WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

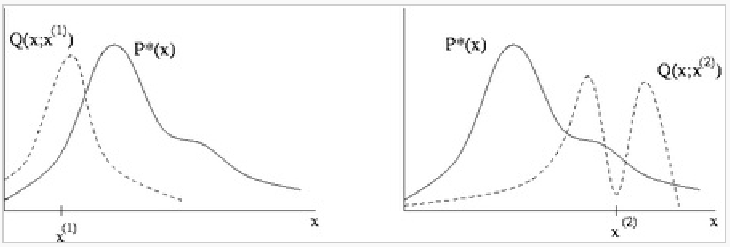
Print/export
Create a book
Download as PDF
Printable version

Languages
Čeština
Deutsch
فارسی
Français
한국어
Հայերեն
Italiano
日本語
Русский
Suomi
Edit links

Metropolis–Hastings algorithm

From Wikipedia, the free encyclopedia

In statistics and in statistical physics, the **Metropolis–Hastings algorithm** is a Markov chain Monte Carlo (MCMC) method for obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult. This sequence can be used to approximate the distribution (i.e., to generate a histogram), or to compute an integral (such as an expected value). Metropolis–Hastings and other MCMC algorithms are generally used for sampling from multi-dimensional distributions, especially when the number of dimensions is high. For single-dimensional distributions, other methods are usually available (e.g. adaptive rejection sampling) that can directly return independent samples from the distribution, and are free from the problem of auto-correlated samples that is inherent in MCMC methods.



The proposal distribution Q proposes the next point that the random walk might move to.

History [edit]

The algorithm was named after Nicholas Metropolis, who was an author along with Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller of the 1953 paper *Equation of State Calculations by Fast Computing Machines* which first proposed the algorithm for the specific case of the canonical ensemble;[1][2] and W. K. Hastings who extended it to the more general case in 1970.[3] There is controversy over the credit for discovery of the algorithm. Edward Teller states in his memoirs that the five authors of the 1953 paper worked together for "days (and nights)".[4] M. Rosenbluth, in an oral history recorded shortly before his death[5] credits E. Teller with posing the original problem, himself with solving it, and A.W. Rosenbluth (his wife) with programming the computer. According to M. Rosenbluth, neither Metropolis nor A.H. Teller participated in any way. Rosenbluth's account of events is supported by other contemporary recollections.[6] According to Roy Glauber and Emilio Segrè, the original algorithm was invented by Enrico Fermi and reinvented by Stan Ulam.

Intuition [edit]

The Metropolis–Hastings algorithm can draw samples from any probability distribution *P(x)*, provided you can compute the value of a function *f(x)* which is *proportional* to the density of *P*. The lax requirement that *f(x)* should be merely proportional to the density, rather than exactly equal to it, makes the Metropolis–Hastings algorithm particularly useful, because calculating the necessary normalization factor is often extremely difficult in practice.

The Metropolis–Hastings algorithm works by generating a sequence of sample values in such a way that, as more and more sample values are produced, the distribution of values more closely approximates the desired distribution, *P(x)*. These sample values are produced iteratively, with the distribution of the next sample being dependent only on the current sample value (thus making the sequence of samples into a Markov chain).

Specifically, at each iteration, the algorithm picks a candidate for the next sample value based on the current sample value. Then, with some probability, the candidate is either accepted (in which case the candidate value is used in the next iteration) or rejected (in which case the candidate value is discarded, and current value is reused in the next iteration)−the probability of acceptance is determined by comparing the values of the function *f(x)* of the current and candidate sample values with respect to the desired distribution *P(x)*.

For the purpose of illustration, the Metropolis algorithm, a special case of the Metropolis–Hastings algorithm where the proposal function is symmetric, is described below.

Metropolis algorithm (symmetric proposal distribution)

Let *f(x)* be a function that is proportional to the desired probability distribution *P(x) (a.k.a. a target distribution)*.

1. Initialization: Choose an arbitrary point x_0 to be the first sample, and choose an arbitrary probability density $Q(x|y)$ which suggests a candidate for the next sample value *x*, given the previous sample value *y*. For the Metropolis algorithm, *Q* must be symmetric; in other words, it must satisfy $Q(x|y) = Q(y|x)$. A usual choice is to let $Q(x|y)$ be a Gaussian distribution centered at *y*, so that points closer to *y* are more likely to be visited next—making the sequence of samples into a random walk. The function *Q* is referred to as the *proposal density* or *jumping distribution*.
2. For each iteration *t*:
   - Generate a candidate *x′* for the next sample by picking from the distribution $Q(x'|x_t)$.
   - Calculate the *acceptance ratio* α = $f(x')/f(x_t)$, which will be used to decide whether to accept or reject the candidate. Because *f* is proportional to the density of *P*, we have that $\alpha = f(x')/f(x_t) = P(x')/P(x_t)$.
   - If α ≥ 1, then the candidate is more likely than x_t; automatically accept the candidate by setting $x_{t+1} = x'$. Otherwise, accept the candidate with probability α; if the candidate is rejected, set $x_{t+1} = x_t$, instead.

This algorithm proceeds by randomly attempting to move about the sample space, sometimes accepting the moves and sometimes remaining in place. Note that the acceptance ratio α indicates how probable the new proposed sample is with respect to the current sample, according to the distribution $P(x)$. If we attempt to move to a point that is more probable than the existing point (i.e. a point in a higher-density region of $P(x)$), we will always accept the move. However, if we attempt to move to a less probable point, we will sometimes reject the move, and the more the relative drop in probability, the more likely we are to reject the new point. Thus, we will tend to stay in (and return large numbers of samples from) high-density regions of $P(x)$, while only occasionally visiting low-density regions. Intuitively, this is why this algorithm works, and returns samples that follow the desired distribution $P(x)$.

Compared with an algorithm like adaptive rejection sampling that directly generates independent samples from a distribution, Metropolis–Hastings and other MCMC algorithms have a number of disadvantages:

- The samples are correlated. Even though over the long term they do correctly follow $P(x)$, a set of nearby samples will be correlated with each other and not correctly reflect the distribution. This means that if we want a set of independent samples, we have to throw away the majority of samples and only take every *n*th sample, for some value of *n* (typically determined by examining the auto-correlation between adjacent samples). Auto-correlation can be reduced by increasing the *jumping width* (the average size of a jump, which is related to the variance of the jumping distribution), but this will also increase the likelihood of rejection of the proposed jump. Too large or too small a jumping size will lead to a *slow-mixing* Markov chain, i.e. a highly correlated set of samples, so that a very large number of samples will be needed to get a reasonable estimate of any desired property of the distribution.
- Although the Markov chain eventually converges to the desired distribution, the initial samples may follow a very different distribution, especially if the starting point is in a region of low density. As a result, a *burn-in* period is typically necessary, where an initial number of samples (e.g. the first 1,000 or so) are thrown away.

On the other hand, most simple rejection sampling methods suffer from the "curse of dimensionality", where the probability of rejection increases exponentially as a function of the number of dimensions. Metropolis–Hastings, along with other MCMC methods, do not have this problem to such a degree, and thus are often the only solutions available when the number of dimensions of the distribution to be sampled is high. As a result, MCMC methods are often the methods of choice for producing samples from hierarchical Bayesian models and other high-dimensional statistical models used nowadays in many disciplines.

In multivariate distributions, the classic Metropolis–Hastings algorithm as described above involves choosing a new multi-dimensional sample point. When the number of dimensions is high, finding the right jumping distribution to use can be difficult, as the different individual dimensions behave in very different ways, and the jumping width (see above) must be "just right" for all dimensions at once to avoid excessively slow mixing. An alternative approach that often works better in such situations, known as Gibbs sampling, involves choosing a new sample for each dimension separately from the others, rather than choosing a sample for all dimensions at once. This is especially applicable when the multivariate distribution is composed out of a set of individual random variables in which each variable is conditioned on only a small number of other variables, as is the case in most typical hierarchical models. The individual variables are then sampled one at a time, with each variable conditioned on the most recent values of all the others. Various algorithms can be used to choose these individual samples, depending on the exact form of the multivariate distribution: some possibilities are adaptive rejection sampling, a one-dimensional Metropolis–Hastings step, or slice sampling.

Formal derivation of the Metropolis-Hastings algorithm [edit]

The purpose of the Metropolis–Hastings algorithm is to generate a collection of states according to a desired distribution P(x). To accomplish this, the algorithm uses a Markov process which asymptotically reaches a unique stationary distribution π(x) such that π(x)=P(x) .[7]

A Markov process is uniquely defined by its transition probabilities, $P(x \rightarrow x')$, the probability of transitioning from any given state, x, to any other given state, x'. It has a unique stationary distribution π(x) when the following two conditions are met:[7]

1. **existence of stationary distribution**: there must exist a stationary distribution π(x). A sufficient but not necessary condition is detailed balance which requires that each transition x→x' is reversible: for every pair of states x, x', the probability of being in state x and transitioning to state x' must be equal to the probability of being in state x' and transitioning to state x, $\pi(x)P(x \rightarrow x') = \pi(x')P(x' \rightarrow x)$.
2. **uniqueness of stationary distribution**: the stationary distribution π(x) must be unique. This is guaranteed by ergodicity of the Markov process, which requires that every state must (1) be aperiodic—the system does not return to the same state at fixed intervals; and (2) be positive recurrent—the expected number of steps for returning to the same state is finite.

Metropolis–Hastings algorithm resides in designing a Markov process (by constructing transition probabilities) which fulfils the two above conditions, such that its stationary distribution π(x) is chosen to be *P(x)*. The derivation of the algorithm starts with the condition of detailed balance:

$$P(x)P(x \rightarrow x') = P(x')P(x' \rightarrow x)$$

which is re-written as

$$\frac{P(x \rightarrow x')}{P(x' \rightarrow x)} = \frac{P(x')}{P(x)}.$$

The approach is to separate the transition in two sub-steps; the proposal and the acceptance-rejection. The **proposal distribution** $g(x \rightarrow x')$ is the conditional probability of proposing a state x' given x, and the **acceptance distribution** $A(x \rightarrow x')$ the conditional probability to accept the proposed state x'. The transition probability can be written as the product of them:

$$P(x \rightarrow x') = g(x \rightarrow x')A(x \rightarrow x').$$

Inserting this relation the previous equation, we have

$$\frac{A(x \rightarrow x')}{A(x' \rightarrow x)} = \frac{P(x')}{P(x)}\frac{g(x' \rightarrow x)}{g(x \rightarrow x')}.$$

The next step in the derivation is to choose an acceptance that fulfils the condition above. One common choice is the Metropolis choice:

$$A(x \rightarrow x') = \min\left(1, \frac{P(x')}{P(x)}\frac{g(x' \rightarrow x)}{g(x \rightarrow x')}\right)$$

i.e., we always accept when the acceptance is bigger than 1, and we reject accordingly when the acceptance is smaller than 1. This is the required quantity for the algorithm.

The Metropolis–Hastings algorithm thus consists in the following:

1. Initialisation: pick an initial state x at random;
2. randomly pick a state x' according to $g(x \rightarrow x')$;
3. accept the state according to $A(x \rightarrow x')$. If not accepted, transition doesn't take place, and so there is no need to update anything. Else, the system transits to x';
4. go to 2 until T states were generated;
5. save the state x, go to 2.

The saved states are in principle drawn from the distribution $P(x)$, as step 4 ensures they are de-correlated. The value of T must be chosen according to different factors such as the proposal distribution and, formally, it has to be of the order of the autocorrelation time of the Markov process.[8]

It is important to notice that it is not clear, in a general problem, which distribution $g(x \rightarrow x')$ one should use; it is a free parameter of the method which has to be adjusted to the particular problem in hand.

Step-by-step instructions [edit]

Suppose the most recent value sampled is x_t. To follow the Metropolis–Hastings algorithm, we next draw a new proposal state x' with probability density $Q(x' \mid x_t)$, and calculate a value

$$a = a_1 a_2$$

where

$$a_1 = \frac{P(x')}{P(x_t)}$$

is the probability (e.g., Bayesian posterior) ratio between the proposed sample x' and the previous sample x_t, and

$$a_2 = \frac{Q(x_t \mid x')}{Q(x' \mid x_t)}$$

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 according to the following rules.

$$\begin{aligned}&\text{If } a \geq 1: \\ &\qquad\qquad x_{t+1} = x', \\ &\text{else} \\ &\qquad x_{t+1} = \begin{cases} x' & \text{with probability } a \\ x_t & \text{with probability } 1-a. \end{cases}\end{aligned}$$

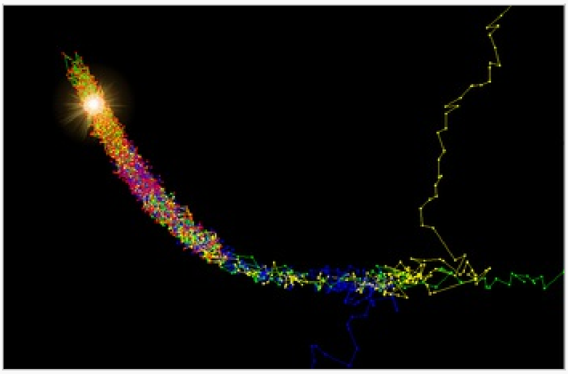
The Markov chain is started from an arbitrary initial value x_0 and the algorithm is run for many iterations until this initial state is "forgotten". These samples, which are discarded, are known as *burn-in*. The remaining set of accepted values of x represent a sample from the distribution $P(x)$.

The algorithm works best if the proposal density matches the shape of the target distribution $P(x)$ from which direct sampling is difficult, that is $Q(x' \mid x_t) \approx P(x')$. If a Gaussian proposal density Q is used the variance parameter σ^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. The desired acceptance rate depends on the target distribution, however it has been shown theoretically that the ideal acceptance rate for a one-dimensional Gaussian distribution is approx 50%, decreasing to approx 23% for an N-dimensional Gaussian target distribution.[9]

If σ^2 is too small the chain will *mix slowly* (i.e., the acceptance rate will be high but successive samples will move around the space slowly and the chain will converge only slowly to $P(x)$). On the other hand, if σ^2 is too large the acceptance rate will be very low because the proposals are likely to land in regions of much lower probability density, so a_1 will be very small and again the chain will converge very slowly.

See also [edit]

- Simulated annealing
- Detailed balance
- Multiple-try Metropolis
- Metropolis light transport
- Gibbs sampling
- Parallel tempering
- Sequential Monte Carlo
- Genetic algorithms
- Mean field particle methods


The result of three Markov chains running on the 3D Rosenbrock function using the Metropolis-Hastings algorithm. The algorithm samples from regions where the posterior probability is high and the chains begin to mix in these regions. The approximate position of the maximum has been

References [edit]

1. ^ Metropolis, N.; Rosenbluth, A.W.; Rosenbluth,

illuminated. Note that the red points are the ones that remain after the burn-in process. The earlier ones have been discarded.

M.N.; Teller, A.H.; Teller, E. (1953). "Equations of State Calculations by Fast Computing Machines". *Journal of Chemical Physics* **21** (6): 1087–1092. Bibcode:1953JChPh..21.1087M. doi:10.1063/1.1699114.

2. ^ Rosenthal, Jeffrey (March 2004). "W.K. Hastings, Statistician and Developer of the Metropolis-Hastings Algorithm". Retrieved 2009-06-02.
3. ^ Hastings, W.K. (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". *Biometrika* **57** (1): 97–109. doi:10.1093/biomet/57.1.97. JSTOR 2334940. Zbl 0219.65008.
4. ^ Teller, Edward. *Memoirs: A Twentieth-Century Journey in Science and Politics*. Perseus Publishing, 2001, p. 328
5. ^ Rosenbluth, Marshall. "Oral History Transcript". American Institute of Physics
6. ^ J.E. Gubernatis (2005). "Marshall Rosenbluth and the Metropolis Algorithm". *Physics of Plasmas* **12** (5): 057303. Bibcode:2005PhPl...12e7303G. doi:10.1063/1.1887186.
7. ^ *a* *b* Robert, Christian; Casella, George (2004). *Monte Carlo Statistical Methods*. Springer. ISBN 0387212396.
8. ^ Newman, M. E. J.; Barkema, G. T. (1999). *Monte Carlo Methods in Statistical Physics*. USA: Oxford University Press. ISBN 0198517971.
9. ^ Roberts, G.O.; Gelman, A.; Gilks, W.R. (1997). "Weak convergence and optimal scaling of random walk Metropolis algorithms". *Ann. Appl. Probab.* **7** (1): 110–120. doi:10.1214/aoap/1034625254.

Further reading [edit]

- Bernd A. Berg. *Markov Chain Monte Carlo Simulations and Their Statistical Analysis*. Singapore, World Scientific, 2004.
- Siddhartha Chib and Edward Greenberg: "Understanding the Metropolis–Hastings Algorithm". *American Statistician*, 49(4), 327–335, 1995
- David D. L. Minh and Do Le Minh. "Understanding the Hastings Algorithm." Communications in Statistics - Simulation and Computation, 44:2 332-349, 2015
- Bolstad, William M. (2010) *Understanding Computational Bayesian Statistics*, John Wiley & Sons ISBN 0-470-04609-0

External links [edit]

- Metropolis-Hastings algorithm on xβ
- Matlab implementation of Random-Walk Metropolis
- R implementation of Random-Walk Metropolis

Categories: Monte Carlo methods | Markov chain Monte Carlo | Statistical algorithms