Overview of lecture slides 02

Generating random numbers with desired distributions

General methods

- Inverse transform sampling
- Rejection sampling
- Markov chain sampling (Metropolis/Hastings algorithm)

Various specialized algorithms for specific distributions

Inverse CDF sampling

x is uniformly distributed.

What transformation $y = f(x)$ will provide variable y with distribution $p(y)$?

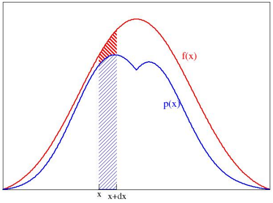
$$\mathcal{C}(y) = \int_{-\infty}^{y} p(z)dz \qquad\qquad y = f(x) = \mathcal{C}^{-1}(x)$$

Rejection sampling

We want to generate random numbers distributed according to $p(x)$, given a prng distributed as $f_0(x)$.

Rescale $f_0(x)$: $f(x) = Af_0(x)$, so that $f(x) > p(x)$ everywhere.

Select points under $f(x)$ curve, accept with probability $p(x)/f(x)$



Implementation

1. Pick number X according to distribution $\frac{1}{A}f(x)$, where $A = \int_{-\infty}^{\infty} f(x)dx$
2. Accept X as your random number with probability $p(X)/f(X)$. i.e., reject X with probability $1 - p(X)/f(X)$.

Markov chain sampling

Build a sequence of numbers which (eventually) have the desired distribution $p(x)$

A Markov chain $\longrightarrow$
series of stochastic values (numbers, states, etc). Each element is determined (stochastically) by previous element alone.

Markov chain algorithms $\longrightarrow$
generate series of numbers/states with desired distribution.

Markov chain sampling

Build a sequence of numbers which (eventually) have the desired distribution $p(x)$

Metropolis-Hastings algorithm

1. Pick an initial value x_0
2. For $i = 0, \ldots,$ until satisfied:
   - A. Given x_i, generate a random candidate update value x' according to probability distribution $g(x'|x_i)$.
   - B. Calculate the acceptance probability

     $$A = \min\left(1,\ \frac{p(x')g(x'|x_i)}{p(x_i)g(x_i|x')}\right)$$

   - C. Accept or reject x' as the next value in the Markov chain, according to probability A:
     If accepting: $x_{i+1} = x'$; If rejecting: $x_{i+1} = x_i$.

Markov chain sampling

- Easiest choice for update proposal:
  choose x' uniform-randomly, independent of x_i.
  - Doesn't work if support is unbounded.
  - Of course, accept/reject probability will depend on x_i)
- Better choice: Choose x' from a region close to x_i,
  e.g., uniformly from interval $(x_i - \delta, x_i + \delta)$.
- For both these choices:
  $g(x'|x_i) = g(x_i|x')$, cancellation $\quad \longrightarrow \quad A = \min\left(1,\ \frac{p(x')}{p(x_i)}\right)$.
- Method can be easily generalized to multi-dimensional distributions.
  (We will use later for statistical mechanics.)
- Elements near each other along a Markov chain are correlated.
  $\longrightarrow$ Use every m-th element.

Gaussian random numbers

Gaussian-distributed random numbers — very useful, widely implemented.

We want a gaussian distribution

$$P(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}$$

Inverse transform sampling not suitable:

- we have no closed expression for $\int P(y)dy$
- we certainly have no closed expression for the inverse!

[scipy (in scipy.special) has functions `erf` and `erfinv` — numerical approximations]

Gaussian random numbers

Many algorithms available:

- summing a large enough number of uniform variates
  (central limit theorem)
- Box-Muller transformation
- Ziggurat method
- .....

Thomas, Luk, Leong, Villasenor, *Gaussian Random Number Generators*, ACM Computing Surveys, Vol. 39, No. 4, Article 11 (2007).

Gaussian random numbers — blackbox

Both rand and numpy.rand have inbuilt gaussian prng's

So do most other scientific computing systems —
gaussian prng's are considered indispensable and widely used

Gaussian random numbers – Box-Muller

Consider two gaussian distributions,

$$P(y_1, y_2) = \frac{1}{2\pi} e^{-(y_1^2+y_2^2)/2} = P(y_1)P(y_2)$$

Transformation of two or more probability distributions:

$$P_Y(y_1, y_2, \ldots) = P_X(x_1, x_2, \ldots) \left| \frac{\partial(x_1, x_2, \ldots)}{\partial(y_1, y_2, \ldots)} \right|$$

Gaussian random numbers – Box-Muller

Box-Muller algorithm

If we take X_1, X_2 uniform on $\langle 0, 1\rangle$ and

$$Y_1 = \sqrt{-2\ln X_1}\cos(2\pi X_2) \qquad Y_2 = \sqrt{-2\ln X_1}\sin(2\pi X_2)$$

then Y_1, Y_2 are gaussian

Note

Avoid calls to $\ln, \sin, \cos, \ldots$ when possible — they are slow

Monte Carlo methods

Monte Carlo is a town in Monaco famous for its casinos

Monte Carlo methods are numerical methods based on random numbers

Various types

- Direct Monte Carlo
  - model complicated or unknown processes by random numbers
  - Stochastic dynamics, eg Brownian motion or traffic modelling
- Monte Carlo integration
  - calculate integrals using random numbers
  - especially useful in many dimensions
- Markov chain Monte Carlo
  - generate statistical distributions using ‘random walks’
  - widely used in many-particle physics, both classical and quantum

Monte Carlo integration – 1D version

Want to integrate function $f(x)$ on interval $[a, b]$:

$$I = \int_a^b f(x)dx$$

Standard numerical integration:

- Take N evenly spaced points x_i in $[a, b]$
- evaluate $f(x)$ at those points

$$I \quad = \quad (b-a) \times \langle f \rangle \quad \approx \quad \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)$$

This is the rectangular integration or the the midpoint rule

Monte Carlo integration – 1D version

Standard numerical integration for $I = \int_a^b f(x)dx$:

- Mid-point rule with uniformly spaced x_i:

$$I \quad = \quad (b-a) \times \langle f \rangle \quad \approx \quad \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)$$

- Vary weights: trapezoidal, Simpson's,
  Simpson's 3/8, Boole's,..... rules
  (Newton-Cotes quadrature formulae)
- Gaussian quadrature: use non-uniformly spaced points
  chosen very specifically
- Other improvements:
  - Vary N and extrapolate to $N \to \infty$
  - Adaptively vary N and interval widths

Monte Carlo integration – 1D version

Mid-point rule with uniformly spaced x_i:

$$I = (b-a) \times \langle f \rangle \approx \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)$$

Instead of more elegant methods, let's consider a more crude version:

Monte Carlo integration

Pick the points x_i randomly!

Monte Carlo integration – 1D version

Sample randomly

$$I \quad = \quad (b-a) \times \langle f \rangle \quad \approx \quad \frac{b-a}{N} \sum_{i=1}^{N} f(\xi_i)$$

Where the ξ_i are uniformly sampled from $[a, b]$; i.e., from distribution

$$p_\xi(x) = \begin{cases} 1/(b-a) & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

Monte Carlo integration – multi-dimensional

Sample randomly for 1D integration

$$I \quad = \quad (b-a)\times\langle f\rangle \quad \approx \quad \frac{b-a}{N}\sum_{i=1}^{N} f(\xi_i) \quad = \quad \frac{b-a}{N}\sum_{i=1}^{N} f_i$$

where ξ_i are uniformly sampled from $[a,b]$.

Multi-variable (multi-dimensional) integration

Works in arbitrary numbers of dimensions:

$$\int_\Omega f(\vec{r})dV = V\langle f\rangle \quad \approx \quad \frac{V}{N}\sum_{i=1}^{N} f(\vec{\xi_i})$$

$\vec{r} = (x_1, x_2, ...x_N)$. $\quad V \longrightarrow$ hyper-volume of integration region Ω.

The $\vec{\xi_i}$ are uniformly sampled from Ω.

Monte Carlo integration - error

$$\langle f \rangle \approx \frac{1}{N}\sum_{i=1}^{N} f(\xi_i) = \frac{1}{N}\sum_{i=1}^{N} f_i \qquad \begin{cases} \text{estimating } \langle f \rangle \\ \text{using sample } \{f_i\} \end{cases}$$

Standard error of mean $\approx \dfrac{1}{\sqrt{N}} \times$ std.dev. of distribution/sample

Monte Carlo integration

$$I \approx \frac{V}{N}\sum_{i=1}^{N} f_i \pm \frac{V}{\sqrt{N}}\sqrt{\frac{1}{N}\sum_{i=1}^{N} f_i^2 - \left[\frac{1}{N}\sum_{i=1}^{N} f_i\right]^2}$$

where

$f_i \rightarrow$ values of integrand at N uniform-randomly chosen points in integration region

$V \rightarrow$ hyper-volume of integration region

Monte Carlo integration - error

Monte Carlo integration

$$I \approx \frac{V}{N}\sum_{i=1}^{N} f_i \pm \frac{V}{\sqrt{N}}\sqrt{\frac{1}{N}\sum_{i=1}^{N} f_i^2 - \left[\frac{1}{N}\sum_{i=1}^{N} f_i\right]^2}$$

Approximation: used $1/\sqrt{N}$ instead of $1/\sqrt{(N-1)}$ above

MC integration only makes sense for large N!

Monte Carlo integration - error derivation

The variance of a stochastic variable X is

$$\sigma_X^2 \equiv \operatorname{var} X \equiv \langle (X-\bar{X})^2 \rangle = \langle X^2 \rangle - \langle X \rangle^2$$

For our MC integral we get

$$\begin{aligned}
\sigma^2 &= \left\langle \left(\frac{b-a}{N} \sum_i f_i \right)^2 \right\rangle - \left\langle \frac{b-a}{N} \sum_i f_i \right\rangle^2 \\
&= \frac{(b-a)^2}{N} \left[\frac{1}{N} \left\langle \sum_i f_i^2 \right\rangle + \frac{1}{N} \left\langle \sum_{i \neq j} f_i f_j \right\rangle - N \langle f \rangle^2 \right] \\
&= \frac{(b-a)^2}{N} \left(\langle f^2 \rangle + \frac{1}{N} \sum_{i \neq j} \langle f_i \rangle \langle f_j \rangle - N \bar{f}^2 \right) \\
&= \frac{(b-a)^2}{N} \left(\langle f^2 \rangle + \frac{N(N-1)}{N} \langle f \rangle^2 - N \langle f \rangle^2 \right) = \frac{(b-a)^2}{N} \left(\langle f^2 \rangle - \langle f \rangle^2 \right)
\end{aligned}$$

In going from the second to the third line we have used that f_i and f_j are independent and uncorrelated.

Monte Carlo integration – (dis)advantages

MC integration is very inefficient for one dimensional integrals.

Advantages

- works in arbitrary numbers of dimensions
- beats ordinary methods for very high numbers of dimensions
- works for complicated boundaries

Complicated boundary example: computing π

π is the area of the unit circle $x^2 + y^2 < 1$
$\frac{\pi}{4}$ is the area of the quarter-circle $x^2 + y^2 < 1; x, y \in \langle 0, 1 \rangle$

This can be written as a 2-dimensional integral:

$$\frac{\pi}{4} = \int_0^1 \int_0^1 \Theta\Big(1 - (x^2 + y^2)\Big) dxdy \approx \frac{1}{N} \sum_i \Theta\Big(1 - (x_i^2 + y_i^2)\Big)$$

Procedure

1. Generate N pairs of random numbers (x_i, y_i)
2. Add 1 each time $x_i^2 + y_i^2 < 1$
3. Divide by N to get the average
4. Multiply by 4, and you have π!

Complicated boundary example continued

The quarter-circle has areal density $\rho(x, y)$. Calculate it's mass.
π is the area of the unit circle $x^2 + y^2 < 1$
$\frac{\pi}{4}$ is the area of the quadrant $x^2 + y^2 < 1; x, y \in \langle 0, 1 \rangle$

This can be written as a 2-dimensional integral:

$$M = \int_0^1 \int_0^1 \Theta\Big(1-(x^2+y^2)\Big)\rho(x, y)dxdy \approx \frac{1}{N}\sum_i \Theta\Big(1-(x_i^2+y_i^2)\Big)\rho(x_i, y_i)$$

Procedure

1. Generate N pairs of random numbers (x_i, y_i)
2. Add $\rho(x_i, y_i)$ if $x_i^2 + y_i^2 < 1$
3. Divide the sum by N

Advantages in many dimensions

Monte Carlo error:

$$\sigma^2 = \frac{V^2}{N}\left(\langle f^2\rangle - \langle f\rangle^2\right) \qquad \text{— the error decreases as } \frac{1}{\sqrt{N}}$$

Standard numerical integration gives errors $\propto$ powers of grid spacing δ.

One dimension: $\quad \text{err} \sim \delta^2 \sim \frac{1}{N^2}\,, \quad \delta^3 \sim \frac{1}{N^3}\,, \ldots \quad \begin{cases}\text{Trapezoidal,}\\ \text{Simpson's,...}\end{cases}$

Error decreases much faster than Monte Carlo:

Monte Carlo:	$N \to 2N$	$\Longrightarrow$	$\text{err} \to \text{err}/\sqrt{2}$
Trapezium:	$N \to 2N$	$\Longrightarrow$	$\text{err} \to \text{err}/4$
Simpson's:	$N \to 2N$	$\Longrightarrow$	$\text{err} \to \text{err}/8$

Monte Carlo integration is really inefficient for one-variable integration!

Advantages in many dimensions

Monte Carlo integration is really inefficient for one-variable integration!
But:

In d dimensions $N \sim (\frac{L}{\delta})^d \implies \text{err} \sim N^{-k/d}$

Not so fast for large d

The Monte Carlo error is still $\sim N^{-1/2}$, independent of d

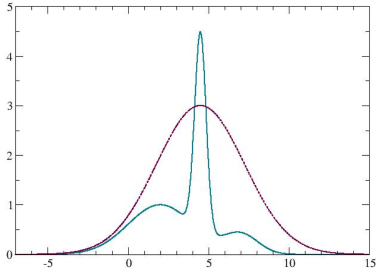
For large enough dimensions, Monte Carlo wins.

E.g., for trapezium ($k = 2$) Monte Carlo is better once $d > 4$

Importance sampling

If integrand is sharply peaked in some region, and small in others:

- Need high accuracy where f is big and varying
- Dont waste our time where it is close to zero



Sample more points where f is greater. How?

Choose x with probability $q(x)$ so that $f(x)/q(x) \approx$ constant.

$$I = \int f(x)dx = \int q(x)\left[\frac{f(x)}{q(x)}\right]dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{q(x_i)}$$

x_i's sampled from probability distribution $q(x)$.

Summary

Random number distributions

- Inverse transform sampling
- Rejection sampling
- Markov chain sampling

Monte Carlo integration

- Integrate functions by randomly sampling points
- Errors decrease as $1/\sqrt{N}$
- Superior for high-dimensional integrals
- Sample uniformly or with a weight function (importance sampling)