Properties of samples from random distributions

The mean or average value or expected value of a set of samples $\{x_i\}$, $i = 1, \ldots, n$, is

$$\mu_n \equiv \frac{1}{n} \sum_{i=1}^{n} x_i$$

and the sample variance is

$$\sigma_n^2 \equiv \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2 .$$

If the mean μ is unknown, then one estimate is

$$\sigma_n^2 \equiv \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu_n)^2 .$$

Properties of distributions

A random distribution is characterized by a probability density function $f(x)$.

- The domain of the function is the set of possible values that could be obtained by taking random samples of the function. Call this domain Ω.
- The range of the function excludes $[-\infty, 0)$.
- The value
$$\int_{\alpha} f(x)\,\mathrm{d}x$$
is the probability that a sample from this distribution is in the set defined by α. (Replace the integral by a summation if the domain is discrete.)
- Because of these properties, we see that
$$\int_{\Omega} f(x)\,\mathrm{d}x = 1\,.$$
(Or the sum of the probabilities is 1 in the discrete domain case.)

The mean of a distribution and variance of the distribution is the average value you would expect to get for the sample mean and variance if you took a large number of very large sets of samples. They can be computed by

$$\mu = \int_\Omega x f(x)\,\mathrm{d}x$$
$$\sigma^2 = \int_\Omega (x-\mu)^2 f(x)\,\mathrm{d}x$$

The uniform distribution over the interval $[0, m]$ has

$$f(x) = \frac{1}{m}$$

Mean:

$$\mu = \int_0^m \frac{x}{m}\,\mathrm{d}x = \frac{m}{2}$$

Variance:

$$\sigma^2 = \int_0^m \frac{1}{m}\left(x - \frac{m}{2}\right)^2 \mathrm{d}x = \frac{m^2}{12}$$

The exponential distribution with parameter μ has

$$f(x) = \frac{1}{\mu} e^{-x/\mu}$$

for $x \in [0, \infty)$.

Mean: μ

Variance: μ^2

The normal distribution with parameters μ and σ: has

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$$

for $x \in (-\infty, \infty)$.

Mean: μ

Variance: σ^2

The world is normal.

The Central Limit Theorem: Let $f(x)$ be any distribution with mean μ and (finite) variance σ^2. Take a random sample of size n from $f(x)$, and call the mean of the sample $\bar{x}_n$. Define the random variable y_n by

$$y_n = \sqrt{n}\frac{\bar{x}_n - \mu}{\sigma}.$$

Then the distribution for y_n approaches the normal distribution with mean 0 and variance 1 as n increases.

Therefore, even if we know nothing about a distribution, except that its variance is finite, we can use samples from it To construct samples from a distribution that is near normal.

Generating random numbers on a computer

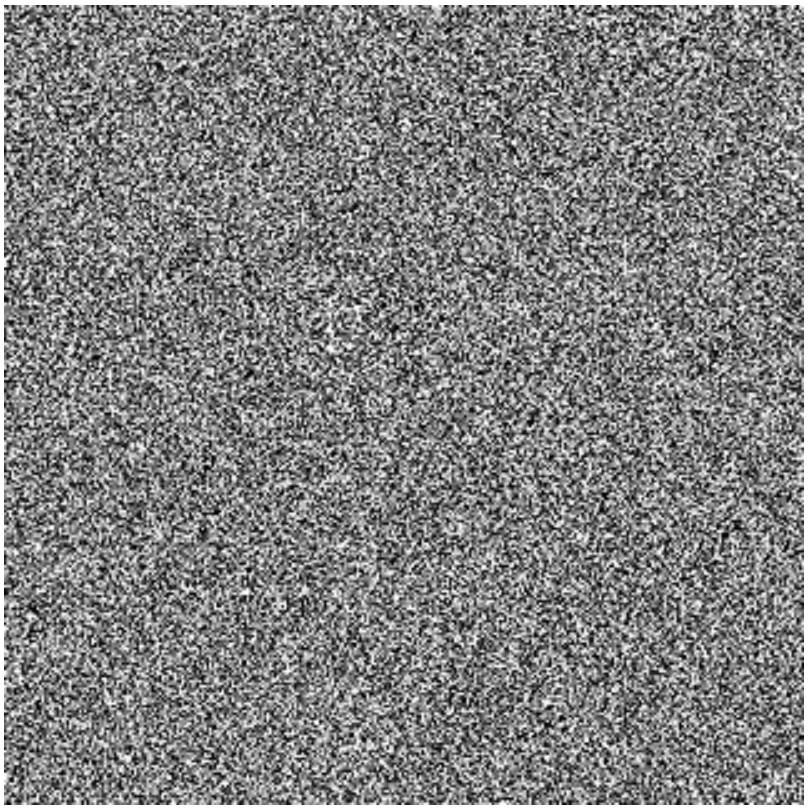
- Could use mechanical or natural processes to generate random samples
- E.g. for uniform distributions use a spinner
- Not good for simulation:
- The samples are irreproducible;
  - if someone else wanted to use the software, they would get a different result and not know whether it was because of their different sequence of random numbers or because of a bug.
- Tedious: Computers run through enormous quantities of random numbers, and it is just not feasible to generate them manually.
- Because of this, we use pseudorandom numbers in computer software.

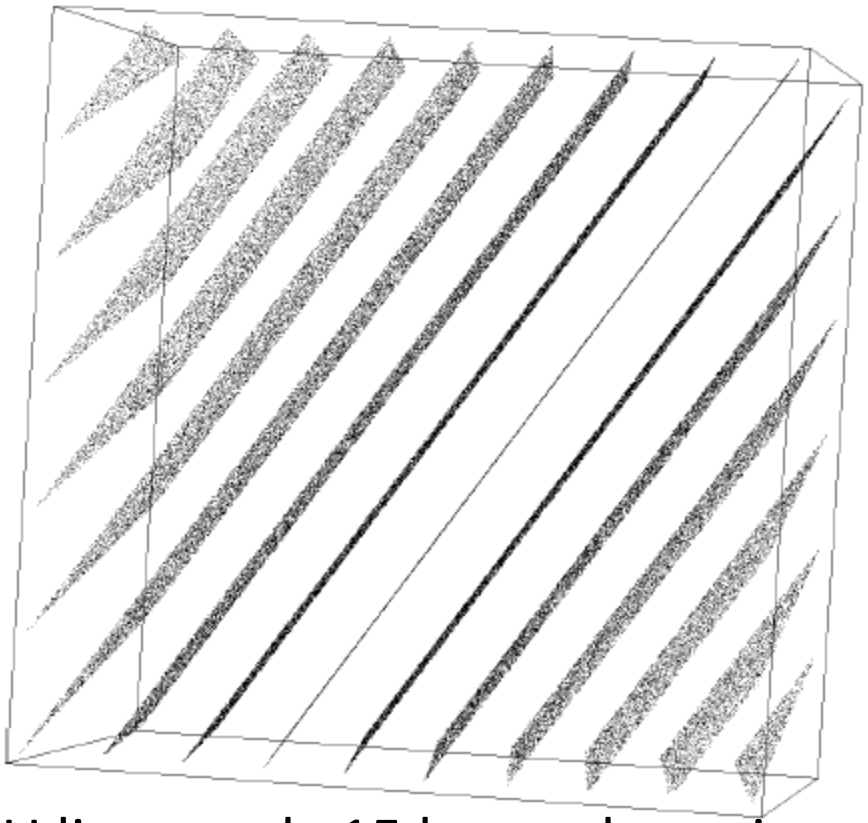
Linear Congruential generator

- Choose natural numbers a, M and N
- Chooseinitial value ("seed") $Z_0 \in \{1, \ldots .M - 1\}$.
- For i = 1, 2, . . . Set $Z_i = (aZ_{i-1} + c) \bmod M$, and $X_i = Z_i/M$.
- Z_i are in $\{0, 1, \ldots , M - 1\}$
- Thus X_i are in the interval [0, 1).
- sequence of pseudo-random numbers only depends on the seed Z_0.
- Running the pseudo-random number generator twice with the same seed thus generates exactly the same sequence of pseudo-random numbers.
- Let $a = 81$, $c = 35$, $M = 256$, and seed $Z_0 = 4$.
- $Z_1 = (81 \cdot 4 + 35) \bmod 256 = 359 \bmod 256 = 103$
- $Z_2 = (81 \cdot 103 + 35) \bmod 256 = 8378 \bmod 256 = 186$
- $Z_3 = (81 \cdot 186 + 35) \bmod 256 = 15101 \bmod 256 = 253$
- . .

RANDU

- very popular in the 1970s (IBM's System/360 & System/370, DEC PDP-11).
- used $a = 2^{16} + 3$, $c = 0$, and $M = 2^{31}$.





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

Random numbers

- Random to the bone
  - Their FFTs are random
  - Their projections are random
  - Their transformations should be random
  - …
  - They should be memoryless

PRN Generator

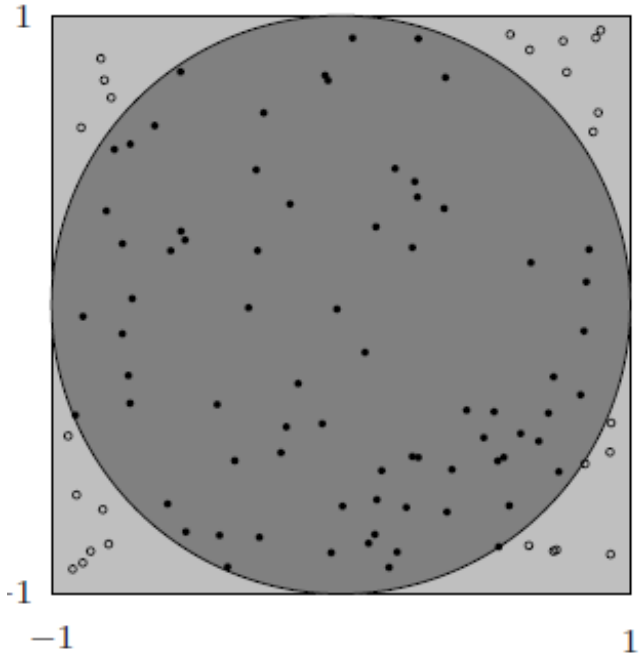
- Fairly cheap to generate uniformly distributed pseudorandom numbers.
- Pseudo-random number generators for the uniform distribution are easy to write
  - **very, very difficult to write a good one.**
- Use good random number generators
- Samples from other distributions are usually generated by taking two or more uniformly distributed pseudorandom numbers and manipulating them using functions such as sin, cos, and exp.
- Art of Computer Programming, vol 2 by Knuth
  - is quite enlightening, talking about the generation of pseudorandom numbers as well as how to test whether a program is generating pseudorandom numbers that are a good approximation to random.
- The standard functions in Matlab for generating pseudorandom normal and exponentially distributed sequences are okay …

Estimating π via simulation

- How I wish I could remember to places eight ..

  3 1 4 1 5 9 2 6 5

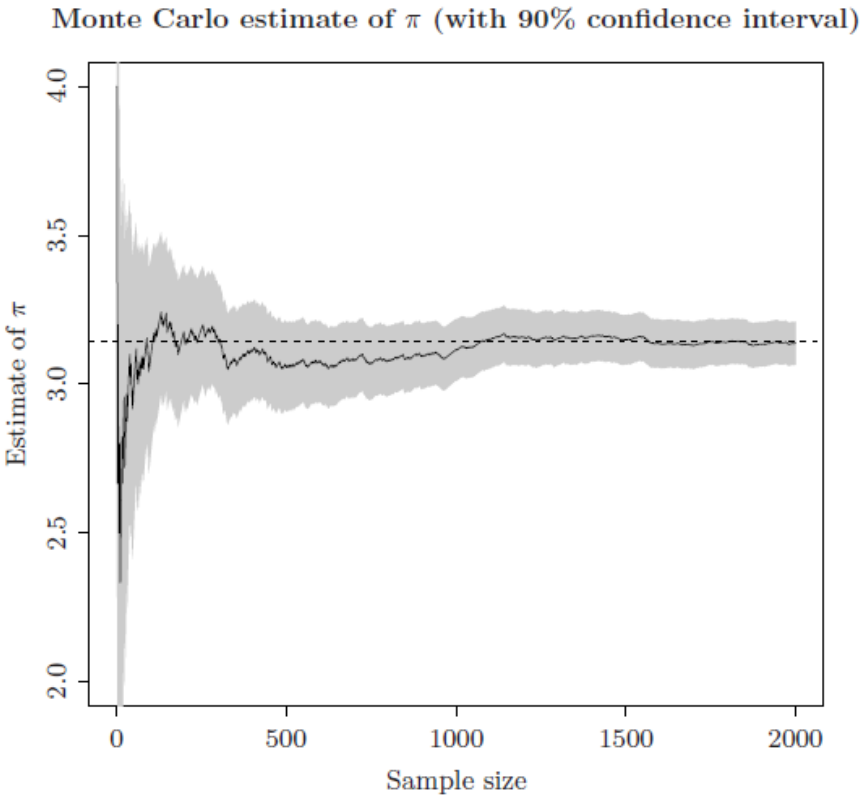
- Idea construct a square sheet with a circle drawn on it.
- Put it into a light rain
  - Into each area some rain must fall
- Number of drops in the square proportional to its area
- Also in circle
- Count drops, and take ratio
- Gives an estimate for π/4



$$\frac{\iint\limits_{\{x^2+y^2\leq 1\}} 1\,dxdy}{\iint\limits_{\{-1\leq x,y\leq 1\}} 1\,dxdy} = \frac{\pi}{2\cdot 2} = \frac{\pi}{4}$$

Questions

- How quickly does it converge?
  - How many samples
  - What is the error?
- Cannot talk of error
  - Expected value is the answer
  - Variance is an estimate of the error
- Knowing we are sampling from a uniform distribution, and that the distribution that the drop is inside or outside is binomial, we can get the variance
- Converges to zero as $n^{1/2}$



Generalize: Monte Carlo Integration

Suppose we are asked to estimate the value

$$I = \int_0^1 \dots \int_0^1 f(x_1, \dots, x_{10}) p(x_1, \dots, x_{10}) \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3 \mathrm{d}x_4 \mathrm{d}x_5 \mathrm{d}x_6 \mathrm{d}x_7 \mathrm{d}x_8 \mathrm{d}x_9 \mathrm{d}x_{10}$$

$$= \int_\Omega f(\mathbf{x}) p(\mathbf{x}) \, \mathrm{d}\mathbf{x}$$

Notation:

- $\mathbf{x} = [x_1, \dots, x_{10}]$.
- $\Omega = [0, 1] \times \dots \times [0, 1]$ is the region of integration, the unit hypercube in $\mathcal{R}^{10}$. It can actually be any region, but this will do fine as an example.
- Usually $p(\mathbf{x})$ is a constant, equal to 1 divided by the volume of Ω, but we'll use more general functions p later.

  We just need $p(\mathbf{x})$ to be a probability density function, so it should be nonnegative with

$$\int_\Omega p(\mathbf{x}) \, \mathrm{d}\mathbf{x} = 1 \, .$$

How might we approach the problem of computing I?

Interpolation

Fit a polynomial (or your favorite type of function) to $f(\mathbf{x})p(\mathbf{x})$ using sample values of the function, and then integrate the polynomial analytically.

For example, a polynomial of degree 2 in each variable would have terms of the form

$$x_1^{\square}x_2^{\square}x_3^{\square}x_4^{\square}x_5^{\square}x_6^{\square}x_7^{\square}x_8^{\square}x_9^{\square}x_{10}^{\square}$$

where the number in each box is 0, 1, or 2. So it has $3^{10} = 59,049$ coefficients, and we would need 59,049 function values to determine these.

But recall from 460 that usually you need to divide the region into small boxes so that a polynomial is a good approximation within each box.

If we divide the interval $[0, 1]$ into 5 pieces, we make 5^{10} boxes, with 59,049 function evaluations in each!

Clearly, this method is expensive!

Option 2: product rules

functions $f(\mathbf{x})p(\mathbf{x})$ can be well approximated by a separable function

$$f(\mathbf{x})p(\mathbf{x}) \approx f_1(x_1)f_2(x_2)\dots f_{10}(x_{10})\,.$$

at case we can approximate our integral by

$$I \approx \int_0^1 f_1(x_1)\,\mathrm{d}x_1 \dots \int_0^1 f_{10}(x_{10})\,\mathrm{d}x_{10}$$

s works, it is great, but we aren't often that lucky.

Option 3: Use your favorite 1-d method

If we have a function quad that integrates functions of a single variable then we can use quad to compute

$$\int_0^1 g(x_1)\,\mathrm{d}x_1$$

where

$$g(z) = \int_0^1 \cdots \int_0^1 f(z, x_2, \ldots, x_{10}) p(z, x_2, \ldots, x_{10})\,\mathrm{d}x_2 \ldots \mathrm{d}x_{10}$$

as long as we can evaluate $g(z)$!

But $g(z)$ is just an integration, so we can evaluate it using quad, too!

We end up with 10 nested calls to quad. Again, this is very expensive!

See Pointer 18.1.

Monte Carlo integration

Idea:

- Generate n points $\{\mathbf{z}^{(i)}\}$ that are randomly distributed with probability density function p
  For our example integration problem, if $p(\mathbf{x})$ is constant, this requires generating $10n$ random numbers, uniformly distributed in $[0, 1]$.
- Then

$$\mu_n = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{z}^{(i)})$$

  is an approximation to the mean value of f in the region, and therefore the value of the integral is

$$I \approx \mu_n \int_\Omega p(\mathbf{x}) \, \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3 \mathrm{d}x_4 \mathrm{d}x_5 \mathrm{d}x_6 \mathrm{d}x_7 \mathrm{d}x_8 \mathrm{d}x_9 \mathrm{d}x_{10} = \mu_n.$$

Error estimate

- The expected value of this estimate is the true value of the integral; very nice!
- In fact, for large n, the estimates have a distribution of $\sigma/\sqrt{n}$ times a normal distribution (with mean 0, variance 1), where

$$\sigma^2 = \int_\Omega (f(\mathbf{x}) - I)^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x}\,,$$

where Ω is the domain of the integral we are estimating and

$$\int_\Omega f(\mathbf{x}) p(\mathbf{x}) \mathrm{d}\mathbf{x} = I\,.$$

Note that the variance is a constant independent of the dimension d of the integration!

Jargon: Methods whose expectation converges to the answer we are looking for are called **maximum likelihood**

Variance-reduction methods

Suppose that we want to estimate

$$I = \int_{\Omega} f(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

where Ω is a region in $\mathcal{R}^{10}$ with volume equal to one.

Method 1: Our Monte Carlo estimate of this integral involves taking uniformly distributed samples from Ω and taking the average value of $f(\mathbf{x})$ at these samples.

Method 2: Let's choose a function $p(\mathbf{x})$ satisfying $p(\mathbf{x}) > 0$ for all $\mathbf{x} \in \Omega$, normalized so that

$$\int_{\Omega} p(\mathbf{x}) = 1.$$

Then

$$I = \int_{\Omega} \frac{f(\mathbf{x})}{p(\mathbf{x})} p(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

We can get a Monte Carlo estimate of this integral by taking samples from the distribution with probability density $p(\mathbf{x})$ and taking the average value of $f(\mathbf{x})/p(\mathbf{x})$ at these samples.

When will Method 2 be better than Method 1?

Recall that the variance of our estimate is proportional to

$$\sigma^2 = \int_\Omega \left(\frac{f(\mathbf{x})}{p(\mathbf{x})} - I \right)^2 p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \, .$$

so if we chose p so that $f(\mathbf{x})/p(\mathbf{x})$ is close to constant, then σ is close to zero!

Note that this requires that $f(\mathbf{x})$ should be close to having a constant sign.

Intuitively, why does importance sampling work?

- In regions where $f(\mathbf{x})$ is big, $p(\mathbf{x})$ will also be big, so there is a high probability that we will sample from these regions.
- In regions where $f(\mathbf{x})$ is small, the $p(\mathbf{x})$ will also be small, so we won't waste time sampling from regions that don't contribute much to the integral.

Intuitively, why does importance sampling work?

- In regions where $f(\mathbf{x})$ is big, $p(\mathbf{x})$ will also be big, so there is a high probability that we will sample from these regions.
- In regions where $f(\mathbf{x})$ is small, the $p(\mathbf{x})$ will also be small, so we won't waste time sampling from regions that don't contribute much to the integral.