Monte Carlo Methods

Stéphane Paltani

ISDC Data Center for Astrophysics
Astronomical Observatory of the University of Geneva

Statistics Course for Astrophysicists, 2010–2011

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Numerical integration

Optimization

Caveat

The presentation aims at being user-focused and at presenting usable recipes

Do not expect a fully mathematically rigorous description!

This has been prepared in the hope to be useful even in the stand-alone version

Please provide me feedback with any misconception or error you may find, and with any topic not addressed here but which should be included

Outline

What are Monte-Carlo methods?
- General concepts
- Applications
- Simple examples

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Numerical integration

Optimization

General concepts

Monte-Carlo methods:

- have been invented in the context of the development of the atomic bomb in the 1940's
- are a class of computational algorithms
- can be applied to vast ranges of problems
- are not a statistical tool
- rely on repeated random sampling
- provide generally approximate solutions
- are used in cases where analytical or numerical solutions don't exist or are too difficult to implement
- can be used by the Lazy Scientist^{TM} even when an analytical or numerical solution can be implemented

Overview of the method

Monte-Carlo methods generally follow the following steps:

1. Determine the statistical properties of possible inputs
2. Generate many sets of possible inputs which follows the above properties
3. Perform a deterministic calculation with these sets
4. Analyze statistically the results

The error on the results typically decreases as $1/\sqrt{N}$

Outline

What are Monte-Carlo methods?

- General concepts
- Applications
- Simple examples

Generation of random variables

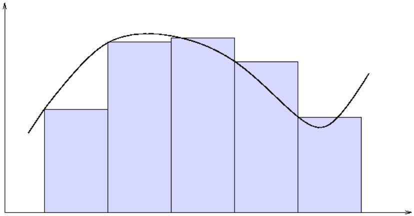
Markov chains Monte-Carlo

Error estimation

Numerical integration

Optimization

Numerical integration



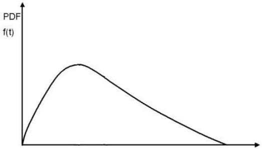
Most problems can be solved by integration

Monte-Carlo integration is the most common application of Monte-Carlo methods

Basic idea: Do not use a fixed grid, but random points, because:

1. Curse of dimensionality: a fixed grid in D dimensions requires N^D points
2. The step size must be chosen first

Error estimation

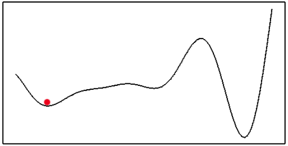


Given any arbitrary probability distribution and provided one is able to sample properly the distribution with a random variable (i.e., $x \sim f(x)$), Monte-Carlo simulations can be used to:

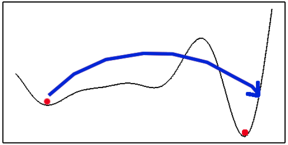
- determine the distribution properties (mean, variance,. . .)
- determine confidence intervals, i.e. $P(x > \alpha) = \int_{\alpha}^{\infty} f(x)\mathrm{d}x$
- determine composition of distributions, i.e. given $P(x)$, find $P(h(x))$, $h(x) = x^2; cos(x) - sin(x); \dots$

Note that these are all integrals!

Optimization problems

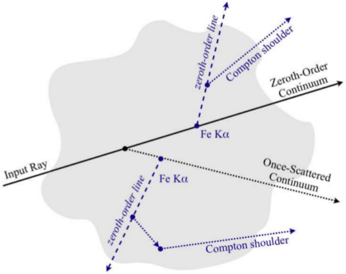


Numerical solutions to optimization problems incur the risk of getting stuck in local minima.



Monte-Carlo approach can alleviate the problem by permitting random exit from the local minimum and find another, hopefully better minimum

Numerical simulations

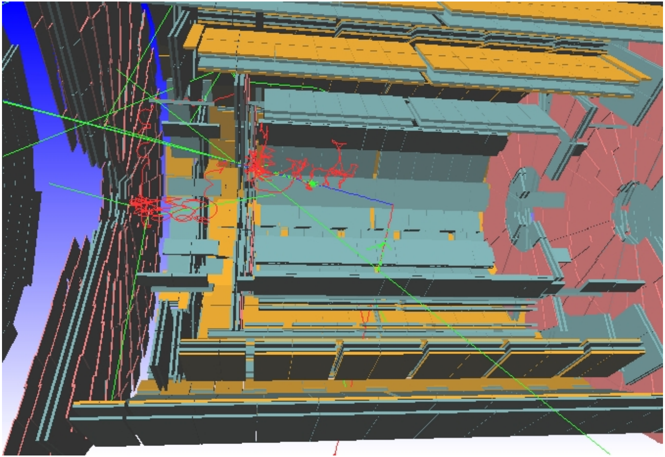


- Radiation transfer is Google-wise the main astrophysical application of Monte-Carlo simulations in astrophysics
- In particle physics and high-energy astrophysics, many more physical processes can be simulated

Some physical processes are discretized and random by nature, so Monte-Carlo is particularly adapted

Numerical simulations

GEANT4



GEANT4 is also used to determine the performance of X-ray and gamma-ray detectors for astrophysics

Outline

What are Monte-Carlo methods?

- General concepts
- Applications
- Simple examples

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Numerical integration

Optimization

Probability estimation

Head vs tail probability

What is the probability to obtain either 3, 6 or 9 heads if one draws a coin ten times?

- Binomial probability:
  $P = B(3; 10, 0.5) + B(6; 10, 0.5) + B(9; 10, 0.5) \simeq 0.33$
- Monte-Carlo simulation:
  1. Given a random variable $y \sim \mathcal{U}(0,1)$, define “head” if $y < 0.5$, “tail” otherwise
  2. Draw 10 random variables $x_i \sim \mathcal{U}(0,1),\ i = 1, \ldots, 10$
  3. Count the number of heads H, and increment T if $H = 3, 6$, or 9
  4. Repeat 2.–3. N times, with N reasonably large
  5. The probability is approximately T/N
- Note that this is an integration on a probability distribution, even if it is discrete!

Error estimation

What is the uncertainty on the mean?

Assuming N random variables $x_i \sim \mathcal{N}(0,\sigma),\ i=1,\dots,N$, the estimator of the mean is: $\bar{x} = N^{-1}\sum_{i=1}^{N} x_i$ and its uncertainty is:

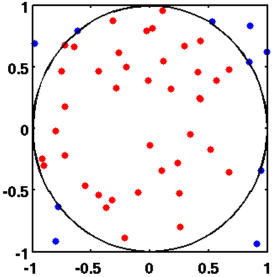
$$\sigma_{\bar{x}} = \sigma/\sqrt{N}$$

The Monte-Carlo way:

1. Draw a set of N random variables $y_i \sim \mathcal{N}(0,\sigma),\ i=1,\dots,N$
2. Calculate the sample mean $\bar{y} = N^{-1}\sum_{i=1}^{N} y_i$
3. Redo 1.–2. M times
4. The uncertainty on the mean $\sigma_{\bar{x}}$ is the root mean square of $\bar{y}_j,\ j=1,\dots,M$, i.e. $\sigma_{\bar{x}}^2 = (M-1)^{-1}\sum_{j=1}^{M}\left(\bar{y}_j - \hat{y}\right)^2$, with $\hat{y} = M^{-1}\sum_{j=1}^{M}\bar{y}_j$

Numerical integration

How to calculate π?



1. Draw N point (x, y) uniformly at random in a square
2. Count the C points for which $x^2 + y^2 < 1$
3. The ratio C/N converges towards $\pi/4$ as $N^{1/2}$

Outline

What are Monte-Carlo methods?

Generation of random variables

- Random-number generators
- Transformation method
- Rejection method
- Quasi-random sequences

Markov chains Monte-Carlo

Error estimation

Numerical integration

Optimization

Random number generators

Basic principles

- We want to draw many random variables $N_i \sim \mathcal{U}(0,1)$, $i = 1, \ldots$ which satisfy (or approximate sufficiently well) all randomness properties
- $N_i \sim \mathcal{U}(0,1)$, $\forall i$ is not sufficient. We also want that $f(N_i, N_j, \ldots)$ $\forall i, j, \ldots$ has also the right properties
- Correlations in k-space are often found with a bad random-number generators
- Another issue is the period of the generator
- The `ran()` function in `libc` has been (very) bad. Do not use this function in applications when good randomness is needed says `man 3 rand`

Random number generators

Basic algorithm

- Many random number generators are based on the recurrence relation:

  $$N_{j+1} = a \cdot N_j + c \pmod{m}$$

  These are called linear congruential generators. c is actually useless.
- “Divide” by $m + 1$ to get a number in the range $[0; 1[$
- Choices of a, m in standard libraries are found to range from very bad to relatively good
- A “minimal standard” set is $a = 7^5 = 16807$, $c = 0$, $m = 2^{31} - 1 = 2147483647$. This is RAN0
- Note that the period is at most m

Random number generators

Improvements on RAN0

1. Multiplication by a doesn't span the whole range of values, i.e. if $N_i = 10^{-6}$, $N_{i+1} \leq 0.016$, failing a simple statistical test
   - Swap consecutive output values: Generate a few values (~ 32), and at each new call pick one at random. This is RAN1
2. The period $m = 2^{31} - 1$ might be too short
   - Add the outcome of two RAN1 generators with (slightly) different m's (and a's). The period is the least common multiple of m_1 and $m_2 \sim 2 \cdot 10^{18}$. This is RAN2
3. The generator is too slow
   - Use in *C* inline $N_{i+1} = 1664525 \cdot N_i + 1013904223$ using `unsigned long`. Patch the bits into a real number (machine dependent). This is RANQD2

Random number generators

Implementations and recommendations

NR: Numerical Recipes
GSL: GNU Scientific Library

Library	Generator	Relative speed	Period
NR	RAN0	1.0	$\sim 2^{31}$
NR	RAN1	1.3	$\sim 2^{36}$
NR	RAN2	2.0	$\sim 2^{62}$
NR	RANQD2	0.25	$\sim 2^{30}$
GSL	MT19937	0.8	$\sim 2^{19937}$
GSL	TAUS	0.6	$\sim 2^{88}$
GSL	RANLXD2	8.0	$\sim 2^{400}$

Always use GSL! See the GSL doc for the many more algorithms available

Outline

What are Monte-Carlo methods?

Generation of random variables

- Random-number generators
- Transformation method
- Rejection method
- Quasi-random sequences

Markov chains Monte-Carlo

Error estimation

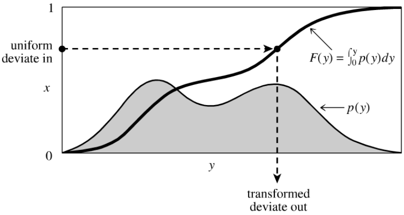
Numerical integration

Optimization

Transformation method

The method

The transformation method allows in principle to draw values at random from any distribution



1. Given a distribution $p(y)$, the cumulative distribution function (CDF) of $p(y)$ is $F(y) = \int_0^y p(w)\,\mathrm{d}w$
2. We want to draw y uniformly in the shaded area, i.e. uniformly over $F(y)$; by construction $0 \leq F(y) \leq 1$,
3. We draw $x \sim \mathcal{U}(0,1)$ and find y so that $x = F(y)$
4. Therefore $y(x) = F^{-1}(x),\ x \sim \mathcal{U}(0,1)$

Transformation method

Example

Exponential deviates: $p(y) = \lambda e^{-\lambda y}$

$$F(y) = 1 - e^{-\lambda y} = x$$

$$y(x) = -\frac{1}{\lambda}\ln(1 - x)$$

Note: this is equivalent to

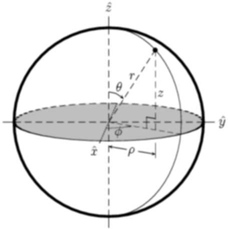
$$y(x) = -\frac{1}{\lambda}\ln(x),$$

since, if $x \sim \mathcal{U}(0, 1)$, then $1 - x \sim \mathcal{U}(0, 1)$ as well

Note also that it is rather uncommon to be able to calculate $F^{-1}(x)$ analytically. Depending on accuracy, it is possible to calculate an numerical approximation

Transformation method

A point in space



To draw a point in a homogeneous sphere of radius R:

1. ϕ can be drawn uniformly from $\mathcal{U}(0, 2\pi)$
2. θ has a sine distribution $p(\theta) = sin(\theta)/2,\ \theta \in [0; \pi[$ Transformation: $\theta = 2\arccos(x)$
3. Each radius shell has a volume $f(R) \sim R^2\,\mathrm{d}R$, so $R \propto \sqrt[3]{x}$
4. Alternatively, draw a point at random on the surface of a sphere $(x, y, z)/\sqrt{x^2 + y^2 + z^2}$ with $x,\ y,\ z \sim \mathcal{N}(0, 1)$

Outline

What are Monte-Carlo methods?

Generation of random variables

- Random-number generators
- Transformation method
- Rejection method
- Quasi-random sequences

Markov chains Monte-Carlo

Error estimation

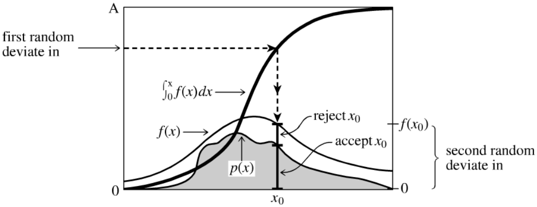
Numerical integration

Optimization

Rejection method

The method

If the CDF of $p(x)$ is difficult to estimate (and you can forget about inversion), the rejection method can be used

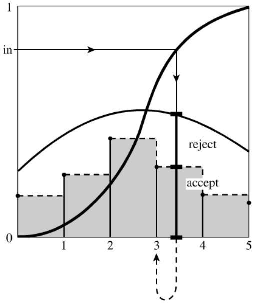


1. Find a comparison function $f(x)$ that can be sampled, so that $f(x) \geq p(x)$, $\forall x$
2. Draw a random deviate x_0 from $f(x)$
3. Draw a uniform random deviate y_0 from $\mathcal{U}(0, f(x_0))$
4. If $y_0 < p(x_0)$, accept x_0, otherwise discard it
5. Repeat 2.–4. until you have enough values

The rejection method can be very inefficient if $f(x)$ is very different from $p(x)$

Rejection method

Example



The Poisson distribution is discrete: $\mathcal{P}(n;\alpha) = \frac{\alpha^n\ e^{-\alpha}}{n!}$

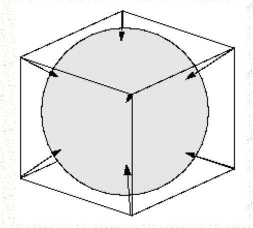
Make it continuous:

$$\mathcal{P}(x;\alpha) = \frac{\alpha^{[x]}\ e^{-\alpha}}{[x]!}$$

A Lorentzian $f(x) \propto \frac{1}{(x-\alpha)^2+c^2}$ is a good comparison function

Rejection method

A point in space



A simpler way to draw a point in a homogeneous sphere of radius R based on rejection:

1. Draw three random variables $x,\ y,\ z$ from $\mathcal{U}(-R, R)$
2. Keep if $x^2 + y^2 + z^2 < R^2$, reject otherwise
3. Repeat 1.–2. until you have enough values

Efficiency is $\frac{4\pi}{3}/2^3 \simeq 0.52$

Distributions

GNU Scientific Library implements (not exhaustive!):

Gaussian
Correlated bivariate Gaussian
Exponential
Laplace
Cauchy
Rayleigh
Landau
Log-normal
Gamma, beta
χ^2, F, t

Binomial
Poisson
Spherical 2D, 3D

Outline

What are Monte-Carlo methods?

Generation of random variables

- Random-number generators
- Transformation method
- Rejection method
- Quasi-random sequences

Markov chains Monte-Carlo

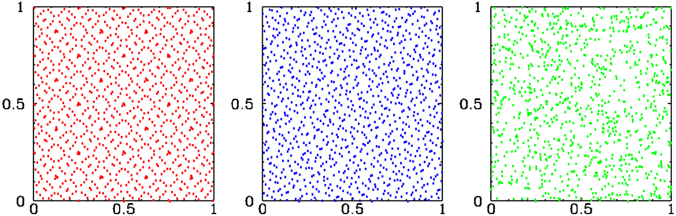
Error estimation

Numerical integration

Optimization

Quasi-random sequences

What is random?



All sets of points fill "randomly" the area [[0; 1]; [0; 1]]

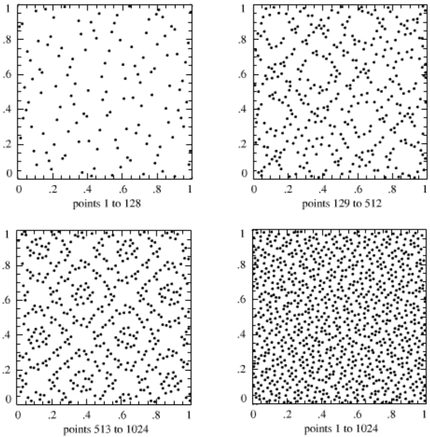
The left and center images are "sub-random" and fill more uniformly the area

These sequences are also called low-discrepancy sequences

These sequences can be used to replace the RNG when $x \sim \mathcal{U}(a, b)$ is needed

Quasi-random sequences

Filling of the plane



The sequence fills more or less uniformly the plane $\forall N$

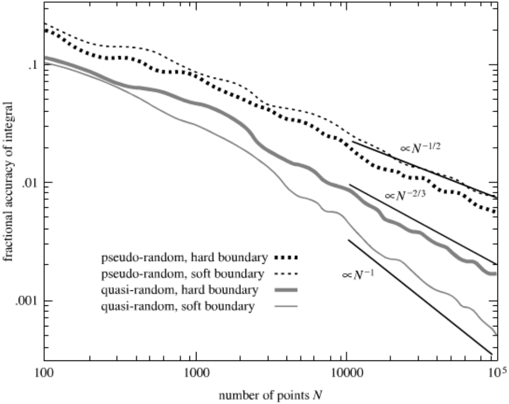
Quasi-random sequences

Examples of algorithms

- Sobol's sequence: Count in binary, but using Gray code and put a radix in front: 0.1, 0.11, 0.01, 0.011, 0.001, 0.101, 0.111, . . .
  This can be generalized to N dimensions
  This is the red set of points
- Halton's sequence: $H(i)$ is constructed the following way: take i expressed in a (small) prime-number base b (say $b = 3$), e.g. $i = 17 \equiv 122$ (base 3). Reverse the digits and put a radix in front, i.e.
  $H(17) = 0.221$ (base 3) $\simeq 0.92593$
  This is generalized to N dimensions by choosing different b's in each dimension
  This is the blue set of points

Quasi-random sequences

Accuracy



Convergence in some cases of numerical integration can reach $\sim 1/N$

Quasi-random sequences

GNU Scientific Library implements:

- Sobol's sequence
- Halton's sequence
- Niederreiter's sequence

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

- Markov chains
- Markov chains Monte-Carlo
- Metropolis-Hastings MCMC

Error estimation

Numerical integration

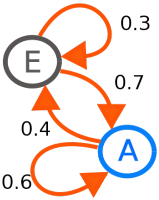
Optimization

Markov chains

Definition

A set of random variables $x_i, i = 1, \ldots$ is a Markov chain if:

$$P(x_{i+1} = x | x_1, \ldots, x_i) = P(x_{i+1} = x | x_i)$$



Markov chains

Examples

- State machine
- White noise: $x_{i+1} = \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}(\mu, \sigma)$
- Random walk: $x_{i+1} = x_i + \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}(\mu, \sigma)$
- AR[1] process: $x_{i+1} = \varphi \cdot x_i + \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}(\mu, \sigma),\ |\varphi| < 1$
- Google's *PageRank*

Markov chains

Properties

- *Homogeneity:* A Markov chain is homogeneous if $P(x_{i+1} = x|x_i) = P(x_{j+1} = x|x_j),\ \forall i, j$
- *Ergodicity:* A Markov chain is ergodic if $E(n|x_n = x_i, n > i) < \infty,\ \forall i$
  (Probably only valid when the number of states is finite)
  Ergodicity means that all possible states will be reached at some point
- *Reversibility:* A Markov chain is reversible if there exists a distribution $\Pi(x)$ such that: $\Pi(\alpha)P(x_{i+1}{=}\alpha|x_i{=}\beta) = \Pi(\beta)P(x_{i+1}{=}\beta|x_i{=}\alpha)$, $\forall i, \alpha, \beta$

Markov chains

More on reversibility

If a Markov chain is reversible:

$$\int_\alpha \Pi(\alpha) P(x_{i+1}=\alpha|x_i=\beta) = \int_\alpha \Pi(\beta) P(x_{i+1}=\beta|x_i=\alpha) =$$

$$= \Pi(\beta) \int_\alpha P(x_{i+1}=\beta|x_i=\alpha) = \Pi(\beta)$$

This property is also called detailed balance. $\Pi(x)$ is then the equilibrium distribution of the Markov chain.

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

- Markov chains
- Markov chains Monte-Carlo
- Metropolis-Hastings MCMC

Error estimation

Numerical integration

Optimization

Markov chains Monte-Carlo

Definition

Can we build a Markov chain that has pretty much any requested equilibrium distribution? YES!! The answer is Markov chain Monte-Carlo

- MCMC is one of the ten best algorithms ever! (along with FFT, QR decomposition, Quicksort, . . .)
- MCMC uses a homogeneous, ergodic and reversible Markov chain to generate consistent samples drawn from any given distribution
- No specific knowledge of the distribution, no specific support function is required
- Construction of the Markov chain Monte-Carlo is even straightforward

Markov chains Monte-Carlo

The Metropolis algorithm

Assuming a distribution $f(x)$ that we want to sample:

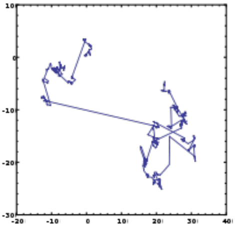
1. Choose a proposal distribution $p(x)$ and an initial value x_1
2. Select a candidate step using $p(x)$, so that: $\hat{x} = x_i + \varepsilon,\ \varepsilon \sim p(x)$
3. If $f(\hat{x}) > f(x_i)$ accept $x_{i+1} = \hat{x}$, otherwise accept $x_{i+1} = \hat{x}$ with a probability $f(\hat{x})/f(x_i)$, else reject and start again at 2.
4. Continue at 2. until you have enough values
5. Discard the early values (burning phase) which are influenced by the choice of x_1

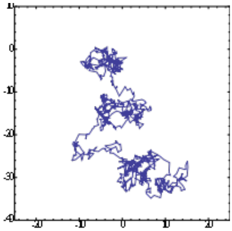
Markov chains Monte-Carlo

Random walk MCMC

Random-walk updates, i.e. proposal distribution is constant, i.e. $\hat{x} = x_i + \varepsilon_i,\ \varepsilon_i \sim p(x)$:

- Gaussian update: $p(x) = \mathcal{N}(0, \sigma)$
- Uniform update: $p(x) = \mathcal{U}(-\upsilon, \upsilon)$
- Student's t update
- Lévy-flight update: $p(x)$ is heavy-tailed (power law)





Markov chains Monte-Carlo

Non random walk MCMC

- Independence sampler: $\hat{x} \sim Y$, i.e. do not use x_i! This is equivalent to the rejection method
- Langevin update: $\hat{x} = x_i + \varepsilon_i + \frac{\sigma}{2}\nabla \log f(x_i)$. Uses some information about $f(x)$ to propose a better move
- Gareth's diffusion: Gaussian update, but $\sigma(x_i) = (\max_x f(x))/f(x_i)$. Makes bigger steps if we are in an area with small probability
- Gaussian AR[1] update: $\hat{x} = a + b(x_i - a) + \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}(0, \sigma), |b| \leq 1$. Not sure what good this does

Markov chains Monte-Carlo

Further notes on implementations

- Check the rejection ratio. Values between 30 and 70% are conventionally accepted
- Discard the burning phase. the autocorrelation function is a standard way to check if the initial value has become irrelevant or not
- The width of the proposal distribution (e.g. σ for a Gaussian update or υ for a uniform update) should be tuned during the burning phase to set the rejection fraction in the right bracket
- Reflection can be used when an edge of $f(x)$ is reached, e.g. set $\hat{x} = |x_i + \varepsilon_i|$ if x_i must remain positive
- Be careful with multimodal distributions. Not all modes may be sampled

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

- Markov chains
- Markov chains Monte-Carlo
- Metropolis-Hastings MCMC

Error estimation

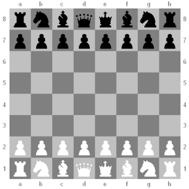
Numerical integration

Optimization

Metropolis-Hastings MCMC

Limitations of the Metropolis algorithm

- A symmetric proposal distribution might not be optimal
- Boundary effects: less time is spent close to boundaries, which might not be well sampled



- A correction factor, the Hastings ratio, is applied to correct for the bias
- The Hastings ratio usually speeds up convergence
- The choice of the proposal distribution becomes however more important

Metropolis-Hastings MCMC

The Hastings ratio

- The proposed update is $\hat{x} \sim Q(x; x_i)$
- The probability to accept the next point, A is modified so that instead of:

$$A = \min\left(1, \frac{f(\hat{x})}{f(x_i)}\right)$$

the probability is corrected by the Hastings ratio:

$$A = \min\left(1, \frac{f(\hat{x})}{f(x_i)} \frac{Q(\hat{x}; x_i)}{Q(x_i; \hat{x})}\right)$$

- If $Q(\alpha; \beta) = Q(\beta; \alpha)$, this is the Metropolis algorithm

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

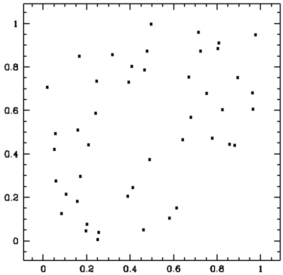
- Error distribution
- Error propagation
- Resampling methods: bootstrap

Numerical integration

Optimization

Error distribution

When hypotheses are not satisfied

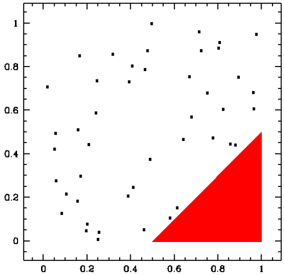


Are these points correlated: $N = 45$ points, $r = 0.42$?

The null-hypothesis probability that there is no correlation is 0.005

Error distribution

When hypotheses are not satisfied

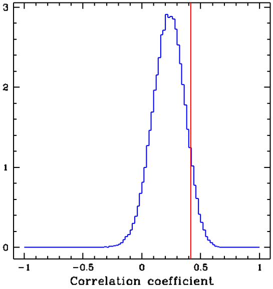


However, some parts of the plane ($y < x - 0.5$) are not accessible

The correct probability can be estimated using a Monte-Carlo simulation

Error distribution

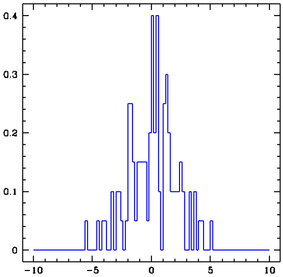
Using Monte-Carlo simulations



1. Draw N couples $(x_i, y_i) \sim (\mathcal{U}(0,1), \mathcal{U}(0,1))$, rejecting those for which $y_i < x_i - 0.5$
2. Calculate the correlation coefficient r
3. Repeat 1.–2. many times
4. Study the distribution. In this case the null hypothesis has a $\sim$ 10% probability

Statistical significance

When hypotheses are not satisfied



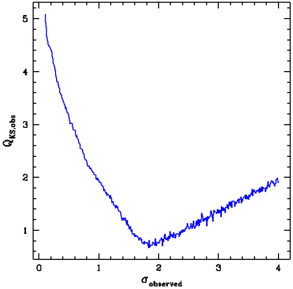
Is the distribution of these *N* values Gaussian?

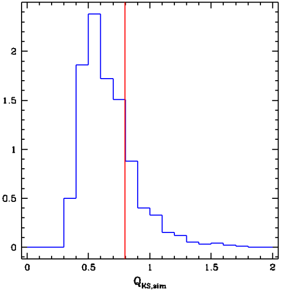
Use a Kolmogorov-Smirnov test Q_{KS}, and then use the null-hypothesis probabilities for this test

OK, but I do not know σ

Statistical significance

When hypotheses are not satisfied





1. Find the parameters $\sigma_{\rm observed}$ that minimizes $Q_{{\rm KS},obs}$, and the corresponding $Q_{{\rm KS},obs,min}$
2. Draw N points from a Gaussian with rms $\sigma_{\rm observed}$
3. Find the minimum of $Q_{{\rm KS},sim}$ as a function of trial σ
4. Repeat 2.–3. many times, and check the distribution of $Q_{{\rm KS},sim,min}$ against $Q_{{\rm KS},obs,min}$. In this case, 20% of the $Q_{{\rm KS},sim,min}$ exceed $Q_{{\rm KS},obs,min}$

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

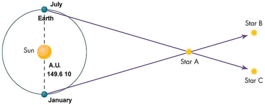
- Error distribution
- Error propagation
- Resampling methods: bootstrap

Numerical integration

Optimization

Error propagation

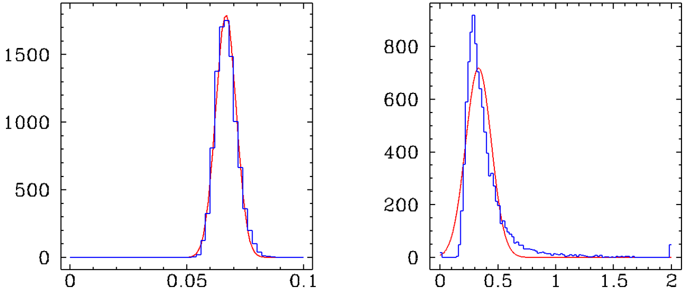
The case of the parallax



- One measures the parallax π of a given objects
- Assume that the known uncertainty σ on π is Gaussian, i.e. the true parallax Π and π are related by $\pi = \Pi + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma)$
- In the limit $\sigma \ll \pi$, we have the error propagation on the distance $D = 1/\pi$
  $\Delta D = \left|\frac{\partial D}{\partial \pi}\right| \Delta\pi = \left|\frac{1}{\pi^2}\right| \Delta\pi = \frac{\sigma}{\pi^2}$
- But higher-order terms in the serial expansion make the error distribution of D very asymmetrical
- A better solution is to perform the error propagation with a Monte-Carlo simulation

Error propagation

The case of the parallax illustrated

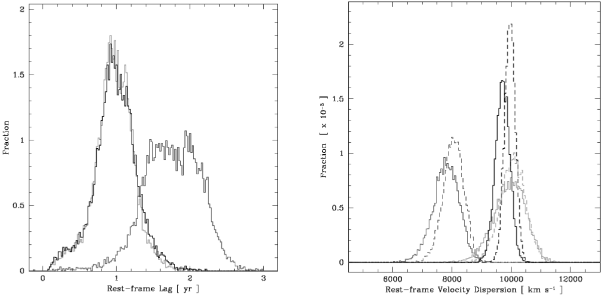


Monte-Carlo simulations allow to simulate the distribution of the distance *D* of a source in case of a parallax of 15 ± 1" (Left) and 3 ± 1" (Right)

Any inference on confidence intervals (for instance) is bound to be erroneous when $\sigma \ll \pi$ is not satisfied

Error propagation

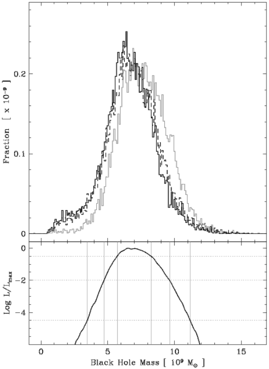
Real life example: Black-hole mass I



- $M_{\rm BH} = c \cdot \tau < v >^2$ is a non-linearly derived parameter from two measured parameters with non-Gaussian uncertainty distributions
- Pick many values τ and $< v >$ from their distributions and build the distribution of $M_{\rm BH}$

Error propagation

Real life example: Black-hole mass II



The resulting distribution can be explored to determine its moment and confidence intervals

Two independent estimates can be combined using a maximum-likelihood estimation

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Error distribution

Error propagation

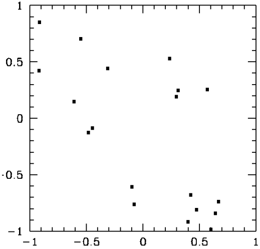
Resampling methods: bootstrap

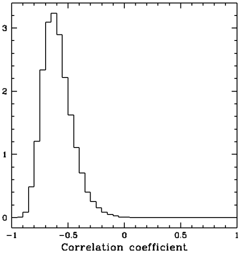
Numerical integration

Optimization

Bootstrap

A simple error estimation problem





Assume if have two sets of random variables $x_i,\ i = 1, \ldots, N$ and $y_j,\ i = 1, \ldots, N$, with $N = 20$ and a correlation coefficient $r = -0.63$. Are the two variables significantly correlated? Alternatively, what is the uncertainty on r?

In this case we can simulate new sets using Monte-Carlo under the null hypothesis, and estimate significance and uncertainty

Bootstrap

Getting something from nothing

What if we have no idea about the underlying distribution?

Intuition says that would be akin to pulling one's bootstraps to lift oneself

But the data contain hidden information about their distribution!

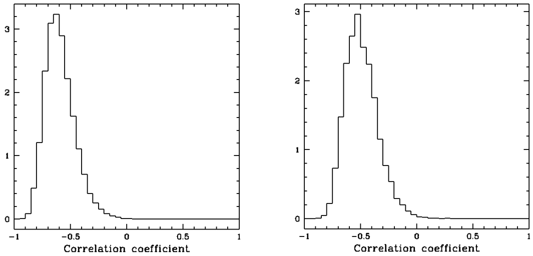
Bootstrap

General principle

- Generate many pseudo-samples by random, Monte-carlo, direct extraction of data points from the original sample
- Each pseudo-sample has the same size as the original sample
- As a consequence most pseudo-sample contain some repeated data points and some missing data points
- The statistics of interest is computed for all pseudo-samples, and its distribution is sampled, allowing to infer its statistical properties (uncertainty, mean, confidence intervals, . . .)

Bootstrap

A practical example

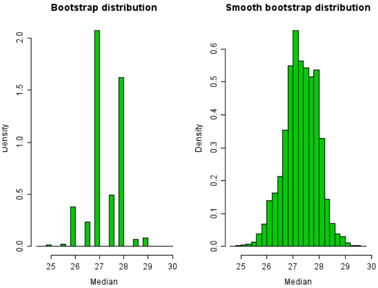


We generate a similar histogram with the data only:

1. Draw N indices (integers) α_i from $[1; N]$
2. Calculate the correlation coefficient r from the new sets x_{α_i} and y_{α_i}, $i = 1, \ldots, N$
3. Build and study the distribution of r by repeating 1.–2. many times

Bootstrap

Smoothed bootstrap



If the statistics of interest is discrete (i.e., can have only few values), the bootstrap distribution will be quite bad

One can add a smoothing random variable $\sim \mathcal{N}(0, \sigma^2)$, $\sigma = 1/\sqrt{N}$, to all the picked values of a bootstrap test

Even in the non-smoothed case, the confidence intervals can be pretty good

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Numerical integration

- Monte-Carlo algorithm
- Stratified sampling
- Importance sampling

Optimization

Monte-Carlo algorithm

Basic principle

$\int_V f \mathrm{d}V$ in M dimensions requires $\sim (1/h)^M$ steps

Given a volume V and a function f (with $<w>$ being the average of w over V):

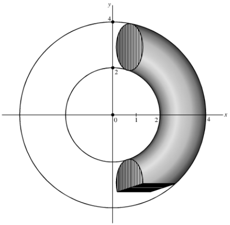
$$\int_V f \mathrm{d}V \simeq V <f> \pm V \sqrt{\frac{<f^2> - <f>^2}{N}}$$

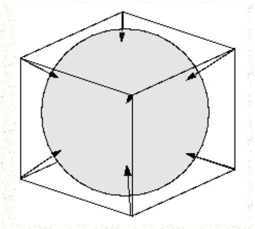
This is the basic theorem of Monte-Carlo integration

1. Draw N points at random within V
2. Calculate $<f>= \frac{1}{N} \sum_{i=1}^{N} f_i$ (and $<f^2>$)
3. Multiply $<f>$ by V to obtain the integral

Monte-Carlo algorithm

Volume determination





The volume of V can be determined by $\int_V f \mathrm{d}V$, with $f \equiv 1$

If it is not possible or not easy to sample uniformly the volume, one has to find a volume W containing V, which can be sampled and whose volume is known

The error is then $W\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}}$, but

$$f(w) = \begin{cases} 1, & \text{if } w \in V \\ 0, & \text{if not} \end{cases}$$

Monte-Carlo algorithm

How to minimize the error

The error term depend on the variance of f within V:

$$V\sqrt{\frac{<f^2> - <f>^2}{N}}$$

Can we reduce $<f^2> - <f>^2$?

If f varies strongly over V (e.g., $f(x,y,z) = e^{5z}$), one can perform a change of variable:

$$s = \frac{1}{5}e^{5z} \to \mathrm{d}s = e^{5z}\mathrm{d}z$$

$$\Longrightarrow \int_x \int_y \int_{z=-1}^{1} e^{5z}\mathrm{d}z = \int_x \int_y \int_{s=e^{-5}/5}^{e^5/5} \mathrm{d}s$$

Equivalent to a transformation method, but useless

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Numerical integration

- Monte-Carlo algorithm
- Stratified sampling
- Importance sampling

Optimization

Stratified sampling

Divide and conquer...

If $\hat{F}$ is the estimator of $F = \frac{1}{V}\int_V f \mathrm{d}V$,

$$\mathrm{Var}(\hat{F}) = \frac{\mathrm{Var}(f)}{N} \equiv \frac{\sigma^2}{N}$$

If one cut V in two equal volumes V_1 and V_2, the new estimator of F is:

$$\hat{F}\prime = \frac{1}{2}\left(\hat{F}_1 + \hat{F}_2\right) \Longrightarrow \mathrm{Var}(\hat{F}\prime) = \frac{1}{4}\left(\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}\right)$$

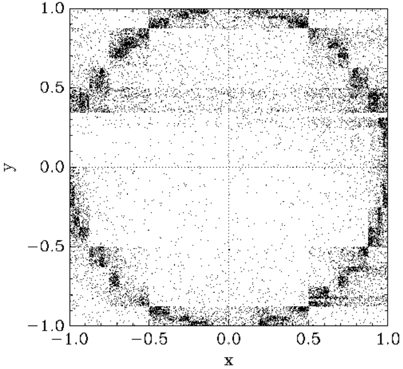
The minimum is reached for: $N_i/N = \sigma_i/(\sigma_1 + \sigma_2)$

$$\Longrightarrow \mathrm{Var}(\hat{F}\prime) = \frac{(\sigma_1 + \sigma_2)^2}{4N}$$

One can therefore cut V into many separate pieces, estimate the rms in each of them, and draw the points in each subvolume proportionally to the rms of f

Stratified sampling

at work…



Sampling is performed in the subvolumes where σ is large

Stratified sampling

The MISER algorithm

When V has many dimensions d, stratified sampling needs to cut the volume in K^d subvolumes

The MISER algorithm uses a recursive stratified sampling. It cuts the volume in two parts one dimension at a time:

1. For each dimension i, cut the current volume in two, V_1 and V_2, and determine $\sigma_{1,2}$
2. Find the most favorable dimension, e.g., by minimizing $\sigma_1^2 + \sigma_2^2$
3. Perform additional MC evaluation in V_1 and V_2 minimizing the error
4. Iterate 1.–3. as long as the number of evaluations remains not too prohibitive

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Numerical integration

- Monte-Carlo algorithm
- Stratified sampling
- Importance sampling

Optimization

Importance sampling

Using a sampling distribution

Instead of sampling uniformly, we use a distribution *p*:

$$\int_V p \mathrm{d}V = 1$$

$$\Longrightarrow \int_V f \mathrm{d}V = \int_V \frac{f}{p} p \mathrm{d}V = \left\langle \frac{f}{p} \right\rangle \pm \sqrt{\frac{< f^2/p^2 > - < f/p >^2}{N}}$$

Given *N*, the square root is minimized if:

$$p = \frac{|f|}{\int_V |f| \mathrm{d}V}$$

The idea of importance sampling is to find a distribution *p* as close as possible to *f*. Importance sampling requires some prior knowledge of *f*

Importance sampling

The VEGAS algorithm

The VEGAS algorithm implements importance sampling with a separable distribution in d dimensions:

$$p \propto g_1(x_1) \cdot g_2(x_2) \cdot g_3(x_3) \ldots \cdot g_d(x_d)$$

VEGAS uses an adaptive strategy to determine $g_i(x_i)$

1. Choose arbitrary $g_i(x_i)$ (e.g., constant)
2. Cut each axes in K pieces, and determine the integral and the Kd importance sampling weights $g_i(k),\ k = 1, \ldots, N$
3. Iterate 2. a (small) number of times to improve the weights $g_i(k)$ and the integral

VEGAS efficiency depends strongly on the validity of the separability of f, and will be small if the integrand is dominated by a subspace of dimension $d_1 < d$

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

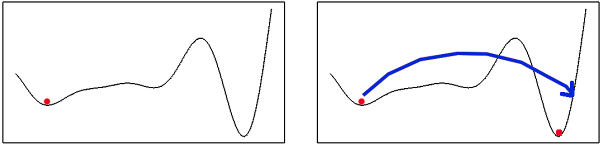
Numerical integration

Optimization

- Stochastic optimization
- Swarm intelligence
- Genetic algorithms
- Simulated annealing

Stochastic optimization

The problem of local minima



Minimization of complex functions in many dimensions often shows multiple minima

Random (i.e., Monte-Carlo) approaches can help

In Nature, the problem is often encountered… and solved (or approximated)

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

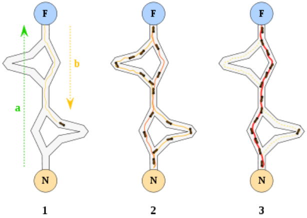
Numerical integration

Optimization

- Stochastic optimization
- Swarm intelligence
- Genetic algorithms
- Simulated annealing

Swarm intelligence

The ant colony optimization



Numerous agents explore the landscape with simple interactions between them

Ants move at random in the solution space and deposit pheromones behind them, which attract other ants

Shorter paths are crossed more rapidly by ants, so more pheromone is deposited, attracting more ants

Try also bees!

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

Numerical integration

Optimization

- Stochastic optimization
- Swarm intelligence
- Genetic algorithms
- Simulated annealing

Genetic algorithms

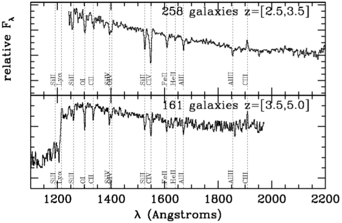
Principle

Use Darwinian evoution of a gene pool to find the fittest genes

- Each chromosome represents a candidate solution to the problem at hand, represented as a series of genes
- A fitness function can be calculated for each chromosome
- At each generation
  1. Genes mutate,
  2. Chromosomes reproduce sexually
  3. Non-biological mechanisms are also possible
  4. Chromosomes are selected according to their fitness
- After the stopping criterion is used, the best chromosome provides the approximate solution

Genetic algorithms

Example



You want to build the best possible average spectrum with a sample of *N* low-S/N high-redshift galaxies, but you know the sample contain 50% interlopers (galaxies with wrongly identified redshifts)

- The chromosomes are a series of *N* bits, half of them set to 1, the rest to 0
- The fitness function is the summed equivalent width of the expected line features
- Mutation are random inclusion or exclusion of galaxies

Outline

What are Monte-Carlo methods?

Generation of random variables

Markov chains Monte-Carlo

Error estimation

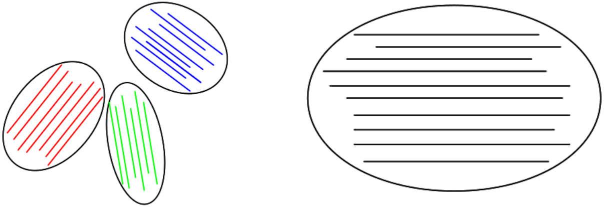
Numerical integration

Optimization

- Stochastic optimization
- Swarm intelligence
- Genetic algorithms
- Simulated annealing

Simulated annealing

Principle



In nature, crystal (e.g., ice) can form over large areas, thus minimizing internal energy

Annealing is the process of slowly cooling a material to relieve internal stress

Nature performs an annealing by slowly decreasing temperature, so that misaligned ice crystals can melt and freeze in a more favorable direction

Simulated annealing

Implementation

Simulated annealing requires:

1. A configuration of all possible states S, i.e. the space of the problem
2. An internal energy E, which we want to minimize
3. “Options” for the next state after the current state, similar to updates in Metropolis algorithms
4. A temperature T which is related to the probability to accept an increase in energy:

$$S_0 \text{ is accepted if } \varsigma \sim \mathcal{U}(0,1) < \exp\left(\frac{-\Delta E}{kT}\right)$$

5. A cooling schedule, which determines how T evolve with the number of iterations

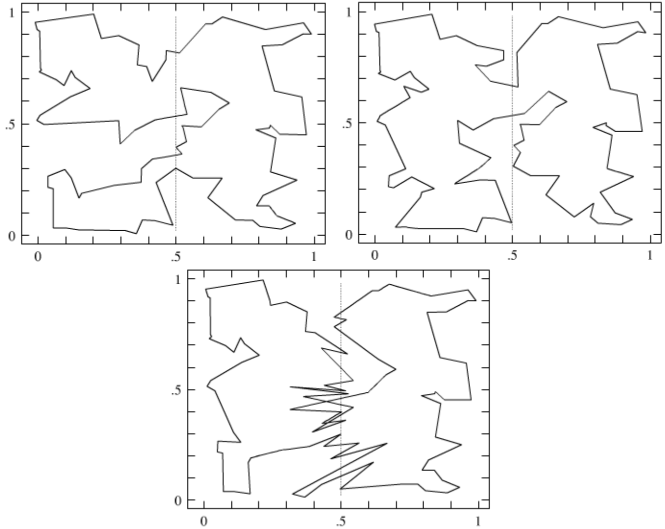
Simulated annealing

Implementation II

- $\{S\}$ and E are the problem to solve, i.e. we want $S_{\rm min} \mid E(S_{\rm min}) < E(S),\ \forall S \neq S_{\rm min}$
- The options are a tricky point, as a completely random move in many dimensions might be very inefficient in case of narrow valleys. The downhill simplex (amoeba) algorithm can be tried
- The acceptance condition makes it similar to the Metropolis algorithm
- There are also several possibilities for the cooling schedule, for instance $T \rightarrow (1-\varepsilon)T$ at each step

Simulated annealing

Example


1
.5
0
0
.5
1
1
.5
0
0
.5
1
1
.5
0
0
.5
1