Notes on Classical Monte Carlo?

]

I. MONTE CARLO METHOD

Monte Carlo methods are algorithms for solving various kinds of computational problems by using random numbers (or more often pseudo-random numbers), as opposed to deterministic algorithms. Monte Carlo methods are extremely important in computational physics and related applied fields, and have diverse applications from esoteric quantum chromodynamics calculations to designing heat shields and aerodynamic forms. These methods have proven efficient in solving the integro-differential equations defining the radiance field, and thus these methods have been used in global illumination computations which produce photo-realistic images of virtual 3d models, with applications in video games, architecture, design, computer generated films and special effects in cinema, and other fields.

Interestingly, the Monte Carlo method does not require truly random numbers to be useful. Much of the most useful techniques use deterministic, pseudo-random sequences, making it easy to test and re-run simulations. The only quality usually necessary to make good simulations is for the pseudo-random sequence to appear "random enough" in a certain sense. That is that they must either be uniformly distributed or follow another desired distribution when a large enough number of elements of the sequence are considered.

Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is a method suited to calculation using a computer, utilizing many techniques of computer simulation.

A Monte Carlo algorithm is a numerical Monte Carlo method used to find solutions to mathematical problems (which may have many variables) that cannot easily be solved, for example, by integral calculus, or other numerical methods. Its efficiency relative to other numerical methods increases when the dimension of the problem increases.

II. HISTORY

Monte Carlo methods were originally practiced under more generic names such as "statistical sampling". The "Monte Carlo" designation, popularized by early pioneers in the field (including Stanislaw Marcin Ulam, Enrico Fermi, John von Neumann and Nicholas Metropolis), is a reference to the famous casino in that area, under the belief that Monte Carlo lends its name to this method because of the use of randomness and the repetitive nature employed to find an approximation to the solution. Stanislaw Marcin Ulam tells in his autobiography "Adventures of a Mathematician" that the method was named in honor of his uncle, who was a gambler at the mentioned casino.

Random methods of computation have their roots in the pre-electronic computing era. Perhaps the most famous early use was by Fermi in 1930, when he used a random method to calculate the properties of the newly-discovered neutron (see also Buffon's needle). Monte Carlo methods were central to the simulations required for the Manhattan Project. (An alternate explanation of the Monte Carlo name is that the Manhattan Project team met in Monte Carlo to create their methods.) However, it was only after electronic computers were first built (from 1945 on) that Monte Carlo methods began to be studied in depth.

III. INTEGRATION

Deterministic methods of numerical integration operate by taking a number of evenly spaced samples from a function. In general, this works very well for functions of one variable. However, for functions of vectors, deterministic quadrature methods can be very inefficient. To numerically integrate a two-dimensional vector, equally spaced grid points over a two-dimensional surface are required. For instance a 10x10 grid requires 100 points. If the vector has 100 dimensions, the same spacing on the grid would require 10100 points that's far too many to be computed. 100 dimensions is by no means unreasonable, since in many physical problems, a "dimension" is equivalent to a degree of freedom, and in a three-dimensional simulation, there are at least three degrees of freedom per particle.

Monte Carlo methods provide a way out of this exponential time-increase. As long as the function in question is reasonably well-behaved, it can be estimated by randomly selecting points in 100-dimensional space, and taking some kind of "average". By the central limit theorem, this method will display $1/\sqrt{N}$ convergerence i.e. quadrupling the number of sampled points will halve the error, regardless of the number of dimensions.

IV. METROPOLIS-HASTINGS ALGORITHM

The Sampling distribution Q determines the next point to move to in the random walk.

In mathematics and physics, the Metropolis-Hastings algorithm is an algorithm to generate a sequence of samples from the joint distribution of two or more variables. The purpose of such a sequence is to approximate the joint distribution (as with a histogram), or to compute an integral (such as an expected value). This algorithm is an example of a Markov chain Monte Carlo algorithm. It is a generalization of the Metropolis algorithm suggested by Hastings (citation below). The Gibbs sampling algorithm is a special case of the Metropolis-Hastings algorithm.

The Metropolis-Hastings algorithm can draw samples from any probability distribution $P(x)$, requiring only that the density can be calculated at x. The algorithm generates a set of states xt which is a Markov chain because each state xt depends only on the previous state x^{t-1}. The algorithm depends on the creation of a proposal density $Q(x';x^t)$ which depends on the current state x^t and which can generate a new proposed sample x'. For example, the proposal density could be a Gaussian function centred on the current state x^t

$$Q(x';x^t) \sim N(x' - x^t, \sigma^2 I)$$

reading $Q(x';x^t)$ as the probability of generating x' given the previous value x^t.

This proposal density would generate samples centred around the current state with variance $\sigma^2 I$. So we draw a new proposal state x' with probability $Q(x';x^t)$ and then calculate a value

$$a = a_1 a_2,$$

where

$$a_1 = \frac{P(x')}{P(x^t)}$$

is the likelihood ratio between the proposed sample x' and the previous sample x^t, and

$$a_2 = \frac{Q(x^t;x')}{Q(x';x^t)}$$

is the ratio of the proposal density in two directions (from x^t to x' and vice versa). This is equal to 1 if the proposal density is symmetric. Then the new state x^{t+1} is chosen with the rule

$$x^{t+1} = \begin{cases} x' & \text{if } a > 1 \\ x' \text{ with probability } a, & \text{if } a < 1 \end{cases}$$

The Markov chain is started from a random initial value x^0 and the algorithm is run for a few thousand iterations so that this initial state is "forgotten". These samples, which are discarded, are known as burn-in. The algorithm works best if the proposal density matches the shape of the target distribution $P(x)$, that is $Q(x';x^t) \approx P(x')$, but in most cases this is unknown. If a Gaussian proposal is used the variance parameter O_2 has to be tuned during the burn-in period. This is usually done by calculating the acceptance rate, which is the fraction of proposed samples that is accepted in a window of the last N samples. This is usually set to be around 60%. If the proposal steps are too small the chain will mix slowly i.e. it will move around the space slowly and converge slowly to $P(x)$. If the proposal steps are too large the acceptance rate will be very low because the proposals are likely to land in regions of much lower probability density so a1 will be very small.

Monte Carlo Method : What's in a Name?

- Casino. Gambling.

Random variables, which is close enough.

Some History.

Monte Carlo methods were originally practiced under more generic names such as "statistical sampling". The "Monte Carlo" designation, popularized by early pioneers in the field (including Stanislaw Marcin Ulam, Enrico Fermi, John von Neumann and Nicholas Metropolis), is a reference to the famous casino in that area, under the belief that Monte Carlo lends its name to this method because of the use of randomness and the repetitive nature employed to find an approximation to the solution. Stanislaw Marcin Ulam tells in his autobiography "Adventures of a Mathematician" that the method was named in honor of his uncle, who was a gambler at the mentioned casino.

Integrals.

Nice to calculate precisely, or to give an analytical formula for the answer.

Often (almost always?) not possible.

When we have to integrate complicated function in N-dimensional space,

$\Rightarrow$ numerical integration.

Monte Carlo Method Example: Calculating Area of a Circle

Area of a circle:

$$A = \int_S dxdy$$

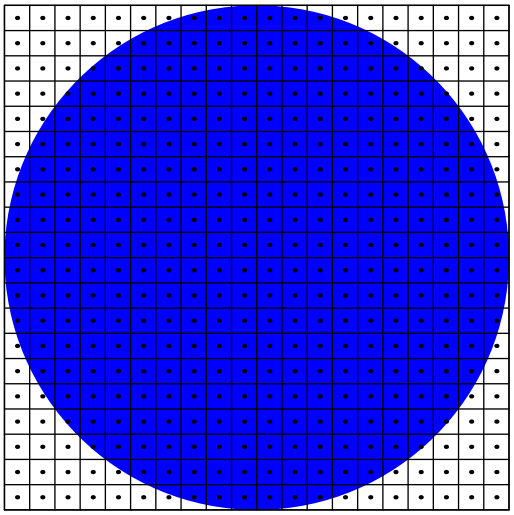
Or

$$A = \int f(x,y)dxdy, \qquad f(x,y) \equiv \left[\begin{array}{ll} 1, & (x,y)\text{inside circle} \\ 0, & otherwise. \end{array} \right.$$

Integral sum:

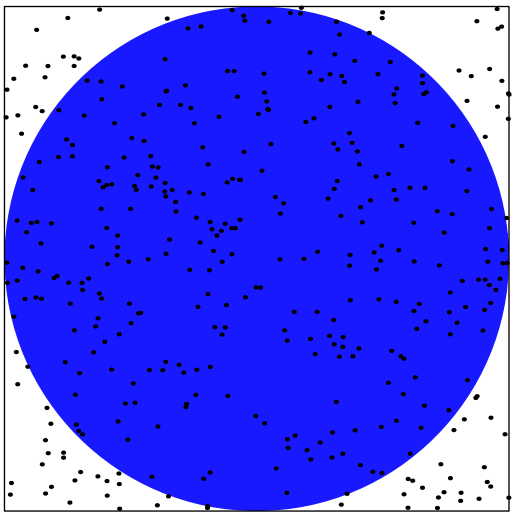
$$A = \sum f(x_i, y_i)\Delta S_i$$

Grid of points at equal spacing:



$$A = \frac{S(square)}{n} \sum_{i=1}^{n} f(x_i, y_i)$$

Monte Carlo method:
Expression for the sum the same, but now points (x_i, y_i) are chosen **randomly**:



A simple program calculating π through the area of a circle:

$$\pi = \frac{\pi r^2}{r^2} = \frac{4\ S(circle)}{S(square)}$$

A simple program calculating volume of 10-dimensional sphere.
The exact formula of N-dimensional sphere of radius R:

$$V_N(R) = \frac{\pi^{N/2}}{(N/2)!} R^N$$

Physics: A Lot of Integrals

Mean value of observable A:

$$< A >=< \psi|A|\psi >= \int \psi^* A \psi d^n x$$

Energy

$$< E >= \frac{< \psi|H|\psi >}{< \psi|\psi >}$$

Dimension may be very high.

Statistical Mechanics / Condensed Matter

Monte Carlo:
Random points r_i:

$$< E >= \frac{\sum_i E(r_i)|\psi(r_i)|^2}{\sum_i |\psi(r_i)|^2}$$

Central Limit Theorem.

Let $X_1, X_2, ...$ be independent random variables which are identically distributed (i.e. all have the same probability function in the discrete case or density function in the continuous case) and have finite mean μ and variance σ^2. Then if $S_n = X_1 + X_2 + ... + X_n$,

$$\lim_{n\to\infty} P\left(a \leq \frac{S_n - n\mu}{\sigma\sqrt{n}} \leq b\right) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-u^2/2} du$$

I.e. random variable $(S_n - n\mu)/\sigma\sqrt{n}$ is asymptotically normal.

If we use for each of X_i above values for some observable obtained in Monte Carlo simulations, e.g. mean value for Y:

$$X_i = \frac{1}{M} \sum_{j=1}^{M} Y_j^{(i)}$$

then if M is large enough, the distribution of X_i must be normal and we'll be able to estimate Y, and the error, with good precision if we keep simulation long enough.

And number of dimensions doesn't seem to matter.

Precision of MC simulation should be roughly $\sim 1/\sqrt{M}$.

Show: 20-d sphere program. Why doesn't work good?

Most of the time is spent outside of the sphere, where there's zero weight of a sample.

Metropolis Algorithm

0. Choose initial position in a (configuration, Hilbert) space, $X_i = X_0$.

1. Suggest a random step X_{trial}.

2. Select a random variable r, $0 < r < 1$.
If $r < P(X_{trial})/P(X_i)$, then accept step X_{trial} as a new position $X_{i+1} = X_{trial}$; otherwise accept X_i: $X_{i+1} = X_i$.
Goto 1.

Observables are calculated as

$$< A >= \frac{1}{N} \sum_{i=1}^{N} A(X_i)$$

Won't give proof now (read: too difficult for me).

Intuitively, it must be clear that thanks to the weighted steps, algorithm will spend most of the time at points with relatively high probability.

It is helpful, especially if the probability space is infinite, or very large, but almost all of it has very low probability density.

Metropolis Algorithm for Statistical Mechanics

0. Choose initial position $X_i = X_0$.

1. Suggest a random step X_{trial}.

2. Select a random variable r, $0 < r < 1$.
If $r < e^{\beta[\epsilon(X_i)-\epsilon(X_{trial})]}$, then accept step X_{trial} as a new position $X_{i+1} = X_{trial}$; otherwise accept X_i: $X_{i+1} = X_i$.
Goto 1.

Quantum Monte Carlo:
Path Integral Monte Carlo.

Classical Mechanics:
The least action

$$S = \int_{x_0,t_0=0}^{x,t} L dt$$

Quantum Mechanics:
Propagator: Dirac-Feynman path integral

$$G(x, x_0, t) = \sum_{\text{all paths}} e^{\frac{i}{\hbar}S} = \int_{x_0,t_0=0}^{x,t} D[x(t)] e^{\frac{i}{\hbar}S}$$

Green function G satisfies

$$\psi(x,t) = \int G(x, x_0, t)\psi(x_0, 0) dx_0$$

$(\psi(x,t) = \sum_n c_n \phi_n(x) e^{\frac{i}{\hbar}E_n t},\ \psi(x_0,0) = \sum_n c_n \phi_n(x_0) e^{\frac{i}{\hbar}E_n t},\ H\phi_n = E_n\phi_n,\ \int \phi_m \phi_n dx = \delta_{m.n}.)$

$$\Rightarrow \quad G(x, x_0, t) = \sum_n c_n \phi_n(x)\phi_n(x_0) e^{\frac{i}{\hbar}E_n t}$$

Go to imaginary time: change of variables $\tau = \frac{i}{\hbar}t$:

$$\Rightarrow \quad G(x, x_0, \tau) = \sum_n c_n \phi_n(x)\phi_n(x_0) e^{-E_n \tau}$$

Ground state:

$$G(x, x, \tau \to \infty) = \phi_0(x)^2 e^{-E_0 \tau}$$

Find action S. For a single particle of mass $m = 1$:

$$L = -\frac{1}{2}\left(\frac{dx}{d\tau}\right)^2 - V(x) = -E$$

Divide τ interval into N steps $\Delta\tau$:

$$E(x_j, \tau_j) = \frac{1}{2}\left(\frac{x_{j+1} - x_j}{\Delta\tau}\right)^2 + V(x_j), \quad \tau_j = j\Delta\tau$$

Probability of the path

$$e^{iS} = e^{-\Delta\tau \sum_j \left[\frac{1}{2}\left(\frac{x_{j+1}-x_j}{\Delta\tau}\right)^2 + V(x_j)\right]}$$

The propagator

$$G(x = x_N, x_0, N\Delta\tau) \ = \ A \int dx_1 ... dx_{N-1} e^{-\Delta\tau \sum_j \left[\frac{1}{2}\left(\frac{x_{j+1}-x_j}{\Delta\tau}\right)^2 + V(x_j)\right]}$$

Path Integral Monte Carlo Algorithm.

1. Choose N, $\Delta\tau$.
Choose δ - the maximum trial change movement of a particle.
Choose initial configuration for positions $\{x_j\}$.

2. Choose j randomly (*a variant: choose j by turn*)
Choose random p, $0 < p < 1$.
Generate a trial displacement $x_{trial} = x_j + (2p-1)\delta$.

$$\Delta E = \left[\frac{1}{2}\left(\frac{x_{j+1} - x_{trial}}{\Delta\tau}\right)^2 + \frac{1}{2}\left(\frac{x_{trial} - x_{j-1}}{\Delta\tau}\right)^2 + V(x_{trial})\right]$$

$$- \left[\frac{1}{2}\left(\frac{x_{j+1} - x_j}{\Delta\tau}\right)^2 + \frac{1}{2}\left(\frac{x_j - x_{j-1}}{\Delta\tau}\right)^2 + V(x_j)\right]$$

Choose random r, $0 < r < 1$.
If $r < e^{-\Delta\tau\Delta E}$, then accept the move: $x_j = x_{trial}$;
Otherwise, accept x_j.

Cumulate the observable(s):

$$\sum A = \sum A + A(x_0, ...x_N); \quad < A >= \frac{1}{n}\sum A \quad (\text{n = number of trials})$$

Repeat 2, until we get satisfied with the results, or get bored.

•

References

* B.L. Hammond, W.L. Lester, Jr, P.J. Reynolds, Monte Carlo Methods in Ab Initio Quantum Chemistry, 1995.

* Feynman, Higgs. Quantum Mechanics and Path Integrals, 1965.

* Professor Scalettar's lecture notes - can be found on his site.

* Murray R. Spiegel, Schaum's Outline of Theory and Problems of Probability and Statistics.

* P. Kevin MacKeown, Stochastic Simulation in Physics, 1997, ISBN 981-3083-26-3

* Harvey Gould & Jan Tobochnik, An Introduction to Computer Simulation Methods, Part 2, Applications to Physical Systems, 1988, ISBN 020116504X

* C.P. Robert and G. Casella. "Monte Carlo Statistical Methods" (second edition). New York: Springer-Verlag, 2004, ISBN 0-387-21239-6

* Chib, Siddhartha and Edward Greenberg: "Understanding the MetropolisHastings Algorithm". American Statistician, 49, 327335, 1995

* W.K. Hastings. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications", Biometrika, 57:97-109, 1970.

* N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. "Equations of State Calculations by Fast Computing Machines". Journal of Chemical Physics, 21:1087-1091, 1953.