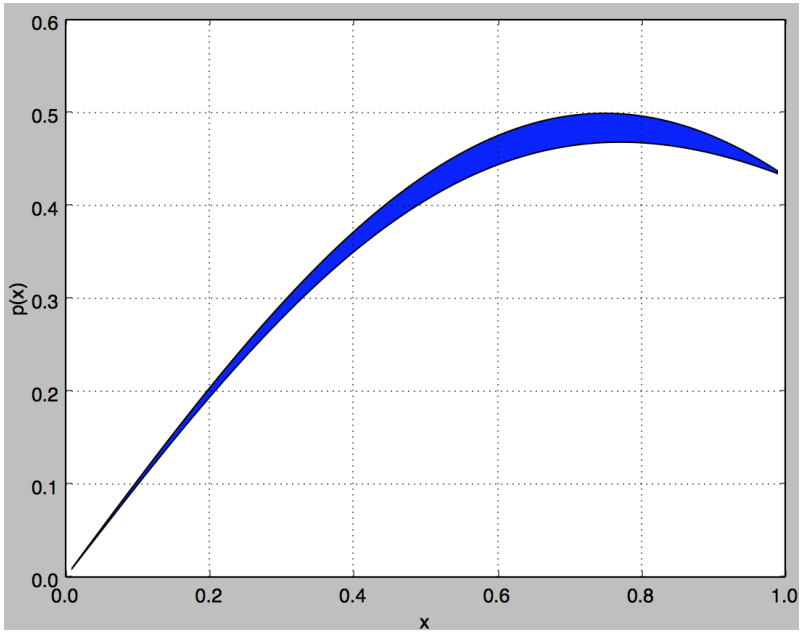
Sampling with Rejection

Clearly, we can bound $e^{-x^2}\sinh(x)$ with 0.5. To use rejection, we sample x uniformly on $[0, 1]$ and accept it with probability $e^{-x^2}\sinh(x)/2$. However, the shaded area represents the proportion of samples that we will reject, lowering the efficiency.



Sampling with Rejection

With a little thought, we can find a better bounding function that is easier to integrate and invert, $0.5\sin(3\pi x/2)$. That is, we choose x by sampling the sine function and accepting with probability $2e^{-x^2}\sinh(x)/\sin(3\pi x/2)$.



Outline

Sampling a distance to collision

Assume that at location s we have a density of photons given by $N(s)$. We then consider the incremental removal due to collisions along a flight path at a location $s + \delta s$. Using the definition of opacity, σ, this is

$$N(s + \delta s) = N(s) - \sigma \delta s N(s) + O(\delta_s^2)\,.$$

That is, to first order, the amount removed in a differential distance δs is the probability of interaction per distance, times the distance, times the original density. Rearranging and allowing $\delta s \to 0$,

$$\frac{dN}{ds}(s) = -\sigma N(s)\,.$$

This ordinary differential equation has the solution

$$N(s) = Ce^{-\sigma s}\,,$$

where C is an unspecified constant.

Sampling a distance to collision

We next normalize this solution on $[0, \infty)$ (the entire domain of a particle's potential flight without collisions) to construct the PDF for the probability that a particle will travel a distance s before experiencing a collision in ds about s,

$$N(s) = \sigma e^{-\sigma s} .$$

We next invert the CDF:

$$\int_0^{\xi} d\xi' = \int_0^{d_C} \sigma e^{-\sigma s} ds ,$$

$$\xi = -e^{-\sigma d_C} + 1 ,$$

$$d_C = \frac{-\ln(1-\xi)}{\sigma} ,$$

$$d_C = \frac{-\ln(\xi)}{\sigma} .$$

Final Thoughts

Some topics that we didn't cover:

- Probability mixing (for additive terms in a PDF, choose a term first)
- Stratified sampling and quasi-random numbers
- Importance Sampling (scoring with weights) by using $\int \frac{f(x)p(x)}{q(x)} q(x)\, dx$
- Metropolis-Hastings (an iterative, advanced rejection technique)

Thanks to Dr. Ronald Pevey for teaching me Monte Carlo. For more on particle transport Monte Carlo, see
`http://web.utk.edu/~rpevey/NE582/outline.htm`